kais manuscript No.
(will be inserted by the editor)

Learning Cell Embeddings for Understanding Table Layouts

Majid Ghasemi-Gol · Jay Pujara · Pedro Szekely

Received: 13 Feb 2020/Revised: 25 Jun 2020/Accepted: 04 Jul 2020

Abstract There is a large amount of data on the web in tabular form, such as Excel sheets, CSV files, and web tables. Often, tabular data is meant for human consumption, using data layouts that are difficult for machines to interpret automatically. Previous work uses the stylistic features of tabular cells (such as font size, border type, and background color) to classify tabular cells by their role in the data layout of the document (top attribute, data, metadata, etc.). In this paper, we propose a deep neural network model which can embed semantic and contextual information about tabular cells in a low-dimensional cell embedding space. We pre-train this cell embedding model on a large corpus of tabular documents from various domains. We then propose a classification technique based on Recurrent Neural Networks (RNNs) to use our pre-trained cell embeddings, combining them with stylistic features introduced in previous work, in order to improve the performance of cell type classification in complex documents. We evaluate the performance of our system on three datasets containing documents with various data layouts, in two settings, in-domain, and cross-domain training. Our evaluation result shows that our proposed cell vector representations in combination with our RNN-based classification technique significantly improves cell type classification performance.

1 Introduction

A vast amount of useful data is available in structured tabular formats, such as spreadsheets, comma-separated value files, and web tables. Tabular data is represented in a structured form following established principles of data organization (34; 11). However, understanding such data can be cognitively challenging for humans, and automated techniques for table understanding still struggle to parse arbitrary datasets. Tabular data covers many different domains and subjects and is expressed in formats

Majid Ghasemi-Gol · Jay Pujara · Pedro Szekely
Information Science Institute, University of Southern California, Marina Del Rey, CA 90292
E-mail: (ghasemig, jpujara, pszekely)@isi.edu

that include hierarchical relationships (e.g. Figure 1c) and concatenation of disparate data (e.g. Figure 1b). One useful step towards understanding tabular data is to identify elements of tabular data layout by understanding the role of each tabular cell in the data layout of the tabular document.

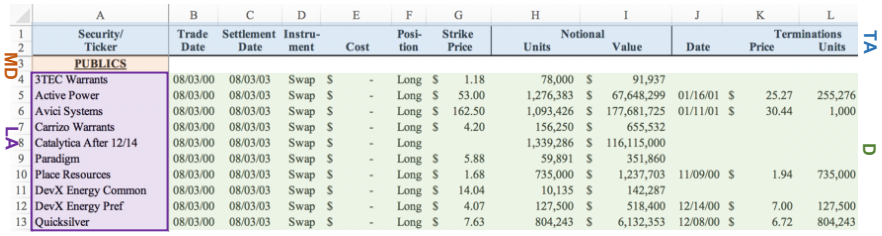
There are different definitions and terminologies used for different roles in tabular data layouts in the literature (33; 8; 22). We combine the terminologies and definitions introduced by Chen et al. (8) and Koci et al. (22) which suggest that there are six major cell types in tabular documents (Figure 1):

1. Metadata (MD): presents meta-data information for the document or part of the document. This meta-data information usually explains what the content of a document (or part of a document) is about. For example, the top meta-data block in figures 1c and 1b contains table titles and explanation of what the table presents. The inner meta-data blocks in Figure 1b are meant to state the categories of characteristics in the first column.
2. Top Attribute (TA): top attributes are the headers for table columns which can be hierarchical as in Fig. 1a.
3. Left Attribute (LA): left attributes are the row headers, and similarly to top attributes, they can be hierarchical.
4. Data (D): data cells are the core body of the table.
5. Derived (B): a cell (often with numerical value) that is derived from other cells in the table, e.g. summation of values in a column.
6. Footnotes (N): present additional information about the document or part of the document.

Pre-trained vector representations are an essential part of state of the art systems for several natural language processing tasks, including sequence tagging (23), text classification (19), and machine translation (26). Pre-training is often performed on a large corpus of unlabeled data, which enables capturing general patterns in the data. The resulting pre-trained vector representations embed the general data patterns and can be used as features for various downstream tasks (14). In this paper, we present a novel method for learning pre-trained vector representations of tabular cells (cell embeddings) and propose a novel approach for classifying tabular cells by their role type using the cell embeddings.

Previous approaches for cell role type classification focused on manually-engineered stylistic, formatting, and typographic features of tabular cells (7; 2; 22). Examples of such features are background color, font size, cell data type, and presence of capitalized letters. These features are often dependent on richly-formatted documents in a particular representation (such as Excel documents or HTML), preventing such approaches from being universally applicable. In particular, a large number of published data sources are represented in textual, tab or comma-separated formats where stylistic features are unavailable. For example *data.gov* contains about 19,000 CSV files from various domains. Moreover, such features can be prone to human error (incorrectly applying bold formatting) or overfitting to specific stylistic, formatting, or typographic conventions that cannot transfer to new domains. Unlike prior work, our proposed pre-trained cell embeddings learn representations from large number of tables using the textual content of cells alongside presentation features.

Our pre-training method leverages regularities in structure, style, and content that are present in tabular data (34; 11). We use these cell embeddings as cell features and propose a supervised classification system to achieve the cell role type classification downstream task.



	Security/ Ticker	Trade Date	Settlement Date	Instrument		Cost	Position		Strike Price	Notional Units		Notional Value	Date		Terminations Price	Terminations Units
3	PUBLICS															
4	3TEC Warrants	08/03/00	08/03/03	Swap	$	-	Long	$	1.18	78,000	$	91,937				
5	Active Power	08/03/00	08/03/03	Swap	$	-	Long	$	53.00	1,276,383	$	67,648,299	01/16/01	$	25.27	255,276
6	Avici Systems	08/03/00	08/03/03	Swap	$	-	Long	$	162.50	1,093,426	$	177,681,725	01/11/01	$	30.44	1,000
7	Carrizo Warrants	08/03/00	08/03/03	Swap	$	-	Long	$	4.20	156,250	$	655,532				
8	Catalytica After 12/14	08/03/00	08/03/03	Swap	$	-	Long			1,339,286	$	116,115,000				
9	Paradigm	08/03/00	08/03/03	Swap	$	-	Long	$	5.88	59,891	$	351,860				
10	Place Resources	08/03/00	08/03/03	Swap	$	-	Long	$	1.68	735,000	$	1,237,703	11/09/00	$	1.94	735,000
11	DevX Energy Common	08/03/00	08/03/03	Swap	$	-	Long	$	14.04	10,135	$	142,287				
12	DevX Energy Pref	08/03/00	08/03/03	Swap	$	-	Long	$	4.07	127,500	$	518,400	12/14/00	$	7.00	127,500
13	Quicksilver	08/03/00	08/03/03	Swap	$	-	Long	$	7.63	804,243	$	6,132,353	12/08/00	$	6.72	804,243

(a)

	Characteristic	1990	2000 \1	2005 \2
1	Table 198. Use of Mammography for Women 40 Years Old and Over by Patient Characteristics: 1990 to 2005			
8	**Women 40 years old and over, total \3**	**51.4**	**70.4**	**66.8**
9	40 to 49 years old	55.1	64.3	63.5
10	50 years old and over	49.7	73.6	68.4
11	50 to 64 years old	56.0	78.7	71.8
12	65 years old and over	43.4	67.9	63.8
13	White, non-Hispanic	52.7	72.2	68.4
14	Black, non-Hispanic	46.0	67.9	65.2
15	Hispanic origin \4	45.2	61.2	58.8
16	Years of school completed:			
17	No high school diploma or GED	36.4	57.7	52.8
18	High school diploma or GED	52.7	69.7	64.9
19	Some college or more	62.8	76.2	72.7
20	Poverty status: \5			
21	Below poverty	28.7	54.8	48.5
22	At or above poverty	54.8	72.1	68.8
24	Source: U.S. National Center for Health Statistics, Health, United States, annual.			

(b)

	Population group	Total law enforcement employees	Percent law enforcement employees: Male	Percent law enforcement employees: Female	Total officers	Percent officers: Male	Percent officers: Female	Total civilians	Percent civilians: Male	Percent civilians: Female	Number of agencies	2007 estimated population
1	Full-time Law Enforcement Employees											
2	by Population Group											
3	Percent Male and Female, 2007											
6	**TOTAL AGENCIES:**	**1,017,954**	**72.8**	**27.2**	**699,850**	**88.3**	**11.7**	**318,104**	**38.5**	**61.5**	**14,676**	**285,866,466**
7	**TOTAL CITIES**	**581,888**	**74.8**	**25.2**	**446,669**	**88.2**	**11.8**	**135,219**	**30.6**	**69.4**	**11,112**	**192,561,315**
8	GROUP I (250,000 and over)	203,771	70.2	29.8	152,594	83.0	17.0	51,177	32.1	67.9	71	53,815,350
9	1,000,000 and over (Group I subset)	113,693	68.5	31.5	83,852	81.7	18.3	29,841	31.3	68.7	10	25,220,230
10	500,000 to 999,999 (Group I subset)	51,812	73.2	26.8	40,163	84.0	16.0	11,649	35.9	64.1	23	15,276,719
11	250,000 to 499,999 (Group I subset)	38,266	71.4	28.6	28,579	85.4	14.6	9,687	30.1	69.9	38	13,318,401
12	GROUP II (100,000 to 249,999)	69,425	72.9	27.1	52,329	88.0	12.0	17,096	26.5	73.5	181	27,270,252
13	GROUP III (50,000 to 99,999)	68,944	76.0	24.0	53,271	90.4	9.6	15,673	27.0	73.0	441	30,331,106
14	GROUP IV (25,000 to 49,999)	64,503	77.9	22.1	50,728	91.4	8.6	13,775	27.9	72.1	806	27,684,136
15	GROUP V (10,000 to 24,999)	70,372	79.1	20.9	56,134	92.3	7.7	14,238	26.9	73.1	1,826	28,984,708
16	GROUP VI (under 10,000)	104,873	79.3	20.7	81,613	91.6	8.4	23,260	36.3	63.7	7,787	24,475,763
17	METROPOLITAN COUNTIES	301,088	69.2	30.8	173,546	86.8	13.2	127,542	45.3	54.7	1,333	65,514,896
18	NONMETROPOLITAN COUNTIES	134,978	72.0	28.0	79,635	92.6	7.4	55,343	42.3	57.7	2,231	27,790,255
19	SUBURBAN AREA[1]	471,974	72.5	27.5	302,867	88.8	11.2	169,107	43.3	56.7	7,594	121,917,604
20	[1] Suburban area includes law enforcement agencies in cities with less than 50,000 inhabitants and county law enforcement agencies that are within a Metropolitan Statistical Area (see Data Declaration). Suburban area excludes all metropolitan agencies associated with a principal city. The agencies associated with suburban areas also appear in other groups within this table.											

(c)

Fig. 1: Table layout examples. From (a) DeEx, (b) SAUS, (c) CIUS. Colors are added for annotation and not part of spreadsheets.

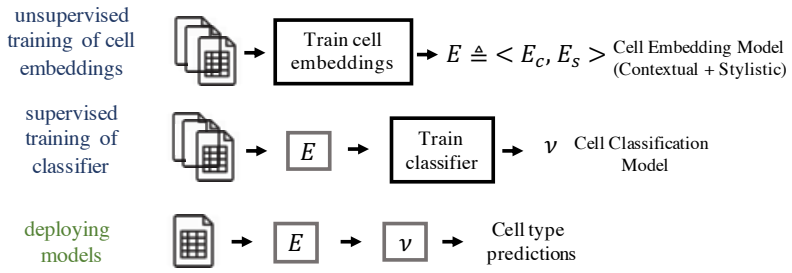


Fig. 2: Overview of our system. A training corpus of tabular documents is used to first train cell embedding models (E), and then to train the classification model (v) using the obtained cell embedding model. For a test document, first the cell embeddings are generated and then the classification model is applied to predict cell types.

To achieve the cell role type classification task using the pre-trained cell embeddings, we develop a novel, supervised cell classification model using recurrent neural networks (RNNs). The RNN model uses our cell embeddings, whose representation captures the context of nearby cells, and introduces additional long-range dependencies and context. Prior work (2; 7) sought to capture these dependencies using graphical models such as conditional random fields (CRFs), but such approaches are time-consuming to train and, in our experiments, show poor performance. Our simple and elegant architecture uses two, independent long short-term memory (LSTM) networks, one for rows and one for columns. Each of the LSTM networks uses cell embeddings with a context learned from prior cells. Together, the output vectors of these LSTMs are used to classify cells into the six cell role types.

Our method for cell type classification consists of two steps. We first build an embedding model to generate vector representations for cells in tabular documents (§2). In the second step, we develop and train an RNN-based classifier that uses these vector representations for cell type classification (§3). The cell vector representation model itself consists of two parts: the first represents global semantic information using contextual cells to produce a latent representation of the cell (§2.1), while the second represents local information from latent patterns of stylistic features of each cell (§2.2). Our classification method observes the sequence of the cells in each row and column to take into account dependencies between cell types for cells in a tabular document. The overview of our system is shown in Figure 2.

As a motivating example of the power of cell embeddings in conjunction with LSTM classification, consider the problem of identifying derived cells, a common classification task for table understanding. This task requires identifying cells whose values are computed from other cells, often using aggregation formulas such as sum, average, or variance. Successful approaches for Excel spreadsheets use the presence of formulas to identify derived cells, but formulas are unavailable for web and text-based representations. Feature-based methods attempt to identify predictive labels (such as the word “total”), but a manual process for curating such features cannot scale to the vast number of tables on the web, where domain- or language-specific

keywords abound (e.g., “ogółem” meaning total in Polish). Our embedding-based approach can use regularities in the use of words such as ”total” or ”ogółem” across a large corpus of tables to improve accuracy of detecting adjacent derived cells. In our experiments, transforming Excel sheets into CSV resulted in a dramatic 68% decline in F1 scores for feature-based classification of derived cells. In contrast, our cell embedding approach outperforms feature-based methods for both richly-formatted Excel documents and impoverished CSV representations and maintain performance with a much smaller 27% decline.

We evaluate our method on three datasets, deexcelerator (DeEx) [1], SAUS [2], and CIUS. The first two datasets have been used in previous work (22; 7). DeEx is an annotated dataset, but SAUS does not contain annotations and we manually annotated its documents. Also, we collected the CIUS dataset from *fbi.gov* website[3], and manually annotated its documents[4]. These datasets contain tables with complex data layouts and contain data from different domains (financial, business, crime, agricultural, and health-care). Example documents shown in figures 1a, 1b, and 1c are from financial, crime, and health-care domains respectively.

We compare the performance of our system with previous feature-based techniques (7; 22). In our evaluations, we test our system under both in-domain and cross-domain evaluation settings. The in-domain setting investigates the trainablity of our proposed methods. The cross-domain setting investigates the generalizability of our methods in a transfer learning scenario. In the in-domain setting, we train and test our system on each dataset separately. In the cross-domain setting, we train the model on two of our datasets and test it on the other dataset. Our experiments show that our system performs better than the baseline systems in both these settings. Finally, we perform several ablation experiments to determine the importance of different components in our proposed method.

The remainder of the paper presents our key technical contributions:

- a method for generating embedding representations for cells in a tabular data leveraging contextual content and stylistic features (§2)
- an RNN-based cell classification model using pre-trained cell embeddings and capturing long-range structural dependencies (§3)
- empirical evaluation on three real-world benchmark datasets that show state-of-the-art performance (§4)

2 Pre-training Cell Embeddings

We aim to build an unsupervised system which learns cell vector representations from unlabeled tabular documents. More formally, given a document D expressed as a tabular matrix with N rows and M columns, $D = \{C_{i,j}; 1 \leq i \leq N,\ 1 \leq j \leq M\}$, we define a collection of cells ($C_{i,j}$'s). We wish to learn an embedding operator

[1] https://wwwdb.inf.tu-dresden.de/research-projects/deexcelarator/

[2] http://dbgroup.eecs.umich.edu/project/sheets/datasets.htm

[3] https://ucr.fbi.gov/crime-in-the-u.s

[4] data and code: github.com/majidghgol/TabularCellTypeClassification

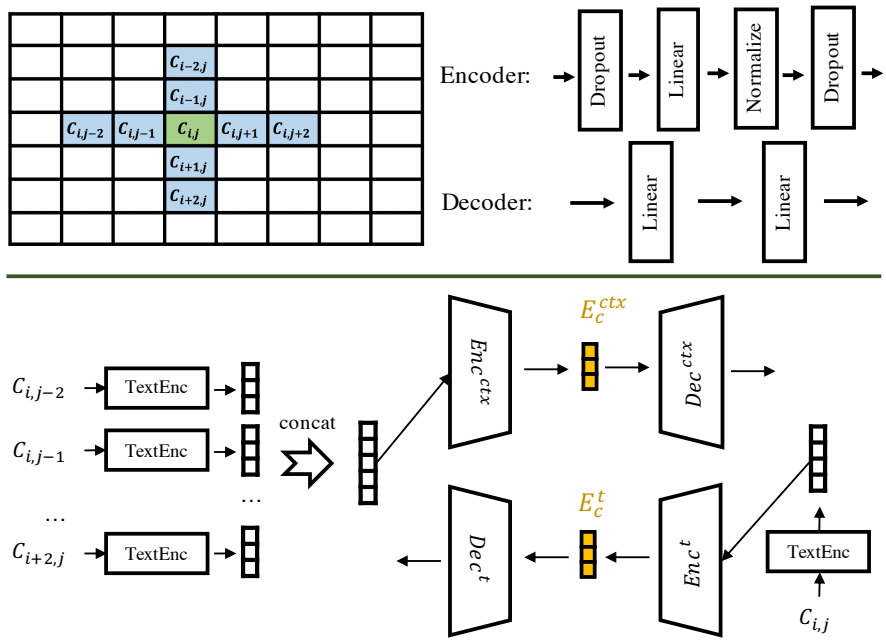


Fig. 3: Contextual cell embeddings. The green cell ($C_{i,j}$) is the target cell for which we want to calculate the cell embedding, and the blue cells are context cells. The cell values are first encoded into numerical vectors using the *TextEnc* module, and then fed into the *ctx* and *t* networks. Enc^{ctx} and Enc^t modules have the same structure, but different parameters. Same applies to Dec^{ctx} and Dec^t. The structure of these modules are depicted on the top right of this figure.

(**E**) that maps a tabular cell $C_{i,j}$ and its context to a k-dimensional vector, $V_{i,j} \in \mathbb{R}^k$. In this paper, our **E** consists of two parts. The first part represents global semantic information for a tabular cells using its textual content and context ($\mathbf{E}_c$). The second part represents local information from latent patterns of stylistic features of each cell ($\mathbf{E}_s$). We then define the cell embedding operator as concatenation of the contextual and stylistic embedding operators, i.e. $\mathbf{E} \triangleq < \mathbf{E}_c, \mathbf{E}_s >$.

2.1 Contextual Cell Embeddings ($\mathbf{E}_c$)

The textual value of a tabular cell alone does not provide much information about the cell role in the data layout. The same texts, such as "Price" may occur in vastly different contexts (e.g. in the table title, column header, or data cells). Therefore, an embedding based on the cell value alone is insufficient (as the experiments in section 4.4 shows). In order to calculate a meaningful cell representation, the context in which tabular cells appear should be taken into account.

Users often follow conventional rules (33) to arrange their data in tabular documents, for example they put the headers on top of the table, put dates in order (the

header column in 1b), and separate different parts of the document (e.g. separate tables) by empty rows or columns. Our contextual cell embeddings utilize such co-occurrences in tabular data, which is predominantly organized in two-dimensional matrices.

In natural language text, important co-occurrences are defined based on the surrounding words. Similarly, in tabular data, surrounding cells contain important information and tabular data formation is often homogeneous along tabular rows or columns. Additionally, tabular data has a non-local nature and important co-occurrences can be spatially diverse. Therefore, tabular cell context includes both its surrounding cells (local context) and some distant cells (distant context). As an example of local cell context, consider a tabular column with hierarchical headers, where the context of a lower level header cell, includes the higher level header cell in the row above. As an example of distant cell context, consider a data cell in the middle of a table, for which the column header may be many rows above and is part of its context. Distant context of tabular cell is hard to identify and requires understanding of tabular data layout (for example identifying column headers) which is not a priori known in an unsupervised setting.

In this paper, we only use the local context of tabular cell to train the contextual cell embedding operator. We define the local context of a target cell as its adjacent cells to the left, right, above and below. Based on preliminary experiments using our development set, we achieved the best performance with a neighborhood window size of 2, and our system uses 8 neighboring cells in horizontal and vertical directions as local context (blue cells in Figure 3). More formally we define the local context of a target cell $C_{i,j}$ in a tabular document D as $X_{C_{i,j}} \triangleq C_{i-2,j}, C_{i-1,j}, C_{i+1,j}, C_{i+2,j}, C_{i,j-2}, C_{i,j-1}, C_{i,j+1}, C_{i,j+2}$.

Our design for the cell embedding model uses two independent auto-encoder networks, E_c^t and E_c^{ctx}, to respectively embed both the target cell and its local context. Figure 3 shows an overview of our contextual cell embedding model. E_c^t (bottom network in Figure 3) tries to predict the value of a context cell given the value of the target cell. E_c^{ctx} (top network in Figure 3) tries to predict the value of a target cell given the value of its context cells. E_c^t and E_c^{ctx} are use similar encoder and decoder modules, which are depicted in Figure 3. Note that in both E_c^t and E_c^{ctx}, the values of target and context cells are encoded into numerical vectors using the *TextEnc* module, before feeding into the encoder modules. E_c^t and E_c^{ctx} follow architectures similar to skip-gram (SG) and continuous bag-of-words (CBOW) word embedding models respectively (25). The remainder of this section explains different parts of our cell embedding model in more details.

2.1.1 Encoding Cell Values

In word embedding methods, a vocabulary of words is assumed to be available during the training stage, allowing the generation of vector representations for all words. In our problem setting, cell values in tabular documents have a large variety and may vary from a single number to multiple sentences, violating this assumption. For our system to be able to use the cell values, they need to be encoded in a latent vector

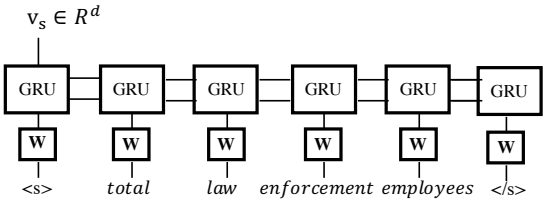


Fig. 4: Applying *InferSent* module to a cell value. The word embedding vector for each word in the cell text is fed into a bidirectional recurrent network of GRU units, and the vector representation for the cell text is collected at the output of the first GRU unit. Note that we use the pre-trained network parameters and no fine-tuning is done in the *InferSent* network parameters.

representation. More formally, for each cell value s, we wish to associate a vector representation $v_s \in \mathbb{R}^d$.

To achieve such vector representations, an encoder module for the cell values may be trained along with the context embedding network. However, in our preliminary experiments, we could not achieve stable performance with such designs, which we hypothesize may be solved by larger training corpus.

In this paper, we address this issue by using pre-trained sentence encoding models which have been shown to work well on short phrases, sentences, or collection of sentences. We experimented with two popular systems for encoding sentences and short texts, Universal Sentence Encoder (6) and InferSent (10), to generate vector representations for cell values. We choose to use InferSent in our model since it showed better performance in our preliminary experiments. Figure 4 shows an example how the InferSent module is applied to cell values. InferSent uses a bi-directional recurrent neural network consits of Gated Recurrent Units (GRUs), which is pre-trained on English sentences. For each word in the cell text (s), the associated word embedding vector from pre-trained word embeddings (W) is fed to the corresponding GRU. The output of the first GRU is collected as the text encoding vector (v_s). We use GloVe (28) pre-trained word embeddings in our model.

The sentence encoding module treats the tokens which are not present in GloVe dictionary of tokens as unknown and discards them. GloVe contains only a small number of numerical tokens. Therefore, many of the numerical tokens will not be present in its dictionary, which may be problematic since a large number of tabular cells in our datasets are numeric and contain only numerical values (e.g. data and derived cells in Figure 1b and 1c). To overcome this issue, we use a different encoding method for cells containing only numerical values. Our encoding method tries to preserve the distribution of numbers in the encoding space, and is motivated by the positional encoding method in (32). To formally introduce this encoding method, let us denote a numeric cell text as the general form of $a_n...a_1a_0.b_1b_2...b_m$. Equation (1) describes the formulas to calculate each element in the text encoding vector v_s for

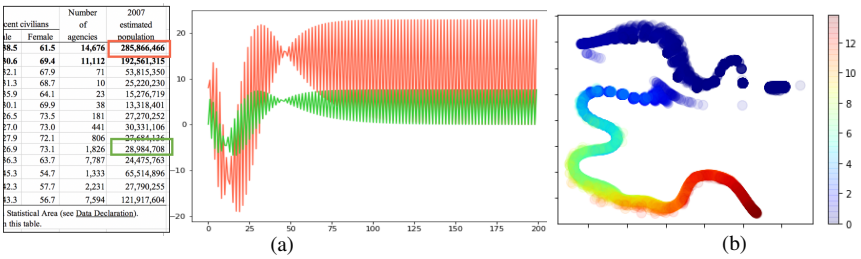


Fig. 5: Encoding numerical cell values. a) Example of applying the numeric value encoding. On the left are the encodings of digits in different positions, and on the right is their summation which is the final encoding vector (v_s). Note that here $d = 100$. b) Visualization of numeric values encodings for a random sample of 100,000 numbers between 0 and 10^7. The color of each point corresponds to natural logarithm of its associated number.

numeric cells. Note that our method assumes the numbers to be positive, and we use the absolute value of negative numbers to apply our method.

$$
\begin{aligned}
v_s^{2j} &= \sum_{i=0}^{n} a_i 2^i sin(i/10000^{2j/d}) + \sum_{i=1}^{m} b_i 2^{-i} sin(-i/10000^{2j/d}) \\
v_s^{2j+1} &= \sum_{i=0}^{n} a_i 2^i cos(i/10000^{2j/d}) + \sum_{i=1}^{m} b_i 2^{-i} cos(-i/10000^{2j/d})
\end{aligned}
\tag{1}
$$

Figure 5b shows an example of encoding vectors achieved by our number encoding formula. Note that the vector representation for "285,866,466" (highlighted with red, a derived cell) is very different from the vector representation for "28,984,708" (highlighted with green, a data cell), since the two values are distant numerically. Figure 5a shows a 2D visualization of the encoding vectors for 100,000 random positive decimal numbers. The color of each point in this figure is proportional to its associated number (warmer color means bigger number). This figure shows that our proposed number encoding method is able to preserve the distribution of numbers in the encoding vector space. Note that such information is especially useful in identifying derived cells which often have larger values compared to data cells.

2.1.2 Training network parameters

To formally explain how our proposed contextual cell embedding framework works, let us denote the TextEnc module as a function that gets the textual value of a cell and outputs a d dimensional vector representation, $I : \mathbb{S} \rightarrow \mathbb{R}^d$, where $\mathbb{S}$ is the set of all sentences. Also, let us denote the encoder and decoder modules as, $Enc^{ctx} : \mathbb{R}^{8d} \rightarrow \mathbb{R}^{d'}$, $Enc^{t} : \mathbb{R}^{d} \rightarrow \mathbb{R}^{d'}$, $Dec^{ctx} : \mathbb{R}^{d'} \rightarrow \mathbb{R}^{d}$, and $Dec^{t} : \mathbb{R}^{d'} \rightarrow \mathbb{R}^{d}$. d' is the dimension of the hidden encoder output which we consider to be the same for both E_c^{ctx} and E_c^{t}. We concatenate the context vectors and feed the resulting vector to Enc^{ctx}, causing the dimension of the input to Enc^{ctx} be $8d$.

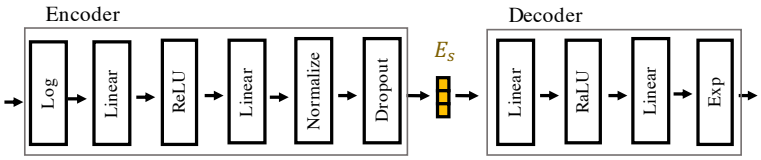


Fig. 6: Stylistic feature embeddings.

At training time, we train E_c^{ctx} and E_c^t networks separately, and try to minimize the prediction error of each network. We use *mean square error* of the network output and the desired vector (target cell value encoding for E_c^{ctx} and a context cell value encoding for E_c^t) as prediction loss measure. More formally, we define the prediction loss of E_c^{ctx} and E_c^t as:

$$l^{ctx}(\phi) = \sum_i \left| I(C_i) - Dec^{ctx}_{\phi_1}\left(Enc^{ctx}_{\phi_2}\left(I(X_{C_i})\right)\right)\right|^2 \tag{2}$$

$$l^{t}(\phi) = \sum_i \sum_{C_j \in X_{C_i}} \left| I(C_j) - Dec^{t}_{\phi_3}\left(Enc^{t}_{\phi_4}\left(I(C_i)\right)\right)\right|^2 \tag{3}$$

where $\phi =< \phi_1, \phi_2, \phi_3, \phi_4 >$ is the network parameters, and i is the training sample index (a cell in the training corpus). X_{C_i} is the set of local context cells for C_i, and $I(X_{C_i})$ is the concatenation of Infersent module output for local context cell values. Our training objective is to find the model parameters that minimize this loss function, i.e. $argmin_\phi \; l^{ctx}(\phi) + l^t(\phi)$.

During evaluation time, when dealing with a document that the model has not seen before, we use the model parameters we trained before and the value of target and context cells to generate cell embeddings for the new cells, using the output of the encoder layers in the networks. More formally, give a tabular cell $C_{i,j}$ and its context cells $X_{C_{i,j}}$, the embedding representation is: $E_c(C_{i,j}, X_{C_{i,j}}) =< E_c^{ctx}(C_{i,j}, X_{C_{i,j}}), E_c^t(C_{i,j}, X_{C_{i,j}}) >$.

It is important to note that our contextual cell embedding framework uses both skip-gram and CBoW networks in order to utilize both the target and context cells values to calculate a target cell vector representation. The Infersent module helps with adding semantic information about the cell value and its context in our cell vector representations. One other solution is to use a cell embedding vector, similar to document or paragraph vectors (24). In these methods, a vector representation for the document is calculated at test time by fixing all the parameters of the network except the document vector, and using gradient descent to infer a document vector using the words it contains. We experimented with designs that used this architecture in our preliminary experiments but were not successful. We hypothesize such approaches may be successful with more training data and training time.

2.2 Stylistic Cell Embeddings (E_s)

Spreadsheets and, to some extent, web tables are richly formatted and contain formatting, styling, and typographic information in many cells. CSV files contain only

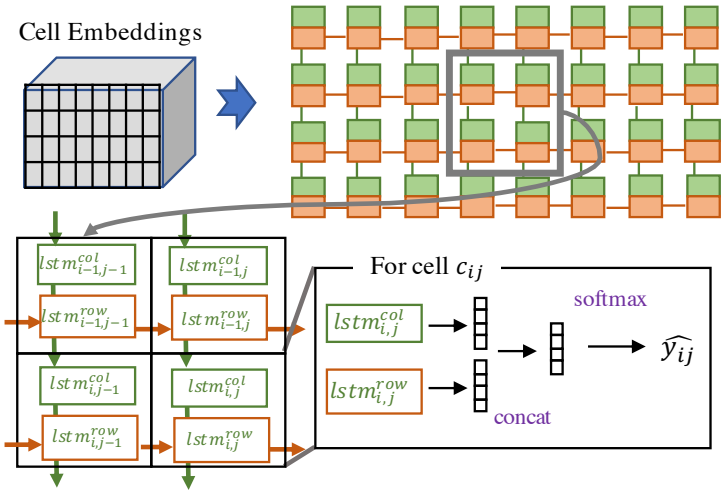


Fig. 7: RNN-based classification method.

limited formatting and typographic features. Koci et al. (22) introduced a large set of features for the cells in spreadsheets, and selected 50 features that proved to be useful in their experiments . These features include cell text features (such as presence of capital letters, presence of numbers, number of leading spaces), and cell styling features (such as font size, font color, background color, border types). These features are categorical or integers, and cannot be used directly in our classification system. We first create an integer representation for all the categorical features by indexing the categories. This results in an integer vector representing the cell features. In order to use these integer vectors alongside the cell embeddings in our classification system, we need to transform them into continuous numerical vectors. We use an auto encoder architecture as illustrated in Figure 6 to achieve this. The auto encoder network tries to reconstruct the input integer vector at its output, and generates continuous vector representations at the output of the encoder layer. We use mean square error between the output of the decoder, and the true integer vector as loss function for training the network. At test time, we feed the integer vector for cell features as input and take the stylistic embeddings (E_s) from the encoder output.

3 Classifying by cell role type

Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have been successfully used to detect coarse-grain elements, such as tables and charts, in tabular documents (4). In these works, CNNs and RNNs are used to encode tabular documents, or part of tabular documents (e.g. rows and columns). (27) uses RNN and CNN network for table type classification, and (35) uses RNNs for validating target relationships between candidate cells in tabular data as part of their framework for knowledge base creation. However, they do not use RNN networks for classifying

the cells in tabular documents. To the best of our knowledge, RNNs have not been investigated for cell level classification in tabular documents.

In our classification method, we use LSTM blocks to capture cell type dependencies in tabular documents. An LSTM block observes a sequence of input vectors $(x_1...x_n)$ and generates a hidden output for each vector in the input sequence $(h_1...h_n)$. It also maintains an internal state, and for every vector in the input sequence, the hidden output of the LSTM is a function of its state, the input vector, and its previous output. An LSTM maintains information about arbitrary points earlier in the sequence and is able to capture long-term dependencies, which is especially useful for capturing some information about the distant cell context which our cell embedding framework does not consider. For example a top attribute may be followed by a long sequence of data cells in its column and it is useful for the classifier to remember the top attribute when classifying the data cells.

Tabular formats impose cell dependencies in both its rows and columns. To capture both of these dependencies, we couple two LSTM networks (with different parameters), one observing the sequence of cells in each row, and the other observing the sequence of cells in each column. This architecture gives the LSTM blocks the ability to consider the cells on the left and above the target cell, when generating the output for the target cell. For example, in Figure 1a, when classifying the cell B17 with value of *70,372*, the column LSTM remembers the column header (and represents information that may be used to infer that this cell is a derived cell because it has the word *total* in its column header), and also the row LSTM remembers the row header. We use the cell embeddings introduced in previous section as input vectors to these LSTM networks.

Fig. 7 shows the overview of our cell classification framework. Given a document with N rows and M columns, we first generate embedding vectors for each cell in the document as explained in the previous section. We then pad the document with special vectors to distinguish borders of the document. We use 1 for left and right padding cells, and -1 for top and bottom padding cells. The result is a tensor (T_D) of size $(N+2)\times(M+2)\times d'$. There will be $N+2$ row sequences and $M+2$ column sequences for the document.

To explain how our classification framework works, let us focus on the cell in row i and column j in the tensor we created (this corresponds to the cell in row $i-1$ and column $j-1$ in the original document because of the padding process), and call it the target cell. In order to classify the target cell, the row LSTM network observes i's row, and the column LSTM network observes j's column in T_D. Moreover, j's hidden output from the row LSTM (h^r_j), and i's output from the column LSTM (h^c_i) corresponds to the target cell. We concatenate these two vectors and use a linear layer to reduce the dimension from $2d'$ to the number of cell types K. We then use a softmax layer to calculate the probabilities for different types for the target cell. More formally,

$$\hat{y}^{\phi_r,\phi_c}_{i,j} = (h^{r,\phi_r}_j, h^{c,\phi_c}_i)\theta^T + b \tag{4}$$

$$\hat{p}_{i,j}(k;\phi_r,\phi_c,\theta) = \frac{e^{\hat{y}^{k,\phi_r,\phi_c}_{i,j}}}{\sum_{k=1}^{K} e^{\hat{y}^{k,\phi_r,\phi_c}_{i,j}}} \tag{5}$$

, where $\hat{y}_{i,j}^{\phi_r,\phi_c}$ is the output of the linear layer, with size K,
$\hat{p}_{i,j}(k;\phi_r,\phi_c,\theta)$ is the k's output of the softmax layer, ϕ_r, ϕ_c, and θ are row LSTM, column LSTM, and linear layer parameters respectively.

We use a weighted *Negative Log Likelihood* as our loss function for training the classification network. The loss function can be formally written as:

$$l(\phi_1,\phi_2,\theta) = -\sum_{d_i}\sum_{i,j}\sum_{k=1}^{K} w_k y_{i,j,d_i}^{k}\hat{y}_{i,j,d_i}^{k,\phi_r,\phi_c} \tag{6}$$

where d_i is the document index in the training corpus, i is the row index, j is the column index, k is the index of cell type label, w_k is the weight of label k, $\hat{y}_{i,j,s_i}^{k,\phi_r,\phi_c}$ is given by equation 4, and y_{i,j,s_i} is a one-hot vector of size K and has a 1 element in the position of the true label for the target cell. We set w_k to be inversely proportional to the number of cells with class type k in training corpus (n_k^{train}), $w_k = 1 - \frac{n_k^{train}}{\sum_{k'=1}^{K} n_{k'}^{train}}$. The training objective is to minimize the loss function, i.e. $argmin_{\phi_1,\phi_2,\theta}\ l(\phi_1,\phi_2,\theta)$.

Given a new document during test time, the cell type for each cell in the document is calculated by using equation 5, and picking the cell type with maximum probability, i.e. $argmax_k\ \hat{p}_{i,j}(k)$.

4 Empirical Evaluation

We investigate the performance of our proposed classification method, and the quality of our proposed cell embeddings in our experiments. We investigate two research questions in our experiments. First, we investigate whether our proposed system can achieve better performance in a given domain, and whether our proposed cell embeddings capture useful information. To this end, we compare the performance of our system with the baseline systems in an in-domain training setting. Second, we investigate if our proposed system can be transferred to new domains with minimal user effort. To this end, we compare our system with baseline systems in a transfer learning scenario, where we train the models (both cell embedding and cell classification models) on two of our datasets and test them on the third one.

We also investigate the performance on documents that are not richly formatted, such as CSV files. To this end, we use a set of reduced cell features related to syntactic features of cell values (*csv features*) for the baseline systems. We refer to the complete set of cell features (which includes csv features) as *excel features*. We perform the experiments in both in-domain and cross-domain settings with csv, as well as excel features to evaluate how much the performance of the systems is dependent on rich styling features. Finally, we perform ablation experiments using additional baselines on SAUS dataset, to determine the importance of different components of our proposed method.

4.1 Evaluation Setup

4.1.1 Datasets

We evaluate our system on three real-world spreadsheet datasets containing tables with a significant variety of data layouts. The first dataset (*DeEx*), used in the DeExcelerator project[5] contains 216 annotated Excel files from ENRON, FUSE, and EUSES. The second dataset, used in (8), is 2010 Statistical Abstract of the United States *SAUS*, consisting of 1,369 Excel files downloaded from the U.S. Census Bureau. The third dataset is from the Crime In the US (*CIUS*) in 2007 and 2017, consisting of 1005 Excel files. We use the annotations provided in DeEx dataset, and manually annotate 200 and 250 Excel files, randomly selected from each of SAUS and CIUS datasets respectively. We use the XCellAnnotator Tool [6] for the annotation task. XCellAnnotator provides a user interface for manually annotating cell ranges in spreadsheets. We put each spreadsheet from these Excel files into a single document. This leads to 457, 210, and 268 annotated documents in DeEx, SAUS, and CIUS datasets.

4.1.2 Train/test split

We randomly split the documents from each dataset into train, validation, and test sets. Note that Koci et al. (22) use a different method for train/test splits in their evaluations which splits on cells rather than documents. They use a heuristic to downsample the cells from the DeEx dataset in order to remove the class imbalance caused by large number of data cells compared to other types. They then shuffle all the cells in the downsampled dataset and generate random stratified train/test splits (22). We believe that splitting by document is more appropriate as it leads to testing performance on unseen documents, where none of the cells in the test documents have been used for training. We were able to recreate the results in (22), using their train/test split approach on their downsampled dataset in our preliminary experiments, with less than 2% error.

4.1.3 Baseline Systems

We compare our system with two baseline methods that have been proposed in previous work. The first baseline is proposed by Koci et al. in (22) uses a set of manually crafted cell features which cover formatting, styling, and typographic features of tabular cells. This baseline uses a Random Forest (RF) classifier to classify individual cells in tabular documents. The second baseline is proposed in (8), and also uses manually crafted formatting, styling, and typographic cell features, but uses a Conditional Random Field (CRF) classifier for cell type classification, in order to take into account cell type dependencies.

[5] https://wwwdb.inf.tu-dresden.de/research-projects/deexcelarator/

[6] https://github.com/elviskoci/XCellAnnotator

			per-class F1						F1-Macro
			TA	D	MD	B	LA	N	
DeEx	CF	RF (22)	73.1	97.9	58.0	31.1	44.3	26.5	55.2 ± 6.3
		CRF (7)	24.4	49.4	27.4	14.0	10.4	2.1	21.3 ± 5.4
		RNN^{S}	82.1	98.7	54.9	55.2	50.2	32.1	62.2 ± 3.2
	CE	RF^{C+S}	71.2	98.8	**65.5**	45.2	55.9	20.1	59.5 ± 5.4
		RNN^{C+S}	**83.3**	**98.9**	65.0	**67.9**	**64.3**	**42.9**	**70.4 ± 3.1***
		# c	*1374*	*75110*	*1503*	*386*	*306*	*227*	-
SAUS	CF	RF (22)	93.4	97.5	84.7	44.2	93.1	90.0	83.7 ± 2.7
		CRF (7)	89.3	97.6	65.4	23.5	86.6	87.0	74.9 ± 4.2
		RNN^{S}	93.2	97.8	90.9	50.7	94.2	95.1	87.0 ± 2.1
	CE	RF^{C+S}	92.1	97.8	87.1	53.2	93.8	93.2	86.2 ± 3.3
		RNN^{C+S}	**95.1**	**98.0**	**92.6**	**62.2**	**95.0**	**95.9**	**90.4 ± 3.5***
		# c	*533*	*12667*	*50*	*486*	*1414*	*85*	-
CIUS	CF	RF (22)	98.2	99.0	99.1	86.9	94.2	**99.3**	96.2 ± 1.2
		CRF (7)	81.8	97.9	93.1	73.2	85.7	93.0	87.4 ± 2.4
		RNN^{S}	**99.9**	99.1	**99.2**	83.4	**97.6**	98.8	96.3 ± 1.3
	CE	RF^{C+S}	99.8	99.2	98.9	86.1	97.1	99.0	96.7 ± 1.1
		RNN^{C+S}	99.8	**99.3**	**99.2**	**89.6**	97.4	99.2	**97.5 ± 0.9***
		# c	*379*	*19552*	*91*	*668*	*2048*	*81*	-

Table 1: Classification scores for the case of excel features availability. *CF* and *CE* denote the manual cell features and our proposed cell embeddings respectively. *RNN* is our proposed classification method, and *RF* and *CRF* denote random forest and conditional random field methods. #c is number of cells in test set averaged over the 20 random splits. Best F1 scores for each case is bold faced, and F1-Macro scores marked with a * are statistically significant with p-value ¡ 0.05.

4.1.4 Experimental Details

In our experiments the text encoding vector dimension is $d = 4096$ (determined by InferSent module). We use $d' = 200$ for contextual cell embeddings and $d" = 30$ for the stylistic cell embeddings. We train the contextual and stylistic cell embeddings for 100 epochs, with batch sizes of 200 cells, on the train set for each dataset. We use Adam optimizer with learning rate of 0.0005 to train the networks. We also set $p = 0.1$ for the dropout layers. On an RTX 2080 GPU, training for each batch takes 10 milliseconds. We use the validation set for early stopping while training our proposed RNN-based cell classification network and use F1-macro score as the stopping criterion. We also use the validation set for tuning the hyper-parameters of the baseline classification methods. In our preliminary experiments mini-batch bagging achieved better results than the downsampling heuristic in (22), and given that it is is a more principled approach to address class type imbalance, we use mini-batch bagging for the RF baseline in our experiments. We also follow the instructions in (7) to implement the CRF baseline classifier. Since the feature set introduced by (22) is more comprehensive and covers the features in (7), we use their feature set for both RF and CRF baselines in our experiments.

			per-class F1						F1-Macro
			TA	D	MD	B	LA	N	
DeEx	CF	RF (22)	48.3	96.1	39.0	9.8	35.8	8.2	39.4 ± 7.7
		CRF (7)	28.3	49.1	28.3	5.6	1.0	0.0	18.7 ± 5.1
		RNN^S	**76.8**	98.2	54.0	46.9	42.7	20.1	56.5 ± 4.8
	CE	RF^C	38.9	97.1	**63.7**	23.2	44.5	18.3	47.6 ± 6.2
		RNN^C	73.5	**98.7**	62.7	**49.5**	**56.9**	**40.2**	**63.6 ± 3.7***
		# c	*1374*	*75110*	*1503*	*386*	*306*	*227*	-
SAUS	CF	RF (22)	93.0	97.4	83.5	24.8	92.8	89.9	80.1 ± 2.3
		CRF (7)	91.9	97.2	76.2	5.1	86.6	85.9	73.8 ± 2.1
		RNN^S	**93.8**	97.6	89.2	44.2	94.3	**95.2**	85.7 ± 2.4
	CE	RF^C	77.6	97.3	75.3	26.1	90.7	90.2	76.2 ± 3.9
		RNN^C	93.8	**98.0**	**90.5**	**57.3**	**95.3**	94.7	**88.3 ± 2.2***
		# c	*533*	*12667*	*50*	*486*	*1414*	*85*	-
CIUS	CF	RF (22)	97.9	97.7	**99.1**	55.5	93.7	**99.2**	90.6 ± 1.7
		CRF (7)	98.0	97.8	97.7	0.8	96.2	96.9	81.2 ± 1.9
		RNN^S	99.0	98.6	98.9	73.2	**97.3**	99.0	94.4 ± 2.1
	CE	RF^C	96.8	98.6	97.2	69.5	96.7	97.1	92.6 ± 1.8
		RNN^C	**99.6**	**98.8**	98.9	**81.2**	97.1	98.7	**95.7 ± 1.9***
		# c	*379*	*19552*	*91*	*668*	*2048*	*81*	-

Table 2: Classification scores for the case of csv features. For manual features (CF), only *CSV* features are used. For CE, only the context embeddings (E_c) is used.

4.2 In-domain evaluation

In order to investigate the ability of each system to learn data layout patterns from a dataset, we evaluate the systems on each dataset separately. We split the documents in each domain into 85% train, 5% validation, and 10% test sets. We repeat this evaluation 20 times on each dataset with different random train, validation, and test sets.

We first perform the experiment utilizing the excel features, i.e. we use the excel features for the baselines and use both stylistic and contextual cell embeddings in our system. Table 1 shows evaluation scores for this experiment, averaged over 20 experiments. In order to separate the effect of our proposed RNN-based classifier and cell embeddings, we add two additional systems in our experiments. The first system uses the stylistic cell embeddings and our proposed RNN-based classifier (RNN^S). The second system uses a random forest classifier on our stylistic and contextual cell embeddings (RF^{C+S}). Our full system, using both the RNN-based classifier and stylistic and contextual cell embeddings is referred to as RNN^{C+S}.

We also repeat this experiment for not richly formatted documents, i.e. we only use the csv features for the baselines and use only contextual cell embeddings in our full system (RNN^C). In order to use the csv features in RNN^S, we encode them with the same auto-encoder structure used for excel features, introduced in section 2.2. The results for this evaluation is shown in Table 2.

To explain some takeaways from these tables, let us focus on the research questions we described above.

Do our proposed contextual cell embeddings embed useful information? To investigate this question, we compare the F1-macro scores when using contextual embeddings with the cases which do not use contextual embeddings. For the case of

rich styling (Table 1), random forest classifier results in better performance using our proposed cell embeddings compared to the cell features in all three dataset. Also, our proposed RNN-based classifier achieves better performance when utilizing the contetxtual cell embeddings, compared to only stylistic cell embeddings (13% better in DeEx). When rich styling is not available (Table 2), again our RNN-based classifier performs better when using the contextual cell embeddings compared to stylistic embeddings created for csv features on all three domains (12% better in DeEx dataset). In this case, random forest classifier performs better with the cell contextual embeddings compared with csv features in DeEx dataset, and performs similarly in CIUS dataset. These results show that the contextual cell embeddings capture useful information about the cells in tabular documents, and combining them with cell stylistic features results in better classification performance specially in complex datasets such as DeEx.

To further investigate this question, Figure 8 shows a 2D visualization (obtained using the t-SNE dimension reduction method) of contextual cell embeddings for the cells in CIUS dataset. The 2D vectors are trained on all the cells in the dataset, but the visualization shows 10% of data cells, randomly selected. The plot shows clearly defined clusters, and also shows the difficulty of separating data and derived cells.

How well does our proposed RNN-based classifier perform? To investigate this question, we compare the performance of our classifier with RF and CRF baseline classifiers. Both RNN and CRF try to take into account the cell type dependencies in tabular documents. In all cases in Tables 1 and 2, RNN performs better than RF and CRF classifiers. When rich styling is not available (Table 2), RNN^{S} performs 43% and 200% better than RF and CRF respectively, when using the stylistic embeddings trained on csv features in DeEx dataset. Also, for this case RNN^{C} performs 33% better than RF^{C} when using the cell embeddings in DeEx dataset. Similar pattern is observed in the scores for the case of rich styling (Table 1). Our proposed RNN-based classifier is especially effective for classifying derived cells, and outperforms RF and CRF classifiers in all cases, except for CIUS dataset in Table 1, on derived cells. Overall, the results in Tables 1 and 2 suggests that our classifier outperforms the baseline classification methods, and is able to learn better models, especially in complex datasets such as DeEx.

How dependent is the classification performance on rich styling? To answer this question, we first compare the performance of baseline systems on csv and excel features. The scores for RF on cell features (CF) in Table 1 and 2 show that performance of RF degrades when rich styling features are unavailable, especially in DeEx, where F1-macro is 28% lower. CRF performance also degrades in all three datasets. Our RNN-based classifier suffers less when the documents lack the styling features, with about 10% drop in F1-macro score. The results suggest that the performance of feature based baselines degrades more than our system on documents without rich styling information.

Next, we analyze performance for different cell types. Comparing results in Table 1 and Table 2 suggest that classification of derived cells is difficult without rich cell styling information. The performance of our proposed system suffer mostly on derived cell type for all three datasets in Table 2. Derived cells are often similar to data cells, and are distinguished using styling (e.g. being of formula cell type or be-

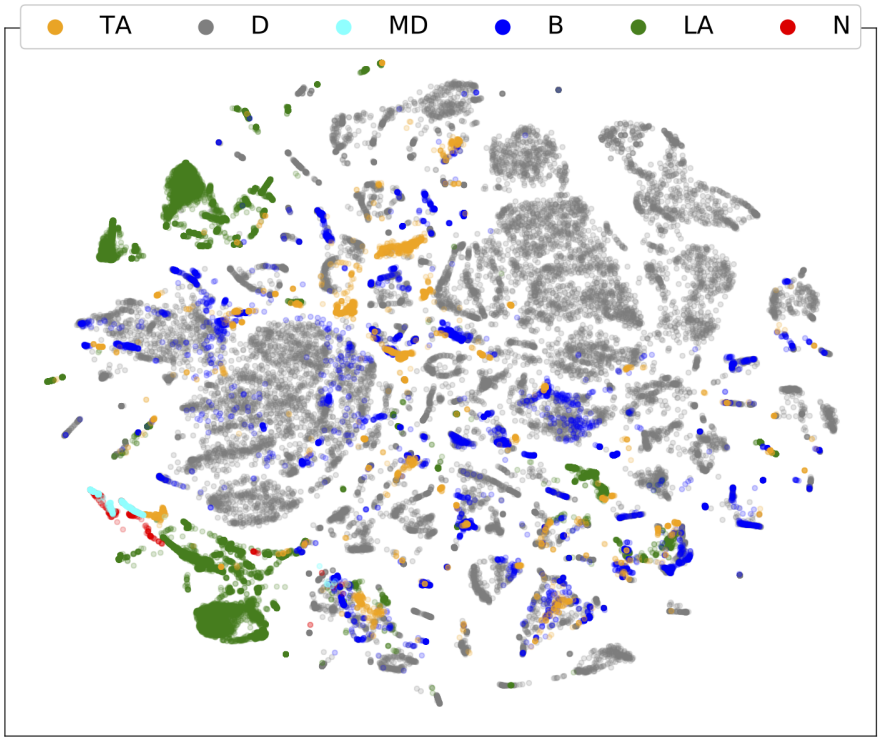


Fig. 8: 2D visualization of cell embeddings for CIUS dataset. The numbers of TA, D, MD, B, LA, and N points in this plot are 3813, 21210, 911, 6380, 22961, and 782 respectively.

ing bold faced). Classifying top attribute cells and note cells in DeEx also depends on rich styling features. For example top attribute cells are bold-faced in many cases and note cells are italic.

To summarize the results of this experiment, Tables 1 and 2 suggests that our proposed contextual embeddings combined with the RNN-based classifier results in superior cell classification performance for in-domain training setting.

4.3 Cross-domain evaluation

In the in-domain evaluation setting, we used a large set of annotated documents from each domain. However, creating such annotated training corpus for every new dataset needs significant user effort. In this section we assume we have a large training corpus available from some datasets (train datasets) which we can pre-train the classification models on. We wish to investigate whether pre-trained models can transfer to a new dataset (target dataset) with minimal user effort. To this end, we use DeEx plus one of SAUS or CIUS benchmark datasets as train datasets, and use the other one (SAUS

or CIUS) as target dataset. Because DeEx dataset has a large diversity of data layouts and styling compared to SAUS and CIUS datasets, we only use it as part of train datasets. We evaluate the performance of different systems with varying the number of annotated training documents available from the target dataset (from 0 to 100). For each case, we report the average F1-macro scores for repeating the evaluation 20 times, with different random set of training documents from the target dataset.

Our proposed cell embedding and classification models can adjust their weights iteratively using the back-propagation algorithm. To reduce the user time and effort, we pre-train the networks for contextual and stylistic embeddings, and RNN-based classification on the train datasets. We then update the model weights using the documents from the target dataset.

Our cell embedding method is unsupervised and does not require cell annotations. We first perform back-propagation for the pre-trained cell embeddings network on the target dataset for 5 epochs. The back-propagation step takes 10ms per batch of 200 cells in our experiments, so for example training on a target dataset of 1M cells takes about 4 minutes.

We then use the new cell embedding model, along with annotated training samples from the target dataset to run back-propagation for 20 epochs for our pre-trained classification network. The back-propagation step takes about 10ms on each document, so for example if there are 100 annotated documents, it takes 20 seconds to train the classification model on target dataset. Therefore, transferring our pre-trained models to a new dataset is convenient.

RF and CRF classifiers cannot adjust their models iteratively and need to be trained at once. We do not consider CRF classifier in this experiment since it takes long to train (we terminated training after 2 hours, on about 600 documents), and in our preliminary experiments, it showed very poor performance for transfer learning scenario. We train RF models on the collection of documents from the train datasets, and training documents from the target dataset. We give the training documents from the target document a larger sample weight when training the RF classsifier. Note that training the RF models only on the train set of the target dataset resulted in worse performance in our preliminary experiments.

Similar to the in-domain setting, we perform the experiments for both when rich styling is available (excel setting) and when it is not (csv setting). Tables 3 and 4 shows the experiment results for excel and csv settings respectively, for different number of training documents from the target domains. To explain some takeaways from these tables, let us focus on the research questions we are investigating.

Can the classification models transfer to a new domain? When no training documents are available from the target dataset, the performance of all systems degrades compared to in-domain training setting. The performance of RF baseline degrades 45% for SAUS dataset and csv setting (Table 4). However, in this case our proposed system (RNN^{C}) suffers less than the RF baseline and its performance degrades by 27%. RNN^{C} performs 46% better than RF baseline on SAUS dataset for csv setting (Table 4). The performance of all systems improves when training samples from the target dataset are provided. Especially, the performance of RF baseline recovers steeply, and it outperforms our system for the cases of 5 and 10 training documents on CIUS dataset. However, with more training data from the target dataset (50 and 100),

our system outperforms the RF baseline for all cases. Overall, our proposed RNN-based classifier achieves much better results in transfer learning scenario where no training data is available from a target domain, which suggests it is able to capture patterns in the tabular data layout which can be generalized to new datasets.

			F1-Macro scores					
		# target docs	0	1	5	10	50	100
SAUS	CF	RF (22)	56.0	58.0	72.0	75.5	80.8	82.7
		RNN^{S}	**67.6**	**70.7**	75.4	78.2	**83.6**	**86.0**
	CE	RF^{C+S}	66.0	66.9	69.6	71.6	81.1	83.6
		RNN^{C+S}	64.3	70.2	**75.6**	**78.6**	**83.6**	85.6
		# c	150183	149695	147281	142868	114612	80168
CIUS	CF	RF (22)	63.1	64.3	**84.7**	88.0	93.2	94.7
		RNN^{S}	71.2	73.6	79.2	81.5	93.4	95.1
	CE	RF^{C+S}	68.0	68.7	78.5	**88.9**	91.2	93.8
		RNN^{C+S}	**71.3**	**73.8**	80.4	82.8	**93.6**	**95.7**
		# c	248783	248275	244401	239847	207134	160310

Table 3: Out-domain training scores for excel setting.

			F1-Macro scores					
		# target docs	0	1	5	10	50	100
SAUS	CF	RF (22)	44.0	56.0	**69.7**	**73.0**	77.3	79.3
		RNN^{S}	59.2	62.0	68.0	71.9	**80.8**	**83.8**
	CE	RF^{C}	49.0	51.0	54.5	58.2	67.0	69.9
		RNN^{C}	**64.5**	**66.4**	68.4	71.2	79.4	82.7
		# c	150183	149695	147281	142868	114612	80168
CIUS	CF	RF (22)	58.0	60.0	75.9	**78.9**	85.1	87.7
		RNN^{S}	66.4	**71.1**	**76.1**	78.2	**88.8**	**91.5**
	CE	RF^{C}	44.0	51.2	56.1	65.1	80.8	85.2
		RNN^{C}	**67.3**	**71.1**	75.1	77.9	86.7	89.8
		# c	248783	248275	244401	239847	207134	160310

Table 4: Out-domain training scores for csv setting.

Do the contextual cell embeddings result in better model transfer? In the in-domain results, our contextual cell embeddings performed well when used in a simple random forest classifier, and also improved the performance of our RNN-based classifier (compared to just using the stylistic embeddings). In the out-domain setting, RF^{C} shows poor performance for both SAUS and CIUS when no or only a few training documents from the target dataset are available (0, 1, 5, and 10 training documents). Our RNN-based classifier achieves better performance when using the contextual cell embeddings rather than the csv features in SAUS dataset when few training documents from the target domain are available (0 and 1 in Table 4). However, RNN^{C} shows similar (or slightly worse) performance compared to RNN^{S} in other cases in Table 4. It also achieves similar results with or without using the contextual cell embeddings when rich styling is available (compare RNN^{C+S} and RNN^{S} in Table 3). This can be justified by the fact that contextual embeddings contain semantic information about the cell value (and value of its local context) which is domain specific. We hypothesize that training the contextual cell embeddings on a larger and

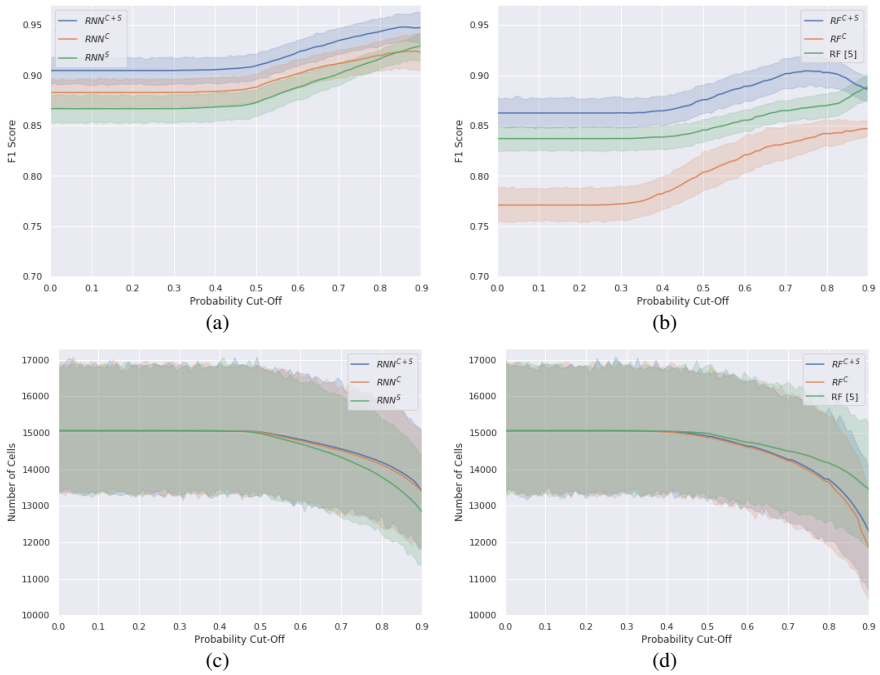


Fig. 9: F1-Macro scores, and number of predicted cell labels for different prediction probability thresholds for SAUS dataset. The highlighted region around each curve corresponds to the confidence interval. Evaluation setting is in-domain training. *RF* and RNN^S both use excel features in this experiment.

more diverse (from different domains) corpus of tabular documents can improve their generalizability.

4.4 Ablation study

Our evaluation results showed that our proposed cell classification method results in better performance both in in-domain and out-domain training settings. Our proposed cell embedding and classification models consists of various steps and components. In this section, we try to understand the contribution of different components in our method. To this end, we perform more detailed evaluations on different variants of our system. In these evaluations, we focus on the in-domain setting with excel features available on the SAUS dataset. SAUS is challenging, and can differentiate between various systems better than DeEx and CIUS datasets, according to our evaluation results in previous section.

We first take a closer look into the in-domain evaluation results in Tables 1 and 2. Figure 9 shows the classification F1-Macro scores for different cut-off thresholds for cell class type predictions probability. The curves in Figure 9a and 9b show the aver-

age F1-Macro over 20 random folds. The purpose of these plots is to determine how accurate the confident predictions are, for different classification models. The higher the probability cut-off, we expect the higher F1-Macro score. Also, Figure 9c and 9d show the number of cells predicted by each method, at each probability cut-off. Larger number of high confidence predictions translate into slower decay in the number of cells as the cut-off probability threshold increases. Note that RNN^{C+S}, RNN^{S}, RF^{C+S} and RF use excel features in this experiment. All the RNN curves in Figure 9a are above the RF curves. This suggests our proposed RNN classifier significantly improves the classification performance. Moreover, using both cell embeddings and stylistic features ($C+S$) results in better scores especially when combined with the RNN classifier. Our cell embeddings are also suited better to our RNN classifier and results in better scores compared to stylistic features (Figure 9a).

We now try to evaluate the effect of various parts of our proposed method. More specifically, we try to answer the following investigative questions:

Are the contextual cell embeddings able to capture extra contextual information? Our cell embeddings provide a vector representation for each tabular cell, which can then be used by the classification model for cell type prediction. To answer this question, we compare our contextual cell embeddings with two baseline embedding methods TE, and $WE-Avg$. TE baseline uses the output of $TextEnc$ module directly as the cell vectors. $WE-Avg$ baseline takes the average of the embedding vectors for each words in the cell text as the cell vector representation. Table 5 shows the comparison of these three cell vector representations, using RNN and RF classifiers on SAUS domain and in-domain setting. The results show that our contextual cell embeddings result in better classification score especially for RF classifier. $WE-Avg$ produce zero vectors for many numeric cells (including derived cells) since the numeric tokens are not present in the word embeddings dictionary. We see that all the derived cells are misclassified when using RF classifier. However, note that the RNN classifier performs better on derived cells (and subsequently data cells) when using $WE-Avg$ vectors. This can be due to the fact that our RNN classifier can use long range dependencies (e.g. the occurrence of the word "Total" in column header). We hypothesize that the $WE-Avg$ vectors for such header cells with short text preserve the semantic information better (note that our cell embeddings use the InferSent module) and help the RNN classifier to perform well on derived cells. However, for other cell types such as top attributes and metadata, this is not the case and RNN^{C} performs better.

Is the proposed number encoding method effective? To answer this question, Table 6 compares the classification scores of RF^{C} and RNN^{C} systems with and without using the number encoding method. Note that for the case of not using the number encoding method, numeric cells that are not in the word embeddings dictionary are treated as empty cells. The results confirms that our number encoding method overall improves the classification scores especially for derived cells. Note the number encoding causes the scores for left attributes and metadata cells slightly degrade. This is because of the fact that some of the numerical values (such as years) that appear in these cells are present in the word embedding dictionary which provides a more semantically rich vector representation than our number encoding method.

		per-class F1						F1-
		TA	D	MD	B	LA	N	Macro
SAUS	RF^{C}	77.6	97.3	75.3	26.1	90.7	90.2	**76.2** $\pm$ **3.9***
	RF^{TE}	60.75	96.3	71.0	23.75	77.8	89.0	69.8 $\pm$ 3.4
	$RF^{WE\text{-}Avg}$	37.9	96.1	59.2	0.0	80.3	85.8	59.9 $\pm$ 3.1
	RNN^{C}	93.8	98.0	90.5	57.3	95.3	94.7	**88.3** $\pm$ **2.2***
	RNN^{TE}	87.1	98.1	91.2	52.3	88.6	91.2	84.7 $\pm$ 2.6
	$RNN^{WE\text{-}Avg}$	86.6	98.5	78.3	62.4	92.4	91.4	84.9 $\pm$ 1.2

Table 5: Classification results for different methods for embedding cell values and classifying them, using in-domain training setting. The superscripts C, TE, and $WE-Avg$ respectively correspond to our proposed cell embeddings, the text encoding vectors, and average of embedding vectors for words in cell text.

		per-class F1						F1-
		TA	D	MD	B	LA	N	Macro
SAUS	RF^{C} (w/o number encoding)	67.2	96.1	82.0	4.4	92.3	88.1	71.7 $\pm$ 3.4
	RF^{C} (w/ number encoding)	77.6	97.3	75.3	26.1	90.7	90.2	**76.2** $\pm$ **3.9***
	RNN^{C} (w/o number encoding)	93.5	97.9	90.5	48.5	95.4	94.7	86.7 $\pm$ 2.1
	RNN^{C} (w/ number encoding)	93.8	98.0	90.5	57.3	95.3	94.7	**88.3** $\pm$ **2.2***

Table 6: Evaluation of the effect of our proposed number encoding method in classification results. The classification scores are for in-domain training setting.

How much the cell embeddings and RNN classifier each contribute to the results? We saw in our evaluation results that the combination of the cell embeddings and RNN classifier achieves the best performance in most of cases. To answer which of the two contribute more to our results, we refer to Table 5. The results suggest that when using a simple classifier (RF), the contextual cell embeddings outperforms other cell vector baselines by large margin (27% better compared to RF^{WE-Avg}). However, when using our RNN classifier, performance gap narrows and RNN^{C} performs 4% better compared to RNN^{WE-Avg}). This result suggests that our RNN classifier is effective in capturing cell dependencies, and our cell embedding method helps to add extra contextual information which can improve the classification result.

5 Related Work

There is a large amount of recent work investigating spreadsheets and web tables for different tasks, such as data transformation, relational data extraction, and query answering. These works often rely on lexical and stylistic features of tabular cells, and use rule-based or supervised classification techniques. We discuss these works below.

Several previous works focus on transforming spreadsheets with arbitrary data layout into database tables. These techniques often use rule based methods for the transformation. These rules are often engineered (12; 30; 29; 31), user-provided (18), or automatically inferred (15; 1). While some of these techniques use semantic information of tabular cells (15), these methods often rely on formatting, styling, and syntactic features of cells.

Some previous techniques try to extract relational data from tabular documents. Bhagavatula et al. proposed a method that relies on the DBpedia knowledge base, and uses a graphical model to jointly model three semantic interpretation tasks: entity linking, column type identification and relation extraction from tables (5). Ahsan et al. proposed a data integration through object modeling framework for spatial-temporal spreadsheets (3). Eberius et al. introduced a framework for extracting relational data from spreadsheets (16). Chen et al. introduced a semi-automatic approach using an undirected graphical model to automatically infer parent-child relationships between given cell annotations (8). (8; 16) both used manually crafted styling, typographic, and formatting features to infer tabular data layout, and are similar to the baselines in our experiments.

There are previous efforts for detecting elements of the data layout, which is the target task in this paper. Chen et al. (9) used active learning and rules to detect properties of spreadsheets, such as aggregated columns and merged cells, and integrates an active learning framework where users can provided rules to save human labeling effort. Their method is tuned for special types of tables (dataframes) . Koci et al. (22) used formatting and typographic features for cell classification, and use the classification result for layout inference. We used their method as a baseline in our experiments. They also proposed a graph representation of spreadsheets to identify layout blocks given imperfect cell layout type labels (20). These cell blocks are then used to detect tables in documents that contain multiple tables, as proposed in (21). Unlike our method, these previous techniques do not learn general cell representations and rely on data-specific stylistic features that may not generalize to new data.

More recently, approaches have been developed using continuous vector representations for tabular documents. Ghasemi-Gol et al. (17) proposed a method for calculating continuous vector representations for classifying web tables. Zhang et al. (36) introduced a system for finding relevant tables to keyword queries. They represented queries and tables in multiple semantic spaces (discrete feature space and continuous dense vector representations), and introduced similarity measures for matching table and query semantic representations. They used pre-trained word embeddings, and knowledge graph entity embeddings (for named entities) for their dense vector representations. Unlike our method, their system does not learn cell representations from tabular data itself and only uses pre-trained word and knowledge graph embeddings. Deng et al. (13) introduced a method for unsupervised training of continuous representations for various components of relational tables. They introduced word, header, entity, and core entity embeddings, and use their system for three downstream tasks, row population, column population, and table retrieval. Although similar to our method they use skip-gram model to train their embeddings and generate cell embeddings for entity and header cells. Unlike our method, they only consider relational tables with a header row and a core entity column. Our method is meant for learning general cell representations in tabular documents with complex layouts. Wu et al. (35) proposed Fonduer, a system for automatic knowledge construction from tabular documents. Their technique has three phases and uses styling, structural, and semantic information to form relations between values in cells. They use an RNN-based method in the last phase of their system to validate the candidate

relations. Unlike our method, they do not use the RNN network to directly classify all the cells in the document.

6 Conclusion

We introduced a method to generate meaningful vector representations for the cells in tabular documents, such as spreadsheets, comma separated value files, and web tables. We proposed contextual cell embeddings that capture local contextual information for tabular cells, and also encoded styling information in stylistic cell embeddings. We used these cell embeddings to classify the cells in tabular documents by their roles in the data layout of the document (cell types). To this end, we introduced an RNN-based classification algorithm which captures the dependencies between cells in the rows as well as columns in tabular documents. We evaluated the performance of our system on three datasets from different domains (financial, business, crime, agricultural, and health-care) in two evaluation settings, in-domain and cross-domain. We compared the performance of our system with two baseline systems which use manually crafted styling, formatting, and typographic features for cell type classification.

Our in-domain evaluation results suggested that our proposed contextual cell embeddings capture meaningful information about tabular cells, and utilizing them along with stylistic cell embeddings results in better cell type classification than baseline methods, especially for datasets containing documents with heterogeneous data layouts and styling conventions. Our evaluations also showed that the baseline methods are very dependent on rich styling information and perform poorly on documents which do not contain this information, such as CSV files. For such documents, using our proposed contextual cell embeddings results in better classification performance. Our cross-domain evaluations suggest that our RNN-based classifier is able to capture more general patterns in data layout of tabular documents, and transfers better than the baselines to new unseen domains with minimal user effort.

Our proposed contextual cell embeddings combined with RNN-based classifier has the potential to learn complex patterns in tabular data layouts and there is room for further investigation of its capabilities in future work. We hypothesize that training the contextual cell embeddings on larger and more diverse (from different domains) data can result in capturing more domain agnostic regularities in tabular data layout.

Acknowledgements This research is supported by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under contract number FA8650-17-C-7715. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA.

References

1. Abraham, R., Erwig, M.: Inferring templates from spreadsheets. In: Proceedings of the 28th international conference on Software engineering, pp. 182–191. ACM (2006)
2. Adelfio, M.D., Samet, H.: Schema extraction for tabular data on the web. Proceedings of the VLDB Endowment **6**(6), 421–432 (2013)

3. Ahsan, R., Neamtu, R., Rundensteiner, E.: Towards spreadsheet integration using entity identification driven by a spatial-temporal model. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing, pp. 1083–1085. ACM (2016)
4. Azunre, P., Corcoran, C., Dhamani, N., Gleason, J., Honke, G., Sullivan, D., Ruppel, R., Verma, S., Morgan, J.: Semantic classification of tabular datasets via character-level convolutional neural networks. arXiv preprint arXiv:1901.08456 (2019)
5. Bhagavatula, C.S., Noraset, T., Downey, D.: Tabel: entity linking in web tables. In: International Semantic Web Conference, pp. 425–441. Springer (2015)
6. Cer, D., Yang, Y., Kong, S.y., Hua, N., Limtiaco, N., John, R.S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., et al.: Universal sentence encoder. arXiv preprint arXiv:1803.11175 (2018)
7. Chen, Z., Cafarella, M.: Automatic web spreadsheet data extraction. In: Proceedings of the 3rd International Workshop on Semantic Search over the Web, p. 1. ACM (2013)
8. Chen, Z., Cafarella, M.: Integrating spreadsheet data via accurate and low-effort extraction. In: Proceedings of the 20th ACM SIGKDD, pp. 1126–1135. ACM (2014)
9. Chen, Z., Dadiomov, S., Wesley, R., Xiao, G., Cory, D., Cafarella, M., Mackinlay, J.: Spreadsheet property detection with rule-assisted active learning. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 999–1008. ACM (2017)
10. Conneau, A., Kiela, D., Schwenk, H., Barrault, L., Bordes, A.: Supervised learning of universal sentence representations from natural language inference data. arXiv preprint arXiv:1705.02364 (2017)
11. Crestan, E., Pantel, P.: Web-scale table census and classification. In: Proceedings of the fourth ACM international conference on Web search and data mining, pp. 545–554. ACM (2011)
12. Cunha, J., Saraiva, J., Visser, J.: From spreadsheets to relational databases and back. In: Proceedings of the 2009 ACM SIGPLAN workshop on Partial evaluation and program manipulation, pp. 179–188. ACM (2009)
13. Deng, L., Zhang, S., Balog, K.: Table2vec: Neural word and entity embeddings for table population and retrieval. arXiv preprint arXiv:1906.00041 (2019)
14. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
15. Dou, W., Han, S., Xu, L., Zhang, D., Wei, J.: Expandable group identification in spreadsheets. In: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, pp. 498–508. ACM (2018)
16. Eberius, J., Werner, C., Thiele, M., Braunschweig, K., Dannecker, L., Lehner, W.: Deexcelerator: a framework for extracting relational data from partially structured documents. In: Proceedings of the 22nd ACM international conference on Information & Knowledge Management, pp. 2477–2480. ACM (2013)
17. Ghasemi-Gol, M., Szekely, P.: Tabvec: Table vectors for classification of web tables. arXiv preprint arXiv:1802.06290 (2018)
18. Kandel, S., Paepcke, A., Hellerstein, J., Heer, J.: Wrangler: Interactive visual specification of data transformation scripts. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 3363–3372. ACM (2011)
19. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
20. Koci, E., Thiele, M., Lehner, W., Romero, O.: Table recognition in spreadsheets via a graph representation. In: 2018 13th IAPR International Workshop on Document Analysis Systems (DAS), pp. 139–144. IEEE (2018)
21. Koci, E., Thiele, M., Romero, O., Lehner, W.: Cell classification for layout recognition in spreadsheets. In: International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management, pp. 78–100. Springer (2016)
22. Koci, E., Thiele, M., Romero Moral, Ó., Lehner, W.: A machine learning approach for layout inference in spreadsheets. In: IC3K 2016: Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management: volume 1: KDIR, pp. 77–88. SciTePress (2016)
23. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. arXiv preprint arXiv:1603.01360 (2016)
24. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International Conference on Machine Learning, pp. 1188–1196 (2014)
25. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)

26. Neishi, M., Sakuma, J., Tohda, S., Ishiwatari, S., Yoshinaga, N., Toyoda, M.: A bag of useful tricks for practical neural machine translation: Embedding layer initialization and large batch size. In: Proceedings of the 4th Workshop on Asian Translation (WAT2017), pp. 99–109 (2017)
27. Nishida, K., Sadamitsu, K., Higashinaka, R., Matsuo, Y.: Understanding the semantic structures of tables with a hybrid deep neural network architecture. In: AAAI, pp. 168–174 (2017)
28. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
29. Shigarov, A.O.: Table understanding using a rule engine. Expert Systems with Applications **42**(2), 929–937 (2015)
30. Shigarov, A.O., Paramonov, V.V., Belykh, P.V., Bondarev, A.I.: Rule-based canonicalization of arbitrary tables in spreadsheets. In: International Conference on Information and Software Technologies, pp. 78–91. Springer (2016)
31. Su, H., Li, Y., Wang, X., Hao, G., Lai, Y., Wang, W.: Transforming a nonstandard table into formalized tables. In: Web Information Systems and Applications Conference, 2017 14th, pp. 311–316. IEEE (2017)
32. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems, pp. 5998–6008 (2017)
33. Wang, X.: Tabular abstraction, editing, and formatting. Ph.D. thesis, University of Waterloo (1996)
34. Wright, P., Fox, K.: Presenting information in tables. Applied Ergonomics **1**(4), 234–242 (1970)
35. Wu, S., Hsiao, L., Cheng, X., Hancock, B., Rekatsinas, T., Levis, P., Ré, C.: Fonduer: Knowledge base construction from richly formatted data. In: Proceedings of the 2018 International Conference on Management of Data, pp. 1301–1316. ACM (2018)
36. Zhang, S., Balog, K.: Ad hoc table retrieval using semantic similarity. In: Proceedings of the 2018 World Wide Web Conference, pp. 1553–1562 (2018)

Majid Ghasemig-Gol is a graduate researcher in the Center for Knowledge Graphs at Informations Science Institute. He earned a BS degree in computer engineering and an MS degree in computer science from Sharif University of Technology and University of Southern California respectively. He is currently pursuing a PhD degree at University of Southern California. His research interests include Information Extraction, Knowledge Graphs, Machine Learning.

Jay Pujara is a research assistant professor at the University of Southern California and a research lead at the Information Sciences Institute whose principal areas of research are machine learning, artificial intelligence, and data science. He completed a postdoc at UC Santa Cruz, earned his PhD at the University of Maryland, College Park and received his MS and BS at Carnegie Mellon University. Prior to his PhD, Jay spent six years at Yahoo! working on mail spam detection, user trust, and contextual mail experiences, and he has also worked at Google, LinkedIn and Oracle. Jay is the author of over thirty peer-reviewed publications and has received four best paper awards for his work. He is a recognized authority on knowledge graphs, and has organized the Automatic Knowledge Base Construction (AKBC) and Statistical Relational AI (StaRAI) workshops, has presented tutorials on knowledge graph construction at AAAI and WSDM, and has had his work featured in AI Magazine.

Dr. Pedro Szekely is a Principal Scientist and Research Director of the Center on Knowledge Graphs at the USC Information Sciences Institute (ISI), and a Research Associate Professor at the USC Computer Science Department. His research focuses on table understanding, knowledge graphs and applications of knowledge graphs. He teaches a graduate course at USC on Building Knowledge Graphs, and has given tutorials on knowledge graph construction at KDD, ISWC, AAAI and WWW.