GLiNER2: An Efficient Multi-Task Information Extraction System with Schema-Driven Interface

Urchade Zaratiana, Gil Pasternak, Oliver Boyd
George Hurn-Maloney, Ash Lewis
Fastino AI
{uz,gil,o8,g,ash}@fastino.ai

Abstract

Information extraction (IE) is fundamental to numerous NLP applications, yet existing solutions often require specialized models for different tasks or rely on computationally expensive large language models. We present GLiNER2, a unified framework that enhances the original GLiNER architecture to support named entity recognition, text classification, and hierarchical structured data extraction within a single efficient model. Built pretrained transformer encoder architecture, GLiNER2 maintains CPU efficiency and compact size while introducing multi-task composition through an intuitive schema-based interface. Our experiments demonstrate competitive performance across extraction and classification tasks with substantial improvements in deployment accessibility compared to LLM-based alternatives. We release GLiNER2 as an open-source pip-installable library with pre-trained models and documentation at github.com/fastino-ai/GLiNER2.

1 Introduction

Information extraction (IE) (Okurowski, 1993; Weischedel et al., 1996) represents one of the most fundamental and practically important tasks in natural language processing, involving the identification and extraction of structured information from unstructured text. While large language models (OpenAI, 2024) have demonstrated remarkable capabilities across various IE tasks (Wang et al., 2025; Han et al., 2024), their deployment presents significant practical challenges that limit their accessibility and adoption. Smaller models (Touvron et al., 2023; Jiang et al., 2023; Yang et al., 2025) (eg. *Llama-2-7b*) require GPU acceleration to achieve reasonable inference speeds, making CPU-based deployment prohibitively slow for production use. The availability of GPU resources can be a barrier for organizations and researchers operating under resource constraints.

Beyond computational requirements, LLMs present additional deployment challenges. Although API costs have decreased significantly in recent years, they can pose serious privacy and security concerns (Shanmugarasa et al., 2025; Wu et al., 2025), particularly when processing sensitive data containing personally identifiable information (PII), financial records, or proprietary business information. Many organizations in the healthcare, finance, and government sectors require on-premises deployment to maintain data sovereignty and comply with regulations such as GDPR, HIPAA, or industry-specific compliance requirements (Lareo, 2023; Zhang et al., 2024; CNIL, 2024). Furthermore, the recurring costs associated with API usage may remain prohibitive for researchers, startups, and practitioners in developing countries, creating inequitable access to advanced NLP capabilities. Our goal is to address these issues specifically in the context of information extraction.

GLiNER (Zaratiana et al., 2024) was proposed to address these fundamental limitations specifically for named entity recognition (NER) tasks. GLiNER (Zaratiana et al., 2024) introduced a paradigm shift by enabling zero-shot NER using a small transformer encoder architecture trained on diverse, LLM-annotated datasets (Zhou et al., 2024; Bogdanov et al., 2024). This approach yielded remarkable results, achieving performance that matched or surpassed contemporary LLMs while running efficiently on standard CPU hardware without requiring GPUs. Furthermore, it maintains a parameter count under 500 million, enabling deployment in edge computing scenarios, resource-constrained environments, and privacy-sensitive applications. GLiNER gained particular traction in the PII redaction domain (Segbroeck, 2024; Presidio, 2024), where its combination of competitive performance, CPU efficiency, and straightforward local deployment made it an ideal solution for han-

Characteristic	GLiNER	GLiNER2	Open LLMs	Closed LLMs
Features				
Scope	NER only	Various IE & Classification	General	General
Label description	✗	✓	✓	✓
CPU Deployment	✓	✓	✗	✗
Privacy Preserving	✓	✓	✓	✗
No API Costs	✓	✓	✓	✗
Fine-tuning Support	✓	✓	✓	∼
Technical Specifications				
Parameters	195M	205M	7B-175B	Unknown
Model Architecture	Encoder	Encoder	Decoder	Decoder
Context Length	512 tokens	2048 tokens	2K-1M tokens	8K-10M tokens
Usage & Licensing				
License Type	Apache 2.0	Apache 2.0	Various	Proprietary
Commercial Use	✓	✓	∼	✓

Table 1: Comprehensive comparison across system categories. ✓= full support, ∼= partial/limited support, ✗= no support.

dling sensitive data.

Following GLiNER's release, several specialized adaptations emerged for different information extraction tasks. GLiREL (Boylan et al., 2025) extended the approach to relation extraction, while GLiClass (Knowledgator, 2025) adapted it for zero-shot text classification. Domain-specific variants included GLiNER-BioMed (Yazdani et al., 2025) for biomedical entity recognition, OpenBioNER (Cocchieri et al., 2025) for lightweight biomedical NER through entity type descriptions, and GLiDRE (Armingaud, 2025) for document-level relation extraction in French. However, each adaptation required *separate model development and deployment*, leading to fragmentation as practitioners needed multiple specialized models for comprehensive information extraction pipelines.

In this work, we introduce GLiNER2, a Python library that transforms the focused capabilities of its predecessors into a *universal information extraction system*. Rather than requiring separate models like GLiNER[1], GLiREL[2], and GLiClass[3], GLiNER2 unifies entity recognition, structured extraction, and text classification within a single architecture. GLiNER2 maintains the core efficiency, running on CPU, while dramatically expanding functionality beyond simple NER to support: entity recognition with natural language type descriptions and nested/overlapping spans, document-level classification with configurable single or multi-label outputs, and complex extraction schemas that capture hierarchical structures with parent-child relationships and repeated patterns. The library's Python API allows developers to define extraction schemas declaratively, compose multiple tasks in a single inference call, and deploy models with just a few lines of code. By unifying these capabilities, GLiNER2 replaces multiple specialized models with a *single efficient solution*. GLiNER2 is available through pip installation (`pip install gliner2`) with pre-trained weights hosted on Hugging Face, and it is released under the *Apache 2.0* license.

[1]https://github.com/urchade/GLiNER
[2]https://github.com/jackboyla/GLiREL
[3]https://huggingface.co/knowledgator/GLiClass

2 System design

Our architecture builds upon the foundational design principles of the original GLiNER (Zaratiana et al., 2024), which prompts a pretrained transformer encoder (Devlin et al., 2019; He et al., 2023) with entity types for zero-shot named entity recognition. We extend this prompting approach to handle more complex schemas that encompass multiple information extraction tasks. The core innovation lies in our unified input formulation that enables diverse extraction tasks through carefully designed prompt templates. The general input format follows:

$$\texttt{[Task Prompt]} \oplus \texttt{[SEP]} \oplus \texttt{[Input Text]}$$

where $\oplus$ denotes concatenation. The `[Task Prompt]` specifies what to extract (e.g., entity types like "person, location" or class labels like "positive, negative"), `[SEP]` is a special separator token, and `[Input Text]` is the text to be analyzed, which is a sequence of text tokens

Dataset	Task Type	# Labels	GPT-4o OpenAI (2024) >100B	GLiClass Knowledgator (2025) 190M	DeBERTa-v3 Laurer et al. (2024) 435M	GLiNER2 *Our model* 205M
SNIPS	Intent	7	0.97	0.80	0.77	0.83
Banking77	Intent	77	0.78	0.21	0.42	0.70
Amazon Intent	Intent	31	0.72	0.51	0.59	0.53
SST-2	Sentiment	2	0.94	0.90	0.92	0.86
IMDB	Sentiment	2	0.95	0.92	0.89	0.87
AG News	Topic	4	0.85	0.68	0.68	0.74
20 Newsgroups	Topic	20	0.68	0.36	0.54	0.49
Average	—	—	*0.84*	0.63	0.69	**0.72**

Table 2: Zero-shot text classification performance across various benchmarks.

$x_1, x_2, \ldots, x_N$. Complete task prompt formats for each extraction type are detailed in Appendix A.

GLiNER2 comprises the following tasks:

- **Entity Recognition:** We support entity type descriptions alongside labels, allowing richer semantic understanding through natural language definitions.
- **Hierarchical Structure Extraction:** We introduce structured schemas that capture parent-child relationships between entities and their attributes, enabling extraction of complex nested information.
- **Text Classification:** We add text classification capabilities with support for both single-label and multi-label, along with label description.
- **Task Composition:** Most significantly, we enable composition of multiple extraction tasks within a single forward pass, allowing simultaneous entity recognition, text classification, and structured extraction with shared contextual understanding.

This unified approach maintains the efficiency advantages of the original GLiNER while dramatically expanding its capabilities to handle diverse information extraction scenarios. Detailed architectural specifications and mathematical formulations are provided in Appendix A.

3 Experiments

3.1 Training Data

We trained our model on 254,334 examples, combining *real-world documents* and *synthetic data*, with balanced coverage of entity recognition, hierarchical extraction, and classification tasks. The real-world set consists of 135,698 documents from *news articles*, *Wikipedia*, *legal texts*, *PubMed abstracts*, and *ArXiv papers*, representing a wide range of writing styles and entity types. All documents were automatically annotated with *GPT-4o* using task-specific prompts and validated for quality. To address gaps and improve robustness, we generated 118,636 *synthetic examples* with *GPT-4o* targeting common business and personal use cases, including *email threads*, *text messages*, *professional documents*, *social media posts*, *transactional data*, and *domain-specific texts*; each synthetic example includes complete annotations for all tasks to support effective multi-task learning. Full dataset statistics and distribution are provided in Appendix B.1.

3.2 Results

We conducted comprehensive zero-shot evaluations on standard benchmarks for both text classification and named entity recognition to assess the effectiveness of our approach. Hierarchical structure extraction was not evaluated due to the absence of established zero-shot benchmarks for this task type, which we plan to address in future work. Details on all baseline models and evaluation protocols are provided in Appendix B.

We evaluate zero-shot classification on seven public benchmarks covering sentiment (*SST-2* (Socher et al., 2013), *IMDB* (Maas et al., 2011)), intent (*SNIPS* (Coucke et al., 2018), *Banking77* (Casanueva et al., 2020), *Amazon-Intent* (FitzGerald et al., 2022)), and topic classification (*AG News* (Zhang et al., 2016), *20 Newsgroups* (Lang, 1995)). **GLiNER2 achieves the highest average accuracy among open-source baselines**, outperforming GLiClass on five datasets and DeBERTa-v3 on three. It performs particularly well on intent

Dataset	GPT-4o	GLiNER-M	GLiNER2
AI	0.547	0.518	0.526
Literature	0.561	0.597	0.564
Music	0.736	0.694	0.632
Politics	0.632	0.686	0.679
Science	0.518	0.581	0.547
Average	0.599	0.615	0.590

Table 3: Zero-shot F1 scores on CrossNER benchmark.

classification, scoring 0.83 on *SNIPS* and 0.70 on *Banking77*, compared to DeBERTa's 0.77 and 0.42, respectively. On *Amazon-Intent* and *20 Newsgroups*, GLiNER2 trails DeBERTa-v3 slightly (by 6 and 5 points), but GLiNER2 is significantly faster as shown in Table 4. On sentiment benchmarks, GLiNER2 scores 0.86–0.87, close to DeBERTa-v3's 0.89–0.92 and within 10 points of GPT-4o. While GPT-4o leads across all tasks, this superior performance is expected given its substantially larger scale and extensive pretraining on diverse text corpora. Overall, GLiNER2 offers competitive accuracy across a range of classification settings and consistently closes the gap between task-specific baselines and large proprietary models.

For NER evaluation, we use the *CrossNER* benchmark (Liu et al., 2020), which measures zero-shot generalization across five specialized domains: *AI*, *Literature*, *Music*, *Politics*, and *Science*. As shown in Table 3, **GLiNER2 closely matches GPT-4o in overall F1 score** (0.590 vs. 0.599) and achieves higher scores in *AI* (0.526 vs. 0.547) and *Literature* (0.564 vs. 0.561). While GLiNER2 trails GLiNER-M in categories like *Science* and *Music*, it maintains strong performance in *Politics* (0.679), suggesting robustness across diverse entity types. Considering that GLiNER2 is a general-purpose model supporting multiple tasks, this level of NER performance with only modest drop-offs compared to a dedicated entity recognition system, demonstrates the effectiveness of our unified architecture.

3.3 Efficiency

We evaluate GLiNER2's computational efficiency by measuring inference latency on text classification tasks across different numbers of labels. All models are evaluated on *CPU* except GPT-4o, which uses the *OpenAI API*. Table 4 presents latency measurements in milliseconds for varying numbers of classification labels. GLiNER2 demonstrates **strong computational efficiency**, achieving latency comparable to GLiClass while providing significantly better performance than DeBERTa-based zero-shot classification. The key advantage becomes evident when comparing against DeBERTa, which performs a *separate forward pass for each label*, resulting in *linear scaling* with the number of labels and substantially higher latency (**6.8× slower** with 20 labels). In contrast, GLiNER2 processes *all labels simultaneously* in a single forward pass, maintaining consistent performance regardless of label count. Both GLiNER2 and GLiClass achieve approximately **2.6× speedup over GPT-4o** while running on standard CPU hardware, demonstrating the practical advantages of compact, specialized models for production deployment scenarios where *latency and computational resources* are critical considerations.

#Labels	GPT-4o	DeBERTa	GLiClass	GLiNER2
5	358	1714	**137**	**130**
10	382	3404	**131**	**132**
20	425	6758	**140**	**163**
50	463	16897	**190**	**208**
Speedup	1.00×	0.10×	2.75×	2.62×

Table 4: CPU Latency (ms) comparison for text classification with varying number of labels.

4 Artifacts

4.1 Python Package

We provide a Python package that makes GLiNER2 accessible through an intuitive API. The `gliner2` library can be easily installed via pip and provides seamless integration with the Hugging Face ecosystem for model distribution and loading. The model can be loaded using the standard `.from_pretrained` method, with weights hosted on Hugging Face Hub for convenient access.

```python
# Installation: pip install gliner2
from gliner2 import GLiNER2

# Load from Hugging Face
extractor = GLiNER2.from_pretrained("gliner/gliner2-base")
```

Figure 1: Model loading.

Named Entity Recognition Named entity recognition can be performed through multiple approaches to accommodate different use cases and complexity levels. The simplest method requires only the input text and a list of target entity types. Moreover, users can provide entity types with natural language descriptions using a dictionary format, where keys represent entity types and values contain descriptive text that helps the model better understand the extraction target. The process and various usage patterns are illustrated in Figure 2.

```
text = "Apple Inc. CEO Tim Cook announced new products in Cupertino."

entities = ["company", "person", "location", "product"]
results = extractor.extract_entities(text, entities)
# {'entities': {'company': ['Apple Inc.'],
#               'person': ['Tim Cook'],
#               'location': ['Cupertino']}}

entity_descriptions = {
    "company":  "Business organizations and corporations",
    "person":   "Names of individuals including executives",
    "location": "Geographical places including cities"
}
results = extractor.extract_entities(text, entity_descriptions)
```

Figure 2: GLiNER2 for Named Entity Recognition with simple and enhanced approaches

Hierarchical Structure Extraction Hierarchical structure extraction is performed by defining a schema as shown in Figure 3. The schema defines a parent entity (termed a *structure*) containing multiple child fields using GLiNER2's field specification syntax. Each field follows the pattern `field_name::type::description`, where `type` specifies either `str` for single values or `list` for multiple values. Fields may incorporate choice constraints through the format `field_name::[option1|option2]::type`, exemplified by the category field restricted to *electronics*, *software*, or *hardware*.

```
text = "The new MacBook Pro costs $1999..."

product_schema = {
    "product": [
        "name::str::Product name and model",
        "price::str::Product cost",
        "features::list::Key product features",
        "category::[electronics|software|hardware]::str"
    ]
}
results = extractor.extract_json(text, product_schema)
```

Figure 3: Hierarchical structure extraction with field constraints and descriptions

The framework supports multiple structures within a single schema for complex extraction scenarios. For instance, Figure 4 shows how users can define two structures, *product* and *company*, in a single query. This enables simultaneous extraction of product details (*name* and *price*) alongside company information (*name* and *headquarters*), all processed efficiently in a single forward pass.

Text Classification Like NER, text classification functionality provides both streamlined and highly customizable interfaces to accommodate various application requirements. For quick deployment, users need only provide the input text and a dictionary mapping task names (e.g., "sentiment") to

```
text = "Apple Inc., based in Cupertino..."

multi_schema = {
    "product": [
        "name::str",
        "price::str"
    ],
    "company": [
        "name::str",
        "headquarters::list"
    ]
}
results = extractor.extract_json(text, multi_schema)
```

Figure 4: Composing multiple hierarchical structures in a single schema

lists of classification labels, as shown in the first example of Figure 5. For more sophisticated applications, the library supports extensive customization options including label descriptions and multi-label classification capabilities. When multi-label classification is enabled, the model applies sigmoid activation to allow multiple simultaneous label assignments, while single-label tasks use softmax normalization for mutually exclusive predictions.

```
text = "This movie was absolutely fantastic! Great acting and plot."

labels = ["positive", "negative", "neutral"]
results = extractor.classify_text(text, {"sentiment": labels})
# {'sentiment': 'positive'}

tasks = {
    "aspects": {
        "labels": ["acting", "plot", "visuals", "music"],
        "multi_label": True,
        "descriptions": {
            "acting":  "Quality of character performances",
            "plot":    "Story structure and narrative",
            "visuals": "Cinematography and visual effects",
            "music":   "Soundtrack and audio design"
        }
    }
}
results = extractor.classify_text(text, tasks)
# {'aspects': ['acting', 'plot']}
```

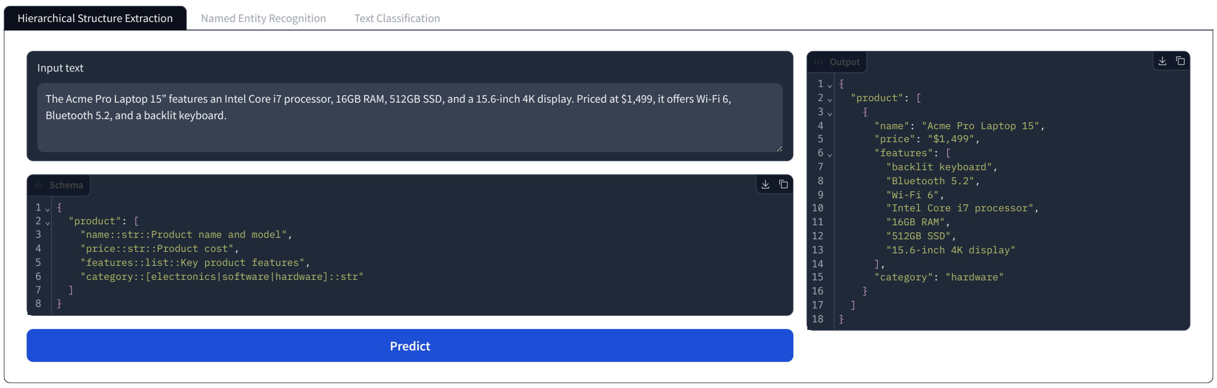
Figure 5: Text classification with simple and advanced configuration options

The library supports multiple classification tasks within a single call, as demonstrated in Figure 6. Each classification task can be independently customized with features such as label descriptions and multi-label settings.

```
results = extractor.classify_text(text, {
    "sentiment": ["positive", "negative", "neutral"],
    "genre":     ["comedy", "drama", "action", "thriller"]
})
```

Figure 6: Simultaneous multi-task classification.

Task Composition A key feature of the library is its ability to efficiently compose multiple extraction tasks within a single unified framework. Figure 8 demonstrates how to construct a comprehensive schema that combines entity recognition, text classification, and structured extraction seamlessly in one inference call.



Input text

The Acme Pro Laptop 15" features an Intel Core i7 processor, 16GB RAM, 512GB SSD, and a 15.6-inch 4K display. Priced at $1,499, it offers Wi-Fi 6, Bluetooth 5.2, and a backlit keyboard.

Schema

```
{
  "product": [
    "name::str::Product name and model",
    "price::str::Product cost",
    "features::list::Key product features",
    "category::[electronics|software|hardware]::str"
  ]
}
```

Output

```
{
  "product": [
    {
      "name": "Acme Pro Laptop 15",
      "price": "$1,499",
      "features": [
        "backlit keyboard",
        "Bluetooth 5.2",
        "Wi-Fi 6",
        "Intel Core i7 processor",
        "16GB RAM",
        "512GB SSD",
        "15.6-inch 4K display"
      ],
      "category": "hardware"
    }
  ]
}
```

Figure 7: GLiNER2 Gradio demo interface showing hierarchical structure extraction.

```
# Multi-task extraction in a single forward pass
schema = (extractor.create_schema()
    # Named Entity Recognition
    .entities(["person", "company", "product", "location", "price"])

    # Text Classification
    .classification("sentiment", ["positive", "negative", "neutral"])
    .classification("urgency", ["low", "medium", "high"])

    # Hierarchical Structure Extraction
    .structure("product_info")
        .field("name", dtype="str", description="Product name")
        .field("price", dtype="str", description="Product cost")
        .field("features", dtype="list", description="Key features")
        .field("company", dtype="str", description="Manufacturer")
)

# Extract all information simultaneously
results = extractor.extract(text, schema)
```

Figure 8: Comprehensive task composition combining all extraction types

4.2 Interactive Gradio Demo

We provide a web-based demonstration interface that allows users to interact with GLiNER2 without writing code. The demo enables real-time experimentation with entity types, classification labels, descriptions and other parameters. The interface consists of three tabs corresponding to GLiNER2's core capabilities. Figure 7 shows the hierarchical structure extraction tab, where users can define schemas with multiple fields and data types to extract structured information from text.

5 Related Work

Several frameworks have addressed information extraction tasks across different domains and approaches.

Traditional NLP Libraries: spaCy (Honnibal et al., 2020), Stanford CoreNLP (Manning et al., 2014), Stanza (Qi et al., 2020) provide comprehensive toolkits for named entity recognition, part-of-speech tagging, and dependency parsing. However, these frameworks require separate models for each task and lack unified architectures, and often does not generalize to unseen labels.

LLM-based Extraction: XNLP (Fei et al., 2024) demonstrated using large language models for diverse IE tasks through prompting strategies, while NuExtract (NuMind, 2024) focused on fine-tuning for JSON extraction. These approaches achieve strong performance but require significant computational resources and GPU inference.

Encoder-based Approaches: GLiNER (Zaratiana et al., 2024) introduced an efficient paradigm leveraging pretrained encoders fine-tuned on synthetic data for zero-shot named entity recognition, achieving fast CPU inference with competitive accuracy. This approach inspired subsequent work including GLiClass (Knowledgator, 2025) for text classification and GLiREL (Boylan et al., 2025) for zero-shot relation extraction. GLiNER2 extends this line of work by integrating multiple tasks within a single efficient framework, enabling multi-task composition while maintaining the computational advantages of compact encoder-based models.

6 Conclusion

We presented GLiNER2, which unifies entity recognition, text classification, and hierarchical extraction in a single CPU-efficient model. Unlike existing approaches requiring separate models per task, GLiNER2 enables multi-task extraction through declarative schemas while maintaining under 500M parameters for practical deployment. We release GLiNER2 as an open-source Python library under Apache 2.0 license, with pre-trained weights on Hugging Face. By combining efficiency with versatility, we hope our library makes advanced information extraction accessible for both research and production use.

References

Robin Armingaud. 2025. Glidre: Modèle généraliste pour l'extraction de relations à l'échelle de documents. In *EGC-Atelier TextMine*.

Sergei Bogdanov, Alexandre Constantin, Timothée Bernard, Benoit Crabbé, and Etienne P Bernard. 2024. NuNER: Entity recognition encoder pre-training via LLM-annotated data. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11829–11841, Miami, Florida, USA. Association for Computational Linguistics.

Jack Boylan, Chris Hokamp, and Demian Gholipour Ghalandari. 2025. GLiREL - generalist model for zero-shot relation extraction. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8230–8245, Albuquerque, New Mexico. Association for Computational Linguistics.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 38–45, Online. Association for Computational Linguistics.

CNIL. 2024. AI and GDPR: the CNIL publishes new recommendations to support responsible innovation.

Alessio Cocchieri, Giacomo Frisoni, Marcos Martínez Galindo, Gianluca Moro, Giuseppe Tagliavini, and Francesco Candoli. 2025. OpenBioNER: Lightweight Open-Domain Biomedical Named Entity Recognition Through Entity Type Description. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 818–837, Albuquerque, New Mexico. Association for Computational Linguistics.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *Preprint*, arXiv:1805.10190.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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. Association for Computational Linguistics.

Hao Fei, Meishan Zhang, Min Zhang, and Tat-Seng Chua. 2024. XNLP: An Interactive Demonstration System for Universal Structured NLP. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 19–30, Bangkok, Thailand. Association for Computational Linguistics.

Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, Swetha Ranganath, Laurie Crist, Misha Britan, Wouter Leeuwis, Gokhan Tur, and Prem Natarajan. 2022. Massive: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages. *Preprint*, arXiv:2204.08582.

Ridong Han, Chaohao Yang, Tao Peng, Prayag Tiwari, Xiang Wan, Lu Liu, and Benyou Wang. 2024. An empirical study on information extraction using large language models. *Preprint*, arXiv:2305.14450.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. DeBERTav3: Improving deBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Knowledgator. 2025. gliclass-base-v1.0-lw. `https://huggingface.co/knowledgator/gliclass-base-v1.0-lw`.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *International Conference on Machine Learning*.

Xabier Lareo. 2023. Large language models (llm). `https://www.edps.europa.eu/data-protection/technology-monitoring/techsonar/large-language-models-llm_en`. European Data Protection Supervisor, TechSonar.

Moritz Laurer, Wouter van Atteveldt, Andreu Casas, and Kasper Welbers. 2024. Building efficient universal classifiers with natural language inference. *Preprint*, arXiv:2312.17543.

Zihan Liu, Yan Xu, Tiezheng Yu, Wenliang Dai, Ziwei Ji, Samuel Cahyawijaya, Andrea Madotto, and Pascale Fung. 2020. Crossner: Evaluating cross-domain named entity recognition. *Preprint*, arXiv:2012.04373.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

NuMind. 2024. Nuextract: A framework for structured data extraction. https://numind.ai/blog/nuextract-a-foundation-model-for-structured-extraction.

Mary Ellen Okurowski. 1993. Information extraction overview. In *TIPSTER TEXT PROGRAM: PHASE I: Proceedings of a Workshop held at Fredricksburg, Virginia, September 19-23, 1993*, pages 117–121, Fredericksburg, Virginia, USA. Association for Computational Linguistics.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Presidio. 2024. Using gliner as an external pii model.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.

Maarten Van Segbroeck. 2024. Gliner models for pii detection through fine-tuning on gretel-generated synthetic documents.

Yashothara Shanmugarasa, Ming Ding, Mahawaga Arachchige Pathum Chamikara, and Thierry Rakotoarivelo. 2025. Sok: The privacy paradox of large language models: Advancements, privacy risks, and mitigation.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Shuhe Wang, Xiaofei Sun, Xiaoya Li, Rongbin Ouyang, Fei Wu, Tianwei Zhang, Jiwei Li, Guoyin Wang, and Chen Guo. 2025. GPT-NER: Named entity recognition via large language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4257–4275, Albuquerque, New Mexico. Association for Computational Linguistics.

Ralph Weischedel, Sean Boisen, Daniel Bikel, Robert Bobrow, Michael Crystal, William Ferguson, Allan Wechsler, and The PLUM Research Group. 1996. Progress in information extraction. In *TIPSTER TEXT PROGRAM PHASE II: Proceedings of a Workshop held at Vienna, Virginia, May 6-8, 1996*, pages 127–138, Vienna, Virginia, USA. Association for Computational Linguistics.

Yuhao Wu, Evin Jaff, Ke Yang, Ning Zhang, and Umar Iqbal. 2025. An in-depth investigation of data collection in llm app ecosystems. *Preprint*, arXiv:2408.13247.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Anthony Yazdani, Ihor Stepanov, and Douglas Teodoro. 2025. Gliner-biomed: A suite of efficient models for open biomedical named entity recognition. *Preprint*, arXiv:2504.00676.

Urchade Zaratiana, Nadi Tomeh, Pierre Holat, and Thierry Charnois. 2024. 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. Association for Computational Linguistics.

Dawen Zhang, Pamela Finckenberg-Broman, Thong Hoang, Shidong Pan, Zhenchang Xing, Mark Staples, and Xiwei Xu. 2024. Right to be forgotten in the era of large language models: Implications, challenges, and solutions. *Preprint*, arXiv:2307.03941.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2016. Character-level convolutional networks for text classification. *Preprint*, arXiv:1509.01626.

Wenxuan Zhou, Sheng Zhang, Yu Gu, Muhao Chen, and Hoifung Poon. 2024. UniversalNER: Targeted distillation from large language models for open named entity recognition. In *The Twelfth International Conference on Learning Representations*.

A Architecture Details

Special Token Vocabulary Our architecture employs a set of learned special tokens, each serving a specific semantic role:

- `[P]` (Prompt): Marks the beginning of task specifications, signaling the model to interpret subsequent tokens as task metadata
- `[E]` (Entity): Precedes each entity type in NER tasks to create distinct embeddings for entity categories
- `[C]` (Child/Component): Indicates attribute fields in hierarchical structures and establishes parent-child relationships
- `[L]` (Label): Denotes classification options, with each label receiving a unique embedding for scoring
- `[SEP]` (Separator): Delimits different input segments to prevent information leakage between task specifications and content.

These tokens are randomly initialized and learned during training, allowing the model to develop task-specific representations.

Named Entity Recognition NER tasks follow the input format:

`[P] entities ([E]` e_1 `[E]` e_2 `...` `[E]` e_n`)` `[SEP]` $x_1, x_2, \ldots, x_N$

During extraction, each `[E]` token generates an embedding representing its entity type. The model creates representations for all possible text spans up to a maximum width, then computes matching scores between span-entity pairs using:

$$\text{score}(s_i, e_j) = \text{sim}(\mathbf{h}_{s_i}, \mathbf{h}_{e_j}) \qquad (1)$$

where $\mathbf{h}_{s_i}$ is the span representation, $\mathbf{h}_{e_j}$ is the entity type embedding, and $\text{sim}(\cdot, \cdot)$ is the dot product with sigmoid activation.

For example, given `[P] entities ([E] person [E] location)` and text *"John works in Paris"*, all span candidates (e.g., *"John"*, *"works"*, *"Paris"*, *"works in"*) are scored against the entity type embeddings (i.e., representations of each [E] token). Spans with a predicted probability above 0.5 for any entity type are selected as entities.

Hierarchical Structure Extraction Hierarchical extraction uses the format:

`[P]` *`parent`* `([C]` a_1 `[C]` a_2 `...` `[C]` a_m`)` `[SEP]` $x_1, x_2, \ldots, x_N$

The process operates in two stages. First, an MLP processes the `[P]` token embedding to predict the number K of parent entity instances in the text. This MLP performs 20-class classification (for counts 0-19), trained using the ground-truth instance counts during training. Then, the model generates K distinct representations for each attribute by conditioning the `[C]` token embeddings on learned occurrence ID embeddings. Specifically, for each instance $k \in \{1, ..., K\}$, the model combines the base `[C]` embeddings with occurrence-specific embeddings learned during training, producing unique representations for each instance-attribute pair. These $K \times m$ representations are matched against text spans using the same scoring mechanism as NER, ensuring each instance maintains separate attribute values. Consider the structured extraction task:

`[P] product ([C] name [C] price)`

Given input text: *"iPhone costs $999. Galaxy is $899."* the model processes this in three steps:

1. **Count Prediction**: The MLP count predictor processes the `[P]` token embedding and outputs $K = 2$, indicating two product instances are present in the text.
2. **Representation Generation**: The count embedding layer generates K sets of conditioned representations for each attribute field. This produces two distinct embeddings for `[C]` name and two for `[C]` price, with each pair corresponding to one product instance.
3. **Span Extraction**: Each conditioned representation computes similarity scores with all possible text spans as for NER. The model selects the highest-scoring spans for each field while maintaining instance coherence:
   - Instance 1: {name: "iPhone", price: "$999"}
   - Instance 2: {name: "Galaxy", price: "$899"}

This parallel processing enables efficient extraction of multiple structured entities while preserving the semantic relationships between fields within each instance.

Text Classification Classification tasks use the format:

`[P]` *task* (`[L]` ℓ_1 `[L]` ℓ_2 ... `[L]` ℓ_k) `[SEP]` $x_1, x_2, \ldots, x_N$

Each `[L]` token produces a label-specific embedding that is refined through a classification head. Specifically, for each label ℓ_i, the model computes:

$$\text{logit}_i = \text{MLP}(\mathbf{h}_{\ell_i}) \tag{2}$$

where $\mathbf{h}_{\ell_i}$ is the contextualized embedding from the `[L]` token for label ℓ_i, and MLP is a multi-layer perceptron that projects these embeddings to scalar logits representing label-text compatibility. Single-label tasks apply softmax over all logits to select the highest-probability label, while multi-label scenarios use sigmoid activation on each logit independently. Consider the text classification task:

`[P] sentiment ([L] positive [L] negative [L] neutral)`

Given input text: *"This movie is amazing!"*. The model processes this in three steps:

1. **Label Embedding Generation**: Each `[L]` token creates a distinct embedding for its corresponding label (*positive*, *negative*, *neutral*).

2. **Classification Head**: The label embeddings are projected through an MLP to produce classification logits, which are then normalized using softmax activation for single-label prediction.

3. **Label Selection**: The model selects the highest-scoring label, predicting *"positive"* for the given input text.

For multi-label scenarios, sigmoid activation replaces softmax, allowing multiple labels to be selected simultaneously.

Task Composition Multiple tasks can be composed for efficient multi-task inference using:

`[Task`$_1$`]` $\oplus$ `[SEP]` $\oplus$ `[Task`$_2$`]` $\oplus$ ... $\oplus$ `[SEP]` $\oplus$ `[`$x_1, x_2, \ldots, x_N$`]`

This enables simultaneous execution of multiple extraction tasks in a single forward pass. For instance, combining NER and sentiment classification on "Steve Jobs loved the iPhone" extracts entities {person: ["Steve Jobs"], product: ["iPhone"]} and sentiment "positive" in one computation, improving efficiency over separate model runs.

B Experimental setup

Baselines We evaluate our approach against several strong baselines. As an upper bound, we use GPT-4o across all tasks. For classification tasks, we compare against two state-of-the-art open-source models with comparable parameter counts: (1) GLiClass (*knowledgator/gliclass-base-v1.0*) (Knowledgator, 2025), a classification-specific adaptation of GLiNER, and (2) DeBERTa-v3-base-zeroshot (*MoritzLaurer/deberta-v3-large-zeroshot-v2.0*) (Laurer et al., 2024), the de facto standard for zero-shot classification on Hugging Face. For NER tasks, we use GLiNER-M performance as reported in Zaratiana et al. (2024), which represents the current state-of-the-art for generalist entity recognition.

Hyperparameters We train our model for 5 epochs using the AdamW optimizer with differential learning rates: 2×10^{-5} for task-specific layers and 1×10^{-5} for the encoder backbone. This differential approach allows fine-tuning of pretrained representations while enabling faster adaptation of task-specific components. We apply weight decay of 0.01 for regularization and gradient clipping at 1.0 to ensure training stability. The learning rate schedule includes 1,000 warmup steps with linear scaling. Table 5 summarizes the training configuration.

Hyperparameter	**Value**
Epochs	5
Optimizer	AdamW
Learning rate (backbone)	1×10^{-5}
Learning rate (task layers)	2×10^{-5}
Weight decay	0.01
Warmup steps	1,000
Gradient clipping	1.0

Table 5: Training hyperparameters used across all experiments.

B.1 Training Data

Our training dataset comprises 254,334 examples combining real-world texts and synthetic data. Table 6 shows the distribution across domains.

Real-world data (135,698 examples) was collected from news articles, Wikipedia, legal documents, ArXiv papers, and PubMed abstracts. All texts were annotated using GPT-4o for entity recognition, hierarchical extraction, and classification

Domain	**Count**
Real-world Data	
Law	19,798
PubMed	16,400
Wikipedia	17,909
ArXiv	7,135
News	74,456
Synthetic Data	
Mixed Domains	118,636
Total	254,334

Table 6: Training data distribution across domains.

tasks. Synthetic data (118,636 examples) was generated using GPT-4o to cover practical use cases including emails, text messages, resumes, social media posts, e-commerce orders, banking records, and sports commentary. Each example includes annotations for all applicable task types.