Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Mind the Decoder: Reassessing Learning Gains in Low-Label Line-Item Recognition

Linda-Sophie Schneider

linda-sophie.schneider@fau.de

Friedrich-Alexander-Universität Erlangen-Nürnberg

Thomas Gorges

Friedrich-Alexander-Universität Erlangen-Nürnberg

Julian Thomas

Friedrich-Alexander-Universität Erlangen-Nürnberg

Mathias Zinnen

Friedrich-Alexander-Universität Erlangen-Nürnberg

Vincent Christlein

Friedrich-Alexander-Universität Erlangen-Nürnberg

Research Article

Keywords:

Posted Date: November 26th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11757876/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Mind the Decoder: Reassessing Learning Gains in Low-Label Line-Item Recognition

Linda-Sophie Schneider [1*], Thomas Gorges [1], Julian Thomas [2], Mathias Zinnen [1], Vincent Christlein [1]

[1*]Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany.
[2]Chair of Applied Cryptography, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany.

*Corresponding author(s). E-mail(s): linda-sophie.schneider@fau.de;

Abstract

Purpose: Line-item recognition extracts billed items from invoices for automated processing. A model scores candidate text segments, and a decoder assembles these scores into fields and items. We ask whether gains from training with automatically checkable invoice rules persist when the same models use a stronger decoder.
Methods: We trained adapters on frozen RoBERTa-base and LayoutLMv3 encoders with few labelled DocILE invoices. We then continued training with two existing approaches: verifier-reward training (VRT), which rewards outputs that satisfy invoice rules, and rule-filtered self-training, which learns from predictions that satisfy them. Each model was evaluated, without retraining, under a segment and a hybrid decoder. Controls included confidence-based filtering, a fully fine-tuned model, unseen layouts and CORD receipts.
Results: With 250 labelled training invoices, VRT improved on the supervised model by 6.6 and 5.2 F1 points under the segment decoder, but only 0.8 and 0.1 under the hybrid decoder. VRT gains disappeared with fewer labels, on unseen layouts and on receipts. Gains from rule-filtered self-training with 50 and 100 labels persisted under the hybrid decoder. A confidence filter recovered most of these gains on the validation split and about half on unseen layouts with 50 labels. Confidence-based self-training of the fully fine-tuned model scored highest with 100 and 250 labels.
Conclusion: Most of the VRT gain could also be achieved through decoding: the hybrid decoder uses a rule to separate items that VRT had learned to separate. Before attributing a training gain to learning, studies should evaluate the same trained models under a second decoder.

Keywords: document information extraction, line-item recognition, label efficiency, constrained learning, structured decoding, self-training

1 Introduction

Invoice processing is a central application of document analysis. Invoices list billed goods or services as line items, each of which records one of them through fields such as its description, quantity, unit price and amount.Traditionally, they are extracted by processing the output of optical character recognition (OCR) to predict fields, their types and their grouping into items. A field is of use only if all three

predictions are right. Since invoice layouts differ from one issuer to the next, extraction systems must account for layout variation, for example by learning from labelled examples.

Labelling invoices is expensive. With only a few labelled invoices, one can either write keywords and grammars by hand or train a model. Self-training lets such a model also learn from its own predictions on unlabelled documents [37, 45]. Learning from unlabelled documents, however, needs a signal that separates good outputs from bad ones. One such signal comes from invoice rules that a program can check, such as amount formats, the number of amounts per item and the arithmetic between quantity, unit price and amount. These rules can select extractions to train on [10], restrict the output during decoding [9] or guide both training and decoding [3, 6]. Yet as a training signal, they can only judge an output that a model proposes.

A training gain, however, is not based on the model's predictions directly. A learned extraction system consists of two parts: a model, which is trained and assigns probabilities to candidate fields and items, and a decoder, which assembles these probabilities into the final output. Because labels are scarce and training should be feasible on CPUs, we keep a pre-trained encoder frozen and train only a small adapter on its features. For each candidate text segment, the adapter estimates how likely it is to be a field of each type, and for each band of consecutive rows how likely it is to form one item. A decoder without trainable parameters then turns these probabilities into fields and items. F1 is computed on this decoded output and therefore belongs to a pair of a trained model and a decoder. The model may correctly recognize the amounts of two adjacent items, but if the decoder merges their rows into one item, that item holds two amounts and its fields count as errors. Training with invoice rules can teach the model to avoid such outputs, but a better decoder might avoid them without any training. When training methods are compared under one fixed decoder, these two effects cannot be told apart. We therefore ask whether gains from further training with invoice rules persist when the same models are evaluated with a stronger decoder.

To answer this question, we perform a decoder audit. Since the decoder is not trained, each trained model is kept fixed while only the decoder is exchanged, at the cost of only one additional evaluation. A training gain that persists after the exchange provides evidence of a benefit beyond the original decoder. If the gain disappears, this suggests that the training mainly compensated for limitations of that decoder.

An audit is only as informative as the decoder it switches to. The official DocILE decoder, released with the DocILE baselines, reads the tags of a word tagger, which mark the field types of each word and its position in an item. It assembles fields and items from these tags, so it cannot process the segment scores of our adapters. Besides a standard segment decoder, we therefore build a hybrid decoder for models that score segments. On a fully fine-tuned tagger, which all three decoders can process, the segment decoder falls well short of the official decoder, whereas the hybrid decoder reaches its level. With it, we audit verifier-reward training (VRT) and rule-filtered self-training on the invoices of DocILE [33], with two encoders and few labelled documents, and VRT also on receipts. Most of the gain that VRT shows under the segment decoder can also be achieved by applying the hybrid decoder to the supervised model alone.

This leads to the following contributions:

- **Decoder audit.** A test requiring no retraining that assesses whether training gains persist under a different decoder. It applies to extractors whose fixed model outputs can be processed by alternative decoders with no trainable parameters. For our experiments, we introduce a hybrid decoder that brings segment models to the level of the official DocILE decoder.
- **Segment adapters for few labels.** A small adapter on a frozen encoder, trainable on CPUs, scores candidate segments and item bands instead of single words. It assigns a probability to every assembled output, so invoice rules can act on whole fields and items during training, and it lets field boundaries fall inside OCR words.
- **Most of the VRT gain is a decoding effect.** On invoices, unseen layouts, and receipts, improved decoding recovers most of the gain from VRT: a decoder rule separates the same items the model had learned to separate.
- **Self-training under the audit.** A comparison of self-training variants under the hybrid decoder. The gain of rule-filtered self-training survives the audit, and a plain confidence filter recovers most of it. With more labels, confidence-based self-training of a fully fine-tuned model leads.

2 Related work

Line-item recognition.

Our question concerns how the decoder affects a comparison, so we first review how line-item systems combine a model with a decoder. DocILE [33] and CORD [23] annotate invoice and receipt fields with their grouping into items. Layout-aware models such as LayoutLMv3 [12] and GraphDoc [39] combine textual, spatial or visual evidence. The official DocILE baselines pair a RoBERTa-base [21] or LayoutLMv3 word tagger with the official decoder, and competition entries assembled items with learned mergers, distance rules or item detectors [34]. For one such detector, line-item F1 changed with the overlap threshold that assigns fields to items [36]. Reported scores thus belong to pairs of a model and a decoder.

Learning from few labels.

Our models learn from few labelled invoices and from pseudo-labels on unlabelled ones. Template systems read header fields from documents of the same layout [29, 32], later with a generic keyword prior [25]. Other work studied active learning for document classification [30], distillation into smaller document models [20], and generative extractors trained or prompted with few documents [8, 16, 24]. Self-training learns from predictions on unlabelled documents, usually filtered by confidence [35], although under equal tuning its gain over supervised baselines is smaller than reported [22]. It has been applied to document extraction [37, 45] and handwriting recognition, by confidence filtering [41] or by training on all weighted transcription variants [17].

Domain knowledge in training.

Verifier-reward training and rule-filtered self-training put invoice rules into training, as earlier work did with other forms of domain knowledge. Hemmer et al. [10] validated language-model extractions against syntactic, task and domain rules before training on them. Knowledge can also shape the training signal through constraints, semantic loss [42] or weak supervision [26]. Chang et al. [3] evaluated training with and without constraints, each with and without constrained inference, and found constrained inference more important when labelled data were plentiful and constrained training more important when they were scarce. For word alignment, Ganchev et al. [6] separated the two effects by decoding models trained with and without constraints both with and without them. Distilled logic rules helped more in training than at test time alone [11]. Reinforcement learning with verifiable rewards scores outputs by rule checks under a Kullback–Leibler (KL) penalty [7, 19], which mainly makes correct answers easier to sample [44].

Domain knowledge in decoding.

Our hybrid decoder applies rules at decoding, so knowledge enters beyond training. Constraints can also be imposed only at inference, as in integer linear programming over independent classifiers [28]. Semi-Markov models [31], spatial dependency parsing [13] and Lazy-k search [9] organise local evidence into structured outputs. Generative extractors depend on decoding too, and constraining beam search to a template raised F1 [2].

Research gap.

Since the metric scores assembled outputs, comparisons of training methods must control the decoder, which adds to known sources of variation such as the training seed [27] and data resampling [1]. Among the document-extraction works cited here, none crosses training methods with decoders. We make this crossing for line-item recognition and test whether the gains of one rule filter and one verifier reward with a KL anchor [46] survive an exchange of the decoder. We also test whether invoice rules select pseudo-labels better than a confidence filter.

3 Method

Business documents such as invoices and receipts list the goods or services they cover as line items. Line-item recognition (LIR) asks a system to find the fields that describe these items, to give each field a type $t \in \mathcal{T}$, such as quantity or amount, and to group the fields into items. The input is a document d with OCR words $W(d)$ and their bounding boxes. An output y is a set of typed fields grouped into items, where each field is a span of text on one page and may extend over several rows. A field is recovered if its extent, its type and its item all agree with the annotated output y^* of d.

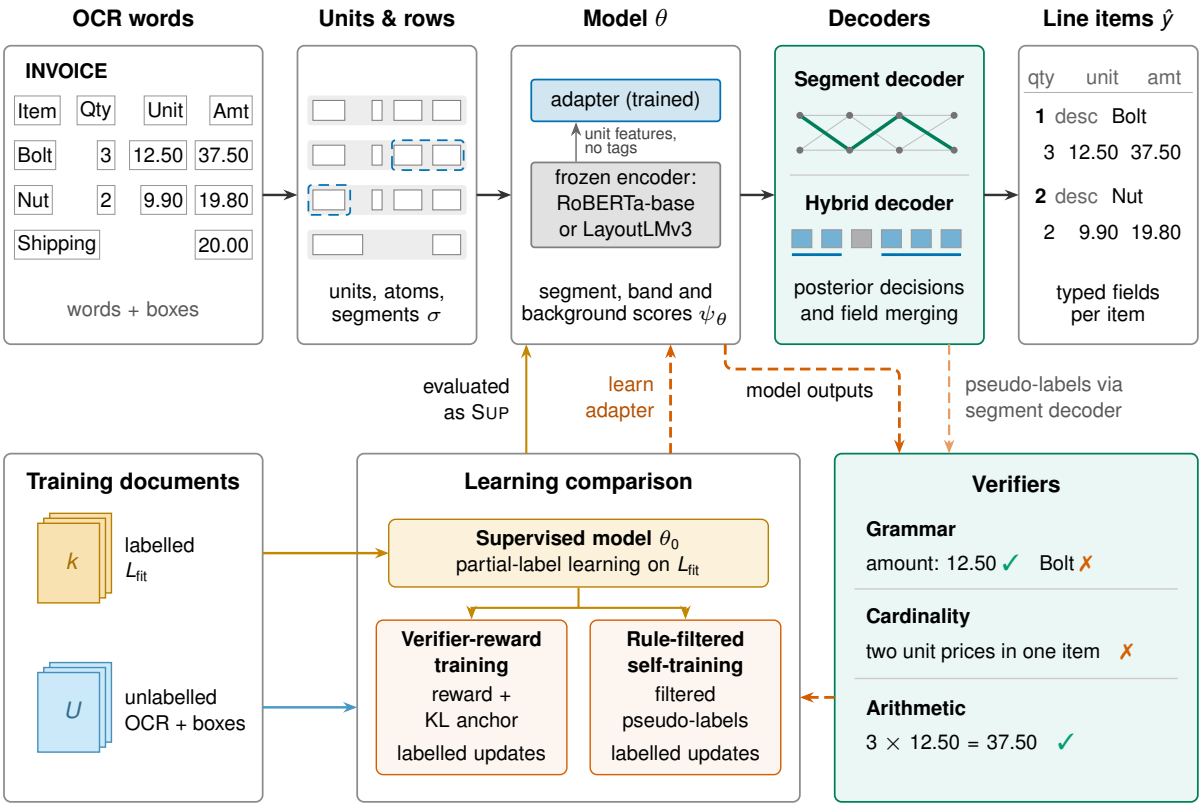


Fig. 1: Pipeline of the adapters on the frozen encoders, and learning comparison. The words and boxes that OCR returns are grouped into rows and cut into units. The model θ, a frozen encoder with a trained adapter, scores segments, bands and background, and the segment or hybrid decoder assembles fields and items. Training the adapter on L_{fit}, a subset of the k labelled documents, gives the supervised model θ_0 (SUP). Verifier-reward training (VRT) and rule-filtered self-training (ST) continue from SUP with unlabelled documents U. The verifiers judge model outputs before decoding to supply the reward of VRT, and they filter the pseudo-labels that the segment decoder writes for ST.

Every system in this study consists of a model and a decoder. The model is trained and assigns scores to parts of the document. The non-trainable decoder turns these scores into fields and items. Decoders can be exchanged after training, and we call the comparison of one trained model under several decoders the *decoder audit*. Training uses labelled documents alone or, in addition, unlabelled ones, with or without invoice rules. The k labelled documents of a label budget are split into a fitting part L_{fit}, from which the adapters learn, and a development part L_{dev}, which serves for configuration choices and monitoring, and U is a pool of unlabelled documents. The section first presents the invoice verifiers and then the model, the decoders, the training methods and finally the audit. Fig. 1 shows the pipeline of the adapters, from the OCR words to the assembled items.

3.1 Invoice verifiers

Invoices follow rules that a program can check without any annotation. An amount has a number format, an item lists each price once, and quantity times unit price gives the amount. We call a fixed check of one such rule an invoice verifier. It reads only the OCR text of the fields in an output and their grouping into items, and it reports which fields or items violate the rule. We use three verifiers, for grammar, cardinality and arithmetic. These verifiers are fallible, since even annotated outputs fail them at times. How much a violation costs therefore depends on how reliable the verifier

is. We measure the rate ρ_c at which annotated outputs of the training pool pass verifier c and set the cost of a violation to the log-odds $\lambda_c = \operatorname{logit} \rho_c$. A verifier that annotated outputs almost always pass thus acts almost as a hard constraint, and one that they often fail counts for little.

Grammar verifier. Amounts, dates and quantities follow formats, so a format check usually detects a wrong field boundary. Thirteen of the 19 field types therefore have a manually defined regular grammar, whereas free-text types such as description have none. The verifier penalises a field whose text fails its grammar, and it penalises fragments, namely a failing field that would pass when extended within its row and same-type neighbours that pass when joined.

Cardinality verifier. A wrong boundary or grouping can leave an item with repeated amounts or unit prices. The verifier therefore allows at most one field per item for each of six monetary types $\mathcal{T}_6 \subset \mathcal{T}$, namely net and gross unit price, net and gross amount, discount and tax. Quantity stays free, since annotated items often list several, such as ordered and delivered quantities. No annotated item of the training pool violates this rule, so we apply Laplace smoothing to the pass rate, which gives a large but finite cost.

Arithmetic verifier. The amount of an item should be the product of its quantity and its unit price, net or gross, after surcharge and discount. The verifier checks this per item up to a small tolerance and does not judge an item that lacks one of these fields. An item violates the verifier if no reading balances. We call a pass decisive if the item would not balance with swapped types and the balance is not trivial. A balance is trivial, for instance, when unit price and amount are equal. Only a decisive pass earns a reward, whereas every other outcome of a verifier is at most a penalty.

3.2 From word taggers to segments

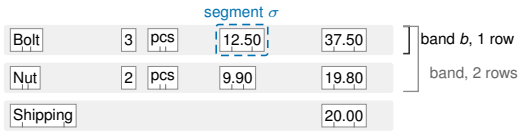
Leading layout-aware extractors [12, 38, 43], including the baselines of DocILE [33], are word taggers. A pre-trained encoder represents each OCR word, and a classification head tags the word with its field types and its position in a line item. The tagger decides on each word separately. Fields and items emerge only afterwards, when a fixed rule-based procedure assembles the tagged words.

Two properties of this design conflict with training from invoice rules. Such rules judge whole fields and items, such as the format of an amount or the arithmetic of an item. Yet, a tagger scores single words and assigns no probability to an assembled output. A reward on sampled outputs or a loss over all outputs that are consistent with an annotation therefore has nothing to act on. A tagger also decides on whole words, whereas an annotated field can end inside an OCR word. For instance, when a currency sign is attached to an amount.

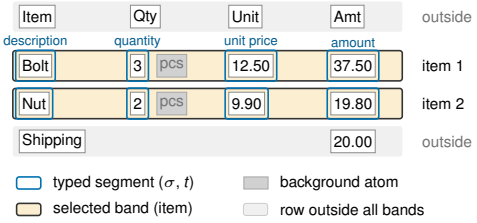
We therefore use a segment model, which scores candidate fields directly. The words of a document are grouped into rows and split into units. Each unit is a maximal sequence of characters of the same class, such as letters, digits or currency symbols. Separators between digits remain part of the number. A field boundary can thus fall inside an OCR word, for instance between a currency sign and an amount. We allow boundaries at two positions within each unit: after the first character and before the last, because annotated boundaries within units often occur at these positions. The pieces between all possible boundaries are called atoms, the smallest units of text that the model and the decoders distinguish. A candidate segment is any sequence of consecutive atoms within one row. A candidate item band is any sequence of consecutive rows. Both have a fixed maximum length. The model scores each typed segment, each background atom and each band, and the score of an output is the sum of the scores of its parts. The probability of an output is proportional to the exponential of its score, and because segments and bands are bounded in length, a dynamic program computes the normaliser exactly. The model is thus a proper distribution over complete outputs, with exact probabilities, samples and best outputs. Definition 1 fixes the length bounds and the notation; and Fig. 2 illustrates all introduced terms.

Definition 1 (Candidate space and output structure) Let a document d have n_r rows and n_a atoms.

- A segment $\sigma \in \Sigma(d)$ is a span of at most 12 units of consecutive atoms in one row.
- A band b is a range of at most 8 consecutive rows of one page.
- An output $y \in \mathcal{Y}(d)$ selects pairwise disjoint bands as items; in every row of a selected band it



(a) Candidate space



(b) Output y and its factors

Fig. 2: Example of Definition 1. Notches mark atom boundaries.

chooses non-overlapping segments and gives each a type $t \in \mathcal{T}$. All other atoms are background.

- The factors $\Phi(y)$ of y are its typed segments (σ, t), the background atoms in the rows of its selected bands, its selected bands, and its rows outside all selected bands.

The candidate space reaches almost all annotated fields. On the 500 training-pool documents on which this configuration was chosen, it contains a candidate with exactly the annotated extent for 97.5 % of the annotated line-item fields, counting the multi-row candidates that the segment decoder of Sect. 3.3 adds.

Since labels are scarce and training should be feasible on CPUs, we keep the encoder frozen, compute its features once and train only a small adapter. We use encoder RoBERTa-base [21] or LayoutLMv3 [12] without a tagging head and with sub-word vectors averaged per unit. The adapter has 0.47 M parameters. It maps unit vectors and box geometry to width 128 and refines them in two relation-aware message-passing layers over neighbour, row, column and header edges. Three heads then produce the factor scores of Definition 1, one for background from single units, one for typed segments from the pooled units of a segment and one for bands from the pooled rows of a band.

To supply layout regularities beyond the labelled examples, the adapter also receives a fixed rule prior with two learned weights. The prior is the output of RULES with templates, a training-free baseline that reads line items from header keywords, grammars and labelled templates. Every adapter on DocILE carries this prior, the purely supervised one included, so it does not separate the training methods.

Given a score $\psi_\theta(a)$ for every factor a, the model is the Gibbs distribution

$$p_\theta(y \mid d) = \frac{\exp F_\theta(y; d)}{Z_\theta(d)}, \qquad F_\theta(y; d) = \sum_{a \in \Phi(y)} \psi_\theta(a), \tag{1}$$

with $Z_\theta(d)$ the sum of $\exp F_\theta$ over $\mathcal{Y}(d)$ and marginals $\mu_\theta(a) = \Pr_{y \sim p_\theta(\cdot \mid d)}[a \in \Phi(y)]$. Although the number of outputs grows exponentially, segments and bands have bounded length. This permits exact inference with a running time linear in the number of atoms and typed candidates, as Proposition 1 shows. This claim concerns the distribution in Eq. (1) itself and not the fields that a decoder later assembles from it. The supervised loss uses the normaliser, the decoders use the best outputs and the marginals, and verifier-reward training uses the samples, the marginals and the Kullback–Leibler (KL) divergence.

Proposition 1 (Exact inference) *For the model Eq. (1) on the outputs of Definition 1, nest a semi-Markov [31] pass over the atoms of each row in an outer pass over the rows that selects bands. These passes compute* $\log Z_\theta(d)$*, all marginals* $\mu_\theta(a)$*, an exact sample and the* K *best outputs for fixed* K*, and for two parameter vectors* θ, θ' *they give*

$$\mathrm{KL}(p_\theta \,\|\, p_{\theta'}) = \sum_a \mu_\theta(a)\big(\psi_\theta(a) - \psi_{\theta'}(a)\big) - \log Z_\theta(d) + \log Z_{\theta'}(d), \tag{2}$$

each in $O(n_\mathrm{a} + |\Sigma(d)|\,|\mathcal{T}|)$ *time.*

The proof is given in Appendix A.1.

As a strong reference that follows the usual approach, we also fully fine-tune LayoutLMv3 as a word tagger and call this model FT. Section 3.5 describes its training, and Sect. 3.4 describes how our decoders read it.

3.3 Decoders

The model only scores, so a decoder has to choose one output and turn it into fields and items that the

metric can match. The official DocILE decoder [33], the rule-based assembly step of the taggers above, would be the natural choice. First, it reads the tags of a word tagger, decides each word on its own and groups the words by the predicted line item. Then, it merges words of the same type into fields, first joining neighbouring words within a text line and then combining the resulting blocks across lines, provided that the gaps between them remain below fixed multiples of the line height. No step looks at the document as a whole, and a word decided as background never becomes part of a field. Our segment model produces neither per-word tags nor item tags, so the official decoder cannot read it. We therefore built two decoders for the segment model, a segment decoder and a hybrid decoder. The segment decoder is the natural choice for a segment model, since it selects among the model's own most probable outputs. Only the hybrid decoder carries the official procedure over to the segment model.

Segment decoder. Among the most probable outputs, the segment decoder (SEG) keeps the one that best respects the invoice rules. In detail, it reranks the $K = 20$ best outputs of Proposition 1 with grammar, cardinality and column penalties. The grammar and cardinality penalties follow the verifiers of Sect. 3.1 with their calibrated costs, except that the cardinality penalty also restricts quantity. The column penalty charges a numeric field whose type differs from the preferred type of its column. Because the model scores only single-row segments, the reranker then adds multi-row field candidates and merges fields of the same type on consecutive rows of an item when the merged field scores higher than the separate fields. This decoder makes decisions about whole segments, and this is its weakness. When segment scores are means of unit scores, as for the tagger in Sect. 3.4, a weak neighbouring atom lowers the mean of a segment only slightly, so the segment can absorb it at little cost.

Hybrid decoder. The hybrid decoder (HYB) transfers the official decoder to the segment model. Like the official decoder, it first assigns a type to every small piece of text and then merges these pieces into fields. Unlike the official decoder, it derives these type assignments from the segment model's exact posterior and groups fields into items using the model's bands rather than item tags. Each atom α receives the most likely class under the posterior atom–type marginals,

$$\hat{t}(\alpha) = \arg \max_{t \in \mathcal{T} \cup \{\varnothing\}} \mu_\theta(\alpha, t), \tag{3}$$

where $\mu_\theta(\alpha, t) = \sum_{\sigma \ni \alpha} \mu_\theta(\sigma, t)$ is the probability that a type-t segment covers α, and $\mu_\theta(\alpha, \varnothing)$ is the probability that α is background. Because these marginals sum over all segmentations, a weak neighbour joins a field only if the field's type is its own most probable class, so absorbing it no longer pays. The decoder takes item bands from the maximum a posteriori (MAP) structure of the nested dynamic program of Proposition 1 and drops atoms outside them. Within each band, it traverses the rows from top to bottom and starts a new item at a row that repeats quantity or one of the six monetary types $\mathcal{T}_6$ within the current item. This item splitting is an invoice rule applied at decoding, with the same types as the cardinality penalty of SEG. Finally, the official threshold-based merging procedure joins same-type atoms within each page and item into fields [33].

3.4 Decoding a word tagger

FT is a word tagger, so the official decoder reads it directly. To compare all three decoders on one trained model, we convert its word scores into the unit, segment and band scores that the segment model would supply. In plain terms, every unit inherits the type scores of its word, a segment receives the mean of its units' scores, and a band is scoredaccording to how strongly the item tags indicate that an item starts in its first row and ends in its last. The conversion leaves the official decoder's decision for each word unchanged.

The tagger has independent sigmoid outputs. For a word w and a type t, let $z_{\mathrm{B}t}(w)$ and $z_{\mathrm{I}t}(w)$ be its logits for the first and for a later word of that type within an item, and let $z_{\mathrm{O}}(w)$ be its logit for carrying no line-item field. Conditioning these outputs on at most one of them being active gives the class scores of a unit u of w,

$$\begin{aligned} s(u, t) &= \log\bigl(e^{z_{\mathrm{B}t}(w)} + e^{z_{\mathrm{I}t}(w)}\bigr), \\ s(u, \varnothing) &= \log\bigl(1 + e^{z_{\mathrm{O}}(w)}\bigr). \end{aligned} \tag{4}$$

The two logits of a type enter only through their sum, because neither marks a field boundary. A

segment receives the mean of the scores of its units, normalised over the classes. On the L_{dev} words of one FT model, the most likely class under Eq. (4) equalled the official decision for every word.

Further, the tagger marks the words that start and end an item. For a row r, let $\iota(r)$ be the mean log-odds over its units that their word lies in some item, and let $\eta^{\text{B}}(r)$ and $\eta^{\text{E}}(r)$ be the largest item-start and item-end logits in the row. A band $b = [\ell_1, \ell_2]$ then receives

$$\begin{aligned}\psi(b) = \tfrac{1}{|b|}\textstyle\sum_{r\in b}\iota(r) + \eta^{\text{B}}(\ell_1) + \eta^{\text{E}}(\ell_2)\\ - \textstyle\sum_{\ell_1<r\le\ell_2}\log\bigl(1+e^{\eta^{\text{B}}(r)}\bigr)\\ - \textstyle\sum_{\ell_1\le r<\ell_2}\log\bigl(1+e^{\eta^{\text{E}}(r)}\bigr),\end{aligned} \tag{5}$$

the log-odds that its rows belong to an item that starts in the first row, ends in the last and neither starts nor ends in between. With these scores, SEG and HYB run on FT unchanged.

3.5 Supervised training

An annotation does not always determine a single output of the model. A field can extend over several rows, whereas a segment lies in one row, and an annotated boundary can lie where no candidate ends. We therefore split multi-row annotated fields into row slots and admit, for a slot without an exact segment, same-type candidates whose endpoints differ by at most one atom. Rows without any admissible segment keep their item but are otherwise unconstrained, and documents with incompatible item bands or overlapping slots are excluded from the loss. We minimise the resulting partial-label loss [4] over the outputs $\mathcal{Y}^*(d)$ consistent with the retained constraints from y^*,

$$\begin{aligned}\ell_{\text{sup}}(\theta; d) &= -\log \sum_{y\in\mathcal{Y}^*(d)} p_\theta(y \mid d)\\ &= \log Z_\theta(d) - \log Z^*_\theta(d),\end{aligned} \tag{6}$$

where Z^*_θ sums over $\mathcal{Y}^*(d)$ only. Fitting the adapter to L_{fit} for 25 epochs gives the supervised model (Sup) with parameters θ_0.

FT follows the official DocILE recipe for the tagger [33]. It differs from Sup in three respects. It updates all LayoutLMv3 weights and not only an adapter. It fits all k labelled documents. It also fits the official auxiliary targets for key-information localisation and extraction (KILE).

3.6 Verifier-reward training

Verifier-reward training (VRT) continues the training of Sup on an unlabelled documents U and rewards outputs that the verifiers accept. The three verifiers form one reward, which is evaluated on an output y of the model and before any decoder acts. Among the typed segments of y, n_{fail} and n_{frag} count grammar failures and fragments. In a selected band I, $n_t(I)$ counts the type-t segments. Over the items, n_{dec} and n_{viol} count arithmetic passes and violations. Further, count n_{tc} belongs to the type guard that the next paragraph introduces. With the costs of Sect. 3.1, the reward is

$$R(y; d) = R_{\text{gram}} + R_{\text{card}} + R_{\text{arith}}, \tag{7}$$

$$R_{\text{gram}} = -\lambda_{\text{gram}}(n_{\text{fail}} + n_{\text{frag}}) - \lambda_{\text{tc}}\, n_{\text{tc}}, \tag{8}$$

$$R_{\text{card}} = -\lambda_{\text{card}} \sum_{I\in y}\sum_{t\in\mathcal{T}_6} \max\bigl(0,\, n_t(I) - 1\bigr), \tag{9}$$

$$R_{\text{arith}} = \lambda_{\text{ar}}\,(n_{\text{dec}} - n_{\text{viol}}). \tag{10}$$

Penalties alone would let the model escape a grammar violation by giving the segment another type, for instance one without a grammar. A type guard therefore penalises a segment whose type differs from that of the overlapping segment in the fixed MAP output of the supervised model, and n_{tc} counts these segments. Two changes are exempt. A change is allowed if the text passes the grammar of the new type and fails that of the old one, and a swap between unit price and amount of the same variant, net or gross, is left to the arithmetic verifier. The guard has to outweigh whatever a single relabelling can gain from the other reward terms. We bound this gain and set λ_{tc} to the bound plus 1, so that an unsupported relabelling always lowers the reward.

The reward shifts probability towards consistent outputs but cannot identify the correct one. It consists almost entirely of penalties, so it also favours outputs with fewer fields. The empty output, which selects no band, has reward zero, and on a document where no item passes decisively no output scores higher. A KL anchor therefore limits how far the model departs from the supervised one. Its reference is the frozen supervised model, $\pi_{\text{ref}} = p_{\theta_0}$, and β sets its strength. The reward phase, the training on U, then maximises

$$J(\theta) = \mathbb{E}_{d\sim U}\Bigl[\,\mathbb{E}_{y\sim p_\theta(\cdot|d)} R(y; d)$$

$$- \beta \,\mathrm{KL}\big(p_\theta(\cdot \mid d) \,\big\|\, \pi_{\mathrm{ref}}(\cdot \mid d)\big)\Big]. \qquad (11)$$

Over all distributions on $\mathcal{Y}(d)$, this objective is maximised by the supervised distribution reweighted by the exponentiated reward, $q^*_\beta(y) \propto \pi_{\mathrm{ref}}(y)\, e^{R(y;d)/\beta}$. This is the known optimum of KL-regularised fine-tuning [14, 46] and of posterior regularisation with fixed multipliers [3, 6]. As β shrinks, the optimum concentrates on the outputs of maximal reward. On a document without a decisively passing item, these are the outputs without any violation, so an output that reproduces an annotated field failing a verifier would lose all probability. The anchored VRT runs use $\beta = 0.3$.

Most of the gradient of Eq. (11) is exact. Grammar failures and the guard attach to single typed segments, so with $r(\sigma, t)$ the reward of a typed segment from these two terms, their expectation is $\sum_{\sigma,t} \mu_\theta(\sigma, t)\, r(\sigma, t)$, and the KL term follows from Eq. (2). Although the cardinality excess couples the segments of a band, its expectation is also exact, at the cost of one masked row pass per monetary type, as Proposition 2 shows.

Proposition 2 (Expected cardinality excess) *Recompute the row sum Z_ℓ from the proof of Proposition 1 without type-t segments, and denote it by $Z_\ell^{\neg t}$. For every $t \in \mathcal{T}_6$,*

$$\mathbb{E}_{p_\theta}\Big[\sum_{I \in y} \max\big(0, n_t(I) - 1\big)\Big] = \sum_{\sigma} \mu_\theta(\sigma, t) - \sum_{b} \mu_\theta(b)\Big(1 - \prod_{\ell \in b} \frac{Z_\ell^{\neg t}}{Z_\ell}\Big).$$

The proof is given in Appendix A.2.

Fragments and arithmetic depend jointly on several segments. We estimate their gradient from $n_s = 8$ exact samples $y_1, \ldots, y_{n_s}$ with the score-function estimator [40]. With R_g the value of these reward terms on y_g, each sample is weighted by its advantage $\delta_g = R_g - \frac{1}{n_s - 1} \sum_{h \neq g} R_h$ over the mean of the others, which keeps the estimate unbiased [18].

Algorithm 1 summarises the training. Starting from θ_0, it alternates a reward step on one document of U with a supervised step on L_{fit}, both at a tenth of the supervised model's learning rate. It returns the final parameters without model selection, and the evaluations on L_{dev} serve only for monitoring. U contributes only OCR text.

Algorithm 1 Verifier-reward training (VRT)

Require: $L_{\mathrm{fit}}, L_{\mathrm{dev}}, U$; reward weights λ; β, n_s, $\tau_{\max}$

fit θ_0 by Eq. (6) on L_{fit}, 25 epochs
$\pi_{\mathrm{ref}} \leftarrow p_{\theta_0}$; $\theta \leftarrow \theta_0$ ▷ π_{ref} stays frozen
evaluation on L_{dev} ▷ $\tau = 0$
for $\tau = 1, \ldots, \tau_{\max}$ **do** ▷ $\tau_{\max} = 2\,000$
  $d \leftarrow$ next document of U
  compute $\mu_\theta, \log Z_\theta$ using Proposition 1
  sample $y_{1..n_s} \sim p_\theta(\cdot \mid d)$
  $\mathcal{L} \leftarrow -\sum_{\sigma,t} \mu_\theta(\sigma, t)\, r(\sigma, t) - \mathbb{E}[R_{\mathrm{card}}]$
    $- \frac{1}{n_s} \sum_g \delta_g F_\theta(y_g; d) + \beta \,\mathrm{KL}$
  gradient step on $\mathcal{L}$ ▷ δ_g held fixed
  gradient step on $\ell_{\mathrm{sup}}(\theta; d')$, d' next in L_{fit}
  at $\tau \in \{1\,000, 2\,000\}$: evaluation on L_{dev}
end for
return final θ ▷ no model selection

3.7 Filtered self-training

Self-training lets a model learn from its own predictions on unlabelled documents, the pseudo-labels, and variants differ in how they select them. Rule-filtered self-training (ST) starts from Sup, decodes U with SEG and keeps the predicted fields that pass the verifiers. It thus shares the frozen encoder features and the model of VRT and uses the verifiers as a filter where VRT uses them as a reward. Confidence-based self-training of the fully fine-tuned model (FT+ST) extends the comparison beyond the adapters and uses no verifiers. A FT teacher labels a second unlabelled pool U_{FT} with the official decoder, and a fresh student is trained on the labelled documents and on the pseudo-labelled documents of highest confidence.

3.8 Decoder audit

The decoder audit asks whether a training gain survives a change of decoder. It takes a trained model as it is, with its candidate space and its scores, and decodes it twice, each time with a different decoder. Whatever differs between the two results is then due to the decoder alone. A decoder that is used inside training, for instance to write pseudo-labels, is not exchanged.

We measure a training method by its training gain, the F1 of the trained model θ minus the F1 of the model θ' it started from, both under the same decoder. The training-by-decoder interaction is the gain under the second decoder minus the

gain under the first. With $F_1(\theta, D)$ the F1 of a model under decoder D, it reads

$$\Delta_D = [F_1(\theta, D_2) - F_1(\theta', D_2)] - [F_1(\theta, D_1) - F_1(\theta', D_1)]. \quad (12)$$

A negative value means that the gain is smaller under the second decoder than under the first. The results report it with reversed sign as the gain lost.

4 Experimental setup

4.1 Datasets and metrics

DocILE [33] provides 5 180 annotated training invoices and 500 official validation documents, the validation split. As a test split, we set aside 522 training invoices from layout clusters that do not occur among the remaining ones, which leaves a training pool of 4 658 documents. We report all main DocILE results on the validation split and repeat the system comparison on the test split, whose layouts occur in neither the labelled nor the unlabelled documents. We run every experiment three times, each with its own draw of labelled documents. In each label draw, the k documents of a budget come from k different layout clusters of the training pool, and the budgets $k \in \{50, 100, 250\}$ are nested. The adapters draw on an unlabelled pool U of 1 000 documents, and FT+ST on a layout-diverse pool U_{FT} of the same size that is disjoint from all labelled and evaluation documents. All intervals are 95 % bootstrap intervals over the label draws and the documents. Differences between two systems or decoders are paired by evaluating the same documents in each resample.

We score DocILE outputs with the official matcher [33, 34]. A predicted field matches an annotated one only if both agree in type and page and cover exactly the same characters, so an amount that misses one digit is both a false positive and a false negative. Predicted and annotated items are paired one to one by a maximum-weight assignment, so one misplaced field can cost its whole item. LIR-F1 is the micro-F1 over the matched fields of all documents.

CORD [23] adds receipts as a second document type. We use its 800 training receipts for three label draws at $k = 25$ and 50, the 665 receipts that no draw labels as the unlabelled pool, and the 100 receipts of the CORD validation split for evaluation. All receipt models use the frozen RoBERTa-base encoder and the verifiers and reward weights calibrated on DocILE. They receive no rule prior, since the rule baseline that supplies it relies on the header row and columns of invoice tables. As on DocILE, all models use the OCR words provided by the dataset. Field-F1 matches predicted and annotated items one to one. A field belonging to one of the four line-item types is counted as correct if its normalised value equals that of the field of the same type in the paired annotated item. Since this score compares field values rather than bounding boxes, receipts are decoded with a receipt variant of the hybrid decoder (HYB-C). This variant retains the atom-level decisions and item splitting of HYB but replaces the official field-merging procedure with one that joins consecutive same-type atoms within a row.

Field-level scores can hide incomplete items, and group-level F1 on receipts lies well below entity F1 [15]. Beside LIR-F1 and Field-F1, we therefore report exact-item F1, which counts an item only if all its fields match and no extra field is predicted.

4.2 Systems and training

Table 1 lists the compared systems. On each encoder, the supervised model (Sup), verifier-reward training (VRT) and rule-filtered self-training (ST) use the same adapter, candidate space and rule prior, and VRT and ST start from the Sup model. We report both encoders so that a change of representation is not credited to the training method.

The Rules baseline needs no training. It finds the header row of a table with a lexicon of header words, which joins a fixed multilingual list with the words that stand above labelled columns in L_{fit}, and it derives one column per field type from these words. Below the header, a row starts a new item if a numeric column holds a word that passes the grammar of its type, any other row continues the previous item, and a stop word such as a total ends the table. Rules with templates first looks for a labelled document with a similar layout and takes the columns from it, and it falls back on Rules if none is similar enough.

To test whether the rules matter, we compare ST using the LayoutLMv3 encoder with three variants that differ only in how pseudo-labels are

Table 1: Compared systems. A check mark under a decoder marks an evaluation with it on DocILE, and the last column marks an evaluation on CORD, where HYB-C replaces HYB. The adapters use both frozen encoders, and the fully fine-tuned model (FT) and confidence-based self-training of the fully fine-tuned model (FT+ST) fine-tune LayoutLMv3. All systems are also evaluated on the DocILE test split.

						Evaluated with			
System	Trained	Labelled	Unlabelled	Rule prior	Verifiers	Official	SEG	HYB	CORD
RULES	none	L_{fit}	—	—	—	—	—	—	—
SUP	adapter	L_{fit}	—	✓	—	—	✓	✓	✓
VRT	adapter	L_{fit}	U	✓	reward	—	✓	✓	✓
ST	adapter	L_{fit}	U	✓	filter	—	✓	✓	—
FT	all	all k	—	—	—	✓	✓	✓	—
FT+ST	all	all k	U_{FT}	—	—	✓	✓	✓	—

selected. The plain variant keeps all pseudo-labels, the confidence variant keeps those with the highest confidence scores, and the random variant selects them at random. The latter two retain the same number of pseudo-labels as the rule filter.

For FT+ST, the FT teacher labels the documents of U_{FT}, and the 500 with the highest mean field confidence are kept. A fresh student starts from the same public checkpoint as the teacher and is trained with the same settings on the labelled and these pseudo-labelled documents. As an ablation at $k = 250$, a masked-label student is trained in the same way but excludes pseudo-labels from its loss. This tests whether the gain comes from learning from those pseudo-labels.

The supervised model is trained for 25 epochs with AdamW at a learning rate of 10^{-3}, weight decay 0.01, gradient clipping at 5, dropout 0.1, one document per update, 5 % warm-up and cosine decay. VRT and ST alternate 2 000 unlabelled with 2 000 supervised steps at 10^{-4} under the same schedule. FT starts from the public LayoutLMv3 base checkpoint [12] and uses AdamW at 2×10^{-5} with weight decay 0.001 and batch size 16. This checkpoint was pre-trained on IIT-CDIP, which the DocILE benchmark rules exclude [5], so all models on this encoder lie outside the benchmark.

5 Results

5.1 Decoders on the word tagger

We first compare the decoders themselves, before any training method. FT is a word tagger, so the official DocILE decoder reads it directly, and SEG and HYB read it through the score conversion of Sect. 3.4. The model stays fixed, so every difference between the rows of Table 2 comes from the decoder. At $k = 250$, SEG scores about six points below the official decoder. The column Δ gives this loss as a paired difference, which subtracts the two scores on the same trained models and the same documents in every bootstrap resample. Whatever makes a document hard affects both scores alike and cancels, so the interval of the difference is narrower than the intervals of the two scores. It excludes zero, so the loss is established and not a chance result of the sampled documents and label draws. HYB recovers most of the loss without retraining. Its Δ is under one point with an interval that includes zero, so no difference to the official decoder is established. On this model, HYB thus reaches the level of the official decoder. The loss of SEG is larger in recall than in precision. This matches the weakness of Sect. 3.3, where a segment absorbs a weak neighbouring atom.

Table 2: FT under the three decoders on the DocILE validation split at $k = 250$, mean over three label draws. Δ is the paired difference in LIR-F1 to the official decoder in F1 points. The trained models are the same in all rows. Bold marks the best value per column.

Decoder	F1 [95 % CI]	Prec.	Rec.	Δ to official
Official	**0.480** [0.444, 0.517]	**0.457**	0.507	—
SEG	0.419 [0.388, 0.451]	0.421	0.417	−6.2 [−8.5, −3.9]
HYB	0.472 [0.440, 0.505]	0.430	**0.524**	−0.9 [−2.4, +0.6]

Table 3: The two steps from SEG to HYB on the DocILE validation split at $k = 250$, mean over three label draws. The two step columns give the paired change in F1 points when each step is added individually to SEG. The last column reports the number of additional items created by item splitting across the 500 documents.

Model	SEG	Field step	Item splitting	HYB	Items
RoBERTa-base					
SUP	0.307	+0.2 [−0.6, +1.0]	+6.2 [+2.8, +10.6]	0.379	809
VRT	0.372	+0.3 [−0.4, +1.0]	+1.2 [+0.3, +2.7]	0.388	148
LayoutLMv3					
SUP	0.359	−0.3 [−1.5, +0.7]	+5.7 [+2.5, +9.8]	0.418	605
VRT	0.410	−0.3 [−1.9, +0.9]	+1.2 [+0.0, +3.1]	0.419	86
FT	0.419	+5.2 [+3.6, +6.8]	+0.1 [+0.0, +0.4]	0.472	43

5.2 From segment to hybrid decoder

HYB differs from SEG in two steps. The field step replaces the decisions on whole segments by decisions per atom with the official merging. Item-band splitting then starts a new item at a row that repeats a type which an item may hold only once. Table 3 reports the paired change from adding each step individually to SEG while keeping the trained models fixed. For FT, the field step brings the whole improvement, and item-band splitting changes almost nothing. Under SEG, the tagger thus had the items right and the field boundaries wrong. Its training includes tags for the first and the last word of an item, so its items need no splitting. For the adapters, it is the other way round. The field step changes little, and every interval of its change includes zero. Item-band splitting adds several points to SUP and about one point to VRT. The problem of the adapters under SEG was thus the grouping of rows into items. The two steps do not depend on each other, since each brings about the same change after the other step as alone. FT differs from the adapters in more than training, so the table is read by row, not across models.

5.3 Decoder audit of reward training

We now apply the audit to VRT. Its training gain is the score of VRT minus that of SUP under the same decoder, and Table 4 gives both scores under SEG and under HYB for the same trained models. On DocILE at $k = 250$, the gain is more than five points on both encoders under SEG, with paired intervals above zero. Under HYB, both intervals include zero. HYB improves both models' scores, but benefits SUP much more than VRT, reducing the lead of VRT to less than one F1 point. Table 3 locates this difference in item-band splitting. Under SEG, the splitting rule separates five to seven times as many items for SUP as for VRT. The reward of VRT penalises an item that repeats such a type, so VRT has learned to form the items that the rule produces without training.

The same holds at the smaller budgets. The last column of Table 4 gives the gain lost, the gain under SEG minus the gain under HYB. It is positive at all three budgets on both encoders, and its paired interval excludes zero in each case. Under HYB, no VRT gain has an interval above zero at any budget. The loss of the gain is thus not tied to one label budget.

We repeat the audit on CORD receipts as a second document type, with HYB-C in place of HYB and Field-F1 as the score. Under SEG, VRT gains about five points over SUP at $k = 25$ and 50 in Table 4, with paired intervals above zero. Under HYB-C the gain is under one point, and both intervals include zero. The gain lost is positive at both budgets, and its paired interval excludes zero. The decoding effect is thus not specific to invoices.

5.4 Label-budget comparison

With HYB as the decoder, Table 5 compares all systems across the label budgets. ST improves on SUP at $k = 50$ and 100 on both encoders, and its paired gains have intervals above zero. At $k = 250$ the two differ by about one point or less, and the intervals include zero. The audit gives the opposite picture to VRT. On both encoders, the gain of ST is larger under HYB than under SEG at every budget, so the decoder exchange removes none of it. ST thus learns something that HYB does not supply, but only while labels are scarce.

The rule filter is what distinguishes ST, so we test whether its gain needs the rules. Table 6 compares ST with three pseudo-label selection variants using LayoutLMv3 at the two budgets where ST improves on the supervised baseline. The variants differ only in which pseudo-labels they retain. Keeping all pseudo-labels improves on SUP only slightly, so selection matters. Confidence-based selection recovers most of ST's gain, with paired confidence intervals entirely above zero at

Table 4: Decoder audit of VRT on the DocILE validation split and on CORD, mean over three label draws. The VRT gain is the paired difference of VRT to SUP under one decoder. The gain lost is the gain under SEG minus the gain under the hybrid decoder, which is the training-by-decoder interaction of Eq. (12) with reversed sign. Both are in F1 points. DocILE rows use LIR-F1, CORD rows HYB-C and Field-F1.

			SEG			HYB/ HYB-C			
Dataset	Encoder	k	SUP	VRT	VRT gain	SUP	VRT	VRT gain	Gain lost
DocILE	RoBERTa-base	50	0.163	0.194	+3.1 [+1.5, +4.9]	0.226	0.216	−0.9 [−3.5, +1.1]	4.1 [1.8, 6.5]
		100	0.217	0.256	+4.0 [+1.9, +6.1]	0.284	0.270	−1.4 [−4.2, +1.0]	5.4 [2.6, 9.1]
		250	0.307	0.372	+6.6 [+3.1, +11.0]	0.379	0.388	+0.8 [−0.5, +2.1]	5.7 [2.1, 11.1]
DocILE	LayoutLMv3	50	0.199	0.248	+4.9 [+1.8, +7.6]	0.266	0.266	−0.1 [−1.3, +1.2]	4.9 [1.9, 7.6]
		100	0.284	0.332	+4.8 [+0.3, +9.7]	0.329	0.342	+1.3 [−0.4, +3.1]	3.5 [0.3, 7.0]
		250	0.359	0.410	+5.2 [+2.4, +8.1]	0.418	0.419	+0.1 [−1.0, +1.3]	5.1 [2.4, 8.2]
CORD	RoBERTa-base	25	0.591	0.642	+5.1 [+1.1, +9.2]	0.660	0.658	−0.2 [−2.3, +1.9]	5.4 [1.3, 10.0]
		50	0.708	0.757	+4.9 [+0.8, +9.3]	0.745	0.754	+0.9 [−1.9, +4.6]	4.0 [0.9, 7.7]

Table 5: LIR-F1 of all systems on the DocILE validation split across label budgets, mean over three label draws with 95 % intervals. The adapters are decoded with HYB, and FT and FT+ST with the official decoder. Bold marks the best and underlining the second-best value per column.

Encoder	Model	$k = 50$	$k = 100$	$k = 250$
—	RULES	0.292 [0.258, 0.326]	0.289 [0.259, 0.322]	0.268 [0.239, 0.298]
RoBERTa-base	SUP	0.226 [0.169, 0.285]	0.284 [0.253, 0.319]	0.379 [0.344, 0.416]
	VRT	0.216 [0.137, 0.291]	0.270 [0.231, 0.308]	0.388 [0.349, 0.428]
	ST	0.266 [0.220, 0.318]	0.324 [0.291, 0.360]	0.391 [0.360, 0.423]
LayoutLMv3	SUP	0.266 [0.208, 0.335]	0.329 [0.300, 0.361]	0.418 [0.381, 0.456]
	VRT	0.266 [0.211, 0.331]	0.342 [0.306, 0.379]	0.419 [0.380, 0.459]
	ST	<u>0.313</u> [0.258, 0.368]	0.364 [0.332, 0.399]	0.422 [0.391, 0.455]
	FT	0.289 [0.219, 0.345]	<u>0.377</u> [0.326, 0.424]	<u>0.480</u> [0.444, 0.517]
	FT+ST	**0.317** [0.245, 0.379]	**0.419** [0.379, 0.458]	**0.507** [0.474, 0.540]

both budgets, while randomly selecting the same number of pseudo-labels yields smaller gains. ST exceeds the confidence variant by less than one point, and both paired intervals include zero. We attribute a gain to the rule filter only if ST exceeds the strongest variant by at least 1.5 points with a paired interval above zero. This is not the case, so the gain comes from selecting pseudo-labels and is not established as an effect of the rules.

5.5 Self-training without rules

Selection by confidence was enough for the adapters, so we apply self-training without any rule to the fully fine-tuned model. Table 7 compares FT+ST with its FT teacher under all three decoders. FT+ST gains about three to five points at all three budgets and under every decoder, and all paired intervals exclude zero. It raises precision and recall together. The gain has about the same size under every decoder, so it is learned and not a decoding effect. We count it as established if it reaches two points under the official decoder and is positive in at least two of the three draws, which holds at all three budgets.

Table 6: Pseudo-label selection on LayoutLMv3 under HYB on the DocILE validation split, mean over three label draws. All rows start from the same SUP model and differ only in the pseudo-labels they train on. The lead of ST is its paired difference to the row in F1 points.

	$k = 50$		$k = 100$	
Pseudo-labels	F1	ST lead	F1	ST lead
SUP, none	0.266	+4.6 [+2.6, +6.7]	0.329	+3.5 [+2.2, +4.8]
Plain	0.276	+3.7 [+2.2, +5.4]	0.342	+2.2 [+1.2, +3.2]
Random	0.292	+2.1 [+0.8, +3.5]	0.354	+1.0 [−0.0, +1.9]
Confidence	0.303	+0.9 [−0.4, +2.2]	0.360	+0.4 [−0.4, +1.2]
ST, rule filter	0.313	—	0.364	—

Table 7: FT+ST against its FT teacher on the DocILE validation split, mean over three label draws. The gain is the paired difference in F1 points.

		FT			FT+ST			
k	Decoder	F1	Prec.	Rec.	F1	Prec.	Rec.	Gain
50	Official	0.289	0.309	0.274	0.317	0.334	0.304	+2.8 [+1.3, +4.2]
	SEG	0.270	0.299	0.249	0.296	0.322	0.276	+2.6 [+1.3, +4.0]
	HYB	0.328	0.313	0.347	0.362	0.352	0.377	+3.5 [+2.3, +4.6]
100	Official	0.377	0.380	0.376	0.419	0.422	0.420	+4.3 [+2.4, +6.2]
	SEG	0.323	0.344	0.306	0.373	0.390	0.359	+5.0 [+3.4, +6.5]
	HYB	0.387	0.363	0.418	0.433	0.415	0.456	+4.6 [+3.0, +6.3]
250	Official	0.480	0.457	0.507	0.507	0.482	0.534	+2.7 [+1.0, +4.6]
	SEG	0.419	0.421	0.417	0.451	0.446	0.456	+3.3 [+1.9, +4.6]
	HYB	0.472	0.430	0.524	0.504	0.466	0.548	+3.2 [+2.0, +4.3]

The FT+ST student sees more documents than its teacher, so its gain could come from the training mixture and not the pseudo-labels. The masked-label student tests this at $k = 250$. It does not improve on the teacher, and FT+ST exceeds it by more than two points, with paired intervals above zero under the official decoder and under HYB. Thus, the gain comes from the pseudo-labels.

Table 5 finally places all systems side by side. At $k = 50$, the training-free RULES baseline has a higher mean F1 than SUP and VRT on both encoders. From $k = 100$ on, FT under the official decoder scores above all adapters. FT+ST has the highest mean F1 of all systems at $k = 100$ and 250, although it uses no invoice rules. As in Table 3, the comparison with FT and FT+ST is one of systems and not of training methods.

All scores so far count single fields, and exact-item F1 shows how many items are complete. Few are. At $k = 250$ under HYB, the exact-item F1 of every system on LayoutLMv3 is about a quarter of its LIR-F1. The gain of FT+ST over FT does not carry over, since its paired exact-item difference has an interval that includes zero. Nor does the decoder exchange. On DocILE, HYB raises the LIR-F1 of every adapter but not its exact-item F1, as Fig. 3a shows. For SUP, it predicts more items, but hardly more of them are complete. On CORD, HYB-C raises the Field-F1 of SUP more than its exact-item F1 at both budgets in Fig. 3b. In exact-item F1, the paired VRT gain has intervals that include zero under the hybrid decoder on both datasets. The gains reported above are thus gains in fields and not in complete items.

5.6 Unseen layouts

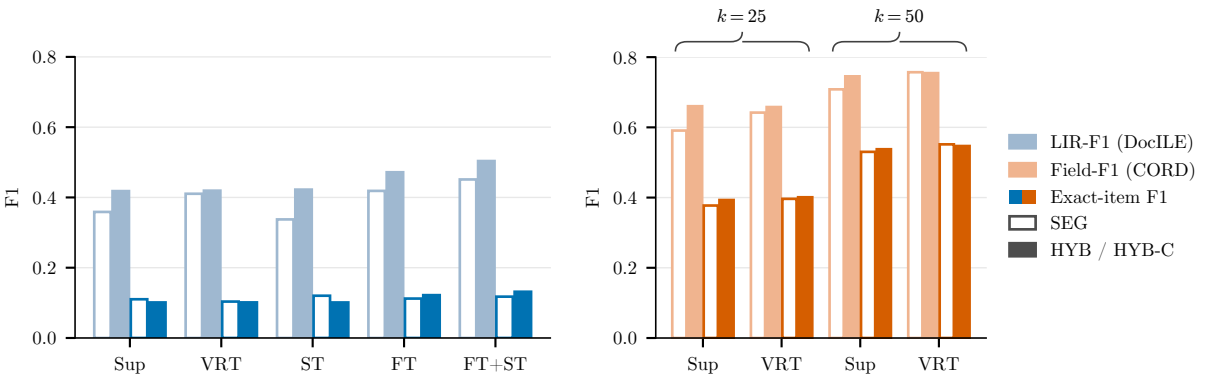
The test split repeats the comparison with the same trained models on invoices whose layouts none of them saw in training. The audit gives the same result. In Table 8, VRT gains five to eight points over SUP under SEG, whereas under HYB all six gains have intervals that include zero. The gain lost is positive for both encoders at all three budgets, and every paired interval excludes zero. In Table 9, ST again improves on SUP at $k = 50$ and 100 and not at $k = 250$. FT+ST again scores highest at $k = 100$ and 250, and at $k = 50$ RULES scores highest of all systems. The main findings thus do not depend on layouts seen in training.

Two findings differ from the validation split. At $k = 50$ and 100, ST exceeds the confidence variant with paired intervals above zero, and at $k = 50$ the lead of about three points is large enough for the rule of Sect. 5.4 to attribute a gain to the rule filter. On unseen layouts, the rules thus add something that confidence alone does not supply. At $k = 250$, the gain of FT+ST over FT holds under HYB, but under the official decoder it is under one point with an interval that includes zero, so it is not established there.

6 Discussion

6.1 Role of the decoder

A reported score reflects the combined performance of a trained model and a decoder together, and the audit separated the two in three cases. The gain of VRT was lost under the hybrid decoder on invoices, on unseen layouts and on receipts. The gain of ST persisted, and the gain of FT+ST remained



(a) DocILE, LayoutLMv3, $k = 250$ (b) CORD, RoBERTa-base

Fig. 3: Field-level against exact-item F1, mean over three label draws. Open bars show SEG and filled bars the hybrid decoder, HYB on DocILE and HYB-C on CORD.

Table 8: Decoder audit of VRT on the DocILE test split, 522 invoices whose layouts were not seen in training, mean over three label draws. Columns as in Table 4.

		SEG			HYB			
Encoder	k	Sup	VRT	VRT gain	Sup	VRT	VRT gain	Gain lost
RoBERTa-base	50	0.110	0.185	+7.5 [+3.3, +11.1]	0.202	0.206	+0.4 [−4.7, +4.1]	7.1 [5.1, 8.8]
	100	0.159	0.212	+5.3 [+3.3, +7.6]	0.253	0.239	−1.4 [−6.4, +2.7]	6.6 [3.5, 10.4]
	250	0.249	0.307	+5.8 [+1.8, +9.7]	0.330	0.330	+0.0 [−3.4, +2.7]	5.8 [0.2, 10.9]
LayoutLMv3	50	0.163	0.231	+6.7 [+4.0, +10.4]	0.228	0.238	+1.0 [−0.7, +2.9]	5.7 [4.0, 7.7]
	100	0.240	0.306	+6.6 [+1.3, +11.2]	0.313	0.323	+0.9 [−1.2, +3.2]	5.6 [1.2, 11.2]
	250	0.311	0.374	+6.2 [−0.1, +11.3]	0.410	0.392	−1.8 [−3.9, +0.3]	8.0 [2.4, 14.1]

approximately the same under every decoder. The audit thus does not eliminate the gains of every training method. It costs one further evaluation of models that are already trained. We therefore recommend it before a training gain is attributed to learning, for every extractor whose decoder has no trainable parameters.

The audit is only as informative as its second decoder. HYB reached the level of the official decoder on FT, a model that both can read, so the audit does not rest on a weak or unusually tuned decoder. This check covers the tagger only, since the adapters have no established decoder to compare with. SEG lost points in field boundaries on the tagger and in the grouping of rows on the adapters, so a decoder validated on one model family need not suit another. A study that replaces the word tagger with a model the official decoder cannot read, as we did for rewards on whole outputs, must build and validate its own decoder.

6.2 Role of rules

The invoice rules entered this study in three roles, as a reward in training, as a rule of the decoder and as a filter for pseudo-labels. The reward penalises an item that repeats a type which an item may hold only once, and VRT had learned to avoid such items. Item-band splitting applies the same rule at decoding and removes such items for any model. This is why the gain of VRT was lost under HYB. Before training with a rule that a program can check, one should therefore test whether the same rule can be applied at decoding. Training then pays only for what decoding cannot supply.

As a filter, the rules select the pseudo-labels of ST. Selection by confidence did nearly as well on the validation split, and FT+ST gained without any invoice rule. Only on unseen layouts at the smallest budget was a gain attributed to the rule filter. A claim that a gain is specific to invoice rules

Table 9: LIR-F1 of all systems on the DocILE test split per label budget (mean over three label draws, 95 % intervals), using the decoders of Table 5. Bold: best, underlined: second-best per column.

Encoder	Model	$k = 50$	$k = 100$	$k = 250$
—	RULES	**0.368** [0.295, 0.431]	0.370 [0.296, 0.433]	0.317 [0.268, 0.377]
RoBERTa-base	SUP	0.202 [0.159, 0.236]	0.253 [0.234, 0.273]	0.330 [0.298, 0.358]
	VRT	0.206 [0.108, 0.269]	0.239 [0.176, 0.292]	0.330 [0.268, 0.379]
	ST	0.243 [0.187, 0.284]	0.294 [0.273, 0.319]	0.340 [0.306, 0.370]
LayoutLMv3	SUP	0.228 [0.163, 0.276]	0.313 [0.288, 0.343]	0.410 [0.385, 0.436]
	VRT	0.238 [0.162, 0.298]	0.323 [0.295, 0.347]	0.392 [0.364, 0.418]
	ST	0.289 [0.236, 0.336]	<u>0.371</u> [0.344, 0.402]	0.412 [0.381, 0.445]
	ST, confidence	0.260 [0.225, 0.294]	0.356 [0.327, 0.392]	0.409 [0.383, 0.437]
	FT	0.307 [0.203, 0.386]	0.358 [0.257, 0.438]	<u>0.473</u> [0.446, 0.501]
	FT+ST	<u>0.325</u> [0.217, 0.410]	**0.429** [0.387, 0.467]	**0.479** [0.429, 0.540]

therefore has to exceed a matched confidence filter and not only SUP.

In none of the three roles do the rules guarantee a complete output. An output that passes the verifiers can still be wrong. A rule cannot recover a word that OCR missed. Arithmetic checks can pass despite incorrect field types: quantity multiplied by unit price equals the amount whether the unit price and amount are labelled as net or gross. These checks can also pass when an entire field is missing, e.g., an item's description. The arithmetic verifier can moreover only check an item whose fields the benchmark annotates and the model recovers. We kept the benchmark annotations and did not relabel the data to a schema as in [10]. A rule thus judges what an output contains and not what it lacks. None of the gains reached complete items, so evaluations should report exact-item F1 alongside field-level scores.

6.3 Limitations

The experimental design covers two encoder architectures, several label budgets, unseen invoice layouts and receipts. However, the results are based on three label draws and one training seed per system, while the hybrid decoder was selected using the development subset of one label draw. The evaluated models rely on the OCR provided with their respective datasets.

Our decoder audit isolates the effect of changing the decoder rather than the contribution of each training component. Specifically, the adapters use a rule prior based on training-pool annotations. Comparisons with fully fine-tuned models involve more than a change in training method.

Future work could extend the audit to other extraction architectures and examine whether decoding can improve item-level consistency. Further evaluation across invoice collections, OCR settings, and training seeds would help establish the generality of the findings.

7 Conclusion

We asked whether a gain from further training with invoice rules is learned by the model or could equally come from a better decoder. For verifier-reward training, it came from the decoder. Under the segment decoder this method gained clearly over the supervised model, but the gain was lost once a hybrid decoder at the level of the official DocILE decoder took its place. This happened on invoices, on unseen layouts, and on receipts alike. The reason is that the reward had taught the model to separate items, which a rule of the hybrid decoder does without any training. However, not all the gains were lost this way. Rule-filtered self-training survived the decoder exchange, although selection by confidence matched most of it. Confidence-based self-training of a fully fine-tuned model even scored highest without any invoice rule. Yet, none of these gains reached complete items. The decoder audit drew these distinctions at the cost of one further evaluation of already trained models. We therefore recommend it before any training gain is credited to learning.

Declarations

Funding. No funding was received for this work.

Competing interests. The authors declare no competing interests.

Ethics approval and consent. Not applicable to the reported benchmark experiments.

Data availability. DocILE [33] and CORD [23] are publicly available under their providers' terms. The code release will link these datasets; we do not redistribute DocILE documents or annotations.

Code and model availability. We will release the code, configurations, build scripts and all experimental result records in a public Git repository; its URL and any model artefacts will follow separately.

Author contributions. Linda-Sophie Schneider: Conceptualization, Methodology, Software, Formal analysis, Visualization, Writing—original draft, Writing—review and editing. Thomas Gorges: Formal analysis, Writing—review and editing. Julian Thomas: Conceptualization, Writing—review and editing. Mathias Zinnen: Writing—review and editing. Vincent Christlein: Supervision, Writing—review and editing.



Linda-Sophie Schneider is a PhD candidate at the Pattern Recognition Lab, Friedrich-Alexander University of Erlangen-Nürnberg (FAU). Her research focuses on task-driven 3D imaging, using machine learning and differentiable programming to optimise CT acquisition and reconstruction for downstream analysis, particularly of carbonised Herculaneum papyri.



Thomas Gorges received his B.Sc. in computer science from the Technische Hochschule Nürnberg Georg Simon Ohm in 2019 and his M.Sc. with distinction from the Friedrich-Alexander-Universität Erlangen-Nürnberg in 2022. From 2022 to 2024, he was a Scientific Researcher at Fraunhofer IIS. Since 2024, he has been a Scientific Researcher at the Pattern Recognition Lab at FAU, where he is pursuing a PhD. His research interests include data scarcity mitigation and document analysis.



Julian Thomas is a PhD candidate in Applied Cryptography at Friedrich-Alexander University of Erlangen-Nürnberg (FAU). He received his M.Sc. in Computer Science from FAU. His research spans theoretical cryptography, particularly formal modeling and security proofs, and applied cryptography, focusing on modern secure messaging protocols and the intersection of cryptography and machine learning.



Mathias Zinnen studied philosophy and computer science in Berlin and Erlangen and is a PhD candidate at the Pattern Recognition Lab, Friedrich-Alexander University of Erlangen-Nürnberg (FAU). His research focuses on computer vision for the arts, in particular object detection, pose estimation and multimodal methods, including the visual representation of smell. More recently, his work extends to document analysis with lightweight vision-language models.



Vincent Christlein heads the Computer Vision group at the Pattern Recognition Lab, Friedrich-Alexander University of Erlangen-Nürnberg (FAU). He received his diploma and Dr.-Ing. degrees from FAU in 2012 and 2018, respectively. His primary research focus is on document analysis, including writer identification and handwriting imitation, as well as environmental projects such as glacier front segmentation and bird detection. In the field of document analysis, his work has earned recognition through several international competition wins and multiple awards, including the prestigious IAPR/ICDAR Young Investigator Award 2024.

Appendix A Proofs

A.1 Proof of Proposition 1

Proof of Proposition 1 The algorithm processes rows sequentially; each row needs only the total weight of the preceding rows. Within row ℓ, a left-to-right semi-Markov pass sums the weight exp(score) of all ways to cover the row by background atoms and typed segments; we call this row sum Z_ℓ. Let V_ℓ be the summed weight of all outputs restricted to rows $1, \dots, \ell$, with $V_0 = 1$, and Ω_ℓ the score of row ℓ outside every band. Row ℓ either lies outside every band or ends a band $b = [\ell_1, \ell]$ that starts at row ℓ_1, and the row covers of this band factorise independently, so

$$V_\ell = V_{\ell-1}\, e^{\Omega_\ell} + \sum_{b=[\ell_1,\ell]} V_{\ell_1-1}\, e^{\psi_\theta(b)} \prod_{\ell'=\ell_1}^{\ell} Z_{\ell'},$$

and $V_{n_r} = Z_\theta(d)$, because a product of sums is the sum of all products. The passes visit each atom once and each segment once per type; at most 8 bands end in a row. The same passes run backwards give all marginals, as in the forward–backward algorithm. A segment's marginal is the probability ν_ℓ that its row ℓ lies in a selected band times its marginal within the row. Retracing the passes at random draws an exact sample,

and keeping the K largest terms instead of their sum gives the K best outputs. Finally, $\log p_\theta(y) - \log p_{\theta'}(y)$ is a sum of factor score differences minus the difference $\log Z_\theta - \log Z_{\theta'}$; averaging it under p_θ weights each factor by $\mu_\theta(a)$, which gives Eq. (2). □

A.2 Proof of Proposition 2

Proof of Proposition 2 The idea is to count all type-t segments and then subtract one for every item that has at least one. Every selected segment lies in exactly one selected band, so the count has expectation $\sum_\sigma \mu_\theta(\sigma, t)$. The model selects a band b with probability $\mu_\theta(b)$ and then covers its rows independently. Row ℓ holds no type-t segment with probability $Z_\ell^{\neg t}/Z_\ell$, the weight share of its covers without one; multiplying over the rows of b gives the chance that the item has none. □

References

[1] Bouthillier X, Delaunay P, Bronzi M, et al (2021) Accounting for variance in machine learning benchmarks. In: Proceedings of Machine Learning and Systems (MLSys), pp 747–769

[2] Cao P, Wang Y, Zhang Q, et al (2023) GenKIE: Robust generative multimodal document key information extraction. In: Findings of the Association for Computational Linguistics: EMNLP 2023, pp 14702–14713, https://doi.org/10.18653/v1/2023.findings-emnlp.979

[3] Chang MW, Ratinov L, Roth D (2007) Guiding semi-supervision with constraint-driven learning. In: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL), pp 280–287

[4] Cour T, Sapp B, Taskar B (2011) Learning from partial labels. Journal of Machine Learning Research 12(42):1501–1536

[5] DocILE organisers (2023) DocILE rules and prize eligibility. URL https://docile.rossum.ai/static/docile_rules_and_prize_eligibility.pdf, accessed 23 October 2026

[6] Ganchev K, Graça J, Gillenwater J, et al (2010) Posterior regularization for structured latent variable models. Journal of Machine Learning Research 11:2001–2049

[7] Guo D, Yang D, Zhang H, et al (2025) DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. Nature 645(8081):633–638. https://doi.org/10.1038/s41586-025-09422-z

[8] He J, Wang L, Hu Y, et al (2023) ICL-D3IE: In-context learning with diverse demonstrations updating for document information extraction. In: IEEE/CVF International Conference on Computer Vision (ICCV), pp 19485–19494

[9] Hemmer A, Coustaty M, Bartolo N, et al (2023) Lazy-k decoding: Constrained decoding for information extraction. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp 6727–6736, https://doi.org/10.18653/v1/2023.emnlp-main.416

[10] Hemmer A, Coustaty M, Bartolo N, et al (2025) Neurosymbolic information extraction from transactional documents. International Journal on Document Analysis and Recognition (IJDAR) 28(3):475–485. https://doi.org/10.1007/s10032-025-00530-0

[11] Hu Z, Ma X, Liu Z, et al (2016) Harnessing deep neural networks with logic rules. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp 2410–2420, https://doi.org/10.18653/v1/P16-1228

[12] Huang Y, Lv T, Cui L, et al (2022) LayoutLMv3: Pre-training for document AI with unified text and image masking. In: Proceedings of the 30th ACM International Conference on Multimedia (MM '22). ACM

[13] Hwang W, Yim J, Park S, et al (2021) Spatial dependency parsing for semi-structured document information extraction. In: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021

[14] Jaques N, Gu S, Bahdanau D, et al (2017) Sequence tutor: Conservative fine-tuning of sequence generation models with KL-control. In: International Conference on Machine Learning (ICML)

[15] Khang M, Jung SC, Park S, et al (2025) KIEval: Evaluation metric for document key information extraction. In: Document Analysis and Recognition – ICDAR 2025. Springer, Lecture Notes in Computer Science, pp 270–286, https://doi.org/10.1007/978-3-032-04624-6_16

[16] Kim G, Hong T, Yim M, et al (2022) OCR-free document understanding transformer. In: European Conference on Computer Vision (ECCV). Springer, Lecture Notes in Computer Science, pp 498–517, https://doi.org/10.1007/978-3-031-19815-1_29

[17] Kišš M, Hradiš M, Beneš K, et al (2024) SoftCTC—semi-supervised learning for text recognition using soft pseudo-labels. International Journal on Document Analysis and Recognition (IJDAR) 27(2):177–193. https://doi.org/10.1007/s10032-023-00452-9

[18] Kool W, van Hoof H, Welling M (2019) Buy 4 REINFORCE samples, get a baseline for free! In: ICLR 2019 Workshop on Deep Reinforcement Learning Meets Structured Prediction

[19] Lambert N, Morrison J, Pyatkin V, et al (2025) Tülu 3: Pushing frontiers in open language model post-training. In: Conference on Language Modeling (COLM)

[20] Lamott M, Shakir MA, Ulges A, et al (2025) SlimDoc: Lightweight distillation of document transformer models. International Journal on Document Analysis and Recognition (IJDAR) 28(3):457–473. https://doi.org/10.1007/s10032-025-00542-w

[21] Liu Y, Ott M, Goyal N, et al (2019) RoBERTa: A robustly optimized BERT pretraining approach. arXiv preprint arXiv:190711692

[22] Oliver A, Odena A, Raffel C, et al (2018) Realistic evaluation of deep semi-supervised learning algorithms. In: Advances in Neural Information Processing Systems (NeurIPS)

[23] Park S, Shin S, Lee B, et al (2019) CORD: A consolidated receipt dataset for post-OCR parsing. In: Workshop on Document Intelligence at NeurIPS 2019

[24] Perot V, Kang K, Luisier F, et al (2024) LMDX: Language model-based document information extraction and localization. In: Findings of the Association for Computational Linguistics: ACL 2024, pp 15140–15168, https://doi.org/10.18653/v1/2024.findings-acl.899

[25] Poulain d'Andecy V, Hartmann E, Rusiñol M (2018) Field extraction by hybrid incremental and a-priori structural templates. In: 2018 13th IAPR International Workshop on Document Analysis Systems (DAS). IEEE, pp 251–256, https://doi.org/10.1109/DAS.2018.29

[26] Ratner A, Bach SH, Ehrenberg H, et al (2017) Snorkel: Rapid training data creation with weak supervision. Proceedings of the VLDB Endowment 11(3):269–282. https://doi.org/10.14778/3157794.3157797

[27] Reimers N, Gurevych I (2017) Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp 338–348, https://doi.org/10.18653/v1/D17-1035

[28] Roth D, Yih Wt (2004) A linear programming formulation for global inference in natural language tasks. In: Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL), pp 1–8

[29] Rusiñol M, Benkhelfallah T, Poulain d'Andecy V (2013) Field extraction from administrative documents by incremental structural templates. In: 2013 12th International Conference on Document Analysis and Recognition (ICDAR). IEEE, pp 1100–1104, https://doi.org/10.1109/ICDAR.2013.223

[30] Saifullah S, Agne S, Dengel A, et al (2023) Analyzing the potential of active learning for document image classification. International Journal on Document Analysis and Recognition 26(3):187–209. https://doi.org/10.1007/s10032-023-00429-8

[31] Sarawagi S, Cohen WW (2004) Semi-Markov conditional random fields for information extraction. In: Advances in Neural Information Processing Systems 17. MIT Press, pp 1185–1192

[32] Schuster D, Muthmann K, Esser D, et al (2013) Intellix – end-user trained information extraction for document archiving. In: 2013 12th International Conference on Document Analysis and Recognition (ICDAR). IEEE, pp 101–105, https://doi.org/10.1109/ICDAR.2013.28

[33] Šimsa Š, Šulc M, Uřičář M, et al (2023) DocILE benchmark for document information localization and extraction. In: International Conference on Document Analysis and Recognition (ICDAR). Springer, Lecture Notes in Computer Science, pp 147–166, https://doi.org/10.1007/978-3-031-41679-8_9

[34] Šimsa Š, Uřičář M, Šulc M, et al (2023) Extended overview of DocILE 2023: Document information localization and extraction. In: Aliannejadi M, Faggioli G, Ferro N, et al (eds) Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2023), pp 546–571

[35] Sohn K, Berthelot D, Li CL, et al (2020) FixMatch: Simplifying semi-supervised learning with consistency and confidence. In: Advances in Neural Information Processing Systems (NeurIPS), pp 596–608

[36] Straka J, Gruber I (2023) Object detection pipeline using YOLOv8 for document information extraction. In: Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2023), pp 583–597

[37] Tran BG, Bao DNM, Bui KG, et al (2023) RoBERTa ensemble technique for document information localization and extraction. In: Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2023), pp 572–582

[38] Wang J, Jin L, Ding K (2022) LiLT: A simple yet effective language-independent layout transformer for structured document understanding. In: Proceedings of ACL

[39] Wang Y, Du J, Ma J, et al (2023) USTC-iFLYTEK at DocILE: A multi-modal approach using domain-specific GraphDoc. In: Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2023), pp 598–610

[40] Williams RJ (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning 8(3–4):229–256. https://doi.org/10.1007/BF00992696

[41] Wolf F, Fink GA (2024) Self-training for handwritten word recognition and retrieval. International Journal on Document Analysis and Recognition (IJDAR) 27(3):225–244. https://doi.org/10.1007/s10032-024-00484-9

[42] Xu J, Zhang Z, Friedman T, et al (2018) A semantic loss function for deep learning with symbolic knowledge. In: International Conference on Machine Learning (ICML)

[43] Xu Y, Xu Y, Lv T, et al (2021) LayoutLMv2: Multi-modal pre-training for visually-rich document understanding. In: Annual Meeting of the Association for Computational Linguistics (ACL)

[44] Yue Y, Chen Z, Lu R, et al (2025) Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In: Advances in Neural Information Processing Systems (NeurIPS)

[45] Zhang Q, Song Y, Guo P, et al (2024) CRMSP: A semi-supervised approach for key information extraction with class-rebalancing and merged semantic pseudo-labeling. arXiv preprint arXiv:240715873

[46] Ziegler DM, Stiennon N, Wu J, et al (2019) Fine-tuning language models from human preferences. arXiv preprint arXiv:190908593