Tabular Functional Block Detection with Embedding-based Agglomerative Cell Clustering

Kexuan Sun, Fei Wang, Muhao Chen, Jay Pujara
Department of Computer Science & Information Sciences Institute, University of Southern California
{kexuansu,fwang598,muhaoche,jpujara}@usc.edu

ABSTRACT

Tables are a widely-used format for data curation. The diversity of domains, layouts, and content of tables makes knowledge extraction challenging. Understanding table layouts is an important step for automatically harvesting knowledge from tabular data. Since table cells are spatially organized into regions, correctly identifying such regions and inferring their functional roles, referred to as functional block detection, is a critical part of understanding table layouts. Earlier functional block detection approaches fail to leverage spatial relationships and higher-level structure, either depending on cell-level predictions or relying on data types as signals for identifying blocks. In this paper, we introduce a flexible functional block detection method by applying agglomerative clustering techniques which merge smaller blocks into larger blocks using two merging strategies. Our proposed method uses cell embeddings with a customized dissimilarity function which utilizes local and margin distances, as well as block coherence metrics to capture cell, block, and table scoped features. Given the diversity of tables in real-world corpora, we also introduce a sampling-based approach for automatically tuning distance thresholds for each table. Experimental results show that our method improves over the earlier state-of-the-art method in terms of several evaluation metrics.

CCS CONCEPTS

• **Information systems → Information systems applications**.

KEYWORDS

Table processing, Block detection, Agglomerative clustering

ACM Reference Format:
Kexuan Sun, Fei Wang, Muhao Chen, Jay Pujara. 2021. Tabular Functional Block Detection with Embedding-based Agglomerative Cell Clustering. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3482484

1 INTRODUCTION

Tables offer a means to efficiently communicate complex relationships between large amounts of data. This convenience has allowed tables to become ubiquitous, with billions of tables now available in Web corpora. However, the diversity of domains, formats, and layouts of tables makes it difficult to automatically parse, extract and relate content in different tables. Consequently, diverse work has addressed these challenges, such as identifying tables in PDFs or spreadsheets [8, 11, 21], and jointly reasoning over natural language and relevant tables [2, 23, 28]. In this paper, we address the problem of understanding the structure of tables.

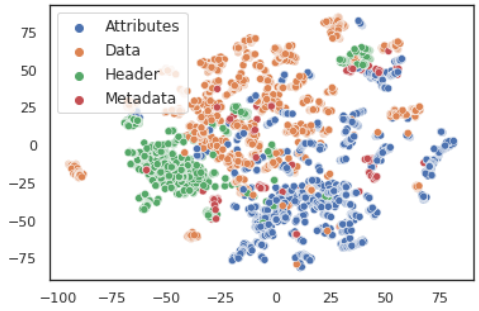


Figure 1: A t-SNE visualization of sampled functional blocks.

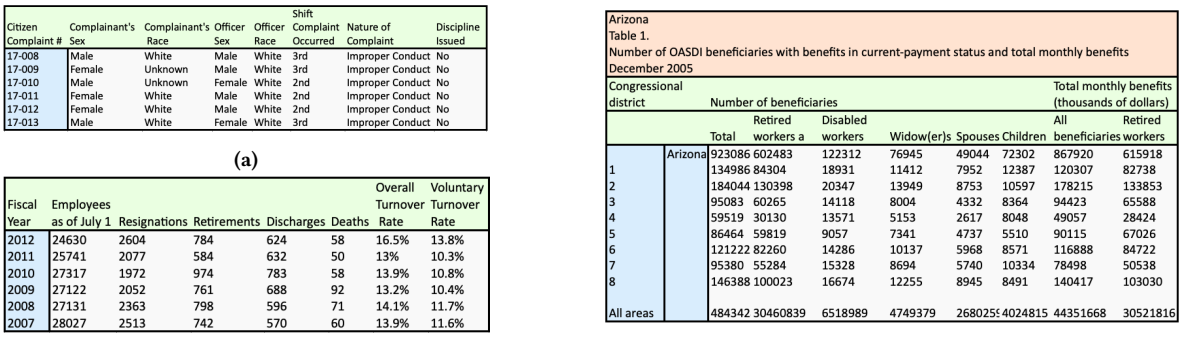
Tables communicate information through the layout of data. Individual cells of tables are spatially organized into regions of coherent functionalities. Through the relative location of groups of cells, relationships such as attribute-value relationships, indexing by dimensions, subsumption hierarchies, and metadata such as provenance can be expressed. Identifying such regions and inferring their functional roles is an essential step for analyzing table layouts. Understanding table layouts could benefit many downstream tasks. For example, table headers can be utilized as *properties* in semantic modeling [29] and identifying columns can improve table representation learning [41].

The task of functional block detection, originally introduced in [30] consists of two goals: detecting block regions and labeling the regions. However, the diversity of table structures, layouts, and content make functional block detection a challenging task. Figure 2 shows three tables and the functional roles of their components. These tables convey different types of information and have different layouts. For example, (a) consists primarily of textual values, while (b) conveys mostly numerical values, and (c) includes meta information and has nested headers or attributes.

In recent years, several attempts have addressed tabular functional block detection [30, 35]. In a recent study, Sun et al. [35] introduced a system that first used a Markov Chain Monte Carlo-based (MCMC) method to identify block boundaries and then applied a neuro-symbolic approach to infer the labels using probabilistic soft logic [1]. The main drawbacks of the MCMC method are two-fold. First, it only leverages data types whereas global spatial constraints

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11.
https://doi.org/10.1145/3459637.3482484



Citizen Complaint #	Complainant's Sex	Complainant's Race	Officer Sex	Officer Race	Shift Complaint Occurred	Nature of Complaint	Discipline Issued
17-008	Male	White	Male	White	3rd	Improper Conduct	No
17-009	Female	Unknown	Male	White	3rd	Improper Conduct	No
17-010	Male	Unknown	Female	White	2nd	Improper Conduct	No
17-011	Female	White	Male	White	2nd	Improper Conduct	No
17-012	Female	White	Male	White	2nd	Improper Conduct	No
17-013	Male	White	Female	White	3rd	Improper Conduct	No

(a)

Fiscal Year	Employees as of July 1	Resignations	Retirements	Discharges	Deaths	Overall Turnover Rate	Voluntary Turnover Rate
2012	24630	2604	784	624	58	16.5%	13.8%
2011	25741	2077	584	632	50	13%	10.3%
2010	27317	1972	974	783	58	13.9%	10.8%
2009	27122	2052	761	688	92	13.2%	10.4%
2008	27131	2363	798	596	71	14.1%	11.7%
2007	28027	2513	742	570	60	13.9%	11.6%

(b)

Arizona
Table 1.
Number of OASDI beneficiaries with benefits in current-payment status and total monthly benefits
December 2005

Congressional district		Number of beneficiaries						Total monthly benefits (thousands of dollars)	
		Total	Retired workers a	Disabled workers	Widow(er)s	Spouses	Children	All beneficiaries	Retired workers
	Arizona	923086	602483	122312	76945	49044	72302	867920	615918
1		134986	84304	18931	11412	7952	12387	120307	82738
2		184044	130398	20347	13949	8753	10597	178215	133853
3		95083	60265	14118	8004	4332	8364	94423	65588
4		59519	30130	13571	5153	2617	8048	49057	28424
5		86464	59819	9057	7341	4737	5510	90115	67026
6		121222	82260	14286	10137	5968	8571	116888	84722
7		95380	55284	15328	8694	5740	10334	78498	50538
8		146388	100023	16674	12255	8945	8491	140417	103030
All areas		484342	30460839	6518989	4749379	2680259	4024815	44351668	30521816

(c)

Figure 2: Metadata, Header, Attribute, and Data blocks are colored in orange, green, blue, and grey.

are ignored. Second, the two steps (row-wise and column-wise splitting) are inflexible for complex tables and may further propagate errors. Another task closely related to functional block detection is *cell functional role classification* [4, 20, 38]. This is a cell-level task that only infers functional roles for cells. Previous works focus on using either handcrafted stylistic or formatting features [3, 20], or embedding models to learn low-dimensional cell representation vectors [14]. Koci et al. [22] introduced an approach that also considers block structures as a way to correct imperfect cell classification predictions. Another relevant task is *table recognition* which aims to recognize tables from documents such as HTMLs [37], PDFs [11] and spreadsheets [8]. Despite the same goal of region identification as block detection, this task does not require classifying the detected regions. There is still a lack of general approaches that consider rich latent cell information for identifying functional blocks.

To address the aforementioned issues, we propose an agglomerative clustering-based method for functional block detection. Agglomerative clustering [25, 26, 40] is a class of algorithms used for cluster analysis in data mining. In general, the algorithms recursively merge small clusters into larger clusters based on a dissimilarity measure. In the context of functional block detection, each block is treated as a cluster. The algorithm starts from treating individual cells as blocks and recursively merges them into larger blocks. An agglomerative clustering method contains three components: *(1) a cluster representation method capturing essential features, (2) a measure of dissimilarity between clusters based on the representation and (3) a termination condition to stop clustering.*

Different from the MCMC method that uses data type distributions to represent blocks, we leverage the embedded cell vector representations [14] which capture individual cell features as well as surrounding context features. Figure 1 shows the positions of sampled blocks from 80 tables in the t-SNE visualization. Each block is denoted by a dot. The blocks are relatively clearly clustered based on their functional roles, which shows the usefulness of leveraging cell embeddings to reflect the latent block functional information. To measure the dissimilarity (or distance) between vector representations, there are various classic distance metrics such as Euclidean distances, Cosine distances, and Manhattan distances. In addition, metric learning algorithms are introduced to learn Mahalanobis distances [15, 34, 39]. To avoid the ambiguity between spatial distance and representational distance, we subsequently refer to representational distance as dissimilarity. Based on these distance metrics, we design a task-specific dissimilarity function considering cell information, rows/columns information, coherence of blocks and domain knowledge of data types. Another requirement for clustering algorithms is the stopping criterion. However, since tables can be diverse, a fixed threshold may not be suitable for all tables. To alleviate this issue, we introduce a personalized sampling-based approach for determining dissimilarity thresholds.

This paper presents a four-fold technical contribution: 1) We introduce a new block detection method based on agglomerative clustering that leverages cell embedding models to represent blocks and a dissimilarity function considering cell, block and table information. 2) We introduce a sampling-based method that uses synthetic blocks (i.e. randomly sampled blocks) to determine dissimilarity thresholds. 3) We analyze two different merging strategies for agglomerative clustering that maintain the rectangular constraint of blocks. 4) We provide a comprehensive set of experiments. The results show that the proposed method improves over the compared methods in both block-level and cell-level evaluations.

2 FUNCTIONAL BLOCK DETECTION

In this section, we start by formalizing the problem of functional block detection (Section 2.1), and then introduce the proposed agglomerative clustering-based block detection method. The method is composed on three parts: task-specific dissimilarity computation (Section 2.2), block merging strategies (Section 2.3), and personalized sampling-based threshold selection (Section 2.4).

2.1 Problem Definition

Given a table $T = \langle m, n \rangle$ where m is the number of rows and n is the number of columns, the goal of **functional block detection** is to identify a set of *non-overlapping rectangular* blocks $\mathcal{B} = \{B_1, B_2, \cdots, B_k\}$ where $B_i = \langle t_i, l_i, b_i, r_i \rangle$ and t_i, l_i, b_i, r_i are four indices representing the top row, left column, bottom row, and right column, respectively. Each block B_i is also assigned a label L_i. We consider four functional labels: *data* presents the main content of a table, *metadata* shows the global information of the table, *header* indicates attribute names of columns, and *attribute* denotes attribute names of rows.

Table 1 R&D Employment

Industry	NAICS Code	2000	2001
All Industries			1059.6
Computer Science	5415		62.5

Figure 3: An example table. The pairs of cells from the same and different functional blocks are denoted by red and green rectangles, respectively.

2.2 Dissimilarity Measure

An essential part of dissimilarity computation is a function that determines the closeness between two blocks based on their feature representations. Experiments by Gol et al. [14] have shown that cell embeddings capture rich information of table cells and can be used for cell functional role classification. Based on the ability of cell embeddings to capture functional roles, we adopt cell embeddings as the foundation for our block representation. In cell embeddings, each cell is represented as continuous-valued vector.

In the context of our task, we design the dissimilarity function combining four types of information: local information between blocks, margin dissimilarity between adjacent rows or columns, coherence of the new block, and domain knowledge about data types. With the dissimilarity function, at each step, the algorithm proceeds by merging a pair of adjacent blocks with minimum dissimilarity.

2.2.1 ***Block Dissimilarity.*** To merge two blocks, we first consider directly measuring the block-level dissimilarity. We leverage a block-embedding, $V(B)$, to represent each block in a vector space. Given a block is composed of a region of cells, where each cell can be represented as a vector, we represent the block using an aggregation of vectors of its constituent cells. In this paper, we use average aggregation. Let CE is the function that maps cells into a vector space, a block $B = \langle t, l, b, r\rangle$ can be represented as $V(B) = \frac{\sum_{i=t}^{b}\sum_{j=l}^{r} CE(T_{i,j})}{|B|}$ where $T_{i,j}$ is the cell at the i^{th} row and j^{th} column of T. Using this mapping, blocks B_1 and B_2 can be represented as $V_{B_1} = V(B_1)$ and $V_{B_2} = V(B_2)$ respectively. The dissimilarity between B_1 and B_2 is

$$D_B(B_1, B_2) = Dist(V_{B_1}, V_{B_2}) \tag{1}$$

where $Dist$ is a distance metric (e.g. Euclidean, Cosine, and Mahalanobis distances).

2.2.2 ***Margin Dissimilarity.*** Measuring dissimilarities between blocks only takes the local information of the overall block into account. However, higher-order information such as adjacent rows or columns are also important. For example, the table in Figure 3 presents the R&D employment information. Since "All industries" does not have an NAICS code, the cell is empty. In addition, since the data for 2000 is unavailable, all cells in that column are empty. If only considering local information, the two empty cells on the third row are likely to be merged because they have the same textual information and are spatially close. However, they have different functional roles. This issue could potentially be solved if a higher-order measure is also incorporated in the dissimilarity function. One way to measure the higher-order dissimilarity is to consider dissimilarity between cells on adjacent rows or columns, which we refer to as *margin dissimilarity*. A margin dissimilarity can be computed using pairwise dissimilarity between vectors of adjacent cells. For example, if B_1 and B_2 are row-wise adjacent where the bottom row of B_1 is row i and the top row of B_2 is row $i+1$, the margin dissimilarity between them is

$$D_M(B_1, B_2) = \frac{\sum_{k=1}^{n} Dist(CE(T_{i,k}), CE(T_{i+1,k}))}{n} \tag{2}$$

$Dist$ is the same metric as the one used for computing local dissimilarities. Likewise, if B_1 and B_2 are column-wise adjacent blocks, $D_M(B_1, B_2) = \frac{\sum_{k=1}^{m} Dist(CE(T_{k,i}), CE(T_{k,i+1}))}{m}$.

2.2.3 ***Coherence.*** In addition to minimizing the block and margin dissimilarities, another way to measure dissimilarity is the *coherence*. Coherence captures the homogeneity of cell content in the block. A highly coherent block will have all cell vectors close to each other in the embedding space, and a block of functionally-equivalent cells will have higher coherence than a random block. One way to measure the coherence is to consider the maximum distance between cell vectors and the mean vector of all cells. Formally, given a block $B = \langle t, l, b, r\rangle$, the coherence of B is:

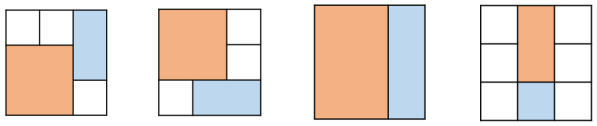
$$C_B(B) = -\max_{i=t,j=l}^{b,r} Dist(CE(T_{i,j}), V(B)) \tag{3}$$

For every block pair, the coherence is measured for the new block produced from merging the two blocks. For example, given adjacent blocks B_1 and B_2, suppose merging B_1 and B_2 results in a new block B', $C_B(B')$ will be used for scoring merging B_1 and B_2.

2.2.4 ***Domain Knowledge about Data Types.*** Besides the aforementioned measures, another useful information for assessing block dissimilarity is data types. Different data types can be a useful signal for separating blocks. For example, cells with the same data type are more likely to be in the same block than cells with different data types. In addition, different pairs of data types may have differing impedance based on the context. For example, considering fine-grained data types, a pair of cells with data types `<nominal, location>` is more likely to be in different functional blocks than those with types `<organization, location>`.

In order to capture the difference between data types, we develop a learning process that estimates the *confidence of separation* for pairs of data types. Specifically, given a table where each cell has an associated data type, each pair of adjacent cells can be indexed based on their data types. For each pair of data types, we separate the respective adjacent cell pairs into positive and negative sets, where pairs in the positive set appear in the same block and pairs in the negative set appear in different blocks. Figure 3 shows an example. For the pair of data types `<string, string>`, the pair of cells `<Industry, NAICS Code>` will be put into the positive set and the pair `<Industry, All Industries>` will be put into the negative set. Each set is associated with a distribution over the dissimilarities of the cell pairs contained in the set, and we estimate the confidence of separation based on the difference between the means of these two distributions.

Formally, for each individual pair of data types $\langle t, t'\rangle$, the two collections with pairs of cells in the same blocks and in different blocks are denoted by S_+ and S_-, respectively. The confidence of



(a) Col-wise (b) Row-wise (c) Max-merging (d) Min-merging

Figure 4: Examples of row-wise and column-wise unaligned blocks, and new blocks after Max- and Min-merging.

separation of $\langle t, t' \rangle$ is defined as

$$w_{t,t'} = \max(D_- - D_+, 0)$$

where $D_- = Avg_{(c,c') \in S\_} Dist(CE(c), CE(c'))$. D_+ is computed in the same way. This equation is designed based on the assumption: the larger the difference between the two collections is, the more separable adjacent cells with this pair of data types are. We use this confidence of separation as a weighting factor for assessing the dissimilarity between blocks. We then normalize all weighting factors such that $w_{t,t'} = 1 + \frac{w_{t,t'}}{\max_{t^*,t^{*'}} w_{t^*,t^{*'}}}$ (If a pair of data types is not seen in the training set, its weighting factor is 1). Accordingly, for adjacent blocks B_1 and B_2, the average weighting factor is

$$w_{pr}(B_1, B_2) = Avg_{(c,c') \in S_{B_1,B_2}} w_{t_c,t_{c'}} \quad (4)$$

where $S_{B_1,B_2} = \{(c, c') | c \in B_1 \& c' \in B_2 \ \& \ adjacent(c, c')\}$, t_c and $t_{c'}$ represent the data types of c and c', respectively.

2.2.5 ***Overall Dissimilarity Function.*** We combine the block dissimilarity (D_B), the margin dissimilarity (D_M), block coherence (C_B), and a domain-specific weight (w_{pr}) into an overall dissimilarity function. Given two adjacent blocks B_1 and B_2, and a new block B' generated after merging B_1 and B_2, their dissimilarity is

$$D_{all}(B_1, B_2) = w_{pr}(B_1, B_2) \cdot (D_B(B_1, B_2) + D_M(B_1, B_2) - C_B(B')) \quad (5)$$

2.3 Block Merging

Based on the task-specific dissimilarity measure introduced in the last section, the next step is to design merging strategies. In our problem setting, one goal of our task is to identify non-overlapping rectangular blocks. This indicates that block merging strategies should always satisfy the rectangular constraint. However, merging two adjacent blocks does not always result in a rectangular block. For example, Figure 4a and Figure 4b present two scenarios. The orange and blue blocks are not perfectly aligned such that merging them would lead to a non-rectangular block. Since such pairs can be very common, avoiding them would lead to early stopping and produce insufficient merging results. Accordingly, we introduce two new merging strategies to address this issue.

2.3.1 ***Max-Merging.*** Given two blocks $B_1 = \langle t_1, l_1, b_1, r_1 \rangle$ and $B_2 = \langle t_2, l_2, b_2, r_2 \rangle$, the two blocks are extended into two smallest-possible properly aligned blocks. Specifically, if B_1 and B_2 are row-wise adjacent and B_1 is above B_2, let $l_{min} = \min(l_1, l_2)$ and $r_{max} = \max(r_1, r_2)$, the new blocks are $B'_1 = \langle t_1, l_{min}, b_1, r_{max} \rangle$ and $B'_2 = \langle t_2, l_{min}, b_2, r_{max} \rangle$. Similarly, if B_1 and B_2 are column-wise adjacent, $B'_1 = \langle t_{min}, l_1, b_{max}, r_1 \rangle$ and $B'_2 = \langle t_{min}, l_2, b_{max}, r_2 \rangle$ where $t_{min} = \min(t_1, t_2)$ and $b_{max} = (b_1, b_2)$. Figure 4c shows new blocks extended from the blocks in Figure 4a. Merging the new blocks leads to a valid rectangular block.

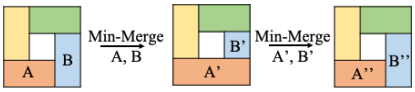
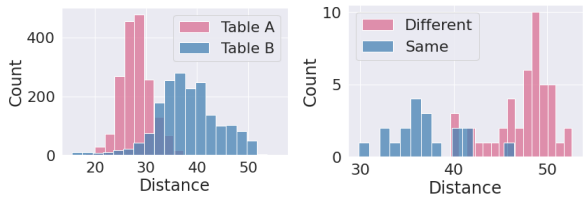


Figure 5: A possible infinite loop led by min-merging

2.3.2 ***Min-Merging.*** In addition to Max-Merging, we introduce a second strategy: *Min-Merging* to handle situations when Max-Merging will split existing blocks. Given two blocks B_1 and B_2, this strategy shrinks them into two largest-possible properly aligned blocks. Specifically, if B_1 and B_2 are row-wise adjacent, and B_1 is above B_2, $B'_1 = \langle t_1, l_{max}, b_1, r_{min} \rangle$ and $B'_2 = \langle t_2, l_{max}, b_2, r_{min} \rangle$ where $l_{max} = \max(l_1, l_2)$ and $r_{min} = \min(r_1, r_2)$. If they are column-wise adjacent, $B'_1 = \langle t_{max}, l_1, b_{min}, r_1 \rangle$ and $B'_2 = \langle t_{max}, l_2, b_{min}, r_2 \rangle$ such that $t_{max} = \max(t_1, t_2)$ and $b_{min} = \min(b_1, b_2)$. Figure 4d shows the resulting blocks from shrinking the two blocks in Figure 4b. A potential shortcoming of this strategy is that it may lead to infinite loops. Figure 5 presents an example. Merging blocks A and B leads to A' and B', and merging A' and B' brings A'' and B'' which are exactly same as A and B. To avoid infinite loops, the algorithm keeps track of block pairs which have been merged in previous steps. We refer to such block pairs as invalid block pairs. The algorithm will ignore invalid block pairs in future steps. We note that directly applying min-merging may increase the number of remaining blocks. In order to make the algorithm as efficient as possible, we also treat such block pairs as invalid block pairs.

2.4 Sampling-based Threshold Selection

Another important component in agglomerative clustering algorithms is the stopping criterion. In our problem, the stopping criterion is the dissimilarity threshold such that two blocks will not be merged if their overall dissimilarity (section 2.2.5) is larger than a threshold. However, since tables can be very diverse, a fixed dissimilarity threshold may not be suitable for all tables. For example, Figure 6a shows distance distributions of sampled block pairs from two different tables. The ranges of two distributions illustrate that different tables may have different thresholds. To deal with this problem, we introduce a sampling-based threshold selection method that provides a personalized threshold for each individual table. The details of this process is shown in Algorithm 1. Given a table, the algorithm first randomly samples a set of synthetic block pairs such that they are either row-wise or column-wise adjacent. With the dissimilarity function, each block pair will have an associated dissimilarity. We set the threshold for this specific table to be the value at the $p\%$ ($0 \leq p \leq 1$) of the frequency distribution where p is a hyper-parameter. Figure 6b shows two distance distributions of block pairs sampled from different adjacent ground-truth blocks, and those sampled from same ground-truth blocks. It is obvious that block pairs from different ground-truth blocks generally have larger distances compared to block pairs from same ground-truth blocks. In addition, the ranges of the distribution of table B in Figure 6a and that in Figure 6b are similar, which indicates that the algorithm can potentially identify reasonable thresholds.



(a) (b)

Figure 6: Dissimilarity distribution analysis. (a) presents distributions of two different tables, and (b) shows distributions of block pairs from the same and different ground-truth blocks for Table B.

2.5 The Overall Algorithm

Algorithm 2 shows the overall algorithm. The function *clustering* takes a table, and parameters p and k as the input and produces candidate blocks. At the beginning of the algorithm, every cell is treated as an active block. At each iteration, an adjacent block pair with the smallest dissimilarity (D_{all} in Section 2.2.5) is selected to merge. To merge the blocks, the algorithm checks if they can be merged without splitting other existing blocks (*valid_max_merge* in Algorithm 2). If so, the algorithm chooses to merge them using the "max-merging", otherwise "min-merging" policy. After merging them, new block pairs become active and some existing block pairs may become invalid. For example, some blocks become inactive because they are subsets of the new blocks. If the smallest dissimilarity is larger than the threshold, the algorithm stops and all current active blocks are returned as candidate blocks.

2.6 Convergence Analysis

We show that the algorithm will converge under two situations: 1) the minimum overall dissimilarity is larger than a threshold, and 2) there is no active block pairs to be merged. Commonly, the algorithm stops when it reaches the threshold. Otherwise, it stops when there are no active block pairs before reaching the threshold. Since there are two merging policies, the algorithm needs to stop in either case. For max-merge, blocks are extended without breaking other blocks. Thus, during each time of extension, at least two blocks are merged into a single block. The number of active blocks monotonically decreases. The algorithm automatically stops when there is only a single active block. For min-merge, since merging two blocks generates two new blocks, the number of active blocks does not change. However, after each min-merging, the block pair becomes invalid and will not be merged again, which avoids the infinite loops. In other words, the number of valid block pairs decreases as the algorithm proceeds. The algorithm is terminated when all active block pairs become invalid.

2.7 Time Complexity Analysis

At the beginning, the agglomerative clustering algorithm treats each individual cell as a block. For the table T with m rows and n columns, the number of active blocks are initialized to be $m \times n$. Since each block can only be merged with its adjacent blocks, there are $O(mn)$ block pairs. If applying max-merging, at each iteration, at least two blocks are merged (i.e. the number of active blocks decreases at least 1), the total number of iterations is at most $m \times n$. During each iteration, each dissimilarity computation between the new block and its one adjacent block is $O(d)$ which depends on the distance measure and the size of cell vector representations. Suppose block pairs are stored in a min-heap indexing the pairwise dissimilarity, the newly generated block pairs should be pushed into the heap, which costs $O(\log(m \cdot n))$. If the new block b has $m_b < m$ rows and $n_b < n$ columns, the maximum number of its adjacent blocks is $2 \times (m_b + n_b)$ (i.e. the number of cells adjacent to b). Therefore, the time complexity for max-merging only is $O(mn \cdot (m+n) \cdot (d+log(mn)))$. When applying min-merging, the number of active blocks does not change. If there are k times min-merging, the amortized time complexity will be $O((mn+k) \cdot (m+n) \cdot (d+log(mn)))$. For each pair of blocks applying min-merging, if we keep track of the new block generated after merging them, in the worst case, k is the total number of possible blocks, $O(n^2m^2)$.

Algorithm 1: Thresholds Selection

```
Function select_thresholds(table, p, k):
    pairs ⟵ sample k adjacent block pairs
    // Keep track of all dissimilarity values
    dists ⟵ []
    for b1, b2 ∈ pairs do
        dists.append(D_all(b1, b2))
    thre = the distance at the p% of the sorted dists
    return thre
```

Algorithm 2: Agglomerative Clustering

```
Function clustering(table, p, k):
    // table is a n × m matrix
    // Keep track of all active blocks
    active ⟵ {}
    invalid ⟵ {}
    thre ⟵ select_thresholds(table, p, k)
    for i < n do
        for j < m do
            // Four indices: top, left, bottom, right
            active.add((i, j, i, j))
    while |active| > 1 do
        // Select the block pair with minimum
           dissimilarity from all valid pairs
        b1*, b2* = argmin_{b1,b2∈active & (b1,b2)∉invalid} D_all(b1, b2)
        if D_all(b1*, b2*) ≥ thre then
            break
        // Check if max-merge without breaking blocks
        if valid_max_merge(b1*, b2*) then
            // Extend b1* and b2* and update active
            max_merge(b1*, b2*, active)
        else
            // Shrink b1* and b2* and update active
            min_merge(b1*, b2*, active)
            // Make the pair invalid
            invalid.add((b1*, b2*))
    // Return all blocks that are still valid
    return active;
```

3 EXPERIMENTS

In this section, we first introduce four datasets (Section 3.1) and experiment settings (Section 3.2). We then show several metrics designed for evaluating block quality (Section 3.3). After that, we present several experiments evaluating the performance of different methods on boundary detection (Section 3.4.1) and functional role classification (Section 3.4.2). We finally provide further analysis for the proposed method (Sections 3.5 and 3.6).

3.1 Datasets

In our experiments, following Sun et al. [35], we use the four datasets CIUS, SAUS, DeEx and DG. **CIUS** was originally collected from the Crime in the US [14] [1] containing 269 annotated sheets. **SAUS** was downloaded from the U.S. Census Burea [2]. It contains 223 annotated sheets. **DeEx** was created collected in the DeExcelerator project [9] [3] which has 444 annotated sheets. These three datasets provide cell-level functional role annotations. The last dataset is **DG** [4] which was collected and introduced in Sun et al. [35]. DG provides block-level annotations (block boundary and labels of functional roles) for 431 tables. In all these datasets, all cells and blocks are classified into four functional types: header, attributes, data and metadata.

3.2 Experimental Setup

The proposed method leverages a cell embedding model to provide cell vector representations and cell data types. We use the cell embedding model introduced in [14] as the default model, and we run the cell classifier component in the table understanding system proposed in [35] to generate cell data types. We use the most fine-grained data types: cardinal, nominal, ordinal, person, location, organization, other string, datetime and empty. We use the supervised Neighborhood Components Analysis (NCA) algorithm implemented in the metric-learn library[5] [5] to learn task-specific distance metric. We run 5-fold cross validation evaluation in all experiments where the train and develop sets have ratio 9:1. For SAUS, CIUS and DeEx datasets, we select the parameter p among [0.2, 0.3, 0.4, 0.5] according to their cell-level macro F1 scores, and for DG dataset, we select the parameter p among [0.2, 0.4, 0.6, 0.8] based on the block-level macro F1 scores. All experiments are run on a machine Intel(R) Xeon(R) Gold 5220 CPU @ 2.20GHz. We compare the proposed method with the following baseline methods:

Conditional Random Field (CRF) was originally used in [3]. The implementation uses the pystruct library with max_iter set to be 1000, tol set to be 0.01, C_range selected from [0.1, 0.3, 0.5, 0.7, 1.0] and uses the stylistic and formatting features.

Random Forest (RF) was used as a base model in the block detector in [35]. We use the implementation from scikit-learn [6]. The parameter n_estimator is selected among [100, 300], max_depth is selected among [5, 50, None], and min_sample_split is selected among [1, 10].

Recurrent Neural Network (RNN) is another classifier introduced in [14]. It uses LSTM blocks to encode neighborhood information. We set the epoch to be 50 and learning rate to be 0.0001.

Markov-Chain Monte Carlo (MCMC) method is proposed in [35]. It first performs random row-wise splitting and then column-wise splitting on row-blocks. It uses a PSL model to label blocks. The PSL model takes either RF or RNN as the base classifier. In this paper, we also use the PSL model to label blocks. In addition, to be fair, we apply a post-processing step as is used in [35].

For the first three baseline methods, we use the region-based approach from [22] to create blocks. It first merges adjacent cells on the same row to build row intervals and then merges adjacent row intervals into rectangular blocks.

3.3 Evaluation Metrics

To evaluate the performance of different methods on functional block detection, we use two main types of metrics.

3.3.1 Error-of-Boundary (EoB). EoB was originally introduced in [8] for evaluating the table detection models. It measures how precisely a predicted rectangular region is aligned with a ground-truth rectangular region. Given a ground-truth block $B = \langle t, l, b, r \rangle$ and a predicted block $B' = \langle t', l', b', r' \rangle$, the EoB between them is

$$EoB(B, B') = \max\left(|t - t'|, |b - b'|, |l - l'|, |r - r'|\right).$$

To evaluate the EoB over all blocks, Sun et al. [35] uses another variant of EoB that measures the table-level average EoB

$$EoB_t = \sum_{1 \le i \le N, 1 \le j \le M} \frac{1}{|B_{ij} \cap B'_{ij}|} EoB(B_{ij}, B'_{ij}),$$

where B_{ij} is the ground-truth block that the cell at i^{th} row and j^{th} column belongs to, and B'_{ij} is the predicted block that this cell belongs to. A smaller EoB_t indicates a better performance. In addition to such table-level EoB, to avoid the effect of the number of blocks in a table, we also evaluate on the pairwise EoB

$$EoB_p = \underset{B, B', |B \cap B'| \ge 1}{Avg} EoB(B, B').$$

3.3.2 Precision and Recall. Although the two variants of EoB are useful in evaluating the block boundary quality, it is unbounded and not designed for evaluating the classification results. We borrow metrics from the multi-class object detection task in computer vision [10] to simultaneously measure both detection and classification challenges. For each table, given a set of ground-truth blocks $\mathcal{B} = \{B_1, B_2, \cdots\}$ and a set of predicted blocks $\mathcal{B}' = \{B'_1, B'_2, \cdots\}$, each predicted block B' is assigned to a ground-truth block B according to the overlap ratio IoU:

$$IoU(B, B') = \frac{area(B) \cap area(B')}{area(B) \cup area(B')}.$$

There are following situations: 1) If multiple predicted blocks are assigned to the same ground-truth block, the one with the same label and the highest IoU is a true positive and the remaining blocks are false positives. 2) If a predicted block has the same label as the ground-truth block but the IoU < 0.5, it is a false positive. 3) If a ground-truth block has not correctly identified, it is a false negative. 4) If a predicted block cannot be matched to any of the ground-truth block properly, it is considered a false positive.

[1] https://ucr.fbi.gov/crime-in-the-u.s
[2] http://dbgroup.eecs.umich.edu/project/sheets/datasets.htm
[3] https://wwwdb.inf.tu-dresden.de/research-projects/deexcelarator/
[4] https://www.data.gov
[5] http://contrib.scikit-learn.org/metric-learn/index.html
[6] https://scikit-learn.org

Table 1: Results of block evaluations. Pr (Precision), Re (Recall) and F1 scores are percentage values. The best scores are highlighted. For AC, scores that are greater than the scores for MCMC are underlined.

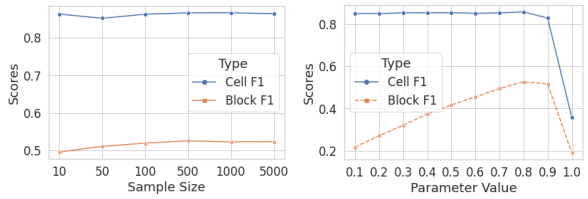
Method	EoB_t	EoB_p	Metadata			Data			Header			Attribute			Average		
			Pr	Re	F1	Pr	Re	F1	Pr	Re	F1	Pr	Re	F1	Pr	Re	F1
CRF	5357	172	**64.6**	30.2	41.0	13.3	31.6	17.6	**88.2**	77.8	**82.5**	5.6	14.5	7.8	42.9	38.5	37.2
RF	63565	321	52.3	**70.2**	**59.6**	0.3	16.7	0.5	15.5	76.3	25.6	0.3	9.5	0.5	17.1	43.2	21.6
RNN	20999	289	39.5	22.4	24.9	2.7	53.4	5.1	42.2	77.2	54.3	1.8	22.1	3.4	21.6	43.8	21.9
MCMC(RF)	3904	161	54.6	35.6	42.9	19.9	68.5	30.0	61.6	**81.5**	70.0	11.1	22.8	14.6	36.8	52.1	39.4
MCMC(RNN)	2146	151	42.3	32.0	36.2	25.8	82.5	38.9	79.4	77.3	78.3	18.6	32.0	23.1	41.5	55.9	44.1
AC(RF)	381	57	45.8	38.7	41.8	52.4	70.3	59.7	73.7	81.4	77.0	44.2	**34.2**	38.4	54.0	56.2	54.2
AC(RNN)	**245**	**39**	45.8	31.6	37.1	**61.0**	**84.3**	**70.1**	79.2	75.8	77.4	**54.4**	33.0	**41.0**	**60.1**	56.2	**56.4**

Given the evaluation criterion, for each class, we can compute a Precision, a Recall and the corresponding F1 measure. To consider the performance of whole predictions, we can evaluate the macro-average F1 over all classes accordingly.

3.4 Main Results

3.4.1 Experiment for Functional Blocks. We first evaluate the proposed method (referred to as *AC*) using the aforementioned metrics on the DG dataset. Table 1 presents the results of different methods. Note that AC uses the same labeling process as the MCMC. The main difference between MCMC and AC is the way they identify blocks. In terms of two variants of EoB which do not consider functional role labels, the proposed AC significantly outperforms the MCMC, which indicates AC has better alignments to the ground-truth blocks. In terms of macro-average Precision, Recall and F1 scores, the AC method with both RF and RNN classifiers shows significant improvements over MCMC. Among four block function classes, only F1 scores on Metadata of AC(RF) and Header of AC(RNN) performs worse than the corresponding MCMC method.

3.4.2 Auxiliary Experiment for Cells. In addition to the block-level evaluation, we also conduct an auxiliary experiment for evaluating cell-level functional roles. This is based on the assumption that better blocks could also assist the classification for cells within the blocks. Given a labeled block, all cells within this block are also automatically assigned the same label of functional role. In this experiment, we use all four datasets CIUS, SAUS, DeEx and DG. For CIUS, SAUS, and DG datasets, AC (RF) performs the best and for both RF and RNN classifiers, AC improves over the corresponding MCMC methods with a relatively large margin. The only exception is the DeEx dataset, AC (RF) shows similar performance and AC (RNN) performs worse than MCMC (RNN). In the proposed AC method, we learn the domain knowledge about data types, which uses data types predicted from a cell classifier and ground-truth blocks. For SAUS, CIUS and DeEx, we automatically create ground-truth blocks by merging same-label adjacent cells. The main possible reason of the worse performance of DeEx dataset is the cell data type errors propagated from the cell classifier and the errors of the automatically created ground-truth blocks. For example, some cells should be in the same ground-truth block but are separated by empty cells which were not annotated.



(a) Effect of the sample size **(b) Effect of the parameter p.**

Figure 7: Ablation Studies on the DG dataset.

3.5 Ablation Studies

In this section, we investigate different components of our method.

3.5.1 Effect of Dissimilarity Function Components. As is introduced in Section 2.2, in the proposed method, besides the block dissimilarity, we also leverage three types of information: margin dissimilarity, coherence of the merged block, and the domain knowledge of data types. We remove each of these three components and provide the results in table 3. In general, after removing these components, the performance on all metrics become worse. Removing the margin dissimilarity and data type knowledge, although the performance on EoB_p does not have much difference, both cell-level and block-level F1 scores decrease significantly. Removing coherence leads to much worse EoB which indicates that coherence can make dissimilarities between block pairs more diverse such that it becomes easier to find a better threshold.

3.5.2 Effect of the Number of Samples. In order to determine personalized threshold, we introduced a sampling-based method to sample synthetic block pairs and create a synthetic distance distribution. In this experiment, we show the effect of the sample size. We select the dissimilarity value at 80% of the distribution (i.e. the parameter $p = 0.8$). We choose values ranging from 10 to 5000 and show the cell-level and block-level F1 scores. Figure 7a presents the results. As the sample size increases, F1 scores on both level first increase and then become stable. This is reasonable because when the sample size is small, the synthetic block pairs are not representative enough to determine a good threshold.

3.5.3 Effect of parameter p. With a synthetic dissimilarity distribution, we still need a parameter to determine the threshold. In our

Table 2: Results of cell evaluations. Bold and underlined values have the same meaning as those in Table 1.

Method	CIUS					SAUS				
	Metadata	Data	Header	Attribute	Macro F1	Metadata	Data	Header	Attribute	Macro F1
CRF	96.5	67.6	94.9	36.8	73.9±8.9	80.7	82.2	95.7	38.2	74.2±5.8
RNN	99.5	99.2	98.2	89.2	96.5±4.0	91.9	97.0	77.6	79.5	86.5±3.7
RF	95.9	99.7	88.9	97.0	95.4±0.6	79.1	98.6	78.8	91.1	86.9±4.0
MCMC(RNN)	94.3	99.2	97.0	89.2	94.9±4.0	85.6	97.7	85.3	80.8	87.4±2.9
MCMC(RF)	93.6	99.7	96.0	97.6	96.7±1.1	80.6	99.0	85.4	92.8	89.4±2.5
AC(RNN)	96.2	99.2	98.6	89.2	95.8±3.8	93.6	97.5	84.5	80.0	88.9±3.2
AC(RF)	94.6	99.8	96.8	97.8	**97.2±0.8**	83.4	99.0	87.4	92.4	**90.6±2.1**
Method	DeEx					DG				
	Metadata	Data	Header	Attribute	Macro F1	Metadata	Data	Header	Attribute	Macro F1
CRF	35.6	55.7	48.0	1.7	35.3±6.9	42.3	53.0	95.2	32.9	55.8±7.0
RNN	33.8	96.1	47.2	39.5	54.2±5.9	25.2	96.5	85.1	80.0	71.7±3.1
RF	53.4	98.4	51.0	26.5	57.3±2.0	71.9	96.0	80.6	78.0	81.6±2.4
MCMC(RNN)	38.5	97.2	53.5	44.9	58.5±8.0	62.0	96.3	89.1	78.4	81.5±4.0
MCMC(RF)	65.4	98.8	60.5	26.0	**62.7±3.9**	73.8	95.8	91.7	76.0	84.3±4.6
AC(RNN)	50.2	98.4	57.2	18.1	56.0±5.4	68.0	96.2	90.7	77.0	83.0±2.4
AC(RF)	64.3	98.7	59.7	27.8	62.6±3.0	77.3	96.1	93.7	76.1	**85.8±4.2**

Table 3: Effect of different components in the overall distance function. $F1_c$ represents the macro-average F1 score for cell-level classification, and F1 represents the macro-average F1 score of block-level measures.

Method	EoB_t	EoB_p	Pr	Re	F1	$F1_c$
AC Full	417	60	52.9	56.6	54.0	86.5
- w/o Margin	529	66	50.2	53.5	50.6	84.7
- w/o Coherence	3212	191	31.1	54.2	37.7	85.1
- w/o Domain	519	59	51.4	49.1	48.7	81.3

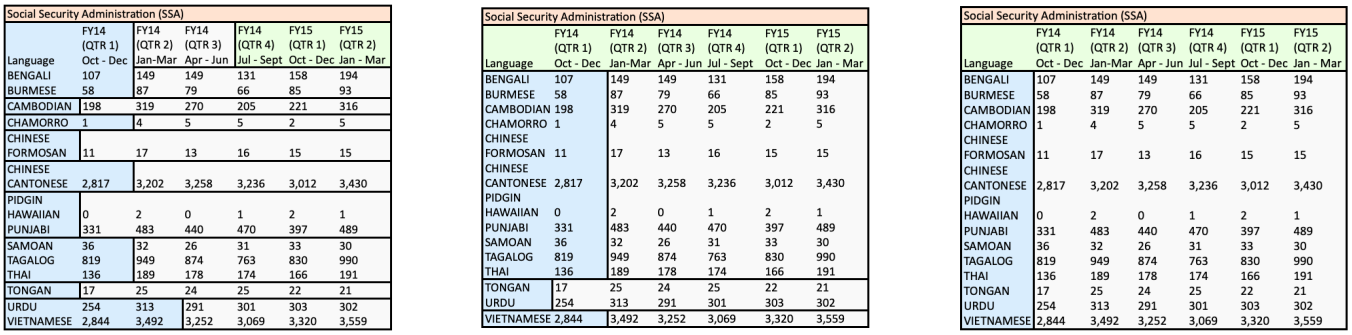
method, we use a parameter p such that the dissimilarity value at the p% of the dissimilarity distribution is selected as the threshold. In this experiment, we investigate on the value of p. We run experiments with p ranging from 0.1 to 1.0 and show the results on two F1 scores in fig. 7b. Before 0.8, as p increases, the block-level F1 score increases and the cell-level F1 score is relatively stable, which is reasonable because when p is small, the algorithm is easier to meet the stopping criterion and the number of blocks will be large and the block-level F1 score will increase. As cells within the same predicted block do have the same role, the cell-level F1 score remains stable. When p is greater than 0.8, the blocks become over-merged and both cell-level and block-level F1 scores decrease.

3.5.4 Effect of Embedding. Since the proposed method highly depends on the cell embedding, in this experiment, we remove the influence of the cell embedding and show how the method will perform in the ideal case. We use ground-truth blocks to generate synthetic dissimilarity instead of using cell embedding-dependent dissimilarity function. The synthetic distance generation is completed using the following steps. 1) We assume the ideal dissimilarity distribution follows normal distribution. 2) For each table, we sample synthetic block pairs using ground-truth blocks and constitute two distributions for block pairs the same ground-truth block, and for block pairs from different ground-truth blocks. We can then compute the mean and the variance of the two distributions. 3) We learn the average variances σ_s^2 and σ_d^2 for distributions of same ground-truth block and different ground-truth block, and the mean μ_d and μ_s of them using a training set. We then construct the two distributions $D_s = (0, \sigma_s)$ and $D_d = (\mu_d - \mu_s, \sigma_d)$. 4) During inference, when two blocks are in the same ground-truth block, we sample a distance from D_s, otherwise, from D_d.

With ideal dissimilarity distributions, the agglomerative clustering algorithm could identify nearly perfect block boundaries (with $EoB_t = 6$, $EoB_p = 1$, block $Pr = 71.1\%$, $Re = 66.3\%$ and $F1 = 68.0\%$) and better classification results (macro F1=92.5%). The not-good-enough F1 scores are attributed to the limitation of the current classifiers. This is also the reason that all Precision, Recall and F1 scores in Table 1 are generally relatively low.

3.6 Case Study

For detailed analysis, we show the output blocks of the proposed method and two strong baselines on an example table. Figure 8 presents the results. The result of AC is the same as gold labels. Since MCMC only depends on data types and some numeric values on the second column were not correctly classified, the second column is incorrectly separated from the rest of the data block. While the RF classifier predicts many cells in the second column as "attribute" and the large block is eventually classified as "attribute" block. Compared to MCMC, AC does not only depend on data types, small distances in the embedding space make the second column merged into the rest of the data block. This also indicates that the proposed AC method is more stable and more error-tolerant.



Social Security Administration (SSA)

Language	FY14 (QTR 1) Oct - Dec	FY14 (QTR 2) Jan-Mar	FY14 (QTR 3) Apr - Jun	FY14 (QTR 4) Jul - Sept	FY15 (QTR 1) Oct - Dec	FY15 (QTR 2) Jan - Mar
BENGALI	107	149	149	131	158	194
BURMESE	58	87	79	66	85	93
CAMBODIAN	198	319	270	205	221	316
CHAMORRO	1	4	5	5	2	5
CHINESE FORMOSAN	11	17	13	16	15	15
CHINESE CANTONESE	2,817	3,202	3,258	3,236	3,012	3,430
PIDGIN HAWAIIAN	0	2	0	1	2	1
PUNJABI	331	483	440	470	397	489
SAMOAN	36	32	26	31	33	30
TAGALOG	819	949	874	763	830	990
THAI	136	189	178	174	166	191
TONGAN	17	25	24	25	22	21
URDU	254	313	291	301	303	302
VIETNAMESE	2,844	3,492	3,252	3,069	3,320	3,559

(a) RF

Social Security Administration (SSA)

Language	FY14 (QTR 1) Oct - Dec	FY14 (QTR 2) Jan-Mar	FY14 (QTR 3) Apr - Jun	FY14 (QTR 4) Jul - Sept	FY15 (QTR 1) Oct - Dec	FY15 (QTR 2) Jan - Mar
BENGALI	107	149	149	131	158	194
BURMESE	58	87	79	66	85	93
CAMBODIAN	198	319	270	205	221	316
CHAMORRO	1	4	5	5	2	5
CHINESE FORMOSAN	11	17	13	16	15	15
CHINESE CANTONESE	2,817	3,202	3,258	3,236	3,012	3,430
PIDGIN HAWAIIAN	0	2	0	1	2	1
PUNJABI	331	483	440	470	397	489
SAMOAN	36	32	26	31	33	30
TAGALOG	819	949	874	763	830	990
THAI	136	189	178	174	166	191
TONGAN	17	25	24	25	22	21
URDU	254	313	291	301	303	302
VIETNAMESE	2,844	3,492	3,252	3,069	3,320	3,559

(b) MCMC

Social Security Administration (SSA)

Language	FY14 (QTR 1) Oct - Dec	FY14 (QTR 2) Jan-Mar	FY14 (QTR 3) Apr - Jun	FY14 (QTR 4) Jul - Sept	FY15 (QTR 1) Oct - Dec	FY15 (QTR 2) Jan - Mar
BENGALI	107	149	149	131	158	194
BURMESE	58	87	79	66	85	93
CAMBODIAN	198	319	270	205	221	316
CHAMORRO	1	4	5	5	2	5
CHINESE FORMOSAN	11	17	13	16	15	15
CHINESE CANTONESE	2,817	3,202	3,258	3,236	3,012	3,430
PIDGIN HAWAIIAN	0	2	0	1	2	1
PUNJABI	331	483	440	470	397	489
SAMOAN	36	32	26	31	33	30
TAGALOG	819	949	874	763	830	990
THAI	136	189	178	174	166	191
TONGAN	17	25	24	25	22	21
URDU	254	313	291	301	303	302
VIETNAMESE	2,844	3,492	3,252	3,069	3,320	3,559

(c) AC

Figure 8: Case study. Metadata, Header, Attribute, and Data blocks are colored in orange, green, blue, and grey.

4 RELATED WORK

In this section, we discuss two lines of works which are closely related to this paper.

Table and Functional Block Detection. Table detection is an important task towards automated table analysis [12, 42, 43], which seeks to recognize table layouts in documents as minimal bounding boxes. There are many research works aiming at solving this problem. Earliest studies focused on rule-based methods. For example, Hirayama [18] distinguished tables from text areas according to horizontal and vertical lines and they used a dynamic programming matching algorithm to arrange character strings. Hu et al. [19] partitions a document into a number of tables based on a set of table quality measures. Shafait and Smith [33] detected tables from more documents with more diverse layouts using an OCR-based method including detecting page columns, locating table columns, and marking table regions. Most recent approaches are based on machine learning algorithms, particularly deep-learning-based perception models. For example, Hao et al. [16] first used convolutional neural networks to detect tables. Schreiber et al. [32] and Gilani et al. [13] used Faster R-CNN [31] to solve this problem. Paliwal et al. [27] developed a model based on FCN architecture [24] to predict table and column regions pixel-wise.

Functional block detection is a recently proposed task, which, being different from detecting whole tables, focuses on identifying functionally consistent subunit blocks of each table [30]. As a first attempt, Sun et al. [35] applied a top-down Bayesian-CART based model which performs randomly row-wise and column-wise splits to produce blocks according to data types of cells. In this context, Koci et al. [22] discussed about building functional blocks in tables despite their goal is to correct imperfect cell classification results. The proposed method leverages rich knowledge encapsulated in cell vectors and constructs blocks in a more general way. In addition, the method builds blocks without knowing cell labels.

Representation Learning for Tabular Data. Among the studies of representation learning for tables, few efforts have been made to structural embeddings, while more have focused on representing semantics of tabular content. For example, TAPAS [17] employs BERT's architecture [7] to encode tables as a way to handle table-aided NLP problems. For the same purpose, TaBERT [41], another model built on top of BERT, is jointly trained over both textual and tabular data. Deng et al. [6] also designed a Transformer-based model to learn representations for tabular data, though its focus in on knowledge extraction from tables.

On the other hand, structural representation learning is more necessary than capturing semantic features in terms of functional block detection. Gol et al. [14] proposed a cell embedding model to encode individual cells considering contextual and stylistic features. Wang et al. [36] introduced a multi-granular representation learning model that aims at capturing both semantic and structural information while the model requires supervised training. Our clustering method uses the cell embedding model to represent blocks in order to determine similarities.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we attempted to address the challenges in the functional block detection task. Specifically, we introduced an agglomerative clustering-based method with different merging strategies. In the algorithm, we apply a task-specific dissimilarity measure that leverages cell vectors to determine block and margin dissimilarity, coherence and domain knowledge about data types. We then used a sampling-based approach for selecting personalized dissimilarity thresholds. There are two directions of future work. First, the method could be extended to multiple starting points (merge multiple pairs of blocks at each step) for better efficiency. Second, the block detection outputs could be used as additional information in other tasks such as table retrieval and table-based QA.

6 ACKNOWLEDGEMENTS

This material is based on research sponsored by Air Force Research Laboratory under agreement number FA8750-20-2-10002, and the Defense Advanced Research Projects Agency with award W911NF-19-20271 and FA8650-17-C-7715. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory or the U.S. Government. We thank JP Morgan for their generous support. We also thank all anonymous reviewers for their valuable comments and suggestions.

REFERENCES

[1] Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2017. Hinge-Loss Markov Random Fields and Probabilistic Soft Logic. *Journal of Machine Learning Research* 18, 1 (2017), 3846–3912.

[2] Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2019. TabFact: A Large-scale Dataset for Table-based Fact Verification. In *International Conference on Learning Representations*.

[3] Zhe Chen and Michael Cafarella. 2013. Automatic Web Spreadsheet Data Extraction. In *International Workshop on Semantic Search Over the Web*.

[4] Zhe Chen and Michael Cafarella. 2014. Integrating Spreadsheet Data via Accurate and Low-Effort Extraction. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1126–1135. https://doi.org/10.1145/2623330.2623617

[5] William de Vazelhes, CJ Carey, Yuan Tang, Nathalie Vauquier, and Aurélien Bellet. 2020. metric-learn: Metric Learning Algorithms in Python. *Journal of Machine Learning Research* 21, 138 (2020), 1–6.

[6] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2020. TURL: Table Understanding through Representation Learning. *Proceedings of the VLDB Endowment (PVLDB)* 14, 3 (2020), 307–319.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[8] Haoyu Dong, S. Liu, S. Han, Z. Fu, and D. Zhang. 2019. TableSense: Spreadsheet Table Detection with Convolutional Neural Networks. In *AAAI Conference on Artificial Intelligence*.

[9] Julian Eberius, Christoper Werner, Maik Thiele, Katrin Braunschweig, Lars Dannecker, and Wolfgang Lehner. 2013. DeExcelerator: a framework for extracting relational data from partially structured documents. In *ACM international conference on Information & Knowledge Management*. 2477–2480.

[10] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. 2010. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vision* 88, 2 (June 2010), 303–338. https://doi.org/10.1007/s11263-009-0275-4

[11] Jing Fang, Prasenjit Mitra, Zhi Tang, and C Lee Giles. 2012. Table header detection and classification. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

[12] Basilios Gatos, Dimitrios Danatsas, Ioannis Pratikakis, and Stavros J. Perantonis. 2005. Automatic Table Detection in Document Images. In *Proceedings of the Third International Conference on Advances in Pattern Recognition - Volume Part I (ICAPR'05)*. 609–618. https://doi.org/10.1007/11551188_67

[13] Azka Gilani, Shah Rukh Qasim, Imran Malik, and Faisal Shafait. 2017. Table Detection Using Deep Learning. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Vol. 01. 771–776. https://doi.org/10.1109/ICDAR.2017.131

[14] Majid Ghasemi Gol, Jay Pujara, and Pedro Szekely. 2019. Tabular Cell Classification Using Pre-Trained Cell Embeddings. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 230–239.

[15] Jacob Goldberger, Sam Roweis, Geoff Hinton, and Ruslan Salakhutdinov. 2004. Neighbourhood Components Analysis. In *Proceedings of the 17th International Conference on Neural Information Processing Systems* (Vancouver, British Columbia, Canada) *(NIPS'04)*. MIT Press, Cambridge, MA, USA, 513–520.

[16] Leipeng Hao, Liangcai Gao, Xiaohan Yi, and Zhi Tang. 2016. A Table Detection Method for PDF Documents Based on Convolutional Neural Networks. In *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*. 287–292. https://doi.org/10.1109/DAS.2016.23

[17] Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly Supervised Table Parsing via Pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 4320–4333. https://doi.org/10.18653/v1/2020.acl-main.398

[18] Y. Hirayama. 1995. A method for table structure analysis using DP matching. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, Vol. 2. 583–586 vol.2. https://doi.org/10.1109/ICDAR.1995.601964

[19] Jianying Hu, Ramanujan S. Kashi, Daniel P. Lopresti, and Gordon Wilfong. 1999. Medium-independent table detection. In *Document Recognition and Retrieval VII*, Vol. 3967. International Society for Optics and Photonics, SPIE, 291 – 302. https://doi.org/10.1117/12.373506

[20] Elvis Koci, Maik Thiele, Oscar Romero, and Wolfgang Lehner. 2016. A Machine Learning Approach for Layout Inference in Spreadsheets. In *International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*. 77–88.

[21] Elvis Koci, Maik Thiele, Oscar Romero, and Wolfgang Lehner. 2017. Table Identification and Reconstruction in Spreadsheets. In *International Conference on Advanced Information Systems Engineering*.

[22] Elvis Koci, Maik Thiele, Oscar Romero, and Wolfgang Lehner. 2019. Cell Classification for Layout Recognition in Spreadsheets. In *Knowledge Discovery, Knowledge Engineering and Knowledge Management*. 78–100.

[23] Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural Text Generation from Structured Data with Application to the Biography Domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 1203–1213.

[24] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 3431–3440. https://doi.org/10.1109/CVPR.2015.7298965

[25] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press. https://doi.org/10.1017/CBO9780511809071

[26] Daniel Müllner. 2011. Modern hierarchical, agglomerative clustering algorithms. arXiv:1109.2378

[27] Shubham Paliwal, Vishwanath D, Rohit Rahul, Monika Sharma, and Lovekesh Vig. 2020. TableNet: Deep Learning model for end-to-end Table detection and Tabular data extraction from Scanned Document Images. arXiv:2001.01469 [cs.CV]

[28] Panupong Pasupat and Percy Liang. 2015. Compositional Semantic Parsing on Semi-Structured Tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 1470–1480.

[29] Minh Pham, Suresh Alse, Craig Knoblock, and Pedro Szekely. 2016. Semantic Labeling: A Domain-Independent Approach. 446–462. https://doi.org/10.1007/978-3-319-46523-4_27

[30] Jay Pujara, Arunkumar Rajendran, Majid Ghasemi-Gol, and Pedro Szekely. 2019. A Common Framework for Developing Table Understanding Models. In *International Semantic Web Conference*.

[31] Shaoqing Ren, Kaiming He Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems (NIPS)*.

[32] Sebastian Schreiber, Stefan Agne, Ivo Wolf, Andreas Dengel, and Sheraz Ahmed. 2017. DeepDeSRT: Deep Learning for Detection and Structure Recognition of Tables in Document Images. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Vol. 01. 1162–1167. https://doi.org/10.1109/ICDAR.2017.192

[33] Faisal Shafait and Ray Smith. 2010. Table Detection in Heterogeneous Documents. In *Document Analysis Systems 2010*. http://doi.acm.org/10.1145/1815330.1815339

[34] Masashi Sugiyama. 2007. Dimensionality Reduction of Multimodal Labeled Data by Local Fisher Discriminant Analysis. *J. Mach. Learn. Res.* 8 (May 2007), 1027–1061.

[35] Kexuan Sun, Harsha Rayudu, and Jay Pujara. 2021. A Hybrid Probabilistic Approach for Table Understanding. In *Thirty-Fifth AAAI Conference on Artificial Intelligence*.

[36] Fei Wang, Kexuan Sun, Muhao Chen, Jay Pujara, and Pedro A. Szekely. 2021. Retrieving Complex Tables with Multi-Granular Graph Representation Learning. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

[37] Jingjing Wang, Haixun Wang, Zhongyuan Wang, and Kenny Q. Zhu. 2012. Understanding Tables on the Web. In *Conceptual Modeling*, Paolo Atzeni, David Cheung, and Sudha Ram (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 141–155.

[38] Yalin Wang and Jianying Hu. 2002. A Machine Learning Based Approach for Table Detection on the Web. In *Proceedings of the 11th International Conference on World Wide Web*. 242–250. https://doi.org/10.1145/511446.511478

[39] Kilian Q. Weinberger and Lawrence K. Saul. 2009. Distance Metric Learning for Large Margin Nearest Neighbor Classification. *J. Mach. Learn. Res.* 10 (June 2009), 207–244.

[40] Rui Xu and D. Wunsch. 2005. Survey of clustering algorithms. *IEEE Transactions on Neural Networks* 16, 3 (2005), 645–678. https://doi.org/10.1109/TNN.2005.845141

[41] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 8413–8426.

[42] Yanhong Zhai and Bing Liu. 2005. Web Data Extraction Based on Partial Tree Alignment. In *Proceedings of the 14th International Conference on World Wide Web*. Association for Computing Machinery, 76–85. https://doi.org/10.1145/1060745.1060761

[43] K. Zuyev. 1997. Table image segmentation. In *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, Vol. 2. 705–708 vol.2. https://doi.org/10.1109/ICDAR.1997.620599