FieldSwap: Data Augmentation for Effective Form-Like Document Extraction

Jing Xie*, James B. Wendt*, Yichao Zhou*, Seth Ebner†, Sandeep Tata*
* *Google Research*
Mountain View, CA, USA
† *Johns Hopkins University*
Baltimore, MD, USA
*{lucyxie, jwendt, yichaojoey, tata}@google.com, †seth@cs.jhu.edu

***Abstract*—Extracting structured data from visually rich documents like invoices, receipts, financial statements, and tax forms is key to automating many business workflows. However, building extraction models in this domain often demands a large collection of high-quality training examples. To address this challenge, we introduce FieldSwap, a novel data augmentation technique specifically designed for such extraction problems. FieldSwap generates synthetic training examples by replacing key phrases indicative of one field with those corresponding to another. Our experiments on five diverse datasets demonstrate that incorporating FieldSwap-augmented data into the training process can enhance model performance by 1–11 F1 points, particularly when dealing with limited training data (10–100 documents). Additionally, we propose algorithms for automatically inferring key phrases from the training data. Our findings indicate that FieldSwap is effective regardless of whether key phrases are manually provided by human experts or inferred automatically.**

***Index Terms*—Date Efficiency, Data Augmentation, Information Extraction**

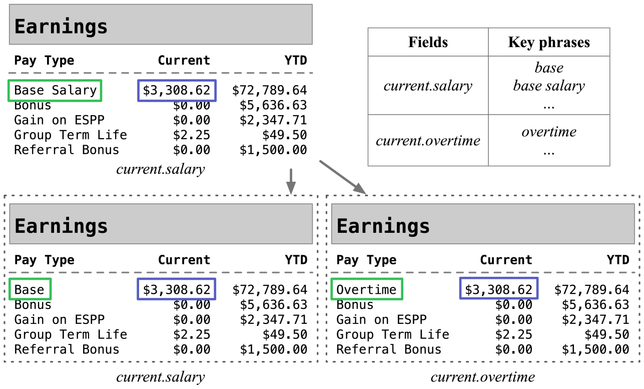
I. Introduction

Visually rich documents like invoices, receipts, paystubs, insurance statements, and tax forms are pervasive in business workflows. Processing these documents continues to involve manually extracting relevant structured information, which is both tedious and error-prone. Consequently, several recent papers have tackled the problem of automatically extracting structured information from such documents [1]–[7]. Given a target document type with an associated set of fields of interest, as well as a set of human-annotated training documents, these systems learn to automatically extract the values for these fields from unseen documents of the same type.

While recent models have shown impressive performance [8]–[11], a major hurdle in the development of high-quality extraction models is the large cost of acquiring and annotating training documents. In this paper, we examine the question of improving the data efficiency for this task especially when only a small number of labeled training documents is available. One approach to address this challenge is data augmentation. Data augmentation techniques can be used to artificially increase the size of the training dataset, which can help to improve the generalization performance of the model.

However, we found that conventional text augmentation methods such as synonym replacement, random swap, random deletion [12], [13] are *not* effective for form extraction tasks. This is because form extraction relies heavily on anchoring on specific key phrases within the document that define each form field. For example, in an invoice, the *total due* field is typically indicated by the phrase “Amount Due”. Altering text unrelated to key phrases does not provide the model with meaningful variations to learn from. Furthermore, randomly swapping or deleting words risks breaking the contextual relationship between the key phrase and the field it defines. This can confuse the model and hinder its ability to accurately extract the correct information.

Based on this observation, we focus on key phrases associated with fields of interest. Inspired by the success of mixing approaches in the image domain [14], we propose a novel data augmentation technique called *FieldSwap*. Given a labeled example of a source field S in the training dataset, FieldSwap creates a synthetic example for another target field T by replacing the key phrase indicative of S in the document with a key phrase associated with T.



Earnings

Pay Type	Current	YTD
Base Salary	$3,308.62	$72,789.64
Bonus	$0.00	$5,636.63
Gain on ESPP	$0.00	$2,347.71
Group Term Life	$2.25	$49.50
Referral Bonus	$0.00	$1,500.00

Fields	Key phrases
current.salary	*base* *base salary* ...
current.overtime	*overtime* ...

Earnings

Pay Type	Current	YTD
Base	$3,308.62	$72,789.64
Bonus	$0.00	$5,636.63
Gain on ESPP	$0.00	$2,347.71
Group Term Life	$2.25	$49.50
Referral Bonus	$0.00	$1,500.00

Earnings

Pay Type	Current	YTD
Overtime	$3,308.62	$72,789.64
Bonus	$0.00	$5,636.63
Gain on ESPP	$0.00	$2,347.71
Group Term Life	$2.25	$49.50
Referral Bonus	$0.00	$1,500.00

Fig. 1: Example of FieldSwap on a paystub document. The source field S is *current.salary* ($3,308.62) and has key phrase “Base Salary”. FieldSwap generates two synthetic examples. At bottom left, the phrase is replaced with “Base”, another key phrase of *current.salary*, the field label for the instance ($3,308.62) is retained. At bottom right, the phrase is replaced with “Overtime”, the key phrase of another field, *current.overtime*, and the field label for the instance is changed to *current.overtime*.

Fig. 1 illustrates FieldSwap on a snippet of a paystub document with typical fields like *salary* and *bonus* for both *current pay period* and *year-to-date*. Beginning with a training document containing a labeled example for the *current.salary* field and the key phrase indicative "Base Salary", FieldSwap generates synthetic examples by substituting key phrases both within and across fields. This method diversifies training examples by varying field values, locations, and surrounding text while maintaining semantic equivalence. This prevents the model from memorizing specific field locations or coincidental text cues, enhancing its adaptability to new and unseen document layouts—an advantage not offered by simpler augmentation methods like synthetic field value generation.

FieldSwap is a simple but effective approach for generating synthetic examples. We evaluated FieldSwap on a diverse collection of datasets with real-world documents. Our results show that FieldSwap can significantly improve the data efficiency of form extraction models, even when only a small number of labeled training documents are available. However, key phrases are not annotated. Form-like documents often exhibit high diversity in layout and terminology, with key phrases potentially being away from their corresponding field instances and obscured by unrelated surrounding text. This complexity makes the task of identifying key phrases for fields non-trivial. It is also important to choose the source-target field pairs carefully. Note that multiple fields may have the same key phrase. For instance, the key phrase "Overtime" could be used to identify both the *current.overtime* field and the *year_to_date.overtime* field. For the example in Fig. 1, replacing the phrase "Base Salary" with "Overtime" while starting with the labeled example of *current.salary* in an attempt to generate a synthetic example for the *year_to_date.overtime* field yields a synthetic example that is not actually an instance of *year_to_date.overtime* but rather an instance of *current.overtime*, resulting in an incorrectly labeled example in the augmented dataset.

The key questions are: (a) how do we infer the key phrases associated with each field, and (b) which field pairs should be considered for generating these synthetic examples? We show that having a human expert supply a handful of key phrases works surprisingly well. The challenge then becomes how to automatically infer key phrases when only a limited number of labeled examples are present. Our findings demonstrate that a small model pre-trained for an extraction task on an out-of-domain corpus can be effectively used to identify the key phrases. In terms of selecting appropriate source-target field pairs, we found that considering the base type of a field is very useful. Form extraction tasks are invariably guided by a predefined schema, which serves as a blueprint for identifying and extracting specific information from a document. This schema comprises a collection of descriptive labels, termed *field*s, that pinpoint the relevant text segments for extraction. Moreover, the schema often encompasses details regarding the characteristics of these fields, such as their base types. In our study, we categorized each field into one of five base types: date, number, money, address, or string. The string data type serves as a comprehensive category for any fields that do not fall into the other four types. For example, a paystub document might have fields like *base salary* (type *money*), *period start date* (type *date*), *employee's address* (type *address*), *employee's name* (type *string*), etc. We show that simply considering all pairs of fields of the same base type can work quite effectively. Experiments on a diverse set of corpora show that FieldSwap can produce an improvement of 1–11 F1 points (i.e., 1–19% over the baseline). For context, novel architecture and pre-training objectives in this space typically resulted in increases of 1–1.5 F1 points [15]. We believe this is an exciting step towards better data efficiency in extraction tasks for visually-rich documents that is orthogonal to larger models and larger pre-training corpora.

We make the following contributions in this paper:

- We introduce a novel data augmentation strategy called *FieldSwap* that generates synthetic examples for a field T using examples from another field S. To our knowledge, this is the first data augmentation strategy designed specifically for visually rich documents.
- We present simple algorithms for automatically inferring key phrases and field pairs for generating synthetic examples.
- Through experiments on multiple real-world datasets, we demonstrate the effectiveness of *FieldSwap* in improving average F1 scores by 1–11 points completely automatically, even with small training sets (10–100 documents).
- With simple human expert inputs like key phrases, we observe further improvement up to 14 F1 points.

II. FieldSwap

FieldSwap exploits the property that form fields are very often indicated by key phrases [16]. For example, the *total due* field on an invoice document is often designated by phrases such as "total" or "amount due". We leverage this observation to generate synthetic examples by taking an instance of a source field, S, substituting associated key phrase with a key phrase of an intended target field, T, and relabeling the instance as an example of the target field. This augmentation is governed by two inputs:

1) The set of valid key phrases for each field. For example, "total" and "amount due" are valid key phrases for a *total due* field in invoices.
2) A list of source-to-target field pairs for which key phrases can be swapped and result in a valid synthetic example for the target field.

These settings can be specified manually or they can be inferred automatically. We find that manually specifying these settings works surprising well (see results in Section IV-C2). The main challenge is in automatically inferring these settings using only a few examples from a small dataset. Below, we present methods for automatically inferring key phrases and field pair mappings.

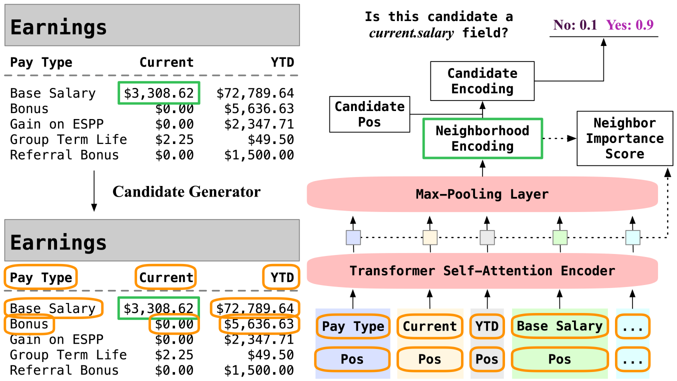

Earnings
Pay Type
Current
YTD
Base Salary $3,308.62 $72,789.64
Bonus $0.00 $5,636.63
Gain on ESPP $0.00 $2,347.71
Group Term Life $2.25 $49.50
Referral Bonus $0.00 $1,500.00
Candidate Generator
Earnings
Pay Type
Current
YTD
Base Salary $3,308.62 $72,789.64
Bonus $0.00 $5,636.63
Gain on ESPP $0.00 $2,347.71
Group Term Life $2.25 $49.50
Referral Bonus $0.00 $1,500.00
Is this candidate a current.salary field?
No: 0.1 Yes: 0.9
Candidate Encoding
Candidate Pos
Neighborhood Encoding
Neighbor Importance Score
Max-Pooling Layer
Transformer Self-Attention Encoder
Pay Type
Current
YTD
Base Salary
...
Pos
Pos
Pos
Pos
...

Fig. 2: Architecture of the candidate-based extraction model. Neighboring tokens of a *current.salary* candidate (e.g. $\$3,308.62$) are fed into a Transformer-based encoder and a max-pooling layer to generate a *Neighborhood Encoding*, which is concatenated with a candidate position embedding to make a binary prediction for the target field. We use the model's intermediate output of each individual neighboring token encoding and the *Neighborhood Encoding* for neighbor importance measurement.

A. *Automatically Inferring Key Phrases*

We observe that only a small number of tokens in a document are directly relevant to specific fields. We propose a method for identifying these important tokens. We then derive important phrases from them and aggregate these important phrases to infer a set of key phrases for the field.

1) Use of OCR Engine: Our method relies on an OCR service for two major purposes. Firstly, we use the OCR service to detect each text element in a document and their corresponding bounding boxes. A bounding box refers to a rectangular region that outlines the spatial position of a text element within a document. It is typically defined by their coordinates, which specify the top-left, top-right, bottom-left, and bottom-right corners of the rectangle. By utilizing bounding boxes, we can locate any text of interest. Secondly, we use the OCR service to detect lines of text. Lines are groups of tokens on the same y-axis that are typically separate from other lines by way of visual features (e.g. vertical bars in a table) or long horizontal stretches of whitespace. Both bounding boxes and line detection are essential for our key phrase inference method. We will explain how we utilize these signals in more detail later.

It is important to acknowledge that the accuracy of the OCR engine directly affects the quality of inferred key phrases. However, modern OCR engines have become remarkably robust, able to handle challenges like handwritten scribbles, background noise, or unusual fonts. Additionally, techniques like utilizing multiple OCR engines or post-processing to remove errors can further minimize the impact of potential OCR mistakes. In this paper, we mainly focus on the augmentation method. The established accuracy of our chosen OCR engine allows us to exclude discussions of OCR performance.

2) Identifying Important Tokens: We observe that most important tokens associated with an instance of a field tend to be located close to the instance, either positioned horizontally or vertically aligned with it in the document. We thus define tokens that are horizontally or vertically close to a text element as its neighboring tokens. We hypothesize that only a few tokens among the neighboring tokens of a field instance hold significant relevance.

To identify neighboring tokens, we first use an OCR service[1] to detect the bounding boxes of each text element in a document. By our definition, an instance's neighboring tokens are tokens that are close to either the x-axis or y-axis of the instance's bounding box center in the document. We thus introduce the off-axis distance metric. Assuming two points, a and b, have x-axis and y-axis coordinates (a_x, a_y) and (b_x, b_y), respectively, the off-axis distance is calculated as $|a_x - b_x| \cdot |a_y - b_y|$. Points that are close together in terms of their x or y axes have a distance of close to 0, while those that are diagonally positioned have a greater distance. We use t closest tokens to an instance based on off-axis distance between the center of their bounding boxes as the instance's neighboring tokens, where t is a tunable hyperparameter.

We propose a method for measuring each neighboring token's importance score to a field instance and identifying important tokens. We leverage the binary classifier architecture described in [16]. In this architecture (as illustrated in Fig. 2), base type candidates are first extracted from an OCR-processed document using common-off-the-shelf annotators like date and number annotators. For each candidate, the model encodes each neighboring token by concatenating its text embedding and relative position embedding. Then, it employs self-attention and max-pooling to generate a single representation of the candidate's neighborhood (i.e., *Neighborhood Encoding*), which is used along with other features to make a binary prediction for field(s) in question. This representation is easy to manipulate for identifying important neighbors. Based on this architecture, we proposed a method to measure the *importance scores* of each neighboring token for a candidate. The *importance score* of a neighboring token to a candidate is calculated as the *cosine similarity* between the model's intermediate output on the candidate *Neighborhood Encoding* and the encoding of that individual neighboring token.

We train the model on a large out-of-domain dataset and directly apply it to get candidate *Neighborhood Encoding* on the target domain. The intuition is that the relative position of a neighbor plays a crucial role in identifying important neighbors, and these positional cues are generally shared across domains. Empirical results show that the model identifies a reasonable set of important neighbors for each candidate. For our purpose, we are only interested in finding important neighboring tokens for positive candidates of the target domain, so we generate candidates from ground truth instances of fields directly.

[1] https://cloud.google.com/vision

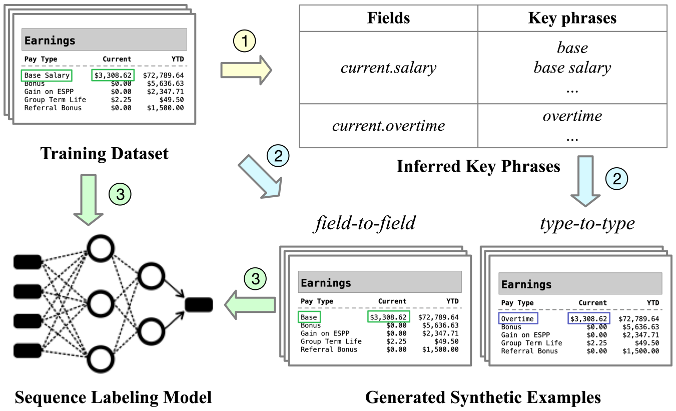


Fig. 3: Overall processing procedures. In step 1, key phrases of all fields are inferred from the training dataset. In step 2, either field-to-field or type-to-type FieldSwap augmentation is applied to the training dataset. In step 3, a form extraction model (such as sequence labeling model) is trained on the union of the original training documents and the synthetic documents.

After obtaining the importance scores of each neighboring token for a labled example, we apply *Sparsemax* [17] across the importance scores to get a sparse output of the neighboring tokens with non-zero scores. We consider these neighboring tokens as the set of *important tokens*.

3) Inferring Important Phrases: We observe that an important phrase typically resides within a single line. We use an OCR service[1] to detect lines in the document. We construct *important phrases* by concatenating all tokens residing on the same OCR line as long as at least one token is identified as an *important token*. Leveraging OCR lines to infer important phrases also makes the process more tolerant to the model's recall loss on important tokens. As long as the model finds one important token, we'll be able to infer the longer phrase, if it exists. We perform post-processing of the OCR line inferred phrases by cleaning up any leading and trailing punctuation.

We define *phrase importance score* as the average token importance score within the phrase.

4) Aggregation and Ranking by Field: Once all important phrases and importance scores have been gathered for all labeled examples, we aggregate the results by field and phrase. For any field F, we use $Score(F, P, C_i)$ to denote the *phrase importance score* for a F example C_i that has important phrase P. We calculate

$$Importance(F, P) = 1 - \exp(\Sigma_i log(1 - Score(F, P, C_i))) \tag{1}$$

as the measurement of how P relates to F. This measurement prefers phrases with higher importance scores and frequency across all labeled examples for F. Additionally, this aggregation approach enhances the robustness of our method against occasional OCR errors (as discussed in Section II-A1) by minimizing their impact on the aggregated level. For each field, we rank all phrases by their *Importance* and select the top k phrases as the key phrases for the field, where k is a tunable hyperparameter.

5) Fields without Key Phrases: Not all fields necessarily have key phrases. Fields such as *company name*, *company address*, and *statement date* often appear in the top corners of documents without any specific phrase indicators. When trying to infer key phrases for such fields, the model may output wrong phrases (usually with low importance scores). For example, the model might infer "LLC" as a key phrase for the *company address* field since it may often find many company names with "LLC" directly above the *company address* field value. However, the values of other fields cannot be part of the key phrase of another field because field values are variable across different documents while key phrases are consistent across the documents (belonging to the same template). To avoid such spurious correlations, we explicitly exclude tokens that are part of the ground truth for any field in the document. We also set a threshold θ to filter out any inferred key phrases with importance score below the threshold, where θ is a tunable hyperparameter.

B. Field Pair Mappings

We explored three options to determine source-target field mappings.

Field-to-Field swap. The simplest and most straightforward option is to swap only examples belonging to the same field. In this case, *source* field, S, and *target* field, T, denote the same field. With this approach, we are less likely to generate out-of-distribution synthetic examples. The downside is that we are usually unable to generate a sizeable number of synthetic examples unless the field has a lot of labeled examples and key phrase variations. In practice, it is the rare fields that we are most interested in augmenting, and these are the fields that benefit the least from this mapping approach.

Type-to-Type swap. As mentioned in Section I, each field is associated with a general base type such as date, number, money, address or string. A simple heuristic is to map fields that are similar, so considering pairs of fields that have the same base type is a natural idea. We can generate synthetic examples for a target field (e.g. *salary*) from other same-type fields (e.g. *bonus*, *overtime*) by swapping the key phrases. Note that our implementation of type-to-type mappings implies that a field will also be mapped to itself. This approach allows us to generate more synthetic examples for rare fields by utilizing examples from other frequent fields with the same base type. It also regularizes the model against spurious correlations with nearby non-related text.

However, we might generate bad synthetic examples if there exist contradictory fields with the same type. For example, *current.bonus* and *year_to_date.bonus* have the same key phrase "bonus", and *current.vacation* and *year_to_date.vacation* have the same key phrase "vacation". FieldSwap would generate contradictory synthetic examples when swapping between the four fields, such as by creating a synthetic *current.vacation* example using a *year_to_date.bonus* example.

All-to-All swap. We also considered swapping between any pair of fields, but found that this was nearly always worse than type-to-type swaps.

C. Generating Synthetic Documents

We generate FieldSwap augmentations at document level, so that it is agnostic to the architecture of the extraction model. However, this brings extra complexity to the implementation of the FieldSwap augmentation since there could be a number of constraints introduced by different model architectures.

For examples, for approaches like sequence labeling [1], [3], every token on the document is an input to the model. When swapping the key phrases for a pair of fields, should we also swap the values for these fields so that the model is not confused by the augmented examples having values too different from the original examples? For instance, the values of fields such as *tax due* and *total due* have different relative magnitudes, which might need to be preserved. Furthermore, should we preserve certain document-level semantics? For example, some fields should occur only once in a document, shall we ensure FieldSwap does not introduce multiple instances in a document for such fields? Will there be other instances of fields which do not belong to the source field but are also affected by phrase change? Accommodating these kinds of constraints leads to even more questions—e.g., must augmentations only be made when two fields can act as both sources and targets for one another? Must both fields always be present on the same document in order to perform an augmentation? How might we generate multiple augmentations on a single document?

In this work, we want to keep the implementation as simple as possible. We generate one augmentation at a time by swapping only one pair of fields so that the augmented data has very slight disturbances. We only change the label for source fields for simplicity – we leave the values unchanged. We treat all fields as they could appear multiple times in the document during training time, and only apply the schema constraints at inference time. We found this simple implementation works surprisingly well with the sequence labeling model we evaluated on. We leave it to future work to adapt FieldSwap for more complex situations.

Once we have the key phrases and field pair mappings, we proceed to generate synthetic documents. For each document in the training data, we iterate through all source-to-target field pairs, if the document contains the source field S and any key phrases for S, we replace all matching source key phrases with target key phrase, relabel all S instances to T and generate one synthetic document corresponding to each key phrase of target field T.

Note that if no phrases in the document match a key phrase for the source field, then no synthetic documents are generated. Furthermore, if the synthetic document remains unchanged after replacing the source key phrase with the target's, we discard it. This helps prevent us from creating semantically incorrect synthetics, such as the case previously described in Section II-B, when two fields have the same key phrase but are semantically different (e.g., *current.bonus* vs. *year_to_date.bonus*).

III. Human Expert

Incorporating human knowledge and expertise into the training and development of NLP models has become increasingly prevalent [18]–[20]. A human familiar with a given document type can easily provide additional inputs in lieu of additional labeled examples. For instance, a human can provide typical key phrases that indicate a field as well as mark potential pairs of fields that should be used for FieldSwap. This idea draws on the literature in rule-based augmentations where rules are provided in addition to training examples.

We design a *human expert* approach by devising a FieldSwap configuration with human inputs. Instead of relying only on automatically detected key phrases and field pairs, we leverage human expertise to protect FieldSwap from some error-prone situations. For example, when configuring key phrases, some fields, such as *company_name* and *company_address*, do not have clear key phrases, so we exclude these fields from FieldSwap entirely in the *human expert* setup. Other fields, particularly rare ones, might not have enough labeled examples in the training set. In that case, we rely on domain knowledge to supply additional key phrases. When configuring field pairs, we start with type-to-type field pairs, then prune those that most likely to appear in different tables or sections of the document.

Using FieldSwap in this setting will generate more useful synthetic documents compared to the field-to-field setting, and fewer incorrect synthetic documents compared to the type-to-type setting. We assume that this *human expert* configuration can achieve higher Max F1 scores than the automatically generated FieldSwap configuration, due to the higher quality human inputs.

IV. Experiments

A. Dataset

We evaluated the performance of FieldSwap on five datasets of form-like documents, including two public datasets (FCC Forms [21], FARA [21]) and three proprietary datasets (Earnings, Brokerage Statements, Loan Payments). Table. I and Table. II provide summaries of the statistical characteristics of each dataset. Each dataset corresponds to a different document type. We herein also refer to document types as domains. For each domain, a predefined schema outlines all the fields to be extracted, with each field categorized into one of the five base types: address, money, date, number, or string. To construct training sets of varying sizes for our experiment, a subset of documents was randomly selected from the larger corpora for each dataset. We conducted evaluation on a fixed hold-out test set for each domain.

TABLE I: Dataset Statistics.

Document Type	# Fields	Train Docs Pool Size	Test Docs
FARA [21]	6	200	300
FCC Forms [21]	13	200	300
Brokerage Statements	18	294	186
Earnings	23	2000	1847
Loan Payments	35	2000	815

TABLE II: Number of fields with different base types for each document type.

	Field Type				
Document Type	Address	Date	Money	Number	String
FARA [21]	0	1	0	1	4
FCC Forms [21]	1	4	2	1	5
Brokerage Statements	2	4	5	0	7
Earnings	2	3	15	0	3
Loan Payments	3	5	20	0	7

B. Experimental Setup

Automatically inferring key phrases. We use the model architecture described in Section II-A2 for automatically inferring key phrases. The model is trained on an out-of-domain document type (invoices) with approximately 5000 training documents. We tune the hyperparameters using grid search and use the most performant values. In all our experiments, we restrict number of neighboring tokens to 100. We use top 3 important phrases as the key phrases for each field. We set the importance score threshold θ at 0.2.

Human expert. One of the authors of this paper examined approximately 10 training documents in domain of interest and recorded the key phrases they observed for each field. For fields that doesn't exist in the training documents they inspected, they rely on domain knowledge to come up with a handful of key phrases. The same person also constructed the field pair mappings using the method described in Section III, which avoids contradictory field pairs.

Backbone form extraction model. FieldSwap is designed to be agnostic to any architecture for form extraction task. In our experiment, we use the sequence labeling model described in [1] and follow the same unsupervised pre-training procedure. We first pretrain the model on approximately $30k$ unlabeled out-of-domain form documents, then fine-tune the model on the training documents in the target domain. When training the model on the target domain, we split the dataset into 90%-10% training-validation sets. We train the models for approximately 6 hours and pick the checkpoint with the highest accuracy across all fields in the validation split.

Evaluation. We believe FieldSwap is complementary to other existing augmentation methods, and can be combined with them to achieve incremental improvement. Therefore, we focus our evaluation on comparing its performance against the baseline setting of no augmentation. We train baseline form extraction models on the original training set without synthetic documents. We add the synthetic documents generated with FieldSwap to the training set and train new form extraction models following the same procedure. We train both models for the same amount of time to ensure that any difference in performance is due to the FieldSwap augmentation technique itself, rather than the model with augmented data simply having more exposure to the training data. By comparing the end-to-end F1 scores of the trained models, we can quantify the effectiveness of FieldSwap augmentation.

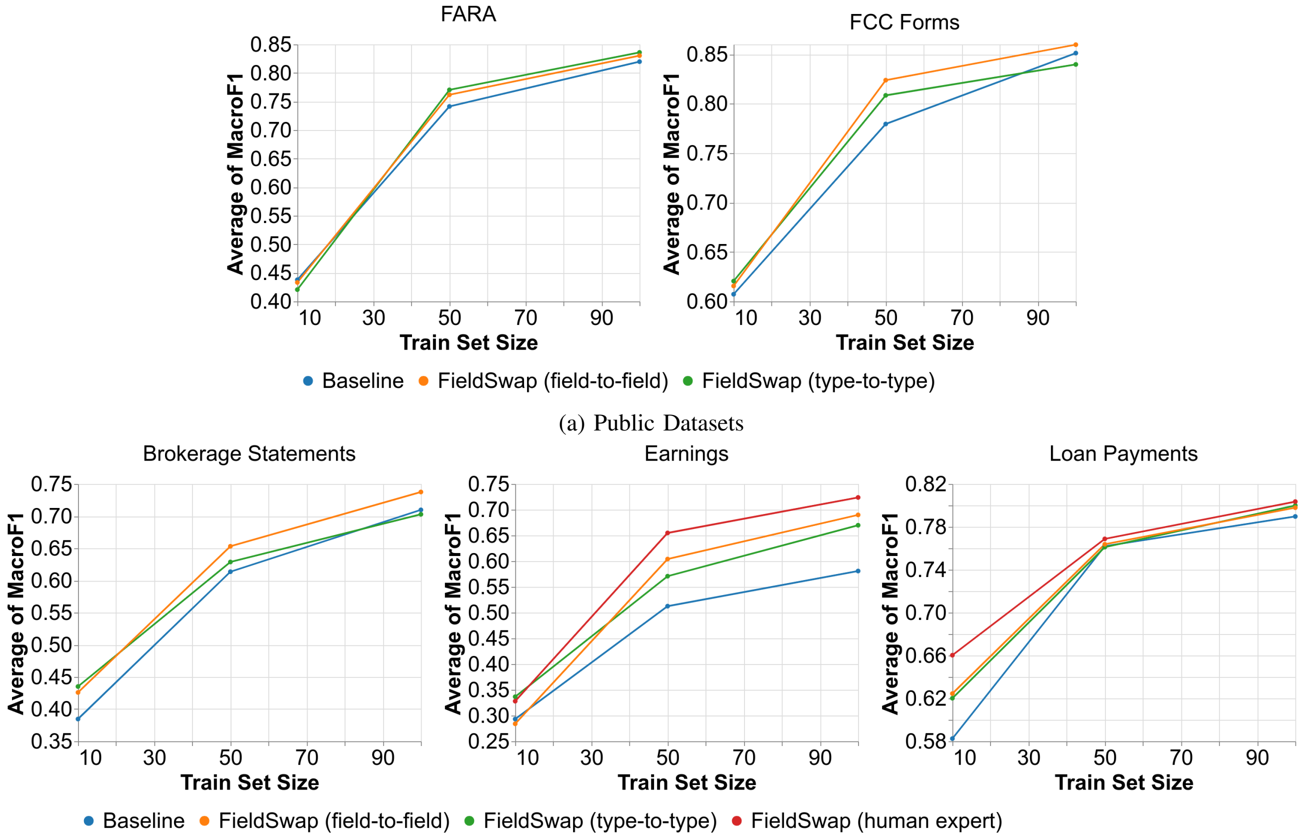
We vary the training set sizes (i.e. 10, 50, 100) to plot learning curves. In order to capture the inherent variability that may arise in experiments with such small dataset sizes, we repeat our experiments across two different axes. For a given domain and dataset size N, we repeat the experiment using (i) 3 different random collections of N documents from the domain's large pool of documents (see Table. I), and (ii) 3 model training trials. This amounts to a total of 9 experiments for a given domain and training set size. Each data point we report on the learning curve corresponds to the average performance across these 9 experiments on the fixed hold-out test set.

C. Results

Our experiments aim at answering the following questions: (1) Is FieldSwap effective in its fully automatic version? (2) Does it work better with human-supplied inputs? (3) How do improvements vary across document types and field types?

1) Automatic FieldSwap: We tested both field-to-field and type-to-type field pair mappings with automatically inferred key phrases. As shown in Fig. 4, FieldSwap consistently yielded neutral or better performance across all datasets and training set sizes we evaluated on. For instance, FieldSwap led to an average macro-F1 improvement of 1–4 points for the FCC Forms dataset, 2–5 points for the Brokerage Statements dataset, and a remarkable 4–11 points for the Earnings dataset. It is worth noting that novel architecture designs and pre-training objectives in this space typically result in F1 gains of 1 – 1.5 points [15]. The substantial improvement achieved by FieldSwap is therefore very exciting. We will further discuss the effect of document type in Section IV-C3.

Field-to-Field vs Type-to-Type. Our findings indicate that type-to-type swap outperforms field-to-field swap when the training set size is small (10 documents). However, as the training set size increases (50–100 documents), field-to-field swap either matches or surpasses the performance of type-to-type swap. Table. III presents the average number of synthetic documents generated by FieldSwap for each document type and training set size. Observations suggest that type-to-type swap typically generates 3-10× more synthetic documents than field-to-field swap. However, as discussed in Section II-B, it is more prone to generating contradictory synthetic examples than field-to-field swap. When the training set is small, the larger amount of synthetic documents generated by type-to-type swap provides a performance boost. However, as the training set size increases, field-to-field swap becomes more effective as it has access to a sufficient number of source examples to produce synthetic documents and is less likely



(a) Public Datasets

(b) Proprietary Datasets

Fig. 4: Mean Macro-F1 scores across various domains and training set sizes.

to produce harmful synthetic documents that could negatively impact the model.

Macro-F1 vs Micro-F1. Current form extraction models perform poorly for rare fields when there is only a small amount of labeled data. We believe FieldSwap is particularly valuable in this scenario, as it leverages labeled training examples from other frequent fields to generate synthetic training data for rare fields. Thus, we prioritize the evaluation metric of macro-F1.

Despite our focus on macro-F1, we also observe that similar improvements still hold when evaluating using micro-F1. As shown in Fig. 5, the same pattern of results persists. For instance, FieldSwap achieves an average micro-F1 improvement of 2-5 points for the Earnings dataset and 1-5 points for the Brokerage Statements dataset. While the improvement gains are less pronounced compared to those observed with macro-F1, this suggests that the most significant gains have come from rare fields, which aligns with our initial hypothesis.

2) FieldSwap with Human Expert: We conducted a comparative analysis of the performance between automatic FieldSwap and FieldSwap utilizing human expert curated phrases and field pair mappings on two domains. As shown in Fig. 4, better key phrases and field pair mappings generally lead to better performance. Human inputs further improves the performance by 4–5 F1 points for the Earnings dataset at 50–100 documents, and by 4 points for the Loan Payments dataset at 10 documents.

The performance gap is mostly attributed to rare fields, as shown in Table. IV. For example, *year_to_date.sales_pay* field has a particularly low frequency, appearing in only 3.9% of the corpus of 2000 training documents. In our low data setting where only a subset of documents is randomly selected, it's possible that there are either no or very few labeled examples for such rare fields. This creates few-shot or even zero-shot scenarios where automatic approach struggle. In these cases, a human expert can provide key phrases *not* found in the limited training data, leading to a significant advantage. This is expected, as the automatic approach cannot discover phrases it has never seen. For such scenarios, the involvement of a human expert is crucial for achieving optimal performance.

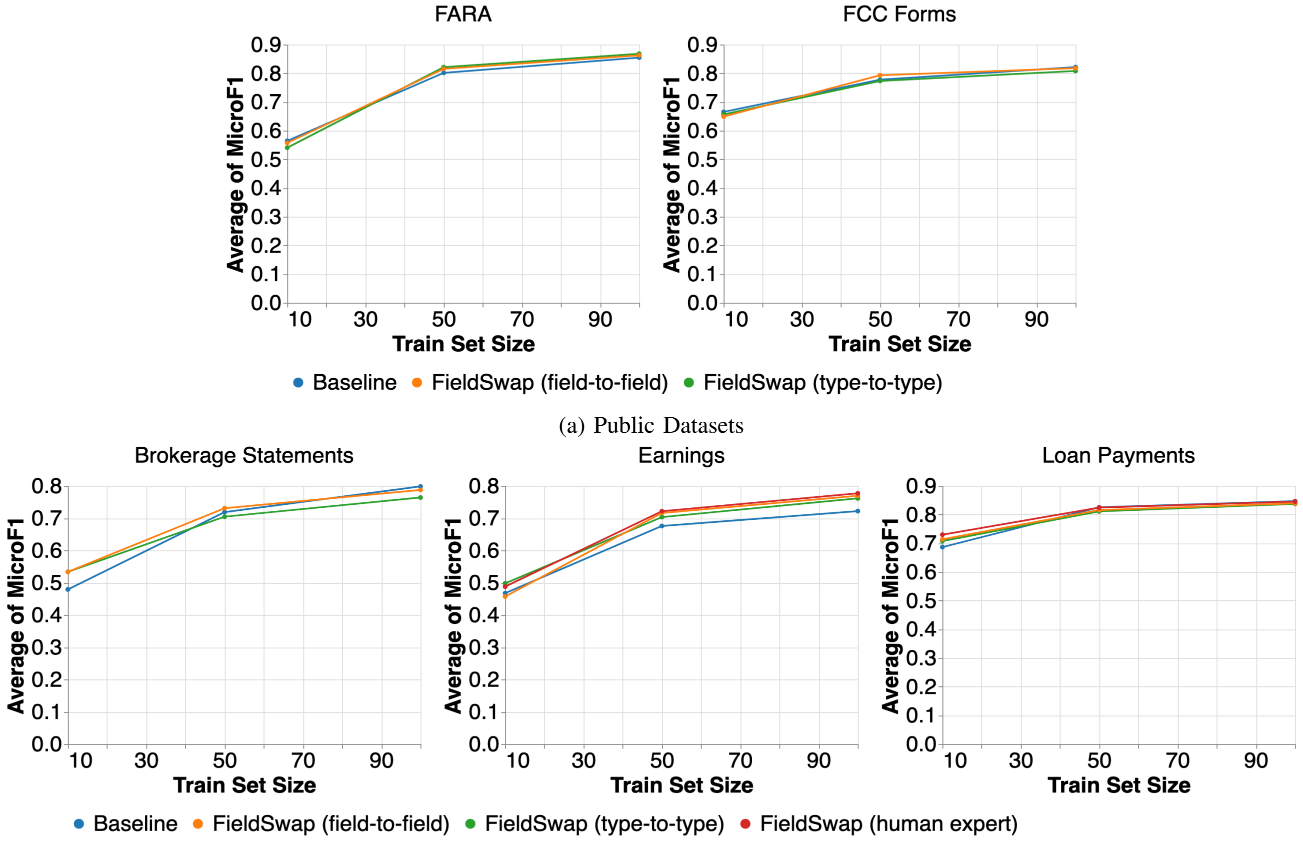
In practice, the decision of whether or not to incorporate human input ultimately depends on the specific use case and the trade-off between human resource and optimal performance.

3) Discussions: In this section, we try to answer the question of how FieldSwap improves across different fields and document types.

Effect of field type. Our experimental setup associate fields with five base types (i.e. address, date, money, number, string). Among the five datasets we evaluated on, there are only two

TABLE III: Average number of FieldSwap synthetic documents at different training set size for each document type.

		Number of Synthetic Documents		
Domain	**Original Training Set Size**	**FieldSwap (field-to-field)**	**FieldSwap (type-to-type)**	**FieldSwap (human expert)**
FARA	10	2	5	-
	50	176	374	-
	100	592	1616	-
FCC Forms	10	246	842	-
	50	1663	5755	-
	100	3310	11346	-
Brokerage Statements	10	256	1266	-
	50	1486	7994	-
	100	2917	16590	-
Loan Payments	10	435	2378	1136
	50	2699	18118	5933
	100	6083	38081	11682
Earnings	10	197	1542	366
	50	1345	11643	1862
	100	2717	26001	3707



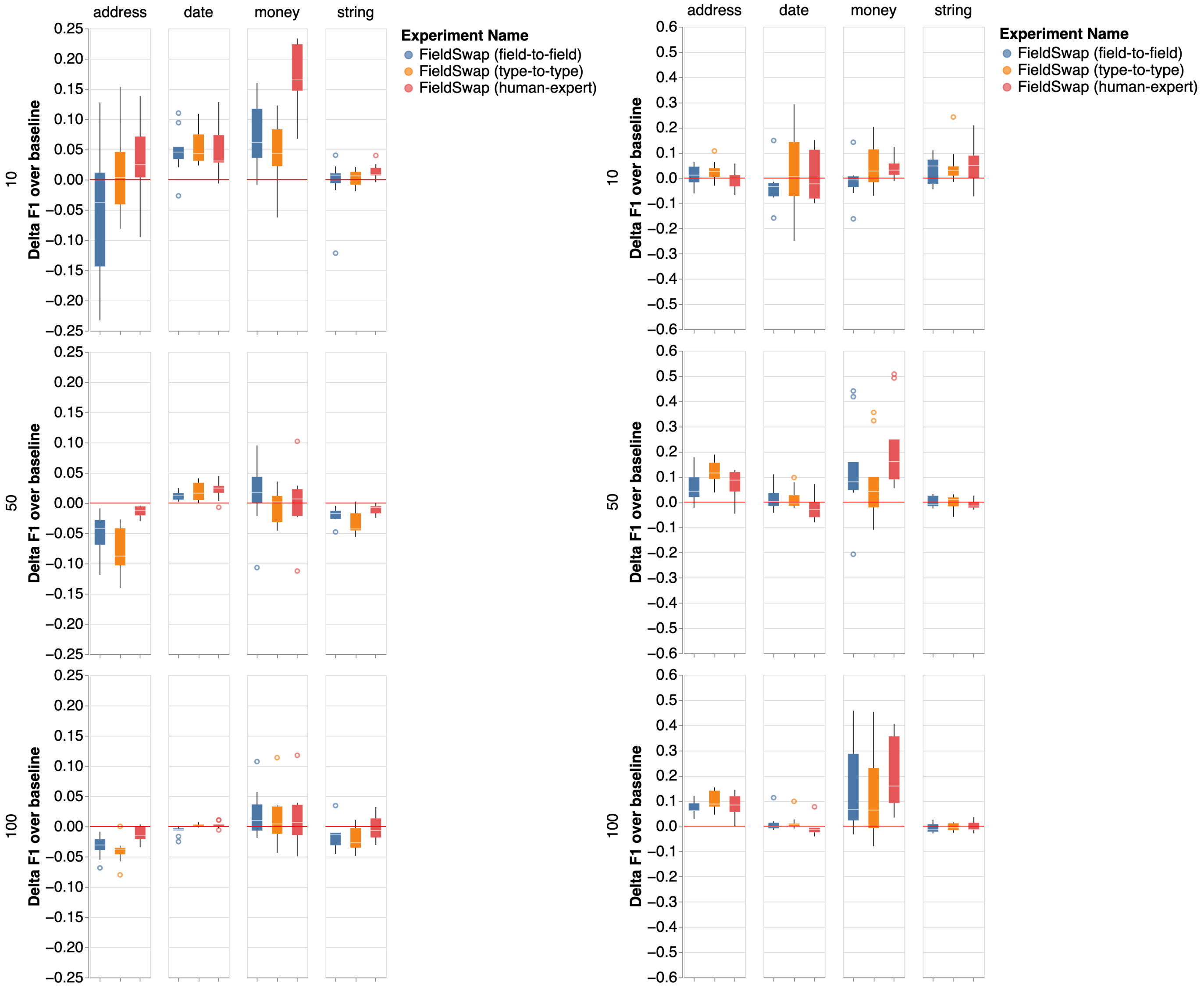
(a) Public Datasets

(b) Proprietary Datasets

Fig. 5: Mean Micro-F1 scores across various domains and training set sizes.

TABLE IV: Fields with the largest mean F1 gains between the automatic (field-to-field) and human expert setting when trained on 50 documents for the Earnings domain. Frequency refers to the fraction of documents that contain said field in a collection of 2000 documents.

Field	Frequency	F1 (FieldSwap, automatic)	F1 (FieldSwap, human expert)	ΔF1
year_to_date.sales_pay	3.9%	27.91	56.27	28.36
current.sales_pay	2.85%	17.97	46.23	28.26
year_to_date.pto_pay	15.9%	50.3	66.78	16.48
current.pto_pay	9.5%	14.36	28.18	13.82



(a) Loan Payments

(b) Earnings

Fig. 6: Field F1 score differences of FieldSwap over baseline on the Loan Payments and Earnings domain. The length of each box plot represents the distance between the upper and lower quartiles. Each whisker extends to the furthest data point in each wing that is within 1.5 times the interquartile range (IQR). The line in the middle of the boxplot indicates the median. The dots denote outliers. The horizontal red lines mark $y = 0$.

fields with *number* type (each in a different domain), making the results unrepresentative. Therefore, we restrict our analysis on the remaining four base types. We believe that the *number* type shall exhibit a similar pattern to the *money* type, as they are somewhat alike.

We investigate the effect of FieldSwap on various field types within the Loan Payments domain across all training set sizes. As shown in Fig. 6a, we observe the performance gains from FieldSwap primarily come from *date* and *money* fields, while we see negative effects on *address* and *string* fields. This is likely because *string* and *address* fields often lack the clear key phrases, making it challenging for FieldSwap to generate meaningful augmentations. FieldSwap may introduce spurious correlations and generate poor synthetic documents that harm the model's performance. To mitigate this issue, we can employ the human expert setting. Domain experts can identify fields that typically lack clear key phrases and exclude them from FieldSwap augmentation. In our experiments with the human expert setting, we found that explicitly excluding such fields dissipated the negative effects.

Besides, it is worth noting that not all *string* or *address* fields lack distinct key phrases. FieldSwap remains effective for those fields that do have distinct key phrases. We extended our investigation into the Earnings domain. As shown in

Fig. 6b, FieldSwap demonstrated positive gains on *address* and *string* fields. According to Table. II, the Earnings domain has fewer fields of the *address*, *date*, and *string* type. This limits the representativeness of the results compared to the Loan Payments domain, but they remain insightful.

Effect of document type. As shown in Fig. 4, the most significant improvement is observed for the Earnings domain. Compare to other document types, the majority of fields in Earnings are presented in tabular format, share similar base types (i.e. money, date), and have clear and succinct phrase indicators. We believe FieldSwap is most effective when dealing with document types with such characteristics. Furthermore, since the order of fields within these tables is unimportant, FieldSwap is particularly well-aligned for augmenting these structures.

That being said, the Earnings domain also poses a challenge since many field pairs can easily yield bad synthetic examples, as discussed in Section II-B (e.g., *current.X* vs. *year_to_date.X*). Yet, even in the presence of these potentially contradicting pairs, type-to-type swap still improves the performance across all training set sizes we evaluated on. This demonstrates that the proposed method tolerates a small number of these contradictory synthetic examples.

The improvement gain on the FARA domain is relatively small. This is because this corpus contains only a handful of fields, with 4 out of the total 6 fields being of the *string* type, which FieldSwap is not well-suited for handling. The remaining two fields belong to different base types and are thus not swappable. Nevertheless, FieldSwap maintains neutral or slightly better results throughout the learning curve.

V. Related Work

Form extraction. Approaches for extracting information from form-like documents typically rely on multimodal features: text, spatial layout, and visual patterns. Models often make use of pre-trained encoders that incorporate such multimodal signals [15], [22], [23], but these encoders require a large amount of pre-training data, although they do exhibit good downstream task data efficiency during fine-tuning [24]. Large amounts of training data are also required by span classification approaches [16], [25], sequence labeling approaches [26], [27], and end-to-end approaches [28]. Rather than suggesting a new model architecture, we propose a data augmentation method that is orthogonal to any model architecture.

Data augmentation. Data augmentation is a class of techniques for acquiring additional training examples automatically. Two main categories of data augmentation are rule-based and model-based techniques, which use hard-coded data transformations or pre-trained models (typically language models), respectively. Rule-based techniques—such as EDA [12]—are easier to implement but have limited benefit, whereas model-based techniques—such as back-translation [29] and example extrapolation [30]—are more difficult to develop but offer greater benefit [31]. FieldSwap contains elements of both categories, as it changes (possibly automatically inferred) key phrases based on a set of swap rules.

Counterfactual data augmentation has emerged as a valuable technique in various NLP tasks like text classification and sentiment analysis. These methods typically involve either human experts revising training data to create counterfactual examples [32] or automated term replacement for label inversion [33], which might have subtle similarities with FieldSwap. However, these approaches do not directly address the unique challenges of form extraction task, where success hinges on the accurate identification of key phrases within documents. FieldSwap offers a specialized solution to identify key phrases automatically. By focusing solely on key phrases rather than any general terms in the document, FieldSwap enhances model robustness without introducing irrelevant or contradictory data that could hinder training. Additionally, FieldSwap ensures the plausibility of generated field labels, maintaining the integrity and relevance of augmented data within the context of form extraction.

Reference [31] suggest that “the distribution of augmented data should neither be too similar nor too different from the original”. FieldSwap achieves this balance by placing known key phrases in the contexts of other key phrases, which increases diversity in a controlled way. The use of schema field types in FieldSwap is similar to the use of entity types for mention replacement in named entity recognition, which is effective especially in low-data settings [13].

Other data augmentation techniques have been used for multimodal tasks that combine text and vision, such as image captioning [34] and visual question answering [35], [36]. FieldSwap, like these other approaches, focuses on modifying the textual component of each input rather than the visual component; that is, the key phrase is replaced but the spatial layout remains the same.

Perhaps the most similar prior work to ours is [37]. The main idea of that work is that if two items appear in similar contexts, then they can be interchanged wherever one of them occurs to generate new examples. In our work, the items we change are key phrases associated with schema fields, and we determine interchangeability based on the base type of the field. Rather than generate new labeled examples by changing the value of the field, we generate examples by changing the surrounding context (via key phrases).

VI. Conclusions

In this paper, we introduce a data augmentation technique specifically designed for extraction tasks on visually rich documents. We leverage the observation that many fields are associated with a “key phrase” that serves as an identifier. By replacing the key phrase in a source field example with that of the target field, we generate augmented examples for the target field. Experiments on a diverse range of datasets demonstrate that this simple technique is very effective for scenarios involving small training sets (10–100 documents), yielding improvements of 1–11 macro-F1 points.

This result opens up two interesting directions for future work. Firstly, how can we refine FieldSwap to better handle the complex scenarios we described in Section II-C? Secondly, there are several extensions to FieldSwap that are worth investigating. Under what circumstances does swapping across document types help? Is it possible to use a large language model (LLM) instead of a human expert to generate a set of key phrases based on field names or descriptions? Can we extract key phrases from an unlabeled corpus to facilitate semi-supervised learning [38]?

References

[1] C.-Y. Lee, C.-L. Li, T. Dozat, V. Perot, G. Su, N. Hua, J. Ainslie, R. Wang, Y. Fujii, and T. Pfister, "FormNet: Structural encoding beyond sequential modeling in form document information extraction," in *ACL*, 2022.

[2] Ł. Garncarek, R. Powalski, T. Stanisławek, B. Topolski, P. Halama, M. Turski, and F. Graliński, "Lambert: layout-aware language modeling for information extraction," in *International Conference on Document Analysis and Recognition*. Springer, 2021, pp. 532–547.

[3] Y. Xu, Y. Xu, T. Lv, L. Cui, F. Wei, G. Wang, Y. Lu, D. Florencio, C. Zhang, W. Che *et al.*, "LayoutLMv2: Multi-modal pre-training for visually-rich document understanding," *arXiv preprint arXiv:2012.14740*, 2020.

[4] S. Wu, L. Hsiao, X. Cheng, B. Hancock, T. Rekatsinas, P. Levis, and C. Ré, "Fonduer: Knowledge base construction from richly formatted data," in *Proceedings of the 2018 International Conference on Management of Data*, 2018, pp. 1301–1316.

[5] R. Sarkhel and A. Nandi, "Visual segmentation for information extraction from heterogeneous visually rich documents," in *Proceedings of the 2019 International Conference on Management of Data*, 2019, pp. 247–262.

[6] Y. Zhou, W.-T. Chen, B. Zhang, D. Lee, J. H. Caufield, K.-W. Chang, Y. Sun, P. Ping, and W. Wang, "Create: Clinical report extraction and annotation technology," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 2677–2680.

[7] M. Rezk, L. A. Alemany, L. Nio, and T. Zhang, "Accurate product attribute extraction on the field," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 1862–1873.

[8] G. Jaume, H. K. Ekenel, and J.-P. Thiran, "Funsd: A dataset for form understanding in noisy scanned documents," in *International Conference on Document Analysis and Recognition Workshops (ICDARW)*, vol. 2, 2019, pp. 1–6.

[9] S. Park, S. Shin, B. Lee, J. Lee, J. Surh, M. Seo, and H. Lee, "Cord: A consolidated receipt dataset for post-ocr parsing," 2019.

[10] Z. Huang, K. Chen, J. He, X. Bai, D. Karatzas, S. Lu, and 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.

[11] T. Stanisławek, F. Graliński, A. Wróblewska, D. Lipiński, A. Kaliska, P. Rosalska, B. Topolski, and P. Biecek, "Kleister: key information extraction datasets involving long documents with complex layouts," in *International Conference on Document Analysis and Recognition*. Springer, 2021, pp. 564–579.

[12] J. Wei and K. Zou, "EDA: Easy data augmentation techniques for boosting performance on text classification tasks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Nov. 2019.

[13] X. Dai and H. Adel, "An analysis of simple data augmentation for named entity recognition," in *Proceedings of the 28th International Conference on Computational Linguistics*, Dec. 2020.

[14] H. Naveed, "Survey: Image mixing and deleting for data augmentation," *arXiv preprint arXiv:2106.07085*, 2021.

[15] Y. Huang, T. Lv, L. Cui, Y. Lu, and F. Wei, "Layoutlmv3: Pre-training for document ai with unified text and image masking," *arXiv preprint arXiv:2204.08387*, 2022.

[16] B. P. Majumder, N. Potti, S. Tata, J. B. Wendt, Q. Zhao, and M. Najork, "Representation learning for information extraction from form-like documents," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Jul. 2020.

[17] A. Martins and R. Astudillo, "From softmax to sparsemax: A sparse model of attention and multi-label classification," in *International conference on machine learning*. PMLR, 2016, pp. 1614–1623.

[18] C. Chai, L. Cao, G. Li, J. Li, Y. Luo, and S. Madden, "Human-in-the-loop outlier detection," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 19–33.

[19] J. Yang, X. Zhao, J. Fan, G. Chen, C. Peng, S. Yao, and X. Du, "A human-in-the-loop approach to social behavioral targeting," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 277–288.

[20] E. Dragut, Y. Li, L. Popa, and S. Vucetic, "Data science with human in the loop," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 4123–4124.

[21] Z. Wang, Y. Zhou, W. Wei, C.-Y. Lee, and S. Tata, "A benchmark for structured extractions from complex documents," *arXiv preprint arXiv:2211.15421*, 2022.

[22] S. Appalaraju, B. Jasani, B. U. Kota, Y. Xie, and R. Manmatha, "Docformer: End-to-end transformer for document understanding," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 973–983.

[23] L. Garncarek, R. Powalski, T. Stanisławek, B. Topolski, P. Halama, M. Turski, and F. Graliński, "Lambert: Layout-aware language modeling for information extraction," 2021.

[24] C. Sage, T. Douzon, A. Aussem, V. Eglin, H. Elghazel, S. Duffner, C. Garcia, and J. Espinas, "Data-efficient information extraction from documents with pre-trained language models," in *International Conference on Document Analysis and Recognition*. Springer, 2021, pp. 455–469.

[25] S. Tata, N. Potti, J. B. Wendt, L. B. Costa, M. Najork, and B. Gunel, "Glean: structured extractions from templatic documents," *Proceedings of the VLDB Endowment*, vol. 14, no. 6, pp. 997–1005, 2021.

[26] M. Aggarwal, H. Gupta, M. Sarkar, and B. Krishnamurthy, "Form2Seq : A framework for higher-order form structure extraction," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Nov. 2020.

[27] C.-Y. Lee, C.-L. Li, T. Dozat, V. Perot, G. Su, N. Hua, J. Ainslie, R. Wang, Y. Fujii, and T. Pfister, "FormNet: Structural encoding beyond sequential modeling in form document information extraction," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*.

[28] Z. Cheng, P. Zhang, C. Li, Q. Liang, Y. Xu, P. Li, S. Pu, Y. Niu, and F. Wu, "Trie++: Towards end-to-end information extraction from visually rich documents," *arXiv preprint arXiv:2207.06744*, 2022.

[29] R. Sennrich, B. Haddow, and A. Birch, "Improving neural machine translation models with monolingual data," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

[30] K. Lee, K. Guu, L. He, T. Dozat, and H. W. Chung, "Neural data augmentation via example extrapolation," *arXiv preprint arXiv:2102.01335*, 2021.

[31] S. Y. Feng, V. Gangal, J. Wei, S. Chandar, S. Vosoughi, T. Mitamura, and E. Hovy, "A survey of data augmentation approaches for NLP," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, Aug. 2021.

[32] D. Kaushik, E. Hovy, and Z. C. Lipton, "Learning the difference that makes a difference with counterfactually-augmented data," *arXiv preprint arXiv:1909.12434*, 2019.

[33] Z. Wang and A. Culotta, "Robustness to spurious correlations in text classification via automatically generated counterfactuals," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 16, 2021, pp. 14 024–14 031.

[34] V. Atliha and D. Šešok, "Text augmentation using bert for image captioning," *Applied Sciences*, vol. 10, no. 17, 2020.

[35] K. Kafle, M. Yousefhussien, and C. Kanan, "Data augmentation for visual question answering," in *Proceedings of the 10th International Conference on Natural Language Generation*, Sep. 2017.

[36] M. Yokota and H. Nakayama, "Augmenting image question answering dataset by exploiting image captions," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, May 2018.

[37] J. Andreas, "Good-enough compositional data augmentation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Jul. 2020.

[38] R. Pryzant, Z. Yang, Y. Xu, C. Zhu, and M. Zeng, "Automatic rule induction for efficient semi-supervised learning," *arXiv preprint arXiv:2205.09067*, 2022.