Discovery of Structured Data Using Unsupervised Spatial Clustering and Human Supervision

Nikitha Rachapudi, Lakshmipathy Ganesh, Abinaya Sekar, Anand KS, and Rajkumar Sakthibalan

***Abstract*—Commercial data has been preserved digitally in portable document format (PDF) for its ease of encapsulating multiple data formats. In this digitization era, there comes a need to capture and store this data in structured format to facilitate its access for automated b2b services and business intelligence. In this paper, we propose a framework that automates discovery and extraction of tabular data incorporating both artificial and human intelligence. The framework involves clustering and heuristics to group cartesian location of text and spaces in a page to determine a table. The discovered table is then validated by the user using a user-interface designed to moderate the determined boundaries and fed back to the layout knowledge repository. The table data obtained is extracted as JSON key-value pairs which can then be loaded into any database. The framework thus provides enhanced accuracy and continuous human assisted learning for the automated document digitization process. The knowledge repository is further used to train the machine to generate document templates to be used for processing unseen documents. However, this paper concentrates on the discovery of structured data alone.**

***Index Terms*—Clustering algorithm, spatial analysis, pdf table extraction, heuristics, human interaction.**

I. Introduction

Over the years with the evolution of technology, important data once stored in paper is now being stored digitally. PDF is looked as the most viable solution as it is independent of any hardware and operating system; and can be transported and accessed over any channel. [1] Educational institutes, government agencies, research centers, construction firms are a few major organizations where PDFs have become indispensable [2].

Due to the high availability of PDF documents, there is a huge scope to mine intelligence from them to make better business decisions. This necessitates the automation of the data-entry process of searching values from individual PDFs and storing them into databases. This ensures an error-free and speedy process, reducing the burden on the process staff, thereby reducing the cost invested by many folds. The data generated can directly be consumed to find patterns or solutions for many business problems.

In this paper, we assume that a major chunk of significant data would be stored as tables [3], as it is one of the best ways to communicate complex details to the human mind. Therefore, we focus on automating the extraction of tabular data and converting them to key-value pairs which can then be consumed by any application. We also believe that the machine would learn better by active learning i.e. with the help of little human intervention [4]. Hence, we provide a user interface, where the user can verify if the tables detected automatically falls within his scope of requirement. Otherwise, the user can manually mark his selection and this information would be fed back to the machine, for future training.

This framework considers a PDF document as a cartesian plane, on which lies the box coordinate of every word of the page. We map the human-eye table detection process to create rules that define a cluster of coordinates as fa potential table. Our algorithm works without considering the horizontal and vertical runs, which are usually used to identify a table in many of the current solutions. This approach also makes the data extraction process independent of any language. Thus, our framework also works for documents that do not have the horizontal and vertical lines bounding a table. Our framework creates templates with a large set of similar PDFs. These templates can then be accessed directly to extract the desired data from PDF documents with similar layout.

Here we explain the basic working of our system with a single PDF document, which personifies any PDF sent into the system.

II. Related Work

Extracting data from tables has been a key topic of research. Many researches have made use of algorithms to define building blocks and find the tables based on the horizontal and vertical lines detection. Yildiz *et al.* [5] developed a heuristic approach for detecting tables in PDF files and store the extracted data in a structured data format (XML) for reuse. This was also implemented with a GUI that gives the user the ability to adjust the detected tables. The work shows that purely heuristic-based approaches can achieve good results, especially for lucid tables. Davulcu *et al.* [6] proposed a clustering technique where each word is considered as tokens and the building blocks are generated based on the distance between the clusters and assigning the tokens to the respective clusters. Also, the user has to specify the header name of the table to extract the table. This method doesn't do great in detecting header less tables.

Hassan *et al*. [7] proposed a method for detecting tables in PDF files. They group tables into three categories: tables with both horizontal and vertical ruling lines, tables which contain only horizontal ruling lines, and tables where ruling lines are absent. Babatunde *et al.* [8] proposed an HMM based method to recognize table and extract data from the HTML tags. Oro *et al.* [9] proposed a table detection software called

Manuscript received May 23, 2019; revised July 3, 2019.

The authors are with CloudIX Inc, Suite 301, 15446, Bel Red Road, Redmond, WA 94052 USA (e-mail: {nikhitha, ganeshl, abinayas, anandks, rajs}@cloudix.io).

doi: 10.18178/ijmlc.2019.9.5.844

PDF-TREX. The PDF-TREX system employs a heuristic approach (experience based) for table detection and structure definition. However, the system has the following limitations: 1) Only single column documents are supported 2) Ruling lines or other visual aids on the page are not considered.

Our paper attempts to enhance the existing methods by using clustering techniques along with our heuristic rules and facilitates a feedback mechanism to the system for an iterative improvement in the overall system accuracy.

III. System Architecture

The system we propose involves recognition of tables by grouping the words in the document, represented as 2D coordinates on a Cartesian plane. The coordinates of the words are generated using a pdf to xml converter. These coordinates are then used for detecting and locating tables. Fig. 1 depicts the overall architecture of our system and Fig. 2 explains in detail about the automated table extraction process.

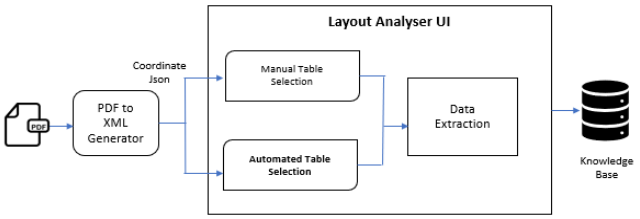

Fig. 1. Overall architecture.

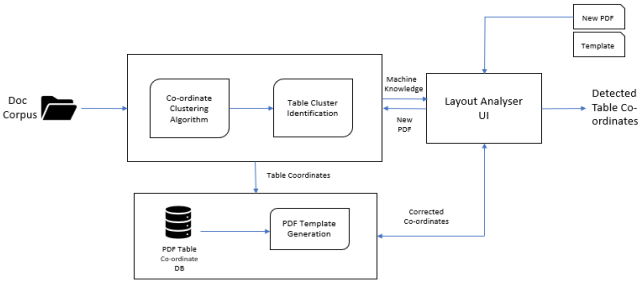

Fig. 2. Automated extraction flow.

A. Overall Setup

The Document Corpus we consider for this paper includes documents with tabular data published as PDF documents. The preliminary step involves figure out the coordinates of each word in the PDF by representing them in a 2D Cartesian plane. To achieve this, we use the Text and Image Extraction Toolkit (TET) [10]. TET gives the text contents of a PDF as Unicode strings, detailed color, font information as well as the position on the page.

TET acts as a PDF to XML generator since TET converts PDF documents to an XML-based format called TETML which contains text, page metadata as well as resource information. The XML tags containing the metadata of the words, color, glyph and text positions are recorded as part of the data for future work. The text positions are captured in Points (pts), as the unit of measurement.

These features are used throughout the table detection process, template generation and knowledge extraction. The system involves a GUI which shows all the words in the document and on clicking the word, will take the user to the exact page and highlights for instant visibility. This gives a quick access to key words the user might be interested in looking at.

The document for which the data is to be extracted is converted to image and displayed to the user along with the mapping of the table regions on the image. The GUI also has an option where the user is given the ability to adjust the tables, we have identified through heuristic clustering and classification algorithm.

We enable the user with two options. The user can choose one of these options to select the table regions:

- Manual table selection
- Automated table selection

The manual table Selection gives full control to the users to select the table regions by click and drag action. To facilitate this, we use a third-party library called Annotorious [11] which allows the users to draw rectangular annotations over PDF pages or edit an existing rectangular annotation or even delete one. Once the right tables are selected by the user, the coordinate layout is recorded and is stored as a template containing the coordinates for tables of that PDF document. Data in the table selected by the user is also extracted and stored. The layout knowledge base gets updated with the new metadata of tables and the data in the tables is stored as JSON key-value pairs in the target database. This facilitates the user to simply drag through the table regions and record this data, without having to read the document and type.

The automated table selection method automatically identifies the potential table regions in the PDF document using the coordinate clustering algorithm and table cluster identification process, which are discussed in detail in the next section. The identified potential table regions are then displayed to the user.

B. Brief Overview of Automated Table Detection Process

Once the coordinates of the words are grouped into clusters by their distance or density, they are fed into a heuristic classification algorithm to find out the potential table clusters among all the clusters that have been identified.

The algorithm considers features that we have defined for a potential table like: the alignment of words, repetition of words with the same alignment over successive rows; definitive line or column spacings; cartesian distance between successive words that distinguishes a paragraph from a table.

With the help of these features, the algorithm gives a binary output to decide if the cluster can potentially be a table. Once the algorithm finds the possible table clusters, we determine its boundary coordinates and display it to the user through the GUI to make any adjustments if the table region predicted is not accurate. Once the user makes the necessary adjustments to the table boundaries, the boundary coordinates are stored in the PDF table coordinate DB. The data in those table regions are extracted as key-value pairs and stored to the knowledge base.

The key distinguishing feature of our system from other existing systems lies in the fact that the user has more control in figuring out the right tables by adjusting the tables identified through the Automated table selection method. The user adjusted table metadata with the corrected coordinates of table regions are saved as a new PDF template for that PDF. Thus, the existing layout database gets updated with the new

metadata of tables. Once the tables are identified, the metadata is recorded in the PDF template generation process. The generated template contains the following data: PDFId of the document, Page Number, ClusterID, Cluster coordinates: upper left x and y coordinates, cluster width and height for all the clusters that were identified as potential tables.

The layout information of the PDF documents thus deduced, are preserved as a function of the documents publishing context, represented by document metadata like publisher, title, published date, author and copyright information, publishing software subject to availability. This context metadata is fed forward as features for predicting layouts of unseen documents. This process gets repeated for every new PDF uploaded for extraction and the algorithm auto learns the possible tables.

IV. Table Detection Algorithm

For detecting tables in every page, we use machine learning algorithms along with a set of predefined rules.

We define the spatial reference of a word-box as {llx, lly, urx, ury} where (llx, lly) and (urx, ury) are lower left and upper right cartesian coordinates. We compute (midx, midy) as the average of the lower and upper box coordinates. We also compute the area of a word-box using the cartesian formula for a rectangle. Fig. 3 shows a plot of a llx vs lly of a single page. This shows that the alignment of the words is neatly captured by plotting them using just one set of cartesian points. We iterate the below process for every page of a PDF using its PDFId and page number.

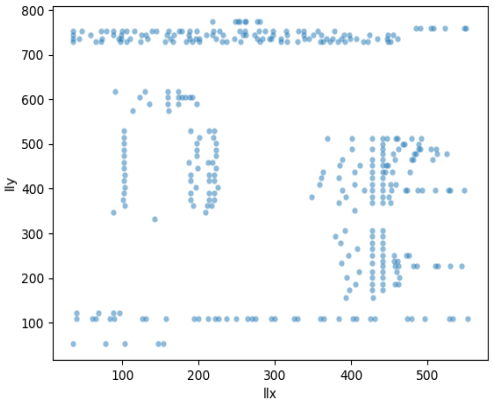

Fig. 3. Cartesian plane with word coordinates.

A. Coordinate Clustering

In this algorithm, we use existing clustering techniques to group the word coordinates as table or paragraph (non-table) clusters. Tables that spread across the width of the page, have columns that are spaced farther than that of the columns of a smaller table.

K-means algorithm [12] which works using centroid and Euclidean distance works good on closely spaced points whereas DBSCAN [13] which uses density for clustering captures widely spread data points which are spaced systematically.

To encompass accurate table detection from different page formats, we use an approach to automate the clustering technique selection. We calculate the distance between words in a line and look at their distribution spread. When the second quartile (50th percentile) and the 3rd quartile (75th percentile) are almost the same, we pass the coordinates of that page to K-means. When the distribution drastically changes, i.e. the 2nd and 3rd quartile vary by a huge margin, we use DBSCAN as the clustering technique. This ensures that table data are clustered accurately in both single and double columned documents.

We use the Silhouette method to interpret the optimum number of clusters to be derived from K-means technique. For DBSCAN we set the hyper parameter epsilon to specify how close points must be to each other to be considered as a part of cluster. We consider that, to define a cluster as a table, there must be a minimum of 5 points in it i.e. a table header, two column headers and two column text words. Fig. 4 and Fig. 5 show a plot of the different clusters identified by K-means and DBSCAN respectively.

With the different sets of clusters obtained, we have to classify each of them as

- A potential table cluster vs a non-table cluster
- Eliminate embedded clusters
- Discard non-table data in a cluster identified as a table cluster.

B. Identification of Potential Table Clusters

For every cluster identified, iterate the below process for defining them as a potential table cluster,

1. Subset the {llx, lly} coordinate of every word-box, and arrange both in ascending order.
2. For every x-coordinate find how many y-coordinates exist.
3. If there are a minimum of 2 y-coordinates for a given x-coordinate, we can assume it to be a potential column of a table, which is left aligned.

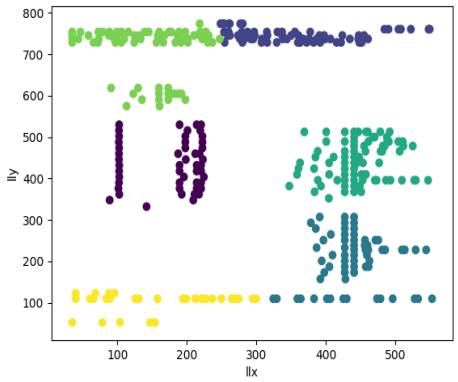

Fig. 4. Clusters identified by K-means.

Repeat the same procedure to identify right aligned columns using {urx, ury} and centrally aligned columns as {Midx, Midy}. If the cluster contains at least 2 columns, we can define it as a potential table cluster.

C. Eliminating Embedded Clusters

It is possible that clustering algorithms used above, detect a small table cluster inside {ulx, uly} a bigger table cluster on the outside {ulx, uly} as in Fig 6, and hence there comes a need to eliminate the smaller table. For this we iterate the below process for every cluster identified:

if ($ulx_i > ulx_o$) &/or ($uly_i > uly_o$)

if ($width_i < width_o$) &/or ($height_i < height_o$)

mark the table inside {ulx,uly} as a non-table cluster

This eliminates the duplication of data being retrieved.

D. Discard non-Table Data in a Table Cluster

Paragraph lines close to a table gets appended to a table cluster. These lines are usually at the top or bottom end of the clustered rows. So, it is essential that we eliminate these non-table rows. Spacing information within these cluster data can again be used to discard the paragraph lines.

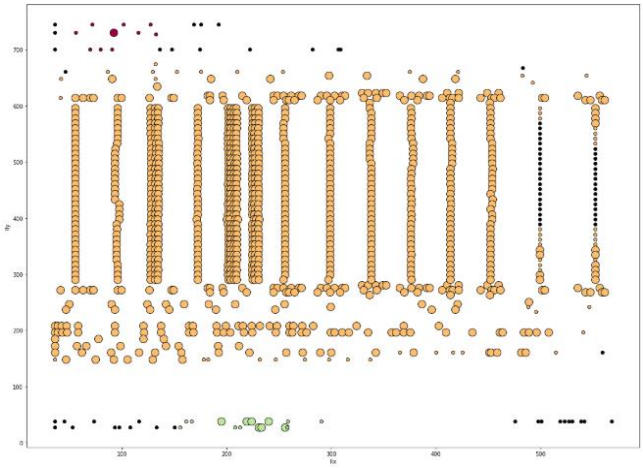

Fig. 5. Clusters identified by DBSCAN.

For every table cluster identified, iterate the below process for discarding the non-table lines,

1. Check if there is any word in every line having llx or ulx or Midx, with the identified columns' llx or ulx or Midx respectively.
2. If not, verify if the distance between two consecutive words in the line is within a threshold of the minimum word distance in each pdf. If yes, mark it as a non-table line and make sure that the line is at the ends of the clustered lines before discarding it.
3. Retain the rest of the lines.
4. If there is any cluster found with just one line after discarding non-table lines, mark that cluster as a non-table cluster.

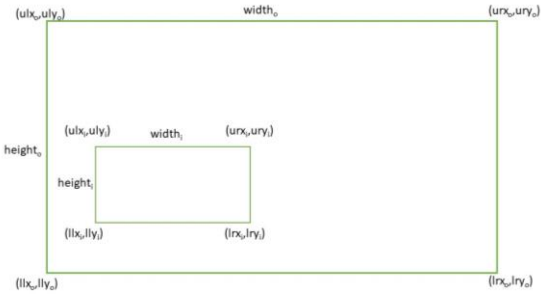

Fig. 6. Embedded cluster.

This process also eliminates the NOTE paragraph that is often seen below a table. For the table clusters thus identified, we calculate the coordinates of the rectangular boundary around it. We define {ulx, uly} of this cluster as {minimum of ulx, minimum of uly} of all the words in that cluster. {llx, lly} of the cluster as {maximum of lrx, maximum of lry}. Using these derived coordinates, we then compute the cluster's width, height and total area. The area occupied by the word boxes are also computed as a sum of all the box areas.

Further tuning of the table identification algorithm would be taken care of, when a user manually adjusts the table selection in the UI.

V. Experimentation and Discussion

Since our algorithm is page specific, we evaluate multiple pages of PDFs with different table structures, individually. Furthermore, our evaluation is limited to validation of the coordinate clustering and table identification algorithm. In total, we evaluated 20 pages, with around 25 tables. Classification of clusters as tables or non-tables, being a binary classification problem, we use the confusion matrix as the algorithm evaluation metrics as shown in Fig. 7.

	Predicted Positive	Predicted Negative
Actual Positive	Tables identified as tables (TP)	Tables identified as non-tables (FN)
Actual Negative	Non-tables identified as tables(FP)	Non-tables identified as non-tables(TN)

Fig. 7. Algorithm evaluation metrics.

For this dataset, 20 table clusters were classified correctly as tables, and 6 non-table clusters were classified as tables. The classification algorithm missed to identify 4 tables as tables and on the other hand identified 51 non-table clusters as non-tables. An accuracy of 89.88% was observed as the average success rate across every page. With an average Precision of 82.45% our algorithm' prediction of a table to be table is precise 8 out of 10 times. Also, the heuristic classification algorithm has an average of 89.47% recognition rate of a table cluster, with an average F1 measure of 83.15 % for detecting tables as in Table I.

TABLE I: Evaluation Metrics

Metric	Precision	Recall	F1-measure
Values (%)	82.45	89.47	83.1

The algorithm fails to identify small tables embedded between paragraph lines as tables. Also, images in a page, with structured marking of points, are classified as table regions. We also came across PDFs having multi-line wrapped text in columns of a table. The wrapped text was clustered as individual non-table clusters. A rule to merge such wrapped text clusters to a single table cluster would solve this issue. The difference in the font size of columns headers of a table to that of normal text can also be capitalized as a feature to mark table clusters, along with the spatial information in future work. Also, the area occupied by word boxes to that of the total area of the cluster can be used as features to enhance the detection process in the future.

We believe that with human intervention to this algorithm using the user interface, the accuracy would improve over time. As more PDFs with similar format are added to the training corpus, a much higher classification rate can be achieved and would facilitate in template creation.

VI. Extension Work

A. Identifying Rows and Columns Using Deep Learning

Once the table boundaries in a document are identified, it is essential that we capture the rows and columns information

to gather the data in them in a structured format. This is a complex problem to solve using traditional heuristic methods or feature-based machine learning models. This is because there are different layouts to stack the individual rows and columns in a tabular structure. Technical documents like component datasheets and financial reports have varying tables like simple descriptive tables storing key-value pairs and pivoted tables having aggregated financial values interspersed between every few rows.

To encompass the various dimensionality of the structure of the tables, the latest Deep Learning techniques shall be used on the spatial distribution of words between rows and columns of boundaries. We can enhance the novel deep learning-based approach for table structure detection, DeepDeSRT [14], as proposed by Sebastian Schreiber *et al.* The PDF documents can be converted to image format. In addition to the preprocessing technique(s) mentioned by Sebastian *et al.*, further processing can be done by creating Local Binary Patterns (LBP) [15], [16] of the image. An LBP image details minute edges, corners and flat pixels of the image more accurately, thereby improving the pixel detection and classification performance.

LBP along with the ground truth annotation dataset of row-column boundaries of the tables can then be processed using Single Shot Multibox Detector (SSD) [17] or a Faster-RCNN model [18]. Heuristics on the probabilities returned on unseen data can be employed to retain the most accurately detected rows and columns. Once the physical boundaries of the rows and columns are defined using the bounding boxes from the Deep Learning model, we can extract the text data in that region from the original TETML file. The data derived is now in a structured format which can be used for further processing.

B. Table Classification Using Deep Learning Models

The structured data derived from various tables should then be stored in their respective information systems. Tables in a document might or might not have a Table Title to define the data it contains. Hence there comes a need to add domain-specific knowledge to the derived data corpus. We can assign a table title tag to each of the derived tables and perform a classification task for predicting table titles in new documents. Various Deep Learning based Natural Language Processing (NLP) techniques can be employed to facilitate this.

Different Deep Learning models have proven very effective in the NLP tasks with advancements in the optimized Deep learning architectures (CNN's, RNN's, etc.) [19]. These models require a good amount of training data for accurate predictions. The training data for the modelling can be prepared by tagging as many tables as possible in the documents using a tagger. This can be done by tagging the tables into appropriate categories, with specific tags for specific tables. With the tagged training data, the neural networks can be trained and then be used to identify distinct tags on new unseen tables. Once the new documents are tagged by the model, the derived data can be stored in information systems.

VII. Conclusion

With an estimated 2.5 trillion PDFs created each year [20], the usage of PDFs has only been growing. With such a rich pool of digital data, in this analytics age, there is a huge need of information to be derived from them for data-oriented decision making. But the process has almost always been cumbersome, involving a lot of man-hours, making it an expensive affair.

Hence there comes a need to automate the process of extracting specific details from these documents and store them in a more intelligible format. In this paper, we proposed a system to select the required content manually or automatically detect the tables. Furthermore, the data from these tables are sent back to the user as JSON key-value pairs.

With larger data corpus we cluster identified tables of a page to create PDF template. This facilitates the user to send a new document to directly fetch the required knowledge, skipping the selection process.

The automatic table detection process goes through a predefined set of rules using the spatial spread of the words, which are used to label clusters of words as a table or non-table cluster. Thus, future work could focus on automating the cluster classification by creating table specific features in addition to spatial spread. This can also be auto-tuned, as the machine learns with more data in live usage.

References

[1] PDF file format accessibility features combined with Adobe® Acrobat® and Adobe Reader® allow universal access to documents. [Online]. Available: https://www.adobe.com/accessibility/pdf/pdf-accessibility-overview.html

[2] Adobe PDF and printing. [Online]. Available: https://web.archive.org/web/20160404222250/https://www.adobe.com/products/postscript/pdf.html

[3] pdf2table: A Method to Extract Table Information from PDF Files. [Online]. Available: http://ieg.ifs.tuwien.ac.at/pub/yildiz_iicai_2005.pdf

[4] Active Learning Literature Survey. [Online]. Available: http://burrsettles.com/pub/settles.activelearning.pdf

[5] B. Yildiz, K. Kaiser, and S. Miksch, "pdf2table: A method to extract table information from PDF files," in *Proc. the 2nd Indian International Conference on Artificial Intelligence*, Pune, December 20-22, 2005, pp. 1773-1785.

[6] H. Davulcu, S. Mukherjee, and I. V. Ramakrishnan, "A clustering technique for mining data from text tables," in *Proc. the 2002 SIAM International Conference on Data Mining*, 2002, pp. 315-332.

[7] T. Hassan and R. Baumgartner, "Table recognition and understanding from pdf files," in *Proc. ICDAR*, 2007, pp. 1143-1147.

[8] F. F. Babatunde, B. A. Ojokoh, and S. A. Oluwadare, "Automatic table recognition and extraction from heterogeneous documents," *Journal of Computer and Communications*, vol. 3. pp. 100-110, 2015.

[9] E. Oro and M. Ruffolo, "PDF-TREX: An approach for recognizing and extracting tables from PDF documents," in *Proc. ICDAR 2009*, 2009, pp. 906-910.

[10] TET. [Online]. Available: https://www.pdflib.com/products/tet/

[11] Annotorious. [Online]. Available: https://annotorious.github.io/

[12] T. Kanungo, D. M. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, pp. 881-892, 2002.

[13] M. Ester, H. P. Krieger, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. the 2nd International Conference on Knowledge Discovery and Data Mining*, Portland, OR, AAAI Press, 1996, pp. 226-231.

[14] S. Schreiber, S. Agne, I. Wolf, A. Dengel, and S. Ahmed, "DeepDeSRT: Deep Learning for Detection and Structure Recognition of Tables in Document Images," in *Proc. 14th IAPR International Conference on Document Analysis and Recognition*.

[15] T. Ojala, M. Pietikäinen, and D. Harwood, "Performance evaluation of texture measures with classification based on Kullback discrimination

of distributions," in *Proc. the 12th IAPR International Conference on Pattern Recognition*, 1994, vol. 1, pp. 582-585.
[16] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on feature distributions," *Pattern Recognition*, vol. 29, pp. 51-59, 1996.
[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," *Ser. Lecture Notes in Computer Science*, vol. 9905, pp. 21–37, 2016.
[18] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster RCNN: Towards real-time object detection with region proposal networks," *Microsoft Research, Tech. Rep*., 2015.
[19] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing", *IEEE Computational Intelligence Magazine*, vol. 13, no. 3, pp. 55-75, August 2018.
[20] PDF in 2016: Broader, deeper, richer. [Online]. Available: https://www.pdfa.org/pdf-in-2016-broader-deeper-richer

Nikitha Rachapudi is a data scientist at CloudIX. She holds a post graduate degree in business analytics, and a bachelor's degree from Anna University, Chennai. She has worked on proving analytics solutions for multiple domains including insurance, telecommunication and manufacturing. She is an algorithm enthusiast who has keen interest for research in the areas of deep learning and artificial intelligence.

Lakshmipathy Ganesh is a data scientist at CloudIX. He holds a bachelor's degree from Anna University, Chennai, India. He has worked with healthcare, manufacturing and operations domains with strong foundations in big data, analytics and operations research. His interest lies in building conversational bots, AI and deep learning.

Abinaya Sekar is a software developer for CloudIX. She holds a bachelor's degree from Anna University, Chennai. She specializes in application development, client/server and NoSQL database modeling. She is particularly interested in cognitive services, BOT technology and cloud services. She is always keen to be part of challenging projects.

Anand KS is a UI developer, at CloudIX. He holds a bachelor's degree from Anna University. He is a passionate UI developer who enjoys turning complex problems into simple, beautiful and intuitive interface designs. He has 3 years of commercial experience as a UI developer, providing responsive and interactive front-end development, and has specialized in HTML, CSS, react JS, JavaScript and react native.

Rajkumar Sakthibalan is a strategic technology research and digital transformation consultant at CloudIX. He holds a post graduate diploma in bioinformatics from Institute of Genomics and Integrative Biology, Delhi, India and a bachelor's degree from University of Madras, Chennai, India. He provides digital transformation consulting, technology road mapping, research consulting in the areas of industry 4.0, bots and AI.