PROCEEDINGS OF THE YEREVAN STATE UNIVERSITY

Physical and Mathematical Sciences 2024, **58**(3), p. 79–87

Informatics

SYNTHETIC DOCUMENT GENERATION FOR THE TASK OF VISUAL DOCUMENT UNDERSTANDING

Kh. S. KHECHOYAN*

Chair of the Theory of Probability and Mathematical Statistics, YSU, Armenia

Solving the problem of document analysis using machine learning methods requires a large amount of labeled data. Such data is not always available, and if available, it only covers certain types of documents.

In this paper, we present a method for creating synthetic data that allows creating documents of any type by pre-defining the document components. By changing the arrangement of document components, text content, and visual elements using configurations, we create diverse and realistic datasets that mimic real documents. This method addresses the problem of the lack of labeled datasets and offers a flexible solution to improve the results of a machine learning model.

https://doi.org/10.46991/PYSUA.2024.58.3.079

MSC2020: 68T20.

Keywords: machine learning, data generation, document understanding.

Introduction. The task of document understanding involves analyzing and interpreting unstructured text data to extract meaningful information, enabling the conversion of raw text into structured data. The process includes several subtasks, including optical character recognition (OCR), key-information extraction (KIE), document question answering, layout parsing, and others. Labeled data are essential for training machine learning models in document understanding tasks, providing the necessary examples for models to learn from. Some of the tasks mentioned above are well-defined and have many high-quality annotated datasets available online.

Other tasks such as KIE and document question answering suffer from a lack of available data. There are several annotated datasets for the KIE task [1–7], but they solve specific tasks, and all cover a narrow set of classes. The main focus of these datasets is financial documents such as receipts and invoices.

* E-mail: khachatur.khechoyan@ysu.am

As labeled datasets are crucial for document understanding tasks, issues in their quality can significantly impact the performance of trained models. Several factors contribute to noise and inconsistencies in labeled data:

• *Human error*. Manual annotation is prone to mistakes, especially when dealing with complex document structures or ambiguous information. Differences in interpretation among annotators can lead to inconsistencies in labeling.

• *OCR errors*. Optical character recognition, while constantly improving, can still introduce errors in the extracted text. These errors can propagate into the labeled data, affecting the training process and model performance.

• *Domain-specific challenges*. Different document types (e.g., invoices, receipts, legal documents) have unique structures and language conventions. Labeled datasets often focus on specific domains, limiting their applicability to broader document understanding tasks.

These challenges associated with the quality and availability of labeled data underscore the need for alternative approaches to improve document understanding models. One such approach is the generation of synthetic data, that allows for the creation of large and diverse datasets with precise control over document characteristics and annotations. This approach offers several advantages, including the elimination of human error in annotations and the guarantee of perfect OCR accuracy, leading to cleaner training data and potentially more robust models.

In their work [8], the authors used similar ideas to generate historical documents in order to solve the task of counting records. The study showed that the well-designed synthetic data set can significantly boost the performance of the machine learning model. The idea of synthetic document generation is also discussed in [9] for the layout recognition task.

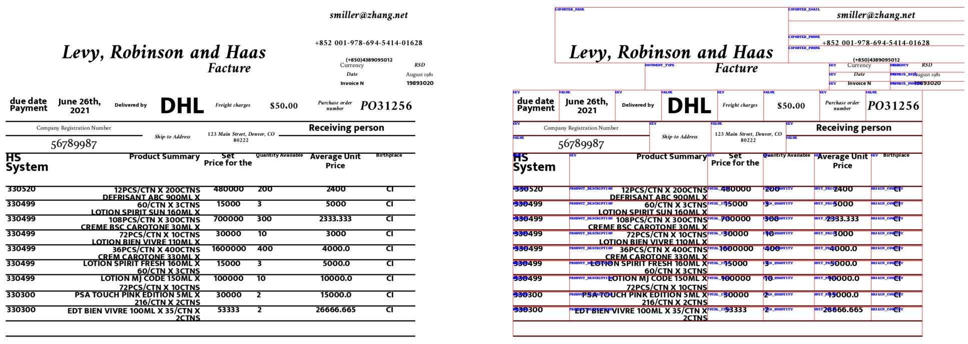
In this paper, we propose a programmatic method for synthetically generating complex documents with random layouts and texts. Furthermore, cover the implementation of text generation strategies that use large language models (LLMs) and random text generators [10] to create specific types of text data.

In the next sections, we will discuss the process of layout generation, text generation, implementation details, and baseline results on the generated data.

Synthetic Document Generation. In the following section, we will provide a comprehensive explanation of the synthetic document generation process.

From this moment, for simplicity, all the examples will be provided in the context of synthetic invoice generation (Fig. 1).

The primary objective of the synthetic data generation process outlined in this research is to facilitate customization. The intention is to provide users with the ability to generate content tailored to their specific needs and requirements and to provide them with full control over the generation process.


smiller@zhang.net

+852 001-978-694-5414-01628

Levy, Robinson and Haas
Facture

(+850)4389095012

Currency	RSD
Date	August 1981
Invoice N	19893020

due date Payment	June 26th, 2021	Delivered by	DHL	Freight charges	$50.00	Purchase order number	PO31256

Company Registration Number 56789987

Ship-to Address 123 Main Street, Denver, CO 80222

Receiving person

HS System	Product Summary	Set Price for the	Quantity Available	Average Unit Price	Birthplace
330520	12PCS/CTN X 200CTNS DEFRISANT ABC 900ML X	480000	200	2400	CI
330499	60/CTN X 3CTNS LOTION SPIRIT SUN 160ML X	15000	3	5000	CI
330499	108PCS/CTN X 300CTNS CREME BSC CAROTONE 30ML X	700000	300	2333.333	CI
330499	72PCS/CTN X 10CTNS LOTION BIEN VIVRE 110ML X	30000	10	3000	CI
330499	36PCS/CTN X 400CTNS CREM CAROTONE 330ML X	1600000	400	4000.0	CI
330499	LOTION SPIRIT FRESH 160ML X 60/CTN X 3CTNS	15000	3	5000.0	CI
330499	LOTION MJ CODE 150ML X 72PCS/CTN X 10CTNS	100000	10	10000.0	CI
330300	PSA TOUCH PINK EDITION 5ML X 216/CTN X 2CTNS	30000	2	15000.0	CI
330300	EDT BIEN VIVRE 100ML X 35/CTN X 2CTNS	53333	2	26666.665	CI

Fig. 1. An example of a synthetic invoice (left), and the ground truth information (right). Bounding boxes for words are not illustrated for simplicity.

Algorithm 1 Recursive Layout Generation

```
 1: procedure GENERATELAYOUT($element$)
 2:     if ISLEAF($element$) then
 3:         exit
 4:     else
 5:         $children \leftarrow$ SPLITELEMENT($element$)
 6:         for $child$ in $children$ do
 7:             GENERATELAYOUT($child$)
 8:         end for
 9:     end if
10: end procedure
11: procedure SPLITELEMENT($element$)
12:     $layoutConfig \leftarrow$ GETLAYOUTCONFIG($element$)
13:     $children \leftarrow$ DIVIDEELEMENT($element, layoutConfig$)
14:     return $children$
15: end procedure
16: function ISLEAF($element$)
17:     return CHECKIFLEAF($element$) ▷ User-defined logic to check if element is a
    leaf
18: end function
19: $document \leftarrow$ new $DocumentElement$
20: GENERATELAYOUT($document$)
```

Document Layout Generation. In this section, we discuss the layout sampling process in detail.

The layout is defined as a collection of the spatial positions of various elements in a given space. To construct a layout, we first need to define the elements that can occur in a layout. In the case of invoices, these elements could include a company logo, address, invoice number, list of items, etc. To construct a meaningful document, we need to be able to group some elements in a logical and visually appealing manner, which brings us to the method of layout generation. For example, some information needs to always be at the top of the document such as an invoice number, invoice date, company name, etc.

To implement this idea, we have adapted a recursive layout generation technique. The core concept is as follows. The process starts with a single “Document” element. Then, this element is split between several other elements defined by the configuration of the user. This process repeats for each new element until the leaf element is reached. The process can be described using a pseudocode 1. In the configuration, the occurrence of each element is modeled with the *Bernoulli* distribution. The split direction of the element also needs to be determined. Each element is split horizontally or vertically. After sampling the occurrence, the size of the element is modeled by sampling the ratio of the element from a *Uniform* distribution. This ratio is then used to calculate the exact size of the element. This process is highly configurable, and by varying the sizes and elements a very diverse set of documents can be generated. The sampling parameters are configured manually. In the Fig. 2, you can see the configuration for generating a commercial invoice layout.

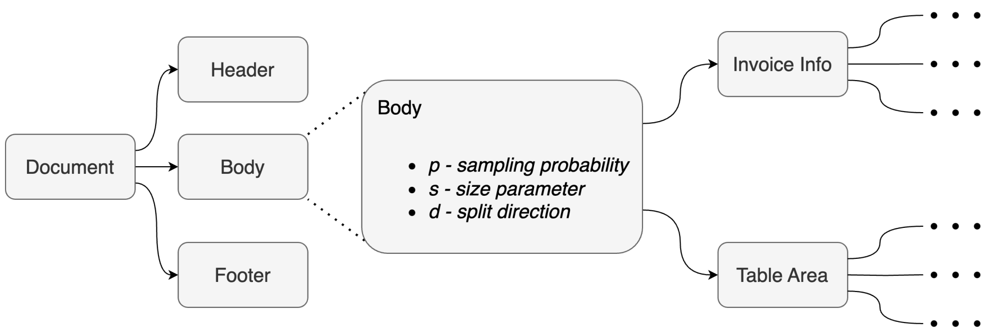


Fig. 2. Illustration of the layout configuration tree. Only a small portion is illustrated. All the elements have the parameters p, s and d.

Generic Text Data Generation. To generate textual data for simple cases, we utilized the Faker library [10]. Faker is a Python library that creates a large amount of high-quality synthetic data. It is intended to produce credible data in a variety of formats, including names, addresses, phone numbers, dates, and others.

Key-Value Data Generation. To generate diverse and realistic key-value pairs, simple techniques are not enough. First, the keys should be diverse and relevant at the same time. Moreover, the generated values should be correct and semantically consistent with the key. Recent advances in LLMs make this kind of data generation possible.

We introduced a setup, where the LLM of choice first generates examples of keys and then generates respective values. Using prompt engineering techniques, we achieved good performance and diversity of key-value pairs. Tab. 1 illustrates some of the generated key-value pairs.

Table 1

Some key-value pairs generated by LLM

Key	Value
Payment method	Credit card
Salesperson Name	Kelly Carter
Purchase order number	P031256
Payment due date	June 26th, 2021
Currency	EUR

Tabular Data Generation For tabular data, we can use any kind of dataframe, to fill the Table. The rendered Table will then be used in the line item detection task. If the user is also interested in the classification of columns into predefined classes, they can achieve it by defining the class of each column.

Text Rendering. To render the text in the given element, we use the Pillow package. Pillow is a Python package that allows one to manipulate images, and provides a rich functionality of text rendering. To diversify the rendered text, we randomly select a font from a collection of fonts found online. Then, it is necessary to fit the sampled text into the element region. To achieve this, we render the text using different font sizes. When the horizontal space ends, we try to start a new line and write from there. If there is a font size that can be used to fit the whole text in the region, without the text being too small, we choose it as the font size. If there are multiple such font sizes, we choose one randomly. If such a font size cannot be found, we cut the sampled text and use increasingly smaller portions until fitting the whole text. We have also implemented text alignment functionality to further diversify the rendered image. The vertical alignment is randomly sampled to be one of the "left", "center", and "right". Using the same logic, we sample the horizontal alignment. An example of a generated invoice and its corresponding ground truth can be found in Fig. 1.

Implementation Details. This section describes the specific software tools, hardware configurations, and parameter settings employed in the development and execution of the synthetic document generation framework.

Software and Libraries. The framework was primarily implemented using the Python programming language due to its ease of use and extensive ecosystem of libraries relevant to image processing and natural language processing tasks. The Faker [10] library was utilized to generate realistic placeholder text for common data types such as names, addresses, and dates. For rendering text onto the generated document images, the Pillow library was employed, providing control over different rendering parameters. To facilitate the creation of diverse and contextually relevant text elements, such as product descriptions and key-value pairs, a LLMs was integrated into the framework. To make the data more realistic, a private set of line items was used to fill the generated tables.

Hardware and Computational Resources. Experiments and evaluations were conducted on a single Nvidia A100 40GB GPU. Generation of 100 000 documents takes approximately 1hr, using multiprocessing with 50 processes.

Parameter Settings and Configurations. The framework provides extensive customization options to suit different types of documents and user preferences. For layout generation, users can specify the probability of occurrence for each document element and define the splitting behavior and spatial arrangement of these elements within the document layout. The text generation parameters allow users to select the appropriate generation methods for different text element types.

The flexible and user-centric design of the implementation allows users to adapt the system to generate a wide variety of synthetic documents customized to their specific requirements and use cases.

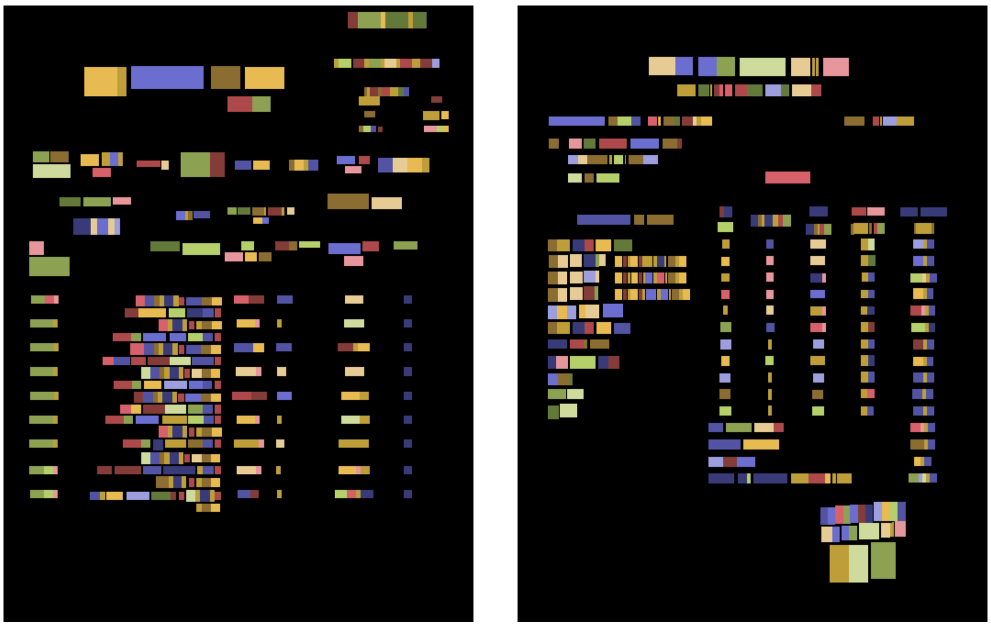


Fig. 3. Tokengrid representations of a synthetic document (left) and a real document (right).

Baseline Model for KIE Task. To assess the quality and complexity of the generated synthetic data, we employed a KIE model trained on a subset of the data and evaluated its performance on a separate held-out subset. For this purpose, we utilized the Tokengrid model [11], specifically designed to extract line items and entity values from a predefined set. In particular, Tokengrid leverages an intermediate document representation that preserves both textual and spatial information while remaining agnostic to the visual characteristics of the document image.

Fig. 3 illustrates the Tokengrid representation for both a synthetic and a real document.

This approach allows us to focus on the realism of the underlying layout and textual content rather than the visual fidelity of the generated document images.

We evaluated the performance of the model, using the token error rate (TER) metric, as presented in the Tokengrid paper [11].

Table 2

1-Token Error Rate (TER) of the predictions on some item-level fields

Field name	1-TER (%)
HS CODE	99.037
TOTAL ITEM	97.412
UNIT PRICE	98.018
ITEM QUANTITY	95.502
ORIGIN COUNTRY	98.084
ITEM NET WEIGHT	64.968
ITEM GROSS WEIGHT	81.007
PRODUCT DESCRIPTION	99.153

The results, displayed in Tab. 2, serve as baseline performance indicators for this dataset. The scores indicate that the dataset presents a sufficient level of complexity, suggesting the validity and quality of the generated data. More research is required to quantify the performance gains achieved by incorporating synthetic data into the training process of various KIE models.

Received 22.05.2024
Reviewed 03.01.2025
Accepted 17.01.2025

REFERENCES

1. Kardas M., Czapla P., et al. AxCell: Automatic Extraction of Results from Machine Learning Papers. *Proc. of the 2020 Conf. on Empirical Methods in Natural Language Processing (EMNLP)* (2020), 8580–8594. https://doi.org/10.18653/v1/2020.emnlp-main.692
2. Park S., Shin S., et al. CORD: A Consolidated Receipt Dataset for Post-OCR Parsing (2022).
3. Jaume G., Ekenel H.K., Thiran J.-P. FUNSD: A Dataset for Form Understanding in Noisy Scanned Documents (2019). https://doi.org/10.1109/ICDARW.2019.10029
4. Huang Z., Chen K., et al. ICDAR2019 Competition on Scanned Receipt OCR and Information Extraction. *2019 Int. Conf. on Document Analysis and Recognition (ICDAR)* (2021), 8580–8594. https://doi.org/10.1109/ICDAR.2019.00244

5. Stanisławek T., Graliński F., et al. Kleister: Key Information Extraction Datasets Involving Long Documents with Complex Layouts. *Lecture Notes in Computer Science* **12856** (2021), 428–444.
https://doi.org/10.1007/978-3-030-86549-8_36
6. Smock B., Pesala R., Abraham R. PubTables-1M: Towards Comprehensive Table Extraction from Unstructured Documents (2021).
http://dx.doi.org/10.1109/CVPR52688.2022.00459
7. Wang Z., Zhou Y., et al. VRDU: A Benchmark for Visually-rich Document Understanding. *Proc. of the ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining* (2023), 5184–5193.
https://doi.org/10.1145/3580305.3599929
8. Capobianco S., Marinai S. DocEmul: A Toolkit to Generate Structured Historical Documents (2017).
http://dx.doi.org/10.1109/ICDAR.2017.196
9. Raman N., Shah S., Veloso M. Synthetic Document Generator for Annotation-free Layout Recognition. *Pattern Recognition* **120** (2021), 108660.
https://doi.org/10.1016/j.patcog.2022.108660
10. Faraglia D., et al. Faker. [Software]. Retrieved from
https://github.com/joke2k/faker
11. Yeghiazaryan A., Khechoyan K., et al. Tokengrid: Toward More Efficient Data Extraction from Unstructured Documents. *IEEE Access* **10** (2022), 39261–39268.
https://doi.org/10.1109/ACCESS.2022.3164674

Խ. Ս. ԽԵՉՈՅԱՆ

ՍԻՆԹԵՏԻԿ ՓԱՍՏԱԹՂԹԵՐԻ ՍՏԵՂԾՈՒՄ ՎԻԶՈՒԱԼ ՓԱՍՏԱԹՂԹԵՐԻ ԸՆԿԱԼՄԱՆ ԽՆԴՐԻ ՀԱՄԱՐ

Փաստաթղթերի վերլուծության խնդիրը մեքենայական ուսուցման մեթոդներով լուծելու համար անհրաժեշտ են մեծ քանակի պիտակավորված տվյալներ: Այդպիսի տվյալներ ոչ միշտ են հասանելի, և հասանելի լինելու դեպքում ընգրկում են միայն հատուկ տիպի փաստաթղթեր:

Այս աշխատանքում ներկայացված է սինթետիկ տվյալների ստեղծման մեթոդ, որի շնորհիվ հնարավոր է ստեղծել ցանկացած տիպի փաստաթուղթ՝ նախապես սահմանելով փաստաթղթի բաղադրիչները: Կոնֆիգուրացիաների միջոցով փոփոխելով փաստաթղթերի բաղադրիչների դասավորությունը, տեքստային բովանդակությունը և վիզուալ տարրերը, մենք ստեղծում ենք բազմազան և իրատեսական տվյալների հավաքածուներ, որոնք նմանակում են իրական փաստաթղթերը: Այս մեթոդը լուծում է պիտակավորված տվյալների հավաքածուների սակավության խնդիրը և առաջարկում է ճկուն լուծում՝ բարելավելու մեքենայական ուսուցման մոդելի արդյունքները:

Х. С. ХЕЧОЯН

ГЕНЕРАЦИЯ СИНТЕТИЧЕСКИХ ДОКУМЕНТОВ ДЛЯ ЗАДАЧИ ВИЗУАЛЬНОГО ПОНИМАНИЯ ДОКУМЕНТОВ

Для решения задачи анализа документов методами машинного обучения необходимо большое количество размеченных данных. Такие данные не всегда доступны, а если и доступны, то охватывают только определенные типы документов.

В этой работе нами представлен метод создания синтетических данных, позволяющий создавать документы любого типа, предварительно определив компоненты документа. Изменяя расположение компонентов документов, текстовое содержание и визуальные элементы с помощью конфигураций, мы создаем разнообразные и реалистичные наборы данных, имитирующие реальные документы. Этот метод решает проблему нехватки размеченных наборов данных и предлагает гибкое решение для улучшения результатов модели машинного обучения.