PM3-KIE: A Probabilistic Multi-Task Meta-Model for Document Key Information Extraction

Birgit Kirsch and **Héctor Allende-Cid** and **Stefan Rüping**
Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS,
Sankt Augustin, Germany,
{hector.allende-cid, birgit.kirsch, stefan.rueping}@iais.fraunhofer.de

Abstract

Key Information Extraction (KIE) from visually rich documents is commonly approached as either fine-grained token classification or coarse-grained entity extraction. While token-level models capture spatial and visual cues, entity-level models better represent logical dependencies and align with real-world use cases. We introduce PM3-KIE, a probabilistic multi-task meta-model that incorporates both fine-grained and coarse-grained models. It serves as a lightweight reasoning layer that jointly predicts entities and all appearances in a document. PM3-KIE incorporates domain-specific schema constraints to enforce logical consistency and integrates large language models for semantic validation, thereby reducing extraction errors. Experiments on two public datasets, DeepForm and FARA, show that PM3-KIE outperforms three state-of-the-art models and a stacked ensemble, achieving a statistically significant 2% improvement in F1 score.

1 Introduction

Key Information Extraction (KIE) focuses on identifying structured key-value pairs from visually rich documents (VRDs) (Huang et al., 2019) based on a predefined schema that specifies the target information types. This capability is essential for automating business document processing across industries such as finance and law. Automating information extraction significantly reduces operational costs: processing a single invoice, for example, can cost $13.11 and takes up to eight days (Girsch-Bock, Mary, 2024; Cohen and York, 2020). Despite recent advances, extracting structured data remains challenging for state-of-the-art models, particularly for documents with complex schemas or semi-structured layouts (Wang et al., 2023b).

KIE can generally be approached through two distinct paradigms, as illustrated in Figure 1:

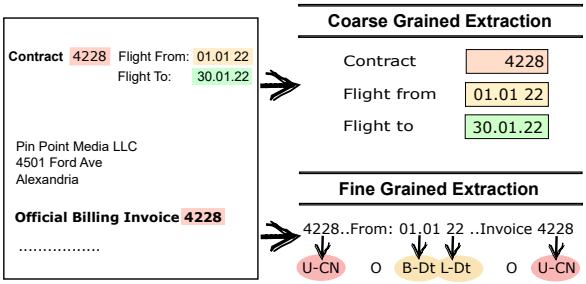

Figure 1: Fine-Grained Token Classification vs. Coarse-Grained Entity Extraction Task for KIE

Fine-Grained Token Classification The fine-grained task operates at the token level (illustrated at the bottom of Figure 1), where a label is assigned to each token. Fine-grained models typically leverage multi-modal transformer encoders that integrate textual, spatial, and visual features, such as LayoutLMv3 (Huang et al., 2022) and LiLT (Wang et al., 2022). While effective in capturing token-level spatial and visual relationships, these models require large annotated datasets for training.

Coarse-Grained Entity Extraction In contrast, coarse-grained entity extraction targets high-level entities within a document (illustrated at the top of Figure 1). Coarse-grained models may structure outputs as key-value pairs in formats such as JSON (Cesista et al., 2024), retrieve entity values (Cao et al., 2023), or classify entities among predefined candidates (Majumder et al., 2020). These models excel at capturing logical dependencies and generating coherent outputs and align more closely with business document processing needs like the extraction of invoice elements.

Both approaches offer distinct advantages, highlighting the need for models that effectively integrate them. Ding et al. (2024) address this by proposing a knowledge distillation framework that combines knowledge from both model types. However, their method relies on model classification

Findings of the Association for Computational Linguistics: ACL 2025, pages 20890–20912

logits, restricting it to specific architectures and making it incompatible with coarse-grained models that generate structured output, such as recent Large Language Model (LLM)-based approaches

Moreover, existing methods often fail to enforce structural constraints defined by the extraction schema. Certain elements, such as document identification numbers, are mandatory, while others are optional. Current models frequently omit required information, leading to schema violations. While heuristic post-processing can mitigate some of these errors, it does not ensure full consistency.

Additionally, state-of-the-art approaches struggle with syntactic errors, such as incomplete address information (Wang et al., 2023b) and semantic errors, such as implausible entity values (see Appendix F).

To address these limitations, we introduce **PM3-KIE**, a probabilistic multi-task meta-model. PM3-KIE integrates fine-grained and coarse-grained approaches using a lightweight probabilistic reasoning layer based on Probabilistic Soft Logic (PSL) (Bach et al., 2017), ensuring robust inference while maintaining computational efficiency. PM3-KIE incorporates four key innovations:

- **Black-Box Model Integration** – PM3-KIE flexibly integrates both fine-grained and coarse-grained KIE models, including generative approaches for structured output, without architectural constraints.
- **Joint Prediction** – The model jointly predicts coarse-grained entities and their corresponding fine-grained mentions, ensuring a consistent overall extraction.
- **Schema Consistency** – Logical constraints enforce adherence to extraction schemas, guaranteeing the presence of required information while minimizing structural errors.
- **LLM-as-a-Judge-based Validation** – Pretrained LLMs validate extractions based on syntactic and semantic plausibility, reducing errors such as incorrect formatting or implausible entity values.

We evaluate PM3-KIE on two public datasets, DeepForm and FARA (Wang et al., 2023b), demonstrating significant improvements over state-of-the-art models and a stacked ensemble. Our experiments cover out-of-distribution and low-resource settings, highlighting PM3-KIE's robustness across diverse real-world challenges such as small training sizes and unknown document formats during application time. Our contributions can be summarized as following:

- We introduce a probabilistic multi-task meta-model that integrates fine- and coarse-grained KIE models with domain-specific schema constraints and LLM-based validation.
- PM3-KIE achieves state-of-the-art performance on two public datasets, outperforming all other models by a statistically significant margin.

The remainder of this paper is structured as follows: Section 2 reviews related work, followed by an outline of the KIE problem in Section 3. Next, Section 4 describes the PM3-KIE architecture, while Section 5 presents experimental results. Finally, Section 6 concludes with a discussion.

2 Related Work

KIE involves identifying key-value pairs from documents (Huang et al., 2019) and can be performed as fine-grained token sequence classification or coarse-grained entity extraction at the document level, detailed in the following subsections.

2.1 Fine-Grained Token Classification Models

Traditional Models Neural networks redefined KIE as a token sequence classification task, initially using Recurrent Neural Networks (RNNs) (Palm et al., 2017) and later incorporating spatial features for improved layout representation (Sage et al., 2019). Convolutional architectures followed, leveraging non-sequential spatial context (Zhao et al., 2019; Katti et al., 2018; Denk and Reisswig, 2019; Zhang et al., 2020). Graph-based models further advanced KIE by modeling textual and spatial relationships with Graph Convolutional Networks (Yao et al., 2024; Shi et al., 2023; Lee et al., 2022, 2021; Wei et al., 2020; Yu et al., 2020; Hwang et al., 2020).

Multimodal Transformer Approaches Transformer-based models now dominate multimodal token classification in KIE, leveraging various enhancements to better integrate textual and visual features in business documents. DocFormer (Appalaraju et al., 2021), LAMBERT (Garncarek et al., 2021), FormNet (Lee et al., 2022), and ERNIE-Layout (Peng et al., 2022)

focus on capturing spatial relationships through attention mechanisms. LayoutLMv3 (Huang et al., 2022), LAMBERT, Docformer, Structext (Li et al., 2021), and UDOP (Tang et al., 2023) incorporate layout-aware embeddings to better represent document structures, a technique employed in models. Furthermore, LayoutMask (Tu et al., 2023), StructextV2 (Yu et al., 2023), Structext, UDOP, DocFormer, ERNIE-Layout and Wokung-Reader (Bai et al., 2023) have introduced pre-training tasks designed to leverage multimodal information. In LiLT, Wang et al. (2022) adapted the transformer concepts to work in a language-independent way.

2.2 Coarse-Grained Entity Extraction

Traditional and Encoder-Decoder Architectures Other works frame KIE as an entity extraction task. This can involve detecting candidates and classifying them, as proposed in RELIE (Majumder et al., 2020), or directly generating structured outputs using encoder-decoder models such as TILT (Powalski et al., 2021) and Donut (Kim et al., 2022), with TILT being one of the first to utilize a decoder to generate sequences that contain all entities.

Generative LLMs Recent works leverage LLMs with natural language prompts for KIE using two primary strategies: *Per-Key Prompting*, where individual prompts query values for each key (e.g., DocLLM (Wang et al., 2023a), LayoutLLM (Luo et al., 2024)); and *Unified Prompting*, where a single prompt queries all keys simultaneously, producing either plain text outputs with all values (e.g., GenKIE (Cao et al., 2023)) or structured outputs such as key-value pairs or JSON (e.g., RASG (Cesista et al., 2024), ICL-D3IE (He et al., 2023), LMDX (Perot et al., 2024)).

Prompting Strategies Approaches like RASG, ICL-D3IE, and LMDX, enhance LLMs for KIE by leveraging in-context learning, document layout encoding, and advanced prompting techniques. RASG employs retrieval-augmented structured generation, framing structured output prediction as a tool-use approach. ICL-D3IE utilizes in-context learning with demonstrations to emphasize positional relationships. LMDX integrates document layout into prompts and introduces a decoding mechanism for structured entity extraction.

Joint Token Classification and Entity Extraction Models Ding et al. (2024) address the joint task of token classification and entity extraction using a student-teacher framework for knowledge distillation. However, their approach is limited to architectures that produce classification logits, making it incompatible with decoder-based LLMs that generate structured outputs.

In contrast, we propose a more flexible multi-task meta-learning approach that integrates fine-grained and coarse-grained black-box models. By jointly predicting entities and their mentions and leveraging domain-specific schemas, our method enforces extraction consistency and overcomes the limitations of existing systems.

3 KIE Problem Statement

In this section, we first review traditional KIE tasks: fine-grained token classification and coarse-grained entity extraction. We then present a unified problem formulation that combines these tasks into a joint extraction framework, enabling the simultaneous extraction of entities and their mentions in a document. Finally, we extend this to a meta-modeling task for joint prediction conditioned on fine-grained and coarse-grained models.

3.1 Fine-grained Token Classification

Let $W = \{w_s\}_{s=1}^{|S|}$ denote the token sequence in a document d, with $|S|$ as the sequence length. $T = \{t_i\}_{i=1}^{|T|}$ represents the set of possible label types and |T| the number of labels. The objective is to assign a label $l_s \in T$ to each token w_s.

3.2 Coarse-grained Entity Extraction

The objective is to identify a set of key-value pairs $K = \{(k_n, v_n)\}_{n=1}^{|N|}$, where each pair (k_n, v_n) consists of a field type $k_n \in T$ and an associated field value v_n. The number of extracted pairs is $|N|$. Each value v_n is typically a subsequence of tokens from the document, $v_n \subseteq W = \{w_s\}_{s=1}^{|S|}$.

3.3 Unified Prediction

We introduce a unified problem formulation integrating both tasks into a joint framework that enables a more holistic representation of the document content. In this formulation, we define an extraction schema $T = \{t_i\}_{i=1}^{|T|}$, which specifies the types of information t_i to be extracted, such as *document IDs*. The entities that fill these schema slots are referred to as *fields*, extracted through coarse-grained entity extraction. Within the document, spans of text referring to these fields are termed *field mentions*. They are extracted through

fine-grained token classification. Multiple field mentions can correspond to the same field. The task is to identify both fields and their associated field mentions, populating the schema.

We further extend this to a meta-modelling task with the objective to jointly predict field mentions and fields conditioned on black-box fine- and coarse-grained models. Specifically, we define a set of $|Q|$ fine-grained prediction models, $O = \{O^q\}_{q=1}^{|Q|}$, each producing a predicted label sequence $L^q = \{l_s^q\}_{s=1}^{|S|}$. and a set of $|R|$ coarse-grained prediction models, $U = \{U^r\}_{r=1}^{|R|}$, each producing key-value pairs $K^r = \{(k_n^r, v_n^r)\}_{n=1}^{|N|}$.

For a document d and token sequence W, we define two sets of candidates for final fields and field mentions, derived from the model outputs:

- **Field Mention Candidates** (M^i): For a given field type t_i, the set of mention candidates is defined as a subset of tokens in w predicted to belong to label t_i by at least one model in O:

$$M^i = \{w_j \mid j \in \{1, \ldots, |S|\}, \exists \; q \in \{1, \ldots, |Q|\}, l_j^q = t_i\}. \tag{1}$$

- **Field Candidates** (F^i): For a given field type t_i, the field candidate set is defined as a subset of all field values v_n predicted to belong to the type t_i by at least one model in U:

$$F^i = \{v_c \mid c \in \{1, \ldots, |N|\}, \exists \; r \in \{1, \ldots, |R|\}, k_c^r = t_i\}. \tag{2}$$

The objective is to predict field mentions, fields and their association links, represented by random variables x, y and a:

Field Mentions $x_{ji} = \begin{cases} 1 & \text{if } w_j \in M^i \text{ is of type } t_i, \\ 0 & \text{otherwise.} \end{cases}$

Fields $y_{ci} = \begin{cases} 1 & \text{if } v_c \in F^i \text{ is field instance of type } t_i, \\ 0 & \text{otherwise.} \end{cases}$

Field Linking: $a_{jc} = \begin{cases} 1 & \text{if } w_j \text{ is linked to } v_c, \\ 0 & \text{otherwise.} \end{cases}$

4 Methodology

4.1 P3M-KIE Model Architecture

We introduce PM3-KIE, a probabilistic reasoning model that jointly predicts fields and their respective mentions in the document, conditioned on fine-grained and coarse- grained models. This meta-model features a logical decision layer based on PSL (Bach et al., 2017), as detailed in Subsection 4.2. Subsequent sections elaborate on PM3-KIE's architecture, that is depicted in Figure 2. Subsection 4.3 presents the integration of fine- and coarse-grained models for joint prediction. Subsection 4.4 introduces the modeling of field cardinalities, while Subsection 4.5 describes the incorporation of LLMs as semantic and syntactic validators. Finally, Subsection 4.6 outlines the learning and inference mechanisms within the proposed framework.

4.2 Probabilistic Framework

Probabilistic Graphical Model PSL represents a probability distribution over a set of random variables using a *Hinge-Loss Markov Random Field* (HL-MRF). The probability density function for unobserved variables $Y = (Y_1, \ldots, Y_{n'})$, conditioned on observed variables $X = (X_1, \ldots, X_n)$, is expressed as:

$$P(Y|X) = \frac{1}{Z(\omega, X)} \exp\left[-\sum_{j=1}^{m} \omega_j \phi_j(X, Y)\right], \tag{3}$$

where ϕ_j are potential functions, ω_j are their associated weights, and $Z(\omega, X)$ is the normalization factor. This formulation allows joint reasoning over interdependent variables.

Declarative Logic Rules PSL employs a declarative language to formulate logical rules that define the underlying HL-MRF. Rules represent templates for potential functions. For example:

$$w : \texttt{Prediction(}\textbf{\texttt{mention}}\texttt{, }\textbf{\texttt{type}}\texttt{)} \implies \texttt{IsType(}\textbf{\texttt{mention}}\texttt{, }\textbf{\texttt{type}}\texttt{).} \tag{Rule 1}$$

Each rule consists of *predicates* (Prediction, IsType) representing observed or unobserved variables and *variables* (mention, type) serving as placeholders for constants. w indicates the rule's importance. The probability distribution described by the model is derived by grounding these rules - which means replacing all variables (e.g. `mention`, `type`) with constants.

Learning and Inference PSL learns optimal rule weights w by maximizing a likelihood function. Inference determines the most probable assignments for unobserved variables, framed as a convex optimization problem (see Bach et al. (2017) for a detailed description).

4.3 Multi-Task Meta-Model Design

Integration of Fine-Grained Models We introduce an unobserved predicate $Me^i(\mathbf{d}, \mathbf{m})$ for each

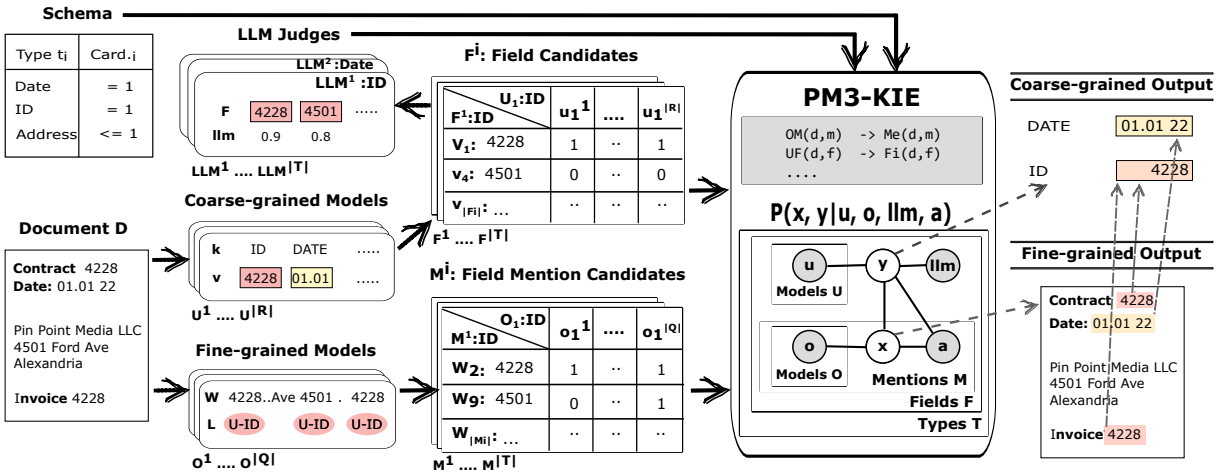

Figure 2: Shows the incoporation of fine- and coarse-grained black-box models (O and U), as well as schema constraints into the PM3-KIE meta-model. From left to right: A document d is processed. Coarse- (O) and fine-grained (U) models generate predctions o_{iq} and u_{iq} for each field type t_i and prediction module O_q, U_q, that define the candidate sets for fields and field mentions. An LLM evaluates each candidate for fact and format correctness. The probabilistic meta-model jointly reasons based on schema constraints, model outputs, LLM scores and candidates. It jointly predicts field values and their respective mentions in the document.

field type $t_i \in T$, where $\mathbf{d}$ represents a document, and $\mathbf{m}$ is a mention candidate in M^i. This predicate encodes the true field mention prediction for each candidate x_{ji}.

Let o^q_{ji} denote the output of a prediction model O^q for a specific label t_i and a mention candidate w_j in M^i. For models that predict a label for each word, we define $o^q_{ji} = \mathbf{1}\left(l^q_j = t_i\right)$, where $\mathbf{1}(\cdot)$ is an indicator function returning 1 if the label assigned to w_j by O^q is t_i, and 0 otherwise.

For models outputting a probability distribution over labels $t_i \in T$, o^q_{ji} is defined as the probability assigned by O^q to word w_j and label t_i: $o^q_{ji} = P(t_i \mid w_j, O^q)$,

We define an observed predicate $OM^{q,i}(\mathbf{d}, \mathbf{m})$ for each model O^q, representing the output o^q_{ji} for field type t_i. Here, $\mathbf{d}$ corresponds to the document and $\mathbf{m}$ to the mention candidate in M^i. This predicate encapsulates the model's predictions for each mention in a document.

Rule 2 and Rule 3 establish the relationship between the outputs of models O and the unobserved field mention predictions $Me^i(d, m)$ for each field type t_i and model O^q:

$$w_1^{qi} : OM^{q,i}(\mathbf{d}, \mathbf{m}) \Rightarrow Me^i(\mathbf{d}, \mathbf{m}), \qquad \text{(Rule 2)}$$

$$w_2^{qi} : \neg OM^{q,i}(\mathbf{d}, \mathbf{m}) \Rightarrow \neg Me^i(\mathbf{d}, \mathbf{m}). \qquad \text{(Rule 3)}$$

These rules adjust the likelihood of assigning a field type t_i to a mention candidate $\mathbf{m}$ based on model outputs. The probability of $Me^i(\mathbf{d}, \mathbf{m})$ increases as more models predict t_i, while the probability decreases when models predict alternative labels.

Integration of Coarse-Grained Models We introduce an unobserved predicate $Fi^i(\mathbf{d}, \mathbf{f})$ for each field type t_i in T, where $\mathbf{d}$ is a constant representing a document and $\mathbf{f}$ a field candidate in F^i. This predicate indicates the true field mention prediction for each candidate y_{ji}.

Let u^r_{ci} denote the output of a prediction model U^r for a specific label t_i and a field candidate v_c in F^i.

For models producing a predicted label for each word, we define $u^r_{ci} = \mathbf{1}\left(k^r_c = t_i\right)$. $\mathbf{1}(\cdot)$ is an indicator function that outputs 1 if the label assigned by the model to value v_c is t_i, and 0 otherwise.

In the case of models outputting a probability distribution over all labels $t_i \in T$, we define u^r_c as the probability assigned by U^q to value v_c and label t_i: $u^r_{ci} = P(t_i \mid v_c, U^q)$.

We introduce an observed predicate $UF^{q,i}(\mathbf{d}, \mathbf{f})$ for each model in U, representing the output u^{qi} for field type t_i, with a constant $\mathbf{d}$ for every document and $\mathbf{f}$ for every field candidate in F^i.

Rule 4 and Rule 5 correlate the outputs of the models U with the unobserved field prediction $Fi^i(\mathbf{d}, \mathbf{f})$ for each field type t_i and model U^r:

$$w_3^{ri} : UF^{ri}(\mathbf{d}, \mathbf{f}) \Rightarrow Fi^i(\mathbf{d}, \mathbf{f}) \qquad \text{(Rule 4)}$$

$$w_4^{ri} : !UF^{ri}(\mathbf{d}, \mathbf{f}) \Rightarrow !Fi^i(\mathbf{d}, \mathbf{f}) \qquad \text{(Rule 5)}$$

These rules increase the likelihood of assigning a field type t_i to a field candidate as more mod-

els predict this label, and conversely, decrease the probability if models predict a different field type.

Linking Fine- and Coarse-Grained Models The task involves predicting a field link a_{jc} for each pair of field mention and field candidate in F_i and M_i. We define a linking function, ranging from 0 to 1, to indicate the likelihood that a field mention candidate belongs to a field candidate. This function utilizes string or embedding similarity, normalized to $[0, 1]$, with the Jaccard distance used for string comparison (see Appendix A). To capture the interdependence between field mentions and fields, we introduce rules Rule 6 and Rule 7. This ensures that field candidates and their mentions share the same field type t_i and enables bidirectional propagation of predicted field types:

$$w_5^i : Me^i(\mathbf{d}, \mathbf{m}) \wedge Lnk(\mathbf{d}, \mathbf{m}, \mathbf{f}) \Rightarrow Fi^i(\mathbf{d}, \mathbf{f}) \quad \text{(Rule 6)}$$

$$w_6^i : Fi^i(\mathbf{d}, \mathbf{f}) \wedge Lnk(\mathbf{d}, \mathbf{m}, \mathbf{f}) \Rightarrow Me^i(\mathbf{d}, \mathbf{m}) \quad \text{(Rule 7)}$$

4.4 Cardinality Constraints

In many domains, field types are subject to cardinality constraints. For instance, business documents often contain a unique identification number and multiple entries for fields such as addresses. We extend the information schema by associating each field type t_i with a cardinality cardinality_i and specifying whether the field is mandatory or optional. For mandatory fields, Rule 8 enforces that exactly cardinality_i fields of type t_i are extracted per document. For optional fields, Rule 9 ensures that the number of instances does not exceed cardinality_i.

$$w_7^i : F^i(\mathbf{d}, +\mathbf{f}) = \text{cardinality}_i \quad \text{(Rule 8)}$$

$$w_8^i : F^i(\mathbf{d}, +\mathbf{f}) \leq \text{cardinality}_i \quad \text{(Rule 9)}$$

Here, $F^i(\mathbf{d}, +\mathbf{f})$ denotes the summation over all extracted fields of type t_i in document $\mathbf{d}$.

4.5 LLM-as-a-Judge

Common model errors in information extraction include format inconsistencies (e.g., missing years in dates) and factual inaccuracies (e.g., misidentifying a postbox as a headquarters). To systematically verify field candidates F^i, we propose using a in-context-tuned LLM that assigns a score $llm_{ci} \in [0, 1]$ to each field candidate $v_c \in F^i$, representing its validity based on format and factual accuracy. This mechanism is integrated using the following rules:

$$w_9^i : LLM^i(\mathbf{d}, \mathbf{f}) \Rightarrow F^i(\mathbf{d}, \mathbf{f}) \quad \text{(Rule 10)}$$

$$w_{10}^i : \neg LLM^i(\mathbf{d}, \mathbf{f}) \Rightarrow \neg F^i(\mathbf{d}, \mathbf{f}) \quad \text{(Rule 11)}$$

The LLM predicate $LLM^i(\mathbf{d}, \mathbf{f})$ indicates document d supports field assignment f. These rules increase confidence in F^i when the score is high and decrease it otherwise. This mechanism can be extended to include human annotations, regular expressions, and database lookups, further enhancing the model's robustness.

4.6 Learning and Inference

As shown in Figure 2, PM3-KIE defines a probability distribution over unobserved random variables (field mentions x and fields y), conditioned on observed variables: fine-grained model outputs (o), coarse-grained model outputs (u), candidate links (a), and LLM scores (llm). The potential functions governing these distribution are defined by logical rules in Formula 3. The weights ω are learned by maximizing the likelihood of observed data. During inference, the most probable field mention and assignment are identified by maximizing the joint probability distribution in Formula 3.

5 Experiments

5.1 Experimental Setup

We evaluate PM3-KIE on two public datasets, DeepForm and FARA, comparing it against three state-of-the-art models: LayoutLMv3, LILT, and a fine-tuned GPT-3.5 Turbo (Radford and Narasimhan, 2018; OpenAI, 2024), as well as a stacked ensemble (Wolpert, 1992). Additionally, we conduct an ablation study to assess the impact of individual PM3-KIE components on overall performance. Appendix G provides a list of software components used, along with their licenses.

Data DeepForm consists of complex political ad invoices, annotated with nine fields: `advertiser`, `contract_num`, `gross_amount`, `property`, `tv_address`, `flight_from`, `flight_to`, `product`, and `agency`. FARA contains foreign agent registration forms, annotated with six fields: `registration_num`, `principal_name`, `registrant_name`, `signer_name`, `file_date`, and `signer_title`.

Both datasets include predefined train, develop, and test splits, covering both in-distribution (ID) and out-of-distribution (OOD) test sets. To evaluate performance under varying data availability, we experiment with different training sizes (10, 50, 100, and 200 samples), using one ID and one

OOD fold per size, with three splits per (train size, distribution)-combination, totaling 24 models.

Evaluating on OOD test documents and limited training data provides insight into model robustness under low-resource conditions and distribution shifts—challenges frequently encountered in real-world applications. Details on the datasets and preprocessing steps are provided in Appendix H.

Evaluation Metrics We apply field-specific matching functions (Wang et al., 2023b) to normalize values and introduce functions to address mismatches (see Appendix H.2). Since document-level metrics provide a more practical measure of workload for correcting document extractions than field-level F1 scores (Nourbakhsh et al., 2024), we report multiple field- and document level metrics, listed in Table 1. Statistical significance is evaluated using paired differences and 95% confidence intervals, calculated with paired standard errors (Miller, 2024). Both are computed for each split and (train size-distribution)-combination.

Metric	Description
F1 per Field	Mean F1 across fields.
F1 per Doc	Mean F1 across documents.
Hit per Doc	Mean hit rate, "hit" = correct field extraction.
Doc-Level Accuracy	perc. of documents with correct extraction

Table 1: Performance Metrics for KIE

Baselines We compare PM3-KIE against three fine-tuned models—LayoutLMv3, Lilt, and GPT-3.5 Turbo—which serve as baselines and input predictors for PM3-KIE (see Section 4). All models are fine-tuned on the training fold with hyperparameters tuned on the validation fold. We also include a logistic regression ensemble over the three baselines, trained on the validation data. Model architectures, prompt templates, training details, and hyperparameters are provided in Appendix B.

PM3-KIE Model The final PM3-KIE model integrates all components from Section 4, combining baseline models—Lilt, LayoutLMv3, and GPT-3.5 Turbo—with cardinality constraints. These specify required fields (DeepForm: contract number, gross amount, advertiser, property; FARA: registration_num, registrant_name and file_date) and optional fields (remaining fields) limited to one value per document. Two LLM-based judges handle fact and format checking using automatically generated prompts without prompt engineering, representing a quality lower bound (see Appendix I). Training is performed on the development fold (Appendix C).

5.2 Results and Analysis

Baseline Comparison Table 2 summarizes the F1 scores with standard errors for PM3-KIE and baseline models. The "best model" is selected per split as the model achieving the highest F1 score. Results are averaged over three splits and four training sizes for each distribution type. The "paired difference" shows the performance difference between PM3-KIE and the best baseline with 95% confidence intervals. Results demonstrate that PM3-KIE outperforms all baselines across all data chunks with statistical significance (95%), highlighting its robustness in a variety of settings.

Document-Level Metrics To assess practical utility, we analyze document level performance and provide insights into error rates per document. Table 3 shows that PM3-KIE achieves significant improvements in document-level accuracy for both FARA and DeepForm, reducing the practical manual correction workload. This indicates the model's capability to fully automate invoice processing for over 58% of the documents and additional 5% compared to the best performing baseline model.

Model Runtime Analysis Appendix D provides model runtimes and system specifications. PM3-KIE introduces minimal computational overhead, with training completing in minutes without GPU acceleration, demonstrating its practicality across diverse hardware environments.

Impact of Train Size and Test Distribution Labeling document extraction corpora is time-consuming, leading to small training sets in practice. Additionally, distribution shifts, e.g. changes in document layouts, are common during application-time. We evaluate PM3-KIE's performance with limited training data and OOD test sets to replicate these challenges. Figure 3 shows performance improvements based on training size and test distribution, averaged over three splits. The results demonstrate that PM3-KIE excels in low-resource settings and on OOD test sets, underscoring its relevance for real-world applications.

5.3 Ablation Study

We conduct an ablation study to assess the contributions of individual model components. Models are trained on three splits for each combination of

Table 2: Performance metrics for FARA and DeepForm (DF) calculated on ID and OOD test sets. Reported F1 scores are averaged across fields, accompanied by their corresponding standard errors in parentheses. Paired difference between PM3-KIE and the best model is presented, along with a 95% confidence interval in parentheses. All values are averaged across three splits and four training sizes. A detailed result table is displayed in Appendix E

Dataset	Dist	#Test	F1 LILT (std. error)	F1 GPT (std. error)	F1 LMv3 (std. error)	F1 STE (std. error)	F1 Best (std. error)	F1 PM3-KIE (std. error)	Paired Diff (conf. interval)
FARA	ID	76	84.2 (1.2)	92.9 (0.8)	69.5 (1.4)	92.6 (0.8)	93.2 (0.8)	**94.5 (0.7)**	1.6 (+0.8, +2.5)
FARA	OOD	224	73.0 (1.3)	93.9 (0.8)	47.1 (1.2)	93.6 (0.7)	94.1 (0.7)	**94.7 (0.7)**	0.8 (+0.1, +1.4)
DF	ID	300	91.7 (2.4)	90.1 (2.6)	85.3 (2.7)	86.6 (2.2)	92.4 (2.0)	**93.7 (1.6)**	1.8 (+1.0, +2.6)
DF	OOD	300	84.7 (1.1)	85.2 (1.4)	77.6 (1.4)	81.9 (1.1)	88.0 (1.1)	**90.5 (0.9)**	3.0 (+0.7, +5.4)

Table 3: Comparison of Best Model and PM3-KIE performance across FARA and DeepForm datasets. Detailed results per dataset are displayed in Appendix E

Dataset	Model	F1 per Field	Hit per Doc	F1 per Doc	Doc Lv. Acc.
FARA	Best Model	93.63	95.45	93.28	78.97
FARA	**PM3-KIE**	**94.59**	**96.50**	**94.47**	**82.92**
DF	Best Model	90.22	89.86	88.83	53.55
DF	**PM3-KIE**	**92.09**	**91.84**	**91.25**	**58.27**

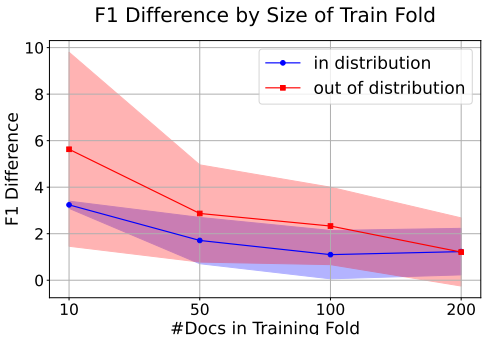

Figure 3: F1 difference (F1_PM3-KIE - F1_BestModel) with 95% confidence interval by train size for both id and ood test folds on DeepForm.

training size and distribution type (ID and OOD). Table 4 summarizes the findings. The basic model is a model learned only on the final field extraction data with two LLM judges for fact- and format-checking and with schema constraints.

Granular Labeling We evaluate the impact of granular field mention labels by comparing a multitask learning approach ("+Token Task") with a single-task setup. Results show a slight decline in performance with multitask learning, likely due to token-level annotation noise. Given the task's document-level requirements, single-task learning appears more efficient in this setting.

Cardinality Constraints We evaluate cardinality constraints by training models without them ("-Constraints"). Performance declines across all metrics, confirming their positive effect, likely due to the constraints ensuring required fields are predicted, which would otherwise be omitted.

Table 4: Ablation study results on DeepForm. Difference in F1 for the basic PM3-KIE model compared to the adapted model version. Results for FARA are displayed in table 10 in Appendix E.

Ablation	F1 Field	Hit Doc	F1 Doc	Acc
PM3-KIE	**92.09**	**91.84**	**91.25**	**58.27**
+ Token Task	-0.43	-0.20	-0.33	-0.03
- Constraints	-1.81	-1.65	-1.72	-5.73
- Fact & Format Judges	-0.31	-0.25	-0.31	-0.74
- Fact Judge	-0.15	-0.09	-0.14	-0.10
- Format Judge	-0.26	-0.25	-0.25	-0.55

Effectiveness of LLM-as-a-Judge PM3-KIE integrates two LLMs for fact- and format-checking. Removing both ("-Fact Format Judges") leads to a notable 0.74% drop in document-level accuracy and slightly reduces all other metrics (0.3%), emphasizing their role in precise extractions. Removing only one ("-Fact Judge" or "-Format Judge") results in better performance than removing both but still lags behind the base model, highlighting the individual contribution of each judge.

6 Conclusion and Discussion

We propose PM3-KIE, a probabilistic multi-task meta-model for KIE that integrates fine-grained token classification and coarse-grained entity extraction models. PM3-KIE flexibly combines arbitrary models through a probabilistic reasoning layer that enforces schema consistency, reducing errors such as missing required elements. Additionally, it incorporates LLM-based validation to ensure logical adherence and semantic plausibility. Experimental results on two public datasets, DeepForm and FARA, demonstrate that PM3-KIE outperforms state-of-the-art methods across in-distribution, out-of-distribution, and low-resource scenarios, highlighting its effectiveness and robustness when facing challenges commonly encountered in practice.

7 Limitations

Sensitivity to Prompt Design and Model Capability The performance of the LLM-as-a-judge component depends on both the LLM's capabilities (e.g., model size and training data) and the quality of the prompts. For instance, effective fact-checking requires the LLM to be trained on data containing the relevant facts. Inadequate model size or poorly designed prompts may lead to unreliable validation scores. To establish a lower bound for model quality, we avoid extensive prompt engineering and instead use automatically generated prompts. While the meta-model's training adjusts weights to minimize the impact of uninformative LLM outputs, it will not benefit overall performance in such cases. Future work will explore robustness checks using noisy models to further evaluate the impact of prompt quality.

Dependency on Base Models: The extraction quality of the meta-model relies on the performance of the base fine-grained and coarse-grained models. If all base models fail to detect a true extraction as potential candidates, they cannot be identified by the overall system. This limitation is common for most ensembling and meta-model approaches.

Computational Costs: While PM3-KIE itself is lightweight with relatively few parameters, the meta-model's complexity arises from integrating fine-grained and coarse-grained models along with LLMs for validation. This integration increases both computational overhead and deployment complexity, as it requires managing multiple models in conjunction with the meta-model. This is a common challenge in ensemble and meta-model approaches.

Additional Processing: PM3-KIE requires data to be formatted as specified in Section 4, including the creation of constants for predicates and truth values for observed predicates. This necessitates additional computational effort for postprocessing both model outputs and input data to meet the required format.

Closed-World Assumption: PM3-KIE assumes a closed-world setting in which all fields to be extracted are known in advance, limiting its applicability in scenarios that require the detection of novel or previously unseen field types. However, this constraint is not a practical limitation in the domain we target—business document process automation—where open-domain extraction is intentionally avoided. In this setting, document schemas are well-defined, and downstream systems rely on a fixed set of fields for automated processing. As such, adherence to a predefined extraction schema is both expected and beneficial.

Dependency on Parsed Strings: This work assumes input documents are in a machine-readable format, typically processed through OCR. OCR error correction and parsing accuracy are beyond the scope of this study, with the approach presuming that such errors have been corrected prior to downstream processing.

8 Ethical Considerations

Automation and Job Displacement: Automating key information extraction from documents, especially in business-critical domains like finance and legal, could reduce the demand for manual data entry and administrative roles. While this improves efficiency and reduces operational costs, it risks unemployment for workers currently performing these tasks.

Risk of Overreliance on Automated Systems: Deploying PM3-KIE in critical sectors, such as healthcare, legal documentation, or property records, may lead to errors being accepted without human verification. Incorrect or incomplete extractions could have significant consequences, including legal disputes, financial losses, or medical errors. PM3-KIE should always operate in a semi-automated manner with manual review.

Bias and Fairness Concerns: Like many AI systems, PM3-KIE's performance depends on the quality and diversity of training data used to train the base models integrated. Biases in the training data could lead to unequal performance across document types, languages, or regions, potentially disadvantaging users from underrepresented groups. Care must be taken to curate balanced datasets and evaluate the model across diverse scenarios.

Acknowledgments

This research has been funded by the Federal Ministry of Education and research of Germany and the state of North Rhine-Westphalia as part of the Lamarr Institute for Machine Learning and Artificial Intelligence and by the Ministry of Economic Affairs, Industry, Climate Action and Energy of the

State of North Rhine-Westphalia [005-2011-0048] as part of the flagship project ZERTIFIZIERTE KI.

References

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)*, page 973–983. IEEE.

Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2017. Hinge-loss markov random fields and probabilistic soft logic. *J. Mach. Learn. Res.*, 18(1):3846–3912.

Haoli Bai, Zhiguang Liu, Xiaojun Meng, Li Wentao, Shuang Liu, Yifeng Luo, Nian Xie, Rongfu Zheng, Liangwei Wang, Lu Hou, Jiansheng Wei, Xin Jiang, and Qun Liu. 2023. Wukong-reader: Multi-modal pre-training for fine-grained visual document understanding. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13386–13401, Toronto, Canada. Association for Computational Linguistics.

Panfeng Cao, Ye Wang, Qiang Zhang, and Zaiqiao Meng. 2023. GenKIE: Robust generative multimodal document key information extraction. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Franz Louis Cesista, Rui Aguiar, Jason Kim, and Paolo Acilo. 2024. Retrieval augmented structured generation: Business document information extraction as tool use. *2024 IEEE 7th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 227–230.

B. Cohen and M. York. 2020. Ardent partners' accounts payable metrics that matter in 2020. technical report. *Ardent Partners (2020)*.

Timo I. Denk and Christian Reisswig. 2019. Bertgrid: Contextualized embedding for 2d document representation and understanding. *NeurIPS*, abs/1909.04948.

Yihao Ding, Lorenzo Vaiani, Soyeon Caren Han, Jean Lee, Paolo Garza, Josiah Poon, and Luca Cagliero. 2024. 3mvrd: Multimodal multi-task multi-teacher visually-rich form document understanding. In *Annual Meeting of the Association for Computational Linguistics*.

Łukasz Garncarek, Rafał Powalski, Tomasz Stanisławek, Bartosz Topolski, Piotr Halama, Michał Turski, and Filip Graliński. 2021. *LAMBERT: Layout-Aware Language Modeling for Information Extraction*, page 532–547. Springer International Publishing.

Girsch-Bock, Mary. 2024. Invoice processing cost: What is it, how to calculate it, and how to reduce it. Reviewed on 26 November 2024.

Jiabang He, Lei Wang, Yi Hu, Ning Liu, Hui Liu, Xing Xu, and Heng Tao Shen. 2023. Icl-d3ie: In-context learning with diverse demonstrations updating for document information extraction. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, page 19428–19437. IEEE.

Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 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.

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

Wonseok Hwang, Jinyeong Yim, Seunghyun Park, Sohee Yang, and Minjoon Seo. 2020. Spatial dependency parsing for semi-structured document information extraction. In *Findings*.

Anoop R Katti, Christian Reisswig, Cordula Guder, Sebastian Brarda, Steffen Bickel, Johannes Höhne, and Jean Baptiste Faddoul. 2018. Chargrid: Towards understanding 2D documents. *EMNLP*, pages 4459–4469.

Geewook Kim, Teakgyu Hong, Moonbin Yim, JeongYeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. 2022. *OCR-Free Document Understanding Transformer*, page 498–517. Springer Nature Switzerland.

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 *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3735–3754, Dublin, Ireland. Association for Computational Linguistics.

Chen-Yu Lee, Chun-Liang Li, Chu Wang, Renshen Wang, Yasuhisa Fujii, Siyang Qin, Ashok Popat, and Tomas Pfister. 2021. Rope: Reading order equivariant positional encoding for graph-based document information extraction. In *Annual Meeting of the Association for Computational Linguistics*.

Yulin Li, Yuxi Qian, Yuechen Yu, Xiameng Qin, Chengquan Zhang, Yan Liu, Kun Yao, Junyu Han, Jingtuo Liu, and Errui Ding. 2021. Structext: Structured text understanding with multi-modal transformers. In *Proceedings of the 29th ACM International Conference on Multimedia*, MM '21, page 1912–1920. ACM.

Jerry Liu. 2022. LlamaIndex.

Chuwei Luo, Yufan Shen, Zhaoqing Zhu, Qi Zheng, Zhi Yu, and Cong Yao. 2024. Layoutllm: Layout instruction tuning with large language models for document understanding. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, page 15630–15640. IEEE.

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*, pages 6495–6504, Online. Association for Computational Linguistics.

Evan Miller. 2024. Adding error bars to evals: A statistical approach to language model evaluations.

Armineh Nourbakhsh, Sameena Shah, and Carolyn Rose. 2024. Towards a new research agenda for multimodal enterprise document understanding: What are we missing? In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14610–14622, Bangkok, Thailand. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 turbo. Accessed: December 2024.

OpenAI. 2024. *GPT-3.5 Turbo*. Accessed: 2024-12-05.

Rasmus Berg Palm, Ole Winther, and Florian Laws. 2017. Cloudscan - a configuration-free invoice analysis system using recurrent neural networks. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, page 406–413. IEEE.

Qiming Peng, Yinxu Pan, Wenjin Wang, Bin Luo, Zhenyu Zhang, Zhengjie Huang, Yuhui Cao, Weichong Yin, Yongfeng Chen, Yin Zhang, Shikun Feng, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2022. ERNIE-layout: Layout knowledge enhanced pretraining for visually-rich document understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3744–3756, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Vincent Perot, Kai Kang, Florian Luisier, Guolong Su, Xiaoyu Sun, Ramya Sree Boppana, Zilong Wang, Zifeng Wang, Jiaqi Mu, Hao Zhang, Chen-Yu Lee, and Nan Hua. 2024. LMDX: Language model-based document information extraction and localization. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 15140–15168, Bangkok, Thailand. Association for Computational Linguistics.

Rafał Powalski, Łukasz Borchmann, Dawid Jurkiewicz, Tomasz Dwojak, Michał Pietruszka, and Gabriela Pałka. 2021. *Going Full-TILT Boogie on Document Understanding with Text-Image-Layout Transformer*, page 732–747. Springer International Publishing.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Clément Sage, Alexandre Aussem, Haytham Elghazel, Véronique Eglin, and Jérémy Espinas. 2019. Recurrent neural network approach for table field extraction in business documents. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1308–1313.

Dengliang Shi, Siliang Liu, Jintao Du, and Huijia Zhu. 2023. Layoutgcn: A lightweight architecture for visually rich document understanding. In *IEEE International Conference on Document Analysis and Recognition*.

Zineng Tang, Ziyi Yang, Guoxin Wang, Yuwei Fang, Yang Liu, Chenguang Zhu, Michael Zeng, Cha Zhang, and Mohit Bansal. 2023. Unifying vision, text, and layout for universal document processing. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, page 19254–19264. IEEE.

Yi Tu, Ya Guo, Huan Chen, and Jinyang Tang. 2023. Layoutmask: Enhance text-layout interaction in multi-modal pre-training for document understanding. In *Annual Meeting of the Association for Computational Linguistics*.

Dongsheng Wang, Natraj Raman, Mathieu Sibue, Zhiqiang Ma, Petr Babkin, Simerjot Kaur, Yulong Pei, Armineh Nourbakhsh, and Xiaomo Liu. 2023a. Docllm: A layout-aware generative language model for multimodal document understanding. *Preprint*, arXiv:2401.00908.

Jiapeng Wang, Lianwen Jin, and Kai Ding. 2022. Lilt: A simple yet effective language-independent layout transformer for structured document understanding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, page 7747–7757. Association for Computational Linguistics.

Zilong Wang, Yichao Zhou, Wei Wei, Chen-Yu Lee, and Sandeep Tata. 2023b. Vrdu: A benchmark for visually-rich document understanding. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23. ACM.

Mengxi Wei, Yifan He, and Qiong Zhang. 2020. Robust layout-aware ie for visually rich documents with pre-trained language models. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing. *Preprint*, arXiv:1910.03771.

David Wolpert. 1992. Stacked generalization. *Neural Networks*, 5:241–259.

Minghong Yao, Zhiguang Liu, Liansheng Zhuang, Liangwei Wang, and Houqiang Li. 2024. A robust framework for one-shot key information extraction via deep partial graph matching. *IEEE Transactions on Image Processing*, 33:1070–1079.

Wenwen Yu, Ning Lu, Xianbiao Qi, Ping Gong, and Rong Xiao. 2020. Pick: Processing key information extraction from documents using improved graph learning-convolutional networks. *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 4363–4370.

Yuechen Yu, Yulin Li, Chengquan Zhang, Xiaoqiang Zhang, Zengyuan Guo, Xiameng Qin, Kun Yao, Junyu Han, Errui Ding, and Jingdong Wang. 2023. Structextv2: Masked visual-textual prediction for document image pre-training. In *ICLR*. OpenReview.net.

Peng Zhang, Yunlu Xu, Zhanzhan Cheng, Shiliang Pu, Jing Lu, Liang Qiao, Yi Niu, and Fei Wu. 2020. Trie: End-to-end text reading and information extraction for document understanding. *Proceedings of the 28th ACM International Conference on Multimedia*.

Xiaohui Zhao, Zhuo Wu, and Xiaoguang Wang. 2019. CUTIE: learning to understand documents with convolutional universal text information extractor. *ICDAR*, abs/1903.12363.

A Similarity Score Calculation

To match field mentions to fields, a similarity score, displayed in Algorithm 1, is calculated based on their string representations.

Algorithm 1 Linking Function

Require: token_str, field_str
Ensure: link

function `calculate_link(token_str, field_str)`
if token_str.isnumeric() and token_str = field_str **then**
  return 1.0
else if ¬token_str.isnumeric() **then**
  sim ← jaccard_similarity(token_str, field_str)
  if sim > 0.7 **then**
    return sim
  else
    return 0.0
  end if
end if
end function

function `jaccard_similarity(text1, text2)`
if len(text1) = 0 or len(text2) = 0 **then**
  return 0
end if
if len(text1) < 4 and len(text2) < 4 **then**
  if text1 = text2 **then**
    return 1.0
  else
    return 0.0
  end if
end if
$n1 \leftarrow$ ngrams(text1, 3)
$n2 \leftarrow$ ngrams(text2, 3)
jsim $\leftarrow 1 -$ jaccard_distance(set($n1$), set($n2$))
jsim $\leftarrow \max(0, \text{jsim} - 0.1 \times |\text{len(text1)} - \text{len(text2)}|)$
return jsim
end function

B Baseline Training Details

Stacked Ensemble: The stacked ensemble model utilizes the Logistic Regression implementation from `sklearn`[1] and during training, hyperparameters are tuned with a grid search over the following hyperparameters: {'C': [0.01, 0.1, 1, 10, 100, 1000], 'penalty': ['l1', 'l2'], 'solver':['liblinear', 'saga']}. For each field candidate, we construct an input vector comprising features that include prediction scores for each field type and model. Field mentions detected by LayoutLMv3 and LiLT are mapped to field candidates during preprocessing using the similarity algorithm detailed in Appendix 1. For the GPT-3 model, prediction scores are binary (0,1), whereas LiLT and LayoutLMv3 provide predicted probabilities for each field candidate. Additionally, LLM judge scores for factual and format correctness per field type are incorporated as features.

[1] `https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.LogisticRegression.html`

Token Sequence Tagging Task LiLT and LayoutLMv3 are fine-tuned on a downstream sequence tagging task with a classification head for token classification, utilizing the BILOU schema. However, certain fields span multiple token sequences within the document, making them incompatible with the BILOU schema. To address this, we introduce an additional label, `LABELNAME_ADD`, for subsequent field sequences following the first. In the DeepForm VRDU dataset, this applies to "tv_address" and "product" fields. During training, the models predict these `_ADD` labels, and in postprocessing, scattered fields are reconstructed by linking each `_ADD`-labeled sequence to its corresponding sequence labeled without `_ADD`.

LiLT: The LiLT model was trained and applied using the `LayoutLMv3TokenizerFast`, `AutoModelForTokenClassification`, and `Trainer` classes from the Transformers library (Wolf et al., 2019)`https://huggingface.co/docs/transformers/index`.

We utilized the pretrained tokenizer and model `SCUT-DLVCLab/lilt-roberta-en-base`, provided by the authors (Wang et al.,

2022) on Hugging Face[2]. The tokenizer was configured with the following settings: `truncation=True`, `stride=128`, `padding="max_length"`, `max_length=512`, `return_overflowing_tokens=True`, and `return_offsets_mapping=True`.

The model was fine-tuned using the `AutoModelForTokenClassification` class and the `Trainer`, with the following hyperparameters:

- Learning rate: 5×10^{-6}
- Batch size: 8
- Gradient accumulation steps: 4
- Maximum steps: 9000
- Metric for model selection: `overall_f1`
- Warmup ratio: 0.1

The experiments were conducted on a system equipped with dual Intel Xeon Gold 6226R CPUs (64 cores, 128 threads), 754 GB of RAM, and two NVIDIA Tesla V100 32GB GPUs. Each experiment utilized a single GPU and required approximately 20 hours to complete.

LayoutLMv3: The LayoutLMv3 model was trained and applied using the `AutoProcessor`, `LayoutLMv3ForTokenClassification`, and `Trainer` classes from the Transformers library.

We utilized the pretrained tokenizer and model `microsoft/layoutlmv3-base`, provided by the authors (Huang et al., 2022) on Hugging Face[3]. The processor was loaded and applied using the `AutoProcessor` class with the setting `apply_ocr=False`.

The model was fine-tuned using the `LayoutLMv3ForTokenClassification` class and the `Trainer` with the following hyperparameters:

- Metric for model selection: `overall_f1`
- Warmup ratio: 0.1
- Learning rate: 5×10^{-6}
- Batch size: 8
- Gradient accumulation steps: 4
- Maximum steps: 9000

The experiments were conducted on a system equipped with dual Intel Xeon Gold 6226R CPUs (64 cores, 128 threads), 754 GB of RAM, and two NVIDIA Tesla V100 32GB GPUs. Each experiment utilized a single GPU and required approximately 20 hours to complete.

GPT-3.5: The GPT-3.5 Turbo model (OpenAI, 2024) is a decoder-based large language model (LLM) fine-tuned for the entity extraction task, following the tool-use approach introduced by Cesista et al. (2024). Unlike Cesista et al. (2024), we do not employ structured prompting to transform PDF content but instead use raw PDF text to minimize additional processing costs.

For supervised fine-tuning, we utilize the OpenAI plattform[4] to fine-tune the `gpt-3.5-turbo-0125` model. The `gpt-3.5-turbo-0125` model was chosen as it is more cost-effective than newer models while maintaining a knowledge cutoff in 2022, ensuring that the base model has not been exposed to the VRDU dataset.

The fine-tuning process follows the guidelines provided in the OpenAI Cookbook[5], and the resulting model generates valid JSON outputs. These outputs are parsed to extract entities for downstream tasks. Training costs amounted to approximately €210 ($221), with additional application costs for the test sets estimated at €75 ($79).

We implement retrieval-augmented generation using OpenAI's ChatCompletion API with Function Calling[6] in conjunction with the Python package `llama-index`(Liu, 2022). Fine-tuning was conducted following the instructions available in the OpenAI Cookbook[7]. The function definition, detailed in Listing 1, defines the data schema for invoice fields and is passed to the ChatCompletion API as a tool[8].

The model produces valid JSON outputs, from which entities are extracted for further processing.

[2] https://huggingface.co/SCUT-DLVCLab/lilt-roberta-en-base

[3] https://huggingface.co/microsoft/layoutlmv3-base

[4] https://platform.openai.com/finetune

[5] https://cookbook.openai.com/examples/chat_finetuning_data_prep

[6] https://platform.openai.com/docs/guides/function-calling

[7] https://cookbook.openai.com/examples/chat_finetuning_data_prep

[8] https://platform.openai.com/docs/api-reference/chat

The cost of training the model was approximately €210 ($221), and applying the model to the test sets incurred an additional cost of approximately €75 ($79).

The system prompt utilized for processing political advertisement invoice documents is shown in Listing 2. This prompt, combined with the function dictionary (Listing 1), is supplied to the OpenAI ChatCompletion API to facilitate fine-tuning and inference.

To align the assistant's responses with ground truth extractions from the VRDU dataset, these extractions are provided as examples during fine-tuning. The format of these examples is presented in Listing 3.

C PM3-KIE Implementation, Training and Inference Details

Model generation, training, and inference were based on the Statistical Relational Learning framework introduced by Bach et al. (2017). For our experiments, we utilized the `pslpython` package (version 2.4.0[9]), which wraps the Java-based PSL implementation[10].

Meta-model weight learning was performed on the development folds using the structured perceptron algorithm[11]. Inference was conducted using ADMM[12] as described by Bach et al. (2017).

Default hyperparameters were employed, with modifications to the regularization settings as follows:

- `gradientdescent.negativelogregularization = 0`
- `gradientdescent.negativeentropyregularization = 0.0001`

All experiments were executed on a CPU system equipped with 48 cores (Intel® Xeon® Gold 5118, 2.3 GHz base clock, 3.2 GHz max clock) and 376 GiB of RAM. Training the meta-model weights required approximately 20 minutes per model on average.

D Model Runtime Analysis

Table 5 presents the training and inference times of the two encoder-based baselines, LayoutLMv3 and LiLT, alongside our proposed PM3-KIE model. Inference time is reported in seconds per sample, while training time is reported in minutes.

Compared to the baseline models, PM3-KIE incurs minimal computational overhead. The meta-model requires only a few minutes to train and does not depend on GPU acceleration, as training was conducted entirely on a CPU. In contrast, both LayoutLMv3 and LiLT were trained using a single NVIDIA Tesla V100 GPU.

All experiments were conducted on a server equipped with dual Intel Xeon Gold 6226R CPUs (64 cores, 128 threads), 754 GB RAM, and two NVIDIA Tesla V100 32GB GPUs.

Model	Train Time (min)	Infer Time (sec)
Lilt	450	1.57
LMv3	1296	4.43
PM3KIE	4	0.60

Table 5: Training and inference times for each model.

E Detailed Evaluation Results for FARA and DeepForm datasets

Tables 6 and 7 summarize the ID and OOD results of all baselines and PM3-KIE on the FARA and DeepForm datasets. Tables 8 and 9 show F1 values per field and documemt as well as hit rate per document and the document-level accuracy for both FARA and DeepForm.

F Case Study and Error Analysis

To analyze model performance in detail, we conducted a case study using the DeepForm dataset (train size: 200, split: 2, unk_template) and the LayoutLMv3 baseline. We filtered instances where the baseline model produced incorrect predictions while PM3KIE generated the correct output.

Tables 11 and 12 categorize errors into **syntactic errors**, where predictions deviate from predefined formats, and **semantic errors**, where extracted entities are factually implausible. For each error case, we report the LLM-judge score—format-based scores for syntactic errors and factual consistency scores for semantic errors—along with the model's justification. Notably, the LLM-judge score consistently aligns with the severity of the formatting or factual errors.

Syntactic errors (Table 11) predominantly occur in address fields, where key components such as city names or postal codes are missing. Similar

[9] https://pypi.org/project/pslpython/

[10] https://github.com/linqs/psl

[11] org.linqs.psl.application.learning.weight.gradient .optimalvalue.StructuredPerceptron

[12] org.linqs.psl.application.inference.mpe.ADMMInference

Listing 1: Function Description for OpenAI Tool Use

```
function_dict = {
 'name': 'AdInvoice', 'description': 'Data model for invoice fields.', 'parameters':
  {'title': 'AdInvoice', 'description': 'Data model for invoice fields.', 'type': 'object',
   'properties': {
     'contract_num': {
       'title': 'Contract Number',
       'description': 'The invoice contract number or order number.',  'type': 'string'},
     'tv_address': {
       'title': 'Tv Address', 'description': 'Physical address of the tv channel.',
       'default': '', 'type': 'string'},
     'property': {
       'title': 'Property', 'description': 'Property, usually equivalent to tv channel name.',
       'default': '', 'type': 'string'},
     'agency': {
       'title': 'Agency', 'description': 'The advertisement agency.', 'default': '', 'type': 'string'},
     'advertiser': {'title': 'Advertiser', 'description': 'The advertiser.', 'type': 'string'},
     'flight_from': {
       'title': 'Flight From',
       'description': 'The order flight start date.',
       'default': '',
       'type': 'string'},
     'flight_to': {
       'title': 'Flight To',
       'description': 'The order flight end date.',
       'default': '',
       'type': 'string'},
     'product': {
       'title': 'Product',
       'description': 'The product that is advertised.',
       'default': '',
       'type': 'string'},
     'gross_amount': {
       'title': 'Gross Amount',
       'description': 'The total amount to be paid.',
       'type': 'string'},
     'line_items': {
       'title': 'Line Items',
       'description': 'List of line items.',
       'type': 'array',
       'items': {
         '$ref': '#/definitions/LineItem' }}},
   'required': ['contract_num', 'advertiser', 'line_items'],
   'definitions': {
     'LineItem': {
       'title': 'LineItem',
       'description': 'Data model for line item fields.',
       'type': 'object',
       'properties': {
         'channel': {
           'title': 'Channel',
           'description': 'Name of the tv channel broadcasting the advertisement.',
           'default': '',
           'type': 'string'},
         'program_start_date': {
           'title': 'Program Start Date',
           'description': 'Program start date (only date without timestamp).',
           'default': '',
           'type': 'string'},
         'program_end_date': {
           'title': 'Program End Date',
           'description': 'Program end date (only date without timestamp).',
           'default': '',
           'type': 'string'},
         'program_desc': {
           'title': 'Program Desc',
           'description': 'Description of the TV program.',
           'default': '',
           'type': 'string'},
         'sub_amount': {
           'title': 'Sub Amount',
           'description': 'Sub amount for one program ad.',
           'default': '',
           'type': 'string'}}}}}}
```

Listing 2: Formatted System Prompt

```
You are receiving content from a political advertisement invoice document.
This invoice is signed between a TV station and a campaign group.
The document uses tables, multi-columns, and key-value pairs to record the information.
Your task is to digitize these documents by extracting their information in a structured format.
Extract the unique header information, such as TV channel addresses and total costs,
along with the list of line items detailing specific ads, TV programs in which they will be broadcasted,
and sub-amounts.
Extract line item fields only from the tabular line item list in the invoice document
and not from the invoice header: if line item fields are not present in the line item list, don't extract them.
```

Listing 3: Ground Truth Assistant Answer provided for Fine-Tuning

```
{
  "role": "assistant",
  "function_call": {
    "name": "AdInvoice",
    "arguments":
     "{"contract_num":"668864 ",
       "tv_address":"PO Box 809229\\nChicago, IL 60680-9229\\n",
       "property":"WAXN-TV\\nWSOC Television, Inc.\\n",
       "agency":"",
       "advertiser":"POL/Donald Trump/R/PRES/US-A\\n",
       "flight_from":"03/05/20 ",
       "flight_to":"03/10/20\\n",
       "product":"TRUMP FOR PRESIDENT\\n",
       "gross_amount":"$3,920.00\\n",
       "line_items":[
        {"channel":"WAXN ","program_start_date":"03/09/20 ","program_end_date":"03/09/20 ",
        "program_desc":"M-F 7a-8a\\n","sub_amount":"$250.00\\n"},
        {"channel":"WAXN ","program_start_date":"03/06/20 ","program_end_date":"03/06/20 ",
        "program_desc":"M-F 7a-8a\\n","sub_amount":"$250.00\\n"},
        {"channel":"WAXN ","program_start_date":"03/09/20 ","program_end_date":"03/09/20 ",
        "program_desc":"M-F 8a-9a\\n","sub_amount":"$260.00\\n"},
        {"channel":"WAXN ","program_start_date":"03/06/20 ","program_end_date":"03/06/20 ",
        "program_desc":"M-F 8a-9a\\n","sub_amount":"$260.00\\n"},
        {"channel":"WAXN ","program_start_date":"03/06/20 ","program_end_date":"03/06/20 ",
        "program_desc":"M-F 8p-830p\\n","sub_amount":"$0.00\\n"},
        ....,}
    ]}"
  }
}
```

Table 6: DeepForm Performance metrics for various training set sizes (Train) and distribution folds (Dist). The reported F1 scores are F1-scores averaged across all fields and are accompanied by their corresponding standard errors in parentheses. Additionally, the paired difference between our model and the best-performing model is presented, along with the 95% confidence interval enclosed in parentheses.

Dist	#Train	#Test	F1 LILT (std. error)	F1 GPT (std. error)	F1 LMv3 (std. error)	F1 STE (std. error)	F1 Best (std. error)	F1 PM3-KIE (std. error)	Paired Diff (conf. interval)
id	10	6	87.36 (4.49)	87.92 (5.14)	80.97 (4.32)	86.84 (4.80)	90.09 (2.65)	**93.31** (2.31)	**+3.24** (+3.09, +3.4)
id	50	64	91.55 (2.36)	87.92 (2.25)	82.30 (3.12)	85.36 (2.08)	91.55 (2.36)	**92.19** (1.82)	**+1.71** (+0.71, +2.7)
id	100	89	93.85 (1.66)	92.41 (1.59)	87.72 (2.11)	88.63 (1.48)	93.85 (1.66)	**94.36** (1.38)	**+1.10** (+0.06, +2.14)
id	200	144	94.21 (1.20)	91.99 (1.64)	90.06 (1.19)	87.21 (1.16)	94.21 (1.20)	**95.08** (1.01)	**+1.23** (+0.23, +2.23)
ood	10	294	64.89 (1.50)	78.00 (1.48)	58.11 (1.60)	72.13 (1.15)	78.00 (1.48)	**82.19** (1.15)	**+5.63** (+1.46, +9.8)
ood	50	236	89.86 (0.97)	85.65 (1.42)	83.46 (1.21)	84.53 (1.07)	89.86 (0.97)	**92.36** (0.78)	**+2.87** (+0.78, +4.96)
ood	100	211	91.10 (1.07)	86.76 (1.21)	83.66 (1.28)	84.01 (1.04)	91.10 (1.07)	**92.95** (0.86)	**+2.33** (+0.67, +4.0)
ood	200	156	93.11 (0.99)	90.19 (1.46)	85.30 (1.41)	88.40 (1.06)	93.11 (0.99)	**94.30** (0.94)	**+1.21** (-0.25, +2.68)

issues are observed in date fields, where predictions are often incomplete or improperly formatted.

Semantic errors (Table 12) are more varied. The most common confusions arise between **advertiser**, **property**, and **tv-company** labels. For instance, in Table 12, "Univision Receivables Co" is misclassified, but the LLM-judge assigns a low factual consistency score, correctly identifying that this entity is not a known broadcasting company. Similarly, **product** extractions frequently contain

Table 7: FARA Performance metrics for various training set sizes (Train) and distribution folds (Dist). The reported F1 scores are F1-scores averaged across all fields and are accompanied by their corresponding standard errors in parentheses. Additionally, the paired difference between our model and the best-performing model is presented, along with the 95% confidence interval enclosed in parentheses.

Dist	#Train	#Test	F1 LILT (std. error)	F1 GPT (std. error)	F1 LMv3 (std. error)	F1 STE (std. error)	F1 Best (std. error)	F1 PM3-KIE (std. error)	Paired Diff (conf. interval)
id	10	300	72.13	90.39	49.71	90.84	90.87	**91.90**	**+1.56**
			(1.45)	(0.91)	(1.53)	(0.89)	(0.86)	(0.84)	(+0.6, +2.52)
id	50	300	85.83	93.13	70.04	91.82	93.13	**94.58**	**+1.68**
			(1.16)	(0.85)	(1.49)	(0.82)	(0.85)	(0.7)	(+0.8, +2.56)
id	100	300	89.02	93.55	76.84	93.07	93.79	**95.56**	**+2.03**
			(1.03)	(0.8)	(1.47)	(0.8)	(0.75)	(0.64)	(+1.18, +2.89)
id	200	300	89.98	94.50	81.23	94.64	94.86	**95.86**	**+1.26**
			(0.97)	(0.69)	(1.28)	(0.67)	(0.66)	(0.6)	(+0.53, +1.98)
ood	10	300	60.18	91.34	38.66	90.87	91.64	**92.40**	**+0.96**
			(1.37)	(0.86)	(1.19)	(0.87)	(0.85)	(0.81)	(+0.21, +1.72)
ood	50	300	75.04	94.16	46.87	93.37	94.16	**94.74**	**+0.77**
			(1.37)	(0.81)	(1.23)	(0.8)	(0.81)	(0.72)	(+0.03, +1.58)
ood	100	300	78.60	94.53	49.21	94.49	94.77	**95.50**	**+0.84**
			(1.22)	(0.74)	(1.19)	(0.69)	(0.7)	(0.63)	(+0.18, +1.50)
ood	200	300	78.05	95.46	53.65	95.82	95.82	**96.19**	**+0.41**
			(1.24)	(0.62)	(1.27)	(0.59)	(0.59)	(0.56)	(+0.1, +0.91)

Table 8: FARA dataset results: Mean F1-scores, Hit-rate and Accuracy per Field and per Document are averaged over 24 models

Model	F1 per Field	Hit per Doc	F1 per Doc	Doc Lv. Acc.
LILT	78.60	83.46	77.82	42.40
LMv3	58.27	67.27	57.39	15.88
GPT3.5	93.38	96.12	93.15	81.60
Stacked	93.12	93.52	92.48	71.11
Best Model	93.63	95.45	93.28	78.97
PM3-KIE	**94.59**	**96.50**	**94.47**	**82.92**

Table 9: DeepForm dataset results: Mean F1-scores, Hit-rate and Accuracy per Field and per Document are averaged over 24 models

Model	F1 per Field	Hit per Doc	F1 per Doc	Doc Lv. Acc.
LILT	88.24	87.40	86.85	48.84
LMv3	81.45	80.10	79.48	24.69
GPT3.5	87.60	88.81	86.28	53.35
Stacked	84.64	79.15	83.13	19.03
Best Model	90.22	89.86	88.83	53.55
PM3-KIE	**92.47**	**92.28**	**91.67**	**62.38**

semantic errors, where generic terms (e.g., "Estimate #") or unrelated names (e.g., "BRABENDER, LIZ") are extracted instead of valid campaign-related products. In all cases, the LLM-judge not only flags these issues with a low score but also provides a plausible explanation for the misclassification.

Table 10: Ablation study results on FARA. Difference in F1 for the basic PM3-KIE model compared to the adapted model version.

Ablation	F1 Field	Hit Doc	F1 Doc	Acc
PM3-KIE	**94.59**	**96.50**	**94.47**	**82.92**
+ Token Task	-0.05	-0.07	-0.06	-0.39
- Constraints	-0.14	-0.26	-0.22	-1.09
- fact & format judges	-0.39	-0.53	-0.46	-2.35

Document ID	Label	Correct PM3KIE Pred.	Wrong LXML Pred.	Score	Reason
462499-wbz-nrcc-richard-tisei-housemacd6-ord58654-fed.pdf	agency	MENTZER MEDIA SERVICES	1170 SOLDIERS FIELD ROAD,	0.0	'Appears to be an address, not an agency.'
463974-kovr-nrcc-r-multi-ord33720-fed-natl-contract.pdf	advertiser	NATIONAL REPUBLICAN CONGRESSIONAL COMM.	NATIONAL MEDIA RESEARCH PLANNING	0.3	'Focuses on media research, not direct advertising.'
493496-315597-up-obama-10-30-12-13516185156783-_pdf.pdf	product	OFA Sked B 10.23-11.6	Estimate	0.0	'"Estimate " is too generic to be a campaign product.'
94007-103012-murphy-invoice-3-13516266235434-_pdf.pdf	product	Political Federal Candidate (1067)	BRABENDER, LIZ	0.2	'A personal name, not a campaign product.'
6a7ef626-8d94-b4bf-0a92-26356a8fef15.pdf	property	KABE	Univision Receivables Co LLC	0.2	'Appears to be a financial entity, not a media property.'
6c195cac-5318-ff53-9699-5da7026ba63e.pdf	agency	Waterfront Strategies	Strategies	0.6	'"Strategies" is too generic to confirm as an agency.'
a60f1b87-bc56-a8e2-5899-f33824e0618a.pdf	advertiser	MIKE BLOOMBERG 2020 INC	The Valley's Choice for Local News!	0.2	'More like a news outlet than an advertiser.'
aa7e85a5-922d-a0fe-1ae7-167e6f839d5c.pdf	product	Issue	13	0.0	'"13" lacks context to be a campaign product.'
36cdd72e-8e53-2f4d-7aaa-e646a0a87acc.pdf	advertiser	Knute for Congress	Congress	0.2	'"Congress" is too vague for an advertiser.'

Table 11: Semantic errors: Shows errors made by the LXML model, along with the factual LLM-judge score for the incorrect prediction and the correct PM3KIE prediction.

Document ID	Label	Correct PM3KIE Pred.	Wrong LXML Pred.	Score	Reason
494007-103012-murphy-invoice-3-13516266235434-_pdf.pdf	tv_address	P.O.Box 640132 Pittsburgh, PA 15264-0132	P.O.Box 640132	0.3	'Missing city, postal code, and country.'
6d885030-9db6-64c0-1601-3e0ce1525b6c.pdf	tv_address	1940 Zanker Road San Jose, CA 95112	1940 Zanker Road	0.5	'City and postal code are missing.'
e0f80dc6-ea62-babb-44b7-9b71546f2618.pdf	tv_address	1414 N. Memorial Parkway Huntsville, AL 35801	1414 N. Memorial Parkway	0.6	'Postal code is missing.'
109d2451-a80d-fdab-9f98-e0bdf2d42895.pdf	tv_address	PO Box 1212 Augusta, GA 30903	PO Box 1212	0.3	'Incomplete without city and postal code.'
078b984c-7349-8b2c-eac4-32517a7ac1be.pdf	product	Candidate	(WV) IC-410	0.1	'Not clearly a campaign product.'
130d63b8-d072-a734-189a-807e0500a7d0.pdf	tv_address	1960 Union Avenue Memphis, TN 38104	1960 Union Avenue	0.5	'Postal code is missing.'
15262983-7227-5abf-803b-1089dfad4494.pdf	tv_address	PO Box 206270 Dallas, TX 75320-6270	PO Box 206270	0.2	'Missing city and postal code.'
1f279e59-7876-b9e2-5a37-73c702d12541.pdf	tv_address	524 West 57th Street New York, NY 10019	524 West 57th Street New York, NY	0.6	'Postal code is missing.'
23a5de85-3f4d-e735-92c2-a029e4121878.pdf	tv_address	7616 Channel 16 Way Jackson, MS 39209	7616 Channel 16 Way Jackson, MS	0.5	'Postal code is missing.'
271ea37f-e7a4-3802-d855-8784c3cb74d3.pdf	tv_address	4625 S. 33rd Place Phoenix, AZ 85040	4625 S. 33rd Place	0.5	'City and postal code are missing.'
34e3acdc-2c6b-1d50-ebc1-973d2ff6ca09.pdf	tv_address	1965 S 4th Ave Yuma, AZ 85364	1965 S 4th Ave	0.4	'Missing city and postal code.'
3bbc5269-ccf6-3f6b-6aaf-1c61a646535c.pdf	tv_address	PO Box 206270 Dallas, TX 75320-6270	PO Box 206270	0.2	'Missing city and postal code.'
416294-collect-files-52527-political-file-2012-non.pdf	tv_address	1100 Fairfield Dr West Palm Beach, FL 33407	1100 Fairfield Dr	0.5	'Postal code is missing.'
44ce4266-07d0-0602-4c2c-9077117bea05.pdf	tv_address	285 N. Foster St. Dothan, AL 36303	285 N. Foster St.	0.5	'City and postal code are missing.'
493546-warren-for-senate-est-12911-10-26-12-10-28-12.pdf	tv_address	7 Bulfinch Place Boston, MA 02114-2977	7	0.0	'Incomplete, missing essential components.'
50e20ed6-1232-4272-a28a-4632d68679fc.pdf	tv_address	1090 KNUTSON AVENUE MEDFORD, OR 97504	1090 KNUTSON AVENUE MEDFORD, OR	0.7	'Postal code is missing.'
6a7ef626-8d94-b4bf-0a92-26356a8fef15.pdf	tv_address	5801 Truxtun Avenue Bakersfield, CA 93309	5801 Truxtun Avenue	0.5	'City and postal code are missing.'
dbd3ea93-a44a-4c42-df97-c4cfb6e33453.pdf	tv_address	Lockbox 742923 Atlanta, GA 30374-2923	30374-2923	0.2	'Appears to be a postal code, not an address.'
8dfa0ca9-2457-2309-3c1c-8301ca6a6d14.pdf	flight_to	/09/20	/09/	0.0	'Incomplete date format.'

Table 12: Semantic errors: Shows errors made by the LXML model, along with the factual LLM-judge score for the incorrect prediction and the correct PM3KIE prediction.

G Additional Software and Licenses

Table 13 lists all first-level import python packages used to perform the experiments described in this work.

Table 13: Software Packages and Their Licenses

Name	Version	License
python	3.10.14	PSF License
editdistance	3.10.14	MIT License
datasets	2.19.0	Apache Software License
pandas	2.2.3	BSD License
scikit-learn	1.5.1	BSD License
tqdm	4.66.4	MIT License; MPL 2.0
pydantic	2.8.2	MIT License
pydantic_core	2.20.1	MIT License
transformers	4.40.1	Apache Software License
numpy	1.26.4	BSD License
torch	2.3.0	BSD License
pillow	10.3.0	HPND
llama-index	0.10.52	MIT License
tiktoken	0.7.0	MIT License
nltk	3.8.1	Apache Software License
pdf2image	1.17.0	MIT License
dataclasses-json	0.6.7	MIT License
mlflow	2.12.2	Apache Software License
plotly	5.22.0	MIT License
matplotlib	3.9.0	Python SF License
scipy	1.13.0	BSD License

H Dataset

H.1 VRDU Dataset and Evaluation

We utilized the VRDU dataset with the two corpora FARA and DeepForm and an evaluation framework implementation provided by Wang et al. (2023b).[13].

H.2 Annotation and Evaluation Corrections

To reduce spurious matching errors during evaluation, we employed the field-specific matching functions available for the dataset[14] to normalize values, such as standardizing date formats. In addition to the existing functions, we introduced three additional matching strategies to address inconsistencies in the dataset:

- **GeneralCaseInsensitiveStringMatch**: Strings are considered equivalent if their lowercase representations match.
- **IgnorePropertySuffixStringMatch**: Accounts for inconsistent annotations of properties (e.g., with or without the suffix *"remit to"*). Strings are matched after removing the phrase *"remit to"* and redundant whitespace.
- **IgnoreLeadingTrailingNumbersStringMatch**: Handles inconsistent annotations for products and agencies where numeric prefixes or suffixes (including those in brackets) may or may not be present. Strings are matched after removing leading or trailing numeric sequences and redundant whitespace.

Several annotations in the VRDU dataset were incorrect, incomplete, or missing, particularly for dates and address elements. For instance, dates were often missing complete year information, and address components were inconsistently annotated. Table 14 lists the corrections we introduced. Both the corrected annotations and the original dataset annotations were considered valid during evaluation.

I LLM as a Judge

We use the `gpt4o mini` model (OpenAI, 2023), chosen for its cost-effectiveness and reliable performance. For both LLMs, we designed one prompt per field type. These prompts were automatically generated for each field type using the following prompt templates for format checking (see Listing 4) and for fact checking (see Listing 5), requiring only minor adjustments afterward.

[13] https://github.com/google-research-datasets/vrdu

[14] https://github.com/google-research/google-research/tree/master/vrdu

Listing 4: Meta-Prompt for Generating Format Correctness Prompts

```
I have a corpus of invoices for political advertisement from tv channels, where information should be extracted.
The following set of information should be extracted:
---
advertiser
agency
channel
contract_num
flight_from
flight_to
gross_amount
product
program_desc
program_end_date
program_start_date
property
sub_amount
tv_address
---
I want to verify the format of each of these information using a prompt.
Please generate a prompt for every information in the style of this prompt for the tv-adress and
add it to this json-file:

format_correctness_prompts = {
    "tv_address": "You receive addresses that require validation. For each address:

    Verify Address Completeness:
    Ensure the address includes all essential components: street, city, postal code, and country.
    Check for Consistency: Identify any inconsistencies, errors,
    or missing elements within the address components.

    Confidence Score:
    Based on your evaluation, provide a confidence score from 0 to 1,
    reflecting how likely it is that the address is in a correct and complete format.

    Output Format (CSV):
    score; justification
    <numerical score (0 to 1)>; '<Short explanation of the score, highlighting
    specific aspects of the address that support or detract from its
    completeness and correctness.'>"}
```

Table 14: List of Documents with the field types and corrected values, that are added to the VRDU ground truth for evaluation

Document	Field Type	Field Value
b0ae0954-274a-f270-797c-76224b78b8ee.pdf	agency	Del Ray Media
89b8c007-4189-bfa6-e0a5-fe1d173edf92.pdf	flight_from	05/27/20
89b8c007-4189-bfa6-e0a5-fe1d173edf92.pdf	flight_to	05/31/20
42adf390-6e50-6fbc-fbbe-65117a1ffcb2.pdf	gross_amount	$500.00
143af697-c6f9-a36e-d43f-1a92e800ffeb.pdf	flight_from	07/01/20
143af697-c6f9-a36e-d43f-1a92e800ffeb.pdf	flight_to	07/07/20
14632210-11d9-a184-e3db-b1b219f52ca8.pdf	flight_from	06/02/20
14632210-11d9-a184-e3db-b1b219f52ca8.pdf	flight_to	06/08/20
48845b9d-9e1b-a9e8-d560-58d35d2b31b2.pdf	flight_from	01/01/20
48845b9d-9e1b-a9e8-d560-58d35d2b31b2.pdf	flight_to	01/08/20
45f3875f-2b24-42fe-ddb4-fa203f4eec30.pdf	flight_from	01/22/20
45f3875f-2b24-42fe-ddb4-fa203f4eec30.pdf	flight_to	02/05/20
4cc700a3-6cb8-b791-2428-890e7fb7cf2a.pdf	flight_from	10/06/20
4cc700a3-6cb8-b791-2428-890e7fb7cf2a.pdf	flight_to	10/12/20
64243566-745a-3edd-224b-542129a844a6.pdf	flight_from	Apr16/20
64243566-745a-3edd-224b-542129a844a6.pdf	flight_to	Apr22/20
64243566-745a-3edd-224b-542129a844a6.pdf	product	POLITICIAL
7b9c8208-d2be-2a81-8a15-215a9a5a26e8.pdf	flight_from	Feb15/20
7b9c8208-d2be-2a81-8a15-215a9a5a26e8.pdf	flight_to	Feb21/20
7b9c8208-d2be-2a81-8a15-215a9a5a26e8.pdf	product	BLOOMBERG 4 PRES
dbd4ed2d-11f1-ba35-cc98-43127897504a.pdf	flight_from	Jun05/20
dbd4ed2d-11f1-ba35-cc98-43127897504a.pdf	flight_to	Jun19/20
dbd4ed2d-11f1-ba35-cc98-43127897504a.pdf	product	OWENS FOR CON UT04
0be55a7b-c4b9-7956-d523-30f79a4ebc1a.pdf	flight_from	1/27/2020
0be55a7b-c4b9-7956-d523-30f79a4ebc1a.pdf	flight_to	2/23/2020
4b330586-f3e8-28ea-b0cc-2d060dc10622.pdf	flight_from	4/27/2020
4b330586-f3e8-28ea-b0cc-2d060dc10622.pdf	flight_to	5/31/2020
38a1ec3a-18bd-0b73-1155-b6ced503f7a1.pdf	flight_from	1/27/2020
38a1ec3a-18bd-0b73-1155-b6ced503f7a1.pdf	flight_to	2/23/2020
c1ede720-d1f9-dcb4-e56f-65bf46300e84.pdf	flight_from	02/11/20
c1ede720-d1f9-dcb4-e56f-65bf46300e84.pdf	flight_to	02/17/20
cda5811d-3cf3-9c50-0941-28094bf9880f.pdf	flight_from	01/01/20
cda5811d-3cf3-9c50-0941-28094bf9880f.pdf	flight_to	01/08/20
b009ea0d-d54e-7410-320d-2dc99dbc8c09.pdf	tv_address	PO BOX 206270 Dallas, TX 75320-6270
b009ea0d-d54e-7410-320d-2dc99dbc8c09.pdf	flight_from	2/1/2020
b009ea0d-d54e-7410-320d-2dc99dbc8c09.pdf	flight_to	2/29/2020
a5a37afc-bbf5-db26-bd19-a71fee1ae67a.pdf	flight_from	05/06/20
88fa6e33-408f-6ac2-a253-d30e32bce302.pdf	flight_from	03/31/20
88fa6e33-408f-6ac2-a253-d30e32bce302.pdf	flight_to	04/05/20
80ff3aa4-3617-496e-fc29-cf9fdbecc54d.pdf	flight_from	04/29/20
80ff3aa4-3617-496e-fc29-cf9fdbecc54d.pdf	flight_to	05/05/20
65ebbb18-8a01-357a-94ce-bfa16723822e.pdf	flight_from	06/09/20
65ebbb18-8a01-357a-94ce-bfa16723822e.pdf	flight_to	06/15/20
64780ed0-180c-a77b-bfe0-478c2a48a3a0.pdf	flight_from	05/19/20
64780ed0-180c-a77b-bfe0-478c2a48a3a0.pdf	flight_to	05/22/20
73badb45-b62c-0c2e-1a2f-4b5fb4fda5b9.pdf	tv_address	6301 Bandel Road NW ROCHESTER Rochester, MN 55901-8798
73badb45-b62c-0c2e-1a2f-4b5fb4fda5b9.pdf	flight_from	10/06/20
73badb45-b62c-0c2e-1a2f-4b5fb4fda5b9.pdf	flight_to	10/12/20
685a2568-66ec-e3e0-27a1-78b56402b8cb.pdf	flight_from	09/15/20
685a2568-66ec-e3e0-27a1-78b56402b8cb.pdf	flight_to	09/21/20
2437f486-c8f2-72ad-1c75-4d45eedd57d2.pdf	flight_from	05/05/20
2437f486-c8f2-72ad-1c75-4d45eedd57d2.pdf	flight_to	05/05/20
c1f3f40f-9003-6d17-9d92-f7d62836f017.pdf	flight_from	04/06/20
c1f3f40f-9003-6d17-9d92-f7d62836f017.pdf	flight_to	04/13/20
35b3207f-bd16-d173-09a5-570acac710b2.pdf	flight_from	03/30/20
35b3207f-bd16-d173-09a5-570acac710b2.pdf	flight_to	04/06/20

Listing 5: Meta-Prompt for Generating Factual Correctness Prompts with JSON Output

```
I have a corpus of invoices for political advertisements from TV channels,
where information should be extracted. The following set of information
should be extracted:
---
advertiser
agency
channel
contract_num
flight_from
flight_to
gross_amount
product
program_desc
program_end_date
program_start_date
property
sub_amount
tv_address
---
I want to verify the factual correctness of each of these information using
a prompt. Please generate a prompt for every information in the style of
this prompt for the tv-address and add it to this JSON file:

factual_correctness_prompts = {
    "tv_address": "You will evaluate addresses to determine if they are
    likely to be the official locations of a TV channel or broadcasting
    company. For each address:

    Assess Suitability: Evaluate whether the address could realistically
    serve as a media or broadcasting location. Consider factors such as
    the presence of corporate offices, proximity to media hubs, or known
    broadcasting facilities that would support its use as a TV channel address.

    Provide a Confidence Score: Based on this assessment, assign a confidence
    score from 0 to 1, reflecting how likely it is that the string is a valid
    location for a TV channel.

    Output Format (CSV):
    score; justification
    <numerical score (0 to 1)>; '<Short explanation of the score, highlighting
    specific aspects of the address that support or detract from its
    completeness and correctness.'>"
}
```