GLiNER2: 一个高效的多任务信息抽取系统，具有基于架构的界面

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

Abstract

信息抽取（IE）是众多自然语言处理（NLP）应用的基础，然而，现有的解决方案往往需要针对不同任务的专用模型或依赖计算量昂贵的大型语言模型。我们提出 GLiNER2，一个统一的框架，增强了原始 GLiNER 架构，以支持在单一高效模型中进行命名实体识别、文本分类和分层结构数据抽取。GLiNER2 建立在预训练的变压器编码器架构上，保持了 CPU 效率和紧凑的体积，同时通过直观的基于模式的接口引入多任务组合。我们的实验展示了在抽取和分类任务上的竞争性表现，与基于 LLM 的替代方案相比，在部署可访问性方面有显著改善。我们将 GLiNER2 作为一个开源的 pip 可安装库发布，其中包含预训练模型和文档，在 github.com/fastino-ai/GLiNER2 上提供。

1 介绍

信息抽取 (IE) (Okurowski, 1993; Weischedel et al., 1996) 是自然语言处理中最基本且实际重要的任务之一，涉及从非结构化文本中识别和提取结构化信息。虽然大型语言模型 (OpenAI, 2024) 在各种 IE 任务中表现出显著的能力 (Wang et al., 2025; Han et al., 2024)，但它们的部署存在重大实用性挑战，限制了其可访问性和采用。较小的模型 (Touvron et al., 2023; Jiang et al., 2023; Yang et al., 2025)（例如 Llama-2-7b）需要 GPU 加速才能实现合理的推理速度，使得基于 CPU 的部署在生产环境中速度慢得令人望而却步。对于在资源受限条件下运行的组织和研究人员来说，GPU 资源的可用性可能成为一道障碍。

除了计算需求，LLMs 还存在额外的部署挑战。尽管 API 成本近年来已经显著降低，但当处理包含个人身份信息（PII）、金融记录或专有商业信息的敏感数据时，它们可能会产生严重的隐私和安全问题。许多医疗、金融和政府部门的组织需要在本地部署，以维护数据主权并遵循 GDPR、HIPAA 或特定行业合规性要求。此外，与 API 使用相关的经常性成本可能仍然是研究人员、初创企业和发展中国家的从业者难以承受的，导致无法公平地获取先进的 NLP 能力。我们的目标是专门在信息提取的背景下解决这些问题。

GLiNER (Zaratiana et al., 2024) 被提议专门用于解决命名实体识别（NER）任务的这些基本限制。GLiNER (Zaratiana et al., 2024) 通过使用在多样化、由 LLM 注释的数据集上训练的小型 transformer 编码器架构，实现了零样本 NER，带来了一种范式转变。该方法取得了显著的成果，性能与当代 LLM 相匹配或超越，同时在标准 CPU 硬件上高效运行，无需 GPU。此外，它保持了不到 5 亿的参数数量，从而能够在边缘计算场景、资源受限环境和隐私敏感应用中进行部署。GLiNER 在 PII 编辑领域 (Segbroeck, 2024; Presidio, 2024) 获得了特别的关注，因为其竞争性的性能、CPU 效率和简单的本地部署，使其成为处理敏感数据的理想解决方案。

在 GLiNER 发布之后，出现了几个针对不同信息抽取任务的专门适应版本。GLiREL (Boylan et al., 2025) 将该方法扩展到关系抽取，而 GLiClass (Knowledgator, 2025) 则将其应用于零样本文本分类。特定领域的变种包括用于生物医学实体识别的 GLiNER-BioMed (Yazdani et al., 2025)，通过实体类型描述进行轻量级生物医学实体识别的 OpenBioNER (Cocchieri et al., 2025)，以及用于法语文档级关系抽取的 GLiDRE (Armingaud, 2025)。然而，每种适应都需要单独的模型开发和部署，导致了碎片化，因为实践者需要多个专门的模型来实现全面的信息抽取管道。

在这项工作中，我们介绍了 GLiNER2，这是一个 Python 库，可以将其前身的专注功能转变为一个通用信息提取系统。与需要像 GLiNER [1]、GLiREL [2] 和 GLiClass [3] 这样的独立模型不同，GLiNER2 将实体识别、结构化提取和文本分类统一在一个架构中。GLiNER2 保留了核

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

心效率，运行在 CPU 上，同时将功能大幅扩展到超越简单 NER，支持：具有自然语言类型描述和嵌套/重叠跨度的实体识别、具有可配置单标签或多标签输出的文档级分类，以及捕获具有父子关系和重复模式的层次结构的复杂提取模式。该库的 Python API 允许开发人员以声明性方式定义提取模式，在单次推理调用中组合多个任务，并仅通过几行代码部署模型。通过统一这些功能，GLiNER2 用一个高效的解决方案取代了多个专业模型。GLiNER2 可以通过 pip 安装（`pip install gliner2`）获得，预训练权重托管在 Hugging Face 上，并在 Apache 2.0 许可证下发布。

2 系统设计

我们的架构基于原始 GLiNER (Zaratiana et al., 2024) 的基础设计原则，使用实体类型来提示预训练变压器编码器 (Devlin et al., 2019; He et al., 2023) 以进行零次命名实体识别。我们扩展了这种提示方法，以处理包含多个信息提取任务的更加复杂的模式。核心创新在于我们统一的输入公式，通过精心设计的提示模板能够实现多样的信息提取任务。一般输入格式如下：

`[Task Prompt]` $\oplus$ `[SEP]` $\oplus$ `[Input Text]`

其中 $\oplus$ 表示连接。`[Task Prompt]` 指定要提取的内容（例如，像"person, location" 这样的实体类型或者像"positive, negative" 这样的类别标签），`[SEP]` 是一个特殊的分隔符标记，而 `[Input Text]` 是待分析的文本，它是一个文本标记序列 $x_1, x_2, \ldots, x_N$ 。每种提取类型的完整任务提示格式详见附录 A 。

GLiNER2 包含以下任务：

- 实体识别：我们支持实体类型描述以及标签，通过自然语言定义允许更丰富的语义理解。
- 层次结构提取：我们引入结构化的模式，以捕捉实体及其属性之间的父子关系，从而能够提取复杂的嵌套信息。
- 文本分类：我们增加了文本分类功能，支持单标签和多标签，并附带标签描述。
- 任务组合：最重要的是，我们在单次前向传递中实现了多个抽取任务的组合，允许在共享的上下文理解下同时进行实体识别、文本分类和结构化抽取。

这种统一的方法保持了原始 GLiNER 的效率优势，同时极大地扩展了其处理多样信息提取场景的能力。详细的架构规格和数学公式在附录 A 中提供。

3 实验

3.1 训练数据

我们在 254,334 个示例上训练了我们的模型，这些示例结合了真实世界的文档和合成数据，平衡覆盖了实体识别、层次提取和分类任务。真实世界的数据集包括来自新闻文章、维基百科、法律文本、PubMed 摘要和 ArXiv 论文的 135,698 个文档，代表了广泛的写作风格和实体类型。所有文档均使用 GPT-4o 通过任务特定的提示自动注释，并经过质量验证。为了弥补差距并提高鲁棒性，我们通过 GPT-4o 生成了 118,636 个合成示例，针对常见的商业和个人用例，包括电子邮件线程、短信、专业文档、社交媒体帖子、交易数据和领域特定文本；每个合成示例都包括所有任务的完整注释，以支持有效的多任务学习。完整的数据集统计和分布见附录 A.1 。

3.2 结果

我们在标准基准上进行了针对文本分类和命名实体识别的全面零样本评估，以评估我们方法的有效性。由于缺乏针对这种任务类型的既定零样本基准，我们未对层次结构提取进行评估，这个问题将在未来的工作中解决。所有基准模型和评估协议的详细信息在附录 **??** 中提供。

我们在七个公共基准上评估了零样本分类，这些基准涵盖了情感（SST-2、IMDB）、意图（SNIPS、Banking77、Amazon-Intent）和主题分类（AG News、20 Newsgroups）。GLiNER2 在开源基准间的平均准确率最高，在五个数据集上超过了 GLiClass，在三个数据集上超过了 DeBERTa-v3。它在意图分类方面表现尤为出色，在 SNIPS 数据集上得分 0.83，在 Banking77 上得分 0.70，而 DeBERTa 分别为 0.77 和 0.42。在 Amazon-Intent 和 20 Newsgroups 上，GLiNER2 稍微落后于 DeBERTa-v3（分别低 6 和 5 分），但是如表 3 所示，GLiNER2 显著更快。在情感基准上，GLiNER2 得分 0.86–0.87，接近于 DeBERTa-v3 的 0.89–0.92，且与 GPT-4o 的相差在 10 分以内。虽然在所有任务上 GPT-4o 处于领先地位，但考虑到其显著更大的规模以及广泛的多样化文本语料库预训练，其出色的表现是意料之中的。总的来说，GLiNER2 在各种分类设置中提供了具有竞争力的准确性，并且持续缩小了任务特定基准和大型专有模型之间的差距。

对于 NER 评估，我们使用 CrossNER 基准 (Liu et al., 2020) ，它衡量在五个专业领域之间的零样本泛化能力：AI、文学、音乐、政治和科学。如表 2 所示，GLiNER2 在整体 F1 得分上

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 1: 在各种基准上的零样本文本分类性能。

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 2: 在 CrossNER 基准上的零样本 F1 分数。

# Labels	GPT4o	DeBERTa	GLiClass	GLiNER2
5	358	1714	137	130
10	382	3404	131	132
20	425	6758	140	163
50	463	16897	190	208
Speedup	1.00Œ	0.10Œ	2.75Œ	2.62Œ

Table 3: 文本分类中不同标签数量的 CPU 延迟（毫秒）比较。

与 GPT-4o 非常接近（0.590 对比 0.599），并在 AI（0.526 对比 0.547）和文学（0.564 对比 0.561）中取得了更高的分数。虽然 GLiNER2 在科学和音乐等类别上落后于 GLiNER-M，但它在政治方面保持了强劲的表现（0.679），表明在不同实体类型上具有鲁棒性。考虑到 GLiNER2 是一个支持多项任务的通用模型，此 NER 性能水平与专用实体识别系统相比仅有小幅下降，展示了我们统一架构的有效性。

3.3 效率

我们通过衡量文本分类任务中不同标签数量下的推理延迟来评估 GLiNER2 的计算效率。除使用 OpenAI API 的 GPT-4o 外，所有模型均在 CPU 上进行评估。表 3 显示了不同分类标签数量下的延迟测量值（以毫秒为单位）。GLiNER2 表现出强大的计算效率，在延迟方面表现与 GLiClass 相当，同时提供了显著优于基于 DeBERTa 的零样本分类的性能。关键优势在于与 DeBERTa 的比较，后者为每个标签单独执行一次前向传播，导致与标签数量呈线性增长的延迟（20 个标签时延迟增加 6.8 倍）。相比之下，GLiNER2 在单次前向传播中同时处理所有标签，无论标签数量多少都保持一致的性能。GLiNER2 和 GLiClass 均在标准 CPU 硬件上实现了约 2.6 倍的加速，相较于 GPT-4o，展示了在实际生产部署环境中，延迟和计算资源至关重要的情况下，紧凑、专用模型的实际优势。

4 伪影

我们提供了一个 Python 包，通过直观的 API 使 GLiNER2 易于访问。`gliner2` 库可以通过 pip 轻松安装，并提供与 Hugging Face 生态系统的无缝集成，用于模型的分发和加载。可以使用标准的 `.from_pretrained` 方法加载模型，权重托管在 Hugging Face Hub 上，以方便访问。

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

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

Figure 1: 模型加载。

命名实体识别 命名实体识别可以通过多种方法进行，以适应不同的用例和复杂性水平。最简单的方法只需要输入文本和目标实体类型的列表。此外，用户可以使用字典格式提供实体类型的自然语言描述，其中键表示实体类型，值包含描述性文本，以帮助模型更好地理解提取目标。该过程及各种使用模式在图 2 中进行了说明。

层次结构提取 通过定义一个如图 3 所示的结构来执行层次结构提取。该结构定义了一个包含多个子字段的父实体（称为结构），使用 GLiNER2 的字段规范语法。每个字段遵循模式 `field_name::type::description`

```python
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 用于命名实体识别的简单和增强方法

，其中 `type` 指定单个值的 `str` 或多个值的 `list` 。字段可以通过格式 `field_name::[option1|option2]::type` 采用选择约束，例如被限制为电子、软件或硬件的类别字段。

```python
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: 带字段约束和描述的层次结构提取

该框架支持在一个模式中定义多个结构以处理复杂的抽取场景。例如，图 4 展示了用户如何在一个查询中定义两个结构：产品和公司。这使得能够同时抽取产品细节（名称和价格）以及公司信息（名称和总部），并且所有这些都在单次前向传递中高效处理。

```python
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: 在单个模式中组合多种层次结构

文本分类 与命名实体识别（NER）类似，文本分类功能提供了既简化又高度可定制的界面，以满足各种应用需求。对于快速部署，用户只需提供输入文本和一个将任务名称（例如，“情感”）映射到分类标签列表的字典，如图 5 中的第一个例子所示。对于更复杂的应用，库支持广泛的自定义选项，包括标签描述和多标签分类功能。当启用多标签分类时，模型应用 sigmoid 激活以允许多个标签同时分配，而单标签任务则使用 softmax 归一化用于互斥预测。

如图 6 所示，该库支持在一次调用中处理多个分类任务。每个分类任务都可以通过标签描述和多标签设置等功能独立定制。

```python
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: 具有简单和高级配置选项的文本分类

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

Figure 6: 同时多任务分类。

任务组合 该库的一个关键特性是其能够在一个统一的框架内高效地组合多个提取任务。图 8 演示了如何构建一个全面的模式，将实体识别、文本分类和结构化提取无缝地结合在一个推理调用中。

4.1 交互式 Gradio 示例

我们提供了一个基于网络的演示界面，允许用户无需编写代码即可与 GLiNER2 交互。该演示支持对实体类型、分类标签、描述和其他参数进行实时实验。界面包含三个选项卡，对应 GLiNER2 的核心功能。图 7 展示了层次结构抽取选项卡，用户可以定义具有多个字段和数据类型的模式，以从文本中提取结构化信息。

5 相关工作

一些框架已经在不同领域和方法中解决了信息提取任务。

传统自然语言处理库： spaCy (Honnibal et al., 2020) 、Stanford CoreNLP (Manning et al., 2014) 、Stanza (Qi et al., 2020) 提供了用于命名实体识别、词性标注和依存句法分析的综合工具包。然而，这些框架需要为每个任务提供单独的模型，缺乏统一的架构，且通常不能推广到未见过的标签。

基于 LLM 的抽取： XNLP (Fei et al., 2024) 通过提示策略展示了使用大型语言模型进行多样化的信息提取任务的能力，而 NuExtract (NuMind, 2024) 则专注于针对 JSON 提取的微

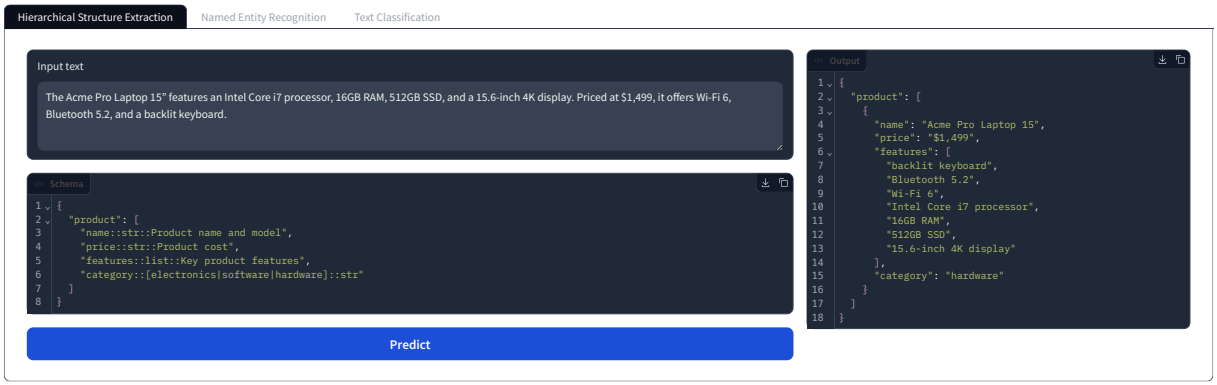

Figure 7: GLiNER2 Gradio 演示界面显示层次结构提取。

```python
# 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: 综合任务组合，结合所有提取类型

调。这些方法取得了很强的表现，但需要大量的计算资源和 GPU 推理。

基于编码器的方法： GLiNER (Zaratiana et al., 2024) 介绍了一种高效的范例，通过预训练的编码器在合成数据上进行微调，实现零样本命名实体识别，具备与之竞争的准确性并实现快速的 CPU 推理。这种方法启发了后续的工作，包括用于文本分类的 GLiClass (Knowledgator, 2025) 和用于零样本关系提取的 GLiREL (Boylan et al., 2025) 。GLiNER2 通过在单一高效框架中集成多个任务扩展了这项工作，能够在保持紧凑型编码器模型的计算优势的同时实现多任务组合。

6 结论

我们介绍了 GLiNER2，它在一个高效的 CPU 模型中统一了实体识别、文本分类和层次提取。与现有方法需要为每个任务单独模型不同，GLiNER2 通过声明性模式实现了多任务提取，同时保持参数数量在 500M 以下，以便于实际部署。我们将 GLiNER2 作为一个开源的 Python 库发布在 Apache 2.0 许可证下，并在 Hugging Face 上提供预训练权重。通过结合效率与多功能性，我们希望我们的库能够使先进的信息提取对研究和生产两者来说都是可及的。

References

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

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.

Alessio Cocchieri, Giacomo Frisoni, Marcos Martínez Galindo, Gianluca Moro, Giuseppe Tagliavini, and Francesco Candoli. 2025. Open-BioNER: 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.

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.

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`.

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.

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.

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.

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.

A 架构细节

我们的架构采用了一组学习的特殊标记，每个标记承担特定的语义角色：

- [P] （提示）：标记任务规范的开始，向模型发出信号将后续的标记解释为任务元数据
- [E] （实体）：在命名实体识别任务中位于每个实体类型之前，以为实体类别创建不同的嵌入
- [C] （子/组件）：表示分层结构中的属性字段并建立父子关系
- [L] （标签）：表示分类选项，每个标签接收一个唯一的嵌入用于评分
- [SEP] （分隔符）：划分不同的输入段，以防止任务规范与内容之间的信息泄漏。

这些标记是随机初始化的，并在训练过程中学习，使模型能够形成特定任务的表示。

命名实体识别 命名实体识别任务遵循以下输入格式：

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

在抽取过程中，每个 [E] 标记生成一个表示其实体类型的嵌入。模型为所有可能的文本跨度创建表示，跨度的最大宽度有限，然后使用以下方法计算跨度-实体对之间的匹配分数：

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

，其中 $\boldsymbol{h}_{s_i}$ 是跨度表示，$\boldsymbol{h}_{e_j}$ 是实体类型嵌入，$\text{sim}(\cdot,\cdot)$ 是带有 sigmoid 激活的点积。

例如，给定 [P] entities ([E] person [E] location) 和文本 “John works in Paris” ，所有的跨度候选（例如，“John”、“works”、“Paris”、“works in”）都会针对实体类型嵌入（即，每个 [E] 标记的表示）进行评分。对于任何实体类型，预测概率超过 0.5 的跨度被选择为实体。

层次结构提取 分层提取使用以下格式：

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

该过程分为两个阶段。首先，一个 MLP 处理 [P] 标记嵌入以预测在文本中父实体实例的数量 K 。这个 MLP 执行 20 类分类（针对 0-19 的计数），在训练过程中使用真实实例计数进行训练。然后，模型通过将 [C] 标记嵌入条件化到学习的出现 ID 嵌入上，为每个属性生成 K 个不同的表示。具体来说，对于每个实例 $k \in \{1, ..., K\}$ ，模型将训练过程中学习的出现特定嵌入与基础的 [C] 嵌入结合在一起，生成每个实例属性对的独特表示。这些 $K \times m$ 表示通过与 NER 相同的评分机制与文本跨度匹配，确保每个实例保持单独的属性值。考虑结构化提取任务：

[P] product (在研究中，我们考察了 XMATHX_ i 在不同条件下的行为。[C] 这个行为遵循一定的模式，从而使我们能够预测未来的趋势。附录 C 中包含了详细的计算方法和公式。我们的分析表明，该方法在各种实验中的表现相对一致，尽管某些参数有时会产生不确定性。 name [C] price)

给定输入文本：“iPhone costs $ 999. Galaxy is $ 899.”，该模型分三步处理：

1. 计数预测：MLP 计数预测器处理 [P] 标记嵌入并输出 $K = 2$ ，表明文本中存在两个产品实例。
2. 表示生成：计数嵌入层为每个属性字段生成 K 组条件表示。这为 [C] 名称 生成两个不同的嵌入，为 [C] 价格 生成两个不同的嵌入，每一对嵌入对应一个产品实例。
3. 跨度抽取：每个条件表示计算与所有可能的文本跨度的相似性分数，如同在命名实体识别中一样。该模型为每个字段选择分数最高的跨度，同时保持实例的一致性：
   - 实例 1: { 名称: "iPhone", 价格: " $ 999" }
   - 实例 2: { 名称: "Galaxy", 价格: "$ 899" }

这种并行处理能够在保留每个实例内字段之间的语义关系的同时，高效地提取多个结构化实体。

文本分类 分类任务使用格式：

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

每个 [L] 标记产生一个标签特定的嵌入，通过分类头进行精细化。具体来说，对于每个标签 ℓ_i ，模型计算：

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

，其中 $\boldsymbol{h}_{\ell_i}$ 是从 [L] 标记对标签 ℓ_i 的上下文嵌入，而 MLP 是一个多层感知器，将这些嵌入

投射到表示标签文本兼容性的标量 logits。单标签任务对所有 logits 应用 softmax 以选择概率最高的标签，而多标签场景对每个 logit 独立使用 sigmoid 激活。考虑文本分类任务：

`[P] sentiment ([L] positive [L] negative （由于缺少具体的英文文本部分，我无法进行翻译。请提供需要翻译的文本。） neutral)`

给定输入文本："这部电影太棒了！"。模型分三个步骤处理此文本：

1. 标签嵌入生成：每个 `[L]` 标记为其对应的标签（积极、消极、中立）创建一个不同的嵌入。

2. 分类头：标签嵌入通过多层感知器（MLP）进行投影以产生分类 logits，然后使用 softmax 激活函数进行归一化以进行单标签预测。

3. 标签选择：模型选择得分最高的标签，为给定的输入文本预测"积极"。

对于多标签场景，sigmoid 激活函数替代 softmax，允许同时选择多个标签。

任务组合 可以组合多个任务以进行高效的多任务推理，使用：

`[Task 1 ]` $\oplus$ `[SEP]` $\oplus$ `[Task 2 ]` $\oplus$ `...` $\oplus$ `[SEP]` $\oplus$ `[` $x_1, x_2, \ldots, x_N$ `]`

这使得多个提取任务可以在单次前向传播中同时执行。例如，结合命名实体识别和情感分类任务可以在对"Steve Jobs loved the iPhone"进行一次计算后，提取出实体 { person: ["Steve Jobs"], product: ["iPhone"] } 以及情感"positive"，比分别运行模型提高了效率。

我们将我们的方法与几个强大的基线进行比较。作为上限，我们在所有任务中使用 GPT-4o。对于分类任务，我们与两个具有相似参数数量的最新开源模型进行比较：（1）GLiClass（knowledgator/gliclass-base-v1.0），这是 GLiNER 的分类特定适应版本，（2）DeBERTa-v3-base-zeroshot（MoritzLaurer/deberta-v3-large-zeroshot-v2.0），这是 Hugging Face 上零样本分类的事实标准。对于 NER 任务，我们使用在 Zaratiana et al. (2024) 中报告的 GLiNER-M 性能，它代表了当前通用实体识别的最新水平。

我们使用 AdamW 优化器对模型进行 5 轮训练，并采用差分学习率：对任务特定层使用 2×10^{-5}，对编码器主体使用 1×10^{-5}。这种差分方法允许对预训练的表示进行微调，同时加速任务特定组件的适应。我们应用了 0.01 的权重衰减进行正则化，并在 1.0 处进行梯度裁剪以确保训练的稳定性。学习率计划包括 1,000 的线性缩放热身步。表格 4 总结了训练配置。

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 4: 所有实验中使用的训练超参数。

A.1 训练数据

我们的训练数据集包括 254,334 个由真实世界文本和合成数据组成的样本。表 5 显示了各领域的分布。

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 5: 跨领域的训练数据分布。

从新闻文章、维基百科、法律文件、ArXiv 论文和 PubMed 摘要中收集了真实世界数据（共 135,698 个示例）。所有文本均使用 GPT-4o 进行了实体识别、层次抽取和分类任务的注释。使用 GPT-4o 生成了覆盖实际使用案例的合成数据（118,636 个示例），这些案例包括电子邮件、短信、简历、社交媒体帖子、电子商务订单、银行记录和体育评论。每个示例都包含所有适用任务类型的注释。