Scalable Hierarchical Metadata Classification in Heterogeneous Large-scale Datasets

Bhimesh Kandibedala[1], Anna Pyayt[2], Christopher Caballero[1], Michael Gubanov[1]
Florida State University[1], University of South Florida[2]

ABSTRACT

Tabular metadata identification and classification is a *fundamental* problem for large-scale structured corpora, especially for complex tables, rich in *hierarchical* metadata. Millions of scientific tables, Web tables have hierarchical metadata, but most often have none or noisy labels for their metadata rows or columns. Missing or incorrect metadata labels or tags prevent many fundamental downstream tasks such as *query processing, data fusion, indexing, analytics, visualization*, and many other. Different authors position and structure metadata differently inside a table, which makes its accurate identification very challenging.

In this work we describe our scalable binary metadata classification architectures. The first is using Machine-learning with our *novel positional features*. The second one is an ensemble of BiGRU models with parallel layers of embeddings. We have performed an extensive evaluation on large-scale datasets, including WDC and CORD-19 with millions of tables. We observed F-measure of up to 97% with the Δ of up to 6%-18.7% compared to the state of the art. The effect of our *novel positional features* was substantial - up to 55% in F-measure Δ observed when models were trained with them.

1 INTRODUCTION

Large-scale heterogeneous, structured datasets are ubiquitous - WDC [29], CORD-19 [36], Census Buerau[1] are just a few examples. Such corpora exhibit a wealth of useful structured data originating from millions of different sources. Even the tables on the same topic (e.g. Songs, Vaccines side-effects, etc) are represented differently, hence efficiently accessing or deriving insights from them is very complicated [2, 12, 15, 16, 30–33]. Some sources use only *relational* tables, other (e.g. medical, scientific, financial) are dominated by more complex non-relational tables with hierarchical vertical and/or horizontal metadata [14, 17–20].

Because of the differences in format and metadata variety, accurate and scalable metadata identification and annotation is a popular subject of recent ongoing research [5, 8, 10, 21, 28]. Although the recent systems showed promising performance, the evaluation sets used in their experiments have relatively small number of sources, hence are very *homogeneous* - do not exhibit high variety of tabular and metadata representations compared to the large-scale, heterogeneous datasets composed from millions of sources. Each source has a liberty to choose the table and metadata format, hence an algorithm or model, which fits one source, usually performs much worse on another source unless the formats are significantly similar. To justify that the approach is robust for diverse sources, the evaluation sets should be composed from as many sources as possible [3, 12, 13, 15, 16, 30, 32, 33].

Here, we describe and evaluate two scalable learning-based approaches suitable for classification of metadata both in relational and non-relational tables with hierarchical metadata [4, 22–24, 26, 27, 35]. To gauge scalability and generality, we have evaluated them on six different publicly available corpora, two being Web-scale - CORD-19 [36] ($\approx$400K sources, 1.5M tables) and Web Data Commons (WDC) [29] (millions of sources, hundreds of millions of tables). The first is a Machine-learning based approach with *novel positional features* that we designed to take into account the tuple horizontal and vertical contexts, unlike most of the previous approaches that are limited to only cell-level features. Our first model takes the 2D context of the tuple into account along with the position of the tuple and its surroundings in the table. The second approach is a BiGRU ensemble with two different layers of embeddings. BiGRU is order-sensitive as order matters for the terms inside a cell (i.e. *First Name* is different from *Name First*). The following flatten and concatenate and drop-out layers are order-insensitive as the order of attribute values does not matter in a tuple. For example, a tuple having *artist* and then *album* value is the same as vice versa.

Figure 1: An Example of a Relational Table with One Level of Horizontal Metadata.

Finally, we made sure to evaluate our models on different kinds of metadata - both relational and hierarchical prevalent in non-relational medical tables. To summarize, the main contributions of this paper are the following:

(1) A scalable Machine-learning model with novel positional features enabling high accuracy even on Web-scale datasets.
(2) A novel Deep-learning BiGRU-based architecture with parallel dual-layer embeddings especially efficient on large-scale datasets with non-relational tables exhibiting hierarchical metadata.
(3) Extensive evaluation and comparison to the state of the art on six publicly available datasets, including two Web-scale datasets with tables coming from hundreds of thousands to millions of 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 description of both models and experimental study with comparative evaluation
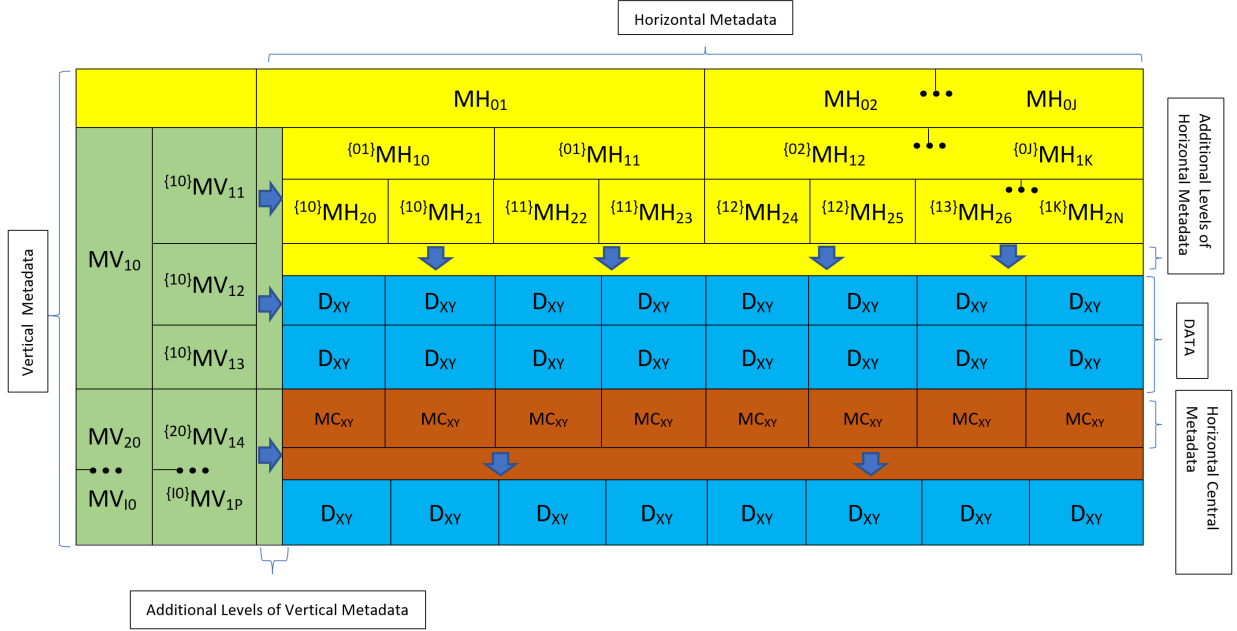
Figure 2: A Table with Multiple Levels of Horizontal, Vertical, and Central Hierarchical Metadata and its Encoding.

against the state of the art. We finish with the related work discussion and conclusion.

2 DEFINITIONS

Relational tables, defined in [9], have the following properties: values are atomic, each column has values of the same type, each column has unique name (i.e. attribute name). The set of all attribute names is called table schema or metadata. An example of a relational table is illustrated in Figure 1.

Def_1: Metadata is a set of *attributes* of a table. Metadata can be stored in a row - e.g. row №1 in Figure 1, or in a column - e.g. column №1 in Figure 2.

Def_2: Cell is a data value (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 number of columns and R number of rows.

Def_3: A table with hierarchical metadata is a table that similar to a relational table has metadata (i.e. attributes), but unlike a relational table it may be found not only in a row, but also in a column. It may also take several rows or columns. Such rows with metadata are called *horizontal metadata* and are denoted as MH_{xy} in Figure 2, where MH is short for "MetadataHorizontal", x and y represent the row/column indices respectively. Horizontal metadata rows are colored yellow in Figure 2. On the other hand, such columns with metadata are called *vertical metadata* and denoted as MV_{xy} in Figure 2, where MV is short for "MetadataVertical", x and y represent the row/column indices. Vertical metadata columns are colored blue in Figure 2. A table may have many horizontal and vertical metadata rows and columns, which is illustrated by arrows in Figure 2. Furthermore, metadata may also exist in the middle of the table, called *central horizontal metadata* and denoted MCH_{xy} in Figure 2, where MC is short for "MetadataCentral", x and y represent the row/column indices. Central horizontal metadata is colored pink in Figure 2. All data cells in the table are denoted as D_{xy}, where D is short for "Data", x and y correspond to the row/column indices. Data rows are colored violet in Figure 2. The table's metadata rows have a relationship with the preceding metadata rows, creating a hierarchical structure which are denoted in the form of superscript

for the element . We also call tables with hierarchical metadata - *non-relational*.

Figure 2 illustrates an example of a table with hierarchical metadata, where the first element of the horizontal metadata is denoted as MH_{00} and the first element of vertical metadata is denoted as MV_{10}. The parent-child relationship between the metadata cells is denoted in the *superscript* for both horizontal metadata MH_{11} and vertical metadata MV_{1y}. For instance, in the horizontal metadata, the cell denoted as MH_{11} has a parent-child relationship with the cells MH_{02} and MH_{03} as indicated by the superscript 02,03 in MV_{11}. Similarly, in the vertical metadata, the cells denoted as MV_{10}, MV_{20}, and MV_{30} have a parent-child relationship with MV_{1Y}, as indicated by its superscript 10,20,30.

3 METHODOLOGY

Now, we describe a our *novel positional features* that can be used with any Machine- or Deep-learning model a new Deep-learning ensemble with parallel embedding layers for high-accuracy Metadata classification in both relational and non-relational tables at scale. Both models can classify apart tabular metadata from data rows and columns. Before, researchers worked mostly on a single cell level, i.e. have analyzed table cells, their position, data value types, and whether the neighboring cells are blank or not [5, 11, 21, 28]. By contrast, we take into account the context for all cells in a tuple or column as well as term-level and cell-level contexts separately in the embedding layers of our custom BiGRU Deep-learning ensemble.

3.1 Positional Features

We construct the feature vector that is passed to the input of a model, by calculating the positional features for each row or column of the table. The feature vector is composed of 13 features $\{f_1, f_2, ..., f_{13}\}$ where f_1 contains the entire tuple/column after the pre-processing step, f_2 indicates to the number of cells in the row/column, f_3 is a Boolean value indicating if there is a row above the current row, f_4 is a Boolean value indicating if there is a row below the current row, f_5 is the number of cells in the row above, f_6 is the number of cells in the row below, f_7 - is a Boolean label indicating if it is a metadata row. It is set for all rows in the

training set and blank for the incoming instances to be classified, f_8 is the № of rows/columns, f_9 - true if the table row or column number equals 2, f_{10} - true if the table row or column number equals 3, f_{11} - true if the table row or column number equals 4, f_{12} - true if the table row or column number equals 5, f_{13} - true if the table row or column number equals 6.

f_1: The content of this feature ensures the statistical dependency between terms appearing as tuple/cell data and metadata and the classification label can be inferred.

f_2: The number of cells in the row, especially compared with the number of cells in the neighbouring rows is often indicative of a row being the metadata row.

f_3-f_6: The numbers of cells in the rows directly below and above the current row and the difference with the current row is often indicative whether the row is data or metadata, especially for non-relational tables. For example, often the number of cells is the same in the top horizontal metadata rows, but it changes immediately when the data rows start.

f_7 to f_{13}: These features affect the probability of a row to be a metadata row. The maximum number of horizontal metadata rows we have encountered in practice in our large-scale datasets is 6. The inclusion of row number as a feature allows the model to assign higher probability to top rows to be metadata rows.

Algorithm 1 illustrates the feature vector construction. The input is a tuple/column, the output is the feature vector having positional features. Each feature carries a certain semantic weight and affects the classification decision. The experimental results comparing models trained with and without such features, justifying their value are illustrated in Tables 1 and 2.

3.2 Machine- and Deep-learning Models

We trained a variety of Machine-learning models and evaluated performance on 5 large-scale datasets. For Machine-learning, we present here the results only for the SVM model with polynomial kernel, since we found out it is superior to all other classifiers that we tried including Logistic Regression, Naive Bayes, Random Forest, AdaBoost, Linear Regression, K-means, KNN, XGBoost, CatBoost. The details on the Training sets composition, training, and evaluation processes are in Section 4.

Algorithm 1: Feature Vector Construction with Positional Features.

---

1 **Input**: a tuple
2 **Output**: a feature vector with positional features.
3 **begin**
4   **F1** = Concatenation of tuple cells
5   **F2** = Number of cells
6   **F3** = 1 if the row above exists, 0 otherwise
7   **F4** = 1 if the row below exists, 0 otherwise
8   **F5** = Number of cells in the row directly above
9   **F6** = Number of cells in the row directly below
10   **F7** = Row number
11   **F8** = 1 if the row number = 2, 0 otherwise
12   **F9** = 1 if the row number = 3, 0 otherwise
13   **F10** = 1 if the row number = 4, 0 otherwise
14   **F11** = 1 if the row number = 5, 0 otherwise
15   **F12** = 1 if the row number > 5, 0 otherwise
16 **end**

The architecture of our new BiGRU Deep-learning ensemble consists of three main stages. In the first stage, a data or metadata tuple/column, $\{x_1, x_2, ..., x_n\}$, where x_i is the ith term from the tuple, is processed to create both cell-wise and term-wise representations. It includes data cleaning along with the replacement of numbers and ranges in data with placeholders such as *NUM, RANGE*. The pre-processed feature vectors are used to to train two kinds of Word2Vec embeddings [34] - cell-level and term-level. At term-level, for each term there is a corresponding embedding vector learned. At cell-level, each cell, even if it has several terms, has one corresponding embedding vector, not several as at term-level. It is because for tabular data - cells are atomic semantic entities, not terms like in Natural Language Processing (NLP), where the standard NLP embeddings were first created and trained at term-level. For tabular data, this difference is important. For example a cell may have an artist name, such as "Britney Spears" and the embedding vector should be trained for the whole cell - "Britney Spears", not just for one term, e.g. "Britney". Similarly, the context unlike for NLP term-level embeddings for tabular data are cells - i.e. "Britney Spears" may be neighbouring with her song title in the next cell - "Born to Make You Happy" and this cell, not just one term from it should contribute to the cell context. This is why we trained cell-level embeddings. At the same time, term-level context inside a cell is also important - i.e. it is remarkable that "Britney" appears next to "Spears", hence we also trained the term-level embeddings and used both types of embeddings in our architecture, capturing both kinds of context. Our architecture has two parallel branches that are initalized with these two kinds of pre-trained embeddings, running both inputs in parallel. This sequence is passed through a BiGRU layer with 200 BiGRU units and the result is concatenated with the original embeddings $\{v_1, v_2, ..., v_n\}$ to create our enriched contextualized vectors, $\{c_1, c_2, ..., c_n\}$. We flatten the output of each path to create both cell- and term-wise input representations. The final stage of the model concatenates the two representations and passes them through a dense layer, a batch normalization layer, and a dropout layer.

4 EXPERIMENTAL STUDY

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

Infrastructure: We run our experiments on a cluster of servers with Intel Xeon CPUs, from 192GB to 1TB of RAM, 10TB disk space each. For implementing the BiGRU model, we have used Tensorflow framework. The SVM model was implemented using Apache Spark MLLib. 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 discarding the noise words and spam [35]. Increasing the dimensionality further led to significantly slower training time, without a noticeable gains in accuracy.

Evaluation Datasets: To evaluate our models and compare to the state of the art we used 5 publicly available datasets, including CORD-19[36] and Web Data Commons [29], both having millions of tables from different sources on different topics. The CORD-19 dataset is a collection of papers related to COVID-19 [36]. Tables used in the papers are extracted from PDF and stored in JSON in HTML format. CORD-19 has mostly medical non-relational tables with hierarchical metadata of all formats - vertical metadata, horizontal metadata, both regular and hierarchical. From the latest CORD-19 snapshot, we extracted ≈1.5 Million tables.

Another large-scale dataset that we have used in this work is WDC [29]. This dataset consists of more than 100M Web tables from 265K sources, including information about their source URL, table type, text before and after the table. The tables are from a variety of domains, including scientific, news articles, product information and many other, so it is very attractive choice for evaluating our models, scalability, and generalizability across domains.

Training the Models: We trained both models on WDC and CORD-19 large-scale datasets. The models achieve F-measure of 97%-98% for 1st level, 93%-98% for the 2nd level metadata classification. These are the best results among all recent solutions to the best of our knowledge.

First, we have selected only English tables from the WDC dataset as the dataset is multilingual and we wanted to experiment first on the English subset, which is still millions of 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.

We amended the positively labeled training instances with the same number of negative instances (i.e. regular data rows and columns, without Metadata) by sampling tables from the entire dataset and taking the second last row from each table in the sample, since we never saw a metadata row in such position in practice. To ensure the training set is balanced, we made sure there is an equal number of positively and negatively labeled instances.

5 EVALUATION

Here we evaluate Precision, Recall, and F-measure of our trained models to classify Metadata in 6 large-scale datasets, including two large-scale corpora - WDC [29] and CORD-19 [36] as well as four other popular structured corpora used in the recent related works for evaluation, such as T2D, T2Dv2, SAUS, and CIUS [7, 8, 10, 28].

Table 1 illustrates Metadata classification accuracy for horizontal and vertical Metadata types. We observe high F-measure for top metadata levels for the Machine-learning model and BiGRU. The Machine-learning model starts degrading on the 2nd level of the metadata hierarchy as well as especially on the vertical metadata. We attribute it to the fact that is not trained on the 2nd level hierarchy of the horizontal metadata - only the 1st level is included in our training sets. Similarly, it is not trained on the vertical metadata. However, we observe the F-measure for BiGRU is slightly higher in these cases, despite the fact it was also not trained on those types of metadata. We attribute it to the fact that the contextual information is leveraged by the BiGRU ensemble.

Table 2 illustrates Precision, Recall, and F-measure of our models trained without including the positional features in the feature vector. We can see a substantial drop in all three measures. Compared to Table 1 we can see that the effect of having our novel positional features in the feature vector is very substantial - up to 55% (97%-22%) in F-measure of the models trained with and without those features.

In recent related work [7, 8, 10, 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.

$DataSet^{Model}$	MDP	MDL	TR	Precision,Recall,F		
$CORD_{19}^{ML}$	Horiz. Top	Lev. 1	315K	97%	98%	97%
$CORD_{19}^{ML}$	Horiz. Top	Lev. 2	60K	96%	89%	91%
$CORD_{19}^{BiGRU}$	Horiz. Top	Lev. 2	60K	94%	93%	93%
$CORD_{19}^{ML}$	Vertical Left	Lev. 1	38K	97%	65%	66%
WDC^{ML}	Horiz. Top	Lev. 1	200k	98%	98%	98%
$T2D^{ML}$	Horiz. Top	Lev. 1	13k	99%	97%	97%
$T2DV_2^{ML}$	Horiz. Top	Lev. 1	13k	99%	97%	97%
SAUS	Horiz. Top	Lev. 1	200K	98%	97%	97%
CIUS	Horiz. Top	Lev. 1	200K	99%	97%	97%

Table 1: Models Performance With Positional Features. MDP abbreviates the Meta data Position, MDL - Meta Data Level, TR - the training set).

DataSet	MDP	MDL	TR	Precision,Recall,F		
$CORD_{19}^{ML}$	Horizontal Top	Level 1	315k	82%	35%	30%
$CORD_{19}^{ML}$	Horizontal Top	Level 2	60k	41%	21%	22%
$CORD_{19}^{ML}$	Vertical Left	level 1	38k	82%	35%	30%

Table 2: Models Performance Without Positional Features. MDP abbreviates the Meta Data Position, MDL - Meta Data Level, TR - training set). We can see the effect of the positional features by comparing to Table 1.

Purely cell-based approaches that consider and classify just a single table cell in isolation [7, 8, 10, 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 concluded that our approach outperforms the latest state of the art solutions on the datasets that were used by the recent works [7, 8, 10, 28] on location and classification of metadata of different complexity. Our work is similar to the header detection in verbose CSV files having complex hierarchical metadata. We picked 2 datasets (SAUS and CIUS), used by the recent studies to comparatively evaluate our approach. 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 1 illustrates performance of our first model on these datasets.

6 RELATED WORK

Accurate, scalable, 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 Management and Science communities [7, 8, 10, 28]. It is due to both fundamental nature of Metadata and it being of critical importance for many downstream data management tasks.

Here, we presented a Machine-learning model with novel positional features and a Deep-Learning ensemble with parallel

two-layer embeddings for location and classification of Metadata rows, columns as well as more complex hierarchical Metadata in structured data at scale. Several recent works study discriminating between relational and non-relational tables, which is related to metadata classification [6], [37]. 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 [8], authors proposed Pytheas, a line classification system for a 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) unlike the Web-scale corpora that we have used to evaluate our approaches. Size and the number of sources used for training and validation sets drastically affect classification performance at scale especially in presence of heterogeneity [11, 25].

The authors in [10] 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 as well as the context, which we do. Our Machine-learning model leverages novel positional features that enable its scalability. These features proved to be very valuable at scale, they are absent in [28] as well as all other recent works on metadata/header identification and classification in (semi-)structured datasets.

7 CONCLUSION

Here, we presented a Machine-learning model with novel positional features and a Deep-Learning ensemble with parallel two-layer embeddings for location and classification of Metadata rows, columns as well as more complex hierarchical Metadata in structured data at scale. Our both architectures achieve high accuracy on six publicly available datasets, including large-scale datasets - WDC [29] and CORD-19 [36].

REFERENCES

[1] Census bureau. https://www.census.gov/data/datasets.html.
[2] B. Alexe, , M. A. Hernandez, H. Ho, J.-W. Huang, Y. Katsis, and L. Popa. Simplifying information integration: Object-based flow-of-mappings framework for integration. In *BIRTE*, 2009.
[3] B. Alexe, M. Gubanov, M. A. Hernández, C. T. H. Ho, J. Huang, Y. Katsis, L. Popa, B. Saha, and I. Stanoi. Simplifying information integration: Object-based flow-of-mappings framework for integration. In *BIRTE*, 2008.
[4] B. Alexe, M. Gubanov, M. A. Hernandez, H. Ho, J.-W. Huang, Y. Katsis, and L. Popa. Simplifying information integration: Object-based flow-of-mappings framework for integration. In *Business Intelligence for the Real Time Enterprise*. Springer, 2009.
[5] 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.
[6] M. J. Cafarella, A. Halevy, Y. Zhang, D. Wang, and E. Wu. Uncovering the relational web. In *WebDB*, 2008.
[7] Z. Chen, S. Dadiomov, R. Wesley, G. Xiao, D. Cory, M. Cafarella, and J. Mackinlay. Spreadsheet property detection with rule-assisted active learning. CIKM. ACM, 2017.
[8] C. Christodoulakis, E. B. Munson, M. Gabel, A. D. Brown, and R. J. Miller. Pytheas: Pattern-based table discovery in csv files. *PVLDB*, July 2020.
[9] E. F. Codd. A relational model of data for large shared data banks. *CACM*, 13(6), June 1970.
[10] J. Fang, P. Mitra, Z. Tang, and C. L. Giles. Table header detection and classification. *AAAI*, 26(1), Jul. 2012.
[11] 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.
[12] M. Gubanov. Hybrid: A large-scale in-memory image analytics system. In *CIDR*, 2017.
[13] M. Gubanov, C. Jermaine, Z. Gao, and S. Luo. Hybrid: A large-scale linear-relational database management system. In *MIT NEDB*, 2016.
[14] M. Gubanov, M. Priya, and M. Podkorytov. Cognitivedb: An intelligent navigator for large-scale dark structured data. In *WWW*, 2017.
[15] M. Gubanov and A. Pyayt. Readfast: High-relevance search-engine for big text. In *ACM CIKM*, 2013.
[16] M. Gubanov and A. Pyayt. Type-aware web search. In *EDBT*, 2014.
[17] M. Gubanov, A. Pyayt, and L. Shapiro. Readfast: Browsing large documents through ufo. In *IRI*, 2011.
[18] M. Gubanov and L. Shapiro. Using unified famous objects (ufo) to automate alzheimer's disease diagnostics. In *BIBM*, 2012.
[19] M. Gubanov, L. Shapiro, and A. Pyayt. Learning unified famous objects (ufo) to bootstrap information integration. In *IRI*, 2011.
[20] M. Gubanov and M. Stonebraker. Large-scale semantic profile extraction. In *EDBT*, 2014.
[21] B. Hancock, H. Lee, and C. Yu. Generating titles for web tables. In *WWW*, New York, NY, USA, 2019. ACM.
[22] R. Khan and M. Gubanov. Nested dolls: Towards unsupervised clustering of web tables. In *IEEE Big Data*, 2018.
[23] R. Khan and M. Gubanov. Towards unsupervised web tables clustering. In *IEEE BigData*, 2018.
[24] R. Khan and M. Gubanov. Weblens: Towards interactive large-scale structured data profiling. In *CIKM*, 2020.
[25] R. Khan and M. Gubanov. Weblens: Towards interactive large-scale structured data profiling. In *CIKM*. ACM, 2020.
[26] R. Khan and M. Gubanov. Weblens: Towards interactive web-scale data integration, training the models. In *IEEE Big Data*, 2020.
[27] A. Kola, H. More, S. Soderman, and M. Gubanov. Generating unified famous objects (ufos) from the classified object tables. In *IEEE Big Data*, 2017.
[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] S. Ortiz, C. Enbatan, M. Podkorytov, D. Soderman, and M. Gubanov. Hybrid.json: High-velocity parallel in-memory polystore json ingest. In *IEEE Bigdata*, 2017.
[31] M. Podkorytov, D. Soderman, and M. N. Gubanov. Hybrid.poly: An interactive large-scale in-memory analytical polystore. In *ICDM Workshops*, pages 43–50. IEEE Computer Society, 2017.
[32] M. Simmons, D. Armstrong, D. Soderman, and M. Gubanov. Hybrid.media: High velocity video ingestion in an in-memory scalable analytical polystore. In *IEEE Bigdata*, 2017.
[33] S. Soderman, A. Kola, M. Podkorytov, M. Geyer, and M. Gubanov. Hybrid.ai: A learning search engine for large-scale structured data. In *WWW*, 2018.
[34] M. T., S. I., C. K., and C. G. D. J. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.
[35] S. Villasenor, T. Nguyen, A. Kola, S. Soderman, and M. Gubanov. Scalable spam classifier for web tables. In *IEEE Big Data*, 2017.
[36] L. L. Wang and K. L. and. The covid-19 open research dataset. *ArXiv*, 2020.
[37] 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.