Language agnostic document key value extraction using Logical Neural Networks

Simone Giovannini[1,*], Fabio Coppini[2] and Simone Marinai[1]

[1]DINFO, Università degli Studi di Firenze, Via di Santa Marta 3, Florence, 50139, Italy
[2]Mathematical Institute, Utrecht University, Budapestlaan 6, 3584 CD Utrecht, The Netherlands & Letxbe, Paris, France

Abstract

Humans exhibit the ability to understand documents—such as forms, tables, and key-value relations—by relying only on the document structure, largely independently of the underlying language. In this work, we investigate whether this language-invariant structural reasoning can be formalized as a logical learning problem. Specifically, we use Logical Neural Networks to learn first-order logical rules over a predicate space derived exclusively from layout and geometric properties of document textual fields, without access to textual content. We evaluate our approach on XFUND, a multilingual form understanding benchmark spanning seven languages, and study two complementary tasks. For *key-value pair prediction*, layout-only predicates prove sufficient to learn accurate, interpretable rules that are strikingly consistent across languages, providing strong evidence for language-invariant logical structure in forms. For *key field classification*, performance is lower and less stable, suggesting that unary field roles are less directly decodable from layout alone. Together, our findings indicate that relational document reasoning can be partially grounded in language-agnostic logical patterns, opening new perspectives for interpretable and low-resource document understanding.

Keywords

Document AI, Logical Neural Networks, Inductive Logic Programming, Multilingual Document Understanding, Neuro-symbolic Learning

1. Introduction

Form understanding is a central problem in modern machine learning, with applications ranging from information extraction to visual question answering. All these tasks involve a strong structural component to be accomplished. For instance, identifying entity relations such as key-value pairs, extracting totals from invoices, or interpreting tables rely primarily on spatial arrangement, alignment, and formatting cues rather than purely lexical content [1, 2]. This observation suggests that part of document reasoning may be inherently language-independent. This hypothesis is supported by cognitive evidence: humans are able to interpret the structure of documents written in unknown languages by relying on layout invariants such as alignment, grouping, and visual hierarchy. Document understanding therefore involves not only linguistic processing, but also higher-level relational reasoning over structured visual inputs.

Recent advances in document AI have been driven by large-scale multimodal architectures inspired by BERT of which the LayoutLM family [3] is widely adopted, and by recent general-purpose multimodal LLMs (e.g. Qwen [4]). All these architecture combine textual, visual, and spatial features to achieve state-of-the-art performance on benchmarks such as FUNSD [5] and XFUND [6]. Despite their effectiveness, these approaches rely heavily on language-specific embeddings, large annotated datasets, and high computational power, limiting their applicability in multilingual or low-resource settings. In addition, the learned representation is fixed at pretraining and cannot be easily updated without retraining. They are also inherently opaque, as the lack of interpretability makes them unsuitable for application contexts requiring an understanding of the rationale behind their predictions.

In this work we are interested in the following research questions:

BEWARE-26 (Bias, Risk, Explainability and the role of Logic and Logic Programming), 8 October 2026, Perugia, Italy, co-located with AIxIA 2026

*Corresponding author.
simone.giovannini1@unifi.it (S. Giovannini); f.coppini@uu.nl (F. Coppini); simone.marinai@unifi.it (S. Marinai)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-11-17

CEUR-WS.org/Vol-4332/paper8.pdf

1. Can key-value extraction tasks be learned using layout features alone, without any access to textual or semantic content?
2. Can such tasks be formalized as fully explainable first-order logical rules over visual and spatial predicates?
3. If visual logical patterns suitably describe document structure, are they language-independent—i.e., do different languages yield consistent logical rules when the predicate space is grounded in layout only—and, if such logical rules are learned on documents in a given language, do they generalize to other languages without further retraining?

In response to these questions, this work frames form understanding as a logical learning problem and defines two tasks grounded in the XFUND benchmark. For completeness, it is worth noting that the first question has already been partially investigated in prior work on XFUND[2]. However, that approach remains largely opaque, as the symbolic component is not fully integrated into the learning process but is instead applied as a post-processing step on top of a neural model based on Graph Neural Networks.

The first task, *key-value prediction*, takes a pair of form fields as input and determines whether one acts as the key and the other as the corresponding value (e.g., `NAME: Simon`). The second, *key field classification*, takes a single field as input and determines whether it plays the role of a key in the form. To address these tasks, we define two families of layout predicates, binary predicates capturing relational properties between pairs of fields, and unary predicates capturing individual field properties such as size, position, and shape. Their truth values are computed directly from the document layout, with no access to textual content. We then instantiate Logical Neural Networks (LNN) [7], over these predicate spaces a recent neuro-symbolic approach capable of tackling inductive logic programming tasks [8], and train them independently on each language in XFUND. The experiments assess both whether the tasks are solvable from layout alone and whether the resulting rules are consistent across languages. A systematic analysis of the learned rules and their cross-lingual behaviour concludes the paper.

2. Motivation

This research is driven by the gap between current Document AI systems and human capabilities in document understanding. Modern deep learning approaches achieve high performance by leveraging computational extensive pre-trainings on large datasets. However, this success comes at the cost of strong dependence on lexical and semantic representations, which are inherently language-specific as well as a high computational cost. As a result, cross-lingual generalization often requires additional training, data augmentation, or explicit multilingual modeling. In contrast, humans can infer document structure even without understanding the underlying language. For example, a reader can identify fields such as names, dates, or totals in an invoice by relying on positional cues, alignment, and formatting conventions. Eye-tracking studies [9] further show that visual layout plays a critical role in guiding attention, with readers exhibiting consistent scanning patterns driven by document structure rather than purely linguistic content.

The central idea of this work is therefore to define a language-invariant predicate space capturing document structure, and to investigate whether learning in this space leads to invariant, explicit, and interpretable logical reasoning patterns. This shifts the focus from language modeling to structure modeling, opening new perspectives for Document AI. Achieving this goal requires a framework that is interpretable, grounded in symbolic reasoning, and robust to the noise inherent in real-world documents. This motivation is closely aligned with classical symbolic AI, which emphasises explicit structure and frugal reasoning, while remaining challenging in noisy settings. Pure neural approaches learn directly from raw data and excel at perception, but remain opaque, do not provide explicit rules, and are typically data- and compute-hungry, relying on large annotated corpora and substantial hardware resources. In contrast, purely symbolic approaches, such as classical Inductive Logic Programming, are interpretable

and computationally efficient, but struggle with noise and often suffer from combinatorial explosion in the rule search space.

Neuro-symbolic methods offer a middle ground, combining differentiable learning with logical reasoning. Among them, we adopt LNNs, which are particularly well suited to our setting due to their ability to learn interpretable structures while maintaining compatibility with first-order logic. In particular, learned weights can be inspected to recover explicit logical rules, enabling frugal inference with modest computational resources. At the same time, since both the predicate space and the rule template are defined a priori, the expressive power of the model is inherently bounded by these design choices. This places significant responsibility on predicate engineering and template specification.

3. State of the Art

In this section, we review recent advances in Document AI, neuro-symbolic learning, and ILP-based approaches for document understanding. We also discuss the main limitations of current methods, particularly regarding interpretability, scalability, and language invariance.

3.1. Document Understanding and Multimodal Models

Document understanding has been significantly advanced by multimodal transformer architectures that integrate textual and visual information. Models such as LayoutLMv3 [3] and LayoutXLM [10] incorporate spatial embeddings alongside textual representations, enabling joint reasoning over content and layout. These approaches achieve state-of-the-art results on tasks such as form understanding, receipt parsing, and document classification. End-to-end models such as Donut [11] remove the dependency on OCR by directly generating structured outputs from images. Graph-based approaches have also been proposed, modeling documents as relational structures where nodes correspond to text elements and edges encode spatial relations[12].

Despite these advances, most approaches remain strongly tied to language representations, either through pretrained language models or token-level supervision. Cross-lingual extensions, such as XFUND [10], highlight the challenges of generalizing these models across languages, often requiring multilingual pretraining or fine-tuning. A notable exception is [2], which shares our vision of language-agnostic document understanding and operates on the same XFUND benchmark. However, their approach relies on Graph Neural Networks combined with a rule-based algorithm for entity relation extraction, whereas we pursue a fully symbolic route by learning explicit first-order rules with LNNs alone.

3.2. Inductive Logic Programming and Neuro-symbolic Learning

Inductive Logic Programming (ILP) aims to learn logical rules from data, typically expressed in first-order logic. Classical ILP systems provide strong interpretability but suffer from combinatorial explosion and sensitivity to noise. Recent work has introduced differentiable extensions of ILP, combining symbolic reasoning with neural learning. Approaches such as the Neural Logic Machines [13], DeepProbLog [14], and differentiable ILP [15] enable gradient-based optimization of logical structures.

Within this line of research, LNN represent logical formulas as neural architectures, where each neuron corresponds to a logical operator in a real-valued logic [16]. Notably, unlike other neural or neuro-symbolic approaches that are not necessarily fully explainable, we choose LNN due to their high degree of interpretability, albeit at the cost of being a template-dependent solution. LNN offer several advantages: they support learning from incomplete or uncertain data, provide interpretable rule representations, and allow integration of prior knowledge through templates [8]. However, they also inherit key limitations from ILP, including dependence on predefined templates, restricted hypothesis spaces, and challenges in scaling to large datasets.

3.3. Neuro-symbolic Approaches for NLP and Document AI

Applications of ILP and neuro-symbolic methods to NLP have explored tasks such as relation extraction, knowledge base completion, and semantic parsing. These approaches demonstrate the ability to learn interpretable rules and incorporate logical constraints into learning.

In the context of Document AI, however, the use of ILP and LNN [8] remains limited. Most systems rely on neural architectures, with symbolic reasoning typically used as a post-processing step or for constraint enforcement [17]. Few works explicitly model document understanding as a logical reasoning problem over structured predicates.

At the same time, the structural nature of documents makes them a natural candidate for symbolic approaches. Layout analysis, table understanding, and key-value extraction can all be expressed in terms of relations such as alignment, proximity, and hierarchy. This suggests that a logical representation may capture essential aspects of document reasoning that are currently handled implicitly by neural models.

3.4. Limitations of Current Approaches

Despite progress in both neural and symbolic paradigms, several limitations remain: (i) Neural models lack interpretability and rely heavily on language-specific features. (ii) ILP and related neuro-symbolic approaches, including LNN, often suffer from scalability issues and require strong prior assumptions. (iii) The integration of symbolic reasoning with modern document representations is still limited. (iv) There is a lack of systematic studies on language invariance in document understanding. These limitations suggest that, while hybrid neural-symbolic approaches are a promising direction, their current formulations still leave open challenges in terms of robustness, interpretability, and language invariance.

4. Logic Neural Networks

In this work, we adopt the neuro-symbolic ILP framework introduced by [7, 8], which combines inductive logic programming with LNN. In what follows, we provide a brief account of this framework, as it constitutes the backbone of our approach.

4.1. Logical Neural Network Operators

LNNs generalize classical Boolean logic to real-valued inputs by replacing crisp truth values in $\{0, 1\}$ with continuous activations in $[0, 1]$, while preserving the semantics of logical connectives through learnable, constrained operators.

The core operators are defined as follows. Let $\mathbf{x} \in [0, 1]^n$ denote a vector of real-valued inputs and let $\alpha \in (\frac{1}{2}, 1]$ be a user-defined hyperparameter that controls the boundary between *low* and *high* truth values. A key property of LNNs is that α acts as a tunable knob: a larger value of α expands the unconstrained region of the operator, granting more flexibility to fit the data while staying close to the semantics of classical logic. This makes LNNs strictly more expressive than fixed t-norms such as Łukasiewicz or product t-norms, which interpolate from Boolean logic in a fixed, parameter-free manner.

LNN-$\wedge$ (conjunction) The conjunction operator is defined as:

$$\text{LNN-}\wedge(\mathbf{x};\, \beta, \mathbf{w}) = \text{relu}_1\Big(\beta - \mathbf{w}^\top(\mathbf{1} - \mathbf{x})\Big)$$

where $\text{relu}_1(x) = \max(0, \min(1, x))$, and the learned parameters $\beta, \mathbf{w}$ are subject to constraints that enforce conjunction-like behavior: the output is low when any input is low, and high when all inputs are high.

LNN-$\vee$ (disjunction) The disjunction operator is defined via De Morgan duality:

$$\text{LNN-}\vee(\mathbf{x};\ \beta, \mathbf{w}) = 1 - \text{relu}_1\Big(\beta - \mathbf{w}^\top \mathbf{x}\Big)$$

subject to the same family of constraints, now ensuring that the output is high when at least one input is high.

LNN-pred (predicate selector) In practice, the structure of the rule to be learned is specified through a *template* (see Section 4.3), whose leaves represent placeholders that can be filled by any predicate from a predefined candidate set, called the *domain* (see Section 4.2). When multiple candidates are available for a given leaf, a dedicated soft-selection operator, LNN-pred, computes a weighted combination of their truth values. Due to the sparsity-inducing properties of relu_1, this operator tends to assign near-zero weight to irrelevant predicates, effectively selecting the most predictive one and producing an interpretable instantiation of the template leaf.

4.2. Predicates

The inputs to an LNN are truth values of *base predicates*, i.e., atomic relations evaluated directly from the data. A predicate is a function that takes one or more elements as input and returns a truth value in $\{0, 1\}$ indicating whether a given property holds. For instance, to learn the rule $\texttt{isCar}(x) \leftarrow \texttt{hasFourWheels}(x) \wedge \texttt{hasEngine}(x) \wedge \neg\texttt{hasPedals}(x)$, one would define unary predicates such as `hasFourWheels`, `hasEngine`, and `hasPedals`, evaluate them on each object x, and feed the resulting truth values to the network. The LNN then learns which combination of predicates best explains the training labels.

Predicates can be defined over any computable property of the input elements. *Unary predicates* capture intrinsic properties of individual objects, such as their size, shape, or category membership. *Binary predicates* capture relational properties between pairs of objects, such as spatial proximity, ordering, containment, or similarity along some dimension. More generally, *n-ary predicates* can encode higher-order relations among arbitrary numbers of elements. In practice, the choice of predicate family is a critical design decision: it determines the expressive power of the model and the kinds of rules that can in principle be learned.

4.3. The Role of the Template

The LNN framework operates over a *program template*: a tree-structured expression in higher-order logic that specifies the shape of the rule to be learned, without committing to specific predicates at its leaves. Formally, a template $\mathcal{T} = (\mathcal{V}, \mathcal{E}, \mathcal{L})$ is a tree whose internal nodes are labeled with logical operators ($\wedge$, $\vee$, $\neg$) and whose leaves are labeled with *domains*, i.e., sets of candidate predicates from which the final rule will be selected.

The template serves a dual purpose. On one hand, it dramatically reduces the combinatorial explosion inherent in rule search: rather than exploring the space of all possible first-order rules, learning is constrained to a fixed logical structure where only the predicate assignments at the leaves are free. On the other hand, this constraint introduces a critical design responsibility: both the shape of the template and the domain assigned to each leaf must be specified a priori, and these choices determine what rules can in principle be discovered. To see why this matters, consider learning the rule:

$$\texttt{isCar}(x) \leftarrow \texttt{hasFourWheels}(x) \wedge \texttt{hasEngine}(x)$$

If one defines a template $A(x) \wedge B(x)$ but assigns the domains as:

$$\mathcal{D}_A = \{\texttt{hasFourWheels}, \texttt{hasEngine}\}, \quad \mathcal{D}_B = \{\texttt{notHasPedals}, \texttt{hasThreeWheels}\}$$

then the correct rule can never be recovered: since the learning process selects one predicate per leaf, and both `hasFourWheels` and `hasEngine` belong to the same domain $\mathcal{D}_A$, they cannot appear simultaneously in the rule. The template thus acts as an inductive bias that, if poorly chosen, may exclude the ground-truth rule from the hypothesis space entirely.

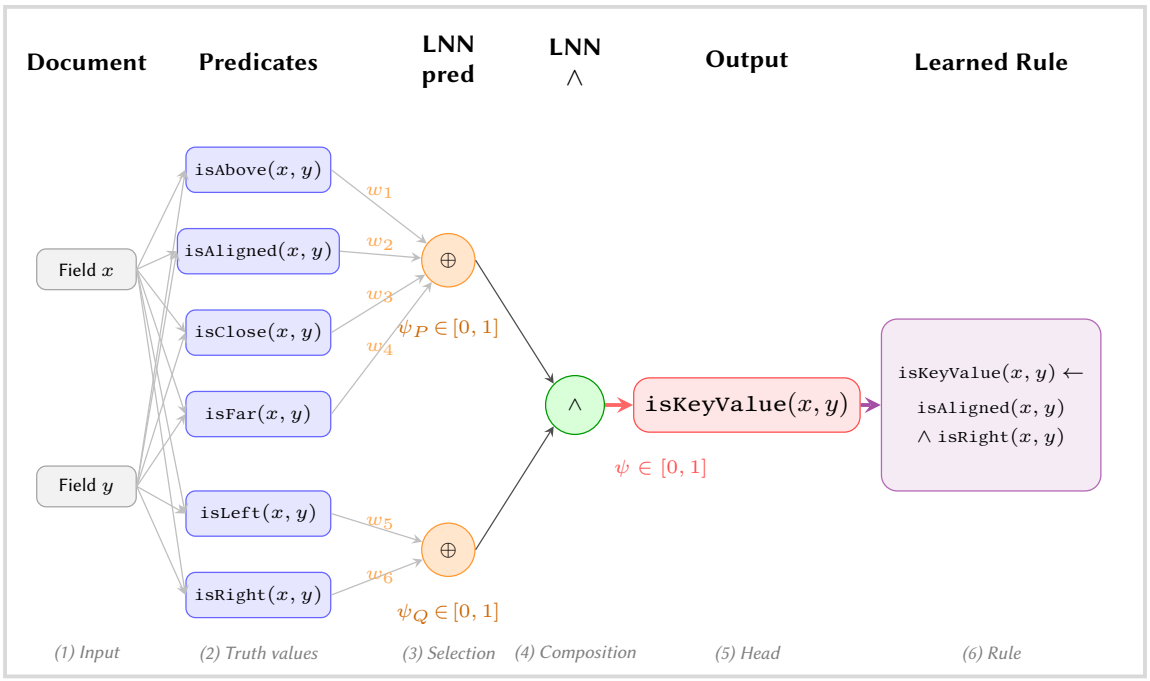


Figure 1: Pipeline of an LNN for key-value pair prediction in forms. Two fields x and y are evaluated against a set of layout predicates, capturing spatial relations each returning a Boolean truth value. The predicates are grouped into two domains $\mathcal{D}_P$ and $\mathcal{D}_Q$, corresponding to the two leaves of the rule template $P(x, y) \wedge Q(x, y)$. For each domain, an LNN-pred operator (orange, $\oplus$) learns a sparse weighted selection over the candidate predicates, concentrating weight on the most predictive spatial relation. The two resulting truth values ψ_P and ψ_Q are composed by an LNN-$\wedge$ operator (green), which enforces conjunction semantics while maintaining close ties to classical Boolean logic. The output $\psi \in [0, 1]$ is the truth value of the head predicate $\texttt{isKeyValue}(x, y)$. Once training converges, the predicate with the highest weight in each domain is retained and the others are pruned, yielding an explicit first-order rule.

4.4. Rule Learning

Given a template and a set of labeled examples, the LNN consists of a neural network that maps truth values of candidate predicates at the leaves to a truth value at the root, corresponding to the head predicate of the rule. Learning proceeds end-to-end via gradient-based optimization, simultaneously (i) selecting which predicate to instantiate at each leaf via the LNN-pred operator, and (ii) learning the parameters of the conjunction and disjunction operators at the internal nodes.

The output of the network for a given grounding is a truth value in $[0, 1]$ for the head predicate. Once training converges, the learned weights can be inspected to recover an explicit, interpretable first-order rule: at each leaf, the predicate with the highest weight is selected, and the resulting assignments, combined via the learned logical operators, constitute the final rule.

5. Learning Rules from Documents

To investigate the research questions outlined in the Introduction, we frame document understanding as a logical learning problem amenable to LNN. Specifically, the objective is to learn first-order logical rules of the form:

$$\texttt{isSomething}(x) \leftarrow A(x) \wedge B(x) \vee C(x) \vee \ldots$$

Such a formulation naturally lends itself to classification tasks over document elements, whether individual tokens or layout blocks (e.g., titles, abstracts, form fields). A central goal of our experiments is to empirically validate the language-invariance hypothesis: since XFUND spans multiple languages over structurally equivalent form documents, it allows us to test whether the logical rules learned by the LNN are consistent across languages. To this end, all predicates are defined exclusively over layout,

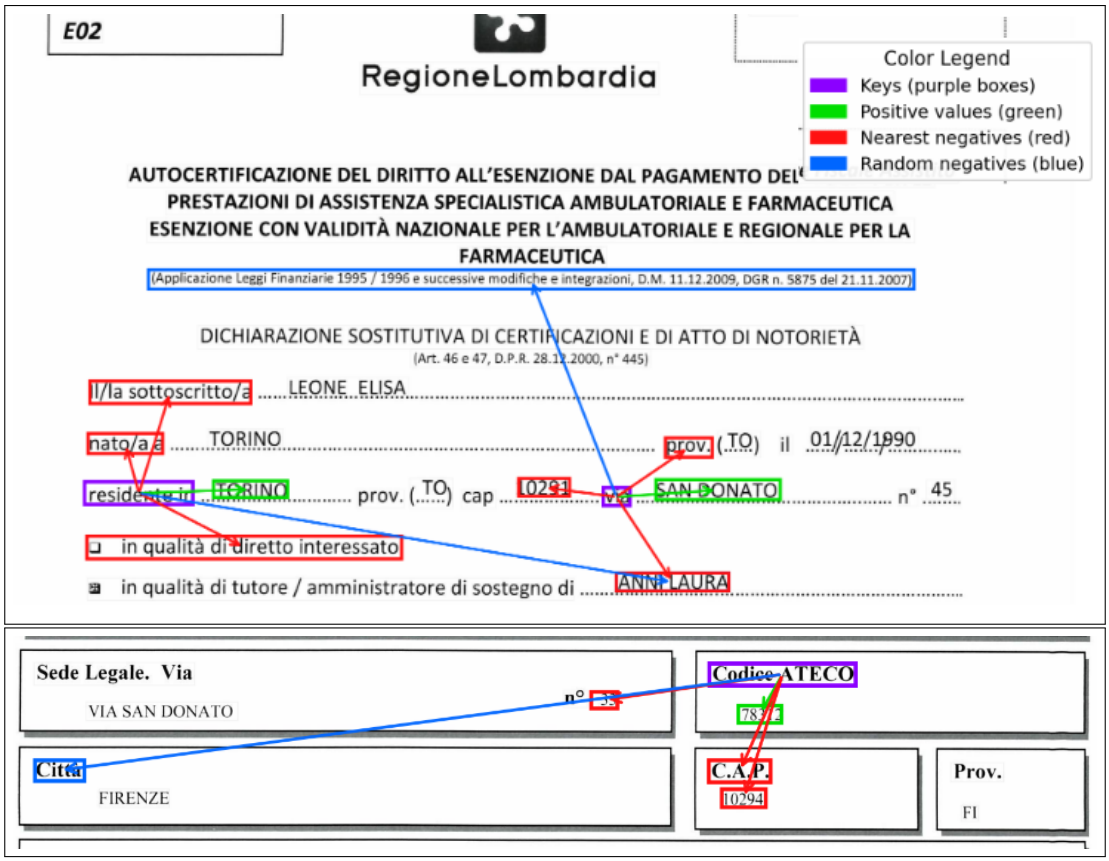


Figure 2: Examples of key-value pairs in XFUND. Key fields, positive answer fields and negative one are shown. The latter include the three spatially closest fields and one randomly sampled field in the page.

positional, and geometric properties of document fields, with no access to textual or semantic content. If the learned rules are similar (or identical) across languages, this constitutes evidence that key-value structure in forms can be captured by language-agnostic logical patterns. An illustrative example of the LNN pipeline applied to a document task is shown in Fig. 1. We define two complementary tasks, key-value relationship prediction and key field classification, and define predicates for each in the following.

5.1. Key-Value Relationship Prediction

The first task consists of learning the binary predicate $\texttt{isKeyValue}(x, y)$, i.e., given a pair of document fields (x, y), predicting whether one is the key and the other is the corresponding value. The LNN receives as input truth values computed over a set of predicates that characterize properties of the pair, such as relative position, spatial alignment, and geometric proximity.

To train and evaluate the model, we construct a dataset of labeled pairs from the XFUND annotations. We consider all fields annotated as *question* that are linked to exactly one *answer* field, discarding cases where the answer is split across multiple tokens, for simplicity. Each such positive pair is complemented by four negative pairs: three are formed by linking the key to its three nearest neighboring fields in the document, and one by pairing it with a randomly sampled field from the same page. This sampling strategy, illustrated in Fig. 2, ensures a realistic and challenging set of negative examples, as the hard negatives are spatially close to the true answer and thus not trivially distinguishable by layout alone. Training and validation splits are derived from the training portion of XFUND, while the test split is derived from the validation portion of XFUND.

Family	Predicates
Spatial	`isBelow, isLeftOf, isRightOf, isOnSameLine, isNotOnSameLine`
Distance	`isCloseEuclidean, isFarEuclidean, isVeryClose, isNotVeryClose, isCloseHorizontally, isFarHorizontally, isCloseVertically, isFarVertically`
Size	`token2WiderThanToken1, token2NarrowerOrEqualToken1, token2TallerThanToken1, token2ShorterOrEqualToken1, similarSize, veryDifferentSize`
Alignment	`verticalAligned, horizontalAligned, centerAlignedH`

Table 1
Binary predicates for key-value prediction, grouped by family.

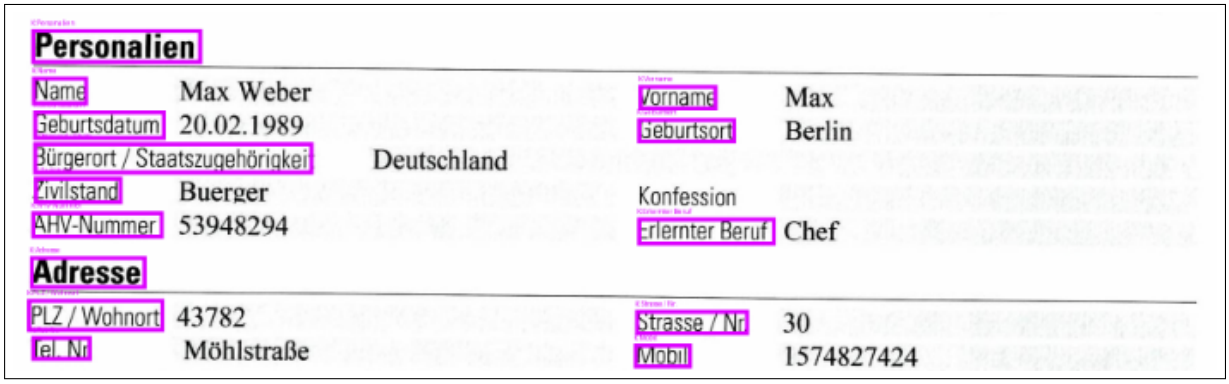


Figure 3: Key field classification on an XFUND page. Boxes indicate fields annotated as *question* (positive examples of key fields); all other fields are negative examples.

5.1.1. Binary predicates

Given a candidate pair (x, y), predicates are defined over the bounding boxes of the two fields and grouped into four families (Table 1). *Spatial* predicates encode the relative position of x with respect to y, distinguishing whether one field lies below, to the left, to the right, or on the same line as the other. *Distance* predicates threshold the Euclidean, horizontal, and vertical distances between the field centers at two granularities, capturing both coarse proximity and fine-grained closeness. *Size* predicates compare the width, height, and overall area of the two fields, encoding whether one is wider, taller, or of similar size to the other. *Alignment* predicates test whether the two fields share a common left edge, top edge, or horizontal center axis, within a fixed tolerance, reflecting the columnar and row-based structure typical of forms.

5.2. Key Field Classification

The second task is a unary classification problem: learning the predicate $\texttt{isKey}(x)$, i.e., determining whether a given field acts as a key in the form. Here, the LNN receives predicates characterizing individual fields.

For this task, no additional dataset construction is required: we directly use the XFUND annotations, treating all fields labelled as *question* as positive examples of keys, and all remaining fields as negative examples. An illustrative example is shown in Fig. 3.

5.2.1. Unary predicates

For individual field classification, predicates capture intrinsic and contextual properties of a single field x within the page (Table 2). *Orthographic* predicates encode surface-level string properties of the field content, such as capitalisation, digit presence, or punctuation, and are the only family that touches the text of a field – though in a purely syntactic, language-agnostic way. *Size* predicates capture

Family	Predicates
Orthographic	`allCaps`, `allDigits`, `capitalized`, `containsColon`, `containsDigit`, `containsSpecialChar`, `multiWord`, `singleWord`
Size	`compactBox`, `largeBox`, `largeWidth`, `smallHeight`, `smallWidth`, `wideAspectRatio`
Position	`bottomRegion`, `centerRegion`, `leftRegion`, `marginAligned`, `rightRegion`, `topRegion`
Context	`hasBottomNeighbor`, `hasRightNeighbor`, `horizontalGapLarge`, `locallyAligned`, `verticallyRepeatedPattern`

Table 2
Unary predicates for key field classification.

the geometric shape of the bounding box, such as aspect ratio, width, and height. *Position* predicates encode the absolute location of the field within the page, distinguishing regions such as left margin, top, or centre. *Context* predicates test relational properties with respect to neighbouring fields, capturing layout regularities typical of form structure, such as vertical repetition patterns, horizontal gaps, or local alignment with adjacent fields.

All thresholds are normalised by page dimensions where applicable, ensuring scale-invariance across the varying resolutions present in XFUND.

6. Experiments

In this section we present the results obtained from the implementation of the framework described above. We begin with the experimental setup, followed by a qualitative and quantitative analysis of the performance achieved on both tasks.

Lang	Best Rule	P	R	F1
de	`(verticalAligned ∧ t.2Nar.OrEq.T.1) ∨ isOnSameLine`	0.758	0.762	0.760
es	`verticalAligned ∧ isOnSameLine`	0.831	0.691	0.754
fr	`verticalAligned ∧ isOnSameLine`	0.860	0.769	0.812
it	`(verticalAligned ∧ isVeryClose) ∨ isOnSameLine`	0.697	0.760	0.727
ja	`(verticalAligned ∧ isVeryClose) ∨ isOnSameLine`	0.836	0.690	0.756
pt	`(verticalAligned ∧ isVeryClose) ∨ isOnSameLine`	0.649	0.723	0.684
zh	`(verticalAligned ∧ t.2Nar.OrEq.T.1) ∨ isOnSameLine`	0.807	0.842	0.824

Table 3
Per-language best rules for key-value pair detection (test set). Precision, Recall, and F1 are reported on the held-out test split.

6.1. Experimental setup

6.1.1. Training and Hyperparameters

All experiments were trained with full-batch Adam (learning rate 0.01, $\alpha = 0.8$), using early stopping on validation loss and restoring the best checkpoint for evaluation. The maximum number of epochs and patience were set to 100 and 15 for the Key-Value task, and to 400 and 60 for the Key classification task, reflecting the higher difficulty of the latter.

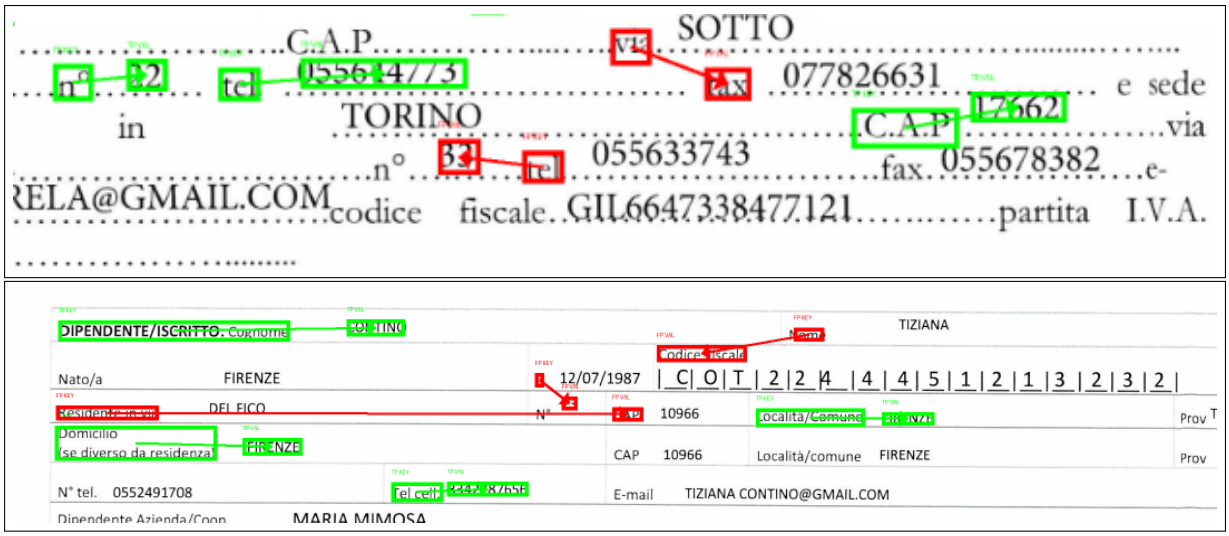


Figure 4: Sample predictions obtained using the rule learned from Italian documents. Correctly identified key-value pairs are highlighted in green, while pairs for which the rule holds but do not constitute a key-value relationship are highlighted in red. Arrows indicate the direction from key to value.

6.1.2. Predicate domains and template sweep

Predicates are grouped into functional domains (e.g., *spatial* or *size*), as shown in Table 1 and in Table 2. We explored three template families: $A \wedge B$, $(A \wedge B) \vee C$, and $(A \wedge B) \vee (C \wedge D)$. For each family we varied assignments of these predicate domains to the roles A, B, C, D and selected the best instantiation by validation F1. For example, for Key Value prediction we evaluated templates such as $(\textit{distance} \wedge \textit{alignment}) \vee \textit{size}$, while for Key classification we evaluated templates like $(\textit{orthographic} \wedge \textit{size}) \vee (\textit{position} \wedge \textit{context})$.

6.1.3. Selection and evaluation

Final symbolic rules were chosen to maximize validation F1; reported metrics (Precision, Recall, F1) are computed on the held-out test split.

6.2. Key-Value Prediction

Table 3 reports, for each source language, the best rule selected on the validation F1, together with test set metrics. A common structure emerges across languages: `isOnSameLine` appears in every rule, either as a conjunct or as a disjunct, confirming that horizontal co-linearity is the strongest single cue for key-value association. Five out of seven languages adopt an $(A \wedge B) \vee C$ template of the form $(\texttt{verticalAligned} \wedge X) \vee \texttt{isOnSameLine}$, where X varies between `isVeryClose` (it, ja, pt) and `token2NarrowerOrEqualToken1` (de, zh). French and Spanish instead collapse to the simpler conjunction $\texttt{verticalAligned} \wedge \texttt{isOnSameLine}$, achieving the two highest F1 scores (0.812 and 0.754 respectively). Portuguese remains the weakest source (F1 0.684). Overall, the learned rules achieve competitive results across all languages, demonstrating that a small set of geometric predicates is sufficient to capture key-value structure in heterogeneous form documents.

A critical factor in the quality of these results is the calibration of the thresholds used to evaluate each predicate. With overly permissive thresholds, predicates such as `isVeryClose` or `isOnSameLine` tend to fire for a large fraction of token pairs, yielding very high recall but poor precision. Conversely, thresholds that are too strict may miss valid associations. Adapting these thresholds to the geometric characteristics of each document type is therefore essential, though it can be challenging when documents exhibit high layout variability across languages or form styles.

A qualitative illustration of the sensitivity to predicate calibration is provided in Fig. 4. As can be

Source	de	es	fr	it	ja	pt	zh
de, zh	0.760	0.746	0.816	0.727	0.751	0.684	0.824
es, fr	0.784	0.754	0.812	0.721	0.749	0.665	0.813
it, ja, pt, zh	0.759	0.746	0.816	0.727	0.756	0.684	0.824

Table 4
Cross-lingual transfer matrix (test F1). Each row corresponds to a distinct rule (shared by the listed source languages) applied to all target languages. Rows are merged when source languages share the same best rule from Table 3.

Lang	Best Rule	P	R	F1
de	`(capitalized ∧ largeBox) ∨ horizontalGapLarge`	0.389	0.811	0.526
es	`(singleWord ∧ hasBottomNeighbor) ∨ (wideAspectRatio ∧ centerRegion)`	0.298	0.867	0.443
fr	`(compactBox ∧ centerRegion) ∨ containsColon`	0.740	0.689	0.714
it	`(multiWord ∧ leftRegion) ∨ horizontalGapLarge`	0.537	0.461	0.496
ja	`singleWord ∧ centerRegion`	0.222	0.746	0.343
pt	`(compactBox ∧ rightRegion) ∨ containsColon`	0.716	0.500	0.589
zh	`(singleWord ∧ horizontalGapLarge) ∨ (largeBox ∧ centerRegion)`	0.383	0.809	0.520

Table 5
Per-language best rules for key field classification with test set metrics (Precision, Recall, F1).

observed, `isOnSameLine` occasionally evaluates to true for token pairs that are not strictly co-linear, due to slight vertical misalignments in the document layout. These borderline cases contribute to the false positives highlighted in red, and suggest that predicate design and threshold tuning remain an open challenge when transferring rules across diverse document collections.

Table 4 reports the cross-lingual transfer results. For each source language, the best rule identified in Table 3 is applied to all target languages, and test F1 is recorded. Since multiple source languages share the same best rule, the corresponding rows yield identical transfer scores and are merged in the table. The results show that the learned rules generalise remarkably well across languages: most cells fall in the 0.75–0.82 F1 range, with low variance across rows. This stability reflects the fact that only three distinct rules cover all seven languages, and their geometric predicates transfer effectively across typologically diverse scripts. Portuguese stands out as a consistent outlier: as a target it yields the lowest F1 in every row (0.665–0.684), suggesting a structural difference in Portuguese form layouts within XFUND.

6.3. Key Field Classification

Table 5 reports results for key field classification. Compared to the Key-Value task, performance drops considerably across all languages, with F1 ranging from 0.343 (ja) to 0.714 (fr), and no language consistently achieving high precision and recall simultaneously. Languages such as ja and es exhibit a high-recall, low-precision profile, suggesting that the learned rules are overly permissive and fire on a large fraction of tokens.

Unlike the Key-Value task, where `isOnSameLine` emerged as a universal anchor, no single predicate dominates across languages here. `containsColon` recurs in fr and pt, reflecting the role of punctuation as a reliable key marker in Romance-language documents, and `horizontalGapLarge` appears in de and it, but beyond these partial patterns the selected rules vary considerably in both structure and predicate choice. Even the template family is inconsistent across languages, pointing to a lack of a shared geometric or textual signature for key tokens.

This difficulty is qualitatively illustrated in Fig. 5, which shows predictions on a Spanish and a

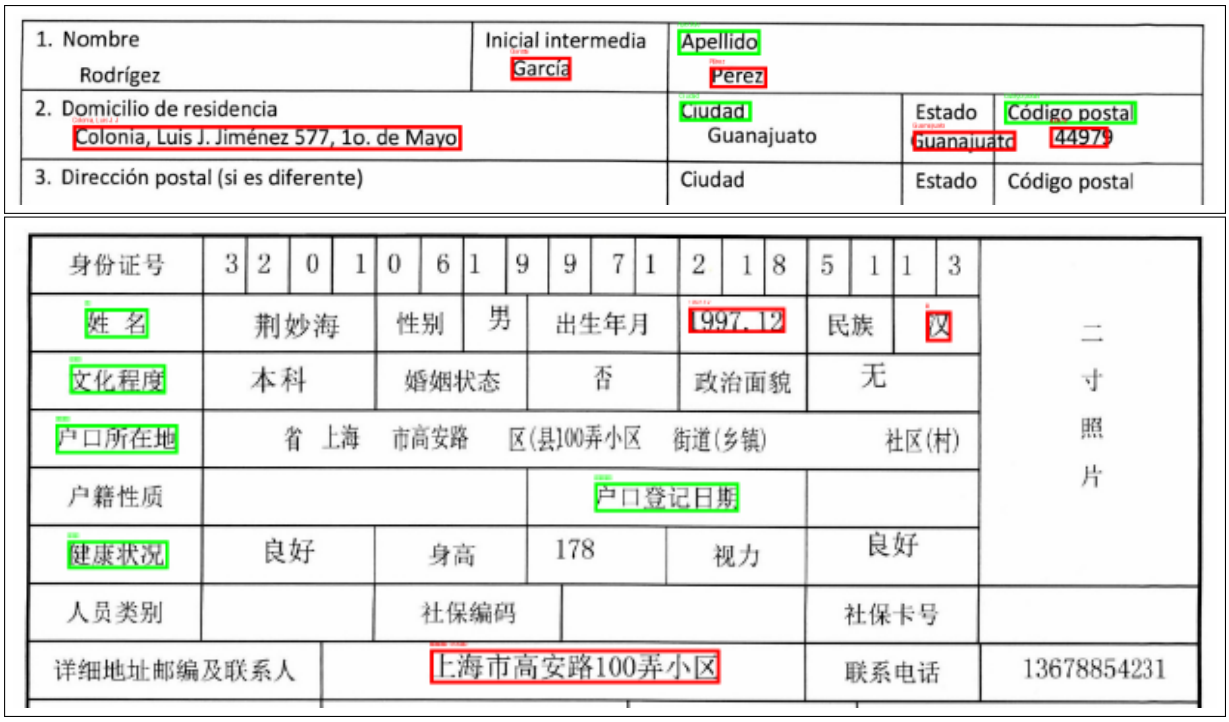


Figure 5: Key classification predictions using the Spanish rule (top) and the Chinese rule (bottom). Green tokens are correctly classified as keys, while red tokens are false positives.

Chinese document. As the examples suggest, determining whether an isolated token is a key is a non-trivial judgment even for a human reader, particularly in an unfamiliar script, and relying solely on single-token properties appears insufficient. The task may fundamentally require predicates that capture relational context between tokens, such as the presence of a plausible value in the neighbourhood, rather than properties of individual tokens in isolation.

6.4. Parameter Efficiency and Model Frugality

We report the number of trainable parameters of each LNN instance to assess model complexity across different template families (Table 6). This analysis is particularly relevant given recent trends toward increasingly large neural models, where parameter counts and computational requirements have grown substantially. LNN-based models operate in contrast to this paradigm, relying on a small number of trainable weights determined solely by the number of instantiated LNN-Pred operators and the size of their associated domains. As a result, more expressive templates naturally induce a larger parameter count, yet the overall scale remains negligible: even the most complex template considered in our experiments requires at most 42 trainable parameters. This makes LNN-based models inherently frugal in terms of both memory and learning requirements, especially when compared to state-of-the-art data-driven approaches. It should be noted that the LNN cannot auto-discover predicates and that this part has been manually done in contrast to the latent representation constructed during pre-training for large neural networks.

7. Conclusion

In this work we presented a neuro-symbolic framework based on Logical Neural Networks for information extraction from multilingual form documents, with the explicit goal of producing fully interpretable models. Rather than relying on textual content or pre-trained language representations, the approach operates exclusively on graphical and layout features, encoding spatial relationships and geometric properties of tokens as logical predicates.

Template family	Key Value	Key Classification
$A \wedge B$	18–28	17–28
$(A \wedge B) \vee C$	29–31	28–31
$(A \wedge B) \vee (C \wedge D)$	42	41

Table 6
Number of trainable parameters for each template family and task. LNN models operate in a low-parameter regime, with complexity driven primarily by predicate instantiation rather than parameter scaling.

On the *Key-Value prediction* task, the framework demonstrates that cross-lingual invariants exist and can be captured by a small set of symbolic rules: only three distinct rules suffice to cover all seven languages, and they transfer effectively across typologically diverse scripts with minimal performance degradation. This result supports the hypothesis that key-value association in form documents is primarily governed by layout invariants that transcend language boundaries.

The *Key Field Classification* task proved considerably more challenging, with lower and less consistent performance across languages. We argue that this difficulty is inherent to the task formulation: classifying a token as a key based solely on its individual properties, without explicit relational context, is an ill-posed problem even from a human perspective. This suggests that the predicate space for this task should be extended to capture inter-token relationships rather than single-token attributes.

It is worth emphasising that all results reported here are obtained with extremely compact architectures, requiring at most 42 trainable parameters. This places LNN-based models in a regime that is qualitatively different from modern data-driven approaches, making them particularly attractive in low-resource or interpretability-critical settings.

Several directions remain open for future work. A principled procedure for adapting predicate thresholds to the statistical properties of each document collection would reduce the need for manual calibration and improve robustness across diverse layouts. Furthermore, automating template selection would remove the last manual design choice from the pipeline, moving toward a fully end-to-end neuro-symbolic learning framework.

Acknowledgments

We thank A. Gemelli for suggesting the reference [2].

Declaration on Generative AI

During the preparation of this work, the authors used Claude (Anthropic) to assist with the creation of Figure 1 (Generate images). The authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] B. L. Davis, et al., Deep visual template-free form parsing, in: ICDAR 2019, Australia, IEEE, 2019, pp. 134–141. doi:10.1109/ICDAR.2019.00030.

[2] B. P. Voutharoja, et al., Language independent neuro-symbolic semantic parsing for form understanding, in: Proc. 17th ICDAR, Part II, LNCS, Springer, 2023, pp. 130–146. doi:10.1007/978-3-031-41679-8_8.

[3] Y. Huang, et al., LayoutLMv3: Pre-training for document AI with unified text and image masking, in: MM '22: 30th ACM Int.l Conf. on Multimedia, ACM, 2022, pp. 4083–4091. doi:10.1145/3503161.3548112.

[4] Q. Team, Qwen3.5-omni technical report, CoRR abs/2604.15804 (2026). doi:10.48550/ARXIV.2604.15804. arXiv:2604.15804.

[5] G. Jaume, et al., FUNSD: A dataset for form understanding in noisy scanned documents, in: 2nd OST@ICDAR, IEEE, 2019, pp. 1–6. doi:10.1109/ICDARW.2019.10029.
[6] Y. Xu, et al., XFUND: A benchmark dataset for multilingual visually rich form understanding, in: Findings of the ACL, Ireland, Findings of ACL, ACL, 2022, pp. 3214–3224. doi:10.18653/V1/2022.FINDINGS-ACL.253.
[7] R. Riegel, et al., Logical neural networks, CoRR abs/2006.13155 (2020). URL: https://arxiv.org/abs/2006.13155. arXiv:2006.13155.
[8] P. Sen, et al., Neuro-symbolic inductive logic programming with logical neural networks, in: Thirty-Sixth AAAI, AAAI Press, 2022, pp. 8212–8219. doi:10.1609/AAAI.V36I8.20795.
[9] K. Rayner, Eye movements in reading and information processing: 20 years of research., Psychological bulletin 124 (1998) 372.
[10] Y. Xu, et al., Layoutxlm: Multimodal pre-training for multilingual visually-rich document understanding, CoRR abs/2104.08836 (2021). URL: https://arxiv.org/abs/2104.08836.
[11] G. Kim, et al., OCR-free document understanding transformer, in: Proc. ECCV, Part XXVIII, LNCS, Springer, 2022, pp. 498–517. doi:10.1007/978-3-031-19815-1_29.
[12] A. Gemelli, et al., Doc2graph: A task agnostic document understanding framework based on graph neural networks, in: ECCV Workshops Proceedings, Part IV, LNCS, Springer, 2022, pp. 329–344. doi:10.1007/978-3-031-25069-9_22.
[13] H. Dong, et al., Neural logic machines, in: 7th ICLR, USA, OpenReview.net, 2019. URL: https://openreview.net/forum?id=B1xY-hRctX.
[14] R. Manhaeve, et al., DeepProbLog: Neural probabilistic logic programming, in: Proc.(BNAIC 2019), CEUR Workshop Proceedings, 2019. URL: https://ceur-ws.org/Vol-2491/abstract68.pdf.
[15] R. Evans, E. Grefenstette, Learning explanatory rules from noisy data, J. Artif. Intell. Res. 61 (2018) 1–64. doi:10.1613/JAIR.5714.
[16] K. Gao, et al., Learning first-order rules with differentiable logic program semantics, in: Proc. IJCAI, 2022, pp. 3008–3014. doi:10.24963/IJCAI.2022/417.
[17] A. Hemmer, et al., Neurosymbolic information extraction from transactional documents, Int. J. Document Anal. Recognit. 28 (2025) 475–485. doi:10.1007/S10032-025-00530-0.