DocQueryNet: Value Retrieval with Arbitrary Queries for Form-like Documents

Mingfei Gao*, Le Xue*, Chetan Ramaiah, Chen Xing, Ran Xu, Caiming Xiong
Salesforce Research, Palo Alto, USA
{mingfei.gao, lxue, cramaiah, cxing, ran.xu, cxiong}@salesforce.com

Abstract

We propose, DocQueryNet, a value retrieval method with arbitrary queries for form-like documents to reduce human effort of processing forms. Unlike previous methods that only address a fixed set of field items, our method predicts target value for an arbitrary query based on the understanding of the layout and semantics of a form. To further boost model performance, we propose a simple document language modeling (SimpleDLM) strategy to improve document understanding on large-scale model pre-training. Experimental results show that DocQueryNet outperforms previous designs significantly and the SimpleDLM further improves our performance on value retrieval by around 17% F1 score compared with the state-of-the-art pre-training method. *Code is available here*.

1 Introduction

Form-like documents are very commonly used in business workflows. However, tremendous forms are still processed manually everyday. When humans need to extract some relevant information from a form-like document, they proceed as conducting a value retrieval task with some text queries. For example in Figure 1, when humans process a form, they usually have a description of the information (query) that they want to extract (e.g., total page number). Then, they examine the form (usually an image or a PDF) carefully to locate the key (e.g., *NUMBER OF PAGES INCLUDING COVER SHEET* in Figure 1) that is most semantically similar to the query and finally infer the target value based on the localized key. This manual process costs a large amount of human efforts as the number of forms and queries increases. Automating information extraction from forms is important to alleviate this problem.

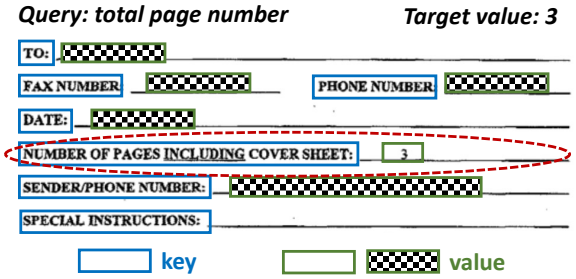

Figure 1: Illustration of the value retrieval with an arbitrary query. A user obtains the target value from a document based on a query of interest. Some values are decorated with a mosaic for privacy purposes.

Existing methods formulate the problem as sequence labeling (Xu et al., 2020b) or field extraction (Gao et al., 2022), where they define a fixed set of items of interest (referred as fields) and train models that only extract values of the pre-defined fields. There are at least two limitations of this formulation. First, forms are very diverse and it is impossible to cover all the items of interest using a fixed set of fields. Second, their models are very domain-specific and hard to be utilized for different form types. For example, an invoice field extractor may not be able to process resumes, since different fields are expected for these two form types.

To handle diverse queries with a unified model, we formulate the problem as value retrieval with arbitrary queries for form-like documents. Under such task formulation, users can extract values from a form by presenting variants of the corresponding keys as queries. We also set up a benchmark for the task by introducing a simple yet effective method, i.e., DocQueryNet. DocQueryNet takes an arbitrary query phrase and all the detected optical character recognition (OCR) words with their locations in the form as inputs. Then, we model the interactions between the query and the detected words from the document using a transformer-based architecture. The training objective encourages the

*Mingfei and Le contributed equally.

Proceedings of the 29th International Conference on Computational Linguistics, pages 2141–2146
October 12–17, 2022.

matching of the positive query-value pairs and discourages that of the negative ones. To further boost the performance, we present SimpleDLM, a simple document pre-training strategy that makes it more flexible to learn local geometric relations between words/tokens compared to existing pre-trained models. Experimental results show that DocQueryNet outperforms the baselines by a large margin under different settings. When initializing using SimpleDLM, our method is further improved largely by about 17% F1 score compared to initializing using a state-of-the-art (SOTA) pre-trained model, i.e., LayoutLM (Xu et al., 2020b).

2 Related Work

2.1 Information Extraction from Documents

Information extraction from documents is crucial for improving the efficiency of form processing and reducing human labor. Information extraction is often formulated as a field extraction task. Palm et al. 2019 propose an invoice field extractor by using an Attend, Copy, Parse architecture. Majumder et al. 2020 present a field-value pairing framework that learns the representations of fields and value candidates in the same feature space using metric learning. Nguyen et al. 2021 propose a span extraction approach to extract the start and end of a value for each field. Gao et al. 2022 introduce a field extraction system that can be trained with large-scale unlabeled documents. Xue et al. 2021 propose form transformations to mimic the variations of forms for robustness evaluation. Unlike previous methods that aim to extracting values for a pre-defined set of fields, our method targets at retrieving values for arbitrary queries.

2.2 Document Pre-training

Document pre-training is an effective strategy to improve document-related downstream tasks. Xu et al. 2020b propose LayoutLM that models interaction between texts and layouts in scanned documents using masked language modeling and image-text matching. Later, LayoutLMv2 (Xu et al., 2020a) introduces a spatial-aware self-attention mechanism to improve learning relative positional relationship among different text blocks. Most recently, Appalaraju et al. 2021 propose DocFormer that encourages the interaction between image and text modalities by adding an image reconstruction task. The existing pre-training methods perform well when applied to downstream tasks such as document classification and token sequence labeling. However, all of the above methods include the absolute 1-D positional embedding (the so called reading order) of the tokens in the inputs. Although this 1-D embedding is a helpful prior knowledge for a holistic understanding of a document, it hinders a model from learning rich geometric relationship among tokens, thus is not beneficial to our value retrieval task, where local geometric relationship between words is essential for prediction. Our method uses permutation-invariant positional encoding to improve model performance.

3 Our Approach

Problem Formulation. The inputs to the system are an expected key phrase as the query, $\mathbf{Q}_i$, and a document $\mathbf{D}_i$. $\mathbf{D}_i$ is represented using a set of OCR words $\{w_{i1}, w_{i2}, ..., w_{iN}\}$ and their bounding-box locations $\{b_{i1}, b_{i2}, ..., b_{iN}\}$ in the document. The input query phrase is tokenized to $\mathbf{Q}_i = \{q_{i1}, q_{i2}, ..., q_{iM}\}$. A value retrieval system reads the document and understands the layout and semantics. The goal is to pick a phrase from the OCR words as the value $\mathbf{V}_i$ for the input query, $\mathbf{Q}_i$. Since the modeling is within one document, we will omit the subscript i for simplicity.

3.1 Value Retrieval with Arbitrary Queries

To the best of our knowledge, there are no existing methods explicitly address value retrieval with arbitrary queries from scanned forms. However, it is possible to apply the recent pairing-based field extraction methods (Majumder et al., 2020; Nguyen et al., 2021) to this task with some modification. In previous design, they first embed fields using a text encoder, extract OCR word representation by modeling the interaction of OCR words using self-attention and then conduct the field-value pairing. To accommodate arbitrary queries, we can simply replace their field (in a fixed set) embedding with the embedding of an arbitrary query as shown in Figure 2 (previous method). There are two obvious drawbacks of these methods: (1) they model the interaction between query and OCR words in a shallow way using only simple fully connected layers and (2) they require an additional model of text encoder for query embedding which introduces extra computational cost. We set this previous design as our baseline. The implementation details are shown in the appendix.

In contrast, our DocQueryNet utilizes a unified

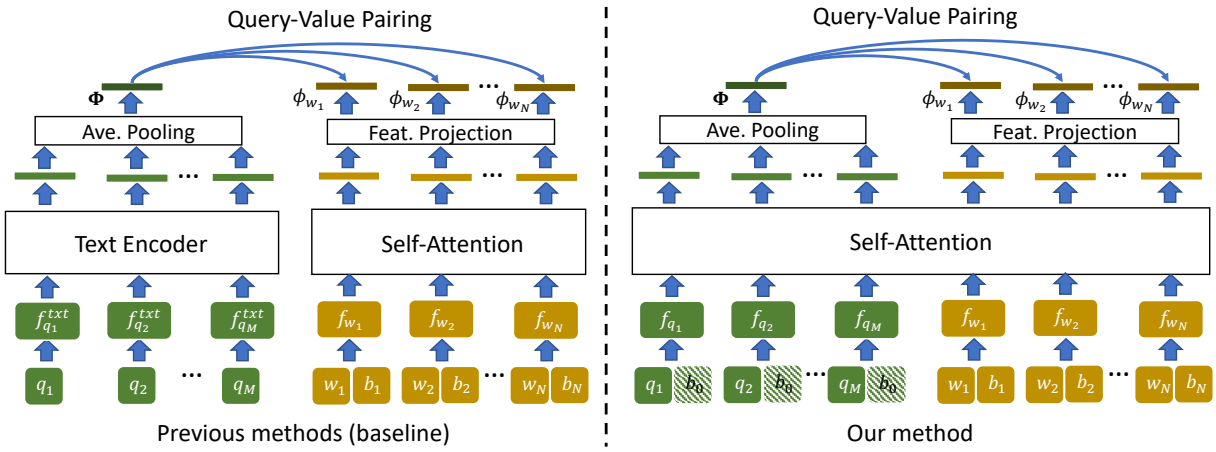

Figure 2: Comparison of previous methods and our method (see details in Section 3).

model to deeply model interactions among the query words and the OCR words. The direct inputs are a query $\{q_1, q_2, ..., q_M\}$ and OCR words $\{w_1, w_2, ..., w_N\}$ associated with their locations $\{b_1, b_2, ..., b_N\}$. We use a fixed dummy location b_0 for each query. Each query/OCR word is embedded as $f^{txt}_{w_j} \in \mathbb{R}^d$ and its location is encoded as $f^{loc}_{b_j} \in \mathbb{R}^d$, where d indicates the length of each vector (see Section 3.2 for details). The final embedding of each word (e.g., $\mathbf{f}_{q_j}$ for a query word or $\mathbf{f}_{w_j}$ for an OCR word) is the summation of its word embedding and location embedding.

A transformer is used to model the interactions among $\{\mathbf{f}_{q_1}, \mathbf{f}_{q_2}.., \mathbf{f}_{q_M}\}$ and $\{\mathbf{f}_{w_1}, \mathbf{f}_{w_2}, ..., \mathbf{f}_{w_N}\}$ via L self-attention layers. In the l^{th} layer, the hidden representation of the j^{th} token is updated following

$$\mathbf{h}^l_j = \text{Softmax}(\frac{\mathbf{h}^{l-1}_j \mathbf{H}^{l-1^T}}{\sqrt{d_h}}) \cdot \mathbf{H}^{l-1}, \quad (1)$$

where $\mathbf{H}^{l-1}$ indicates the representation of all tokens from the $(l-1)^{th}$ layer and d_h denotes the length of the hidden feature $\mathbf{h}^l_j$. $\mathbf{H}^0 = \{\mathbf{f}_{q_1}, \mathbf{f}_{q_2}, ..., \mathbf{f}_{q_M}, \mathbf{f}_{w_1}, \mathbf{f}_{w_2}..., \mathbf{f}_{w_N}\}$. In the final layer, $\mathbf{H}^L = \{\hat{\mathbf{f}}_{q_1}, \hat{\mathbf{f}}_{q_2}, ..., \hat{\mathbf{f}}_{q_M}, \hat{\mathbf{f}}_{w_1}, \hat{\mathbf{f}}_{w_2}..., \hat{\mathbf{f}}_{w_N}\}$. The self-attention mechanism deeply models the interactions among query words and OCR words.

We obtain the query phrase representation, Φ, using average pooling over $\{\hat{\mathbf{f}}_{q_1}, \hat{\mathbf{f}}_{q_2}, ..., \hat{\mathbf{f}}_{q_M}\}$. The final representation of w_j is obtained by $\phi_{w_j} = FC(\hat{\mathbf{f}}_{w_j})$, where FC indicates a fully connected layer. The likelihood score of w_j being a part of the target value of the query is obtained in Equation 2

$$\mathbf{s}_j = \text{Sigmoid}(\phi_{w_j} \cdot \Phi^T). \quad (2)$$

Our model is expected to learn (1) the layout and semantics of the document (2) the mapping between the input query phrase and the actual key texts in the document and (3) the geometric and semantic relationship between the key and value.

Model optimization. During training, each w_j is associated with a ground-truth label $y_j \in \{0, 1\}$, where 1 means this word is a part of the target value and 0 means it is not. The model is optimized using binary cross entropy loss as $\frac{1}{N}\sum_{j=1}^{N} y_j \text{logs}_j + (1 - y_j)(1 - \text{logs}_j)$.

Inference. Since the target value may contain multiple words, we group nearby OCR words horizontally as value candidates, $\{g_1, g_2, ..., g_D\}$, based on their locations using DBSCAN algorithm (Ester et al., 1996), where D is the number of grouped candidates. The value score of each candidate, g_r, is the maximum of all its covered words as $\mathbf{S}_r = \max_{w_j \in g_r} \mathbf{s}_j$, where, $w_j \in g_r$ indicates the OCR word w_j is a part of the grouped g_r. The value candidate with the highest score is used as the value prediction.

3.2 SimpleDLM for Document Pre-training

We introduce a Simple Document Language Modeling (SimpleDLM) method to encourage the understanding of the geometric relationship among words during pre-training.

The inputs to our pre-trained model are the OCR words $\{w_1, w_2, ..., w_N\}$ associated with their locations $\{b_1, b_2, ..., b_N\}$. The word/location embedding protocol and the transformer structure of our pre-trained model are the same as those of our value retrieval model such that the pre-trained model can be directly used for initializing the parameters of the value retrieval model. Specifically, the word embedding, $f^{txt}_{w_j}$, is constructed by using a simple

look up table. Previous works (Xu et al., 2020b,a; Appalaraju et al., 2021) require the input text sorted in the reading order, so that they can process texts of the document in a similar way of processing languages. They leverage this prior knowledge by adding the ranking of each word as the 1-D positional embedding in the final location embedding, $f^{loc}_{b_j}$. This 1-D embedding provides a holistic view of the geometric relationship of words. However, it introduces extra dependence on the OCR engines (most SOTA OCR engines do not have the capability of sorting detected OCR words in the reading order), and it also restricts the model from learning local geometric relations between words in a flexible way. To encourage a model to better learn the local geometric relations, we exclude the 1-D positional embedding and only encode the 2-D bounding-box location (top-left, bottom-right, width and height) of each word using a lookup table. For simplicity, we only use the masked language modeling as the pre-training objective. We show in Section 4 that our method improves the SOTA largely by using this simple pre-training strategy.

4 Experiments

The following datasets are used in our experiments.

IIT-CDIP (Lewis et al., 2006) is a large-scale unlabeled document dataset that contains more than 11 million scanned images. Following prior works (Xu et al., 2020b,a; Appalaraju et al., 2021), our model is pre-trained using this dataset.

FUNSD (Jaume et al., 2019) is a commonly used dataset for spatial layout analysis. It contains 199 scanned forms with 9,707 semantic entities annotated, where 149 samples are for training and 50 for testing. The semantic linking annotations for all the key-value pairs are provided in the dataset.

INV-CDIP (Gao et al., 2022) is document dataset which contains 350 real invoices for testing. This dataset has key-value pair annotations for 7 commonly used invoice fields including *invoice_number*, *purchase_order*, *invoice_date*, *due_date*, *amount_due*, *total_amount* and *total_tax*. We evaluate our model using this test set.

Settings. By default the annotated key texts of each dataset is used as the queries. The location of the keys are not used. Models are pre-trained on IIT-CDIP and fine-tuned on the train set of FUNSD. More implementation details are in the appendix.

Evaluation Metric. We use F1 score to evaluate models. Exact string matching between our predicted values and the ground-truth ones is used to count true positive, false positive and false negative. If a query has multiple value answers, a prediction is counted as correct if it equals one of them.

Model	Pretrain	Precision	Recall	F1
Baseline	Bert	31.7	32.0	31.9
	LayoutLM	41.8	42.1	41.9
	SimpleDLM	56.0	56.5	56.3
Ours	Bert	35.1	35.4	35.3
	LayoutLM	43.6	43.9	43.8
	SimpleDLM	**60.4**	**60.9**	**60.7**

Table 1: Comparisons on FUNSD when our model and baseline use different pre-trained models. **Ours** indicates **DocQueryNet**.

Model	Query	Precision	Recall	F1
Baseline	Exact Key	33.5	31.5	32.5
Ours		**50.5**	**47.6**	**49.0**
Baseline	Field Name	6.0	5.6	5.7
Ours		**21.2**	**19.9**	**20.5**

Table 2: Comparison in the transfer setting. Models are trained on FUNSD and evaluated on INV-CDIP. SimpleDLM is used for both methods. **Ours** indicates **DocQueryNet**.

Experimental Results. The comparisons between our DocQueryNet and the baseline when pre-trained using different approaches are shown in Table 1. The performance of both methods are improved largely with SimpleDLM. For the baseline, the F1 score is improved by 14.4% when replacing the LayoutLM with our SimpleDLM as the pre-trained model. Similarly, using SimpleDLM increases the F1 score of our method by 16.9% compared to using LayoutLM. Our method outperforms the baseline by 3-4% using different pre-trained models.

Transferring ability to another dataset is important in real-world applications. We measure this ability by directly evaluating the trained models using FUNSD to the test set of INV-CDIP in Table 2. As shown, when transferring to a new dataset, the performance of both our method and the baseline drops, compared to the numbers in Table 1. When using the exact key as the query, our method largely surpasses the baseline by 16.5% in F1 score.

In practice, we may not assume the input queries match exactly with the actual keys shown in a form. Here, we experiment using the field names directly as the queries. Using field names is a more conve-

nient way, since users don't need to design different queries that match keys for different forms. However, field names are more abstract, which makes the problem more challenging. When using the abstract field name as a query, our method achieves 20.5% F1 which is 14.8% better than our baseline.

Discussion. We use a dummy box to serve as the location of a query word, since there are no actual boxes for the query words in practice. This choice may make our model lose the sequential ordering information of a query phrase. To investigate this effect, we conduct an experiment and leverage the ordering information of query words by adding different positional ids to different words of a query in our model. The results show that this change decreases our performance from 60.7% to 60.3% in F1 score on FUNSD and from 49.0% to 47.6% in F1 score on INV-CDIP. It may be because that queries with the same meaning might be represented with words in different orders. For example, *total amount* could be referred to as *amount total*. Ignoring the ordering could help improve our model's generalization ability.

5 Conclusion

We introduce DocQueryNet, a framework for value retrieval with arbitrary queries for form-like documents. It takes a query and the detected OCR words from a document as inputs, models their interactions and predicts the best value corresponding to the input query. We also present SimpleDLM as a pre-training strategy to boost performance. Experimental results show that our method significantly outperforms previous designs in different settings.

6 Broader Impacts

This work is introduced to automate the information extraction from forms to improve document processing efficiency. It has positive impacts such as reducing human labor. However, reducing human labor may also cause negative consequences such as job loss or displacement, particularly amongst low-skilled labor who may be most in need of gainful employment. The negative impact is not specific to this work, but should be addressed broadly in the field of AI research.

References

Srikar Appalaraju, Bhavan Jasani, Bhargava Urala Kota, Yusheng Xie, and R Manmatha. 2021. Docformer: End-to-end transformer for document understanding. *arXiv preprint arXiv:2106.11539*.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*.

Mingfei Gao, Zeyuan Chen, Nikhil Naik, Kazuma Hashimoto, Caiming Xiong, and Ran Xu. 2022. Field extraction from forms with unlabeled data. *ACL Workshop*.

Guillaume Jaume, Hazim Kemal Ekenel, and Jean-Philippe Thiran. 2019. Funsd: A dataset for form understanding in noisy scanned documents. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 2, pages 1–6. IEEE.

David Lewis, Gady Agam, Shlomo Argamon, Ophir Frieder, David Grossman, and Jefferson Heard. 2006. Building a test collection for complex document information processing. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 665–666.

Bodhisattwa Prasad Majumder, Navneet Potti, Sandeep Tata, James Bradley Wendt, Qi Zhao, and Marc Najork. 2020. Representation learning for information extraction from form-like documents. In *ACL*.

Tuan-Anh D Nguyen, Hieu M Vu, Nguyen Hong Son, and Minh-Tien Nguyen. 2021. A span extraction approach for information extraction on visually-rich documents. *arXiv preprint arXiv:2106.00978*.

Rasmus Berg Palm, Florian Laws, and Ole Winther. 2019. Attend, copy, parse end-to-end information extraction from documents. In *ICDAR*.

Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, et al. 2020a. Layoutlmv2: Multi-modal pre-training for visually-rich document understanding. *arXiv preprint arXiv:2012.14740*.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020b. 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*, pages 1192–1200.

Le Xue, Mingfei Gao, Zeyuan Chen, Caiming Xiong, and Ran Xu. 2021. Robustness evaluation of transformer-based form field extractors via form attacks. *arXiv preprint arXiv:2110.04413*.

A Appendix

A.1 Implementation Details

Our code is implemented using Pytorch. We used Tesseract[1] to extract OCR words from documents for IIT-CDIP and INV-CDIP. Since FUNSD provides an official OCR annotation, we use it directly. The total number of query words M and the OCR words are different for different queries and documents. We keep $M + N = 512$ and pad with 0s when needed. We follow LayoutLM-base [2] to setup the structure of our transformers. The fully connected layer used for feature projection has 768 units. Each document is rescaled to [1000, 1000] and the dummy location, b_0, is set to [0, 0, 1000, 1000]. Adam is used as the optimizer. During pre-training, the learning rate is $5e^{-5}$. During fine-tuning, the learning rate is set to $3e^{-5}$ with weight decay equals to 0.9. SimpleDLM is initialized by LayoutLM and pre-trained on IIT-CDIP using 8 Nvidia A100 GPUs with a batch size of 36 for one epoch. We use a single A100 GPU for fine-tuning, where the training batch size is 8 and the total number of epochs is 45. In our experiments, all the pre-trained models including Bert, LayoutLM and SimpleDLM are base models. The pre-training and fine-tuning of our method take about 10 hours and 2 hours, respectively.

To perform a fair comparison, our method and baseline adopt the same experimental settings except for the interaction strategy. The baseline has the same transformer architecture and the feature projection layer as our method. The transformer takes OCR words with locations as inputs and obtain ϕ_{w_j} for each word w_j. And then, the query-value pairing score $\mathbf{s}_j$ is obtained by measuring the distance between the query representation, Φ, and ϕ_{w_j}. The query representation is obtained by average pooling over word embeddings extracted from a pre-trained Bert model.

[1] https://github.com/tesseract-ocr/tesseract (Apache License 2.0)

[2] https://github.com/microsoft/unilm/tree/master/layoutlm/deprecated (MIT License)