Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 1918–1925
Marseille, 11–16 May 2020

TableBank: Table Benchmark for Image-based Table Detection and Recognition

Minghao Li[1], Lei Cui[2], Shaohan Huang[2], Furu Wei[2], Ming Zhou[2], Zhoujun Li[1]
[1]State Key Lab of Software Development Environment, Beihang University, Beijing, China
[2]Microsoft Research Asia, Beijing, China
{liminghao1630, lizj}@buaa.edu.cn, {lecu, shaohanh, fuwei, mingzhou}@microsoft.com

Abstract

We present TableBank, a new image-based table detection and recognition dataset built with novel weak supervision from Word and Latex documents on the internet. Existing research for image-based table detection and recognition usually fine-tunes pre-trained models on out-of-domain data with a few thousand human-labeled examples, which is difficult to generalize on real-world applications. With TableBank that contains 417K high quality labeled tables, we build several strong baselines using state-of-the-art models with deep neural networks. We make TableBank publicly available and hope it will empower more deep learning approaches in the table detection and recognition task. The dataset and models can be downloaded from `https://github.com/doc-analysis/TableBank`.

Keywords: TableBank, table detection and recognition, weak supervision, image-based deep learning network

1. Introduction

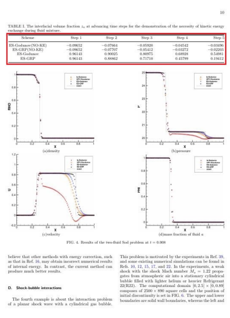
Table detection and recognition is an important task in many document analysis applications as tables often present essential information in a structured way. It is a difficult problem due to varying layouts and formats of the tables as shown in Figure 1. Conventional techniques for table analysis have been proposed based on the layout analysis of documents. Most of these techniques fail to generalize because they rely on handcrafted features which are not robust to layout variations. Recently, the rapid development of deep learning in computer vision has significantly boosted the data-driven image-based approaches for table analysis. The advantage of the image-based table analysis lies in its robustness to document types, making no assumption of whether scanned images of pages or natively-digital document formats. It can be applied to a wide variety of document types to extract tables including PDF, HTML, PowerPoint as well as their scanned copies. Although some document types may have structured tabular data, we still need a general approach to detect tables on different kinds of documents. Therefore, large-scale end-to-end deep learning models make it feasible for achieving better performance.

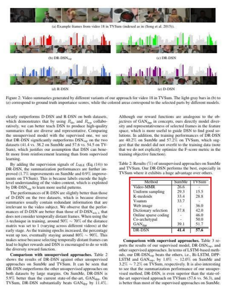
Existing deep learning based table analysis models usually fine-tune pre-trained object detection models with several thousand human-labeled training instances, while still difficult to scale up in real-world applications. For instance, we found that models trained on data similar to Figure 1a 1b 1c would not perform well on Figure 1d because the table layouts and colors are so different. Therefore, enlarging the training data should be the only way to build open-domain table analysis models with deep learning. Deep learning models are massively more complex than most traditional models, where many standard deep learning models today have hundreds of millions of free parameters and require considerably more labeled training data. In practice, the cost and inflexibility of hand-labeling such training sets is the key bottleneck to actually deploying deep learning models. It is well known that ImageNet (Russakovsky et al., 2015) and COCO (Lin et al., 2014) are two popular image classification and object detection datasets that are built in a crowdsourcing way, while they are both expensive and time-consuming to create, taking months or years for large benchmark sets. Fortunately, there exists a large number of digital documents on the internet such as Word and Latex source files. It is instrumental if some weak supervision can be applied to these online documents for labeling tables.

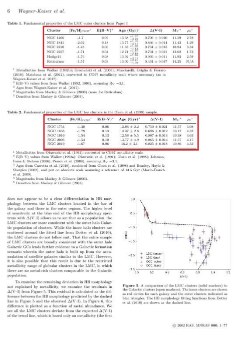
To address the need for a standard open-domain table benchmark dataset, we propose a novel weak supervision approach to automatically create the TableBank, which is orders of magnitude larger than existing human-labeled datasets for table analysis. Distinct from the traditional weakly supervised training set, our approach can obtain not only large scale but also high quality training data. Nowadays, there are a great number of electronic documents on the web such as Microsoft Word (.docx) and Latex (.tex) files. These online documents contain mark-up tags for tables in their source code by nature. Intuitively, we can manipulate these source code by adding bounding boxes using the mark-up language within each document. For Word documents, the internal Office XML code can be modified where the borderline of each table is identified. For Latex documents, the tex code can be also modified where bounding boxes of tables are recognized. In this way, high quality labeled data is created for a variety of domains such as business documents, official filings, research papers, etc, which is tremendously beneficial for large-scale table analysis tasks.

The TableBank dataset totally consists of 417,234 high quality labeled tables as well as their original documents in a variety of domains. To verify the effectiveness of TableBank, we build several strong baselines using state-of-the-art models with end-to-end deep neural networks. The table detection model is based on the Faster R-CNN (Ren et al., 2015) architecture with different settings. The ta-

The paper was finished while the first author was visiting MSRA.


(a)


(b)


(c)

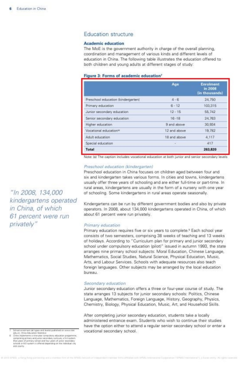

(d)

Figure 1: Tables in electronic documents on the web with different layouts and formats. Among them, Figure (a)(b)(c) come from the latex format documents while Figure (d) comes from a word format document

ble structure recognition model is based on the encoder-decoder framework for image-to-text. The experiment results show that the layout and format variation has a great impact on the accuracy of table analysis tasks. In addition, models trained on one specific domain do not perform well on the other. This suggests that there is plenty of room for advancement in modeling and learning on the TableBank dataset.

2. Existing Datasets

We introduce some existing publicly available datasets that we compare as the baselines:

ICDAR 2013 Table Competition. The ICDAR 2013 Table Competition dataset (Göbel et al., 2013) contains 128 examples in natively-digital document format, which are from the European Union and US Government.

UNLV Table Dataset. The UNLV Table Dataset (Shahab et al., 2010) contains 427 examples in scanned image format, which are from a variety of sources including Magazines, Newspapers, Business Letter, Annual Report, etc.

Marmot Dataset. The Marmot Dataset[1] contains 2,000 pages in PDF format, where most of the examples are from research papers.

DeepFigures Dataset. The DeepFigures Dataset (Siegel et al., 2018) includes documents with tables and figures from arXiv.com and PubMed database. The DeepFigures Dataset focuses on the large scale table/figure detection task while it does not contain the table structure recognition dataset.

3. Data Collection

Basically, we create the TableBank dataset using two different file types: Word documents and Latex documents. Both file types contain mark-up tags for tables in their source code by nature. Next, we introduce the details in three steps: document acquisition, creating table detection dataset and table structure recognition dataset.

[1]`http://www.icst.pku.edu.cn/cpdp/data/marmot_data.htm`

3.1. Document Acquisition

We crawl Word documents from the internet. The documents are all in '.docx' format since we can edit the internal Office XML code to add bounding boxes. Since we do not filter the document language, the Word documents contain English, Chinese, Japanese, Arabic, and other languages. This makes the dataset more diverse and robust to real applications.

Latex documents are different from Word documents because they need other resources to be compiled into PDF files. Therefore, we cannot only crawl the '.tex' file from the internet. Instead, we use documents from the largest pre-print database arXiv.org as well as their source code. We download the Latex source code from 2014 to 2018 through bulk data access in arXiv. The language of the Latex documents is mainly English.

3.2. Table Detection

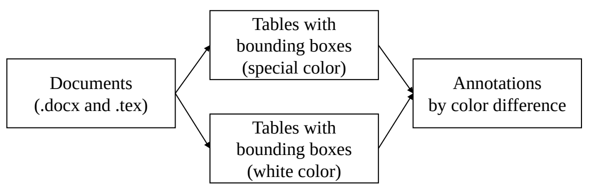

Documents (.docx and .tex)
Tables with bounding boxes (special color)
Tables with bounding boxes (white color)
Annotations by color difference

Figure 2: Data processing pipeline

Intuitively, we can manipulate the source code by adding bounding boxes using the mark-up language within each document. The processing pipeline is shown in Figure 2. For Word documents, we add the bounding boxes to tables through editing the internal Office XML code within each document. Actually, each '.docx' file is a compressed archive file. There exists a 'document.xml' file in the decompressed folder of the '.docx' file. In the XML file, the code snippets between the tags '<w:tbl>' and '</w:tbl>' usually represent a table in the Word document, which is shown in Figure 3. We modified the code snippets in the XML file so that the table borders can be changed into a distinguishable color compared to other parts of the document. Figure 3 shows that we have added a green bounding box in the PDF file where the table is perfectly identified.

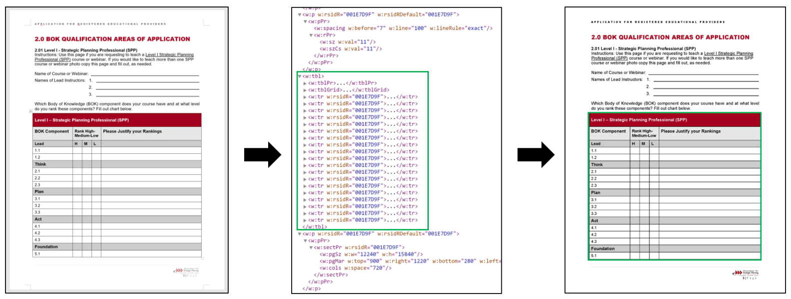


Figure 3: The tables can be identified and labeled from "<w:tbl>" and "</w:tbl>" tags in the Office XML code

Finally, we get PDF pages from Word documents that contain at least one table on each page.
For Latex documents, we use a special command in the Tex grammar 'fcolorbox' to add the bounding box to tables. Typically, the tables in Latex documents are usually in the format as follows:

```
\begin{table}[]
        \centering
        \begin{tabular}{}
                ...
        \end{tabular}
\end{table}
```

We insert the 'fcolorbox' command to the table's code as follows and re-compile the Latex documents. Meanwhile, we also define a special color so that the border is distinguishable. The overall process is similar to Word documents. Finally, we get PDF pages from Latex documents that contain at least one table on each page.

```
\begin{table}[]
        \centering
        \setlength{\fboxsep}{1pt}
        \fcolorbox{bordercolor}{white}{
        \begin{tabular}{}
                ...
        \end{tabular}}
\end{table}
```

To obtain ground truth labels, we extract the annotations from the generated PDF documents. For each page in the annotated PDF documents, we also add the bounding boxes to the tables in the original page with the white color so that the two pages align in the same position. Then, we simply compare two pages in the pixel-level so that we can find the different pixels and get the upper left point of the tables as well as the width and height parameters. In this way, we totally create 417,234 labeled tables from the crawled documents. We randomly sample 1,000 examples from the dataset and manually check the bounding boxes of tables. We observe that only 5 of them are incorrectly labeled, which demonstrates the high quality of this dataset. We give several typical weak supervised labeled errors in Figure 4.

3.3. Table Structure Recognition

Table structure recognition aims to identify the row and column layout structure for the tables especially in non-digital document formats such as scanned images. Existing table structure recognition models usually identify the layout information as well as the textual content of the cells, while textual content recognition is not the focus of this work. Therefore, we define the task as follows: given a table in the image format, generating an HTML tag sequence that represents the arrangement of rows and columns as well as the type of table cells. In this way, we can automatically create the structure recognition dataset from the source code of Word and Latex documents. For Word documents, we simply transform the original XML information from documents into the HTML tag sequence. For Latex documents, we first use the LaTeXML toolkit[2] to generate XML from Latex, then transform the XML into HTML. A simple example is shown in Figure 5, where we use '<cell_y>' to denotes the cells with content and '<cell_n>' to represent the cells without content. After filtering the noise, we create a total of 145,463 training instances from the Word and Latex documents.
In the process of inferring, we get the row and column structures from table structure recognition results, as well as the content and bounding boxes of all the closely arranged text blocks from OCR results. Assuming a table contains N rows, row gaps between OCR bounding boxes are detected by a heuristic algorithm to split the table content blocks into N groups. Finally, we fill the blocks in the '<cell_y>' tag of N lines with the order from left to right. For example, the table in Figure 5 has two rows according to table structure recognition result, e.g. the HTML tag sequence. According to the row gap between text block "1" and text block "3", we divide them into two groups: $\langle 1, 2\rangle$ and $\langle 3\rangle$. The first group corresponds the first row containing two '<cell_y>' tags, and the second group corresponds the second row containing a '<cell_y>' tag and a '<cell_n>' tag.

4. Baseline

4.1. Table Detection

We use the Faster R-CNN model as the baseline. Due to the success of ImageNet and COCO competition, the Faster R-CNN model has been the de facto implementation that is widespread in the computer vision area. In 2013, the R-CNN model (Girshick et al., 2013) was first proposed to solve the object detection problem. After that, (Girshick,

[2]`https://dlmf.nist.gov/LaTeXML/`

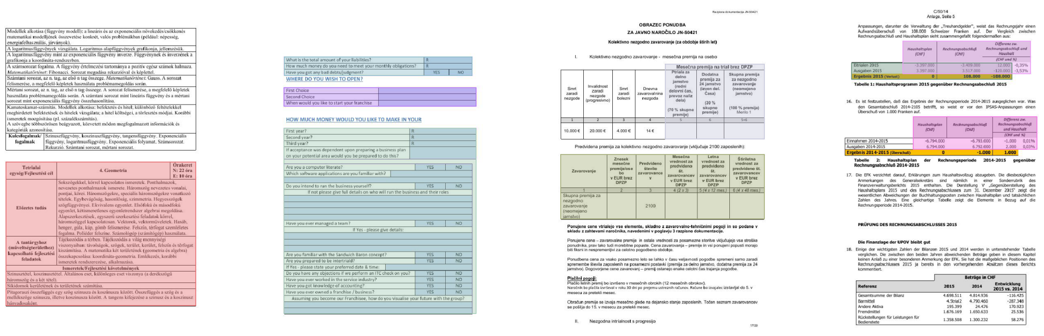
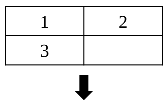


Figure 4: Figure (a): Miss a table; Figure (b): Two tables are annotated as one; Figure (c): One of the annotations is incomplete; Figure (c): Some annotations are overlapped.



<tabular> <tr> <cell_y> <cell_y> </tr> <tr> <cell_y> <cell_n> </tr> </tabular>

Figure 5: A Table-to-HTML example where '<cell_y>' denotes cells with content while '<cell_n>' represents cells without content

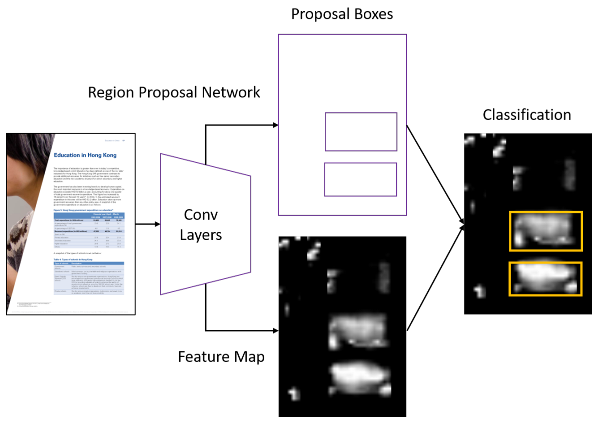


Figure 6: The Faster R-CNN model for table detection

2015) also proposed the Fast R-CNN model that improves training and testing speed while also increasing detection accuracy. Both two models use selective search to find out the region proposals, while selective search is a slow and time-consuming process affecting the performance of the network. Distinct from two previous methods, the Faster R-CNN method introduces a Region Proposal Network (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. Furthermore, the model merges RPN and Fast R-CNN into a single network by sharing their convolutional features so that the network can be trained in an end-to-end way. The overall architecture of Faster R-CNN in shown in Figure 6.

4.2. Table Structure Recognition

We leverage the image-to-text model as the baseline. The image-to-text model has been widely used in image captioning, video description, and many other applications. A typical image-to-text model includes an encoder for the image input and a decoder for the text output. In this work, we use the image-to-markup model (Deng et al., 2016) as the baseline to train models on the TableBank dataset. The overall architecture of the image-to-text model is shown in Figure 7.

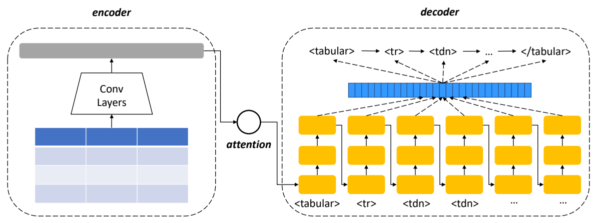


Figure 7: Image-to-Text model for table structure recognition

5. Experiment

5.1. Data and Metrics

The statistics of TableBank is shown in Table 1. To evaluate table detection, we sample 2,000 document images from Word and Latex documents respectively, where 1,000 images for validation and 1,000 images for testing. Each sampled image contains at least one table. Meanwhile, we also evaluate our model on the ICDAR 2013 dataset to verify the effectiveness of TableBank. To evaluate table structure recognition, we sample 500 tables each for validation and testing from Word documents and Latex documents respectively. The size of the training set for table detection and table structure recognition is 415,234 and 144,463 respectively. The entire training and testing data have been made available to the public. For table detection, we calculate the precision, recall, and F1 in the same way as in (Gilani et al., 2017), where the metrics for all documents are computed by summing up the area of overlap, prediction and ground truth. The definition is defined as follows:

$$Precision = \frac{\text{Area of Ground truth regions in Detected regions}}{\text{Area of all Detected table regions}},$$

$$Recall = \frac{\text{Area of Ground truth regions in Detected regions}}{\text{Area of all Ground truth table regions}},$$

$$F1\ Score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

For table structure recognition, we use the 4-gram BLEU score as the evaluation metric with a single reference. The 4-gram BLEU score has the advantages of fast calculation speed, low cost, and being easy to understand.

Task	Word	Latex	Word+Latex
Table detection	163,417	253,817	417,234
Table structure recognition	56,866	88,597	145,463

Table 1: Statistics of TableBank

5.2. Settings

For table detection, we use the open source framework Detectron (Girshick et al., 2018) to train models on the TableBank. Detectron is a high quality and high-performance codebase for object detection research, which supports many state-of-the-art algorithms. It is written in Python and powered by the Caffe2 deep learning framework. In this task, we use the Faster R-CNN algorithm with the ResNeXt (Xie et al., 2016) as the backbone network architecture, where the parameters are pre-trained on the ImageNet dataset. All baselines are trained using 4×P100 NVIDIA GPUs using data parallel sync SGD with a minibatch size of 16 images. For other parameters, we use the default values in Detectron. During testing, the confidence threshold of generating bounding boxes is set to 90%.

For table structure recognition, we use the open source framework OpenNMT (Klein et al., 2017) to train the image-to-text model. OpenNMT is an open-source ecosystem for neural machine translation and neural sequence learning, which supports many encoder-decoder frameworks. In this task, we train our model using the image-to-text method in OpenNMT. The model is also trained using 4×P100 NVIDIA GPUs with the learning rate of 0.1 and batch size of 24. In this task, the vocabulary size of the output space is small, including <tabular>, </tabular>, <thead>, </thead>, <tbody>, </tbody>, <tr>, </tr>, <td>, </td>, <cell_y>, <cell_n>. For other parameters, we use the default values in OpenNMT.

5.3. Results

The evaluation results of table detection models are shown in Table 2. We observe that models perform well on the same domain. For instance, the ResNeXt-152 model trained with Word documents achieves an F1 score of 0.9166 on the Word dataset, which is much higher than the F1 score (0.8094) on Latex documents. Meanwhile, the ResNeXt-152 model trained with Latex documents achieves an F1 score of 0.9810 on the Latex dataset, which is also much higher than testing on the Word documents (0.8863). This indicates that the tables from different types of documents have different visual appearance. Therefore, we cannot simply rely on transfer learning techniques to obtain good table detection models with small scale training data. When combining training data with Word and Latex documents, the accuracy of larger models is comparable to models trained on the same domain, while it performs better on the Word+Latex dataset. This verifies that model trained with larger data generalizes better on different domains, which illustrates the importance of creating a larger benchmark dataset.

In addition, we also evaluate our models on the ICDAR 2013 table competition dataset shown in Table 3. Among all the models trained with TableBank, the Latex ResNeXt-152 model achieves the best F1 score of 0.9625. Furthermore, we compare our model with models trained with ICDAR, UNLV, Marmot, and DeepFigures, which demonstrates the great potential of our TableBank dataset. This also illustrates that we not only need large scale training data but also high quality data. We also compare with the open source Tesseract[3] and Camelot[4] toolkits, which are based on line information. The results show that image-based deep learning models are significantly better than conventional approaches.

The evaluation results of table structure recognition are shown in Table 4. We observe that the image-to-text models also perform better on the same domain. The model trained on Word documents performs much better on the Word test set than the Latex test set and vice versa. Similarly, the model accuracy of the Word+Latex model is comparable to other models on Word and Latex domains and better on the mixed-domain dataset. This demonstrates that the mixed-domain model might generalize better in real-world applications.

5.4. Analysis

For table detection, we sample some incorrect examples from the evaluation data of Word and Latex documents for the case study. Figure 8 gives three typical errors of detection results. The first error type is **partial-detection**, where only part of the tables can be identified and some information is missing. The second error type is **un-detection**, where some tables in the documents cannot be identified. The third error type is **mis-detection**, where figures and text blocks in the documents are sometimes identified as tables. Taking the ResNeXt-152 model for Word+Latex as an example, the number of un-detected tables is 164. Compared with ground truth tables (2,525), the un-detection rate is 6.5%. Meanwhile, the number of mis-detected tables is 86 compared with the total predicted tables being 2,450. Therefore, the mis-detection rate is 3.5%. Finally, the number of partial-detected tables is 57, leading to a partial-detection rate of 2.3%. This illustrates that there is plenty of room to improve the accuracy of the detection models, especially for un-detection and mis-detection cases.

We observed that some experiments show inferior accuracy during cross-domain testing. The precision of the Latex model is not satisfactory when tested on the Word dataset and the Word+Latex dataset. Meanwhile, the recall of the Word model is poor on all the test dataset, and the recall of the mixed model is also under-performed on the Word dataset and Word+Latex dataset. It shows that there is still a lot of room to improve on the TableBank dataset, as well as some bad cases to be investigated especially under cross-domain settings.

[3]`https://github.com/tesseract-ocr/tesseract`

[4]`https://github.com/socialcopsdev/camelot`

Models	Word			Latex			Word+Latex		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
ResNeXt-101 (Word)	0.9496	0.8388	0.8908	0.9902	0.5948	0.7432	0.9594	0.7607	0.8486
ResNeXt-152 (Word)	0.9530	0.8829	**0.9166**	0.9808	0.6890	0.8094	0.9603	0.8209	0.8851
ResNeXt-101 (Latex)	0.8288	0.9395	0.8807	0.9854	0.9760	0.9807	0.8744	0.9512	0.9112
ResNeXt-152 (Latex)	0.8259	0.9562	0.8863	0.9867	0.9754	**0.9810**	0.8720	0.9624	0.9149
ResNeXt-101 (Word+Latex)	0.9557	0.8403	0.8943	0.9886	0.9694	0.9789	0.9670	0.8817	0.9224
ResNeXt-152 (Word+Latex)	0.9540	0.8639	0.9067	0.9885	0.9732	0.9808	0.9657	0.8989	**0.9311**

Table 2: Evaluation results on Word and Latex datasets with ResNeXt-{101,152} as the backbone networks

Models	Precision	Recall	F1
ICDAR 2013 (train)	0.9748	0.7997	0.8786
UNLV	0.9185	0.9639	0.9406
Marmot	0.7692	0.9844	0.8636
DeepFigures	0.8527	0.9348	0.8918
TableBank (ResNeXt-152, Word)	0.9725	0.8528	0.9087
TableBank (ResNeXt-152, Latex)	**0.9658**	**0.9594**	**0.9625**
TableBank (ResNeXt-152, Word + Latex)	0.9635	0.9039	0.9328
Tesseract	0.9439	0.7144	0.8133
Camelot	0.9785	0.6856	0.8063

Table 3: Evaluation results on ICDAR 2013 dataset

Models	Word	Latex	Word+Latex
Image-to-Text (Word)	**0.7507**	0.6733	0.7138
Image-to-Text (Latex)	0.4048	**0.7653**	0.5818
Image-to-Text (Word+Latex)	0.7121	0.7647	**0.7382**

Table 4: Evaluation results (BLEU) for image-to-text models on Word and Latex datasets

For table structure recognition, we observe that the model accuracy reduces as the length of output becomes larger. Taking the image-to-text model for Word+Latex as an example, the number of exact match between the output and ground truth is shown in Table 5. We can see that the ratio of exact match is around 50% for the HTML sequences that are less than 40 tokens. As the number of tokens becomes larger, the ratio reduces dramatically to 8.6%, indicating that it is more difficult to recognize big and complex tables. In general, the model totally generates the correct output for 338 tables. In order to avoid the influence of the length distribution deviation in the training data, we also count the length distribution in the training data in Table 6. We believe enlarging the training data will further improve the current model especially for tables with complex row and column layouts, which will be our next-step effort.

Length	0-20	21-40	41-60	61-80	>80	All
#Total	32	293	252	145	278	1,000
#Exact match	15	169	102	28	24	338
Ratio	0.469	0.577	0.405	0.193	0.086	0.338

Table 5: Number of exact match between the generated HTML tag sequence and ground truth sequence

6. Related Work

6.1. Table Detection

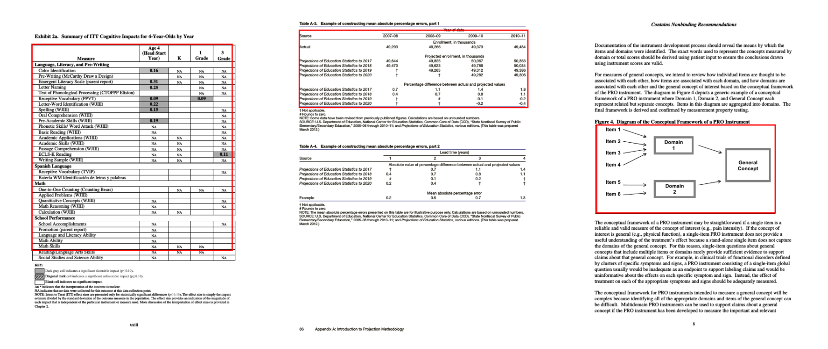
Table detection aims to locate tables using bounding boxes in a document. The research of table detection dates back to the early 1990s. (Itonori, 1993) proposed a rule-based approach that leverages the text block arrangement and ruled line position to detect table structures. At the same time, (Chandran and Kasturi, 1993) designed a structural table detection method based on horizontal and vertical lines, as well as the item blocks. Following these works, there is a great deal of research work (Hirayama, 1995; Green and Krishnamoorthy, 1995; Tupaj et al., 1996; Hu et al., 1999; Gatos et al., 2005; Shafait and Smith, 2010) focus on improving rule-based systems. Although these methods perform well on some documents, they require extensive human efforts to figure out better rules, while sometimes failing to generalize to documents from other sources. Therefore, it is inevitable to leverage statistical approaches in table detection.

Length	0-20	21-40	41-60	61-80	>80	All
#Total	4,027	44,811	36,059	19,757	40,809	145,463
Ratio	0.028	0.308	0.248	0.136	0.281	1.000

Table 6: Length distribution in training data

To address the need for generalization, statistical machine learning approaches have been proposed to alleviate these problems. (Kieninger and Dengel, 1998) was one of the first to apply unsupervised learning method to the table detection task back in 1998. Their recognition process differs significantly from previous approaches as it realizes a bottom-up clustering of given word segments, whereas conventional table structure recognizers all rely on the detection of some separators such as delineation or significant white space to analyze a page from the top-down. In 2002, (Cesarini et al., 2002) started to use supervised learning method by means of a hierarchical representation based on the MXY tree. The algorithm can be adapted to recognize tables with different features by maximizing the performance on an appropriate training set. After that, table detection has been cast into a set of different machine learning problems such as sequence labeling (Silva and e., 2009), feature engineering with SVM (Kasar et al., 2013) and also ensemble a set of models (Fan and Kim, 2015) including Naive Bayes, logistic regression, and SVM. The application of machine learning methods has significantly improved table detection accuracy.

Recently, the rapid development of deep learning in computer vision has a profound impact on the data-driven image-based approaches for table detection. The advantage of the images-based table detection is two-fold: First, it is robust to document types by making no assumption of



(a) (b) (c)

Figure 8: Table detection examples with (a) partial-detection, (b) un-detection and (c) mis-detection

whether scanned images of pages or natively-digital document formats. Second, it reduces the efforts of hand-crafted feature engineering in conventional machine learning. (Hao et al., 2016) first used convolutional neural networks in table detection, where some table-like areas are selected first by some loose rules, and then the convolutional networks are built and refined to determine whether the selected areas are tables or not. In addition, the Faster R-CNN model has been also used in table detection. (Schreiber et al., 2017) presented a system that is totally data-driven and does not need any heuristics or metadata to detect as well as to recognize tabular structures on the ICDAR 2013 dataset. At the same time, (Gilani et al., 2017) achieved state-of-the-art performance using the Faster R-CNN model on the UNLV dataset. Despite the success of applying deep learning models, most of these methods fine-tune pre-trained models on out-of-domain data with just a few thousand human-labeled examples, which is still difficult to be practicable in real-world applications. Till now, there is not any standard benchmark training dataset for both table detection and recognition. Therefore, we make the TableBank dataset publicly available and hope it can release the power of many deep learning approaches.

6.2. Table Structure Recognition

Table structure recognition aims to identify the row and column layout information for a table. The research of table structure recognition also includes rule-based approaches, machine learning approaches, and deep learning approaches. (Ramel et al., 2003) developed a method using the analysis of the graphic lines to detect and extract tables. After that, (Yildiz et al., 2005) used the 'pdftohtml[5]' tool that returns all text elements in a PDF file and computed horizontal overlaps of texts to recognize columns. (Hassan and Baumgartner, 2007) grouped tables into three categories by the principle if a table has horizontal or vertical ruling lines, and they developed heuristic methods to detect these tables. Unlike other methods designing rules for specified tables, (Shigarov et al., 2016) presented a more general rule-based approach to apply to different domains. For machine learning-based approaches, (Hu et al., 2000) identified columns by hierarchical clustering and then spatial and lexical criteria to classify headers, and also addressed the evaluating problem by designing a new paradigm "random graph probing".

Recently, (Schreiber et al., 2017) used the deep learning-based object detection model with pre-processing to recognize the row and column structures for the ICDAR 2013 dataset. Similarly, existing methods usually leveraged no training data or only small scale training data for this task. Distinct from existing research, We use the TableBank dataset to verify the data-driven end-to-end model for structure recognition. To the best of our knowledge, the TableBank dataset is the first large scale dataset for both table detection and recognition tasks.

7. Conclusion

To empower the research of table detection and structure recognition for document analysis, we introduce the TableBank dataset, a new image-based table analysis dataset built with online Word and Latex documents. We use the Faster R-CNN model and image-to-text model as the baseline to evaluate the performance of TableBank. In addition, we have also created testing data from Word and Latex documents respectively, where the model accuracy in different domains is evaluated. Experiments show that image-based table detection and recognition with deep learning is a promising research direction. We expect the TableBank dataset will release the power of deep learning in the table analysis task, meanwhile fosters more customized network structures to make substantial advances in this task.

For future research, we will further enlarge the TableBank from more domains with high quality. Moreover, we plan to build a dataset with multiple labels such as tables, figures, headings, subheadings, text blocks and more. In this way, we may obtain fine-grained models that can distinguish different parts of documents.

8. Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant Nos.U1636211, 61672081,61370126),the Beijing Advanced Innovation Center for Imaging Technology (Grant No.BAICIT-2016001), and the Fund of the State Key Laboratory of Software Development Environment (Grant No.SKLSDE-2019ZX-17).

[5]`http://pdftohtml.sourceforge.net/`

9. Bibliographical References

Cesarini, F., Marinai, S., Sarti, L., and Soda, G. (2002). Trainable table location in document images. In *Object recognition supported by user interaction for service robots*, volume 3, pages 236–240 vol.3, Aug.

Chandran, S. and Kasturi, R. (1993). Structural recognition of tabulated data. In *Proc. of ICDAR 1993*, pages 516–519, Oct.

Deng, Y., Kanervisto, A., and Rush, A. M. (2016). What you get is what you see: A visual markup decompiler. *CoRR*, abs/1609.04938.

Fan, M. and Kim, D. S. (2015). Table region detection on large-scale PDF files without labeled data. *CoRR*, abs/1506.08891.

Gatos, B., Danatsas, D., Pratikakis, I., and Perantonis, S. J. (2005). Automatic table detection in document images. In *Proc. of ICAPR 2005 - Volume Part I*, ICAPR'05, pages 609–618, Berlin, Heidelberg. Springer-Verlag.

Gilani, A., Qasim, S. R., Malik, I., and Shafait, F. (2017). Table detection using deep learning. In *Proc. of ICDAR 2017*, volume 01, pages 771–776, Nov.

Girshick, R. B., Donahue, J., Darrell, T., and Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524.

Girshick, R., Radosavovic, I., Gkioxari, G., Dollár, P., and He, K. (2018). Detectron. `https://github.com/facebookresearch/detectron`.

Girshick, R. B. (2015). Fast R-CNN. *CoRR*, abs/1504.08083.

Green, E. and Krishnamoorthy, M. (1995). Recognition of tables using table grammars. In *Proc. of the Fourth Annual Symposium on Document Analysis and Information Retrieval*, pages 261–278.

Göbel, M., Hassan, T., Oro, E., and Orsi, G. (2013). ICDAR 2013 table competition. In *Proc. of ICDAR 2013*, pages 1449–1453, Aug.

Hao, L., Gao, L., Yi, X., and Tang, Z. (2016). A table detection method for pdf documents based on convolutional neural networks. In *12th IAPR Workshop on Document Analysis Systems*, pages 287–292, April.

Hassan, T. and Baumgartner, R. (2007). Table recognition and understanding from pdf files. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 2, pages 1143–1147. IEEE.

Hirayama, Y. (1995). A method for table structure analysis using dp matching. In *Proc. of ICDAR 1995*, volume 2, pages 583–586 vol.2, Aug.

Hu, J., Kashi, R. S., Lopresti, D. P., and Wilfong, G. (1999). Medium-independent table detection.

Hu, J., Kashi, R. S., Lopresti, D. P., and Wilfong, G. (2000). Table structure recognition and its evaluation.

Itonori, K. (1993). Table structure recognition based on textblock arrangement and ruled line position. In *Proc. of ICDAR 1993*, pages 765–768, Oct.

Kasar, T., Barlas, P., Adam, S., Chatelain, C., and Paquet, T. (2013). Learning to detect tables in scanned document images using line information. In *Proc. of ICDAR 2013*, pages 1185–1189, Aug.

Kieninger, T. and Dengel, A. (1998). The t-recs table recognition and analysis system. In *Document Analysis Systems: Theory and Practice*, pages 255–270, Berlin, Heidelberg. Springer Berlin Heidelberg.

Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. M. (2017). OpenNMT: Open-source toolkit for neural machine translation. In *Proc. of ACL*.

Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312.

Ramel, J.-Y., Crucianu, M., Vincent, N., and Faure, C. (2003). Detection, extraction and representation of tables. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, pages 374–378. IEEE.

Ren, S., He, K., Girshick, R. B., and Sun, J. (2015). Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252.

Schreiber, S., Agne, S., Wolf, I., Dengel, A., and Ahmed, S. (2017). Deepdesrt: Deep learning for detection and structure recognition of tables in document images. In *Proc. of ICDAR 2017*, volume 01, pages 1162–1167, Nov.

Shafait, F. and Smith, R. (2010). Table detection in heterogeneous documents. In *Document Analysis Systems 2010*.

Shahab, A., Shafait, F., Kieninger, T., and Dengel, A. (2010). An open approach towards the benchmarking of table structure recognition systems. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, pages 113–120, New York, NY, USA. ACM.

Shigarov, A., Mikhailov, A., and Altaev, A. (2016). Configurable table structure recognition in untagged pdf documents. In *Proceedings of the 2016 ACM Symposium on Document Engineering*, pages 119–122. ACM.

Siegel, N., Lourie, N., Power, R., and Ammar, W. (2018). Extracting scientific figures with distantly supervised neural networks. *CoRR*, abs/1804.02445.

Silva and e., A. C. (2009). Learning rich hidden markov models in document analysis: Table location. In *Proc. of ICDAR 2009*, pages 843–847, Washington, DC, USA. IEEE Computer Society.

Tupaj, S., Shi, Z., Chang, C. H., and Chang, D. C. H. (1996). Extracting tabular information from text files. In *EECS Department, Tufts University*.

Xie, S., Girshick, R. B., Dollár, P., Tu, Z., and He, K. (2016). Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431.

Yildiz, B., Kaiser, K., and Miksch, S. (2005). pdf2table: A method to extract table information from pdf files. In *IICAI*, pages 1773–1785.