SPRINT: Script-agnostic Structure Recognition in Tables

Dhruv Kudale(✉), Badri Vishal Kasuba, Venkatapathy Subramanian, Parag Chaudhuri, and Ganesh Ramakrishnan

Department of Computer Science and Engineering, IIT Bombay, Mumbai, India
{dhruvk,badrivishalk,venkatapathy,paragc,ganesh}@cse.iitb.ac.in

Abstract. Table Structure Recognition (TSR) is vital for various downstream tasks like information retrieval, table reconstruction, and document understanding. While most state-of-the-art (SOTA) research predominantly focuses on TSR in English documents, the need for similar capabilities in other languages is evident, considering the global diversity of data. Moreover, creating substantial labeled data in non-English languages and training these SOTA models from scratch is costly and time-consuming. We propose TSR as a language-agnostic cell arrangement prediction and introduce SPRINT - Script-agnostic Structure Recognition in Tables. SPRINT uses recently introduced Optimized Table Structure Language (OTSL) sequences to predict table structures. We show that when coupled with a pre-trained table grid estimator, SPRINT can improve the overall tree edit distance-based similarity structure scores of tables even for non-English documents. We experimentally evaluate our performance across benchmark TSR datasets including PubTabNet, FinTabNet, and PubTables-1M. Our findings reveal that SPRINT not only matches SOTA models in performance on standard datasets but also demonstrates lower latency. Additionally, SPRINT excels in accurately identifying table structures in non-English documents, surpassing current leading models by showing an absolute average increase of 11.12%. We also present an algorithm for converting valid OTSL predictions into a widely used HTML-based table representation. To encourage further research, we release our code and Multilingual Scanned and Scene Table Structure Recognition Dataset, MUSTARD labeled with OTSL sequences for 1428 tables in thirteen languages encompassing several scripts at https://github.com/IITB-LEAP-OCR/SPRINT.

Keywords: Table Structure Recognition · Layout Detection · Document Analysis

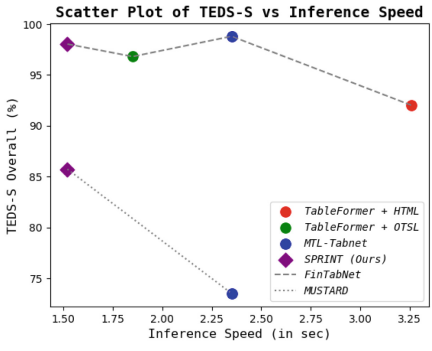
D. Kudale and B. V. Kasuba—These authors contributed equally to this work.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-70549-6_21.

E. H. Barney Smith et al. (Eds.): ICDAR 2024, LNCS 14808, pp. 350–367, 2024.
https://doi.org/10.1007/978-3-031-70549-6_21

1 Introduction

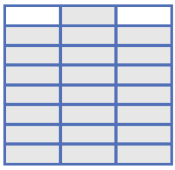
Tables in documents serve as powerful tools for organizing and presenting complex data in a structured and visually comprehensible manner. The structure of tables can vary from simple rows and columns to intricate designs with merged cells and hierarchical arrangements. Additionally, the styling of tables, including multilingual content, font choices, colors, and the presence of borders contribute to the overall diversity of tables. To perform OCR, table reconstruction, or any other document analysis tasks, it is necessary to understand the structure of tables. Depending upon the application and nature of representation, table structures are generally divided into two notions namely physical structure and logical structure. Both these structures are important to be deduced for various downstream applications. The physical structure of the table is denoted by the actual demarcation of table regions like rows, columns, cells, etc. which are generally represented using bounding boxes. The logical structure of a table represents the underlying topology and gives us more information about the cell adjacency relations. It also conveys more about the cells spanned or merged which benefits table reconstruction in the required format. Logical structures of tables are generally represented using HTML, LATEX, or more recently through OTSL [21] sequences. Logical structure prediction is considered to be a sequence generation task. Recently, as seen in Fig 1, many Image-to-sequence-based (Im2Seq) models have been employed to predict the logical structures of tables.



(a) Graph illustrating the comparative TSR performance of SPRINT (our approach) along with other approaches [20–22] for FinTabNet and MUSTARD datasets.

పట్టిక 1 ప్రపంచ వాణిజ్య నౌకారవాణా (అంక్టాడ్ (UNCTAD) నివేదిక 2021)	
ప్రపంచ సముద్ర రవాణా పరిమాణం	10.6 బిలియన్ టన్నులు
ప్రపంచ కంటైనర్ రాకపోకలు	813.5 మిలియన్ 'టీఈయూ'లు
ప్రపంచవ్యాప్త నౌకల సంఖ్య	99,800
(100 కు పైన 'జీఆర్టీ'గల వాణిజ్య నౌకలు)	(డీడబ్ల్యూటీ 213,46,39,907)

(b) Sample table from MUSTARD



(c) Slow, inaccurate TSR using MTL-TabNet [20]

F	L
F	F
F	F
F	F
F	F

(d) Fast, accurate script agnostic TSR by SPRINT

Fig. 1. Recent performance trends of various Im2Seq models for TSR

Im2Seq TSR models are transformer-based models that take a table image as an input and produce a sequence denoting the logical structure of the table.

A lot of popular Im2Seq models [20,39] have adapted Global Context Attention (GCA) [18] in their encoders to boost the TSR performance on popular benchmark datasets. But even though the usage of such GCA-incorporated Im2Seq models like MTL-TabNet [20] have improved the TSR performance by proposing new architectures, they suffer from two major drawbacks: First of all, they are unable to generalize well on tables having content in different scripts. Secondly, they end up using only HTML-based tag sequences for representing table structures. Such drawbacks are highlighted in the graph shown in Fig 1a. MTL-TabNet [20] that achieves the best TSR performance on the FinTabNet dataset is not only relatively slower but also is unable to generalize well on MUSTARD achieving an overall tree edit distance-based similarity structure (TEDS-S) score of under 75%. On the other hand, the OTSL-based table structure representation [21] has proven to be beneficial for faster decoding. It is also evident that using the same TableFormer [22] architecture, the performance is better and faster using OTSL representation as compared to the HTML representation. Most of these Im2Seq models rely on large HTML tag-based vocabularies making them relatively slower. To design script-agnostic TSR, the model must generalize well on diverse data. MUSTARD comprises of diverse tables from different sources. As shown in Fig 1c, MTL-TabNet incorrectly estimates the structure of MUSTARD table shown in Fig 1b by interpreting three columns with empty cells on either extreme side of the first row. This is mainly because SOTA models like MTL-TabNet are trained on upsampled table images that mostly have English content and they implicitly capture some language-specific or font-specific features from the image. Since MUSTARD consists of document tables as well as scene tables in several scripts, sizes, and styles, it becomes challenging to perform script-specific or modality-specific TSR. Moreover, all the above-mentioned conventional approaches require training of heavy DL models that require lots of labeled data. Labeled data for tables in non-English languages is limited. To the best of our knowledge, no labeled datasets are available for tables with content in various scripts with their logical structures annotated.

Thus, we propose SPRINT, for a fast and script-agnostic structure recognition in tables that addresses the above highlighted gaps. Since SPRINT views table structures as a script-agnostic arrangement of cells, it is beneficial to blur the language-specific or font-specific peculiarities in the table images before they are sent to Im2Seq models. We model the global context of such blurred or downsampled images, using GCA-based encoders as used in SOTA. This helps SPRINT to generalize better on tables having content in different scripts. The SPRINT decoder also leverages a minimal OTSL vocabulary rather than a larger error-prone HTML vocabulary [21]. As seen in Fig. 1d, the table structure of the sample image (Fig. 1b) is accurately predicted with two columns and five rows represented using the OTSL sequence. The 'F' in OTSL representation stands for a filled cell and 'L' stands for a left-ward-looking cell indicating a column span of 2 in the first row. Employing OTSL vocabulary helps in faster decoding of structure sequences. Thus, downsampling table images, usage of a GCA-based encoder, and adapting the OTSL-based representation help in

making SPRINT both fast and script agnostic. Through this work, we make the following contributions

1. We present the architecture and design of SPRINT, an Im2Seq model that consists of a GCA-based encoder and a transformer-based decoder that uses the OTSL-based representation for fast and accurate predictions.
2. We propose the usage of SPRINT along with a loosely coupled Table Grid Estimator [29] for deducing the logical structure of tables having content in various scripts.
3. We present an algorithm to convert a well-defined OTSL prediction into an HTML tag-based sequence for accessibility and evaluation purposes. We subsequently highlight the TEDS-S scores achieved by our method for popular TSR datasets.
4. Finally we release our code and MUSTARD, a dataset labeled with OTSL-based sequences to represent the logical structures of 1428 tables having content in several scripts and modalities.

The outline of the rest of the paper is as follows. Section 2 showcases related work being done in the field of TSR. We describe our methodology in Sect. 3. Section 4 gives a detailed overview of our experimentation. We finally present our results in Sect. 5 and conclude in Sect. 6 respectively.

2 Related Literature

TSR is a widely studied problem [1] encompassing various innovative approaches. TSR methods that determine the complete physical and logical structure of tables can be broadly divided into object detection-based methods and Im2Seq-based methods.

Object detection-based methods are predominantly used for deducing the physical structure of tables. They detect and demarcate regions in the input table image. These include popular object detection networks like Faster R-CNN [26], Mask R-CNN [10], Cascade R-CNN [2,23], and YOLO [25] to detect cells. Recently transformer-based architectures like DETR [3] have also been employed for object detection-like tasks that can be used to determine the physical structure of tables. The 'cell bbox decoder' of TableFormer [22], and TATR [29] are some transformer-based approaches that use DETR for determining the cells (as bounding boxes) from detected tables in document images. Various methods like TSRFormer [17], TSR using enhanced DETR [34] and split and merge-based methods [35,40] also follow similar approaches to determine rows and columns of detected tables. Most of the object detection methods predict the physical structure of tables and eventually rely on simple bounding boxes-based post-processing algorithms to associate the detected cells with the logical structure of tables. Hence, these methods can be used in a standalone manner. However, one major drawback of object detection-based methods is their emphasis on enhancing detection performance, which may not necessarily

result in a more effective deduction of the logical structure of tables during post-processing. Further, if the object detection stage performs poorly, the deduced structure is also incorrectly estimated.

Im2Seq-based methods are TSR methods that perform sequence generation from input table images. Transformers have proven to be beneficial for such tasks. Most of these approaches use encoder-decoder-like architectures equipped with attention like the structure decoder of TableFormer [22], EDD [43], etc. TableFormer is an amalgamation of two different models that employ the usage of encoder-decoder with attention [33] for logical structure and DETR [3] for physical structure. The output sequence for such Im2Seq models corresponds to a representation of the logical structure of the table. These sequences can be represented in many forms including HTML, Latex, Markdown, etc. Most of these Im2Seq approaches use the HTML representation for decoding the table structure. There are a lot of drawbacks to using the HTML-based vocabulary [21] as it can be error-prone and unreasonably large. Recently the Optimized Table Structure Language (OTSL) [21] has been introduced to represent tables' logical structure. OTSL is an efficient token-based representation with minimized vocabulary and well-defined rules. Apart from that, the strict syntax and rules to interpret the OTSL structure can also be used to convert it into an HTML sequence. It reduces the number of tokens to be decoded and also helps in faster inference. We exploit this usage of OTSL for predicting the sequence corresponding to the logical structure. Various Im2Seq models like TableMaster [39] end up using pre-existing transformer-based models like MASTER [18]. Im2Seq models used for logical structure prediction are seldom standalone. They end up relying on physical structure predictors for tables in documents to assist in the entire reconstruction of tables.

Hybrid, combined, and miscellaneous works include various new methods like graph-based networks that perform TSR. They usually represent tables as graphs with nodes as table cells, and their edges represent cell relationships by posing this as a graph reconstruction problem. Examples of such approaches include Global Table Extractor (GTE) [42], and Table Graph Reconstruction Network (TGRNet) [37]. Even though graph-based TSR approaches can generalize better, they rely on external graph-based models and require constrained optimizations [15]. Multi-Type-TD-TSR [6] proposes a multi-stage pipeline to perform table detection followed by TSR but processes bordered and borderless tables differently. Some other hybrid approaches of TSR include end-to-end attention-based models [19,20] that try to predict physical structure, logical structure, and cell content using three or more decoders together. Few approaches try to obtain more aligned bounding boxes (physical structure) by utilizing visual information [24,27] or reformulating losses (VAST) [12] to deduce the table structure. Such hybrid works eventually end up working with distinct physical and logical TSR models in synchronization, eliminating the need to explicitly map or align the independently predicted logical and physical structures during the reconstruction stage.

TSR Datasets include a lot of popular huge TSR datasets like PubTabNet [43], FinTabNet [41], SynthTabNet [22], and PubTables-1M [29] that have tables from mostly English documents. TabLeX [5], SciTSR [4] and TableBank [16] are a few datasets of moderate sizes that focus on document tables. Few older and relatively smaller datasets include ICDAR 2013 [8], ICDAR 2019 [7], UNLV [28], etc. All of the datasets mentioned above have data annotated with the logical structure of tables in either HTML or LATEX formats. Recently, a collection of OTSL-based canonical TSR datasets [21] has also been released for a subset of tables in the PubTabNet, FinTabNet, and PubTables-1M datasets which we leverage for training SPRINT. TabRecSet [38] is a recently released bilingual end-to-end table detection, TSR, and table content recognition dataset with tables annotated from English and Chinese documents. However, to the best of our knowledge, there are no multilingual TSR datasets. Taking inspiration from multilingual datasets proposed for text detection [14] and text recognition [9], we propose MUSTARD having structures annotated for various multilingual tables. Apart from that, a lot of metrics have been used to evaluate TSR models including Mean Average Precision (MAP) for physical structure, TEDS-S score for logical structure, and TEDS score [43], GriTS [31] score for both logical as well as physical structure.

3 Our Methodology

Our objective is to determine the structure of the input table image. Figure 2 highlights the design of our proposed methodology.

The first step involves interpreting the logical structure of the input table. This makes use of SPRINT, the working and training details of which we explain in Sect. 4.3. The next step involves interpreting the physical structure of the table by demarcating the rows and columns of the input table image. The goal is to identify the number of objects for two classes namely 'table-row' and 'table-column'. We estimate the number of rows (R) and number of columns (C) in this step. The technical details of the table grid estimator are mentioned in Sect. 4.4. Since OTSL has a well-defined syntax associated with the table structure, it is necessary to convert the predicted string into a valid OTSL matrix. For a table with R rows and C columns, a valid OTSL matrix has $R * (C + 1)$ entries. The estimated number of rows and the number of columns obtained in Step 2 are used to verify the length of the predicted string. The string is made to have a length of $R * (C + 1)$ by appropriate padding or trimming techniques to reshape it into a proper OTSL matrix as shown in the output of Step 3 in Fig 2. The periodicity of character 'N' is also verified by making sure that every $(C + 1)^{th}$ character is 'N'. Similarly, incorrectly placed 'L', 'U', 'X', and 'N' characters are replaced with 'F' denoting a fundamental table cell. This grid-based alignment step is beneficial for two reasons. First of all, it produces a syntactically valid OTSL matrix that can be converted to other popular formats like HTML. Secondly, it eliminates the need to explicitly map the cells of the table with a logical structure through extensive post-processing. As the aligned OTSL sequence and

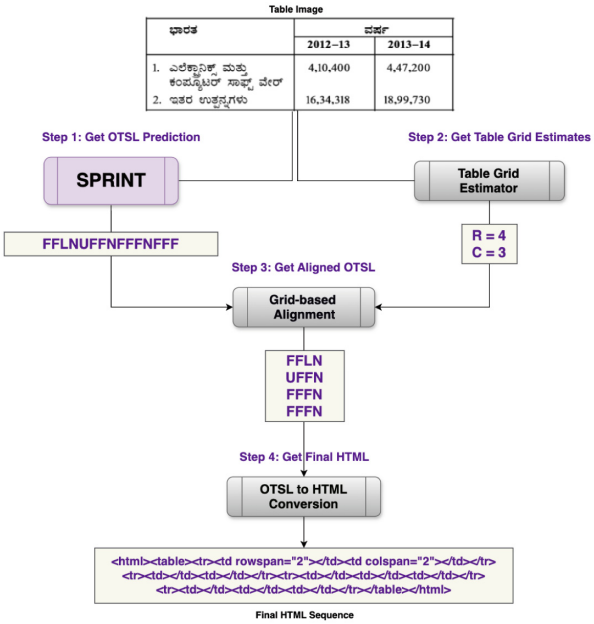


Fig. 2. Our methodology for TSR using SPRINT

the estimated grid indicate the same number of rows, columns, and cells, every cell is implicitly mapped with a certain row and column by default. Finally, in the last step, the aligned and validated OTSL matrix is converted into an HTML sequence using the procedure mentioned in Algorithm 1. The intuition for the intermediate method for finding the cell spans of a single entry in the OTSL matrix is shown in Algorithm 2. Eventually, we determine the tree-like representation of the table's structure which can be used for further evaluation and reconstruction.

4 Experiments

We begin by mentioning the OTSL-based datasets and then elaborating upon the architecture of SPRINT and usage of TATR [29] as our table grid estimator that is used to deduce the logical structure of the input table image.

Algorithm 1: Generate HTML tag sequence from a valid OTSL matrix

Data: R: Number of rows, C: Number of columns, otsl_matrix: Valid sequence of OTSL string converted into a 2-D matrix of dimensions $R*(C+1)$
Result: html_string: HTML string representing a table

html_string ← $<html><table><tbody>$
for *i ← 0 to R - 1* **do**
  html_string += $<tr>$
  for *j ← 0 to C* **do**
    entry ← otsl_matrix[i][j]
    if *entry is 'E' or entry is 'F'* **then**
      rs, cs ← get_cell_spans(otsl_matrix, i, j)
      if *rs != 0 and cs != 0* **then**
        html_string += $<td\ rowspan="\{cs+1\}"colspan="\{rs+1\}"></td>$
      end
      else if *rs != 0 and cs == 0* **then**
        html_string += $<td\ colspan="\{rs+1\}"></td>$
      end
      else if *rs == 0 and cs != 0* **then**
        html_string += $<td\ rowspan="\{cs+1\}"></td>$
      end
      else
        html_string += $<td></td>$
      end
    end
    else if *entry is 'N'* **then**
      html_string += $</tr>$
    end
  end
end
html_string += $</tbody></table></html>$
return *html_string*

Algorithm 2: Get cell spans for a corresponding entry in OTSL matrix

Data: otsl_matrix: Matrix of OTSL sequence, i: Row index, j: Column index
Result: rs: Row span, cs: Column span

entry ← *otsl_matrix[i][j]*
if *entry is not 'E' or entry is not 'F'* **then**
  return *0, 0*
end
else
  row_seq ← Concatenate characters in *otsl_matrix[i]* slicing from *j + 1*
  col_seq ← Join characters in column 'j' of *otsl_matrix* slicing from *i + 1*
  rs ← Count of contiguous occurrences of 'L' in *row_seq* after *entry*
  cs ← Count of contiguous occurrences of 'U' in *col_seq* below *entry*
  return *rs, cs*
end

4.1 Datasets

We have used popular benchmark TSR datasets of PubTabNet [43], FinTabNet [41] and PubTables-1M [29] which contain tables predominantly from English documents. The canonical subsets of table images from these datasets have their corresponding OTSL sequences released [21] which we use for training and validation purposes. We internally split the train set of PubTabNet for training and validation and further have used the non-overlapping table images in the PubTabNet validation set to report our results to compare our performance with other approaches. Table 1 gives a brief idea of the datasets used and the number of images in every split for our experimentation. More details and OTSL-character-specific statistics are mentioned in the supplementary material. To ensure consistency, we use the test sets of canonical datasets to compare with OTSL baselines and that of original datasets to compare with HTML baselines. The further bifurcation of the test sets in terms of the types of tables (simple or complex) is also enlisted. Simple tables do not have any spanned or merged cells, whereas complex tables have at least one merged or spanned cell in them. Besides, we also report results on our internal dataset, MUSTARD. MUSTARD consists of 1428 tables in thirteen languages that are cropped and annotated from multiple sources [13,32,38] that originally contain page-level images. MUSTARD encompasses 1214 cropped document table images (scanned or printed) in twelve languages, including eleven Indian languages [13], each with approximately 100 tables, namely Assamese, Bengali, Gujarati, Hindi, Kannada, Malayalam, Oriya, Punjabi, Tamil, Telugu, Urdu, and an additional 102 Chinese cropped tables sourced from CTDAR documents [7,32]. MUSTARD also includes 214 English and Chinese scene tables that are cropped and annotated

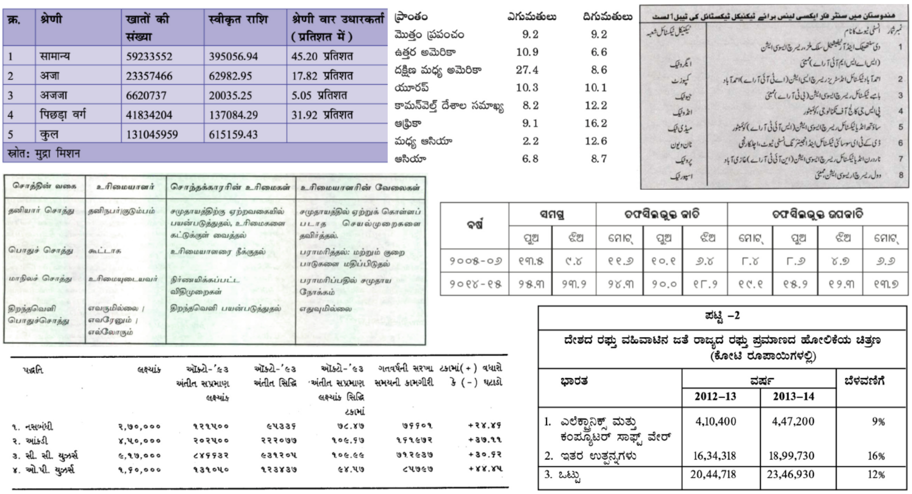


Fig. 3. Sample images from our internal dataset, MUSTARD

from a subset of images present in the TabRecSet [38] dataset. A few of the table images from MUSTARD are highlighted in Fig 3.

Table 1. Overview of the TSR Datasets used in our experimentation. * indicates evaluating on the non-overlapping images in the PubTabNet validation set

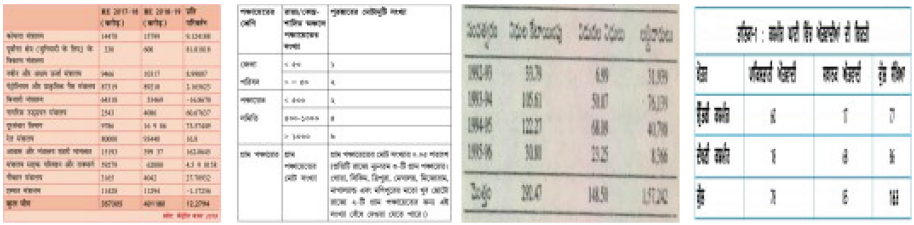
Dataset Name		Number of Images			Test Set Details	
		Training	Validation	Testing	Simple	Complex
PubTabNet [43]	Original	320000	68002	*9115	4653	4462
	Canonical [21]			*6942	4636	2306
FinTabNet [41]	Original	88441	10505	10635	5126	5509
	Canonical [21]			10397	5126	5271
PubTables-1M	Canonical [21]	522874	93989	92841	44377	48464
MUSTARD		-	-	1428	662	766

4.2 Image Preprocessing

We resize all the table images in our datasets to a standard size. Unlike the popular approaches [12,20,22], which upscale the image or pass it to the model as it is, we choose to downsample the images into 128∗128 pixels. This preprocessing is done for two main reasons. First of all, it introduces uniformity for the model and generates features of equal dimensions for all images. Secondly, resizing helps to convert the content of table cells in the images into blobs of pixels. These blobs are sufficient enough to convey the presence of some data in the cell and create distortions to ensure that the script-based peculiarities are blurred. This helps to achieve a table image with blobs of pixels representing a script-agnostic arrangement of cells. As seen in Fig 4, we see the resized table images originally having content in different languages reduced to a size of $128 * 128$ pixels having blobs. By standardizing these images, SPRINT becomes more robust and adaptable to comprehend different tabular structures.

4.3 Training SPRINT

We employ resized images of $128 * 128$ pixels for all subsequent experiments. Our vocabulary comprises six OTSL characters, namely 'F', 'E', 'L', 'X', 'N', and 'U', and *start* and *stop* tokens added manually to the sequences before training. More details about the syntax of representing table structure using OTSL [21] sequences and how we have adapted it in SPRINT are given in the supplementary material. For every sample, the table image serves as input, with the OTSL sequence as the target (ground truth). As shown in Fig. 5, SPRINT comprises a GCA-based encoder and a transformer-based decoder. We use Multi Aspect Global Attention [18] fused between RESNET31 [11] layers in



(a) Hindi (b) Bengali (c) Telugu & English (d) Punjabi

Fig. 4. Resized table images in different languages after preprocessing stage

the encoder. The encoder generates a feature tensor that denotes 512 channels of $16 * 32$ feature maps. This feature tensor undergoes positional embedding to generate encoded feature vectors each of size 512 which are passed on to the decoder. The decoder comprises six layers and a feed-forward neural network with 2048 nodes in intermediate layers. As most OTSL sequences in our datasets are less than 224 characters in length, we set the maximum permissible length of predictions as 224 for the decoder. We use the categorical cross-entropy loss between the ground truths and predicted OTSL sequences to train SPRINT using the above-mentioned configuration. One is trained solely on the FinTabNet dataset which we refer to as SPRINT_{FTN}. The other model, SPRINT_{ALL}, is trained on a merged dataset comprising FinTabNet, PubTabNet, and PubTables-1M. Both models undergo training for over 80 epochs with a learning rate of 0.0001. We have performed all our training and relevant experiments on an NVIDIA RTX A6000 single GPU.

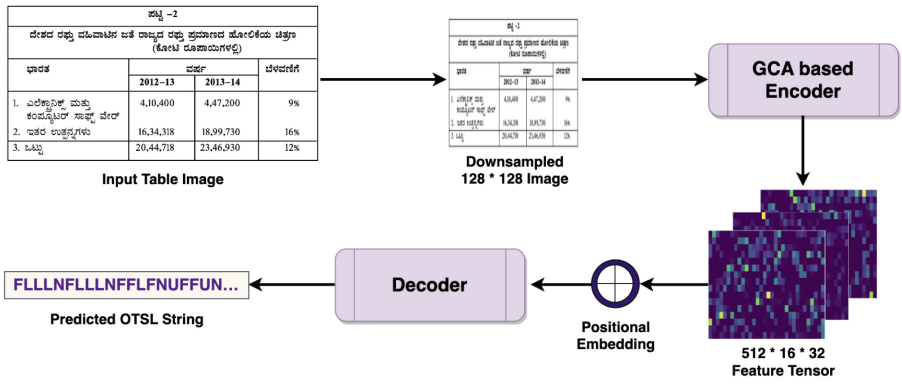


Fig. 5. Architecture diagram explaining the structure and working of SPRINT

4.4 Table Grid Estimator

We have used the TATR [30] V1.1 model pretrained on FinTabNet, PubTabNet, and PubTables-1M datasets for determining the physical structure of the input table image. The underlying architecture for TATR is DETR [3] which is trained for six object classes. For grid alignment, we make use of two classes of 'table-row' and 'table-column' as they are sufficient to accurately determine the number of rows (R) and columns (C). We use a detection threshold of 0.25 during the inference of the TATR V1.1 model. We further carry out NMS for table row class with an IOU threshold of 0.25 to minimize overlapping predictions and improve the row match. Once R and C are determined, these values align with the OTSL sequence predicted by SPRINT. The supplementary material shows the qualitative results of the TATR model for detecting table rows and table columns respectively. Eventually, it is the number of detections that are used to determine the number of rows (R) and columns (C) to align the OTSL sequence. The actual bounding boxes are useful only for further reconstruction of tables. However, only R and C values are sufficient to align the OTSL sequence and deduce the logical structure. To show that, pretrained TATR is a promising grid estimator to determine the number of rows and columns, we present the exact match (in %) and mean L1 error in estimating the number of rows, number of columns, and both rows and columns together for the different test sets in Table 2. As shown in the table, for all datasets under consideration, there is a highly accurate match for PubTabNet, FinTabNet, and PubTables-1M datasets. There is a decent match of rows and columns for MUSTARD because TATR is pretrained from abundant tables from only English documents. Even though the match in the number of rows and columns for MUSTARD has scope for improvement, the average L1 error in predicted numbers and actual numbers is less than 0.55 indicating that the difference in the predicted and actual rows is quite low in number. This low L1 error ensures that the TEDS-S score is not penalized much for the HTML sequence generated which is evident in our results in Sect. 5. The better the exact match accuracy, the better would be the aligned OTSL and thus more accurately would be the table structure preserved.

Table 2. Performance of TATR [29] as our table grid estimator for determining the number of rows and columns in the image. We use v1.1-PubTables-1 m for reporting results on the PubTables-1M dataset and v1.1-all on the other datasets.

Test Dataset	Rows Only		Columns Only		Rows and Cols	
	Exact Match	Avg L1 Error	Exact Match	Avg L1 Error	Exact Match	Avg L1 Error
PubTabNet	88.82	0.278	90.97	0.138	81.28	0.208
FinTabNet	88.51	0.212	98.42	0.019	87.22	0.115
PubTables-1M	94.39	0.123	98.19	0.022	92.81	0.073
MUSTARD	67.67	0.576	75.00	0.451	54.62	0.513

5 Results and Discussions

In this section, we present the TEDS-S scores achieved by SPRINT along with the assistance of the table grid estimator. The final output is an HTML tag sequence. PubTabNet, FinTabNet, and PubTables-1M have their ground truths available in HTML format. To evaluate only the structure, we filter the content and only consider the pure HTML tag sequence as the corresponding ground truths. The OTSL-labeled sequences of the MUSTARD are converted to HTML tag sequences using Algorithm 1.

5.1 Performance on TSR in English Documents

Table 3 highlights our performance against the pre-reported TEDS-S scores of the OTSL baseline [21] for all three datasets. We have reported our results on the canonical test sets of FinTabNet, PubTabNet, and PubTables-1M [21]. We have used SPRINT_{FTN} to evaluate the FinTabNet canonical test set and SPRINT_{ALL} on the other two test sets respectively. We consistently perform better than the OTSL baseline for all the enlisted datasets. We compare our performance with the pre-reported TEDS-S scores of different approaches that use the HTML-based vocabulary for TSR tasks in Table 4. We use SPRINT_{FTN} on FinTabNet and SPRINT_{ALL} on the PubTabNet test set for evaluation. We match the performance of the best performing MTL-TabNet [20] for both datasets. Our approach falls short by an average of 1.5% because both VAST and MTL-TabNet rely on upscaling images leading to larger feature maps that are leveraged by cascaded decoders trained on the HTML-based vocabulary. Further, MTL-TabNet uses two decoders for decoding the table structure making the model slower than our approach. After testing for 50 iterations on a random subset of 400 FinTabNet images, it is observed that while MTL-TabNet takes an average of *2.35 s* per prediction, SPRINT can give faster prediction in an average time of *1.52 s* per image indicating better latency. We give a detailed overview of inference time estimation in the supplementary material. We are faster not only due to the small-sized feature tensors produced by downsampled images by the GCA-based encoder [18] but also due to the usage of minimized OTSL vocabulary [21] for the decoder to produce faster predictions. Besides, our approach also generalizes well for tables having content in other languages (scripts) as described in the upcoming Sect. 5.2.

Table 3. Comparison of our approach against the OTSL baseline on popular TSR datasets for tables in English documents. * indicates that OTSL [21] have reported percentage score rounded off to one digit after decimal point

Dataset	TableFormer + OTSL [21]			Ours		
	TEDS-S Simple	TEDS-S Complex	TEDS-S Overall	TEDS-S Simple	TEDS-S Complex	TEDS-S Overall
PubTabNet	96.50	93.40	95.50	**98.20**	**96.24**	**97.55**
FinTabNet	95.50	96.10	95.90	**98.36**	**97.99**	**98.17**
PubTables-1M	98.70	96.40	***97.70**	**98.92**	**96.54**	**97.68**

Table 4. Comparison of our approach on PubTabNet and FinTabNet datasets for tables in English documents with other pre-reported TEDS-S scores of HTML-based Im2Seq approaches

Dataset	PubTabNet [43]			FinTabNet [41]		
Model	TEDS-S Simple	TEDS-S Complex	TEDS-S Overall	TEDS-S Simple	TEDS-S Complex	TEDS-S Overall
EDD [36,43]	91.10	88.70	89.90	88.40	92.08	90.60
GTE [42]	-	-	93.01	-	-	91.02
TableFormer [22]	98.50	95.00	96.75	97.50	96.00	96.80
VAST [12]	-	-	97.23	-	-	98.63
MTL-TabNet [20]	**99.05**	**96.66**	**97.88**	**99.07**	**98.46**	**98.79**
Ours	98.00	93.32	95.71	98.35	97.74	98.03

5.2 Performance on MUSTARD

Table 5 shows the comparative overview of our performance against MTL-TabNet [20]. Since source code and checkpoints for OTSL baselines [21] and VAST [12] have not been released, we compare the TEDS-S scores obtained by our approach with the MTL-TabNet (SOTA) scores. For determining TEDS-S on MUSTARD, we use the MTL-TabNet checkpoint trained on the FinTabNet dataset and SPRINT_{FTN} for evaluating our approach. Our approach consistently performs better than MTL-TabNet for all the enlisted languages and shows an average increase of *11.12%* in the overall TEDS-S score. We believe that MTL-TabNet does not perform as well as our approach since it has been extensively trained on English TSR datasets for upsampled images on HTML vocabulary. As a result, MTL-TabNet is unable to capture the script-agnostic arrangement of cells. Fig 6 showcases our results on a few images from MUSTARD which show our aligned OTSL predictions projected on the HTML-based table skeleton reflecting the ground truth structures.

Table 5. Comparing MTL-TabNet [20] with our approach on tables in MUSTARD for various languages (scripts) and modalities

Modality	Language	MTL-TabNet			Ours		
		TEDS-S Simple	TEDS-S Complex	TEDS-S Overall	TEDS-S Simple	TEDS-S Complex	TEDS-S Overall
Document Tables (Printed and Scanned)	Assamese	79.39	73.40	76.54	**88.09**	**88.74**	**88.40**
	Bengali	71.68	60.02	61.42	**77.24**	**78.52**	**78.36**
	Gujarati	85.12	76.72	79.63	**87.79**	**81.34**	**83.58**
	Hindi	73.80	76.60	75.04	**85.68**	**88.22**	**86.81**
	Kannada	68.82	66.73	67.20	**71.84**	**79.02**	**77.34**
	Malayalam	82.57	79.34	81.07	**86.41**	**85.13**	**85.81**
	Oriya	85.28	78.03	82.84	**91.55**	**85.20**	**89.41**
	Punjabi	65.08	48.63	51.54	**86.91**	**79.65**	**80.93**
	Tamil	81.96	71.88	77.83	**94.91**	**85.87**	**91.21**
	Telugu	85.07	79.28	82.17	**93.70**	**86.00**	**89.85**
	Urdu	70.94	69.74	70.03	**81.39**	**75.38**	**76.86**
	Chinese	92.43	81.58	86.15	**98.11**	**86.00**	**91.10**
Scene Tables	English	76.19	78.01	76.53	**88.98**	**76.14**	**85.71**
	Chinese	69.40	66.65	68.94	**88.62**	**81.96**	**87.27**
Overall		77.70	71.90	74.07	**87.23**	**82.66**	**85.19**

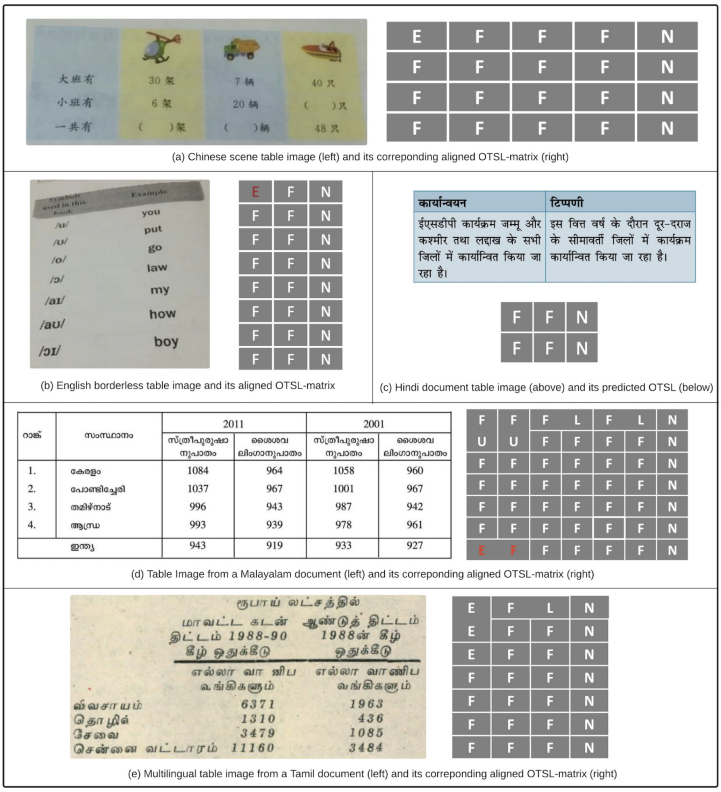


Fig. 6. Each subfigure showcases the final aligned OTSL matrix predicted by SPRINT_{FTN} adjacent to the input table image. The OTSL matrix is projected on an HTML-based table skeleton reflecting the ground truth structure. The red characters denote incorrect OTSL prediction that alters TEDS-S score

6 Conclusion

We present SPRINT as a solution for fast, robust and script-agnostic TSR. It achieves this via the downsampling of input images, GCA-based encoder, and OTSL-based table structure representation. The minimized OTSL vocabulary not only helps in faster decoding but also in conversions to more popular formats like HTML making the scope of this task more extensive and versatile. We also present an algorithm to convert the OTSL sequence to an HTML-based representation of the table structure. Finally, we release MUSTARD which has been meticulously annotated using OTSL sequences, opening new doors for further research in script-agnostic TSR. In the future, we want to investigate incorporating cells detected using the table grid estimator with the predicted logical structure, to reconstruct complex tables in documents as accurately as possible. Along with the help of table detection models and powerful OCR-based

frameworks, we can design end-to-end pipelines that reconstruct the structure and text inside a wide variety of tables having content in different scripts.

Acknowledgement. We acknowledge the support of a grant from IRCC, IIT Bombay, and MEITY, Government of India, through the National Language Translation Mission-Bhashini project.

References

1. Ajayi, K., Choudhury, M.H., Rajtmajer, S.M., Wu, J.: A study on reproducibility and replicability of table structure recognition methods. In: Fink, G.A., Jain, R., Kise, K., Zanibbi, R. (eds.) Document Analysis and Recognition - ICDAR 2023, pp. 3–19. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-41679-8_1
2. Cai, Z., Vasconcelos, N.: Cascade R-CNN: delving into high quality object detection (2017)
3. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers (2020)
4. Chi, Z., Huang, H., Xu, H.D., Yu, H., Yin, W., Mao, X.L.: Complicated table structure recognition (2019)
5. Desai, H., Kayal, P., Singh, M.: TabLeX: A benchmark dataset for structure and content information extraction from scientific tables. In: Lladós, J., Lopresti, D., Uchida, S. (eds.) ICDAR 2021. LNCS, vol. 12822, pp. 554–569. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86331-9_36
6. Fischer, P., Smajic, A., Abrami, G., Mehler, A.: Multi-Type-TD-TSR – extracting tables from document images using a multi-stage pipeline for table detection and table structure recognition: from OCR to structured table representations. In: Edelkamp, S., Möller, R., Rueckert, E. (eds.) KI 2021. LNCS (LNAI), vol. 12873, pp. 95–108. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-87626-5_8
7. Gao, L., et al.: ICDAR 2019 competition on table detection and recognition (CTDAR). In: 2019 International Conference on Document Analysis and Recognition (ICDAR), pp. 1510–1515 (2019). https://doi.org/10.1109/ICDAR.2019.00243
8. Göbel, M.C., Hassan, T., Oro, E., Orsi, G.: ICDAR 2013 table competition. In: 2013 12th International Conference on Document Analysis and Recognition, pp. 1449–1453 (2013). https://api.semanticscholar.org/CorpusID:206777311
9. Gongidi, S., Jawahar, C.V.: iiit-indic-hw-words: A dataset for Indic handwritten text recognition. In: Lladós, J., Lopresti, D., Uchida, S. (eds.) ICDAR 2021. LNCS, vol. 12824, pp. 444–459. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86337-1_30
10. He, K., Gkioxari, G., Dollár, P., Girshick, R.B.: Mask R-CNN. CoRR **abs/1703.06870**arXiv:1703.06870 (2017)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (2015)
12. Huang, Y., et al.: Improving table structure recognition with visual-alignment sequential coordinate modeling (2023)
13. Ministry of Information & Broadcasting, Goverment of India: Yojana Archives (2023). https://www.publicationsdivision.nic.in/journals/index.php?route=page/archives

14. Kudale, D., Kasuba, B.V., Subramanian, V., Chaudhuri, P., Ramakrishnan, G.: TEXTRON: weakly supervised multilingual text detection through data programming (2024)
15. Lee, E., Park, J., Koo, H.I., Cho, N.I.: Deep-learning and graph-based approach to table structure recognition. Multimedia Tools Appl. **81**(4), 5827–5848 (2022). https://doi.org/10.1007/s11042-021-11819-7
16. Li, M., Cui, L., Huang, S., Wei, F., Zhou, M., Li, Z.: TableBank: a benchmark dataset for table detection and recognition (2020)
17. Lin, W., et al.: TSRFormer: table structure recognition with transformers (2022). https://doi.org/10.48550/arXiv.2208.04921
18. Lu, N., et al.: MASTER: multi-aspect non-local network for scene text recognition. Pattern Recogn. **117**, 107980 (2021). https://doi.org/10.1016/j.patcog.2021.107980
19. Ly, N.T., Takasu, A.: An end-to-end local attention based model for table recognition. In: Fink, G.A., Jain, R., Kise, K., Zanibbi, R. (eds.) Document Analysis and Recognition - ICDAR 2023, pp. 20–36. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-41679-8_2
20. Ly, N.T., Takasu, A.: An end-to-end multi-task learning model for image-based table recognition, pp. 626–634 (2023). https://doi.org/10.5220/0011685000003417
21. Lysak, M., Nassar, A., Livathinos, N., Auer, C., Staar, P.: Optimized table tokenization for table structure recognition. In: Fink, G.A., Jain, R., Kise, K., Zanibbi, R. (eds.) Document Analysis and Recognition - ICDAR 2023, pp. 37–50. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-41679-8_3
22. Nassar, A., Livathinos, N., Lysak, M., Staar, P.: TableFormer: table structure understanding with transformers. arXiv preprint arXiv:2203.01017 (2022)
23. Prasad, D., Gadpal, A., Kapadni, K., Visave, M., Sultanpure, K.: CascadeTabNet: an approach for end to end table detection and structure recognition from image-based documents (2020)
24. Qiao, L., et al.: LGPMA: complicated table structure recognition with local and global pyramid mask alignment (2022)
25. Redmon, J., Divvala, S.K., Girshick, R.B., Farhadi, A.: You only look once: unified, real-time object detection. CoRR **abs/1506.02640**arXiv:1506.02640 (2015)
26. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. CoRR **abs/1506.01497**arXiv:1506.01497 (2015)
27. Raja, S., Mondal, A., Jawahar, C.V.: Table structure recognition using top-down and bottom-up cues. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12373, pp. 70–86. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58604-1_5
28. Shahab, A., Shafait, F., Kieninger, T., Dengel, A.: An open approach towards the benchmarking of table structure recognition systems, pp. 113–120 (2010). https://doi.org/10.1145/1815330.1815345
29. Smock, B., Pesala, R., Abraham, R.: PubTables-1M: towards comprehensive table extraction from unstructured documents. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4634–4642 (2022)
30. Smock, B., Pesala, R., Abraham, R.: Aligning benchmark datasets for table structure recognition (2023)
31. Smock, B., Pesala, R., Abraham, R.: GriTs: grid table similarity metric for table structure recognition (2023)

32. ICDAR 2019 Competition on Table Detection and Recognition (2019). https://cndplab-founder.github.io/cTDaR2019/index.html
33. Vaswani, A., et al.: Attention is all you need (2023)
34. Wang, J., et al.: Robust table structure recognition with dynamic queries enhanced detection transformer. Pattern Recogn. **144**, 109817 (2023). https://doi.org/10.1016/j.patcog.2023.109817, https://www.sciencedirect.com/science/article/pii/S0031320323005150
35. Xiao, B., Simsek, M., Kantarci, B., Alkheir, A.: Table structure recognition with conditional attention (2022)
36. Xiao, B., Simsek, M., Kantarci, B., Alkheir, A.A.: Rethinking detection based table structure recognition for visually rich documents (2023)
37. Xue, W., Yu, B., Wang, W., Tao, D., Li, Q.: TGRNet: a table graph reconstruction network for table structure recognition (2021)
38. Yang, F., Hu, L., Liu, X., Huang, S., Gu, Z.: A large-scale dataset for end-to-end table recognition in the wild. Sci. Data **10**(1), 110 (2023)
39. Ye, J., et al.: PingAn-VCGroup's solution for ICDAR 2021 competition on scientific literature parsing task B: table recognition to HTML (2021)
40. Zhang, Z., Zhang, J., Du, J., Wang, F.: Split, embed and merge: an accurate table structure recognizer. Pattern Recogn. **126**, 108565 (2022). https://doi.org/10.1016/j.patcog.2022.108565, https://www.sciencedirect.com/science/article/pii/S0031320322000462
41. Zheng, X., Burdick, D., Popa, L., Zhong, P., Wang, N.X.R.: Global table extractor (GTE): a framework for joint table identification and cell structure recognition using visual context. In: Winter Conference for Applications in Computer Vision (WACV) (2021)
42. Zheng, X., Burdick, D., Popa, L., Zhong, X., Wang, N.X.R.: Global table extractor (GTE): a framework for joint table identification and cell structure recognition using visual context (2020)
43. Zhong, X., ShafieiBavani, E., Jimeno-Yepes, A.: Image-based table recognition: data, model, and evaluation. CoRR **abs/1911.10683**arXiv:1911.10683 (2019)