Learning Circular Tabular Embeddings for Heterogeneous Large-scale Structured Datasets

Michael Gubanov[1], *Anna Pyayt*[2], *Sophie Pavia*[1]
Florida State University[1], *University of South Florida*[2]

ABSTRACT

Deep-learning models, most recently *with* embeddings have been used and yield promising results for many data fusion problems. Some methods directly use the embeddings, pre-trained on large corpora, such as Wikipedia and naively treat table tuples as sentences, when it comes to structured data[1]. This leads to loss of valuable 2D contextual information, inherent to structured data. Other methods *partially* take into account 2D context, but insufficiently to attain high accuracy on downstream tasks, such as classification on Web-scale datasets, known to exhibit extreme *heterogeneity*.

In this paper, we propose scalable algorithms to construct and train embeddings, *fully* embodying complex 2D tabular context, unique to structured data, which turns out to be especially important to attain high accuracy at scale. Compared to the baselines, we report a substantial gain in *F-Measure* of up to 34.4%. We also observed that not only the average downstream classification task accuracy is much higher with our embeddings, but also classification of some categories completely fail with state of the art embeddings.

1 INTRODUCTION

Word embeddings [15, 24, 28] is a popular dimensionality reduction technique producing vectors corresponding to words that capture their context. These vectors have a variety of important downstream applications such as clustering similar terms [10], retrofitting [3], machine translation [8], classification [6], etc. One of the main advantages of using the embedding vectors instead of words is that the embedding vectors, corresponding to words also encode their context, which is valuable. However, standard embedding learning or construction algorithms are built for natural language text, which is inherently one dimensional, i.e. is composed of sentences having "flat" sequences of words.

Here we propose a new scheme to construct embeddings purposed for *structured* data and exploiting its inherent 2D context. We evaluate them on a *fundamental* task in large-scale structured data management, such as table tuple classification [20]. This task plays a key role in downstream applications, such as schema matching [21, 22], data cleaning and discovery [12], data fusion and transformation, all at scale [5, 7, 18, 23, 26, 27, 29].

Table tuple classification is not only *fundamental*, but also *very challenging* task at scale. For example, Table 1 illustrates how different tuple schemas are at scale, even within one category *Job postings*. Without AI models, trained to recognize all such *Job postings*, one would need, first, to somehow find out all their schemas, and then write hundreds of different structured queries, each conforming to a specific schema in order to get access *all* postings. In this paper we make the following *contributions*:

- We *define* our embeddings for relational tables, and describe an algorithm that we call a "circular walk" that allows capturing a rich set of relationships inherent to a table cell and generates a "sentence", embodying its 2D context incorporated into our circular embeddings.
- We *evaluate* our new embeddings by performing extensive experiments on two Web-scale structured datasets, having millions of relational and medical tables from hundreds of thousands of sources and compare performance against the state of the art baselines.

We begin with definitions in Section 2, continue with methodology, Neural Network and Evaluation architectures in Section in Section 3. After it we proceed to describe the training process in Section 4 and our experimental evaluation in Section 5. We finish with discussion of related work in Section 6 and conclude in Section 7.

2 DEFINITIONS

2.1 Large-scale Structured Datasets

Def 1: Let $\mathcal{T}$ be a (very large) set of relational "entity-centric" tables. We define "entity-centric" tables as tables, where each tuple represents some real-world *entity e* of category *E*. For example, *Songs* would be an "entity-centric", relational table, where each tuple stores a specific song.

We assume that table names might be missing or incorrect, which is often the case at scale [20], and that information about the same entity can be scattered among many tables from multiple sources, and that those tables can have different schemas. In that case, for example, a relation of *Songs* category can be identified by a combination of a *song title* and the *singer name* in the relation's Meta-data. We call such combination of attributes - "core attributes", as they can be used to identify entities of a certain category.

Def 2: Let $\{a_{core_1} .. a_{core_n}\}$ be a set of n attributes that is inherent to all entities of a specific category. For example, each *Song* must have {album, artist, and title}, so these 3 attributes would be *core* attributes for all *Songs*.

Notice a remarkable difference with the traditional relational database setting [13]. There, a relational database was meant to store a dataset in its entirety, a table storing one category of entities had one fixed schema. Here, at scale, entities of the same category are very likely to be scattered among multiple relations in different sources, all having different schemas, which is drastically different from the standard expectations in the relational world [13]. In large-scale structured datasets, such as $\mathcal{T}$, especially from the Web, each entity-centric table for entities of the same *category* (i.e. *Companies, People, Songs, Products*, etc) can have different combination of columns with different labels for the same *domain*.

Def. 3. We call a dataset $D_{Ho} \subset T$ - *homogeneous* if it consists of a set of tuples of the same *category*, all having the same fixed schema.

[1] I.e. "TABERT linearizes the structure of tables to be compatible with a Transformer-based BERT model." [33]

Job Postings Attributes
Position Title, Position Number, Location, Position Type, Required Skill Set, Position Description, Department/Team description, Position summary, Position duties, Requirements, Skill & Competency Requirements, Preferences, Benefits
Salary, Job Type, Number of hires for this role, Full job description, Department, Location, Salary range, Status, Reports to, Duties & responsibilities, Related functions, Skills & abilities, Requirements, Minimum education, Competencies, Other, Physical demands and work environment, Benefits, Schedule, Work remotely
Job description, Overview of Global Risk Analytics, Overview of the Role, Position Overview, Required Education, Skills and Experience, Desired Skills and Experience

Table 1: Different set of attributes in *Job postings* tables on the Web.

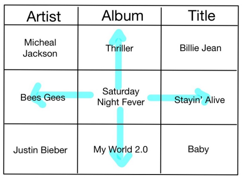
Def. 4. We call a dataset $D_{He} \subset T$ - *heterogeneous* if it consists of a set of tuples of the same *category*, where tuples can have different schemas. The number of columns can vary among tuples, columns, carrying the same semantics can have different labels, columns being order-insensitive.

2.2 Tuple classification

We evaluate our circular embeddings on a downstream task of binary table tuple classification at scale (i.e. if a given tuple belongs to a category). It is central to a variety of structured data management applications, such as data cleaning and fusion [12], transformation [5], schema extraction [16], entity matching [17], and many other [21, 30].

In tuple classification, the goal is to identify *all* tuples (even having different schemas) that belong to a *category E* of entities $e \in \mathcal{T}$. Columns having the same semantic meaning (e.g. *artist, singer*) can have different labels - $M_1..M_n$. The data values stored in these columns can be found in multiple tables. For each column labeled M_j, we label corresponding data items $d_{j1} - d_{ji}$. At the same time, each data item d_{km} might be in multiple columns, labeled with different labels: $(C_1, .., C_m)$.

2.3 Embeddings and Circular Walk



Artist	Album	Title
Micheal Jackson	Thriller	Billie Jean
Bees Gees	Saturday Night Fever	Stayin' Alive
Justin Bieber	My World 2.0	Baby

Saturday Night Fever, Album, Thriller, Album, Stayin' Alive, Title, My World 2.0, Album, Bees Gees, Artist

Figure 1: A Schematic for a "Circular Walk" and the Corresponding Generated Embedding "Sentence".

Our circular tabular embeddings training "sentence" is composed by performing a circle-like walk around a table cell in order to capture not only 1D context like a standard embedding would do, but also the context in a perpendicular dimension. The walk takes into consideration the rows directly above and below each cell. For columns it traverses the width of the table, to the left and right of the cell of interest. During the walk, not only data cells, but also their attribute labels are added. As a result a contextual sentence is formed for a cell from the table. An example of this walk is found in Figure 1. It is important to note that the walk is done in *triples* (three tuples at a time), the walk's width expands with the tables. For a table for more than three columns, each column thereafter is also included in the walk. The contextual cell's metadata is added along with the contextual cell. You can see in Figure 1 the sentence is formed by traversing the row above, following by the column to the right, row below, and finally the column to the left, thus forming a circular pattern. For the cell "Saturday Night Fever", the contextual sentence, *Saturday Night Fever, Album, Thriller, Album, Stayin' Alive, Title, My World 2.0, Album, Bees Gees, Artist*, is formed. The sentences formed are used to train our circular embeddings. Figure 2 demonstrates the walk. The number of rows remains the same for all walks, as walks are performed in triples, however the number of columns pertains to the number of attribute-data pairs present in the table. For example a table with at least three rows and exactly three columns is shown in Figure 2. However, the number of columns could increase, which would result in more horizontal contextual terms being added to the walk.

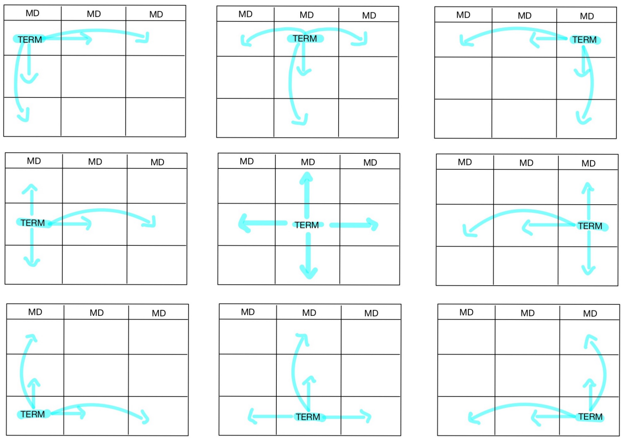


Figure 2: Examples of "Circular Walks" Depending on the Term Position.

3 METHODOLOGY

First we generate the training sets to train our circular embeddings by performing the *circular walks* in a manner defined in Section 2 over each tuple in the training sets constructed for tuple classification (see Section 4.3). The latter are composed from sets of relational tables of a certain category E extracted from $\mathcal{T}$ (i.e. WDC or CORD-19 here) . Then we train a Neural Network (a sequential model with an Embedding layer, Global Average Pooling layer, and two dense layers on this generated training set and extract the trained embedding layer. It has our "circular embedding vectors", which are ready to use in a separately trained model. To have state of the art baselines to compare against, we fune-tuned the embeddings such as Word2Vec [24] and ELMo [28] on the same training sets.

Next, we train several Neural Networks, already having these pre-trained embeddings as an embedding layer, for our validation task - binary tuple classification. We first load the pre-trained embeddings layer, then train the model on the same training sets. In Section 5, we compare performance of the model trained with our circular embeddings used as an embedding layer against the same model, but pre-loaded with the baseline embedding layers and then trained for the same task.

Neural Network Architecture: We used an Artificial Neural Network (ANN) - a feed-forward Multi-Layer Perceptron (MLP) [3], for our models, because of its known top performance on short text documents [2, 3]. The first layer is the Keras Embedding layer [11] and is pre-loaded either with the standard - Word2Vec, ELMo or our circular embeddings; the 2nd layer is Global Average Pooling, followed by 2 dense layers.

Evaluation Architecture: We used two Web-scale structured datasets to train and evaluate our embeddings and models - WDC, having more than 15 Million relational English Web tables coming from more than 248 thousand Web sources [20] and CORD-19 [32], composed of more than 300,000 medical papers on COVID-19, altogether having more than 1.5 Million medical research tables.

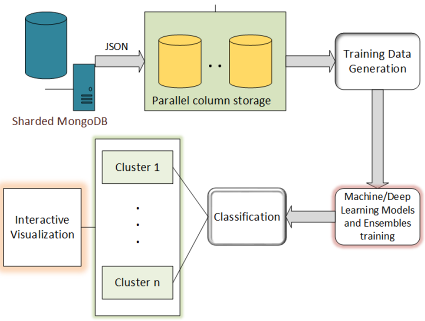


Figure 3: Scalable Experimental Evaluation Architecture.

Our scalable evaluation architecture components are depicted in Figure 3. WDC and CORD-19 datasets are stored in sharded MongoDB - a scalable distributed JSON storage [1]. The JSON data from MongoDB is converted and ingested into a parallel column store [31] for further efficient query processing. Column stores are known to be efficient in processing the queries having *selection* conditions on columns, which turns out to be the most popular workload pattern for sampling and training data generation queries. The *training data generation* box corresponds to *selection* queries, slicing the dataset by source, sets of attributes, and patterns in order to produce the training sets for the next step. The *Machine/Deep Learning Training* and *Classification* boxes correspond to training the Neural Networks with different embedding layers using TensorFlow [4] and further classification of the test sets using these models. The classification step produces *N* categories (e.g. COVID-19 vaccine side-effects, jobs, songs, etc) that can be examined by the user using the "interactive visualization component".

Hardware: For training and validation we used *P3dn.24xlarge* Amazon AWS EC2 instances having 96 Intel Xeon CPUs, 8 NVidia Tesla V100 Tensor Core GPUs, and 768GB or RAM.

4 TRAINING EMBEDDINGS AND MODELS

4.1 Training Circular Embeddings

First we generate the training sets to train our circular embeddings by performing the *circular walks* in a manner defined in Section 2 over each tuple in the training sets constructed for tuple classification (see Section 4.3). The latter are composed from sets of relational tables of a certain category *E* extracted from $\mathcal{T}$ (i.e. WDC or CORD-19 here) . Then we train a Neural Network (a sequential Keras model with an Embedding layer [11]) on this generated training set and extract the trained embedding layer. It has the trained "circular embedding vectors", which are ready to use. To have state of the art baselines to compare against, we also fune-tuned the most popular NLP embeddings such as Word2Vec [24] and ELMo [28] on the same training sets.

Next, we train several Neural Networks, already having these pre-trained embeddings as an embedding layer, for binary tuple classification. We first load the pre-trained embeddings layer, then train the model on the same training sets. In Section 5, we compare performance of the model trained with our circular embeddings used as an embedding layer against the same model, but pre-loaded with the baseline embedding layers.

4.2 Training the Models

We are training two kinds of models, where the *feature vector* includes and excludes tuple attribute names (metadata). We refer to them as "with" and "without" metadata further below. The models trained without metadata are useful for classifying tables with missing metadata. In some cases, we also observed them outperform models trained with metadata, more details will be discussed below.

The *feature vector* for models trained with metadata includes both the attribute names and a sequence of all attribute values from the tuple. The models marked "without metadata" are trained using feature vectors having no attribute names, just a sequence of the attribute values from the tuples. All models are *binary* classifiers, i.e. are trained to recognize only one tuple category (e.g. *Books*).

4.3 Homogeneous Training Sets

An algorithm generating training sets requires only *one sample table* from a known large source to initialize for a category of interest. All further steps are *unsupervised*, i.e. do not need any human intervention. All tables with the same attribute set are extracted from that source and used as a *positively* labeled *homogeneous* training set. For example, for *Books*, ≈362K tuples can be extracted from *Amazon.com* by following this approach. For categories, heavily represented in at least one large source, this method results in a large amount of clean training data with one fixed attribute set. Hence, we call it *homogeneous* training set.

Negatively labeled training data of the same size (to ensure the training set is balanced) is drawn uniformly at random from other sources in WDC or CORD-19, excluding the sources used for positively labeled data. We observed that this method to balance the training set works well for large-scale datasets and used it to create 20 large-scale balanced training sets each having more than ≈4K tuples per set. We created such training sets for 20 different categories, purposely from a *variety of application domains* (i.e. COVID-19 Side-effects, COVID-19 impact on Mental health, COVID-19 impact on fertility, Job postings, Patents, Books, Cars, Cities, and other) present in WDC and CORD-19 to evaluate our models and embeddings and also validate generality of our approach.

4.4 Heterogeneous Test Sets

For evaluation of generalizability of our trained embeddings and models we first drew 20 uniform samples from WDC and CORD-19, one per category, then hired two independent annotators to manually label these *heterogeneous* test sets. The test sets have 500 labeled tables per category - 50% positive, 50% negative

Models	**Circular Embeddings**		**Word2Vec**		**ELMo**	
	With Metadata	W/Out Metadata	With Metadata	W/Out Metadata	With Metadata	W/Out Metadata
Songs	88%, 87%, 88%	66%, 96%, 78%	9%, 7%, 8%	53%, 11%, 18%	80%, 84%, 82%	68%, 77%, 72%
Books	100%, 34%, 51%	78%, 52%, 62%	100%, 33%, 50%	83%, 51%, 63%	61%, 64%, 62%	54%, 57%, 51%
Cities	99%, 95%, 97%	91%, 72%, 81%	96%, 95%, 95%	86%, 60%, 71%	51%, 54%, 52%	53%, 67%, 52%
$Average_{20}$	97.6%, 74%, 80.6%	80.3%, 75.3%, 75.6%	63.3%, 40.6%, 46.3%	72%, 38.6%, 48.6%	70%, 54%, 61%	78%, 47%, 52%

Table 2: Precision/Recall/F-measure of Scalable Tuple Classification Models Trained with Circular, Word2Vec, and ELMo Embeddings on WDC [20] and CORD-19 [32]. $Average_{20}$ **provides an average for 20 different categories from both datasets.**

Models	**Metrics**		**Balanced Pos./Neg.**	**№ of Unique Sources**
	With Metadata	**W/Out Metadata**		
Songs 46k	88%, 87%, 88%	66%, 96%, 78%	yes	58
Songs 46k	53%, 3%, 40%	66%, 26%, 38%	no	62
Songs 67k	100%, 98%, 99%	64%, 8%, 14%	yes	67
Songs 420k	71%, 46%, 56%	72%, 9%, 16%	yes	43
Songs 600k	57%, 51%, 54%	72%, 34%, 46%	yes	52

Table 3: Precision/Recall/F-measure of Scalable *Songs* Tuple Classification Models Trained with Circular Embeddings using Training Sets Composed from Different Number of Sources in WDC [20].

Embeddings	**Time to Train**
Word2Vec: Cities (800k)	125 Minutes
Circular Embeddings: Cities (800k)	23 Hours

Table 4: Time to Train Circular and Baseline Embeddings for WDC *Cities*.

labels, since it is infeasible (and unnecessary) to manually label large-scale datasets such as WDC. Since, labeling is known to be expensive and time consuming, it is understood that it is infeasible to create such test sets for all categories in WDC and CORD-19.

To ensure that the models are not overfitting and to preserve fair evaluation on the test sets, unseen before, *none of the models were trained on the test sets*, which were instead *used exclusively for evaluation.* To create the test sets representative of both WDC and CORD-19, two uniform samples were drawn per category. The first sample, used by two independent annotators to produce the positively labeled instances, was obtained by taking a *uniform* sample from the entire dataset. For example, one out of every N tables starting from the beginning of the dataset. N was chosen to ensure the annotators get enough data to produce 250 positive labels. The negative data of the same size was obtained by similarly taking a *uniform* sample, so that the annotators can produce 250 negative labels. The overlap with the training data was avoided by excluding the training data source. To exclude all incorrect labels, whenever two annotators disagreed, the label was discarded.

Such *heterogeneous* test set has a much wider *variety* of table representations within the category, because it was purposely drawn from the entire dataset composed from hundreds of thousands of sources. Hence, we expect it to be a more "challenging" for our models, but also more *indicative* of their real-world performance and generalizability. In other words, we expect it should be harder for a model with embeddings to perform well on the *heterogeneous* test set having a variety of different table representations of a certain category, compared to a *homogeneous* test set having just one static representation (i.e. the same attribute set). We created such test sets for 20 different categories from different application domains and report in Tables 2,3 *Precision, Recall*, and *F-measure* of models, trained with our circular and baseline embeddings, evaluated on all these different test sets.

5 EXPERIMENTAL EVALUATION

We evaluated our embeddings on tables from WDC [20] and CORD-19 [32], on 20 various table categories. To measure the performance of our circular embeddings we conducted baseline experiments with the state of the art Word2Vec [24] and ELMo [28] embeddings for comparison. We report a substantial gain in *F-Measure* of up to 34.4%[2] due to the usage of our circular embeddings. We also see that some categories completely failed when using standard state of the art embeddings (i.e. *Songs* Word2Vec with Metadata has 8% F-Measure, which is very low).

Table 2 illustrates Precision/Recall/F-Measure we got for models trained to recognize *Songs, Books, Cities* tuples with our embeddings versus models with Word2Vec or ELMo embeddings. Note that the $Average_{20}$ row is calculated for 20 different categories from both datasets, excluded from Table 2, *not just for the three categories* reported above. Each experiment has a model trained *with* and *without* metadata. The training set used for *Songs* contained a wide variety of sources, which led to better generalization of the trained Neural Network model with our circular embeddings. We observed with *Books* that the F-measure tended to be lower due to poor generalization, because of lesser variety in the training set. *Cities* performed the best as the training set had a substantial variety of data from many sources.

Table 3 demonstrates more experiments conducted for *Songs*, when varying the number of sources used to compose the training set. We started training our embeddings with a training set that lacked heterogeneity, and found as we increased the number of unique sources that we used to compose it, the performance improved. We also noticed that increasing just the size of the training set did not necessarily increase performance as metrics lacked for the models trained with our embeddings on *Songs* 420k and 600k training sets, having 420,000 and 600,000 song tuples respectively.

Finally, we also conducted 20 experiments to compare the time it takes to train our circular embeddings, compared to Word2Vec. We observed the training time was the same for all 20 categories, so we report it for one of the categories - *Cities* in Table 3. We can see the difference is substantial - training Word2Vec is 11 faster than our circular embeddings. Classification was both very

[2] Table 2: $Average_{20}$ - 20 models trained/averaged with our circular embeddings versus with Word2Vec embeddings.

fast (under one second) for both regular and circular embeddings. For training and validation we used *P3dn.24xlarge* Amazon AWS EC2 instances having 96 Intel Xeon CPUs, 8 NVidia Tesla V100 Tensor Core GPUs, and 768GB or RAM.

6 RELATED WORK

The authors in [9] construct embeddings from structured data specifically purposed for data integration tasks. Our embeddings are designed to scale up table classification, which is very different from data integration tasks. Our algorithm ("circular walk") to generate a "sentence", used to train the embeddings fully embodies 2D context around a cell in a table, which leads to substantial gains in generalization, hence table classification accuracy at Web scale. The sentence generation algorithm in [9] uses a graph, constructed per entity found in tables (i.e. *Paul in Figure 1*). It does not take into account vertical (contextual cells in the same column) context or metadata, which play significant role in Web-scale datasets and omitting this information negatively affects generalization ability of the trained models.

TABERT [33] trains a Language Model (LM) on Wikitables and shows it outperforming NLP BERT [15] on two benchmarks - SPIDER text-to-SQL [34] and WikiTableQuestions, "where a system has to infer latent DBqueries from its execution results" [25]. Both benchmarks are substantially different from Web-scale table classification that we focus on. Finally, "TABERT linearizes the structure of tables to be compatible with a Transformer-based BERT model", which wipes out 2D context, present in a table.

TAPAS [19] and is a weakly-supervised question answering model trained over tabular data. Both the target task and evaluation datasets - conversational SQA, WikiSQL, WikiTQ are substantially different from WEb-scale table classification as well as our evaluation datasets.

TURL [14] - a structure aware transformer encoder, trained and evaluated on tasks for table understanding, such as relation extraction, row population, cell filling, schema augmentation, entity linking, column type annotation. All of them are substantially different from table classification at Web-scale. Finally, TURL is a transformer encoder, which is quite different from the embedding layer in a Deep-Learning model for scalable table classification that we are constructing.

7 CONCLUSION

We introduced, defined, and evaluated new *circular embeddings*, specifically designed for structured data on two Web-scale datasets - WDC [20] having hundreds of millions of tables and CORD-19 [32] having 1.5 million tables from hundreds of thousands of sources. Compared to the state of the art baseline embeddings, we report a gain in *F-Measure* of up to 34.4% on a downstream tuple classification task. We also report that not only our average classification accuracy of structured tuples is higher, but also classification of some categories of tuples completely failed when we used state of the art baselines. This justifies our initial hypothesis that 2D context, inherent to structured data and embodied by our embeddings is valuable.

REFERENCES

[1] online: http://www.mongodb.com.

[2] When to use mlp cnn and rnn neural networks. https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/.

[3] Multilayer perceptrons for classification and regression. *Neurocomputing*, 2(5):183–197, 1991.

[4] M. Abadi. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[5] Z. Abedjan, J. Morcos, M. Gubanov, I. F. Ilyas, M. Stonebraker, P. Papotti, and M. Ouzzani. Dataxformer: Leveraging the web for semantic transformations. In *CIDR*, 2014.

[6] H. S. Alex Judea and S. Brügmann. Unsupervised training set generation for automatic acquisition of technical terminology in patents. In *ICCL*, 2014.

[7] P. A. Bernstein and S. Melnik. Model management 2.0: manipulating richer mappings. In *SIGMOD '07*, pages 1–12, New York, NY, USA, 2007. ACM Press.

[8] F. Camastra and A. Vinciarelli. Machine learning for audio, image and video analysis.

[9] R. Cappuzzo, P. Papotti, and S. Thirumuruganathan. Creating embeddings of heterogeneous relational datasets for data integration tasks. In *SIGMOD*, 2020.

[10] S. Cen, H. Zhang, Y. Chi, W. Chen, and T. Liu. Convergence of distributed stochastic variance reduced methods without sampling extra data. *CoRR*, abs/1905.12648, 2019.

[11] F. Chollet. Keras. https://keras.io, 2015.

[12] X. Chu, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In *SIGMOD*, 2015.

[13] E. F. Codd. A relational model of data for large shared data banks. *CACM*, 26(1):64–69, Jan. 1983.

[14] X. Deng, H. Sun, A. Lees, Y. Wu, and C. Yu. Turl: Table understanding through representation learning. *Proc. VLDB Endow.*, 14(3):307–319, nov 2020.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *arXiv:1810.04805v2*, 2019.

[16] X. Dong and A. Y. Halevy. Malleable schemas: A preliminary report. In *WebDB '05*, 2005.

[17] Gentile, A. Lisa, P. Ristoski, S. Eckel, D. Ritze, and H. Paulheim. Entity matching on web tables: a table embeddings approach for blocking. In *EDBT*, 2017.

[18] M. N. Gubanov, P. A. Bernstein, and A. Moshchuk. Metadata management engine for data integration with reverse-engineering support. In *ICDE*, 2008.

[19] J. Herzig, P. K. Nowak, T. Müller, F. Piccinno, and J. Eisenschlos. TaPas: Weakly supervised table parsing via pre-training. In *ACL*, July 2020.

[20] 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.

[21] J. Madhavan, P. A. Bernstein, K. Chen, A. Halevy, and P. Shenoy. Corpus-based schema matching. In *IJCAI*, 2003.

[22] J. Madhavan, P. A. Bernstein, A. Doan, and A. Halevy. Corpus-based schema matching. In *ICDE*, 2005.

[23] S. Melnik, E. Rahm, and P. A. Bernstein. Rondo: a programming platform for generic model management. In *SIGMOD*, 2003.

[24] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[25] P. Pasupat and P. Liang. Compositional semantic parsing on semi-structured tables. In *ACL*, pages 1470–1480, Beijing, China, July 2015. Association for Computational Linguistics.

[26] S. Pavia, R. Khan, A. Pyayt, and M. Gubanov. Learning tabular embeddings at web scale. In *BigData*. IEEE, 2021.

[27] S. Pavia, R. Khan, A. Pyayt, and M. Gubanov. Towards unveiling dark web structured data. In *BigData*. IEEE, 2021.

[28] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *arXiv:1802.05365v2*, 2018.

[29] R. Pottinger and P. A. Bernstein. Merging models based on given correspondences. In *VLDB*, 2003.

[30] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB JOURNAL*, 10:2001, 2001.

[31] M. Stonebraker, D. Abadi, and A. B. et al. C-store: A column-oriented dbms. In *VLDB*, 2005.

[32] L. L. Wang, K. Lo, and Y. C. et al. Cord-19: The covid-19 open research dataset. In *arXiv, cs.DL 2004.10706*, 2020.

[33] P. Yin, G. Neubig, W.-t. Yih, and S. Riedel. TaBERT: Pretraining for joint understanding of textual and tabular data. In *ACL*, pages 8413–8426, Online, July 2020. Association for Computational Linguistics.

[34] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *EMNLP*, Oct.-Nov. 2018.