GLiFormer: A Generalist Multitask Transformer Encoder

Ihor Stepanov[1], **Mykhailo Shtopko**[1], **Dmytro Vodianytskyi**[1], **Oleksandr Lukashov**[1], **Mykyta Yaroshenko**[1]

[1]Knowledgator Engineering, Kyiv, Ukraine

Correspondence: ingvarstep@knowledgator.com

Abstract

We introduce GLiFormer, a compact schema-conditioned encoder framework that unifies multiple information-processing tasks through a shared representation and generalized anchor formulation. Runtime labels are matched against anchors representing entities, relations, classes, or records, allowing named-entity recognition, relation extraction, classification, and hierarchical text structuring to share a single source encoding. For hierarchical structuring, GLiFormer grounds field values directly in the source, assigns them to unordered record anchors, and predicts parent–child relations before deterministically assembling nested JSON, eliminating autoregressive output generation. The framework additionally supports document inputs with spatial and optional visual features. We initialize GLiFormer-base from a DeBERTa backbone further pretrained on 100 billion diverse tokens and train the resulting models using broad multitask training followed by task-focused post-training. Across 26 NER and 13 classification datasets, GLiFormer demonstrates strong performance relative to its parameter count. On a 500-example multilevel structuring benchmark, GLiFormer-large achieves 91.10% F1, compared with 91.96% for GPT-5.6-luna. On a separate depth-balanced efficiency benchmark, GLiFormer-base achieves median end-to-end latencies of 69,ms on GPU and 547,ms on CPU; under explicit autoregressive throughput assumptions, the corresponding GPU workload is estimated to be up to 95.8× faster. These results show that a single compact encoder can support diverse schema-conditioned tasks while providing accurate, source-grounded structured extraction without autoregressive generation.

1 Introduction

Information extraction is often deployed as a collection of specialized models: one recognizes entities, another classifies documents, and a third reconstructs records. Yet these tasks share a basic operation: encode the source, represent the requested concepts, and score their compatibility. In this paper, we present a general architecture that makes those concepts explicit and allows one model to serve several output spaces.

Hierarchical text structuring is a demanding instance of this problem. An invoice schema, for example, may request a seller, repeated line items, and relationships among nested records. Autoregressive models can express such outputs flexibly, but generate field names, punctuation, and values in sequence. We instead separate extraction into field grounding, record membership, and hierarchy prediction, followed by deterministic JSON assembly. This removes output-token generation from the inference path while retaining explicit alignment to the source.

GLiNER introduced joint representations of text and natural-language entity labels [Zaratiana et al., 2024]; subsequent work extended this interface to token-level multitask extraction and schema-driven prediction [Stepanov and Shtopko, 2024, Zaratiana et al., 2025]. We generalize the object

matched to each label through an *anchor*: a group-level representation for classification, an entity pair for a relation, or a record slot for structuring. Parent-scoped labels and task-specific anchors preserve the semantics of each request while sharing the encoder and reusable layers.

Multiple schemas for one document are executed as task-local groups after a single source encoding. This allows several ontologies or classification questions to reuse source features; computation in the heads still scales with the requested groups, labels, and anchors. Figure 1 summarizes the task families served by this interface. Our contributions are:

1. A shared anchor formulation that separates candidate acquisition, source-conditioned refinement, and anchor–label interaction.
2. An entity-first structuring head that grounds fields, forms unordered records, and reconstructs their hierarchy using directed relation scores.
3. A configurable multitask model for text and document inputs, evaluated on extraction, classification, and multilevel structuring.

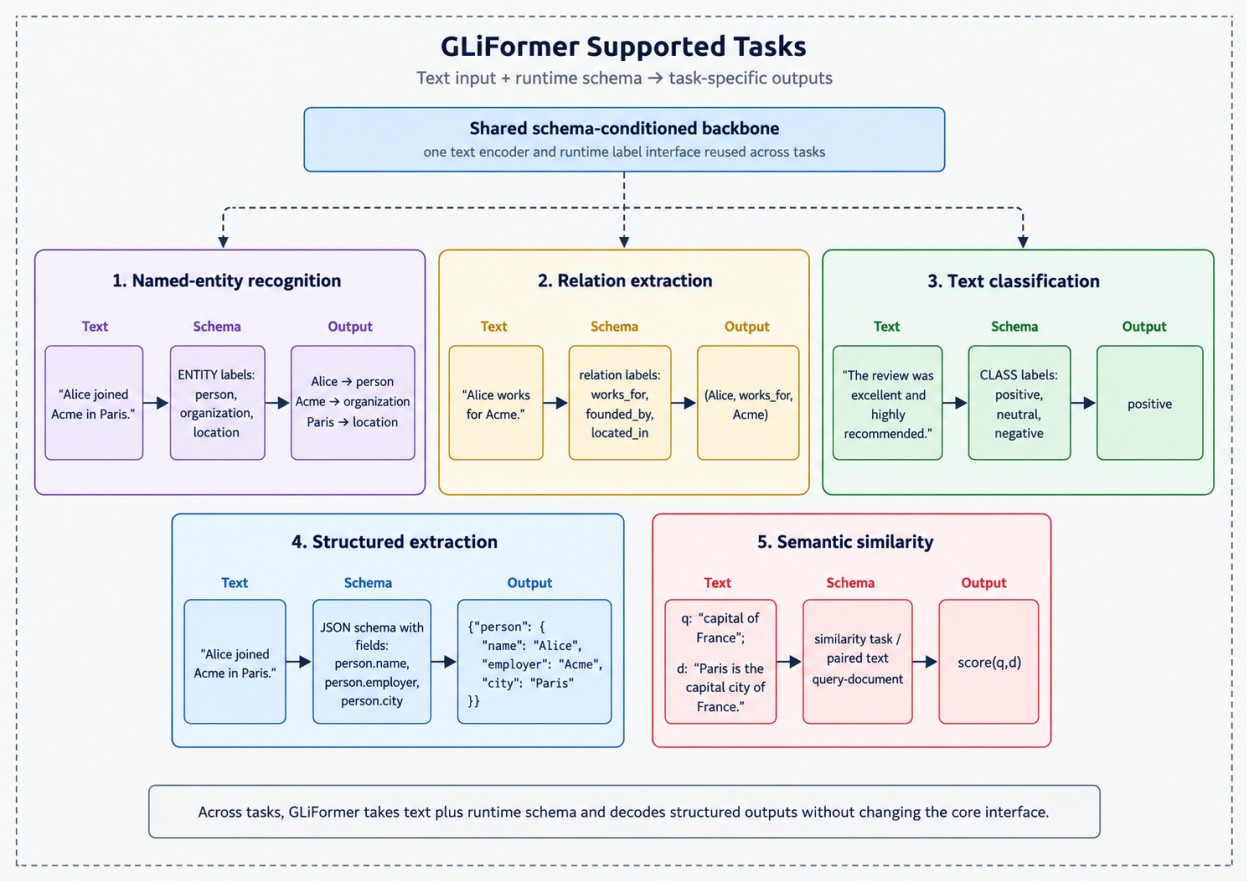

Figure 1: Task overview of GLiFormer. Runtime schemas specify the requested labels and output structure, while task-specific predictors and decoders turn shared source features into spans, relations, classes, or nested records.

2 Related Work

Bidirectional encoders. BERT established pretrained bidirectional Transformers as general feature encoders for language understanding [Devlin et al., 2019]. DeBERTa separates content and position representations in attention [He et al., 2021b], and DeBERTaV3 improves pretraining with replaced-token detection [He et al., 2021a]. The model studied here uses a continued-pretrained DeBERTa-base backbone, while GLiFormer's contribution is the computation above the encoder. Backbone choice and anchor design are separate modeling decisions.

Runtime-label information extraction. GLiNER casts named-entity recognition (NER) as compatibility learning between text spans and natural-language entity labels [Zaratiana et al., 2024]. Its multitask token architecture predicts start, end, and inside evidence for each token–label pair and applies the interface to several extractive tasks [Stepanov and Shtopko, 2024]. GLiNER2 composes NER, text classification, and hierarchical structured extraction through a runtime schema [Zaratiana et al., 2025]. GLiClass studies zero- and few-shot sequence classification with related text–label interaction [Stepanov et al., 2025]. GLiFormer retains runtime-label matching but separates each group into a parent context and child labels, then adds an anchor axis. Multiple anchors can denote records, relation hypotheses, visual objects, or temporal segments; multiple groups can coexist for one input.

Relation extraction depends on entity arguments rather than isolated spans. GLiNER-Relex shares an encoder across text, entity labels, and relation labels; recognized entities are converted into pair representations and scored against runtime relation types [Stepanov et al., 2026]. GLiFormer supports this entity-first route as well as slot-based formulations. We treat GLiNER2, GLiClass, and GLiNER-Relex as architectural precedents, not interchangeable empirical baselines: their corpora, candidate construction, thresholds, and metrics differ.

Set prediction. DETR predicts a bounded, unordered collection of objects with learned queries and a global bipartite assignment between predictions and targets [Carion et al., 2020]. The Hungarian method makes the supervision invariant to target ordering [Kuhn, 1955]. This principle transfers to records and relation tuples: queries denote potential instances, unmatched queries receive negative supervision, and inactive slots are discarded. Our term *anchor* means a general hypothesis representation, not a hand-designed geometric box anchor. Dense token and slot objectives may also use focal loss to reduce the contribution of easy negatives [Lin et al., 2017]; its presence is a configurable mechanism rather than evidence of accuracy.

Layout and representation learning. Document understanding often requires spatial and visual evidence. LayoutLMv3 jointly pretrains document text and image patches [Huang et al., 2022]. GLiFormer instead extends a text encoder with 2-D box/page features and optional patch tokens while preserving the same task-head contract. Sentence-BERT popularized pooled bi-encoder representations for semantic comparison [Reimers and Gurevych, 2019], and BEIR offers a heterogeneous zero-shot retrieval benchmark [Thakur et al., 2021]. These works motivate the layout and embedding branches, whose value still requires task-specific evaluation.

3 Model Architecture

Figure 2 outlines the neural computation. The encoder produces source memory and schema representations; task-local grouping then routes these features to the appropriate predictors. Anchors represent the candidate objects required by each task, with optional refinement against source evidence. The following subsections formalize this shared interface and show how it specializes to different output structures.

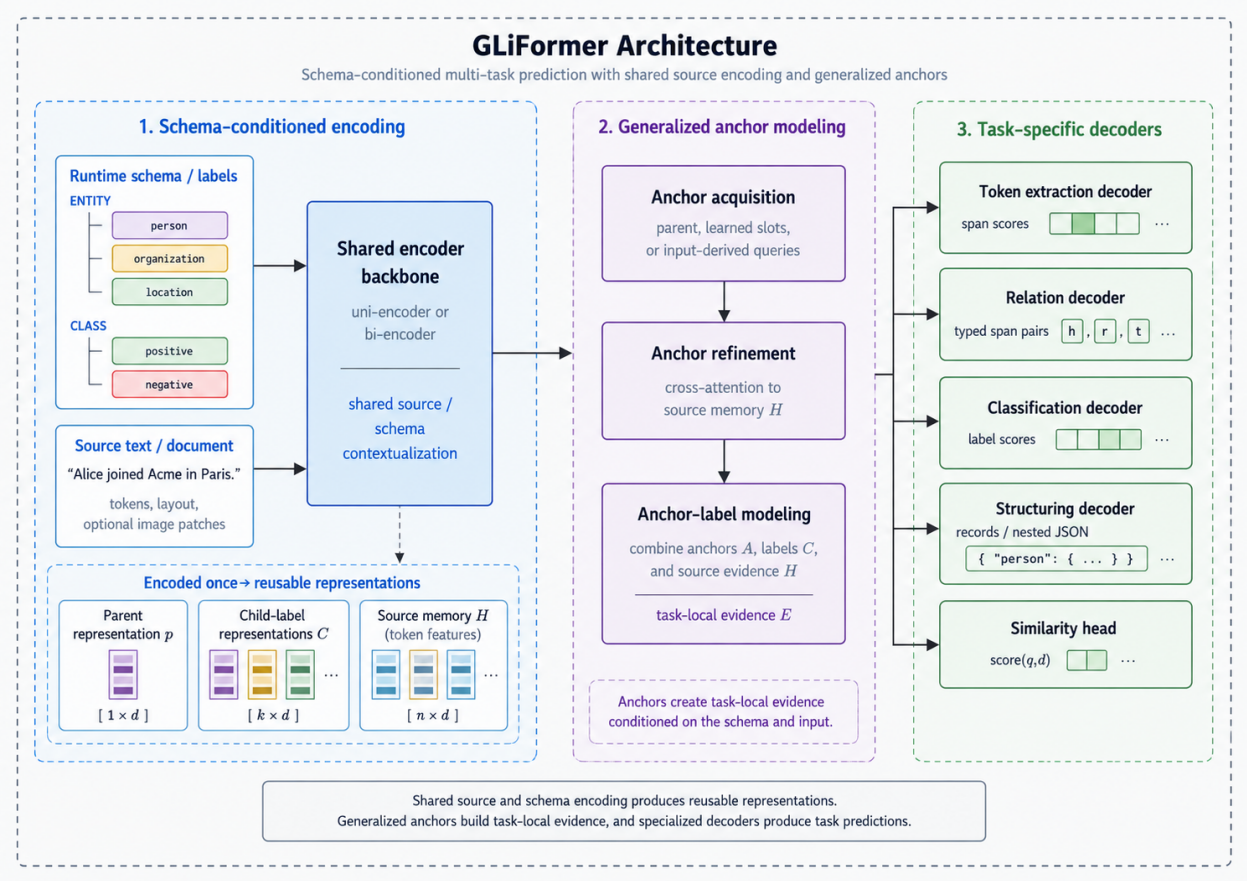

Figure 2: Model architecture of GLiFormer. Source memory H, group-parent features p, and child-label features C form the shared representation contract. Task-specific pathways acquire and optionally refine anchors before predicting label compatibility, token evidence, or record membership and hierarchy. Global heads can operate directly on pooled representations.

3.1 Problem formulation

Let a batch contain source inputs $\{x_b\}_{b=1}^{B}$. For task t, source b may be queried by $G_{b,t} \geq 0$ schema groups. A group

$$\mathcal{S}_{btg} = \left(s^{p}_{btg}, \{s^{c}_{btgk}\}_{k=1}^{K_{btg}}\right) \tag{1}$$

contains a parent description s^p and a set of child labels s^c. The parent establishes a local semantic scope—for example, a classification question, an object type, or a relation query—while the children denote its classes, fields, entity types, or relation types. A task-specific predictor combines each source with its schema group to produce the corresponding output. Different groups may reuse the

same source while defining unrelated output spaces. Child labels are therefore runtime inputs rather than fixed indices in a global classifier.

The common anchored pathway separates three axes throughout its computation: *source positions* retain evidence, *schema labels* describe the requested semantics, and *anchors* represent candidate output objects. Group-level decisions and token extraction use one anchor; slot-based set prediction uses several. This factorization is the main architectural invariant, although global heads such as counting and pair similarity may operate directly on pooled source representations.

3.2 Schema-conditioned representations

A shared encoder maps a source and all of its requested schemas to

$$(H_b, \{p_{btg}, C_{btg}\}_{t,g}) = \mathcal{E}_\theta(x_b, \mathcal{S}_b), \tag{2}$$

where $H_b \in \mathbb{R}^{L_b \times d}$ is source memory, $p_{btg} \in \mathbb{R}^d$ is the group-parent representation, and $C_{btg} \in \mathbb{R}^{K_{btg} \times d}$ contains child-label representations. The encoder may contextualize source and schema jointly or encode them in separate streams; downstream computation depends only on the resulting contract. A joint encoder allows schema and source states to interact inside the backbone, whereas separate streams permit label representations to be computed independently. In either case, the parent and child states are projected to the same width as the source memory. Masks m^H and m^C accompany the source and child axes, so padding never acquires semantic meaning.

The same abstraction admits non-textual evidence. Geometry, image regions, or acoustic frames can be incorporated while constructing H, provided that the result is a masked sequence in the common feature space. Position descriptors may remain attached to the memory for later refinement. Thus the task architecture does not depend on which backbone produced H, p, and C.

3.3 Grouped schema execution

For an active task, all source–group pairs are collected on a task-local axis. Define

$$N_t = \sum_{b=1}^{B} G_{b,t}, \qquad \rho_t(n) = (b_n, g_n), \qquad K_t^{\max} = \max_{n \le N_t} K_{b_n t g_n}, \qquad L_t^{\max} = \max_{n \le N_t} L_{b_n}. \tag{3}$$

The gather operation

$$(H_n, p_n, C_n, o_n) = \mathrm{Gather}_t(H_{b_n}, p_{b_n t g_n}, C_{b_n t g_n}, b_n), \quad \rho_t(n) = (b_n, g_n), \tag{4}$$

repeats source memory only at the task-head boundary, pads child labels within that task, and records $o_n = b_n$ for reconstruction. This mathematical reindexing makes multiple schemas part of one forward computation without requiring repeated source encoding. For example, two NER schemas for source 1 and one for source 2 produce $N_{\mathrm{NER}} = 3$ with $o = [1, 1, 2]$. Predictions and losses remain group-local: labels from different groups are padded into a common tensor for execution but are neither compared nor normalized against one another. The origin map later restores the source and schema grouping.

Algorithm 1 summarizes the complete computation. It is written in terms of functions, not software components: $\mathcal{G}_t$ acquires anchors, $\mathcal{R}_t$ refines them, Φ_t combines them with labels, h_t predicts task quantities, and D_t reconstructs outputs.

Algorithm 1 Schema-conditioned multitask prediction

Input: Sources $\{x_b\}$, grouped schemas $\{\mathcal{S}_{btg}\}$, and active tasks $\mathcal{T}$.
Output: Predictions $\{y_{btg}\}$.

1. Encode each source once and obtain source memory H_b, group parents p_{btg}, and child-label banks C_{btg}.
2. For each task t, gather its source–group pairs into rows (H_n, p_n, C_n, o_n) and construct the corresponding masks.
3. For every anchor-conditioned row, acquire candidates $A_n^0 = \mathcal{G}_t(p_n, H_n)$ and discard padded candidates through the anchor mask.
4. Refine candidates against source evidence, $\widetilde{A}_n = \mathcal{R}_t(A_n^0, H_n)$, then form label-conditioned hypotheses $F_n = \Phi_t(\widetilde{A}_n, C_n)$.
5. Apply the task predictor h_t to F_n and the required source evidence. Global heads that do not require anchor–label hypotheses may instead operate directly on pooled source memory.
6. Use o_n and the schema masks to restore group predictions, then apply the deterministic task decoder D_t.

3.4 Anchor-conditioned prediction

3.4.1 Anchor acquisition and refinement

For group n, the acquisition map provides a padded capacity of up to Q_t candidate objects:

$$(A_n^0, m_n^A) = \mathcal{G}_t(p_n, H_n), \qquad A_n^0 \in \mathbb{R}^{Q_t \times d}, \quad m_n^A \in \{0,1\}^{Q_t}. \tag{5}$$

The source of an anchor determines its inductive bias, not its interface. A parent-derived anchor represents one group-level decision; input-derived anchors represent grounded candidates; learned slots represent a bounded unordered set; and position-derived anchors impose a spatial or sequential prior. The active mask lets capacity exceed the number of meaningful candidates and separates structural availability from a learned prediction that a slot is occupied.

Anchors are optionally normalized and refined as queries over source memory. A generic pre-normalized refinement layer can be written

$$\begin{aligned} A^s &= A + \mathrm{Attn}_s(\mathrm{N}(A) + P_q, \mathrm{N}(A) + P_q, \mathrm{N}(A); B_s), \\ A^c &= A^s + \mathrm{Attn}_c(\mathrm{N}(A^s) + P_q, \mathrm{N}(H) + P_h, \mathrm{N}(H); B_c), \\ \widetilde{A} &= A^c + \mathrm{FFN}(\mathrm{N}(A^c)). \end{aligned} \tag{6}$$

P_q and P_h describe anchor and memory positions, while B_s and B_c are optional attention biases. They may encode order, geometry, locality, or causality. Self-attention lets candidate objects exchange information before cross-attention grounds them in the source; both operations respect the anchor and memory masks. Refinement as a whole is optional, so a task can use the acquired anchors directly.

For token extraction, each refined anchor and child label form a fused hypothesis $F_{qk} = \Phi_t(\widetilde{A}_q, C_k)$. The token scorer combines this hypothesis with source position ℓ:

$$S_{q\ell k:} = \mathrm{MLP}([U_1(H_\ell); V_1(F_{qk}); U_2(H_\ell) \odot V_2(F_{qk})]) \in \mathbb{R}^3. \tag{7}$$

Here U_1, U_2, V_1, V_2 are learned affine projections, $\odot$ denotes elementwise multiplication, and the three output channels score start, end, and inside evidence. Group indices are omitted for clarity; the single-anchor case yields an $L \times K \times 3$ score tensor.

3.5 Task instantiations

The same representation contract yields different tasks by changing the semantics of anchors, the evidence consumed by the final predictor, and the decoder. Figure 1 gives the task-level overview; Table 1 details the corresponding anchor semantics, prediction evidence, and decoded outputs.

Table 1: Implementation-independent task instantiations of the shared architecture.

Task	Anchor semantics	Evidence and label interaction	Decoded output
Named entities	one group anchor	token evidence against entity labels	grounded labeled spans
Relations	entity pairs or relation slots	pair features against relation labels	typed directed links
Classification	one group anchor	pooled source evidence against class labels	one or more group-local classes
Entity-first structuring	record slots with set supervision	grounded fields, membership, presence, and record relations	typed nested records
Pair similarity	none	pooled source pair; no child-label axis	scalar score or embedding
Spatial/temporal sets	object or event slots	modality memory against runtime labels	boxes, masks, or intervals

3.5.1 Runtime task schemas

Runtime schemas define parent-scoped labels: entity types for recognition, classes for classification, relation types for links, and field paths for structuring. Several groups may request independent predictions from the same source. Nested schemas additionally specify object types, repeated records, and permitted parent–child relationships. These requests share the representation in Equation 2 while preserving their own output semantics.

3.5.2 Named-entity recognition

Named-entity recognition specializes Equation 7 to one anchor, predicting start, end, and inside evidence for every source-token and entity-type pair. For a supervised mention of type k covering inclusive token indices $[a, b]$, the processor sets the start target at a, the end target at b, and the inside target at every position $a{:}b$. All other valid cells are negative. The three sigmoid objectives are independent, so two types may share a boundary and nested mentions can be represented without forcing one token into a single BIO class.

At inference let p^s, p^e, and p^i be the sigmoid probabilities. Every above-threshold start is paired with a later above-threshold end of the same type, and all intervening inside probabilities must pass the threshold. The decoder ranks a candidate with a conservative boundary score

$$d(a, b, k) = \min\left(p^s_{ak}, p^e_{bk}, \min_{a \le \ell \le b} p^i_{\ell k}, 1 - p^i_{a-1,k}, 1 - p^i_{b+1,k}\right), \tag{8}$$

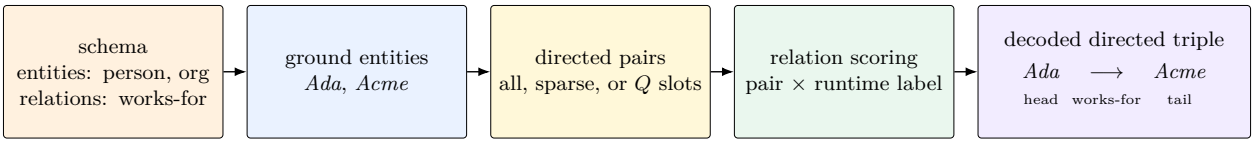

Figure 3: Entity-first relation extraction. Runtime schemas define endpoint types and directed relation labels. Pair construction can retain all grounded pairs, prune them with an adjacency model, or assign endpoints to an unordered set of relation slots; the final scorer is shared across the resulting pair representations.

where nonexistent neighbor terms are omitted. The outside-inside terms favor complete values over high-scoring suffixes. Candidates are then processed in descending score order, with overlap constraints chosen for flat or nested entities. Predictions retain source offsets, entity labels, and confidence scores.

The neural path scores only $L \times K$ token–label cells. Span construction occurs after scoring and therefore has no trained maximum width: a retained span may cover any number of tokens within the encoded context. Independent label inventories reuse the same source encoding.

The same operator grounds fields for structured extraction. Field paths replace entity types, while decoded spans retain source offsets and field identity. Pooling the tokens of a decoded or supervised span produces the grounded representation used by relation endpoint assignment and record membership.

3.5.3 Relation extraction

Relation extraction is directed: reversing the arguments changes the hypothesis. The entity-first path first applies the NER procedure above, pools each retained span into E_i, constructs candidate ordered pairs, and then compares each pair with every runtime relation label:

$$r_{ij} = \psi([E_i; E_j]), \qquad \ell_{ijk} = s_{\text{rel}}(r_{ij}, C_k^{\text{rel}}), \qquad i \neq j. \tag{9}$$

The source and target roles are not symmetrized. A positive output therefore contains two grounded endpoint spans, the relation name supplied at runtime, and a score rather than only an ungrounded class index.

Scoring all directed entity pairs costs $O(E^2R)$. A learned adjacency model can reduce this cost by selecting promising pairs before relation typing.

An alternative replaces pair enumeration with Q unordered relation slots. Each source-conditioned slot predicts one source assignment and one target assignment over the grounded entity set, followed by one or more relation labels. Hungarian matching aligns these slots to gold directed pairs during training; objectness can suppress unused capacity at inference. This bounds pair computation but also caps the number of returned pairs at Q.

When entity types are not part of the request, the open-relation head extracts relation-conditioned spans, pools them, assigns two endpoints to each slot, and scores each slot against the relation bank. This entity-first formulation returns directed triples without requiring a separate entity-type inventory.

3.5.4 Hierarchical text structuring

GLiFormer uses an entity-first set formulation for structured extraction: it grounds fields once, assigns the resulting mentions to record slots under permutation-invariant supervision, and connects

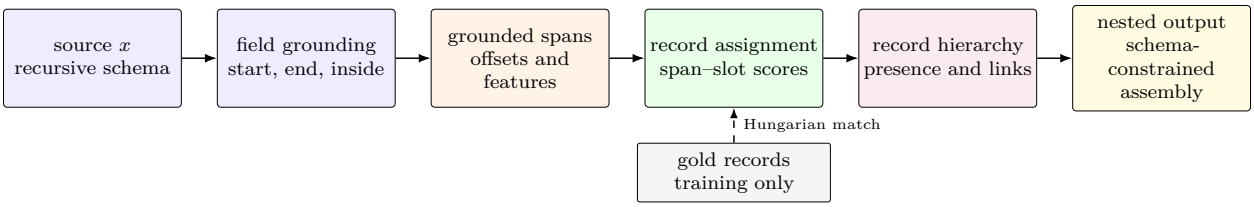

Figure 4: A concrete entity-first structuring schema and its execution. Colors separate the source, typed runtime schema, grounded fields, record slots, hierarchy, and deterministic JSON assembly. Field grounding is shared across records; Hungarian matching aligns record slots with supervision only during training.

those records into a hierarchy. Separating these decisions avoids autoregressive generation of field names, punctuation, and copied values.

For each schema group, a recursive schema is represented by qualified field paths and metadata identifying object and array nodes together with permitted parent–child paths. Qualified paths distinguish repeated leaf names under different branches while retaining the field's structural role. The neural computation then follows the simple sequence in Figure 4.

For example, consider the public schema

```
invoice = {
  id:  string, seller:  {name:  string},
  items:  [{sku:  string, quantity:  integer}]
}.
```

It compiles the four leaf labels `invoice.id`, `invoice.seller.name`, `invoice.items.sku`, and `invoice.items.quantity`, together with three record paths (invoice, seller, and item) and permits only schema-valid directed edges such as invoice→seller and invoice→item. The type words are output constraints rather than text to generate: the model grounds a surface span first, and the decoder subsequently converts, for example, `"2"` to the integer 2.

The first stage is the single-anchor specialization of Equation 7 and produces $S^{\text{fld}} \in \mathbb{R}^{L\times K\times 3}$. Let $z_e = (a_e, b_e)$ denote a retained span. Its evidence for field k and its source-grounded representation are

$$F^{\text{fld}}_{ek} = \min\left\{ S^{\text{start}}_{a_e k}, S^{\text{end}}_{b_e k}, \min_{\ell=a_e}^{b_e} S^{\text{inside}}_{\ell k} \right\},$$
$$E_e = \text{SpanEnc}(H, z_e) \in \mathbb{R}^d. \tag{10}$$

A candidate span is retained once even when it is compatible with several fields or records; its field scores remain in $F^{\text{fld}}_{e:}$. During training, gold spans, optionally augmented with sampled negative spans, supply the second-stage candidates; inference uses spans decoded from the token evidence. Record slots therefore neither propose nor type field values.

The second stage acquires and refines Q candidate record anchors. The same anchor representation supports three complementary predictions:

$$M_{qe} = \widetilde{A}_q^\top E_e, \qquad O_q = s_{\text{pres}}(\widetilde{A}_q), \qquad R_{qq'} = s_{\text{hier}}(\widetilde{A}_q, \widetilde{A}_{q'}), \quad q \neq q'. \tag{11}$$

M_{qe} scores the assignment of grounded span e to record q without repeating its field classification. The optional presence score O_q separates predicted records from unused slot capacity. The relation-style score $R_{qq'}$ is directed and restores parent–child links for nested schemas; flat schemas require only membership and, when enabled, presence.

At inference, inactive slots are removed, field and membership decisions are combined, and directed links are restricted to paths allowed by the schema. The resulting graph is converted deterministically into dictionaries and lists, preserving source-grounded values and the requested field types.

Grounding each distinct field span once avoids repeating token–field prediction for every record. The second stage instead predicts the $Q \times E$ membership matrix in Equation 11. Flat schemas omit record links; multilevel schemas add directed hierarchy prediction. Slot count fixes simultaneous record capacity, while presence, membership, field, and hierarchy thresholds control decoding.

3.5.5 Global predictions and pair similarity

For classification, a masked pooling operator summarizes source memory as $r_n = \mathrm{Pool}(H_n, m_n^H)$. A parent-derived anchor may be refined against H_n, paired with each runtime class label, and scored as

$$\ell_{nk} = s_{\mathrm{cls}}(r_n, F_{n1k}), \qquad \ell \in \mathbb{R}^{N_{\mathrm{cls}} \times K_{\mathrm{cls}}^{\max}}. \tag{12}$$

Independent sigmoid decisions give multi-label output. For single-label output, the highest sigmoid score is returned if it exceeds the threshold. Since classes are group-local, one source may answer multiple questions with different class inventories.

Not every global prediction requires the full anchor–label grid. For example, a counting head maps the group-parent representation directly to a scalar regression output or a distribution over bounded counts. Pair similarity projects and pools each source, then compares the resulting vectors using cosine similarity, a dot product, or negative Euclidean distance. The pooled vectors can also be exported directly for retrieval.

3.5.6 Spatial and temporal set prediction

The anchor abstraction is not tied to text. Over visual memory, anchors can denote objects and predict runtime classes, objectness, boxes, and masks. Over acoustic memory, they can denote events and predict classes or temporal intervals. These tasks reuse acquisition, source-conditioned refinement, anchor–label interaction, and set matching. Their final predictors, losses, and output mappings remain modality-specific.

3.6 Learning and permutation-invariant matching

Each supervised task contributes a masked loss, and a batch optimizes only the tasks for which supervision is present:

$$\mathcal{L}(\theta) = \sum_{t \in \mathcal{T}_{\mathrm{sup}}} \lambda_t \mathcal{L}_t(\theta). \tag{13}$$

Token, class, and adjacency decisions may use binary cross-entropy or focal objectives; similarity and geometry terms use task-appropriate losses. The weights λ_t place heterogeneous objectives on a common optimization scale.

Set-prediction slots have no fixed target identity, so supervising them according to an arbitrary gold-record order would make that order part of the learning problem. Instead, a rectangular Hungarian assignment aligns active predictions with valid targets for each group [Kuhn, 1955]. In entity-first structuring, let $\mathcal{Q}$ index the Q_a active record slots, let J be the number of gold records, and let $Y^{\mathrm{mem}} \in \{0,1\}^{J \times E}$ contain their span-membership profiles. The pairwise cost combines

membership agreement, soft Dice overlap, and optional record presence:

$$\mathcal{C}_{qj} = \frac{w_m \mathcal{C}_{qj}^{\mathrm{mem}} + w_d \mathcal{C}_{qj}^{\mathrm{Dice}} + w_o \mathcal{C}_{qj}^{\mathrm{pres}}}{w_m + w_d + w_o}, \tag{14}$$

with padded span candidates masked out and $w_o = 0$ when presence is disabled. Field identity is supervised by the extraction stage. Hierarchy targets are applied after the record correspondence is fixed.

When $Q_a \geq J$, the assignment is the minimum-cost one-to-one map from gold records to active slots,

$$\hat{\pi} = \arg \min_{\pi \in \Pi(J, \mathcal{Q})} \sum_{j=1}^{J} \mathcal{C}_{\pi(j), j}, \tag{15}$$

where $\Pi(J, \mathcal{Q})$ is the set of injective maps from $\{1, \ldots, J\}$ into $\mathcal{Q}$. This global constraint prevents two gold records from claiming the same prediction. If $J > Q_a$, rectangular matching instead selects Q_a one-to-one record–slot pairs. The remaining gold records cannot receive record-level membership, presence, or hierarchy supervision, although their visible fields may still contribute to the first-stage extraction loss. This exposes the finite record capacity directly.

After matching, each gold membership vector is placed on its assigned slot; unmatched active slots receive an all-zero membership target. When presence is predicted, matched slots receive positive targets and unmatched active slots receive negative targets. For a nested schema, the same assignment reindexes both axes of the gold parent–child adjacency matrix before the hierarchy loss is evaluated. The matching itself is discrete and used only to choose training correspondences; inference thresholds presence and relation scores without solving an assignment. Other set-valued heads use the same principle with costs suited to relation endpoints, classes, boxes, masks, or temporal intervals.

4 A Modular Framework

GLiFormer separates schema processing, neural prediction, and decoding (Figure 5). A task processor aligns inputs and annotations; a task head consumes shared representations; and a task decoder restores labels, source offsets, and structured outputs. Several heads and schema groups can therefore reuse one encoder pass without sharing an output vocabulary.

The framework exposes these stages as task-specific components around the shared model interface in Figure 2. Adding a task requires aligning its schema and supervision, defining its prediction quantities, and mapping those predictions back to the requested output. This separation lets heads reuse anchor and attention layers while retaining task-specific loss and decoding rules. For hierarchical structuring, for example, the decoder turns grounded fields, record memberships, and parent–child scores into nested JSON rather than generating a serialized output token by token.

Shared anchor, attention, and position representations support text and document inputs, including page geometry. Vision and audio are possible extensions of this design but are not evaluated here.

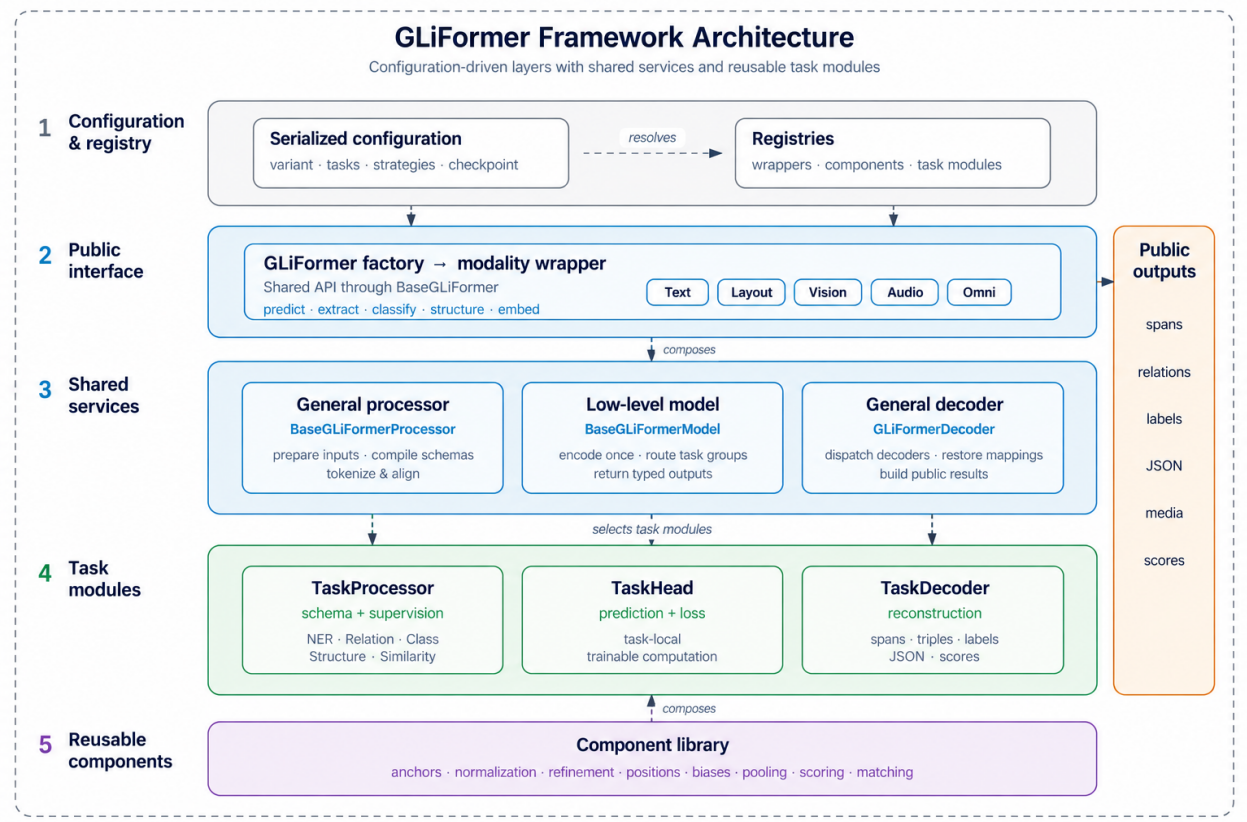

Figure 5: Modular framework architecture of GLiFormer. Task processors align sources, runtime schemas, and available supervision; the shared encoder and task heads produce predictions; task decoders restore source offsets, schema labels, and output structure. This organization supports multiple task groups per source and keeps task-specific processing and reconstruction explicit.

5 Training and Data

5.1 Training stages

GLiFormer-base starts from a DeBERTa backbone further pretrained on **100 billion diverse tokens**. We then apply broad multitask training, followed by task-focused post-training. Both supervised stages combine extraction, classification, structuring, embedding, and document-layout tasks.

Table 2 summarizes the two training mixtures: 1,357,671 examples for initial training and 372,090 for post-training.

5.2 Model and supervision

GLiFormer-base contains 264.2 million parameters, with a 12-layer DeBERTa encoder and task-specific heads. Its document pathway incorporates spatial features and optional image tokens; hierarchical structuring uses 100 record anchors with source-conditioned refinement. GLiFormer-large contains 575.6 million parameters (575,637,510 in total).

A training example can supervise any compatible subset of extraction, classification, structuring, layout processing and similarity tasks. Character spans ground entities and field values; relation

Table 2: Training examples by stage. Initial document-layout examples are repeated three times; counts reflect this repetition.

Task family	Initial training	Post-training
Classification	508,846	40,001
Span extraction	372,930	98,895
Relation extraction	167,136	29,664
Hierarchical structuring	136,867	61,874
Embedding	126,538	126,538
Document layout	45,354	15,118
Total	1,357,671	372,090

triples connect grounded mentions; class groups provide candidate labels and targets; nested objects and lists provide record-membership and parent–child targets. Only annotated tasks contribute to Equation 13. Hungarian matching aligns unordered record slots with supervision. Unresolved field values provide no grounding target, so the model learns extraction rather than unrestricted value generation.

6 Evaluation

We evaluate GLiFormer-base and GLiFormer-large on named-entity recognition, relation extraction, classification, and multilevel structuring. Scores are percentages unless stated otherwise, and dataset means are unweighted. In result tables, **bold** marks the best score and **light bold** marks the second-best score; ties share a rank. Higher quality scores and lower latency values are better.

6.1 Named-entity recognition

The NER evaluation covers 26 datasets, including the five CrossNER domains. GLiFormer-base and GLiFormer-large average 65.10 and 64.35 F1, respectively, on CrossNER (Table 3). Gemma-4-31B-IT, GPT-OSS-120B, and GPT-5-mini average 70.74, 70.24, and 69.97 F1, respectively. Published baseline scores are from Knowledgator Engineering [2026b] and Newhauser and Zaratiana [2026]. Complete GLiFormer per-dataset results appear in Appendix A.

6.2 Relation extraction

Table 4 compares relation micro-F1 with the published results of Stepanov et al. [2026]. Entity inputs are shown explicitly because gold-entity relation classification differs from end-to-end extraction. GLiFormer-base averages 18.94 across the four evaluated benchmark subsets. GLiFormer-large averages 21.33 over the supplied DocRED, CrossRE, FewRel, and CoNLL04 zero-shot evaluations. Gemma-4-31B-IT and GPT-OSS-120B average 25.08 and 23.99, respectively, over their four evaluated subsets. GLiFormer per-dataset metrics appear in Appendix B.2.

6.3 Text classification

Across 13 classification datasets (79,828 examples), GLiFormer-base achieves 72.36 mean macro-F1 and 73.83 mean accuracy. GLiFormer-large achieves 75.03 mean macro-F1, a gain of 2.67 points, while GLiNER2.5 achieves 64.89 on the same dataset suite. GPT-5-mini, GPT-OSS-120B, and Gemma-4-31B-IT report mean macro-F1 of 79.79, 77.76, and 79.69, respectively. Table 5 compares the same

Table 3: CrossNER F1 (%). Params. are in millions; NR denotes an unavailable count.

Model	Params. (M)	CrossNER F1 (%)					
		AI	Literature	Music	Politics	Science	Avg.
GLiNER							
NuNER Zero-span	448.9	60.12	64.89	**69.95**	72.73	66.42	66.82
GLiNER Multitask Large v0.5	440.2	51.05	68.96	**74.30**	**78.27**	67.29	67.97
GLiNER Large v2.1	459	53.45	57.87	67.08	64.90	62.00	61.06
GLiNER Large News v2.1	459	56.96	60.44	65.20	62.92	65.92	62.29
GLiNER2							
GLiNER2 Base	208.5	52.12	56.91	64.27	66.52	55.47	59.06
GLiNER2 Multilingual	307.1	50.31	55.06	63.06	62.47	58.31	57.84
GLiNER2.5 Base	193.6	50.69	54.56	68.96	56.41	60.85	58.29
GLiNER2.5 Multilingual	287.4	45.60	51.52	65.80	55.26	56.08	54.85
LLMs							
Gemma-4-31B-IT	31,000	63.12	**74.72**	67.28	**76.97**	**71.60**	**70.74**
GPT-OSS-120B	117,000	**64.72**	**71.36**	69.00	76.55	69.58	**70.24**
GPT-5-mini	NR	**64.99**	71.10	65.68	76.72	**71.33**	69.97
Gemma4-12B	11,959.7	58.7	61.5	64.9	68.2	60.8	62.8
Gemma-4-E4B	7,996.2	57.9	51.4	60.8	70.3	60.5	60.2
Gemma-4-E2B	5,123.2	50.3	49.0	54.7	60.0	53.0	53.4
Qwen3.5 9B	9,653.1	48.6	48.7	55.9	62.9	58.0	54.8
Qwen3.5-0.8B	873.4	14.9	21.7	19.3	25.2	24.9	21.2
Qwen3.5-2B	2,274.1	20.9	23.3	28.3	28.8	30.9	26.4
GLiFormer							
GLiFormer-base	264.2	56.38	62.60	67.65	71.36	67.51	65.10
GLiFormer-large	575.6	51.76	61.57	66.82	71.48	70.13	64.35

Table 4: Relation extraction micro-F1 (%). †: gold entities; ‡: CoNLL04 zero-shot.

Model	Entities	CoNLL04	DocRED	FewRel	CrossRE	Avg.
GLiREL†	gold	4.5	2.4	24.0	1.4	8.1
GLiNER2	predicted	32.9	11.7	20.8	6.0	17.8
GPT-5-mini	prompted	**42.4**	**18.6**	15.0	12.4	22.1
GLiNER-Relex	predicted	40.4	**31.3**	12.5	**18.1**	**25.6**
Gemma-4-31B-IT	prompted	**42.32**	17.96	30.18	9.84	25.08
GPT-OSS-120B	prompted	38.40	17.77	28.79	10.99	23.99
Gemma4-12B	prompted	34.1	14.0	15.5	7.9	17.9
Gemma-4-E4B	prompted	22.2	11.6	13.7	6.1	13.4
Gemma-4-E2B	prompted	7.4	2.9	9.6	3.5	5.9
Qwen3.5 9B	prompted	16.0	11.6	**71.6**	3.8	**25.7**
Qwen3.5-0.8B	prompted	1.8	0.1	47.2	0.6	12.4
Qwen3.5-2B	prompted	3.2	1.2	**53.5**	0.7	14.7
GLiFormer-base	predicted	32.27	10.00	23.49	10.00	18.94
GLiFormer-large	predicted	35.83‡	12.78	24.08	**12.61**	21.33

dataset set with published GLiClass, GLiNER2, and NLI baselines [Knowledgator Engineering, 2026a] and these measured runs. GLiFormer per-dataset metrics and sample counts appear in Appendix B.1.

Table 5: Mean macro-F1 (%) across 13 classification datasets. Params. are in millions; NR denotes an unavailable count.

Model	Params. (M)	Mean macro-F1
GLiClass		
GLiClass Multilang ultra	1,708.2	71.01
GLiClass Multilang mini	283.7	66.68
GLiClass Multilang edge	143.0	61.74
GLiClass Instruct large v1.0	438.7	70.99
GLiClass Instruct base v1.0	186.5	65.26
GLiClass Instruct edge v1.0	32.7	48.80
GLiNER2		
GLiNER2 large v1	486.4	66.19
GLiNER2 multi v1	307.1	62.66
GLiNER2 base v1	208.5	62.28
GLiNER2.5	193.6	64.89
NLI encoders		
BGE-M3 zero-shot v2.0	567.8	58.33
mDeBERTa-v3-base MNLI/XNLI	278.8	51.93
LLMs		
GPT-5-mini	NR	**79.79**
GPT-OSS-120B	117,000	77.76
Gemma-4-31B-IT	31,000	**79.69**
Gemma4-12B	11,959.7	77.8
Gemma-4-E4B	7,996.2	74.1
Gemma-4-E2B	5,123.2	70.8
Qwen3.5 9B	9,653.1	71.4
Qwen3.5-0.8B	873.4	45.2
Qwen3.5-2B	2,274.1	57.3
GLiFormer		
GLiFormer-base	264.2	72.36
GLiFormer-large	575.6	75.03

6.4 Multilevel text structuring

The structuring evaluations span short and long texts, repeated records, and JSON depths from 3 to at least 6. All runs use 500 examples. Each example provides a schema derived from its reference structure. F1 compares path–value pairs using order-free matching.

GLiFormer-base achieves 87.20 F1 (Table 6), with scores ranging from 85.95 to 91.41 across depth groups. GPT-5.6-luna achieves 91.96 F1 under the same order-free, boundary-tolerant metric, a difference of 4.76 percentage points overall. Its depth-specific scores range from 86.21 to 93.47; GLiFormer-base scores higher in the depth-4 and depth-5 buckets. GPT-5-mini achieves 82.56 F1, 4.64 points below GLiFormer-base. GLiFormer-large achieves 91.10 F1, with depth-specific scores from 89.94 to 95.11. All 500 predictions are valid JSON, with no inference failures and a reported JSON consistency of 1.0000. Appendix B.3 gives the depth and length breakdowns and results by

Table 6: Structuring F1 (%) by JSON depth.

Model	Depth 3	Depth 4	Depth 5	Depth 6+	Avg.
GPT-5.6-luna	**93.47**	90.24	86.21	**89.98**	**91.96**
GPT-5-mini	85.43	77.35	75.86	77.33	82.56
Qwen3.5 9B	79.2	82.4	76.4	71.3	78.5
GLiFormer-base	85.95	**91.41**	**86.89**	89.95	87.20
GLiFormer-large	**89.94**	**95.11**	**91.69**	**92.80**	**91.10**

gold record count.

6.5 Inference efficiency

We measure multilevel structuring on 40 documents, with ten examples at each depth from 3 to 6 and batch size 1. Timing includes input processing, prediction, and nested-output reconstruction. GPU results use an NVIDIA RTX PRO 6000 Blackwell (FP16); CPU results use an AMD EPYC 9B45 with eight threads (FP32).

For a generic autoregressive LLM, we add prefill time for the source, schema, and instruction at 2,000 input tokens/s to generation time for the complete reference JSON at 60 output tokens/s. These rates are illustrative deployment assumptions.

Table 7: Structuring latency by JSON depth. LLM slowdowns are analytical estimates.

	Mean tokens		GPU (ms)		CPU (ms)		Est. LLM slowdown	
Depth	Input	Output	Mean	p50	Mean	p50	vs. GPU	vs. CPU
3	456	175	**64.3**	**52.2**	**374.1**	**240.0**	48.9×	8.4×
4	540	243	**56.7**	**54.2**	**488.4**	**303.1**	76.1×	8.8×
5	771	522	**108.6**	**108.1**	**709.8**	**551.7**	83.6×	12.8×
6	947	778	**140.3**	**127.8**	**890.4**	**941.7**	95.8×	15.1×
Overall	678	429	**92.5**	**69.0**	**615.7**	**546.9**	81.0×	12.2×

GLiFormer-base averages 92.5 ms on GPU and 615.7 ms on CPU (Table 7); the respective 95th percentiles are 205.6 ms and 1,446.1 ms. The LLM scenario averages 7.49 s, including 0.34 s of prefill. It excludes queueing, network delay, and hidden reasoning, and assumes no equivalence in extraction accuracy. Document length and record count also vary with depth, so these workloads do not isolate nesting cost.

6.6 Relative performance and parameter efficiency

We aggregate the CrossNER mean F1 and classification mean macro-F1 for models evaluated on both tasks. Each task score s is min–max normalized as $(s - s_{\min})/(s_{\max} - s_{\min})$ over the comparison set, and the two normalized scores are averaged with equal weight. Figure 6 plots this relative performance against parameter count on a logarithmic axis.

GLiFormer-base and GLiFormer-large achieve relative scores of 0.836 and 0.867 with 264.2M and 575.6M parameters, respectively. Both exceed GLiNER2 Base (0.629) and GLiNER2.5 Base (0.659), and are the only plotted models in the highlighted region. Gemma4-12B reaches 0.891 with 11.96B parameters, illustrating the trade-off between aggregate performance and model size.

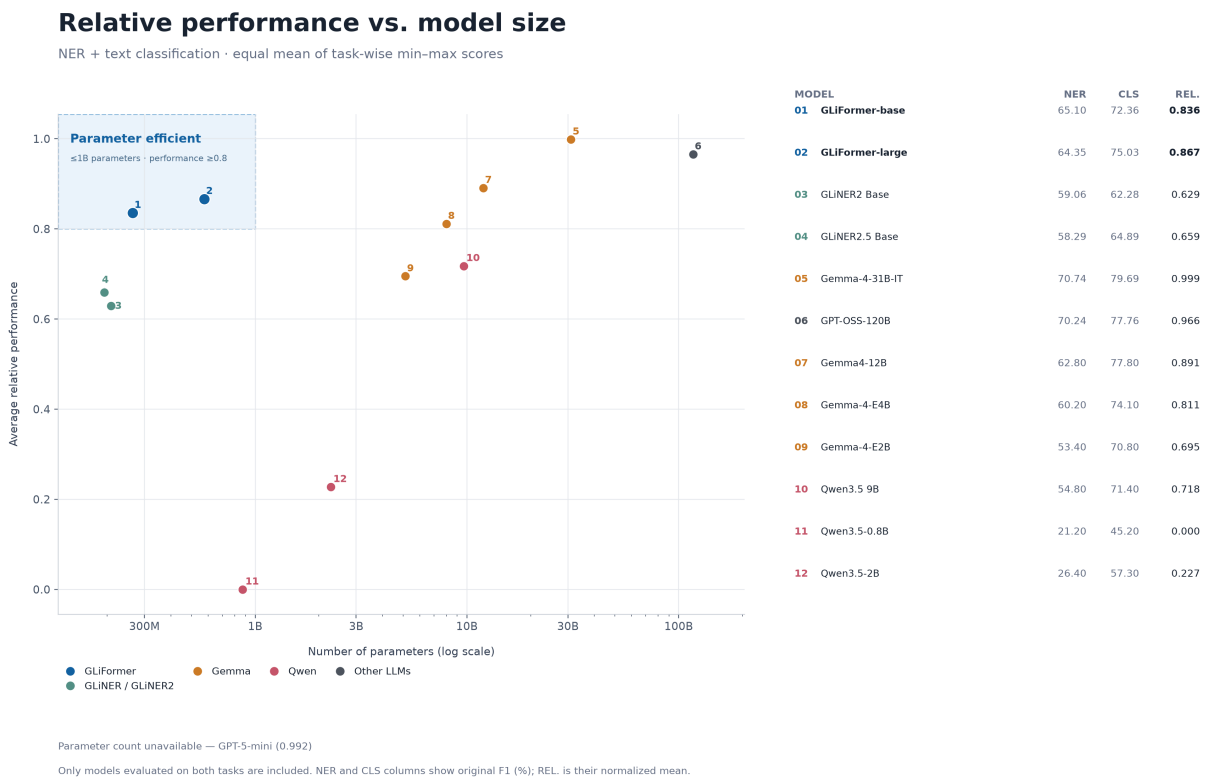

Figure 6: Average normalized NER and classification performance versus model size. The shaded region marks models with at most one billion parameters and relative performance of at least 0.8.

7 Discussion and Limitations

GLiFormer expands the diversity of tasks a single encoder can support through a common anchor interface, spanning entity recognition, relation extraction, classification, and hierarchical record formation. The joint NER and classification results extend the empirical Pareto frontier of performance and parameter count among the evaluated models (Figure 6). GLiFormer-large exceeds Gemma-4-E4B on both task aggregates with approximately 14× fewer total parameters, demonstrating more than an order of magnitude improvement in parameter efficiency for these tasks. Larger LLMs still achieve the highest aggregate scores and may be preferable for more complex use cases. Further research is needed to determine how scaling GLiFormer's backbone, anchor capacity, and training data affects this trade-off.

Hierarchical structuring also provides an efficient route to JSON extraction. Parallel field grounding, record assignment, and hierarchy prediction remove sequential output generation, while a deterministic decoder assembles nested records. GLiFormer-base achieves median latencies of 69 ms on GPU and 547 ms on CPU; the analytical autoregressive comparison indicates substantial potential savings (Table 7). By design, extracted field values come from source spans, preventing the model from hallucinating value text absent from the input. This constraint does not guarantee correct span selection, typing, record assignment, or hierarchy. When fields require generated text, such as summaries or explanations, an LLM or another generative component is still needed.

The fixed record capacity and shared context budget limit dense schemas and long documents; joint training can also introduce task interference. The model was trained with a sequence length

of 16,384 tokens. DeBERTa's relative attention motivates studying transfer to longer contexts, but reliable generalization beyond the training length remains to be established. Autoregressive latency is estimated under explicit throughput assumptions. Future experiments should compare encoders and generative models on identical schemas and documents, controlling hardware, precision, output completeness, and batching. Ablations of anchor capacity, hierarchy prediction, and the two supervised training stages would clarify their individual contributions.

The runtime-label interface also brings zero-shot schema conditioning to LayoutLM-like document understanding, combining requests for unseen field labels with spatial and optional visual evidence. More fine-grained evaluation is needed to characterize this capability across unseen schemas, document layouts, and domains, and to separate the contributions of textual, spatial, and visual features.

Extending multimodal capabilities to audio classification and image segmentation is a next step for research. The anchor interface offers a common formulation for matching audio representations to runtime labels and assigning image regions to object or mask anchors. These extensions require appropriate modality encoders, supervision, and task-specific evaluation to establish whether the breadth and efficiency observed on text tasks transfer across modalities.

8 Conclusion

We presented GLiFormer, a compact encoder framework for schema-conditioned multitask prediction. Generalized anchors and shared source representations support named-entity recognition, relation extraction, classification, and hierarchical text structuring within a common interface. The structuring head grounds field values, assigns them to unordered records, and predicts parent–child links before deterministic JSON assembly. On the 500-example multilevel evaluation, GLiFormer-base and GLiFormer-large achieve 87.20% and 91.10% F1, respectively. A separate structuring benchmark measures median GLiFormer-base latencies of 69 ms on GPU and 547 ms on CPU. Evaluations across 26 NER and 13 classification datasets further demonstrate the framework's breadth, while the joint comparison extends the empirical Pareto frontier of performance and parameter count among the evaluated models.

These results expand the range of tasks served by a single encoder and support efficient JSON extraction with field values grounded in the source. Larger generative models remain useful for complex requests and fields requiring newly generated text. Further work should study model scaling, transfer beyond the 16,384-token training length, and zero-shot document understanding with spatial and visual features. Audio classification and image segmentation offer additional directions for extending and evaluating the anchor interface.

References

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision – ECCV 2020*, volume 12346 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2020. doi: 10.1007/978-3-030-58452-8_13. URL `https://doi.org/10.1007/978-3-030-58452-8_13`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June

2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing, 2021a. URL https://arxiv.org/abs/2111.09543.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. DeBERTa: Decoding-enhanced BERT with disentangled attention. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=XPZIaotutsD.

Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. LayoutLMv3: Pre-training for document AI with unified text and image masking. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 4083–4091. Association for Computing Machinery, 2022. doi: 10.1145/3503161.3548112. URL https://doi.org/10.1145/3503161.3548112.

Knowledgator Engineering. GLiClass Multilang Mini: Model card and benchmarks. Hugging Face, 2026a. URL https://huggingface.co/knowledgator/gliclass-multilang-mini. Accessed 9 September 2026.

Knowledgator Engineering. GLiNER Multitask Large v0.5: Model card and benchmarks. Hugging Face, 2026b. URL https://huggingface.co/knowledgator/gliner-multitask-large-v0.5. Accessed 9 September 2026.

H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, 1955. doi: 10.1002/nav.3800020109. URL https://doi.org/10.1002/nav.3800020109.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017. doi: 10.1109/ICCV.2017.324. URL https://openaccess.thecvf.com/content_iccv_2017/html/Lin_Focal_Loss_for_ICCV_2017_paper.html.

Mary Newhauser and Urchade Zaratiana. Introducing GLiNER2.5: Efficient span-free information extraction with schema-driven interface. Fastino Labs, August 2026. URL https://fastino.ai/blog/gliner2-5-span-free-information-extraction.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

Ihor Stepanov and Mykhailo Shtopko. GLiNER multi-task: Generalist lightweight model for various information extraction tasks, 2024. URL https://arxiv.org/abs/2406.12925.

Ihor Stepanov, Mykhailo Shtopko, Dmytro Vodianytskyi, Oleksandr Lukashov, Alexander Yavorskyi, and Mykyta Yaroshenko. GLiClass: Generalist lightweight model for sequence classification tasks, 2025. URL https://arxiv.org/abs/2508.07662.

Ihor Stepanov, Oleksandr Lukashov, Mykhailo Shtopko, and Vivek Kalyanarangan. GLiNER-Relex: A unified framework for joint named entity recognition and relation extraction, 2026. URL https://arxiv.org/abs/2605.10108.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In Joaquin Vanschoren and Serena Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/65b9eea6e1cc6bb9f0cd2a47751a186f-Abstract-round2.html.

Urchade Zaratiana, Nadi Tomeh, Pierre Holat, and Thierry Charnois. GLiNER: Generalist model for named entity recognition using bidirectional transformer. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5364–5376, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.300. URL https://aclanthology.org/2024.naacl-long.300/.

Urchade Zaratiana, Gil Pasternak, Oliver Boyd, George Hurn-Maloney, and Ash Lewis. GLiNER2: Schema-driven multi-task learning for structured information extraction. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 130–140, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-demos.10. URL https://aclanthology.org/2025.emnlp-demos.10/.

A Named-Entity Recognition Results

Across 26 datasets (131,156 examples), GLiFormer-base achieves mean F1 of 50.45. The seven-domain transfer group—MIT Movie, MIT Restaurant, and the five CrossNER domains—averages 60.40; the remaining 19 datasets average 46.79. GLiFormer-large achieves a reported mean F1 of 50.91 on the same 26 datasets, with a standard-group average of 47.91 and a transfer-group average (reported as "zero-shot") of 59.05. Summary values retain the evaluation reports' precision; averaging the rounded per-dataset scores can differ by 0.01. Table 8 gives the full GLiFormer-base and GLiFormer-large results side by side.

Table 8: GLiFormer NER results (%). P: precision; R: recall.

		GLiFormer-base			**GLiFormer-large**		
Dataset	Examples	P	R	F1	P	R	F1
ACE 2004	812	46.54	23.50	31.23	**51.67**	**29.00**	**37.15**
ACE 2005	1,060	39.69	17.93	24.70	**44.88**	**22.42**	**29.90**
AnatEM	3,830	**31.98**	**35.80**	**33.78**	26.18	31.74	28.70
bc2gm	5,000	44.96	**54.83**	**49.41**	**45.46**	51.53	48.30
bc4chemd	26,364	37.50	65.67	47.74	**40.74**	**67.58**	**50.84**
bc5cdr	4,797	59.96	67.65	63.57	**61.92**	**72.05**	**66.60**
Broad Tweet Corpus	2,000	54.44	**71.43**	61.79	**55.55**	70.99	**62.33**
CoNLL 2003	3,453	56.72	69.39	62.42	**58.96**	**72.82**	**65.16**
CrossNER_AI	431	**64.43**	50.11	**56.38**	51.69	**51.83**	51.76
CrossNER_literature	416	**66.50**	59.13	**62.60**	63.11	**60.10**	61.57
CrossNER_music	465	**69.06**	**66.29**	**67.65**	67.48	66.17	66.82
CrossNER_politics	650	**71.08**	71.64	71.36	70.46	**72.52**	**71.48**
CrossNER_science	543	**71.86**	63.67	67.51	71.63	**68.69**	**70.13**
FabNER	2,064	26.53	**19.01**	**22.15**	**27.17**	18.47	21.99
FindVehicle	20,777	41.33	**52.97**	**46.43**	**42.11**	51.21	46.21
GENIA_NER	1,854	44.25	54.76	48.95	**48.42**	**57.23**	**52.46**
HarveyNER	1,303	9.86	**24.18**	**14.01**	**9.95**	22.54	13.81
mit-movie	2,442	61.23	**52.98**	56.80	**63.10**	**52.98**	**57.60**
mit-restaurant	1,520	**45.15**	**36.67**	**40.47**	38.88	30.25	34.03
MultiNERD	10,000	**56.43**	88.66	**68.97**	54.96	**91.91**	68.79
ncbi	940	**49.33**	61.44	**54.72**	47.63	**64.05**	54.63
Ontonotes	8,262	**30.77**	35.86	33.12	28.20	**42.18**	**33.80**
PolyglotNER	10,000	34.93	68.31	46.23	**35.52**	**70.85**	**47.31**
TweetNER7	576	40.39	48.21	43.96	**43.46**	**48.62**	**45.90**
WikiANN en	10,000	**55.22**	**58.60**	**56.86**	54.85	57.74	56.26
WikiNeural	11,597	72.29	87.15	79.03	**73.93**	**87.53**	**80.16**
Macro average (26 datasets)				50.45			**50.91**
Standard average (19 datasets)				46.79			**47.91**
Transfer average (7 datasets)				**60.40**			59.05

B Additional Evaluation Results

B.1 Text classification breakdown

Table 9 compares GLiFormer-base and GLiFormer-large on the same classification datasets. Reported mean macro-F1 weights all 13 datasets equally. Means retain the evaluation reports' precision and may differ slightly from averages of the rounded entries.

Table 9: GLiFormer classification results (%). Acc.: accuracy; Macro/Weighted: macro/weighted F1. Micro-F1 equals accuracy.

		GLiFormer-base			**GLiFormer-large**		
Dataset	Examples	Acc.	Macro	Weighted	Acc.	Macro	Weighted
CR	376	**90.43**	**89.70**	**90.45**	**91.22**	**90.45**	**91.20**
sst2	1,821	**90.44**	**90.44**	**90.44**	**92.97**	**92.97**	**92.97**
sst5	2,210	**38.05**	**31.95**	**34.81**	**44.34**	**40.33**	**43.39**
imdb	25,000	**91.27**	**91.24**	**91.24**	**93.94**	**93.93**	**93.93**
20_newsgroups	7,532	**48.67**	**47.38**	**48.56**	**57.94**	**57.18**	**58.70**
enron_spam	2,000	**96.70**	**96.70**	**96.70**	**97.95**	**97.95**	**97.95**
massive	2,974	**69.84**	**68.18**	**70.97**	**71.32**	**69.98**	**71.93**
banking77	3,080	**67.21**	**66.43**	**66.43**	**70.97**	**70.55**	**70.55**
financial_phrasebank	1,129	**95.31**	**94.90**	**95.24**	**97.25**	**96.77**	**97.24**
ag_news	7,600	**82.59**	**82.26**	**82.26**	**82.07**	**81.53**	**81.53**
emotion	2,000	**53.55**	**48.29**	**54.63**	**54.75**	**48.07**	**55.59**
cap_sotu	23,040	**51.11**	**48.63**	**51.15**	**51.74**	**49.00**	**51.15**
rotten_tomatoes	1,066	**84.62**	**84.59**	**84.59**	**86.68**	**86.68**	**86.68**
Mean macro-F1			**72.36**			**75.03**	

B.2 Relation extraction breakdown

Table 10 compares the two GLiFormer variants on DocRED, CrossRE, FewRel, and CoNLL04 zero-shot. On CoNLL04 zero-shot, typed F1 is 29.68 for base and 34.73 for large.

Table 10: GLiFormer relation extraction results (%). P: precision; R: recall; time is in seconds.

Dataset	Model	Gold	P	R	Micro-F1	Macro-F1	Time
DocRED	Base	1,186	**11.80**	**8.68**	**10.00**	**1.03**	**27.6**
	Large	6,003	**29.90**	**8.13**	**12.78**	**3.64**	**232.1**
CrossRE	Base	415	**16.22**	**7.23**	**10.00**	**10.10**	**6.9**
	Large	1,926	**22.76**	**8.72**	**12.61**	**11.72**	**14.0**
FewRel	Base	100	**18.23**	**33.00**	**23.49**	**26.48**	**5.0**
	Large	500	**21.86**	**26.80**	**24.08**	**21.40**	**12.5**
CoNLL04 zero-shot	Base	139	**21.25**	**61.15**	**31.54**	**33.59**	**0.5**
	Large	677	**38.47**	**33.53**	**35.83**	**35.32**	**3.3**

B.3 Multilevel structuring breakdown

All F1 values in this subsection use order-free matching with boundary tolerance. GLiFormer-base reports 89.10 precision, 85.66 recall, and 87.20 F1. GLiFormer-large reports 91.87 precision, 90.98 recall, and 91.10 F1 on 500 examples. These are the reported aggregate metrics; F1 is not recomputed from aggregate precision and recall. Table 11 gives each run's bucket-level F1 scores.

Table 11: GLiFormer structuring F1 (%) by record count, text length, and JSON depth.

Partition	Bucket	GLiFormer-base	GLiFormer-large
Gold records	3	**78.23**	**81.13**
	4–5	**85.81**	**88.99**
	6–10	**89.06**	**94.50**
	11+	**89.71**	**92.57**
Source words	0–64	**88.66**	**90.94**
	65–128	**90.40**	**92.33**
	129–256	**87.63**	**91.64**
	257–512	**83.98**	**90.22**
	513+	**85.02**	**90.34**
JSON depth	3	**85.95**	**89.94**
	4	**91.41**	**95.11**
	5	**86.89**	**91.69**
	6+	**89.95**	**92.80**
Overall		**87.20**	**91.10**