




Converting Image Files to LaTeX Format Using Computer Vision, Natural Language Processing, and Machine Learning

Resim Formatındaki Dokümanların Bilgisayarlı Görü, Doğal Dil İşleme ve Makine Öğrenmesi Kullanılarak Latex Formatına Dönüştürülmesi

Murat Kazanç¹ , Tolga Ensari² , Mustafa Dağtekin³ 

¹(M.Sc.), Istanbul University-Cerrahpasa, Department of Computer Engineering, Istanbul, Türkiye
²(Assist. Prof.), Arkansas Tech University, College of Engineering & Applied Science, Department of Computer and Information Science, Arkansas, USA
³(Assist. Prof.), Istanbul University-Cerrahpasa, Department of Computer Engineering, Istanbul, Türkiye

Corresponding author : Mustafa DAĞTEKİN
E-mail : dagtekin@iuc.edu.tr

ABSTRACT

A few decades ago, people used printed resources such as books and magazines to learn. With the development of technology, digital documents have replaced printed resources. These documents can occur in the form of images or various text formats. Many different applications exist for preparing digital documents, one of these being LaTeX. LaTeX is a document preparation system and typesetting software that is used especially in the field of scientific publications and mathematics for preparing high quality documents. When preparing a document using LaTeX, the content is made ready using a markup language, which creates difficulties for some users. However, one of the main advantages of using the LaTeX system is that it distinguishes the document's content from its formatting. Once the content is created, the formatting can be easily replaced. Generating LaTeX code from an image-formatted document requires both the use of computer vision and NLP. This study discovers the boundaries (blocks) of the places where text, tables, and figures are located on an image before making a text classification using the natural language processing methods of these blocks. The next stage of the study determines the reading order to enable meaningful flow. The final stage of the study produces a LaTeX code using the obtained information.

Keywords: Computer vision, text classification, reading order, machine learning

ÖZ

Birkaç on yıl önce insanlar bilgi edinmek için kitap ve dergi gibi basılı kaynakları kullanmaktaydılar. Teknolojinin gelişmesi ile basılı kaynakların yerini dijital dokümanlar almıştır. Bu dokümanlar görüntü biçiminde veya farklı metin formatları şeklinde olabilmektedir. Dijital dokümanları hazırlamak için birçok farklı uygulama bulunmaktadır. Bunlardan bir tanesi LaTeX' tir. LaTeX doküman hazırlama sistemi ve dizgi yazılımıdır. Yüksek kalitede dokümanlar hazırlamak için özellikle bilimsel yayınlar ve matematik alanında kullanılmaktadır. LaTeX ile doküman hazırlanırken içerik bir işaretleme dili kullanılarak hazırlanmaktadır. Bu durum bazı kullanıcılar için bir zorluk oluşturmaktadır. Ancak LaTeX sistemini kullanmanın avantajlarından biri doküman içeriğini biçimlendirmeden ayırmasıdır. Bir kere içerik oluşturulduktan sonra biçimlendirme kolaylıkla değiştirilebilmektedir. Görüntü formatındaki bir dokümandan LaTeX kodunun üretilmesi bilgisayarlı görüş ve doğal dil işleme alanlarının birlikte kullanılmasını gerektirmektedir. Bu çalışmada öncelikle görüntü üzerinde metin, tablo ve şekillerin bulunduğu yerlerin sınırları (bloklar) tespit edilmiştir. Sonrasında bulunan bu blokların doğal dil işleme metotları kullanılarak metin sınıflama yapılmıştır. Bir sonraki aşamada anlam akışının bozulmaması için okuma sırası tespit edilmiştir. Son aşamada elde edilen bilgiler kullanılarak LaTeX kodu üretilmiştir.

Anahtar Kelimeler: Bilgisayarlı görüş, metin sınıflama, okuma sırası, makine öğrenmesi

Submitted : 01.03.2023
Revision Requested : 24.08.2023
Last Revision Received : 31.08.2023
Accepted : 19.09.2023
Published Online : 26.10.2023



This article is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0)

1. INTRODUCTION

A few decades ago, people used such resources as books and magazines to obtain information. With the current developments in technology and the widespread use of the Internet, digital documents are used more often than printed documents. Different applications are also available for preparing digital documents. One such tool is LaTeX, a document preparation system and typesetting software (CTAN Team). LaTeX is used to create high-quality documents, particularly in the fields of science and mathematics. When creating a document using LaTeX, one writes the contents of the document in a plain text file using a simple markup language to indicate the structure of the document (e.g., headings, paragraphs, lists) and to include such things as mathematical equations and citations. This plain text file is then processed by the LaTeX software, which converts it into a typeset document in one's chosen format (e.g., PDF). One of the main advantages of using LaTeX is that it separates the contents of a document from its formatting. This means one can focus on writing the content and let LaTeX handle the document's typesetting details. This makes creating consistent, high-quality documents easier and allows facilitates later changes. The production of LaTeX code from an image of a document is obtainable using computer vision and natural language processing (NLP).

This study focuses on academic publications. Due to their structure, the first page of academic publications contains sections such as the title, author information, abstract, and keywords. The middle pages of a publication contain content groups such as headings, paragraphs, tables, figures, and lists in single- or double-column format. The last page or pages of a publication contain a unique reference list. In order for a digital academic publication to be analyzed, these content groups should initially be found in the digital document as a content block, and then the content should be parsed as text or figures. The contents of the blocks identified as text should be classified according to the type of the page (left justified, centered, or right justified). In addition, in order to not disrupt the flow of a digital publication, the reading order must also be correctly identified. LaTeX code should be created in the final stage, using all the information collected so far.

This study does not include certain features within its scope, such as font type and font size, which should be represented in a digital document. In addition, converting the equation and table of content groups of a digital document into LaTeX code requires a completely different study on the subject. The operations performed in this study will prepare a starting point and an expandable working basis for future studies.

2. Literature Review

LaTeX is a document preparation system. Widely preferred word processing programs such as Microsoft Word and Libre Office Writer can be expressed as follows: What you see is what you get (Klatsky, 2003). An attribute of these programs is that they allow the user to display a document on a computer monitor in exactly the format the document will take when printed. With LaTeX, the content of the document is written in plain text. The desired formatting is made using LaTeX codes, with the result displayed after compilation. In this respect, LaTeX can be compared to the Hypertext Markup Language (HTML) used to develop web pages.

LaTeX was founded in 1978 by Donald E. Knuth, and the development process has been non-stop ever since (CTAN Team). The starting point was the insufficient print quality of the documents. Over the years, LaTeX has been adopted by academic circles in particular for writing books and articles and continues to be developed as an open-source code today. The main reasons why LATEX is preferred are as follows: It can be published professionally, it is a standard, it can be used across platforms, and it is constantly evolving and expanding due to its open-source nature.

In order to convert a document's image file to LaTeX code, the first thing to do is to have blocks such as text or shapes on the image. One of the libraries developed for this process is LayoutParser (Shen et al., 2021). Using deep learning methods, this open-source library facilitates digital document analysis. Another feature that makes LayoutParser stand out is its pre-trained artificial neural network models. These are PubLayNet for academic documents (Zhong et al., 2019), PRImA for journals and academic reports (PRImA), and TableBank for business and academic documents (Li et al., 2019). These pre-trained models have common classes, as well as their own specialized classes. For example, the TableBank pre-trained model finds only tables. LayoutParser also has an optical character recognition (OCR) feature that uses the Tesseract library (Wang et al., 2021).

Another study related to the discovery of blocks in pictures involves Layout (Y. Xu et al., 2020). That study stated that in order to classify the blocks found in the picture, the texts contained in the blocks should be taken into account in addition to their formal characteristics. For example, the bidirectional encoder representations for transformers (BERT) is a pre-trained NLP models that is used to understand texts.

For the purpose of finding blocks, two deep learning models have been used for segmentation (C. Xu et al., 2021). The first deep learning model determined the locations of the blocks using Mask region-based convolutional neural

networks (R-CNN). The second deep learning model uses feature pyramid networks (FPNs) to perceive objects at different scales, and the blocks are classified as a result of the collaboration of these two deep learning models.

Clark & Divvala (2016) claimed that shapes are important in academic publications with a different focus, and their main goal was to identify pictures and the explanatory text underneath them. Instead of using a deep learning model in the development stage, they developed OCR and a rules-based detection method.

Deivalakshmi et al. (2013) proposed an algorithm-based system for detecting text and non-text blocks in a digital document. Horizontal and vertical lines are detected and removed. The blocks are then revealed by grouping text or non-text content using the dilation method. Pictures have different texture properties than text. Texture properties occur with gray-level co-occurrence matrices (GLCMs) in the found blocks. A GLCM is able to work on grayscale images by creating a matrix using just the position numbers of pixels in the vertical, horizontal, and diagonal directions for every pixel in the entire image or in divisions. The next stage classifies the blocks as text or non-text using the k-means clustering algorithm.

Kavasidis et al. (2019) used convolutional neural networks (CNNs) to detect tables and figures in digital documents. Nowadays, a rapid increase in data production is taking place that continues to accelerate. At this point, no person or group of people can monitor this information flow. Instead of trying to get information in digital documents with NLP methods, the solution to this problem is to conclude that the tables and figures in academic publications provide more information than the textual contents of the document.

Deng et al. (2019) used the encoder-decoder model was used to detect tables in documents. They extracted the feature map of the image using CNN in the encoder model. Meanwhile, the decoder model detects table cells using the attention mechanism and the standard long- and short-term memory (LSTM) model.

One of the strongest points of LaTeX involves equations. Finding equations in a picture and expressing them as LaTeX code would be quite convenient. To perform this process, Wang and Liu, (2021) again used the encoder-decoder model, extracting the feature map using CNN as the encoder. A LaTeX code block was produced with an LSTM using a soft attention mechanism as a decoder.

Pang et al.'s (2021) study on converting equations to LaTeX code used a transformer model instead of the encoder-decoder models most researchers use. The main reason for this was that the encoder-decoder model cannot know the entire context and hierarchical positions of the symbols. The study consisted of three stages: extracting the entire context, using a transformer model that reveals the dependencies between location information and symbols, and using a mask-based attention mechanism decoder to produce LaTeX code.

Another process performed in extracting information from digital documents is to determine the fonts of the texts. This feature provides structured information about the document rather than the content. To do this, Wang et al.'s (2015) study used a stacked convolutional autoencoder (SCAE), a special type of CNN.

Ding et al. (2019) conducted on OCR systems, stating that OCR models using CNN and LSTM achieved a high accuracy rate. However, the model was said to occupy a large amount of disk space and to have a high computational cost. The method they proposed was to reduce the LSTM side of the model without making changes to the feature extraction process. Finally, they used a compression method to reduce the size of the model.

Safnuk and Hu's (2018) research was conducted on PDF files with the goal of obtaining LaTeX code by reverse engineering the PDF files produced with LaTeX code. A deep learning model was used to understand the relationships in the metadata of the source PDF file and to produce appropriate codes. The LaTeX code of the model output a digital document that had been successfully reconstructed.

Apart from the literature study, real-world applications have also been examined. Web applications are found that convert word processing document formats (e.g., docx) to LaTeX code. These applications contain text information in XML format in the source file as well as font, font size, and all visual features related to the text. The main difficulty in converting a document's image file to LaTeX code is the availability of these features. Again, by performing OCR on the digital document provided by different applications, all the found characters are placed in their positions on the page with fixed coordinates. These studies were unable to produce a code similar to LaTeX code written by a human.

3. Method

3.1. Dataset

The current study has found no conducted stud to have a suitable dataset. For this reason, 103 academic publications in PDF format were accessed on the web environment. These were then used during the literature search on the subject.

The PDFs collected for training the system under development were first converted to an image file so that each page is a separate image. The information then needs to be extracted from the pictures. This information involves the

location of the text, title, lists, tables, and image blocks, as well as the text contained in the blocks. In order to find the classes of the blocks, the study uses the LayoutParser library (Shen et al., 2021) instead of labeling them individually. This library allows the location and classes of the blocks in each image to be found and recorded in CSV format in accordance with the segmentation system to be developed later using OCR as the last operation on the texts the blocks contain. Table 1 shows the dataset sample records. In order to create the data set, 11,965 blocks were extracted from 1,141 images.

Table 1. *Dataset Sample Records*

Index	Box Index	Box X1	Box X2	Box Y1	Box Y2	Box Text	Box Type	Page Size	Page Number
200	10	951	1574	333	1419	Intelligent Multimedia...	List	2200x1700	3
201	0	225	1440	1792	1848	Fig. 2. Results: a) Input...	Text	2200x1700	4
202	2	219	1450	196	1742	fay\n'nie\n'ni f'n\n \n'n...	Figure	2200x1700	4
203	3	226	843	215	1375	RT ee aa ates caret\n'...	Figure	2200x1700	5
205	9	112	1545	425	1824	Classification Kate for...	Table	2200x1700	5

3.1.1. Data Preprocessing

In order to distinguish between figures and text in the system to be developed in the next stage, the ratio of the area covered by a block to the entire area of the image in the data set and the number of words contained in the block are needed. Firstly, the width and height are calculated from the required columns of the block expressing the coordinates as a rectangle and added to the dataset table as a new column. Then, the page area is calculated using the page width and height from the data set. Finally, the area occupied by the block on the page as a percentage was found and added to the data set. At the development stage, the positions of the blocks and other unnecessary information were discarded, and a data set was formed.

The texts in the blocks contain meaningless characters left over from the OCR process, as well as characters that are not important for the number of words, such as punctuation marks. These need to be cleared in order to reach a correct number of words. The text is then converted to words, and the word count is calculated. Finally, a parameter is obtained using the ratio of the block to the page area and the number of words; this is then added to the data set. Table 2 was formed once the basic data analysis had been performed on the obtained data set.

Table 2. *Percentages for the Number of Words in the Field*

Index	Box Index	Box X1	Box X2	Box Y1	Box Y2	Box Text	Box Type	Page Size	Page Number
200	10	951	1574	333	1419	Intelligent Multimedia...	List	2200x1700	3
201	0	225	1440	1792	1848	Fig. 2. Results: a) Input...	Text	2200x1700	4
202	2	219	1450	196	1742	fay\n'nie\n'ni f'n\n \n'n...	Figure	2200x1700	4
203	3	226	843	215	1375	RT ee aa ates caret\n'...	Figure	2200x1700	5
205	9	112	1545	425	1824	Classification Kate for...	Table	2200x1700	5

Block types are arranged in such a way that their shapes and text (e.g., list, table, text, and title) are combined. In the experiments, the Table Block detection was unsuccessful.

3.1.2. Page and Block Types

Academic publications do not have a complete set of structural standards, but certain structures are common. For example, the first page of an academic publication should contain a main title and an abstract. However, keywords are not found in every publication format. As another example, the last page should contain reference information. Due to these differences, instead of using the same text classification model for each page of the digital document, three different classes have been determined for the pages of a digital document. These are the first pages, the middle pages, and the last (reference) pages. After determining the page type, the models to be developed for the text classes suitable for that page type will be used. For this reason, the selection was made from the data set obtained at the beginning. Afterward, the data set was then recreated by applying the OCR process to the entire page.

Data with block type figures are decoded from the middle pages data set. This stage will take no action regarding that data class. The middle pages' text classes are decoded as text, titles, tables, or lists.

For the first pages, the block types in the dataset are determined as main title, author(s), keywords, title, and text. For the last pages (i.e., references), the block types in the dataset are text, reference, title, and table.

3.2. Training the Model

This study, carries out different model trainings within its scope to perform different tasks. The model trainings divide the datasets into 80% training and 20% testing. The study uses the following machine learning models: decision tree, support vector machine (SVM), k-nearest neighbor (KNN), random forest, and linear classifier. The decision tree model has internal nodes that can be taken as tests on input data patterns and leaf nodes that can be taken as categories. These tests are filtered down through the tree to get the right output-to-input pattern (Navada et al., 2011). SVM is a useful methodology for finding the best possible surface to separate positive samples from negative samples (Ali et al., 2016). KNN is categorized as an unknown document. The KNN classifier ranks the document's neighbors among the training documents and uses the class labels of the k-most similar neighbors (Uğuz, 2011). As the name implies, the random forest classifier consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest in this algorithm gives a class prediction, and the class with the most votes among the trees becomes the model's predicted class (Akpan & Starkey, 2021). Linear classifier is a method for dividing data into classes by finding a linear combination of attributes (Doğan et al., 2019).

3.2.1. Training the Text and Figure Classifications Model

Table 3 provides the algorithms used to perform the figures and text classifications of the block type and the complexity matrix formed as a result of the prediction applied to the dataset test section.

Table 3. Text/Figure Model Evaluation Values

Classification Model	Accuracy
Decision Tree	0.9736
SVM	0.9690
KNN	0.9803
Random Forest	0.9791

Looking at the results from the machine learning models, the most successful model is KNN.

3.2.2. Training the Page Type Classification Model

In order to use the dataset prepared for determining the page type when training the model, the data are converted into a vector showing the representation of words. The accuracy values of the trained models are given in Table 4.

Table 4. Page Type Model Training Accuracy Values

Model	Accuracy
Decision Tree	0.955307
SVM	0.944134
Random Forest	0.910615
Linear	0.966480

Looking at the results, the linear classifier is the most successful model. The complexity matrix for this model is given in Table 5.

Table 5. Linear Classifier Model Training Complexity Matrix

		Estimation		
		First Pages	Middle Pages	Last Pages
Real Results	First Pages	16	3	0
	Middle Pages	1	130	0
	Last Pages	0	2	27

3.2.3. Training the Middle Pages Text Classification Model

With an 87% accuracy, the SVM classifier was the most successful model in the trainings conducted using the text classification methods. When examining the results, table blocks were seen to have the lowest accuracy of about 33% success in being classified.

3.2.4. Training the First Pages Text Classification Model

The linear classifier was the most successful model in the trainings conducted using the text classification methods, with an 82% accuracy rate. When examining the results, the keyword and main title blocks were seen to have the lowest accuracy in being classified, with a success rate of about 75%.

3.2.5. Training the Last Pages (i.e., Reference Pages) Text Classification Model

The SVM classifier was the most successful model in the trainings conducted using the text classification methods, with an 89% accuracy. When examining the results, failure occurred in the table class due to a sufficient sample size not being provided.

3.3. Segmentation

The first stage in producing LaTeX code from a digital document's image file is the presence of content regions (blocks) on the image. The OpenCV library was used for this task. The color scale of the pictures were converted to grayscale so that the boundaries of the blocks could be found more easily. The pictures consist of three channels (i.e., red, green, and blue [RGB]). For each pixel, there are values for these three channels. For the grayscale operation, the OpenCV RGB multiplies each channel by an empirical number, as shown in Equation 1 (*Recommendation ITU-R BT.601-7, 2011*). R is the red channel. G is the green channel. B is the blue channel, and Y is the grayscale channel of the picture.

$$RGBtoGray : Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \quad (1)$$

The Otsu (1979) threshold determination method was used to convert the grayscale image to binary toning (black and white). This method first extracts the histogram of the picture. By assuming the threshold value of each class in the histogram, pixel values with smaller thresholds are then calculated as the background, and pixel values with bigger thresholds are calculated as the foreground. The variance value is calculated for the remaining classes as shown in Equation 2. The goal is to minimize the intra-class variance and maximize the inter-class variance. The variable t is a step. $P(i)$ is the probability found from the histogram, q_1 and q_2 are the cumulative sum of the classes. The sum of q_1 and q_2 equals 1, μ_1 and μ_2 are means, and σ_1^2 and σ_2^2 are variances.

$$\begin{aligned}
 q_1(t) &= \sum_{i=1}^t P(i) & q_2(t) &= \sum_{i=t+1}^I P(i) \\
 \mu_1(t) &= \sum_{i=1}^t \frac{iP(i)}{q_1(t)} & \mu_2(t) &= \sum_{i=t+1}^I \frac{iP(i)}{q_2(t)} \\
 \sigma_1^2(t) &= \sum_{i=1}^t [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)} & \sigma_2^2(t) &= \sum_{i=t+1}^I [i - \mu_2(t)]^2 \frac{P(i)}{q_2(t)}
 \end{aligned} \quad (2)$$

The peculiarity of the Otsu method is that the threshold value is not a fixed value but rather is determined dynamically. The pixels above the threshold value will be white, and the pixels below the threshold value will be black. This method is used to obtain the coordinates of the individual blocks by examining the pixels.

This study has found contours to be unnecessary. One example is the page number found in a picture, or informational messages placed at the top and bottom of a publication downloaded on the web environment stating which academic institution through which the website connection has been established. These contours should not be taken into account. In order to remove them from the contour list, the following situations were observed: If the area occupied by the stroke is less than 10% of the page, if the aspect ratio is less than 20%, if the y-point value of the stroke is less than 3% of the height of the page, or if the y-point value of the stroke is greater than 95% of the height of the page, this stroke is not included in the block list being created. As a separate process, OCR was applied to the image fragments that were created using contour coordinates. The OCR results found the number of words by removing special characters and punctuation marks.

3.4. Reading Order

The order of reading plays an important role in understanding a document. If the document is more than one column, the number of columns should be found, and then the contents (e.g., text, figures, tables) should be identified. The pseudo-code of the reading order algorithm is as follows:

```

GET → blocks(), pageWidth
FOR i TO blocks.count STEP i++
    IF blocks.width(i)-blocks.x(i) > pageWidth /2
        fullColumns.add(blocks.id(i))
    IF blocks.width (i) < pageWidth /2
        leftColumns.add(blocks.id(i))
    IF blocks.x(i) > pageWidth /2
        rightColumns.add(blocks.id(i))
ENDFOR
sortTopToBottom(fullColumns)
sortTopToBottom (leftColumns)
sortTopToBottom (rightColumns)
IF blocks.count = fullColumns.count
    readingOrder = fullColumns
ELSEIF fullColumns.count = 0
    readingOrder = leftColumns + rightColumns
ELSE
    FOR fullColumns.count TO m STEP m—
        FOR rightColumns.count TO n STEP n—
            IF fullColumns.y(m) < rightColumns.y(n)
                readingOrder.add(rightColumns (n))
            ENDFOR
        FOR leftColumns.count TO k STEP k—
            IF fullColumns.y(m) < leftColumns.y(k)
                readingOrder.add(leftColumns (k))
            ENDFOR
        ENDFOR
    readingOrder = reverse(readingOrder)
ENDIF
OUT → readingOrder

```

After the segmentation of the document in the previous processing step, the process of finding the reading order is then carried out using the x and y coordinates and width and height information from which the blocks were formed. In addition, page margin information and inter-column space information, if any, are also found while creating the reading order.

3.5. Text/Figure Classification

For the blocks found as a result of segmentation and placed in reading order, the next stage is to find out which blocks are text and which are shapes. For this process, the values required for normalization are calculated from the data used in the training of the model, then the normalization process is applied to the newly found page block area ratio and word number values. The machine learning model prediction results are then added as a new column to the dataset. In the next stage, the blocks that have been classified as shapes are cropped from the source image according to their coordinates and saved as a new image file. The ID number for that block is given as the name.

3.6. Page Type Classification

OCR is performed on the entire source image to determine the page type. In order to determine the page type, the OCR result made using the vector state of the dataset used in the model's training as a source is converted into a vector, and then predictions are made on the model. The prediction results can be one of three classes: first pages, middle pages, and last pages.

3.7. Classification Of Text According to Page Type

Three different machine learning models have been trained to perform the text classification process according to page type. The predictions are performed using one of these models based on the page type information that was found. The extracted data is then saved to the source folder in CSV format. The page border information (if present), the space between the columns, the width and height of the source image, the information about the columns (if present), and the page type information are saved to the source folder in JSON format. The study has been carried out in this way, with all the operations performed up to this stage having been recorded.

3.8. Generating the LaTeX Code

The presence of special characters that are also used for writing commands in LaTeX code (e.g., % \$ &) in the text will cause compilation errors. In order to prevent errors that may occur, the back space character (i.e., \) is added before these characters should they occur with a function written in the text column in the dataset. The generated LaTeX code is then saved to the source folder in a file with the extension .tex. The text classifications are taken from a list based on page type, and the process then is started.

According to the information from the JSON file for the first pages section, the blocks found as such are identified as single or double columns. The necessary code is then added for the page borders and, if necessary, for the blank information between the columns. The necessary packages are then added. If the source image is classified as a first page, the main page and author information are then added there. LaTeX codes have been added for the middle pages section in accordance with the text classifications contained in the dataset.

3.9. Application Development

The development stages are mentioned in detail. The pseudo-code for the application algorithm is as follows:

4. Findings

Figure 1 shows the .tex file for the compiled PDF output examples; this file's output is formed as a result of a source digital publication given as input to the developed application.


```

GET → image
grayImage = doGrayScale(image)
blackWhiteImage = otsuThreshold(grayImage)
dilationImage = dilation(blackWhiteImage)
blocks() = findContour(dilationImage)
readingOrder = findReadingOrder(blocks())
blocks() = findTextFigure(blocks())
pageType = findPageType(image)
IF pageType = "firstPage"
    blocks() = firstPageTextClassification(blocks())
ELSEIF pageType = "middlePage"
    blocks() = middlePageTextClassification(blocks())
ELSEIF pageType = "lastPage"
    blocks() = lastPageTextClassification(blocks())
ENDIF
OPEN texFile AS tex
    tex.addPreamble(preambleInformation)
    IF pageType = "firstPage"
        tex.preambleAdd("Main Title", "Author",)
        tex.addBody("Title", "Text", "Keyword", "Figure")
    ELSEIF pageType = "middlePage"
        tex.addBody("Title", "Text", "List", "Table", "Figure")
    ELSEIF pageType = "lastPage"
        tex.addBody("Title", "Text", "Referance", "Table", "Figure")
CLOSE texFile
OUT → texFile

```

Figure 1a shows the original document and Figure 1b shows the PDF output of the developed application. The application developed in this example was found to be successful. A two-column page structure has been created. Blocks are found that have been made with errors. In addition, no exact match is found due to features that were not taken into account in this application (e.g., font type, paragraph spaces).

4.1. Segmentation

When evaluating the blocks obtained by finding contours in OpenCV, the results are concluded to be a partial success. Successful results were seen to have been obtained in some experiments, while the results from others were unsatisfactory. If the figures in the digital document have an irregular fragmented structure or occur on the first pages of the document, the segmentation performance for the main heading is observed to be low. The reason why the segmentation system does not exhibit repeatable output is that the documents do not have a specific standardized appearance.

4.2. Text/Figure Classification

The classification was theoretically developed by taking into account the ratio of a shape block to the total area of the document and the ratio of the area of the block and the number of words contained in it. This is theoretically true. However, errors are seen to have occurred in the estimates made on real samples. For example, the outputs given unsuccessfully regarding segmentation are seen for text blocks can be classified as shapes.

4.3. Reading Order

Since the reading order algorithm works heuristically, it works as a rules-based algorithm in accordance with the information coming from segmentation. As a result of the experiments carried out, errors are seen to be able to occur due to errors in segmentation.

incorporated into encoder-decoder framework for calculating the fixed-length context vector, i.e., the variable-length representations could be summed using the attention as the weighting coefficients. After adopting an attention mechanism into encoder-decoder, salient regions in the static representation can dynamically rise to the forefront. The attention mechanism plays an indispensable role in image captioning for obtaining a state-of-the-art performance. For example, [42] proposed the "hard" attention for image captioning system to know where to attend and its effectiveness was shown through attention visualization. [43] proposed the concept of "attention correctness" to strengthen the alignment for image captioning. In [44], an adaptive attention was implemented via a visual sentinel so that the captioning system could also know when to attend, and with a similar motivation, [45] also proposed a global-local attention so that model could selectively pay attention to spatial objects and context information.

D. Neural Network-Based Approaches for OHMER

The generality of the attention based encoder-decoder framework suggests that OHMER may also be one proper application. Recently, [46], [47] used the attention based encoder-decoder model for OHMER and significantly outperformed the best system on CROHME 2014. In [46] the proposed model consisted of a FCN encoder and a GRU decoder equipped with a coverage-based attention model while [47] employed a CRNN as the encoder and the decoder is equipped with a coarse-to-fine attention model. However, both [46] and [47] treated the HMEs input as static images which ignores the handwriting dynamics (namely the temporal order and trajectory). As we can see in Fig. 1, besides the symbol shape information, the writing order is also preserved in the online sequential data, which is important information and can not be recovered from the static image. Therefore, to capture the dynamic information to reduce handwritten ambiguities, [48] proposed to employ a GRU encoder that directly takes the raw sequential data as input. Validated on CROHME 2014, [48] showed a significant improvement of recognition accuracy over [46], [47].

This study is an extension of the previous work in [48] with the following new contributions: 1) We propose to employ a temporal attention to teach the parser when to rely on the representations extracted by tracker and when to just rely on the built-in language model. 2) To compute the temporal attention, the spatial attention is slightly adjusted. Meanwhile, we newly introduce an attention guider to help improve the learning of spatial attention. 3) We blend a FCN watcher into TAP by considering the strong complementarity between static-image based input and dynamic-trace based input. By processing HMEs from two different modalities, the strengths of [46] and [48] can be fully utilized simultaneously. 4) We use an extra official text dataset containing only \LaTeX notations to train an additional language model for enhancing our parser. 5) More experiments and analyses are included.

III. NETWORK ARCHITECTURE OF TAP

In this section, we elaborate the proposed TAP architecture which parses a mathematical expression structure into a \LaTeX

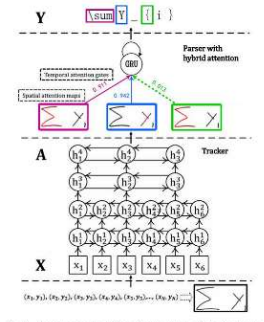


Fig. 2. Overall architecture of Track, Attend, and Parse. X denotes the input sequence in Section III-A. A denotes the annotation sequence in Section III-C. Y denotes the output sequence in Section III-C.

string by tracking a sequence of online handwritten points. As illustrated in Fig. 2, the raw data is a sequence of points containing xy coordinates which can be visualized as the bottom-right image by drawing the trajectory. A preprocessing is first applied to extract trajectory information from raw sequential data. The tracker is a stack of bidirectional GRU while the parser combines a GRU based language model and a hybrid attention mechanism. As for the hybrid attention mechanism, spatial attention can potentially well learn the alignment between input traces and output string while temporal attention can well know when to rely on the product of spatial attention and when to just rely on the language model. For example, in Fig. 2, the purple, blue and green rectangles denote three symbols with the red color representing the spatial attention probabilities of each handwritten symbol (lighter color denotes higher probability) and the probabilities linking to rectangles represent their reliability produced by temporal attention. When predicting the math symbol "sum", the spatial attention model aligns well to the stroke of \sum (in the purple spatial attention map) which corresponds to the human intuition and the temporal attention probability linking to the purple rectangle is extremely high as the spatial attention map is accurate. Conversely, when predicting the math symbol " \int ", there is no object for spatial attention model to attend to, leading to an inaccurate spatial attention map. Therefore the temporal attention probability linking to the green spatial attention map is small which tells the parser should rely on the built-in language model at this time.

incorporated into encoder-decoder framework for calculating the fixed-length context vector, i.e., the variable-length representations could be summed using the attention as the weighting coefficients. After adopting an attention mechanism into encoder-decoder, salient regions in the static representation can dynamically rise to the forefront. The attention mechanism plays an indispensable role in image captioning for obtaining a state-of-the-art performance. For example, [42] proposed the "hard" attention for image captioning system to know where to attend and its effectiveness was shown through attention visualization. [43] proposed the concept of "attention correctness" to strengthen the alignment for image captioning. In [44], an adaptive attention was implemented via a visual sentinel so that the captioning system could also know when to attend, and with a similar motivation, [45] also proposed a global-local attention so that model could selectively pay attention to spatial objects and context information.

D. Neural Network-Based Approaches for OHMER

The generality of the attention based encoder-decoder framework suggests that OHMER may also be one proper application. Recently, [46], [47] used the attention based encoder-decoder model for OHMER and significantly outperformed the best system on CROHME 2014. In [46] the proposed model consisted of a FCN encoder and a GRU decoder equipped with a coverage-based attention model while [47] employed a CRNN as the encoder and the decoder is equipped with a coarse-to-fine attention model. However, both [46] and [47] treated the HMEs input as static images which ignores the handwriting dynamics (namely the temporal order and trajectory). As we can see in Fig. 1, besides the symbol shape information, the writing order is also preserved in the online sequential data, which is important information and can not be recovered from the static image. Therefore, to capture the dynamic information to reduce handwritten ambiguities, [48] proposed to employ a GRU encoder that directly takes the raw sequential data as input. Validated on CROHME 2014, [48] showed a significant improvement of recognition accuracy over [46], [47].

This study is an extension of the previous work in [48] with the following new contributions: 1) We propose to employ a temporal attention to teach the parser when to rely on the representations extracted by tracker and when to just rely on the built-in language model. 2) To compute the temporal attention, the spatial attention is slightly adjusted. Meanwhile, we newly introduce an attention guider to help improve the learning of spatial attention. 3) We blend a FCN watcher into TAP by considering the strong complementarity between static-image based input and dynamic-trace based input. By processing HMEs from two different modalities, the strengths of [46] and [48] can be fully utilized simultaneously. 4) We use an extra official text dataset containing only \LaTeX notations to train an additional language model for enhancing our parser. 5) More experiments and analyses are included.

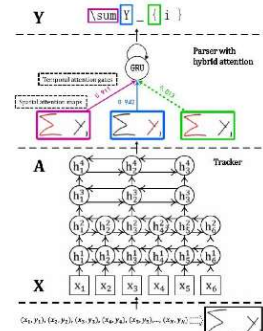


Figure 1: Fig. 2. Overall architecture of Track, Attend, and Parse. X denotes the input sequence in Section III-A. A denotes the annotation sequence in Section III-C. Y denotes the output sequence in Section III-C.

III. NETWORK ARCHITECTURE OF TAP

In this section, we elaborate the proposed TAP architecture which parses a mathematical expression structure into a \LaTeX

Parser with hybrid attention

string by tracking a sequence of online handwritten points. As illustrated in Fig. 2, the raw data is a sequence of points containing xy coordinates which can be visualized as the bottom-right image by drawing the trajectory. A preprocessing is first applied to extract trajectory information from raw sequential data. The tracker is a stack of bidirectional GRU while the parser combines a GRU based language model and a hybrid attention mechanism. As for the hybrid attention mechanism, spatial attention can potentially well learn the alignment between input traces and output string while temporal attention can well know when to rely on the product of spatial attention and when to just rely on the language model. For example, in Fig. 2, the purple, blue and green rectangles denote three symbols with the red color representing the spatial attention probabilities of each handwritten symbol (lighter color denotes higher probability) and the probabilities linking to rectangles represent their reliability produced by temporal attention. When predicting the math symbol "sum", the spatial attention model aligns well to the stroke of \sum (in the purple spatial attention map) which corresponds to

(a)

(b)

Figure 1. Sample application of (a) original document and (b) application output.

4.4. Page Type Classification

The machine learning model applied for finding page types gave completely accurate results in the performed experiments. No erroneous estimates were made.

4.5. Finding Text Classes Specific to the Page Type

For a source document whose page type is a first page, two sample results are shown in Figures 2a and 2b. Segmentation problems regarding finding the main title and author blocks negatively affect the classification process. The errors experienced in segmentation are seen to negatively affect the rest of the system.

For a source document whose page type is a middle page, experiments have shown no repeatable success to occur regarding detecting list and table blocks. For a source document whose page type is a last page, two sample results are shown in Figures 3a and 3b. Experiments have shown that tables and reference blocks are confused with text blocks. No repeatable success has occurred in detecting these blocks.

4.6. Generating the LaTeX Code

LaTeX code is heuristically produced. Therefore, the outputs created in the previous stages of the application are used in this step. Because the errors that occurred in the previous steps are cumulatively transferred to this step, no accurate metric has been found for measuring this stage.

5. Discussion and Conclusion

When evaluating the application on different documents, repeated high success was not achievable. One of the reasons for this is that the system has a complex structure consisting of many stages. This situation causes errors that occur at one point to accumulate and negatively affect the rest of the system.

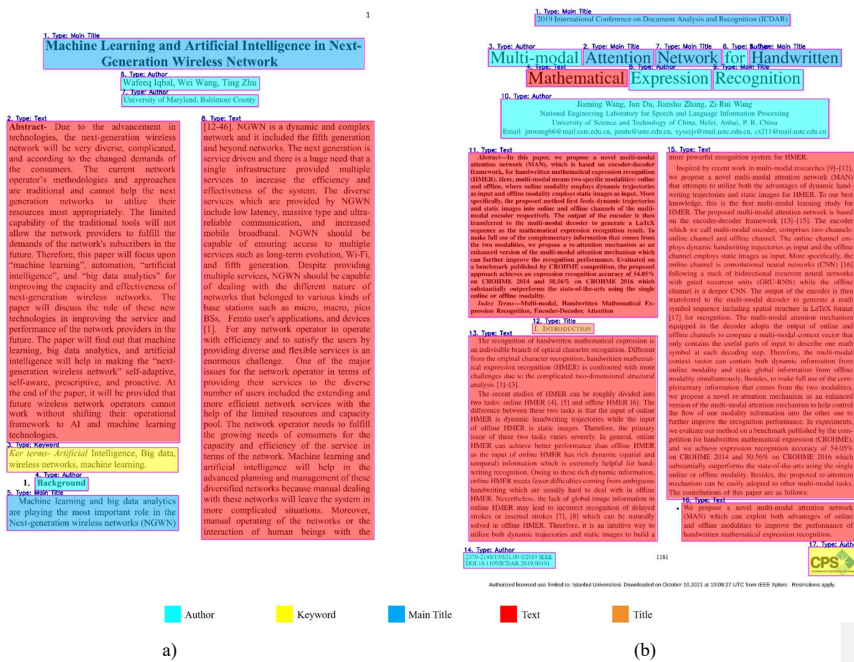


Figure 2. Segmentation results for (a) a successful example and (b) an unsuccessful example.



Figure 3. (a) Segmentation and text classification results for last pages and (b) an example.

Due to its nature, an academic publication consists mainly of text blocks. These text blocks are divided into classes within themselves. Separating these blocks of text from one another is often quite a difficult task. Reference pages are given in Figures 3a and 3b as examples. The presence of different reference systems makes success here difficult. Although NLP methods are used to separate a reference block from a list or a regular paragraph structure, this is a still a monumental task.

During the literature review and the examined commercial applications, focus on one topic is always observed. For example, in a commercial solution that was developed only for converting equations to LaTeX code, the equation block was marked by having the user select it in the browser, after which it is converted to LaTeX code. This is also available in solutions that produce output in JSON format with metadata obtained from PDF files. Apart from these, studies are found to have converted the text obtained with OCR at a very simple level to LaTeX code heuristically. The application developed in this study has been comprehensive and sophisticated based on the mentioned studies.

Comparisons can be made in terms of the segmentation and classification of blocks with the LayoutParser library, which uses the deep learning method in this study to create a dataset. LayoutParser can use different pre-trained models. In accordance with our study, the PublayNet model was preferred. This model has been trained by IBM laboratories using more than 1 million academic publications.

For the comparison process, making a comparison of a document classified as a middle page will be appropriate because the text classes for the documents classified as first and last pages in the developed application were unique. Figure 4b shows a comparison of the system developed in the study with the results of the LayoutParser model used in Figure 4a.



Figure 4. (a) Comparison of the LayoutParser Result and (b) the Developed Application Result.

When examining the system outputs given in Figures 4a and 4b, the LayoutParser made errors in both segmentations and also failed to take some content into account. The work this study has done on these examples was more successful.

When comparing the results from this study with those from the LayoutParser regarding the samples, the success rate is seen to vary. However, the system developed here has entailed a more comprehensive study with text classes specific to reading order and page types. Although a certain success was achieved with the computer vision method applied in this study, the success achieved using the system had varied results among the selected samples.

With regard to the application developed in this study, this study has been shown machine learning models to still be useful with short training times, small model sizes, and fast response times in datasets containing keywords. When additionally considering the long training period and large model sizes of popular NLP models, the NLP models developed for this study can be said to be very practical. As such, the work done in this study is thought to be able to continue being developed in the future or to lay the foundation for future studies.

Peer Review: Externally peer-reviewed.

Author Contributions: Conception/Design of Study- M.K., T.E., M.D.; Data Acquisition- M.K.; Data Analysis/Interpretation- M.K.; Drafting Manuscript- M.K.; Critical Revision of Manuscript- T.E., M.D.; Final Approval and Accountability- M.K., T.E., M.D.

Conflict of Interest: The authors have no conflict of interest to declare.

Grant Support: The authors declared that this study has received no financial support.

ORCID IDs of the authors / Yazarların ORCID ID'leri

Murat Kazanç 0000-0002-8405-0181
Tolga Ensari 0000-0003-0896-3058
Mustafa Dağtekin 0000-0002-0797-9392

REFERENCES

- Akpan, U. I., & Starkey, A. (2021). Review of classification algorithms with changing inter-class distances. *Machine Learning with Applications*, 4, 100031. <https://doi.org/10.1016/j.mlwa.2021.100031>
- Ali, F., Kwak, K.-S., & Kim, Y.-G. (2016). Opinion mining based on fuzzy domain ontology and Support Vector Machine: A proposal to automate online review classification. *Applied Soft Computing*, 47, 235–250. <https://doi.org/10.1016/j.asoc.2016.06.003>
- Clark, C., & Divvala, S. (2016). PDFFigures 2.0. *Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries*, 143–152. <https://doi.org/10.1145/2910896.2910904>
- CTAN Team. (n.d.). *What are TEX and its friends?* Retrieved May 8, 2022, from <https://www.ctan.org/tex>
- Deivalakshmi, S., Palanisamy, P., & Vishwanathan, G. (2013). A novel method for text and non-text segmentation in document images. *2013 International Conference on Communication and Signal Processing*, 255–259. <https://doi.org/10.1109/iccsp.2013.6577054>
- Deng, Y., Rosenberg, D., & Mann, G. (2019). Challenges in End-to-End Neural Scientific Table Recognition. *2019 International Conference on Document Analysis and Recognition (ICDAR)*, 894–901. <https://doi.org/10.1109/ICDAR.2019.00148>
- Ding, H., Chen, K., & Huo, Q. (2019). Compressing CNN-DBLSTM models for OCR with teacher-student learning and Tucker decomposition. *Pattern Recognition*, 96, 106957. <https://doi.org/10.1016/j.patcog.2019.07.002>
- Doğan, M. İ., Orman, A., Örkücü, M., & Örkücü, H. H. (2019). Çok gruplu sınıflandırma problemlerine regresyon analizi ve matematiksel programlama tabanlı yeni bir yaklaşım. *Gazi Üniversitesi Mühendislik-Mimarlık Fakültesi Dergisi*. <https://doi.org/10.17341/gazimmfd.571643>
- Kavasidis, I., Pino, C., Palazzo, S., Rundo, F., Giordano, D., Messina, P., & Spampinato, C. (2019). A Saliency-Based Convolutional Neural Network for Table and Chart Detection in Digitized Documents. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11752 LNCS, 292–302. https://doi.org/10.1007/978-3-030-30645-8_27
- Klatsky, S. (2003). WYSIWYG. *Aesthetic Surgery Journal*, 23(4), 274–275. [https://doi.org/10.1016/S1090-820X\(03\)00150-X](https://doi.org/10.1016/S1090-820X(03)00150-X)
- Li, M., Cui, L., Huang, S., Wei, F., Zhou, M., & Li, Z. (2019). *TableBank: A Benchmark Dataset for Table Detection and Recognition*. <http://arxiv.org/abs/1903.01949>
- Navada, A., Ansari, A. N., Patil, S., & Sonkamble, B. A. (2011). Overview of use of decision tree algorithms in machine learning. *2011 IEEE Control and System Graduate Research Colloquium*, 37–42. <https://doi.org/10.1109/ICSGRC.2011.5991826>
- Otsu, N. (1979). A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1), 62–66. <https://doi.org/10.1109/TSMC.1979.4310076>
- Pang, N., Yang, C., Zhu, X., Li, J., & Yin, X.-C. (2021). Global Context-Based Network with Transformer for Image2latex. *2020 25th International Conference on Pattern Recognition (ICPR)*, 4650–4656. <https://doi.org/10.1109/ICPR48806.2021.9412072>
- PRImA. (n.d.). Retrieved May 22, 2022, from <https://www.primaresearch.org/>
- Recommendation ITU-R BT.601-7. (2011, March). <https://www.itu.int/dmspubrec/itu-r/rec/bt/R-REC-BT.601-7-201103-1!!PDF-E.pdf>
- Safnuk, B., & Hu, G. (2018). Reconstructing LaTeX Source Files from Generated PDFs - a Neural Network Approach. *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, 890–895. <https://doi.org/10.1109/INDIN.2018.8472050>
- Shen, Z., Zhang, R., Dell, M., Lee, B. C. G., Carlson, J., & Li, W. (2021). *LayoutParser: A Unified Toolkit for Deep Learning Based Document Image Analysis*. <http://arxiv.org/abs/2103.15348>
- Uğuz, H. (2011). A two-stage feature selection method for text categorization by using information gain, principal component analysis and genetic algorithm. *Knowledge-Based Systems*, 24(7), 1024–1032. <https://doi.org/10.1016/j.knosys.2011.04.014>
- Wang, Z., & Liu, J. C. (2021). Translating math formula images to LaTeX sequences using deep neural networks with sequence-level training. *International Journal on Document Analysis and Recognition*, 24(1–2), 63–75. <https://doi.org/10.1007/s10032-020-00360-2>
- Wang, Z., Xu, Y., Cui, L., Shang, J., & Wei, F. (2021). *LayoutReader: Pre-training of Text and Layout for Reading Order Detection*. <http://arxiv.org/abs/2108.11591>
- Wang, Z., Yang, J., Jin, H., Shechtman, E., Agarwala, A., Brandt, J., & Huang, T. S. (2015). DeepFont: Identify Your Font from An Image. *Proceedings of the 23rd ACM International Conference on Multimedia*, 451–459. <https://doi.org/10.1145/2733373>
- Xu, C., Shi, C., Bi, H., Liu, C., Yuan, Y., Guo, H., & Chen, Y. (2021). A Page Object Detection Method Based on Mask R-CNN. *IEEE Access*, 9, 143448–143457. <https://doi.org/10.1109/ACCESS.2021.3121152>

- Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., & Zhou, M. (2020). LayoutLM: Pre-training of Text and Layout for Document Image Understanding. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 20, 1192–1200. <https://doi.org/10.1145/3394486.3403172>
- Zhong, X., Tang, J., & Yepes, A. J. (2019). *PubLayNet: largest dataset ever for document layout analysis*. <http://arxiv.org/abs/1908.07836>

How cite this article

Kazanc, M., Ensari, T. & Dagtekin, M. (2023). Converting Image Files to LaTeX Format using computer vision, natural language processing, and machine learning. *Acta Infologica*, 7(2), 253-266. <https://doi.org/10.26650/acin.1258719>