FieldSwap: Data Augmentation for Form-Like Documents

Anonymous ACL submission

Abstract

Extracting structured data from visually rich documents like invoices, receipts, financial statements, and tax forms is key to automating many business workflows. Building extraction models in this space typically requires a large number of high-quality training examples. We propose a novel data-augmentation technique called FieldSwap for such extraction problems. FieldSwap converts a candidate for a source field into a candidate for a target field by replacing a key phrase indicative of the source field with a key phrase indicative of the target field. Using experiments on five different datasets, we show that training on data augmented with FieldSwap improves performance by 1–11 F1 points at low data setting (10–100 documents). We demonstrate that FieldSwap is effective when key phrases are manually specified or inferred automatically from the training data.

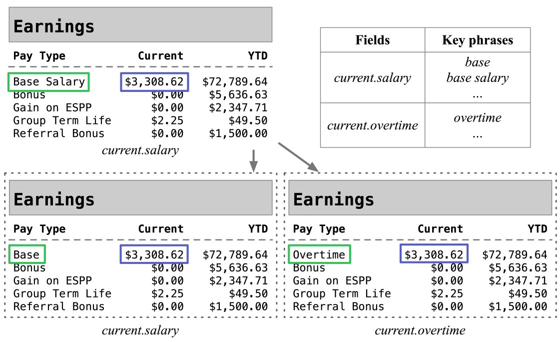


Figure 1: Example of FieldSwap on a paystub document. The source field S is *current.salary* ($3,308.62) and has key phrase "Base Salary". FieldSwap generates two synthetic examples. At bottom left, the phrase is replaced with "Base", another key phrase of *current.salary*, the field label for the instance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

1 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 (Lee et al., 2022; Garncarek et al., 2021a; Xu et al., 2020; Wu et al., 2018; Sarkhel and Nandi, 2019). 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 (Jaume et al., 2019; Park et al., 2019; Huang et al., 2019; Stanisławek et al., 2021), 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. We propose a novel data augmentation technique called *FieldSwap* loosely inspired by the success of mixing approaches in the image domain (Naveed, 2021). We exploit the observation that most fields in a document are indicated by a short key phrase (like "Amount Due" for the *total due* field in an invoice or "SSN" for the *Social Security number* field in a US tax form). FieldSwap generates synthetic examples for a target field T from examples for a source field S by replacing the key phrase associated with S with the key phrase for T.

Figure 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*. The FieldSwap algorithm generates synthetic examples by replacing key phrases in one field with key phrases in another field. This can help to regularize the model by presenting the same field with different values and surrounding contexts. However, it is important to choose the source-target pairs carefully. Note that multiple fields may have the same key phrase. For example, substitute the key phrase "Base Salary" to "Overtime" for the instance of the year_to_date.salary field would turn it into an instance of year_to_date.overtime and *not* an instance of current.overtime.

The key questions are: (a) how do we infer the key phrases corresponding to each field, and (b) what field pairs should be considered for generating these synthetic examples? We show that having a human expert supply a few key phrases works surprisingly well. The challenge then becomes how to infer key phrases automatically when only a small number of labeled examples are present. We show 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. To find good field pairs for the swap, we show that simply considering all pairs of fields of the same base type (e.g. date, money) 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 resulted in increases of 1–1.5 F1 points (Huang et al., 2022). 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.

Note that FieldSwap differs from simple text augmentation such as random swap or synonym replacement (Wei and Zou, 2019), which we argue is *not* effective for form extraction tasks since form extraction relies heavily on anchoring on specific key phrases in the document that define each form field. However, key phrases are not annotated, and thus, finding and swapping them is non-trivial.

We make the following contributions in this paper:

- We introduce a 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 several real-world datasets, we show that *FieldSwap* is effective—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 improvement up to 14 F1 points.

2 FieldSwap

FieldSwap exploits the property that form fields are very often indicated by key phrases (Majumder et al., 2020). 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 4.3.2). 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 mappings.

2.1 Automatically Inferring Key Phrases

We observe most fields have short key phrases, meaning only a few tokens in the neighborhood of a field instance matters. We thus define a method for measuring neighbor importance and identify important tokens. We infer important phrases from important tokens and then aggregate them to infer a set of key phrases for the field.

2.1.1 Neighbor Importance Measurement

We use the binary classifier architecture described in (Majumder et al., 2020) for the purpose of measuring neighbor importance. In this architecture (Figure 2), base type candidates are first extracted from an OCR'ed document using common-off-the-

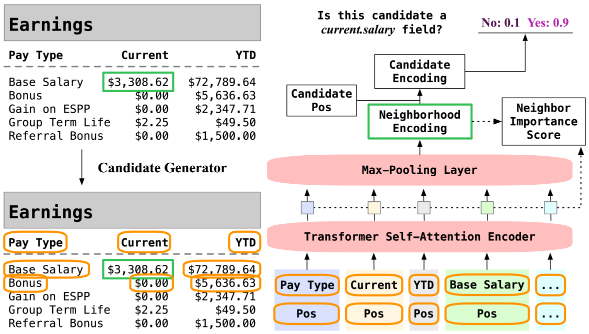


Figure 2: Architecture of the candidate-based extraction model. Neighboring tokens of a *current.salary* candidate (e.g. $\$3,308.62$) are fed into a Transformer-based encoder and a max-pooling layer to generate a *Neighborhood Encoding*, which is concatenated with a candidate position embedding to make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

shelf annotators like date and number annotators. For each candidate, the model encodes each neighbor 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 the purpose of finding important neighbors. To measure the *importance scores* of each neighbor tokens for a candidate, we calculate *cosine similarity* between the model's intermediate output on the encoding of each individual neighbor token and candidate *Neighborhood Encoding*.

We train the model on a large out-of-domain dataset and use it directly to get candidate *Neighborhood Encoding* on the target domain. The intuition is that the neighbor's relative position plays a critical role in identifying important neighbors and those positional signals are usually 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 positive candidates of target domain, so we generate candidates from ground truth instances of fields directly.

2.1.2 Inferring Important Phrases

After obtaining the importance scores of each neighbor for each candidate, we apply *Sparsemax* (Martins and Astudillo, 2016) across the importance scores to get a sparse output of the neighbors with non-zero scores. We consider these neighbors as the set of *important neighbor tokens*. We use an OCR service[1] that detects characters, tokens, and lines in the document. 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. Using these signals we form *important phrases* by concatenating all tokens on the same OCR line as long as one token is considered an important neighbor token. This is based on our observation that a key phrase is usually short and lies in a single line. Leveraging OCR lines to infer important phrases also makes the process more tolerant to the model's recall loss on important neighbor tokens. As long as the model finds one important token, we'll be able to infer the longer phrase, if it exists. We define *phrase importance score* as the average token importance score in the phrase.

2.1.3 Aggregation and Ranking by Field

Once all important phrases and importance scores have been gathered for all candidates, 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 candidate C_i that has important phrase P. We calculate $Importance(F, P) = 1 - \exp(\Sigma_i log(1 - Score(F, P, C_i)))$ as the measurement of how P relates to F. This measurement prefers phrases with higher importance scores and frequency across F candidates. 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.

2.1.4 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

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

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.

2.2 Field Pair Mappings

We explored three options to determine which fields can be swapped with each other.

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 key phrase variation.

Type-to-Type swap. Each field is associated with a general base type such as date, number, or address. 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 mapping 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. 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.

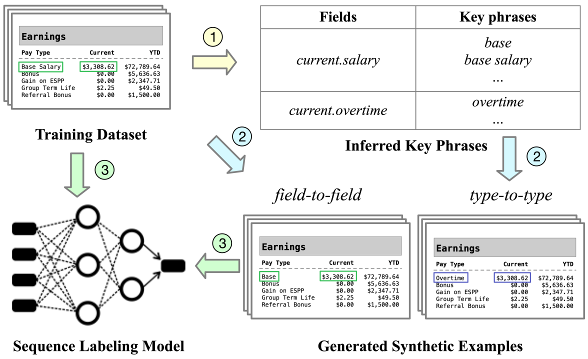


Figure 3: Overall processing procedures. In step 1, key phrases of all fields are inferred from the training dataset. In step 2, either field-to-field or type-to-type FieldSwap augmentation is applied to the training dataset. In step 3, a form extraction model (such as sequence labeling model) is trained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

2.3 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 (Xu et al., 2020; Lee et al., 2022), 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?

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.

2.3.1 Implementation Details

For each document in the training data, we iterate through all source-to-target pairs in the FieldSwap configuration, 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 source phrase in the configuration, then no synthetic documents are generated. Furthermore, if, after replacing the source key phrase with the target's, we are left with an unchanged document, we also omit the synthetic document. This helps prevent us from creating semantically incorrect synthetics, such as the case previously described in Section 2.2, when two fields have the same key phrase but are semantically different (e.g., *current.bonus* vs. *year_to_date.bonus*).

3 Human Expert

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 the automatically detected key phrases and field pairs, we apply human inputs to protect FieldSwap from some error-prone situations. For example, in configuring key phrases, some fields, such as *company_name* and *company_address*, do not have clear key phrases, so we skip these fields for FieldSwap entirely in the *human expert* setup. Other fields, particularly rare fields, might not have enough labeled examples in the train set, so 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 in the document. We believe using FieldSwap with this setting produce more useful synthetic documents than field-to-field setting, and less bad synthetic documents than type-to-type setting.

4 Experiments

4.1 Dataset

We evaluate FieldSwap on 5 datasets of form-like documents, including two public datasets (FCC Forms(Wang et al., 2022), FARA(Wang et al., 2022)) and three proprietary datasets (Earnings, Brokerage Statements, Loan Payments). Each dataset corresponds to a different document type. We herein also refer to document types as domains. All field types are defined in the schema and assigned with one of the five base types: string, address, money, date and number. For each domain, we evaluate on a fixed hold-out test set. Dataset statistics are summarized in Table 2 of Appendix A.

4.2 Experimental Setup

Automatically inferring key phrases. We use the model architecture described in Section 2.1.1 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 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 mappings using the method described in Section 3, 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 (Lee et al., 2022) and follow the same unsupervised pretraining 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 on all entities in the validation split.

Evaluation. We train baseline form extraction models on the training set without synthetic documents. We add the synthetic documents generated with FieldSwap to the training set and train new form extraction models follow the same procedure. We compare end-to-end F1 scores of the trained models to evaluate 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 2 of Appendix A), and (ii) 3 model training trials. This amounts to a total of 9 experiments for a given domain and train 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.

4.3 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?

4.3.1 Automatic FieldSwap

We test both field-to-field and type-to-type field mappings with automatically inferred key phrases. As shown in Figure 4, FieldSwap, in general, leads to neutral or better performance of all datasets and across all train set sizes we evaluated on. For instance, FieldSwap improves average macro-F1 by 1–4 points on FCC Forms, by 2–5 points on Brokerage Statements, and by 4–11 points on Earnings.

Field-to-Field vs Type-to-Type. We observed type-to-type swap performs better than field-to-field swap when training set size is small (10 documents), as training set size increases (50–100 documents), field-to-field swap catches up or wins over. As shown in Table 4 of Appendix B, type-to-type generally creates 3-10× more synthetic documents than field-to-field. However, it also has more chances to generate contradictory synthetic examples, as discussed in Section 2.2. When the training set is small, larger amount of synthetic documents helps more. However, as the training set size increases, field-to-field swap becomes more effective as it gets enough source examples to produce synthetics and is less likely to create bad synthetics that might harm the model.

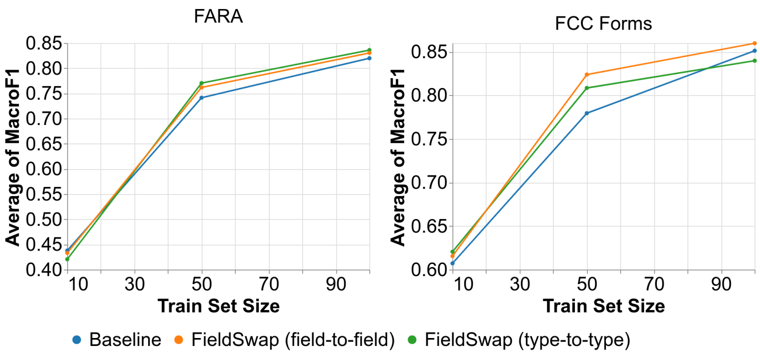
Macro-F1 vs Micro-F1. Current form extraction models perform poorly for rare fields when there is only a small amount of labeled data. We believe FieldSwap is most helpful in this situation since it can use the labeled training examples from other frequent fields to generate synthetic training examples for rare fields. Therefore, we focus on macro-F1.

However, we also observe that similar improvements hold when evaluating using Micro-F1. As shown in Figure 5 of Appendix C, the same pattern of results still holds. For instance, FieldSwap improves the average micro-F1 by 2–5 points for Earnings domain, and by 1–5 points for Brokerage Statements domain. The improvement gain is less than what we see with macro-F1 though, indicating that the most improvement have come from rare fields, which aligns with our hypothesis.

4.3.2 FieldSwap with Human Expert

We compare the performance between automatic FieldSwap and FieldSwap using human expert-curated phrases and field pair mappings on two domains. As shown in Figure 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 Earnings at 50–100 document, and 4 points for Loan Payments at 10 document.

The gap is mostly attributed to rare fields, as shown in Table 1. For example, *year_to_date.sales_pay* field has particularly low frequency, it only exists in 3.9% of train documents. In a low data setting, it's possible that there are no or very few labeled examples for such rare fields, leading to few-shot or even zero-shot scenarios. Relying on an expert to supply key phrases that are *not* present in the small training set results in a substantial advantage over the automatic approach. In practice, the decision of whether or not to use human input depends on specific scenarios.



(a) Public Datasets

(b) Proprietary Datasets

Figure 4: Experiment results on different domains with different train doc sizes. Under each setting, we repeat the experiments with different random seeds as mentioned in Section 4.2. We report the average Macro-F1 scores.

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

Table 1: Fields with the largest mean F1 score gaps between automatic (field-to-field) and human expert setting when trained on 50 documents for Earnings domain. Frequency refers to the fraction of documents that contain said field in a pool of 2000 documents.

4.3.3 Discussions

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

Effect of field type. We associate fields with five base types (i.e. address, date, money, number, string) in our settings. Among the 5 datasets we evaluated on, there are only two fields with *number* type (each in different domain), making the results not representative. Thus, we only study the effect of the remaining 4 base types. We believe number type shall have similar pattern with money type, as they are somewhat alike.

We study the effect of FieldSwap on different field types in Loan Payments domain across all train set sizes. As shown in Figure 6 of Appendix D, we observe the improvement of FieldSwap mainly come from *date* and *money* fields, while we see negative effects on *address* and *string* fields. Recall that *string* and *address* fields usually do not have indicative key phrases, FieldSwap should not generate augmentations. For the most part, our implementation does exactly this, however at times it does fail and find spurious phrases for these fields. This leads to bad synthetic documents that harm the model. In our human expert setting, we explicitly exclude FieldSwap for fields that do not have key phrases. We see the negative effect dissipates.

Effect of document type. As shown in Figure 4, the biggest improvement we observe is on the Earnings domain. Compared to other document types, most of the fields in Earnings are in tabular format, with similar base type (i.e. money, date), and have clear and succinct phrase indicators. We believe FieldSwap is most helpful when dealing with document types with such characteristics. Furthermore, since ordering of fields in 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 2.2. Yet, even in the presence of these potentially contradicting pairs, type-to-type mapping still improves the performance across all train set sizes we evaluated on. This demonstrates that the proposed method tolerates a small number of these contradictory synthetic examples.

The improvement gain on FARA domain is very small. This is because this corpus contains only a handful of fields, 4 of the total 6 fields are string type, which FieldSwap does not work well. The other 2 fields belong to different base types and are thus not swappable. However, FieldSwap still maintain neutral or slightly better results throughout the learning curve.

5 Related Work

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 (Wei and Zou, 2019)—are easier to implement but have limited benefit, whereas model-based techniques—such as back-translation (Sennrich et al.) and example extrapolation (Lee et al., 2021)—are more difficult to develop but offer greater benefit (Feng et al., 2021). FieldSwap contains elements of both categories, as it changes (possibly automatically inferred) key phrases based on a set of swap rules.

Feng et al. (2021) 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 (Dai and Adel, 2020).

Other data augmentation techniques have been used for multimodal tasks that combine text and vision, such as image captioning (Atliha and Šešok, 2020) and visual question answering (Kafle et al., 2017; Yokota and Nakayama, 2018). 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 (Andreas, 2020). 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 identity or 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).

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 (Appalaraju et al., 2021; Garncarek et al., 2021b; Huang et al., 2022), but these encoders require a large amount of pre-training data, although they do exhibit good downstream task data efficiency during fine-tuning (Sage et al., 2021). Large amounts of training data are also required by span classification approaches (Majumder et al., 2020; Tata et al., 2021), sequence labeling approaches (Aggarwal et al., 2020; Lee et al.), and end-to-end approaches (Cheng et al., 2022). Rather than suggest a new model architecture, we propose a method for augmenting form extraction data.

6 Conclusions

In this paper, we describe a data-augmentation technique designed for extraction problems on visually rich documents. We exploit the fact that many fields have a "key phrase" to indicate them: we generate an augmented example for a target field by taking an example of a source field and replacing the key phrase with that of the target field. Experiments on a variety of datasets show that this simple technique is very effective for small training sets (10–100 documents), with improvements of 1–11 macro-F1 points.

This result opens up two interesting directions for future work. First, how do we design a version of FieldSwap that works better with the complex situation we described in Section 2.3? Second, there are several extensions to FieldSwap that are worth investigating. When does swapping across document types help? Can we use a pre-trained LLM instead of a human expert to generate a set of key phrases given the name or description of a field? Can we learn information about key phrases from an unlabeled corpus to enable semi-supervised learning (Pryzant et al., 2022)?

References

Milan Aggarwal, Hiresh Gupta, Mausoom Sarkar, and Balaji Krishnamurthy. 2020. Form2Seq : A framework for higher-order form structure extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jacob Andreas. 2020. Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Srikar Appalaraju, Bhavan Jasani, Bhargava Urala Kota, Yusheng Xie, and R. Manmatha. 2021. Docformer: End-to-end transformer for document understanding. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 973–983.

Viktar Atliha and Dmitrij Šešok. 2020. Text augmentation using bert for image captioning. *Applied Sciences*, 10(17).

Zhanzhan Cheng, Peng Zhang, Can Li, Qiao Liang, Yunlu Xu, Pengfei Li, Shiliang Pu, Yi Niu, and Fei Wu. 2022. Trie++: Towards end-to-end information extraction from visually rich documents. *arXiv preprint arXiv:2207.06744*.

Xiang Dai and Heike Adel. 2020. An analysis of simple data augmentation for named entity recognition. In *Proceedings of the 28th International Conference on Computational Linguistics*.

Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for NLP. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*.

Łukasz Garncarek, Rafał Powalski, Tomasz Stanisławek, Bartosz Topolski, Piotr Halama, Michał Turski, and Filip Graliński. 2021a. Lambert: layout-aware language modeling for information extraction. In *International Conference on Document Analysis and Recognition*, pages 532–547. Springer.

Łukasz Garncarek, Rafał Powalski, Tomasz Stanisławek, Bartosz Topolski, Piotr Halama, Michał Turski, and Filip Graliński. 2021b. Lambert: Layout-aware language modeling for information extraction.

Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022. Layoutlmv3: Pre-training for document ai with unified text and image masking. *arXiv preprint arXiv:2204.08387*.

Zheng Huang, Kai Chen, Jianhua He, Xiang Bai, Dimosthenis Karatzas, Shijian Lu, and CV Jawahar. 2019. Icdar2019 competition on scanned receipt ocr and information extraction. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1516–1520. IEEE.

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

Kushal Kafle, Mohammed Yousefhussien, and Christopher Kanan. 2017. Data augmentation for visual question answering. In *Proceedings of the 10th International Conference on Natural Language Generation*.

Chen-Yu Lee, Chun-Liang Li, Timothy Dozat, Vincent Perot, Guolong Su, Nan Hua, Joshua Ainslie, Renshen Wang, Yasuhisa Fujii, and Tomas 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*.

Chen-Yu Lee, Chun-Liang Li, Timothy Dozat, Vincent Perot, Guolong Su, Nan Hua, Joshua Ainslie, Renshen Wang, Yasuhisa Fujii, and Tomas Pfister. 2022. FormNet: Structural encoding beyond sequential modeling in form document information extraction. In *ACL*.

Kenton Lee, Kelvin Guu, Luheng He, Tim Dozat, and Hyung Won Chung. 2021. Neural data augmentation via example extrapolation. *arXiv preprint arXiv:2102.01335*.

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 *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Andre Martins and Ramon Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International conference on machine learning*, pages 1614–1623. PMLR.

Humza Naveed. 2021. Survey: Image mixing and deleting for data augmentation. *arXiv preprint arXiv:2106.07085*.

Seunghyun Park, Seung Shin, Bado Lee, Junyeop Lee, Jaeheung Surh, Minjoon Seo, and Hwalsuk Lee. 2019. Cord: A consolidated receipt dataset for post-ocr parsing.

Reid Pryzant, Ziyi Yang, Yichong Xu, Chenguang Zhu, and Michael Zeng. 2022. Automatic rule induction for efficient semi-supervised learning. *arXiv preprint arXiv:2205.09067*.

Clément Sage, Thibault Douzon, Alex Aussem, Véronique Eglin, Haytham Elghazel, Stefan Duffner, Christophe Garcia, and Jérémy Espinas. 2021. Data-efficient information extraction from documents with pre-trained language models. In *International Conference on Document Analysis and Recognition*, pages 455–469. Springer.

Ritesh Sarkhel and Arnab Nandi. 2019. Visual segmentation for information extraction from heterogeneous visually rich documents. In *Proceedings of the 2019 International Conference on Management of Data*, pages 247–262.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Tomasz Stanisławek, Filip Graliński, Anna Wróblewska, Dawid Lipiński, Agnieszka Kaliska, Paulina Rosalska, Bartosz Topolski, and Przemysław Biecek. 2021. Kleister: key information extraction datasets involving long documents with complex layouts. In *International Conference on Document Analysis and Recognition*, pages 564–579. Springer.

Sandeep Tata, Navneet Potti, James B Wendt, Lauro Beltrão Costa, Marc Najork, and Beliz Gunel. 2021. Glean: structured extractions from templatic documents. *Proceedings of the VLDB Endowment*, 14(6):997–1005.

Zilong Wang, Yichao Zhou, Wei Wei, Chen-Yu Lee, and Sandeep Tata. 2022. A benchmark for structured extractions from complex documents. *arXiv preprint arXiv:2211.15421*.

Jason Wei and Kai Zou. 2019. 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)*.

Sen Wu, Luke Hsiao, Xiao Cheng, Braden Hancock, Theodoros Rekatsinas, Philip Levis, and Christopher Ré. 2018. Fonduer: Knowledge base construction from richly formatted data. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1301–1316.

Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, et al. 2020. LayoutLMv2: Multi-modal pre-training for visually-rich document understanding. *arXiv preprint arXiv:2012.14740*.

Masashi Yokota and Hideki Nakayama. 2018. Augmenting image question answering dataset by exploiting image captions. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

A Datasets

Table 2 presents the total number of train documents available in the pool for each document type, along with the number of total fields. We select a subset of documents at random from the pool to create the training sets for our experiments. Table 3 presents number of fields with different base types for each document type.

B Augmentation Stats

Number of synthetic documents various across different sampled train sets, for each document type and train set size, we presents the average number of synthetic documents generated by FieldSwap with different settings in Table 4.

C Micro F1 Results

Figure 5 shows the average of micro-F1 scores for different document types across different train set sizes.

D Effect of Field Type

Figure 6 shows the F1 score differences of FieldSwap with different settings over the baseline for different field types and training set sizes on Loan Payments domain.

Document Type	# Fields	Train Docs Pool Size	Test Docs
FARA (Wang et al., 2022)	6	200	300
FCC Forms (Wang et al., 2022)	13	200	300
Brokerage Statements	18	294	186
Earnings	23	2000	1847
Loan Payments	35	2000	815

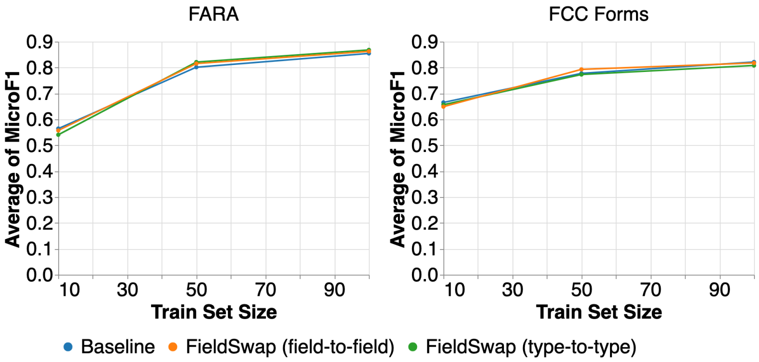
Table 2: Datasets. To plot learning curves we select a subset of documents at random from the corpora's larger pool to create the training sets for our experiments.

	Field Type				
Document Type	**Address**	**Date**	**Money**	**Number**	**String**
FARA (Wang et al., 2022)	0	1	0	1	4
FCC Forms (Wang et al., 2022)	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

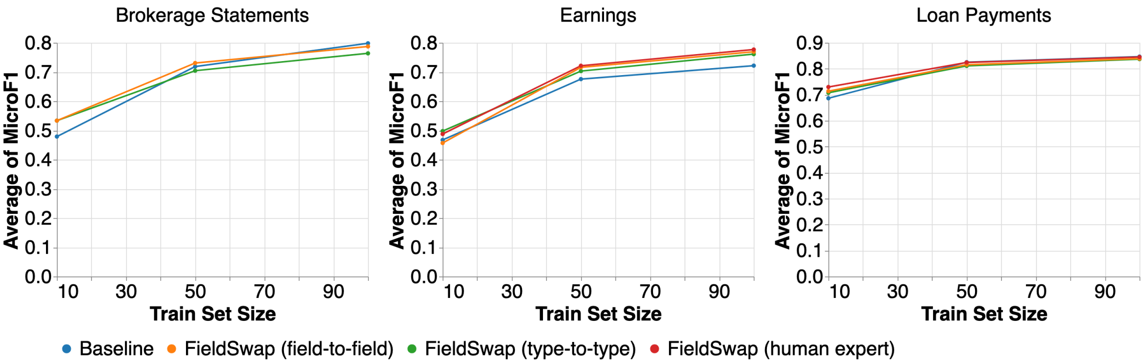
Table 3: Number of fields with different base types for each document type.

		Number of Synthetic Documents		
Domain	**Original Train 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

Table 4: Average number of FieldSwap synthetic documents at different train set size for each document type.



(a) Public Datasets



(b) Proprietary Datasets

Figure 5: Average of micro-F1 scores on different domains with different train doc sizes. Under each setting, we repeat the experiments with different random seeds as mentioned in Section 4.2.

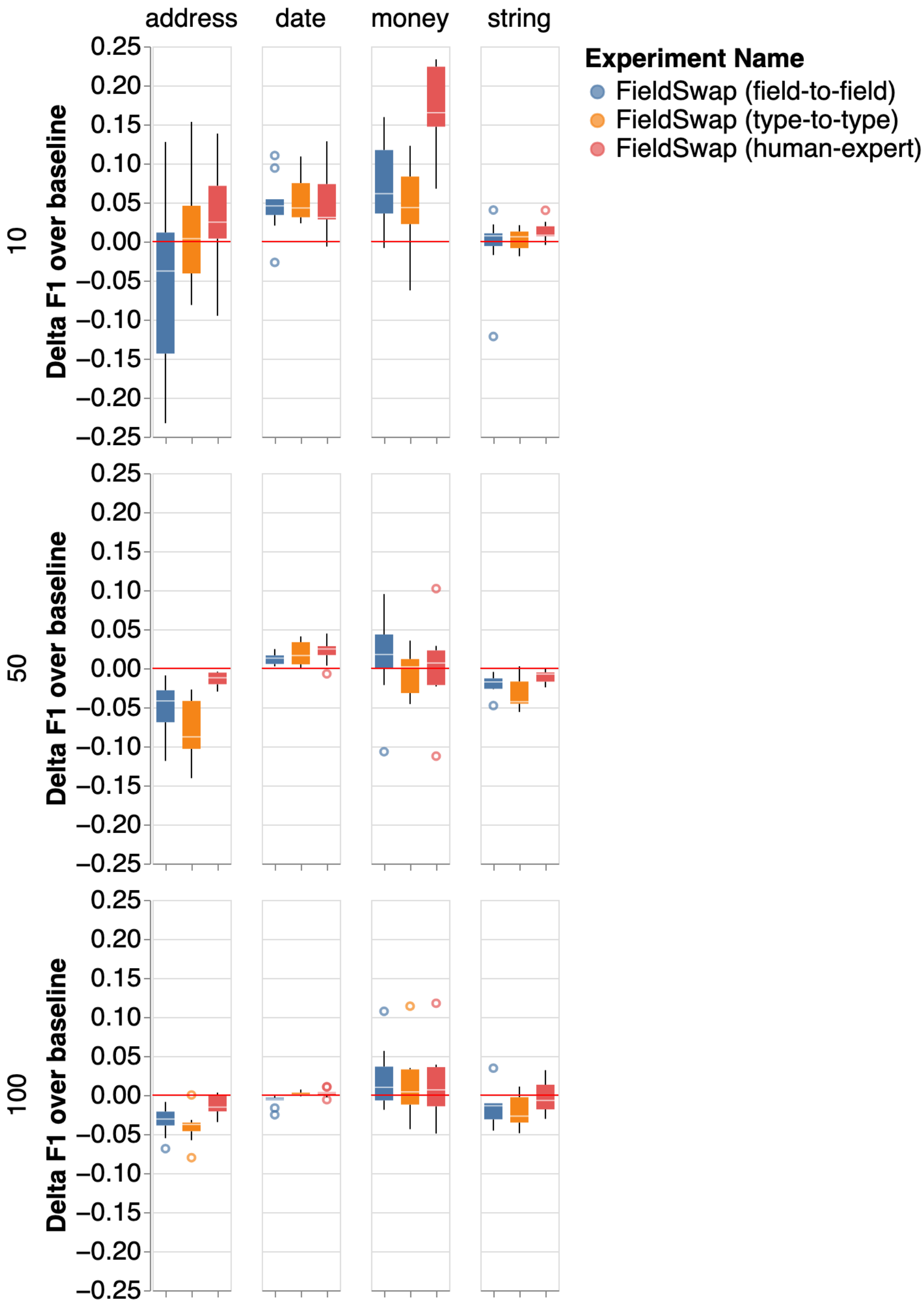


Figure 6: Field F1 score differences of FieldSwap over baseline on Loan Payments domain. The length of each box plot shows the distance between the upper and lower quartiles. Each whisker extends to the furthest data point in each wing that is within 1.5 times the IQR. The line in the middle of the boxplot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