Journal of Data Intelligence, Vol. 3, No. 4 (2022) 460–473

HYBRID METADATA CLASSIFICATION IN LARGE-SCALE STRUCTURED DATASETS

SOPHIE PAVIA, NICK PIRAINO
BigLab!, Department of Computer Science, Florida State University
Tallahassee, FL, 32306, USA
srp18@my.fsu.edu, np17c@my.fsu.edu

KAZI ISLAM
Department of Computer Science, Florida State University
Tallahassee, FL, 32306, USA
km20fr@cs.fsu.edu

ANNA PYAYT
Department of Medical Engineering, University of South Florida
Tampa, FL, 33620, USA
pyayt@usf.edu

MICHAEL GUBANOV
BigLab!, Department of Computer Science, Florida State University
Tallahassee, FL, 32306, USA
gubanov@cs.fsu.edu

Metadata location and classification is an important problem for large-scale structured datasets. For example, Web tables [29] have hundreds of millions of tables, but often have missing or incorrect labels for rows (or columns) with attribute names. Such errors [24] significantly complicate all data management tasks such as *query processing, data integration, indexing*, etc. Different sources or authors position metadata rows/columns differently inside a table, which makes its reliable identification challenging. In this work we describe our scalable, hybrid two-layer Deep- and Machine-learning based ensemble, combining Long Short Term Memory (LSTM) and Naive Bayes Classifier to accurately identify Metadata-containing rows or columns in a table. We have performed an extensive evaluation on several datasets, including an ultra large-scale dataset containing more than 15 million tables coming from more than 26 thousands of sources to justify scalability and resistance to variety, stemming from a large number of sources. We observed superiority of this two-layer ensemble, compared to the recent previous approaches and report an impressive 95.73% accuracy at scale with our ensemble model using regular LSTM.

Keywords: Hierarchical metadata Metadata classification Web tables

1. Introduction

Large-scale structured datasets such as WDC [29], CORD-19 [34], Census Buerau[1] exhibit a wealth of useful structured data usually originating from hundreds to millions of different sources. Each source represents even the tables of the same category (e.g. Songs, COVID vaccines side-effects) differently, hence efficiently accessing or deriving insights from such heterogeneous and large-scale data is extremely complicated. Not only that, but also some

sources use only *relational* tables, other sources may use tables of different *non-relational* formats. For example, it is very common that information about a particular product is stored differently by different Websites with different attributes and formatting. The problem becomes worse when the table attributes are *hierarchical* (i.e. several attributes nested inside another one).

Because of the table format and metadata variety, efficient metadata annotation and identification is still a subject of ongoing research [28, 24, 9]. Since tabular data can be stored not only in a relational database, but also in CSV format, as a spreadsheet, etc, there have been related research focused on CSV structure detection [28], [12] and Web table metadata annotation [16]. Although these systems showed promising performance, the evaluation sets used in their experiments have relatively small number of sources, hence are very *homogeneous* (i.e. do not exhibit high variety of tabular and metadata representations by contrast to heterogeneous datasets composed from many sources that we used to evaluate our approach). Each source has a liberty to choose the table and metadata format, hence an algorithm, which works good for one source, usually performs much worse for tables from another source unless the formats are significantly similar. To prove that the approach is robust for diverse sources, the evaluation sets should be composed from as many sources as possible.

Here, we describe and evaluate an ensemble of a Deep- and Machine-Learning model to classify metadata rows/columns in a table. To gauge generality, we have evaluated it on four large-scale datasets - CORD-19 [34] and Web Data Commons (WDC) [29], Lehmberg [19]. WDC has more than 15 million tables in English respectively and hundred thousands of different sources, storing data and forming tables in different ways. We have designed an ensemble, combining either *regular* Long Short Term Memory (LSTM) or *Bi-directional* LSTM [25] Recurrent Neural Network (RNN) [26] with a *keras-embedding* layer and a Naive Bayes Classifier. Most of the previous approaches are limited to only on table cell-level analysis, whereas we take into account the cell context (i.e. the whole tuple or a column) along with the position of the cell and the surroundings of that cell in the table. The first-layer model is *order-sensitive* as order matters for the terms inside a cell (i.e. *First Name* is different from *Name First*). However, the second-layer model is order-insensitive as the order of cells does not matter in a tuple. For example, a tuple having *artist* and then *album* value is the same as first *album* then *artist*. Finally, we designed an algorithm that using our hybrid metadata classification ensemble can distinguish different kinds of metadata in a table - row/column-based or hierarchical. To summarize, the main contributions of this paper are the following:

1. A novel two-layer ensemble comprised of a regular LSTM or Bi-directional LSTM with a *keras-embedding* layer and Naive Bayes models for metadata detection. On the first layer, the RNN model performs analysis of an input cell and encodes the context. In the second step, the feature vectors composed of encodings of all cells in a tuple or a column are classified with the Naive Bayes classifier combined with the decision-tree. The output is a binary label indicating whether a a tuple/column contains metadata or just regular data as well as whether it is of hierarchical or plain type.

2. A Web-scale training and evaluation infrastructure that we architected and which is essential for experiments with Web-scale datasets.

3. Extensive evaluation on several Web-scale datasets with tables coming from thousands

to millions of different sources from a a wide variety of domains.

The rest of the paper is structured as follows. First we define the terminology that we used throughout the paper. Then we describe the methodology, followed by the metadata classification ensemble description. After it we explain the large-scale evaluation architecture and experimental study on large-scale datasets with comparative evaluation against the previous approaches. We finish with the related work discussion and conclusion.

2. Definitions

Relational tables, defined in [13], have the following properties: values are atomic, each column has values of the same type, each column has unique name. An example of a relational table is illustrated in 1(a).
Def_1: Cell is a data value of a non-composite type (i.e. can be a number, string, etc) found at the intersection of a row and a column in a table. A relational table has C*R cells total, where C is a number of columns and R is a number of rows.
Def_2: Metadata is a sequence of the *attribute* names of a table, found in a row or a column. Metadata can be represented as a row - Figure 1(a), or a column - Figure 1(c)
Def_3: We call non-relational here the tables that have *hierarchical/nested* metadata. For example, a metadata row or column may have nested metadata, such as in Figure 1(b) - "Name" column is divided into two columns having "First name" and "Last name" separately.

(a)

Job	Income
Microsoft Developer	$200,000
Google Developer	$180,000

(b)

Job		Income
Company	Title	
Microsoft	Developer	$200,000
Google	Developer	$180,000

(c)

Job	Google Developer
Income	$180,000

Table 1. Vertical and horizontal alignment of Metadata (a,c); hierarchical metadata (b). Metadata rows and columns are colored blue.

2.1. *Non-relational Table Representation and Conversion*

In Figure 1, three different types of tables that our ensenble is working with are shown - two relational tables (a,c) and a non-relational table having hierarchical Metadata (b). In Figure 1 we illustrate how we *convert* non-relational tables with hierarchical metadata into the relational format by adding a non-breaking space in the corresponding cells to bring it to a *valid relational table* format. We converted both the WDC and CORD-19 non-relational tables like that to the relational format used by the ensemble.

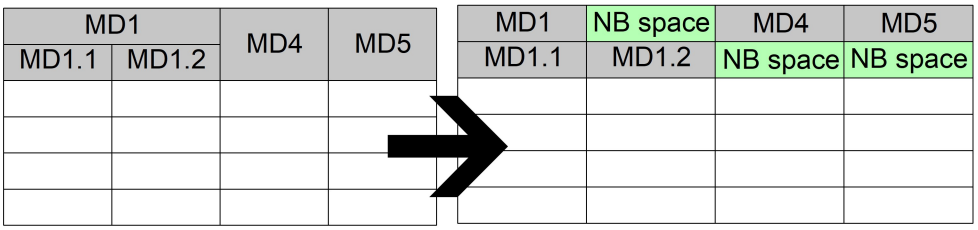

Fig. 1. Tables with hierarchical metadata and their structural representation. MD denotes Metadata.

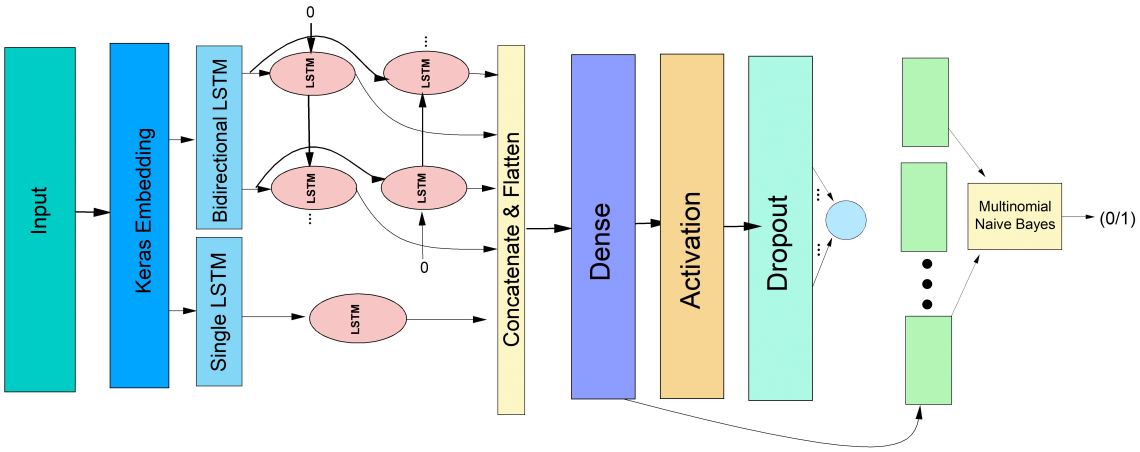

Fig. 2. Our Ensemble Architecture. The LSTM/Bi-LSTM model is on the first layer, followed by the Naive Bayes model on the second layer. The green boxes before the Naive Bayes are the output feature vectors of the Dense layer that represent cell encodings.

3. Methodology

The model architecture is depicted in Figure 2. We use an entire table tuple or column as the input of our model and the model predicts whether it contains Metadata or not. We do not check each tuple or column, because to the best of our knowledge metadata is never stored in the middle, at the very bottom or as a rightmost table column, but if the dataset specifics requires that, our architecture does not prevent that. Hence, we take the first table tuple, the first column and the second tuple as input for the model and the model processes them. The second tuple is taken as input, because for non-relational tables with *hierarchical* metadata, the second row contains the second layer of the Metadata as depicted in Figure 1(b). Finally, to post-filter false-positives, we have designed a custom decision-tree model, based on the number of spaces in the column or row.

We take two table rows and one column as input and stores the predicted value for each input. Then it generates the final decision by using the decision-tree, based on the spaces. As any model, it has false-positives, which we mitigate with this space-based logic. For example, if both the first and second rows of a table are classified as Metadata rows, but there are no blank cells in either row, then the second row cannot be the second row of a hierarchical Metadata. If the first and second rows has the same number of cells without any spaces and

it can not be hierarchical metadata of a non-relational table.

3.1. *Two-layer Ensemble Architecture*

In this section, we describe the two-layer ensemble we have designed for tabular Metadata classification and the ideas behind its design. The first layer consists of a Recurrent Neural Network (RNN) [26] model containing either regular LSTM or Bidirectional LSTM [25] units for the analysis of a table cell. The number of rows and columns are different for each table and also the number of terms inside each cell is different. Keeping this in mind, we have designed our RNN model to process one cell of a tuple or column at a time. Each cell may contain different number of terms as its value and the order matters in this case. For example, "Last Name" and "Name Last" are different values both syntactically and semantically. Hence, we decided to use the LSTM-based model, which is *order-sensitive*, to predict the probability of a cell having Metadata. Although, it is understood that many common terms, some of them referring to *data types* are used in the Metadata cells, such as "Time", "Date", "Name", etc., a purely rule-based classification would not be accurate enough at scale given enormous variety of sources in large-scale datasets, all having different naming conventions, even for the data types. Moreover, the whole Metadata cells as components forming a tuple or a column are *order-insensitive*. For example, suppose there are two columns in a table - "Name" and "Age". It does not matter if we swap "Name" with "Age" in a tuple, the tuple still remains the same, retaining its semantics. Hence, the second layer that we added (Naive Bayes Classifier [4]) is order-insensitive. After feeding the cell to the RNN model, we layer our model including Dense, Activation, and a drop out layer. We have taken the output of the intermediate dense layer as an encoding of the cell and for a whole row or column, we concatenated all such encodings to form the feature vector, input to the Naive Bayes Classifier. Figure 2 illustrates the ensemble architecture.

3.1.1. *Feature space*

We have used 100,000 dimensional feature space, i.e. 100K English terms in our vocabulary that we have selected by taking all terms from our datasets, sorting by frequency and cutting off the noise words and spam [33]. Increasing the dimensionality further led to significantly slower training time, which would prevent or make the experiments much more difficult (see the Section below for the configuration of our cluster).

3.1.2. *Feature vectors*

First the term sequence in a cell is converted into a feature vector, encoded by one-hot encoding [6]. The maximum number of terms in a cell is 734 for our datasets. Each cell is converted to a collection of 734 vectors, each vector corresponding to a term using zero-padding. We did not use any pre-trained word embedding, rather we have trained our a keras- embedding layer using our vocabulary and datasets. The dimensionality of our keras-embedding layer is 100, which helps reduce the dimensionality of input vector space (100K) significantly and alleviates sparsity. The vectors output of the embedding layer are passed to the input of the LSTM/Bi-LSTM units. We have used 64 LSTM/Bi-LSTM units in this case followed by a fully connected layer, containing 256 dense nodes. Then we have added

a *dropout* layer, usually used to avoid overfitting and finally, the output layer has one dense unit with the sigmoid function as the activation function, which outputs the probability of the input cell is a Metadata cell, however it does not generate the final decision for an entire tuple or column.

To generate the final label, the first layer, first, processes all cells of a tuple or column sequentially, before the second layer starts processing and producing the final decision. Although the individual cells contain sequences of terms that are order-sensitive as discussed, the order of each cell within the row or column does not matter. So we need an order-insensitive model to process an order-insensitive sequence of cells. After training the RNN model, for each cell, we extract the output of the intermediate dense layer that contains 256 dense units. This layer produces an encoding for each input cell. We have taken each cell's encodings, concatenated them and used it to form the input feature vector for the Naive Bayes Classifier. For example, if there are 4 cells in a row, each cell will have a feature vector of 256 dimensions that of the intermediate dense layer. For a tuple, concatenating the individual cell's feature vector, we'll have a vector of 4*256 = 1024 dimensions as the input feature vector for the Naive Bayes Classifier. The final output is binary, whether the input sequence of cells, i.e. a tuple or column is a Metadata row/column or not. The output of the Naive Bayes Classifier is then used with the decision-tree rules to produce the final output having the Metadata type.

3.2. *Large-scale Evaluation Architecture*

In this work, we have used two different datasets. One is named CORD-19[34] and the other is Web Data Commons [29], a large-scale corpus having Web Tables from different sources on different topics. The CORD-19 dataset is a novel collection of papers related to COVID-19. Tables used in the papers are stored in a JSON file having HTML format and they have used the structured representation shown in Figure 1. We have extracted the HTML formatted tables to JSON. Given mostly medical tables in CORD-19, there are tables of all three formats, metadata on left, metadata on top and hierarchical metadata. Most tables having hierarchical metadata have metadata on top. From the latest CORD-19 snapshot papers, we were able to extract ≈256K tables.

Another large-scale dataset that we have used in this work is WDC [29]. This dataset consists of more than 100M Web tables, including information about their source URL, table type, text before and after the table, where more than 15M tables are in English language. The tables are from a variety of domains, including scientific, news articles, product information and many other. This dataset is a very robust, large-scale dataset with significant representation of tables from many domains, so it is very attractive choice for evaluating our models, scalability, and generalizability across domains and sources. Regarding the metadata position, we have found six types of tables in total: relational, metadata on the left, metadata on the top, hierarchical metadata on the left, hierarchical metadata on the top. In total there are more than 100 million tables from ≈265K different sources.

4. Experimental Study

In this section, we first describe the training and test sets construction followed by the experimental evaluation on several large-scale datasets.

Hardware: We run all our experiments on Amazon AWS EC2 p3.8xlarge instances, each

having 4 NVIDIA® V100 Tensor Core GPUs, 32 vCPUs, 244 GiB of memory, 10 Gbps network.

Software: For implementing the RNN model, we have used Keras [3], a popular python library for deep learning, having Tensorflow [8] framework as the backend. The second step i.e. the Naive Bayes classifier is implemented using Scikit-learn [7], a popular machine learning library for python. Throughout the experimentation and implementation, we have used python as the programming language.

4.1. *Training the Models*

Our ensemble has 2 layers that we trained separately. On our large-scale datasets, we are able to achieve at best 93.6 percent accuracy for classifying Metadata with on top with a Bidirectional LSTM layer, 95.73 percent for Metadata on the left with a regular LSTM layer, and 93 percent for hierarchical Metadata on top with a Bidirectional LSTM layer. These are the best results in class, to the best of our knowledge, given the scale, a huge number of sources, and a variety of domains represented in the datasets.

Main parameter settings: In the first layer, for both LSTM and Bi-LSTM models, we have used 64 LSTM/Bi-LSTM units and an embedding layer where the keras embedding dimensionality of 100. LSTM units are either regular or bi-directional and order sensitive. The input feature vector is composed of a collection of terms in a cell and it is first encoded using *one-hot* encoding and then is passed through the embedding layer that reduces the dimensionality to 100. After the LSTM/Bi-LSTM units, we have added a fully connected layer with 256 nodes. We have used rectified linear activation function for all nodes. We are aware that such models tend to overfit even on a large training set. To alleviate that we have used a dropout layer after the activation layer with a value of 0.2, known to be a balance between dropping too many features and degree of overfitting. There is a node having *sigmoid* as the activation function. Before the activation and dropout layers we have used a Dense layer that's output serves as input for the Multinomial Naive Bayes Classifier.

First we have trained the LSTM model as a binary classifier i.e. to classify a cell being a Metadata cell or not. We have taken the output feature vector of the dense layer having 256 nodes as an encoding of the input cell, i.e. the dimensionality of an encoded vector is 256. For a tuple or column, there are several cells. We have concatenated their encodings to form an encoding for a tuple/column. This feature vector is the input of the Multinomial Naive Bayes Classifier, which is also a binary classifier that predicts the input tuple or column being Metadata or not.

Dimensionality: Each table has a different number of columns and rows, hence a different number of cells. Each cell has a different number of terms, some of them can be blank, e.g. with hierarchical Metadata or just missing values. The largest cell had 764 terms among the datasets we have used. So after tokenization, we had to *pad* each cell with zeros to align all cells to 734 dimensionality. As discussed above, the input we have used in the Naive Bayes model is the concatenated encodings of all cells in the tuple or column. The widest table has 36 cells in one row. Other vectors, having less cells were amended with zero matrices.

4.2. *Training Sets*

First, we have selected only English tables from the WDC dataset as the dataset is multilin-

gual and we wanted to experiment first on the English subset, which is more than 15 million tables. From our experiments, we have observed that one source maintains a common format for most of its tables. So if we compose the training set from one source it will be biased to that source. Hence, we have uniformly sampled at random an equal number of tables from all sources and constructed the training set containing 570K tables. We used space-based logic to find the Metadata rows/columns to form the training set. Per the representation illustrated in Figure 1, tables having hierarchical metadata have spaces inside the Metadata row. However, this is not a 100% guarantee, sometimes there is a blank space purposefully in the leftmost column. So we first pre-selected all tables having more than one space in the top row. Second, we pre-selected the well-formed relational tables that should not have blank spaces neither in the first row *nor* the column. After that, we hired two independent annotators who took uniform samples of sufficient size from these subsets and checked the table type and Metadata location manually to ensure there are 500 correct positive labels for each kind of metadata. We amended the positively labeled training instances with the same number of negative (regular data rows and columns, without Metadata) instances by uniformly sampling tables from the entire dataset and taking the second last row from each table in the sample. We have asked two independent annotators to manually check and discard any mistakes, but there were none on this step. To ensure the training set is balanced, we made sure there is equal number of positively and negatively labeled instances.

4.3. *Test Sets*

We have constructed three separate test sets - for Metadata in the first row, in the first column, and hierarchical Metadata. To ensure heterogeneity of our test sets, we have first uniformly sampled the tables from the entire dataset, excluding the tables used for the training set. In total, we had 6,785 different sources for the three test sets. Then we hired two independent annotators to annotate 500 tables for each test set and the same number without metadata to ensure a balanced test set. We discarded the label, whenever annotators disagreed.

5. Experimental Evaluation

Here we evaluate the accuracy of our trained ensemble to recognize Metadata on two large-scale tabular corpora - WDC [29] and CORD-19 [34] as well as four other popular corpora used in the recent related work [12, 16, 11, 28]. We have used *accuracy* as defined in Equation 1 below as the metric. *TP, TN, FP, FN* denote the true positives, negatives, false positives, and negatives respectively as explained in more detail in [2].

$$Accuracy = \frac{\text{TP+ TN}}{\text{TP + TN + FP + FN}} \tag{1}$$

Table 2 illustrates Metadata classification accuracy for three Metadata types evaluated on WDC [29] and CORD-19 [34] datasets. The accuracy varies, depending on the dataset size and number of sources. The CORD-19 dataset contains only medical tables extracted from scientific publications on COVID-19. Although the scientific papers are from different sources, there might be an inherent similarity of formats common among tables in this domain, hence the accuracy is slightly better than that for WDC [29], which is also much larger and has hundreds of thousands of sources.

Dataset	Metadata Type	Accuracy
Cord-19	Metadata in the first row	83.76%
	Metadata in the leftmost first column	82.9%
	Hierarchical Metadata	84.35%
WDC	Metadata in the first row	79.87%
	Metadata in the leftmost first column	76.47%
	Hierarchical Metadata	86.4%

Table 2. Classification Accuracy for Three Different Metadata Types on Two Large-scale Corpora.

In recent related work [12, 16, 11, 28], the authors evaluated their Metadata classification models on much smaller datasets, composed from much fewer sources. By contrast, CORD-19 and especially WDC are *ultra large-scale heterogeneous* datasets and we would like to highlight high accuracy and generalizability of our approach in such challenging context. Purely cell-based approaches that consider and classify just a single table cell in isolation [12, 16, 11, 28] without treating them sequentially as a row or a column, unlike our approach, do not scale and generalize very well across domains, datasets, and sources. We implemented a naive standard cell-level classification including Random Forest classifier as in [28] and evaluated it on CORD-19 dataset. Its accuracy was 65% (for Metadata in the first row) compared to our 83.7 %, which yields an impressive ≈19% delta in accuracy. For this evaluation we used out regular LSTM model.

We picked 3 largest and most heterogeneous datasets, used by the authors in [12, 16, 11, 28] and other recent related work on location and classification of the components of verbose CSV files, such as Metadata, Header, Group, Data, Notes, etc. to comparatively evaluate our approach. Our work is most similar to the header detection in these files (i.e. Metadata in our terminology). Other two datasets were not available for public access as well as are manually crawled and post-processed, which makes them more customized and less attractive for comparison. Table 3 compares these and our datasets used for comparative evaluation by size and the number of sources.

Our approach performs approximately the same on all datasets from these recent works with a slight delta in F-measure of a few percent. I.e. our F-1 score is higher on DeEx (93.6% vs. 80.7%) with our Bi-LSTM and slightly lower on SUAS datasets (95.7% vs. 96%) with our regular LSTM. The heterogeneity of our training and test sets is much higher, because they are composed from a large number of sources, which significantly affects generalizability of our trained ensemble as justified by the comparative evaluation against a regular cell-based approach on our large-scale datasets (≈ 19% delta in accuracy, see this Section above for the detailed description). I.e. instead of using many tables from one or several large sources for training and evaluation, we have uniformly at random taken tables from a wide variety of sources both for training and evaluation purposes to ensure the trained model is more robust and naturally resistant to heterogeneity as well as making the evaluation set much more challenging and realistic at scale.

Except many *fundamental* data management activities, naturally dependent on metadata such as *query processing, data integration, warehousing, replication* and many, another important application of Metadata location and classification is distinguishing *relational* and *non-relational* tables. In our datasets, we had both kinds of tables and we have evaluated

Related work	Dataset Size	Number of Sources
Christodoulakis *et. al.* [12]	4500	2
Fang *et. al.* [16]	255	2
Zhe *et. al.* [11]	1.1 Million	Not mentioned
Lan *et. al.* [28]	1345	5
Our Datasets [29, 34]	**More than 15 Million**	**More than 26512**

Table 3. Large-scale datasets that were used for evaluation. For "Our Datasets", we count only the subsets of WDC and CORD-19 in English.

Dataset	Algorithm	MD on Top	MD on Left	Hierarchical MD
Cord-19 [34]	w/o NB	77.97%	80.75%	83.43%
	with NB	**83.7%**	**82.9%**	**84.3%**
Lehmberg *et. al.* [29]	w/o NB	67.3%	69.8%	74.3%
	with NB	**78.8%**	**76.4%**	**81.5%**
DeEx [19]	w/o NB	83.7%	91.8%	81.9%
	with NB	**93.6%***	**95.73%***	**84.2%**
SAUS [19]	w/o NB	**78.5%**	89.3%	91.4%
	with NB	74.9%	**95.7%**	**93%***

Table 4. Accuracy of our hybrid ensemble, evaluated on the datasets from the recent works. Here MD abbreviates Metadata; "w/o NB" means the results are calculated by using only the first-layer LSTM/Bi-LSTM model; "with NB" means the ensemble is two-layer as usual with NB as a second layer. The best results per dataset are in bold, * indicates best in category

performance of our ensemble on this task as well. Although the number of sources is ultra large, it performs remarkably well on WDC. We were more interested in *Recall*, because of the large number of sources and our goal to evaluate *generalizability* of our trained ensemble in context of many sources. *Precision* was ≈75% for both datasets with regular LSTM, but we thought *Recall* is more interesting as in context of thousands of sources it characterizes the ability to recognize new representations and resistance to heterogeneity at scale, which is a big challenge in practice [27, 23, 22, 21]. For classifying relational tables we observed Recall - 79.9% on WDC and 94.6% on CORD-19 and for non-relational tables on WDC - 76.5% and 96.7% on CORD-19 . We would like to highlight stellar Recall on CORD-19 and a lower, but still remarkable Recall on WDC due to the vast scale and unprecedented heterogeneity of this unique Web-scale dataset.

We finish by discussing Table 4, which illustrates the comparative evaluation results of our ensemble for the same classification task of three different kinds of Metadata - Metadata on the Top, Metadata on the Left and Hierarchical Metadata. This experiment is performed on 4 different datasets and the accuracy of locating these metadata types is presented.

From these last evaluation results, we can infer that using the order-insensitive Naive Bayes classifier as a second layer generally improves the Metadata classification performance compared to just using the Deep Learning regular LSTM or Bi-LSTM model alone. In most cases the ensemble performs better, because analyzing only the cell content, which is done in the first layer is insufficient. Analyzing the cell composition in a tuple or a column in an order-insensitive way improves performance. In only two cases for the SAUS metadata on top, just the first layer performs better, because in these case the Metadata rows have many numeric values and Naive Bayes is classically term-based [5, 4].

6. Related Work

Generalizable Metadata location and classification in Web Tables, CSV files, tables, extracted from scientific publications, other large-scale structured corpora is starting gaining momentum in the Data Science community [12, 16, 11, 28]. It is due to both fundamental nature of Metadata and it being of critical importance for many data management tasks on structured data and the fact that large-scale structured datasets are becoming ubiquitous, lack accurate Metadata labeling, and are, at the same time invaluable assets full of useful information.

Here, we presented a hybrid, two-layer Machine- and Deep-Learning ensemble for location and classification of Metadata rows, columns as well as more complex hierarchical Metadata in tables. Several recent works study discriminating between relational and non-relational tables, which is related to metadata classification [10], [35]. Except being capable to do the same and at scale, our approach is also useful for precise Metadata rows or columns location and Metadata type identification. In [12], the authors proposed Pytheas, a line classification system for CSV files. Using fuzzy logic, it determines whether a field is data or not and based on the rules, it detects table border and hence it can differentiate table from metadata to some extent. Their algorithm uses two phases for the task, offline (training) phase and online table discovery (inference) phase. In the offline phase, the algorithm learns the weights for the rule set and in the inference phase, using fuzzy logic it computes confidence value for each line whether it belongs to data or not data. They have evaluated their algorithm on two manually annotated datasets containing a total of 4500 CSV files with a recall value of 95.7%. Although the recall is high, the evaluation set size is much smaller and less heterogeneous (composed only from two different sources) than the Web-scale corpora, we used to evaluate our approach. Size and the number of sources used for training and validation sets drastically affect classification performance at scale and in presence of heterogeneity [27, 17].

The authors in [11] proposed rule-assisted active learning for header spreadsheet property detection including differentiating header from data to reduce human annotation. They have used a hybrid iterative learning framework, with and without user-provided rules for learning property detection of a CSV document. At first, a human labeler labels the spreadsheet properties manually and the models are trained based on the human labeling activity. They have worked with two different sources of data, a web-crawled dataset containing 1.1M sheets and Web400 data containing 400 sheets. Authors have shown 89% accuracy for header detection in a CSV file. Again the validation set size is much less compared to what we have used and the number of sources is not specified by the authors. Table 3 compares the datasets including the one used by the authors.

The authors in [16] used Random Forest classifier to detect and classify table headers. They have proposed two heuristic strategies to separate data and the header. As a baseline, they have used the first row and first table columns as default headers. Their evaluation set contains only 255 tables, which is remarkably small compared to all recent works and our datasets. They have achieved 92% accuracy on their test set, which does not unfortunately mean it will remain the same when the validation set becomes larger and more heterogeneous, even slightly.

To finalize the discussion, the authors in a very recent work [28] performed line and cell classification in verbose CSV. In this work, authors have focused on CSV structure detection and for this purpose, they have detected various cell types like metadata, header, group,

data, derived, notes etc. Our work is similar to the part of their work on cell classification, more precisely, the header cell classification. For cell classification, they have used content, contextual and computational features of the cell. That is they have analyzed the number of empty cells, position of row or column, is there any empty column besides the column being analyzed, block size, data type etc. This analysis is, however, is purely cell-based and does not take into account compositional features of cells when they become tuples or columns, which we do. Our ensemble is also two-layered, where the first layer takes into account term order inside cells and on the second layer the cells are order-insensitive. This idea has been proven valuable and sound for structured data in context of set-based relational model and is absent in [28] as well as all other recent works to the best of our knowledge. Lastly, they trained a multi-class random forest classifier and evaluated performance of their system on 5 different datasets. Their evaluation results are good compared to the related works, but the evaluation set is again much smaller than ours. Moreover, the number of sources of their datasets is also very small compared to our Web-scale datasets, such as WDC, composed from hundreds of thousands of sources.

We used a keras-embedding layer in our regular and Bi-LSTM. The authors of [14] use different relational table elements to generate embeddings, and then apply them to different tasks, such as row population, column population and table retrieval. They trained four variants of table embeddings: Table2VecW, Table2VecH, Table2VecE and Table2VecE using words appearing in a page title, caption, table headings, table cells and columns. Their goal is to return a list of entities based on their likelihood of being added to the core column of T. For table retrieval a ranked list of tables are returned based on a keyword query and relevance. In comparison with that approach we train embeddings using only tabular data, such as tuples, column values and column/row headers. Our embedding layer are used for a very different task of classifying Metadata tuples and columns. Another interesting study [32] projects tabular data into two-dimensional embeddings similar to an image, and feeds those images into fine-tuned two-dimensional CNN models for classification. One of the challenges of this approach is that columns of the same category might have a very different order, and a result the features might end up being quasi 1-dimensional, in that case conversion to images might be not needed. One more creative publication [18] describes embeddings for tables that can be used to perform blocking in the context of entity matching. They proposed three approaches for transforming tables to sequences: converted header rows as a sequence of attributes, values of subject column as sequence of entities and generated a sequence of triples of the form ¡entity, header, value¿ combining row values and headers. The authors trained only Word2Vec embeddings on these sequences, obtained the embeddings and used cosine similarity to find similar vectors/sentences. We used a different context encoding approach, tried several types of embeddings and chose a local keras-embeddings layer.

[20] used word embedding based similarity measure between tabular datasets to identify joinable and unionable tabular open data. First, the authors processed the column values to remove punctuation, split CamelCase and hyphenated words, and converted text to lowercase. Then a word embedding model was used to extract two vectors for each column: one for the column name and the second for text in the column. These embedding vectors were used to compute a *cosine* similarity between each columns of two tabular datasets. Based on the similarity scores the authors determined if the tables were *joinable* and *unionable*. For

embedding models they used Word2Vec pre-trained on part of Google News dataset, fastText pre-trained on Wikipedia 2017 and Word2Vec pre-trained on 1.6 million Wikipedia relational tables. Compared to their approach we trained our embedding layer on our training sets and used our model to classify tabular Metadata at scale.

7. Conclusion

Here, we discussed our new two-layer Machine- and Deep-Learning ensemble for location and classification of Metadata rows and columns as well as more complex hierarchical Metadata in large-scale structured datasets. For cell-level analysis in the first layer, we used a Hybrid Recurrent Neural Network (RNN) model composed of both regular and Bi-directional LSTM units. The cell-level analysis is order-sensitive as the cell content - a sequence of terms is order-sensitive. On the second layer, which analyses the composition of several cells into a tuple or a column, we have used the Naive Bayes classifier, which is order-insensitive in accord with order-insensitivity of a tuple or a column regarding the cell order.

Except this architectural novelty, in the recent works, the authors used relatively small detests, composed from a small number of sources to evaluate their approaches. We have evaluated scalability and generalizability our approach on ultra large-scale heterogeneous datasets such as WDC. We have specifically constructed our training and evaluation sets to contain tables from thousands of sources. Tables are represented very differently depending on a source, so the algorithms trained on one source, having high accuracy one or several sources, do not generalize well to thousands of sources [27, 17, 15, 31, 30]. That is why rule-based or regular Machine-learning based approaches would be incapable of inferring additional source-dependent features, unlike Deep-Learning. Our work achieves high accuracy on two large-scale datasets, WDC and CORD-19 that confirms scalability and generalizability in context of a large number of sources.

References

1. Census bureau. `https://www.census.gov/data/datasets.html`.
2. Evaluation of binary classifiers. `https://en.wikipedia.org/wiki/Evaluation\textunderscoreof\textunderscorebinary\textunderscoreclassifiers`.
3. Keras. `https://keras.io/guides/functional\textunderscoreapi/`.
4. Naive bayes. `https://en.wikipedia.org/wiki/Naive_Bayes_classifier`.
5. Naive bayes. `https://en.wikipedia.org/wiki/Naive_Bayes_classifier`.
6. One hot encoding. `https://en.wikipedia.org/wiki/One-hot`.
7. Scikit-learn. `https://scikit-learn.org/stable/modules/naive\textunderscorebayes.html`.
8. Tensorflow. `https://www.tensorflow.org/`.
9. M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Webtables: exploring the power of tables on the web. In *VLDB*, 2008.
10. M. J. Cafarella, A. Halevy, Y. Zhang, D. Wang, and E. Wu. Uncovering the relational web. In *WebDB*, 2008.
11. Z. Chen, S. Dadiomov, R. Wesley, G. Xiao, D. Cory, M. Cafarella, and J. Mackinlay. Spreadsheet property detection with rule-assisted active learning. CIKM '17, page 999–1008, New York, NY, USA, 2017. ACM.
12. C. Christodoulakis, E. B. Munson, M. Gabel, A. D. Brown, and R. J. Miller. Pytheas: Pattern-based table discovery in csv files. *Proc. VLDB Endow.*, 13(12):2075–2089, July 2020.
13. E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, June 1970.
14. L. Deng, S. Zhang, and K. Balog. Table2vec: Neural word and entity embeddings for table

population and retrieval. In *SIGIR*, 2019.
15. X. L. Dong. Challenges and innovations in building a product knowledge graph. In *KDD*, 2018.
16. J. Fang, P. Mitra, Z. Tang, and C. L. Giles. Table header detection and classification. *AAAI*, 26(1), Jul. 2012.
17. A. L. Gentile, P. Ristoski, S. Eckel, D. Ritze, and H. Paulheim. Entity matching on web tables: a table embeddings approach for blocking. In *EDBT*, 2017.
18. P. R. S. E. D. R. H. P. Gentile, Anna Lisa. Entity matching on web tables: a table embeddings approach for blocking. In *EDBT*, 2017.
19. M. Ghasemi Gol, J. Pujara, and P. Szekely. Tabular cell classification using pre-trained cell embeddings. In *ICDM 2019*, pages 230–239, 2019.
20. C. González-Mora, D. Tomás, I. Garrigós, J. J. Zubcoff, and J. N. Mazón. Model-driven development of web apis to access integrated tabular open data. *IEEE Access*, 8:202669–202686, 2020.
21. M. Gubanov. Polyfuse: A large-scale hybrid data fusion system. In *ICDE*, 2017.
22. M. Gubanov, M. Priya, and M. Podkorytov. Cognitivedb: An intelligent navigator for large-scale dark structured data. In *WWW*, 2017.
23. M. N. Gubanov, L. Popa, H. Ho, H. Pirahesh, J.-Y. Chang, and S.-C. Chen. Ibm ufo repository: Object-oriented data integration. *VLDB*, 2009.
24. B. Hancock, H. Lee, and C. Yu. Generating titles for web tables. In *WWW*, New York, NY, USA, 2019. ACM.
25. S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
26. L. C. Jain and L. R. Medsker. *Recurrent Neural Networks: Design and Applications*. CRC Press, Inc., USA, 1st edition, 1999.
27. R. Khan and M. Gubanov. Weblens: Towards interactive large-scale structured data profiling. In *CIKM*. ACM, 2020.
28. F. N. Lan Jiang, Gerardo Vitagliano. Structure detection in verbose csv files. EDBT, March 2021.
29. O. Lehmberg, D. Ritze, R. Meusel, and C. Bizer. A large public corpus of web tables containing time and context metadata. In J. Bourdeau, J. Hendler, R. Nkambou, I. Horrocks, and B. Y. Zhao, editors, *WWW*, 2016.
30. G. Limaye, S. Sarawagi, and S. Chakrabarti. Annotating and searching web tables using entities, types and relationships. 2010.
31. D. Ritze and C. Bizer. Matching web tables to dbpedia - A feature utility study. In *EDBT*, 2017.
32. B. Sun, L. Yang, W. Zhang, M. Lin, P. Dong, C. Young, and J. Dong. Supertml: Two-dimensional word embedding for the precognition on structured tabular data. In *CVPR*, 2019.
33. S. Villasenor, T. Nguyen, A. Kola, S. Soderman, and M. Gubanov. Scalable spam classifier for web tables. In *IEEE Big Data*, 2017.
34. L. L. Wang, K. Lo, Y. Chandrasekhar, R. Reas, J. Yang, D. Eide, K. Funk, R. M. Kinney, Z. Liu, W. Merrill, P. Mooney, D. Murdick, D. Rishi, J. Sheehan, Z. Shen, B. B. S. Stilson, A. D. Wade, K. Wang, C. Wilhelm, B. Xie, D. M. Raymond, D. S. Weld, O. Etzioni, and S. Kohlmeier. Cord-19: The covid-19 open research dataset. *ArXiv*, 2020.
35. Y. Wang and J. Hu. A machine learning based approach for table detection on the web. WWW '02, page 242–250, New York, NY, USA, 2002. ACM.