USTC-iFLYTEK at DocILE: A Multi-modal Approach Using Domain-specific GraphDoc

Notebook for the DocILE Lab at CLEF 2023

Yan Wang[1], Jun Du[1,*], Jiefeng Ma[1], Pengfei Hu[1], Zhenrong Zhang[1] and Jianshu Zhang[2]

[1]NERC-SLIP, University of Science and Technology of China (USTC), No. 96, JinZhai Road, Hefei, Anhui, P.R.China.
[2]iFLYTEK Research, No.666 West Wangjiang Road Hefei, Anhui, China.

Abstract

With the development of digitalization in business, the automatic extraction of information from semi-structured business documents is becoming increasingly important. This paper introduces the methods we employed in two tasks of the DocILE competition. We utilize the GraphDoc model firstly pre-trained on provided invoice-domain documents to extract embeddings for text boxes and perform classification for each of them. Based on the classification results, we feed the embeddings into our proposed *Merger* module to aggregate separate text boxes into semantic instances and line items. Our approach achieved an AP result of 71.25% on the test set of Task 1 and a micro-F1 result of 75.93% on the test set of Task 2. Scripts and pre-trained models used in our experiments have been made publicly available at here[1].

Keywords

Document intelligence, Multi-modal, Domain-specific pre-train, Model ensemble

1. Introduction

Business documents are files that provide details related to a company's internal and external transactions. Despite the widespread adoption of digital business practices, many of them are still presented in unstructured or semi-structured formats, such as contracts, invoices, and reports, making it difficult to process automatically. And businesses often rely on manual data entry and processing, which can be time-consuming, error-prone, and expensive.

Automatic information extraction from business documents is a highly challenging task and the most immediate obstacle is the lack of suitable datasets, as mentioned by several authors [1, 2]. Although several publicly available datasets exist for document understanding, only a few of them are tailored to extracting information from business documents. And they are typically limited in size [3, 4, 5] and lack annotations for field-level locations [6, 7, 8]. Additionally, the variability in document structures and the scattered nature of the information make it difficult

[1]https://github.com/SPRATeam-USTC/DocILE-Competition

CLEF 2023: Conference and Labs of the Evaluation Forum, September 18–21, 2023, Thessaloniki, Greece

*Corresponding author.
yanwangsa@mail.ustc.edu.cn (Y. Wang); jundu@ustc.edu.cn (J. Du); jfma@mail.ustc.edu.cn (J. Ma); hudeyouxiang@mail.ustc.edu.cn (P. Hu); zzr666@mail.ustc.edu.cn (Z. Zhang); jszhang6@iflytek.com (J. Zhang)
http://staff.ustc.edu.cn/~jundu/ (J. Du); http://home.ustc.edu.cn/~xysszjs/ (J. Zhang)

to develop models that can accurately and efficiently extract key information from different types of documents.

To address these challenges, the Document Information Localization and Extraction (DocILE) competition [9] presents the DocILE dataset and benchmark [10], which includes two tasks: *Key Information Localization and Extraction* (KILE) [11] and *Line Item Recognition* (LIR) [11]. Unlike *Key Information Extraction* (KIE) [7, 8], which focuses on extracting information from documents, KILE requires additionally precise localization of the extracted information. The goal of LIR task is to extract a list of line items [12, 13], such as those commonly found in invoice goods and service tables and each item is represented by a set of key information such as name, quantity, and price. Therefore, for Task 1, the main objective of the experiment is to accurately localize key information (instances) of pre-defined categories in the document, while Task 2 builds on Task 1 to further group instances into line items. The two benchmark tasks use different primary metrics for evaluation. For Task 1, the Average Precision metric (AP) is used as the official evaluation metric, while for Task 2, the primary evaluation metric is the micro F1 score over all line item fields.

In this paper, we present our systems in the DocILE competition in both tasks. We employ the pre-trained GraphDoc [14] model to extract embeddings for text boxes and perform classification for each of them. Based on the classification results, the embeddings are fed into the proposed *Merger* module to aggregate the instances. And for Task 2, another *Merger* module is employed to group instances into line items.

2. Dataset

This competition provides three subsets, which are derived from the UCSF Industry Documents Library[1] and Public Inspection Files[2]: an annotated set of 6,680 real business documents, an unlabeled set of 932k real business documents, and a synthetic set of 100k documents with full task labels generated using proprietary document generation techniques. The annotated set is further split into training (5,180), validation (500), and test (1,000) sets, and includes annotations for 36 pre-defined categories in the KILE task and 19 others in the LIR task.

3. Methods

Due to the competition's prohibition on using external document datasets for training, we pre-trained several GraphDoc models under different configurations on the provided unlabeled set before fine-tuning the annotated set for both tasks. For both KILE and LIR tasks, we first classify text boxes into various categories using the output embeddings of the GraphDoc model, followed by the instance aggregation process handled by the proposed *Merger* module. And for Task 2, we further employ another *Merger* module to gather instances into line items. There are also some pre/post-processings according to the text contents and distances between text boxes. Finally, we also adopt a model ensemble strategy to further enhance the system performance.

[1] https://www.industrydocuments.ucsf.edu/
[2] https://publicfiles.fcc.gov/

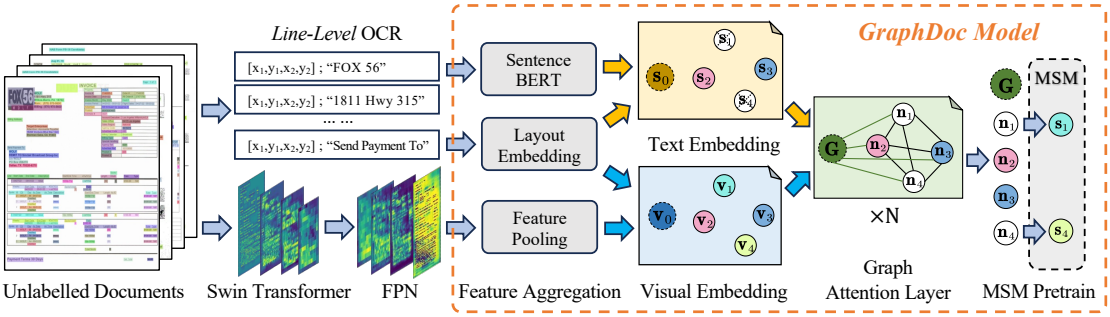


Figure 1: The pre-training schema. MSM is short for Masked Sentence Modeling task.

3.1. GraphDoc pre-training

GraphDoc is a document understanding model that applies a multimodal graph attention-based approach. Unlike previous pre-training models such as RoBERTa [15] and LayoutLM [16], GraphDoc treats the semantic regions of document images extracted by optical character recognition (OCR) as fundamental input elements instead of individual words. This novel approach allows the model to better capture the semantic information in documents and generalize across different types of business documents.

The pre-training process is illustrated in Figure 1. We rendered the unlabelled 932k PDF documents into images for separate pages. To guarantee the quality of the pre-training corpus, we remove those words whose confidence is lower than 0.5 and text lines whose average confidence is lower than 0.85. After that, only those pages which have more than 5 text lines and 50 characters will be reserved for pre-training. Finally, we can get 2,736,766 valid pages in total. The Masked Sentence Modeling (MSM) is used as the pre-training task for GraphDoc. Each sentence is randomly and independently masked, while its corresponding layout information is preserved. For the masked sentence, its text content is replaced with a special symbol named [MASK]. The training target is to predict the sentence embeddings of masked ones based on the sentence embeddings and the visual embeddings of others.

During pre-training, we freeze the parameters of Sentence-BERT [17] and jointly train the visual backbone and GraphDoc in an end-to-end fashion. GraphDoc contains 12 layers of graph attention blocks, with the hidden size set to 768 and the number of heads to 12. We pre-train the GraphDoc using Adam optimizer, with the learning rate of 5×10^{-5}. The learning rate is linearly warmed up over the first 10% steps and then linearly decayed. As for the MSM task, 15% of all input sentences are masked among which 80% are replaced by the [MASK] symbol, 10% are replaced by random sentences from other documents, and 10% remain the same.

We pre-train two different GraphDoc models with the parameter top-k for the graph attention layer set to 36 and 60, respectively. The pre-training is conducted on 8 Telsa A100 48GB GPUs with a batch size of 240. When top-k is set to 36, it takes around 13 hours to pre-train on 2.7 million unlabelled pages for 3 epochs. When changing the parameter top-k to 60, it takes around 19.5 hours to complete the pre-training for 4 epochs. More choices of top-k are not explored further due to time constraints.

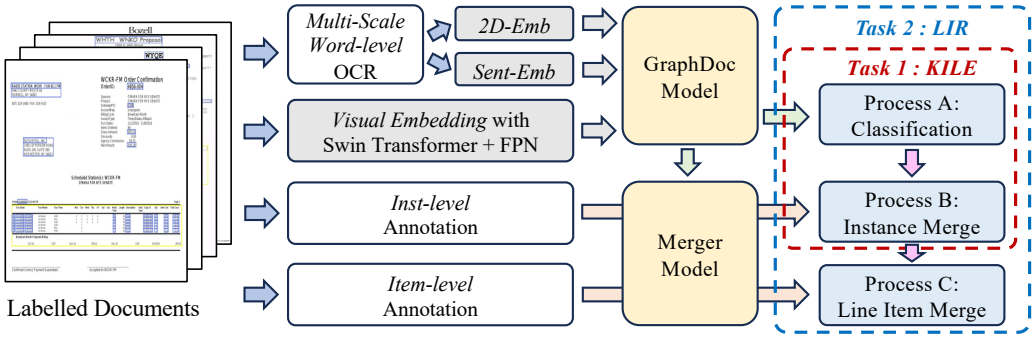


Figure 2: The finetuning schema.

3.2. Data pre-processing

Data pre-processing is one of the important steps in both tasks. When it comes to models such as RoBERTa, LayoutLM, and GraphDoc, OCR results are essential for accurately locating key information in documents. The competition organizers provide pre-computed OCR results using the DocTR library [18] with the DBNet [19] detector and the CRNN [20] recognition model, which are further post-processed by snapping word boxes to text to obtain snapped text boxes (B_{snap}).

To improve the recall rate of instances in the detection process, we convert the document PDF into image I and scale it by 1.25, 1.5, and 1.75 times the original size before text detection and recognition.

$$B_{\text{DocTR}}(I') = f_{\text{DocTR}}(I'),\ I' \in \{I_{1.25}, I_{1.5}, I_{1.75}\} \tag{1}$$

Then, we apply non-maximum suppression (NMS) with an IoU threshold of 0.3 to generate an intermediate set of text boxes B_{nms}:

$$B_{\text{nms}} = \text{NMS}(B_{\text{DocTR}}(I'), 0.3) \tag{2}$$

Lastly, we incorporate missed text boxes into the snapped text boxes provided by the competition organizers and eliminate erroneous large-sized text boxes generated by DocTR. Specifically, we add approximately 2% more text boxes to the existing set, while the percentage of removed text boxes is roughly 0.1% of the total count.

$$B_{\text{final}} = \text{Filter}(B_{\text{snap}} \cup \{b \mid b \in B_{\text{nms}},\ \text{IoS}(b, b') < 0.3,\ b' \in B_{\text{snap}}\}) \tag{3}$$

where Filter is a filtering operation that removes erroneous text boxes. Intersection over Smaller (IoS) is similar to IoU, but the score between two text boxes b_i and b_j is defined as:

$$\text{IoS}(b_i, b_j) = \frac{\text{area of intersection}}{\min(\text{area of } b_i, \text{area of } b_j)} \tag{4}$$

3.3. Model

The overview of our model is illustrated in Figure 2. Binary cross-entropy loss (BCELoss) is used for classification due to overlaps between instances and the possibility of one text box

belonging to multiple categories, which is consistent with the published baselines.

$$\mathcal{L}_{\text{cls}} = -\frac{1}{N}\sum_{i=1}^{N}\left(c_i \log \hat{c}_i + (1-c_i)\log(1-\hat{c}_i)\right) \tag{5}$$

where N is the number of classes, c_i is the true label for each class, and $\hat{c}_i$ is the classification logit for each class. To prevent memory errors caused by a large number of text boxes in some document images, we group the text boxes in each image after sorting them from left to right and top to bottom and feed them into the model in batches. Additionally, to avoid losing instances that may be split between different groups, a sliding window strategy is also adopted.

In contrast to the rule-based approach used in the baselines [10], we introduce a learnable *Merger* module to automatically aggregate text boxes into semantic instances based on their embeddings extracted by GraphDoc, as depicted in Figure 3. In Task 1, for each category, we calculate the attention score between text boxes of the same category to determine which text boxes should be merged into an instance. The attention score between the i-th text box and the j-th text box is calculated as follow:

$$a_{ij} = (\mathbf{E}_i \mathbf{W}^k)(\mathbf{E}_j \mathbf{W}^q)^\top \tag{6}$$

where $\mathbf{E}_i \in \mathbb{R}^d$, $\mathbf{E}_j \in \mathbb{R}^d$, $\mathbf{W}^k \in \mathbb{R}^{d\times d}$, $\mathbf{W}^q \in \mathbb{R}^{d\times d}$. d is the dimension of the embedding extracted by GraphDoc. The loss function for instance merging is as follow:

$$\mathcal{L}_{\text{inst}} = -\frac{1}{W*H}\sum_{i=1}^{W}\sum_{j=1}^{H}\left(M_{i,j}\log \hat{M}_{i,j} + (1-M_{i,j})\log(1-\hat{M}_{i,j})\right) \tag{7}$$

where W and H represent the width and height of the attention map, respectively, $M_{i,j}$ is the true label at position (i,j), and $\hat{M}_{i,j}$ is the attention score at position (i,j). Moreover, to avoid conflicts, we employ a method similar to NMS by using the mean attention scores of text boxes in the merged instance (NMS score). Specifically, if there are overlapping text boxes between two instances, the instance with the higher score will be selected.

Differing from Task 1, the input instances in Task 2 are restricted to a single class and are non-overlapping. Therefore, it's not necessary to merge instances separately for each class. All text boxes with pre-defined classes are input into the *Merger* module with the background texts excluded. The inputs of the second merge stage are instance-level embeddings using the average of all text boxes' embeddings in each instance, which can be formulated as:

$$\mathbf{E}_j^{\text{Inst}} = \frac{\sum_{i=1}^{n}\mathbf{E}_i}{n} \tag{8}$$

where $\mathbf{E}_j^{\text{Inst}}$ represents the embedding of the j-th instance, and n is the number of text boxes contained in the j-th instance. And $\mathcal{L}_{\text{item}}$, the loss function for line item merging, is similar to $\mathcal{L}_{\text{inst}}$.

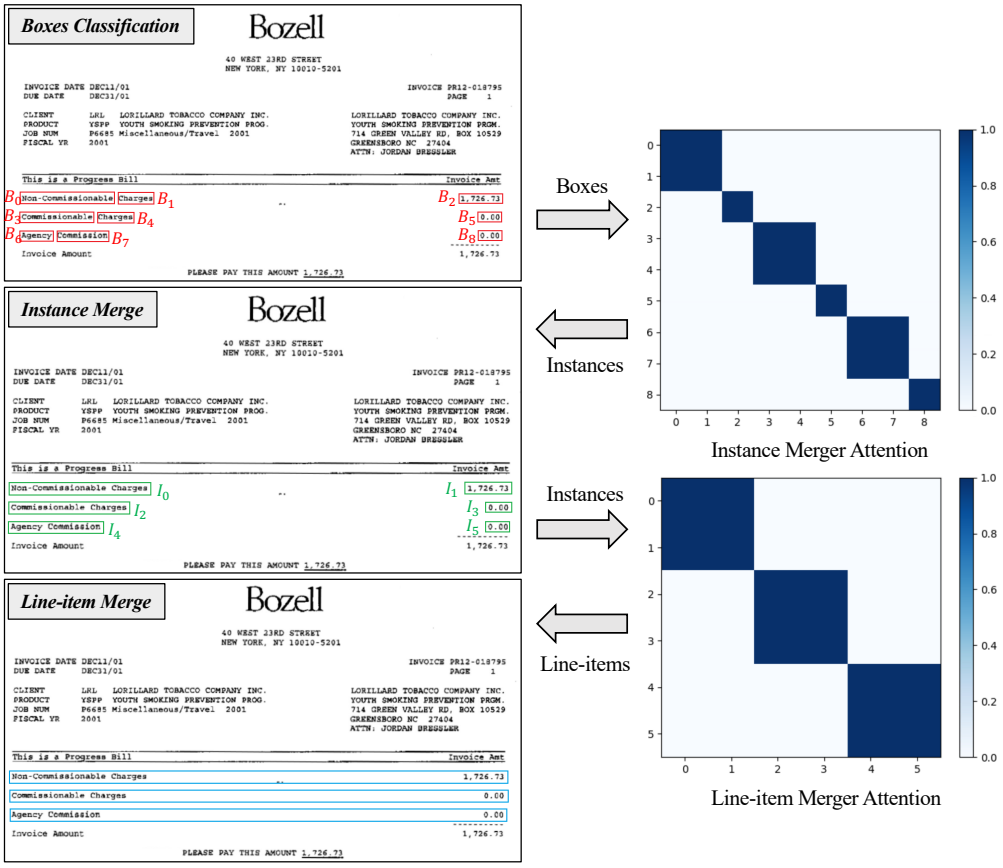


Figure 3: The merge results of instance merging and line item merging. The red, green, and blue boxes represent text boxes, instances, and line items, respectively. The marks near boxes are the indexes. And the attention maps on the right indicate which boxes should be merged.

3.4. Model ensembling

In order to further enhance model performance, we ensemble several models using different GraphDoc models to improve the text box classification accuracy. Specifically, after the fine-tuning stage, we select the top best-performing models under two different configurations as described in Section 3.1 for ensembling. There are four ensemble strategies depending on the ensemble method and the evaluation metric for model performance. Moreover, the micro F1 score for text box classification under each strategy is also calculated.

- strategy 1 : Averaging + official. The final probability scores used for classification are generated by averaging the probability scores of various models for each text box. Model selection criteria is based on the official evaluation metric of each task.
- strategy 2 : Averaging + F1. Employ the Averaging method as a model ensemble technique as strategy 1. But the selection criteria for models is based on their micro F1 score for text box classification.

- strategy 3 : Voting + official. The final classification results are voted by various models for each text box. Model selection criteria is based on the official evaluation metrics of each task.
- strategy 4 : Voting + F1. Employ the Voting method as a model ensemble technique and the selection criteria for models is based on their micro F1 scores for text box classification.

The number of models under different top-k configurations used for the ensemble in each strategy is determined based on the micro F1 score for text box classification of the ensemble models. As for the final selection of the four strategies, for Task 1, it is determined based on the micro F1 score of the ensemble models for text box classification. For Task 2, the official evaluation metric on the task is used to determine the optimal strategy. Lastly, the ensemble classification results are fed into the *Merger* module of the best-performing model prior to an ensemble.

3.5. Post-processing

Although our classification and *Merger* modules have achieved impressive results in both tasks, there still exists some challenges that cannot be addressed by the models alone. There are mainly two types of post-processing strategies performed: text box splitting and instance splitting. For example, instances of the "currency_code_amount_due" class are typically represented by boxes only containing the character "$", which are below the detection threshold of DocTR due to their small scales. We extract the text box containing the "$" symbol as the location of the instance if the text box belongs to this class and contains "$" in the text. Moreover, we observe that text box belonging to certain classes with "id" in their class names usually has the "#" symbol in its text, which is absent in the text of the corresponding instance. We remove the character "#" from the text and split the text box.

Furthermore, to prevent occasional errors in the *Merger* module, we devise rules to keep each text box as a separate instance if the distance between the text boxes in a single instance exceeds the predefined threshold. Specifically, for Task 1, we apply instance splitting when the minimum horizontal distance between text boxes is greater than 1.5 times the minimum width of text boxes and the maximum vertical distance is greater than 2 times the maximum height. For Task 2, instance splitting is applied when the minimum horizontal distance between text boxes is greater than twice the maximum width.

Additionally, for Task 1 whose official evaluation metric is AP, the score (confidence) in the final predicted result is critical since predictions are sorted by score from the highest to the lowest. And there are also three different score selection methods evaluated as described below:

- NMS score : As introduced in section 3.2, NMS score refers to the average attention score of text boxes within the same instance.
- Classification score : The average probability scores for the text boxes contained within the instance.
- NMS classification score : Average of Classification score and NMS score.

4. Resources

We use BCELoss with the transformers[3] framework to train the classification, instance merge, and line item merge tasks in a joint manner. The loss functions for two tasks are as follows:

$$\mathcal{L}_{\text{KILE}} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{inst}} \tag{9}$$

$$\mathcal{L}_{\text{LIR}} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{inst}} + \mathcal{L}_{\text{item}} \tag{10}$$

The initial learning rate is set to 5×10^{-5} and is linearly warmed up over the first 10% steps and then linearly decayed. Adam is used as the optimizer. The training is conducted on 2 Telsa V100 24GB GPUs with a batch size of 8. For Task 1, we train the model for 300-500 epochs, while for Task 2, we train the model for 500-1000 epochs.

5. Experiments and Results

For Task 1 and Task 2, we fine-tuned the model with different top-k configurations, and the variation curves of losses used in each task are shown in Figure 4.

Firstly, the instance detection recall rates on the validation set are compared in Table 1. When the IoS between a text box and an instance exceeds the threshold, we regard the text box as a part of the instance and assign it the same class label as the instance, indicating that the instance is detected. It can be observed that our data pre-processing significantly improves the recall rate of instance detection. Specifically, it achieves a recall rate of 0.995 at the threshold of 0.3, while the official method only achieves a recall rate of 0.979.

Table 1
The instance detection recall rates for the official OCR and our processed OCR

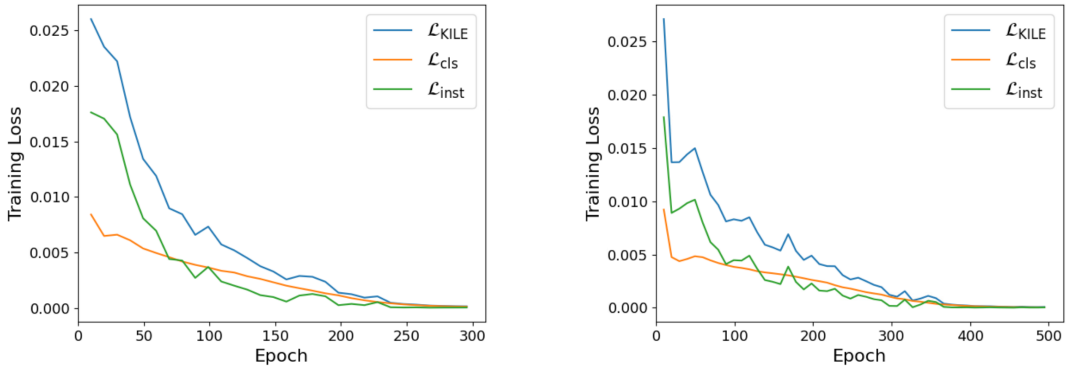
Config	Threshold				
	0.1	0.3	0.5	0.7	0.9
Official	0.982	**0.979**	0.968	0.944	0.885
Multi-scale	0.998	**0.995**	0.983	0.957	0.891

Table 2
Impact of different classification thresholds in the text box classification stage on the official evaluation metrics

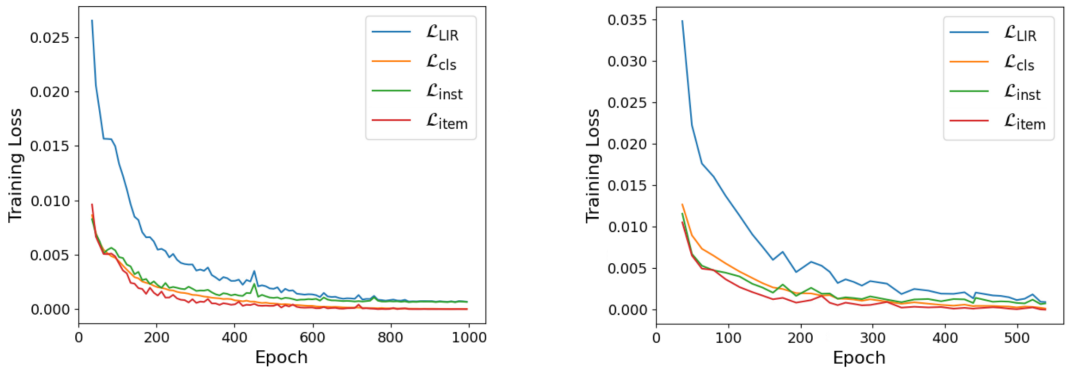
Task	Threshold				
	0.1	0.3	0.5	0.7	0.9
Task 1	**0.714**	0.701	0.690	0.676	0.643
Task 2	0.782	0.783	**0.784**	**0.784**	0.782

Secondly, for the selection of the classification threshold during the text box classification stage, Table 2 presents the performance of both tasks on the validation set at different thresholds.

[3]https://github.com/huggingface/transformers



(a) Task 1, top-k=36 (b) Task 1, top-k=60



(c) Task 2, top-k=36 (d) Task 2, top-k=60

Figure 4: The loss curves for Task 1 and Task 2 under different top-k configurations, trained using annotated training data.

It can be seen that the selection of the classification threshold can exert a substantial influence and the results of Task 1 can differ by up to 7 percent when the threshold changes from 0.1 to 0.9. For Task 1, a threshold of 0.1 is considered optimal, while for Task 2, a threshold of 0.5 is deemed appropriate.

Then, as discussed in Section 3.4, we ensemble multiple models using four schemes to enhance the text box classification accuracy and improve the model's performance. Based on the observations for Task 1, the accuracy of text box classification is positively correlated with the official evaluation metric. The *Merger* module performs well when the classification is correct. As shown in Table 3, when strategy 4 is chosen and the ensemble number is 11, the micro F1 score for text box classification on the validation set is the highest. For Task 2, two *Merger* modules are employed, making it difficult to ascertain the performance of each module. To address this issue, the selection of the final ensemble strategy is based on the official evaluation metric of Task 2 at the optimal ensemble quantity for each of the four strategies. As indicated in Table 3, strategy 1 is the final ensemble approach and the number of models used for ensembling is 11.

Table 3
The micro F1 scores for text box classification under different model ensembling strategies and model numbers and the official evaluation metric at the optimal ensemble quantity

Task	Strategy	Model Number				Official Evaluation Metric
		1	5	11	20	
Task 1	Strategy 1 (Averaging + official)	0.912	0.914	0.914	0.914	-
	Strategy 2 (Averaging + F1)	0.916	0.916	0.915	0.915	-
	Strategy 3 (Voting + official)	0.906	0.918	0.918	0.918	-
	Strategy 4 (Voting + F1)	0.914	0.919	**0.921**	0.920	-
Task 2	Strategy 1 (Averaging + official)	0.872	0.872	0.874	0.871	**0.785**
	Strategy 2 (Averaging + F1)	0.882	0.887	0.884	0.885	0.784
	Strategy 3 (Voting + official)	0.892	0.880	0.875	0.873	0.784
	Strategy 4 (Voting + F1)	0.902	0.891	0.888	0.887	0.761

Besides, Table 4 presents the performance of our final ensemble model with different score selection methods on the AP for Task 1 on the validation set. And Table 5 showcases the performance of our final ensemble model with NMS classification score in different post-processing approaches. Additionally, there are approximately 6% of instances generated through text box splitting in Task 1 and the percentage is around 2% in Task 2.

Table 4
The influence of different score selection methods on the AP for Task 1

Score Selection	AP
None	0.606
NMS score	0.625
Classification score	**0.740**
NMS classification score	**0.740**

Table 5
Post-processing results on official evaluation metrics on the validation set

Instance Splitting	Text Box Splitting	Task 1	Task 2
-	-	0.672	0.769
✓	-	0.677	0.770
✓	✓	**0.740**	**0.785**

Furthermore, ablation experiments are also conducted to summarize and justify each process as shown in Table 6. Finally, we evaluate the performance of our ensemble model on the test sets and the model achieves an AP of 71.25% for Task 1 and a micro F1 score of 75.93% for Task 2 as shown in Table 7.

Table 6
The results of ablation experiments on pre-processing, post-processing and model ensembling on the validation set

Pre-processing	Post-processing	Model Ensembling	Task 1		Task 2	
			AP	F1	AP	F1
-	-	-	0.638	0.700	0.577	0.755
✓	-	-	0.643	0.702	0.594	0.766
✓	✓	-	0.714	0.750	0.615	0.784
✓	✓	✓	**0.740**	0.769	0.619	**0.785**

Table 7
The results on the test set

Task	AP	F1	P	R
Task 1	**71.25%**	74.25%	71.41%	77.31%
Task 2	57.89%	**75.93%**	80.82%	71.60%

6. Conclusion and Future Work

This paper presents the model and methods we used in the DocILE competition. We utilize the GraphDoc model to extract embeddings for each text box and perform classification for each of them. Based on the classification results, we feed the embeddings into our proposed *Merger* module to aggregate the instances and line items. To further improve the model performance, we also conduct model ensembling and some post-processing operations. The results of the validation and test sets demonstrate the effectiveness of our approach in both KILE and LIR tasks. As mentioned, our approach still requires manual rule-making to address some errors. In the future, we will continue to explore ways to automatically solve these issues.

References

[1] P. Dhakal, M. Munikar, B. Dahal, One-shot template matching for automatic document data capture, in: 2019 Artificial Intelligence for Transforming Business and Society (AITB), volume 1, IEEE, 2019, pp. 1–6.

[2] F. Krieger, P. Drews, B. Funk, T. Wobbe, Information extraction from invoices: a graph neural network approach for datasets with high layout variety, in: Innovation Through Information Systems: Volume II: A Collection of Latest Research on Technology Issues, Springer, 2021, pp. 5–20.

[3] H. Sun, Z. Kuang, X. Yue, C. Lin, W. Zhang, Spatial dual-modality graph reasoning for key information extraction, arXiv preprint arXiv:2103.14470 (2021).

[4] E. Medvet, A. Bartoli, G. Davanzo, A probabilistic approach to printed document understanding, International Journal on Document Analysis and Recognition (IJDAR) 14 (2011) 335–347.

[5] J. Wang, C. Liu, L. Jin, G. Tang, J. Zhang, S. Zhang, Q. Wang, Y. Wu, M. Cai, Towards robust

visual information extraction in real world: New dataset and novel solution, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, 2021, pp. 2738–2745.
[6] Ł. Borchmann, M. Pietruszka, T. Stanislawek, D. Jurkiewicz, M. Turski, K. Szyndler, F. Graliński, Due: End-to-end document understanding benchmark, in: Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2), 2021.
[7] Z. Huang, K. Chen, J. He, X. Bai, D. Karatzas, S. Lu, C. Jawahar, Icdar2019 competition on scanned receipt ocr and information extraction, in: 2019 International Conference on Document Analysis and Recognition (ICDAR), IEEE, 2019, pp. 1516–1520.
[8] T. Stanisławek, F. Graliński, A. Wróblewska, D. Lipiński, A. Kaliska, P. Rosalska, B. Topolski, P. Biecek, Kleister: key information extraction datasets involving long documents with complex layouts, in: Document Analysis and Recognition–ICDAR 2021: 16th International Conference, Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part I, Springer, 2021, pp. 564–579.
[9] Š. Šimsa, M. Uřičář, M. Šulc, Y. Patel, A. Hamdi, M. Kocián, M. Skalický, J. Matas, A. Doucet, M. Coustaty, D. Karatzas, Overview of DocILE 2023: Document Information Localization and Extraction, in: A. Arampatzis, E. Kanoulas, T. Tsikrika, S. Vrochidis, A. Giachanou, D. Li, M. Aliannejadi, M. Vlachos, G. Faggioli, N. Ferro (Eds.), Proceedings of the Fourteenth International Conference of the CLEF Association (CLEF 2023), LNCS Experimental IR Meets Multilinguality, Multimodality, and Interaction., 2023.
[10] Š. Šimsa, M. Šulc, M. Uřičář, Y. Patel, A. Hamdi, M. Kocián, M. Skalický, J. Matas, A. Doucet, M. Coustaty, D. Karatzas, DocILE Benchmark for Document Information Localization and Extraction, in: 17th International Conference on Document Analysis and Recognition, ICDAR 2021, San José, California, USA, August 21–26, 2023, Lecture Notes in Computer Science, Springer, 2023.
[11] M. Skalickỳ, Š. Šimsa, M. Uřičář, M. Šulc, Business document information extraction: Towards practical benchmarks, in: Experimental IR Meets Multilinguality, Multimodality, and Interaction: 13th International Conference of the CLEF Association, CLEF 2022, Bologna, Italy, September 5–8, 2022, Proceedings, Springer, 2022, pp. 105–117.
[12] O. Bensch, M. Popa, C. Spille, Key information extraction from documents: evaluation and generator, arXiv preprint arXiv:2106.14624 (2021).
[13] T. I. Denk, C. Reisswig, Bertgrid: Contextualized embedding for 2d document representation and understanding, arXiv preprint arXiv:1909.04948 (2019).
[14] Z. Zhang, J. Ma, J. Du, L. Wang, J. Zhang, Multimodal pre-training based on graph attention network for document understanding, IEEE Transactions on Multimedia (2022).
[15] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, arXiv preprint arXiv:1907.11692 (2019).
[16] Y. Xu, M. Li, L. Cui, S. Huang, F. Wei, M. Zhou, Layoutlm: Pre-training of text and layout for document image understanding, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 1192–1200.
[17] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, arXiv preprint arXiv:1908.10084 (2019).
[18] Mindee, doctr: Document text recognition, https://github.com/mindee/doctr, 2021.

[19] M. Liao, Z. Wan, C. Yao, K. Chen, X. Bai, Real-time scene text detection with differentiable binarization, in: Proceedings of the AAAI conference on artificial intelligence, volume 34, 2020, pp. 11474–11481.
[20] B. Shi, X. Bai, C. Yao, An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition, IEEE transactions on pattern analysis and machine intelligence 39 (2016) 2298–2304.