GLiNER: Generalist Model for Named Entity Recognition using Bidirectional Transformer

Urchade Zaratiana[1,2], **Nadi Tomeh**[2], **Pierre Holat**[1,2], **Thierry Charnois**[2]
[1] FI Group, [2] LIPN, CNRS UMR 7030, France
zaratiana@lipn.fr
https://github.com/urchade/GLiNER

Abstract

Named Entity Recognition (NER) is essential in various Natural Language Processing (NLP) applications. Traditional NER models are effective but limited to a set of predefined entity types. In contrast, Large Language Models (LLMs) can extract arbitrary entities through natural language instructions, offering greater flexibility. However, their size and cost, particularly for those accessed via APIs like ChatGPT, make them impractical in resource-limited scenarios. In this paper, we introduce a compact NER model trained to identify any type of entity. Leveraging a bidirectional transformer encoder, our model, GLiNER, facilitates parallel entity extraction, an advantage over the slow sequential token generation of LLMs. Through comprehensive testing, GLiNER demonstrate strong performance, outperforming both ChatGPT and fine-tuned LLMs in zero-shot evaluations on various NER benchmarks.

1 Introduction

Named Entity Recognition plays a crucial role in various real-world applications, such as constructing knowledge graphs. Traditional NER models are limited to a predefined set of entity types. Expanding the number of entity types can be beneficial for many applications but may involve labeling additional datasets. The emergence of Large Language Models, like GPT-3 (Brown et al., 2020), has introduced a new era for open-type NER by enabling the identification of any types of entity types only by natural language instruction. This shift signifies a significant departure from the inflexibility observed in traditional models. However, powerful LLMs typically consist of billions of parameters and thus require substantial computing resources. Although it is possible to access some LLMs via APIs (OpenAI, 2023), using them at scale can incur high costs.

Recently, researchers have explored the fine-tuning of open source language models such as

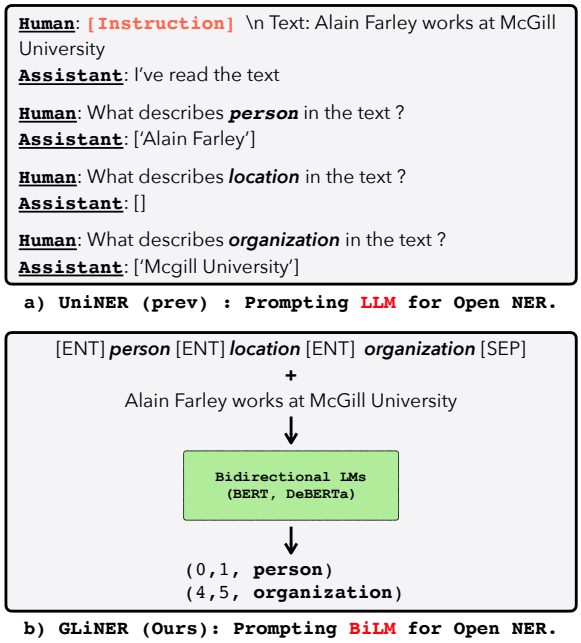

Figure 1: **BiLM for Open NER.** Previous models, such as UniNER (Zhou et al., 2023) (Fig. a), approach the task of open type NER by prompting LLMs with natural language instructions (using a multi-turn dialogue style). Our proposed GLiNER utilizes small bidirectional LMs and treats the NER task as matching entity types with textual spans in a latent space.

LLaMa (Touvron et al., 2023) for named entity recognition tasks. Wang et al. (2023), for example, introduced InstructUIE, a fine-tuned FlanT5-11B (Raffel et al., 2019; Chung et al., 2022) model on existing information extraction datasets, achieving excellent performance in zero-shot settings. Additionally, GoLLIE (Sainz et al., 2023) was introduced as an extension of InstructUIE work by by finetuning a CodeLLaMa (Rozière et al., 2023) using detailed annotation guidelines, resulting in significant performance improvements. Another recent proposal by Zhou et al. (2023), called UniversalNER, involves the fine-tuning of LLMs using diverse data from various domains annotated with ChatGPT instead of relying on standard NER

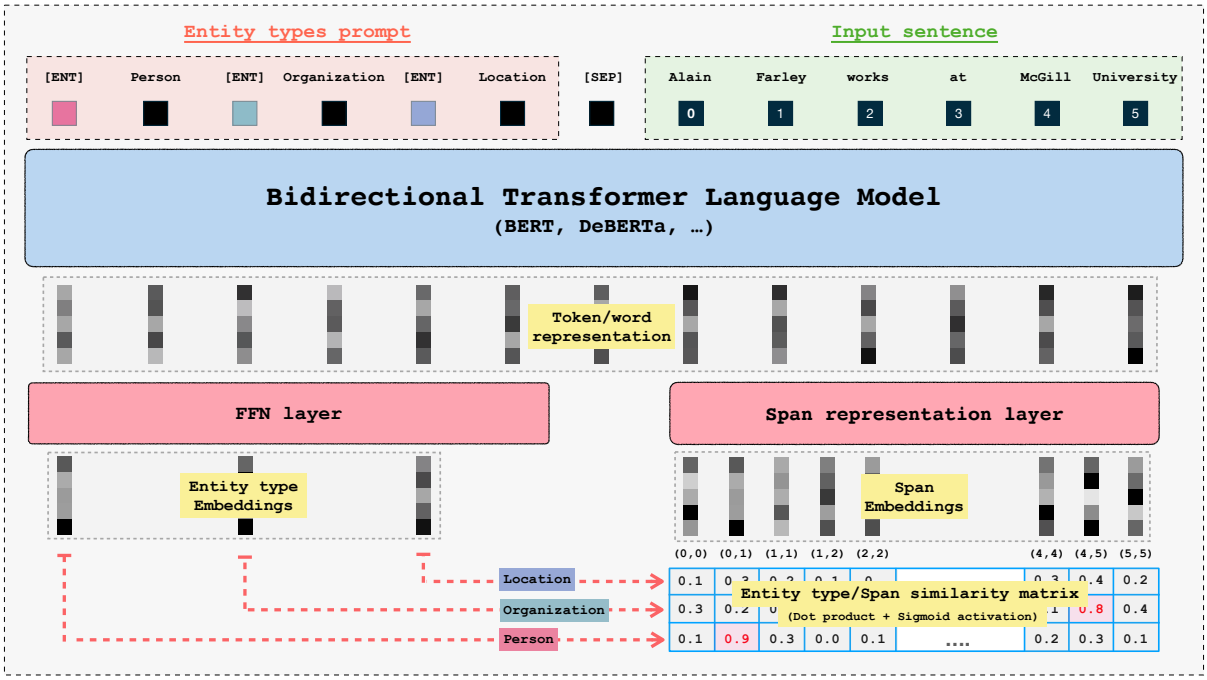

Figure 2: **Model architecture**. GLiNER employs a BiLM and takes as input entity type prompts and a sentence/text. Each entity is separated by a learned token [ENT]. The BiLM outputs representations for each token. Entity embeddings are passed into a FeedForward Network, while input word representations are passed into a span representation layer to compute embeddings for each span. Finally, we compute a matching score between entity representations and span representations (using dot product and sigmoid activation). For instance, in the figure, the span representation of (0, 1), corresponding to "Alain Farley," has a high matching score with the entity embeddings of "Person".

datasets. Their approach enables the replication and even surpassing of the original capability of ChatGPT when evaluated in zero-shot on several NER datasets. While these works have achieved remarkable results, they present certain limitations we seek to address: They use autoregressive language models, which can be slow due to token-by-token generation; Moreover, they employ large models with several billion parameters, limiting their deployment in compute-limited scenarios. Furthurmore, as NER is treated as a text generation problem, the generation of entities is done in several decoding steps, and there is no way to perform the prediction of multiple entity types in parallel.

In our work, we propose a model that addresses the above-mentioned problems. Instead of relying on large autoregressive models, we utilize smaller-scale Bidirectional Language Models (BiLM), such as BERT (Devlin et al., 2019) or deBERTa (He et al., 2021). The core concept of our model involves treating the task of Open NER as matching entity type embeddings to textual span representations in latent space, rather than as a generation task. This approach naturally solves the scalability issues of autoregressive models and allows for bidirectional context processing, which enables richer representations. When trained on the dataset released by (Zhou et al., 2023), which comprises texts from numerous domains and thousands of entity types, our model demonstrates impressive zero-shot performance. More specifilcally, it outperforms both ChatGPT and fine-tuned LLMs on zero-shot NER datasets (Table 1). Our model's robustness is further evident in its ability to handle languages not even encountered during training. Specifically, our model surpasses ChatGPT in 8 of 10 languages (Table 3) that were not included in its training data.

2 Method

This section presents our model, GLiNER, which is train to extract any types of entity using a Bidirectional Language Models. Our model has three main components: i) a pre-trained textual encoder (a BiLM such as BERT), ii) a span representation module which computes span embeddings from token embeddings, iii) an entity representation mod-

ule which computes entity embeddings that the model seeks to extract. The goal is to have entity and span embeddings in the same latent space to assess their compatibility (degree of matching). The overall architecture of our model is depicted in Figure 2.

2.1 Architecture

Input format The input to our model comprises a unified sequence combining entity types (expressed in natural language) and the input text from which entities are to be extracted. The input format is as follows:

[ENT] t_0 [ENT] t_1 ... [ENT] t_{M-1} [SEP] x_0 x_2 ... x_{N-1}

[ENT] token represents a special token placed before each entity type and the [SEP] token functions as a delimiter, separating the sequence of entity types from the input text. They are initialized randomly at the start of training.

Token representation The token encoder processes the unified input to compute interactions between all tokens (both entity types and input text), producing contextualized representations. Let $\boldsymbol{p} = \{\boldsymbol{p}_i\}_0^{M-1} \in \mathbb{R}^{M \times D}$ represent the encoder's output for each entity type, corresponding to all the [ENT] token representations. Similarly, $\boldsymbol{h} = \{\boldsymbol{h}_i\}_0^{N-1} \in \mathbb{R}^{N \times D}$ denotes the representation of each word in the input text. For words tokenized into multiple subwords, we use the representation of the first subword, which is a standard choice in the NER literature (Zaratiana et al., 2022).

Entity and Span Representation In our model, we aim to encode entity types and span embeddings into a unified latent space. The entity representation is computed by refining the initial representation $\boldsymbol{p}$ using a two-layer feedforward network, resulting in $\boldsymbol{q} = \{\boldsymbol{q}_i\}_0^{M-1} \in \mathbb{R}^{M \times D}$. The representation of a span starting at position i and ending at position j in the input text, $\boldsymbol{S}_{ij} \in \mathbb{R}^D$, is computed as:

$$\boldsymbol{S}_{ij} = \text{FFN}(\boldsymbol{h}_i \otimes \boldsymbol{h}_j) \tag{1}$$

Here, FFN denotes a two-layer feedforward network, and $\otimes$ represents the concatenation operation. In practice, The computation of all span representations can be easily parallelized. Moreover, we set an upper bound to the length (K=12) of the span in order to keep linear complexity, without harming recall.

Entity Type and Span Matching To evaluate whether a span (i, j) corresponds to entity type t, we calculate the following matching score:

$$\phi(i, j, t) = \sigma(\boldsymbol{S}_{ij}^T \boldsymbol{q}_t) \in \mathbb{R} \tag{2}$$

In this equation, σ denotes a sigmoid activation function. As we train with binary cross-entropy loss (see next sec. 2.2), $\phi(i, j, t)$ can be interpreted as the probability of the span (i, j) being of type t.

2.2 Training

During training, our objective is to optimize model parameters to enhance the matching score for correct span-type pairs (positive pairs) and reduce it for incorrect pairs (negative pairs). A span (i, j) paired with an entity type t forms a positive pair ($s \in \mathcal{P}$) if the span is labeled with type t in the training data. Otherwise, it is a negative pair ($s \in \mathcal{N}$). The training loss for an individual example, comprising spans $\mathcal{S}$ and entity types $\mathcal{T}$, is defined as:

$$\begin{aligned}\mathcal{L}_{\text{BCE}} = -\sum_{s \in \mathcal{S} \times \mathcal{T}} & \mathbb{I}_{s \in \mathcal{P}} \log \phi(s) + \\ & \mathbb{I}_{s \in \mathcal{N}} \log (1 - \phi(s))\end{aligned} \tag{3}$$

The variable s represents a pair of span/entity type and $\mathbb{I}$ is an indicator function, which returns 1 when the specified condition is true and 0 otherwise. This loss function corresponds to binary cross-entropy.

2.3 Decoding algorithm

In the decoding phase, we employ a greedy span section (Zaratiana et al., 2022) that selects entity spans based on matching scores, to ensure task/dataset specific constraints. This strategy is applied independently to each sentence. Only, spans (i, j) with matching scores $\phi(i, j, c) > 0.5$ are considered for selection.

Flat NER: The algorithm chooses the highest-scoring non-overlapping span and continues this process until all spans are evaluated.

Nested NER: Similar to Flat NER, but the algorithm allows selection of fully nested spans within other entities while still avoiding partial overlaps.

Algorithm Efficiency: The decoding is implemented using a priority queue for spans, ensuring an $O(n \log n)$ complexity, with n being the number of candidate spans.

Model	Params	Movie	Restaurant	AI	Literature	Music	Politics	Science	Average
Vicuna-7B	7B	6.0	5.3	12.8	16.1	17.0	20.5	13.0	13.0
Vicuna-13B	13B	0.9	0.4	22.7	22.7	26.6	27.0	22.0	17.5
USM	0.3B	37.7	17.7	28.2	56.0	44.9	36.1	44.0	37.8
ChatGPT	–	5.3	32.8	52.4	39.8	66.6	68.5	**67.0**	47.5
InstructUIE	11B	**63.0**	21.0	49.0	47.2	53.2	48.1	49.2	47.2
UniNER-7B	7B	42.4	31.7	53.6	59.3	67.0	60.9	61.1	53.7
UniNER-13B	13B	48.7	36.2	54.2	60.9	64.5	61.4	63.5	55.6
GoLLIE	7B	**63.0**	**43.4**	**59.1**	62.7	67.8	57.2	55.5	58.0
GLiNER-S	50M	46.9	33.3	50.7	60.0	60.9	61.5	55.6	52.7
GLiNER-M	90M	42.9	37.3	51.8	59.7	69.4	68.6	58.1	55.4
GLiNER-L	0.3B	57.2	42.9	57.2	**64.4**	**69.6**	**72.6**	62.6	**60.9**

Table 1: **Zero-Shot Scores on Out-of-Domain NER Benchmark.** We report the performance of GLiNER with various DeBERTa-v3 (He et al., 2021) model sizes. Results for Vicuna, ChatGPT, and UniNER are from Zhou et al. (2023); USM and InstructUIE are from Wang et al. (2023); and GoLLIE is from Sainz et al. (2023).

3 Experimental Setting

3.1 Training data

Our objective is to construct a versatile NER model capable of accurately identifying a wide array of entity types across different textual domains. To achieve this, it is essential that our training dataset encompasses a diverse range of entity types. For this, we utilize the training data released by Zhou et al. (2023), known as Pile-NER[1]. This dataset is derived from the Pile corpus (Gao et al., 2020), commonly used for pretraining large language models, and comprises text from diverse sources. More specifically, Zhou et al. (2023) sampled 50,000 texts from the Pile data and employed ChatGPT to extract their associated entity types. Notably, they did not specify the entity types to the LLMs, aiming to extract a diverse range of entity types. They used the following prompt:

```
System Message: You are a helpful information extraction system.

Prompt: Given a passage, your task is to extract all entities and identify their entity types. The output should be in a list of tuples of the following format: [("entity 1", "type of entity 1"), ... ].

Passage: {input_passage}
```

Figure 3: **Prompting ChatGPT for entity extraction**. This prompt was used Zhou et al. (2023) to construct the Pile-NER dataset.

Finally, after filtering bad outputs their datasets results in 44889 passages containing in total 240k entity spans and 13k distinct entity types.

[1]https://huggingface.co/datasets/Universal-NER/Pile-NER-type

3.2 Hyperparameters

Our model, GLiNER, is trained on the Pile-NER dataset, which we described in the previous section. We use the deBERTa-v3 (He et al., 2021) as our backbone due to its proven empirical performance. All non-pretrained layers have a width dimension of 768 and a dropout rate of 0.4. Regarding the training process, we employ the AdamW optimizer (Loshchilov and Hutter, 2017), setting a base learning rate of 1e-5 for pretrained layers (the transformer backbone) and 5e-5 for non-pretrained layers (FFN layers, span representation). We trained our models for a maximum of 30k steps, starting with a 10% warmup phase, followed by a decay phase using a cosine scheduler. The Pile-NER dataset natively contains only positive entities (i.e., entities that are present in the sentence), and we found it useful to include negative entity types during training. This is achieved by sampling random entities from other examples in the same batch. In addition, we follow the strategies outlined in Sainz et al. (2023) as a form of regularization, which includes *shuffling entity order* and *randomly dropping entities*. Furthermore, we limit the number of entity types to 25 per sentence during training. The larger variant of our model, GLiNER-L, takes 5 hours to train on an A100 GPU.

3.3 Baselines

In our evaluation, we compare our model, GLiNER, with several recent models designed for open-type NER. First, we examine chat models like **ChatGPT** and **Vicuna** (Chiang et al., 2023), which utilize the prompting from Ye et al. (2023); we

report their results reported by Zhou et al. (2023). Next, we focus on three recent Large Language Models (LLMs) that have been fine-tuned for NER: **InstructUIE** (Wang et al., 2023), based on the FlanT5 11B model and fine-tuned on various NER datasets; **UniNER** (Zhou et al., 2023), which employs a LLaMa model fine-tuned on a dataset generated by ChatGPT; **GoLLIE** (Sainz et al., 2023), fine-tuned to adhere to detailed annotation guidelines for enhanced performance in unseen IE tasks, utilizing CodeLLama as its base model. Finally, we include **USM** (Lou et al., 2023) in our comparison, which is similar in size to ours but features a different architecture.

3.4 Evaluation

Datasets We primarily evaluate our model in a zero-shot context on common NER benchmarks, following previous works (Wang et al., 2023; Zhou et al., 2023). The first is the *OOD NER Benchmark* (Table 1), which comprises seven diverse NER datasets from CrossNER and MIT. This benchmark is typically used for evaluating out-of-domain generalization capabilities of NER models. The second benchmark consists of *20 NER datasets* (Table 2) from a wide range of domains, including biomedical, news articles, and tweets. These datasets are commonly used for training supervised NER models. Additionally, we evaluate our model on multilingual NER datasets (Table 3) for further investigation. For this purpose, we use the recently released *Multiconer* (Multilingual Complex NER) (Malmasi et al., 2022), which contains data in 11 languages across various domains.

Metric We adopt the standard NER evaluation methodology, calculating F1-score based on the exact match between predicted and actual entities.

4 Results

4.1 Zero-shot on English datasets

In this section, we discuss the performance of our model in a zero-shot context, i.e., by only training on the Pile-NER dataset without further fine-tuning on target datasets.

OOD NER Benchmark We first evaluate our model on the OOD benchmark as reported in Table 1. We compare three different sizes of our model (small, medium, and large) against the baselines. The results demonstrate our model's impressive capability, irrespective of its size. For example, even our smallest model, with only 50M parameters, outperforms general-purpose models such as ChatGPT and Vicuna. It also shows better performance than the 11B InstructUIE, which has been instruction-tuned for the NER task. Furthermore, when compared to UniNER, which used the same training data as GLiNER, our medium-sized model (90M) achieves comparable results to UniNER-13B (55 F1 for both), despite being 140 times smaller, while our largest version outperforms it by an average of 5 points. Our most best competitor, GoLLIE, which leads among the LLMs, achieves better performance than most of our models but is still less effective than GLiNER-L. Against USM, which has a comparable number of parameters to ours, the performance is significantly lower, highlighting the superiority of our architecture.

Dataset	ChatGPT	UniNER-7B	GLiNER-L
ACE05	26.6	**36.9**	27.3
AnatEM	30.7	25.1	**33.3**
bc2gm	40.2	46.2	**47.9**
bc4chemd	35.5	**47.9**	43.1
bc5cdr	52.4	**68.0**	66.4
Broad Tweeter	61.8	**67.9**	61.2
CoNLL03	52.5	**72.2**	64.6
FabNER	15.3	**24.8**	23.6
FindVehicle	10.5	22.2	**41.9**
GENIA	41.6	54.1	**55.5**
HarveyNER	11.6	18.2	**22.7**
MIT Movie	5.3	42.4	**57.2**
MIT Restaurant	32.8	31.7	**42.9**
MultiNERD	58.1	59.3	**59.7**
ncbi	42.1	60.4	**61.9**
OntoNotes	29.7	27.8	**32.2**
PolyglotNER	33.6	41.8	**42.9**
TweetNER7	40.1	**42.7**	41.4
WikiANN	52.0	55.4	**58.9**
WikiNeural	57.7	69.2	**71.8**
Average	36.5	45.7	**47.8**

Table 2: **Zero-shot performance on 20 NER datasets.** Results of ChatGPT and UniNER are reported from (Zhou et al., 2023).

20 NER Benchmark Table 2 presents a comparison of our model against ChatGPT and UniNER across 20 diverse NER datasets. First, similar to the OOD benchmark, ChatGPT significantly lags behind fine-tuned models for NER, trailing behind UniNER. Furthermore, GLiNER achieves the highest performance on 13 of these datasets, surpassing UniNER by an average of 2 points. This superior performance underscores GLiNER's robustness and adaptability across a broad spectrum of domains. However, a notable observation is that GLiNER underperforms compared to UniNER on

	Language	Sup.	ChatGPT	GLiNER	
				En	Multi
Latin	German	64.6	37.1	35.6	**39.5**
	English	62.7	37.2	**42.4**	41.7
	Spanish	58.7	34.7	38.7	**42.1**
	Dutch	62.6	35.7	35.6	**38.9**
Non-Latin	Bengali	39.7	23.3	0.89	**25.9**
	Persian	52.3	25.9	14.9	**30.2**
	Hindi	47.8	27.3	11.3	**27.8**
	Korean	55.8	**30.0**	20.5	28.7
	Russian	59.7	27.4	30.3	**33.3**
	Turkish	46.8	**31.9**	22.0	30.0
	Chinese	53.1	18.8	6.59	**24.3**
Average		54.9	29.9	23.6	**32.9**

Table 3: **Zero-Shot Scores on Different Languages.** The baseline, **Sup.**, is an XLM-R (Conneau et al., 2019) model fine-tuned on the training set of each language separately, as reported by Malmasi et al. (2022). ChatGPT evaluation is taken from Lai et al. (2023). GLiNER-En employs deBERTa-v3-Large, and Multi uses mdeBERTa-v3-base.

Dataset	InstructUIE	UniNER-7B	GLiNER-L	
	w/o	w/	w/	w/o
ACE05	79.9	**86.7**	82.8	81.3
AnatEM	88.5	88.5	**88.9**	88.4
bc2gm	80.7	82.4	**83.7**	82.0
bc4chemd	87.6	**89.2**	87.9	86.7
bc5cdr	89.0	**89.3**	88.7	88.7
Broad Twitter	80.3	81.2	82.5	**82.7**
CoNLL03	91.5	**93.3**	92.6	92.5
FabNER	78.4	**81.9**	77.8	74.8
FindVehicle	87.6	**98.3**	95.7	95.2
GENIA	75.7	77.5	**78.9**	77.4
HarveyNER	**74.7**	74.2	68.6	67.4
MIT Movie	89.6	**90.2**	87.9	87.5
MIT Restaurant	82.6	82.3	**83.6**	83.3
MultiNERD	90.3	93.7	**93.8**	93.3
ncbi	86.2	87.0	**87.8**	87.1
OntoNotes	88.6	**89.9**	89.0	88.1
PolyglotNER	53.3	**65.7**	61.5	60.6
TweetNER7	**65.9**	65.8	51.4	50.3
WikiANN	64.5	**84.9**	83.7	82.8
wikiNeural	88.3	**93.3**	91.3	91.4
Average	81.2	**84.8**	82.9	82.1

Table 4: **In-domain Supervised Finetuning.** All the models are fine-tuned on the mix of all training data of the benchmark. **w/** indicates that the model was trained on the Pile-NER dataset before finetuning.

tweet-based NER datasets. This highlights potential areas for improvement in GLiNER's ability to process informal, colloquial, or noisy data, typical of social media content.

4.2 Zero-Shot Multilingual Evaluation

In this section, we evaluate the performance of our model in a zero-shot context on unseen languages to assess its generalizability. This evaluation uses the Multiconel dataset, with results detailed in Table 3. Our model, GLiNER, is presented in two variants: **En**, which employs deBERTa-v3-Large as its backbone, and **Multi**, which utilizes a multilingual version of deBERTa-v3 (mdeBERTa-v3). Both versions were fine-tuned on the Pile-NER dataset. For comparative purposes, we include results from ChatGPT and a supervised baseline, the latter being fine-tuned on the training set of each dataset using separate models.

Results As expected, the supervised baseline demonstrated superior performance, significantly outperforming the zero-shot models. Among these, GLiNER-Multi showed the most promising results, surpassing ChatGPT in most languages, which is noteworthy considering that the fine-tuning dataset, Pile-NER, consists exclusively of English examples. Interestingly, its performance on Spanish data slightly exceeded that in English. While GLiNER-En generally underperformed compared to ChatGPT on average, it achieved competitive, and at times superior, results in languages using the Latin script, such as Spanish and German. However, its performance was markedly less competitive in non-Latin languages, particularly in Bengali, where it scored only 0.89 in the F1 score.

4.3 In-domain Supervised tuning

In our work, we also perform in-domain supervised fine-tuning (on 20 NER datasets) of our model to compare its capabilities against LLMs under this setup. Specifically, we compare our model against InstructUIE and UniNER, both of which have also been fine-tuned. The main difference is that UniNER has been pre-trained on the Pile-NER dataset before fine-tuning.

Training Setup For the supervised setting, we primarily adhere to the same experimental setup as described in the main experiment (using deBERTa-v3 large), except for the training dataset. Regarding the training data, we follow the approach of InstructUIE: we randomly sample 10,000 data points for each dataset in the 20 NER benchmark. If a dataset does not contain 10,000 samples, we include all available data. We implement two variants of our model: the first one initializes the weights from our zero-shot model, which is a pretrained on the Pile-NER dataset. The second variant is trained without the Pile-NER dataset, same as InstructUIE.

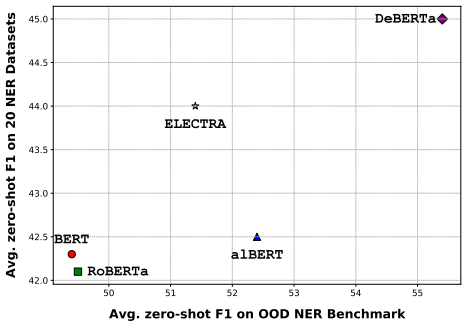

Figure 4: **Zero-shot performance for different backbones.** It reports the avg. results on 20 NER and OOD NER datasets

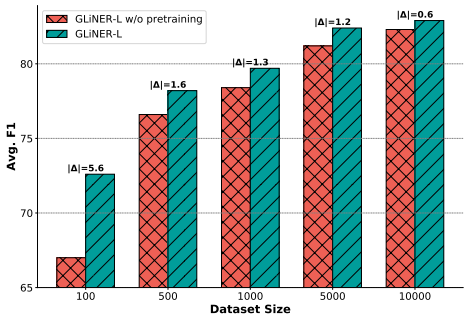

Figure 5: **Supervised performance across different dataset sizes.** The evaluation is conducted on the 20 NER datasets (in table 4).

Results The results of our experiment are reported in Table 3. Firstly, we observe that for the in-domain fine-tuning, our GLiNER model, pretrained on Pile-NER, achieves slightly better results than the non-pretrained variant, with an average difference of 0.8. Moreover, our pretrained GLiNER model outperforms InstructUIE (with an average difference of 0.9) despite being fine-tuned on the same dataset, whereas InstructUIE is significantly larger (approximately 30 times so). This demonstrates that our proposed architecture is indeed competitive. However, our model falls behind UniNER by almost 3 points. Nevertheless, our model still manages to achieve the best score in 7 out of 20 datasets.

5 Further analysis and ablations

Here we conduct different set of experiments to better investigate our model.

5.1 Effect of Different Backbones

In our work, we primarily utilize the deBERTa-v3 model as our backbone due to its strong empirical performance. However, we demonstrate here that our method is adaptable to a wide range of BiLMs.

Setup Specifically, we investigate the performance of our model using other popular BiLMs, including BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), AlBERT (Lan et al., 2019), and ELECTRA (Clark et al., 2020). We also conducted experiments with XLNet (Yang et al., 2019) but did not achieve acceptable performance (achieving at most 3 F1 on the OOD benchmark) despite extensive hyperparameter tuning. For a fair comparison, we employed the base size (GLiNER-M) and tuned the learning rate for each model. We report the zero-shot results on both the OOD benchmark and the 20 NER benchmark in Figure 4.

Result The results of our experiment, as shown in the Figure 4, clearly demonstrate the superiority of deBERTa-v3 over other pretrained BiLMs. It achieves the highest performance on both benchmarks by a clear margin. ELECTRA and AlBERT also show notable performance, albeit slightly lower, while BERT and RoBERTa lag behind with similar scores. However, it should be noted that all of the backbones we tested demonstrate strong performance compared to existing models. More specifically, even BERT-base, which ranks among the lower performers, achieves around 49 F1 on the OOD benchmark. This score is still 2 F1 points higher than the average for models like ChatGPT and InstructUIE.

5.2 Effect of Pretraining on In-domain Performance

In this section, we investigate the impact of pretraining on the Pile-NER dataset for supervised in-domain training on the 20 NER datasets, across various data sizes. The experiments range from 100 samples per dataset to 10,000 (full training setup). We use the same hyperparameters for all configurations. The results are reported in Figure 5.

Results As shown in the figure, models pretrained on Pile-NER consistently outperform their counterparts that are only trained on supervised data, indicating successful positive transfer. We further observe that the gain is larger when supervised data is limited. For instance, the difference in performance is 5.6 when employing 100 samples per dataset, and the gap becomes smaller as the size of the dataset increases.

Negative Samples	Prec	Rec	F1
0%	49.3	58.1	53.3
50%	**62.3**	**59.7**	**60.9**
75%	61.1	56.5	58.6

Table 5: **Effect of negative entity types sampling.**

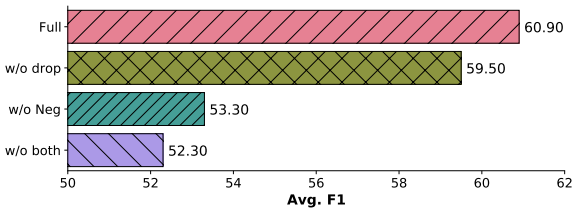

Figure 6: **Randomly dropping entity types.** We report the results with and without negative entity sampling.

5.3 Ablations

Negative Entity Sampling The original Pile-NER dataset, curated by Zhou et al. (2023), features passages with positive entity instances, i.e., entities that are directly present in the text. To better align training with real-world scenarios, where some entity types might be absent, we implemented negative entity sampling as mentioned in Section 3.2. In this study, we evaluate different sampling ratios: 0% (only positive entities), 50%, and 75% negative entities. Table 5 shows that training with only positive entities results in lower precision but higher recall, indicating that the model often makes false positive errors. Conversely, using 75% negative entities increases precision but decreases recall, as the abundance of negatives makes the model more cautious, leading to missed correct entities. A 50% negative entity ratio proves to be the most effective, providing a balanced approach.

Entity type dropping In the experiment, we employed a strategy of randomly varying the number of entity prompts during training. This approach aimed to expose the model to different quantities of entity types in each training instance, thereby increasing its adaptability to handle scenarios with varying numbers of entities. The usage of this technique results in an average improvement of over 1.4 points in out-of-domain evaluation, as shown in the Figure 5.

6 Related Works

Named Entity Recognition NER is a well-established task in the field of NLP, with numerous applications. Initially, NER models relied on rule-based system (Weischedel et al., 1996) that were built using handcrafted algorithms and gazetteers (Mikheev et al., 1999; Nadeau et al., 2006; Zamin and Oxley, 2011). However, these models had limitations in terms of scalability and adaptability to new domains or languages. To overcome these issues, machine learning approaches have been proposed (Lafferty et al., 2001). In the early stages, NER tasks were designed as sequence labeling (Huang et al., 2015; Lample et al., 2016; Akbik et al., 2018) where the objective was to predict tagged sequences (e.g., BILOU tags (Ratinov and Roth, 2009)). Since then, several paradigm shifts have occurred: span-based approaches treating NER as span classification (Sarawagi and Cohen, 2004; Fu et al., 2021; Li et al., 2021; Zaratiana et al., 2023); NER being treated as a question answering problem (Li et al., 2019); and even as a generation task (Yan et al., 2021).

Zero-shot learning for NER The advent of large-scale autoregressive models has recently transformed many paradigms in NLP through natural language prompting (Min et al., 2022; Wei et al., 2022; Qin et al., 2023). This is also the case for NER (Li et al., 2022; Ashok and Lipton, 2023; Agrawal et al., 2022). Others have fine-tuned these models for tasks to better align their capabilities with the requirements of entity recognition (Cui et al., 2021; Zhou et al., 2023) or information extraction in general (Lou et al., 2023; Wang et al., 2023; Sainz et al., 2023; Lu et al., 2022). This is done through-instruction tuning (Mishra et al., 2021; Wang et al., 2022; Longpre et al., 2023).

7 Conclusion

In this paper, we introduced GLiNER, a new method for identifying various types of entities in text using bidirectional language models. Our model not only outperforms state-of-the-art Large Language Models like ChatGPT in zero-shot scenarios but also offers a more resource-efficient alternative, crucial for environments with limited computing power. GLiNER is versatile, performing well in multiple languages, including those it wasn't trained on. In future work, we aim to further improve GLiNER's design for enhanced performance and to better adapt it for low-resource languages.

References

Monica Agrawal, Stefan Hegselmann, Hunter Lang, Yoon Kim, and David A. Sontag. 2022. Large language models are few-shot clinical information extractors. In *Conference on Empirical Methods in Natural Language Processing*.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Dhananjay Ashok and Zachary Chase Lipton. 2023. Promptner: Prompting for named entity recognition. *ArXiv*, abs/2305.15444.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. In *Annual Meeting of the Association for Computational Linguistics*.

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using bart. In *Findings*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*.

Jinlan Fu, Xuanjing Huang, and Pengfei Liu. 2021. SpanNER: Named entity re-/recognition as span prediction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7183–7195, Online. Association for Computational Linguistics.

Leo Gao, Stella Rose Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The pile: An 800gb dataset of diverse text for language modeling. *ArXiv*, abs/2101.00027.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. *ArXiv*, abs/2111.09543.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging.

John D. Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*.

Viet Dac Lai, Nghia Trung Ngo, Amir Pouran Ben Veyseh, Hieu Man, Franck Dernoncourt, Trung Bui, and Thien Huu Nguyen. 2023. Chatgpt beyond english: Towards a comprehensive evaluation of large language models in multilingual learning. *ArXiv*, abs/2304.05613.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *North American Chapter of the Association for Computational Linguistics*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *ArXiv*, abs/1909.11942.

Dongfang Li, Baotian Hu, and Qingcai Chen. 2022. Prompt-based text entailment for low-resource named entity recognition. *ArXiv*, abs/2211.03039.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2019. A unified mrc framework for named entity recognition. *ArXiv*, abs/1910.11476.

Yangming Li, lemao liu, and Shuming Shi. 2021. Empirical analysis of unlabeled entity problem in named entity recognition. In *International Conference on Learning Representations*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

S. Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Jie Lou, Yaojie Lu, Dai Dai, Wei Jia, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. 2023. Universal information extraction as unified semantic matching. In *AAAI Conference on Artificial Intelligence*.

Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. 2022. Unified structure generation for universal information extraction. In *Annual Meeting of the Association for Computational Linguistics*.

Shervin Malmasi, Anjie Fang, Besnik Fetahu, Sudipta Kar, and Oleg Rokhlenko. 2022. Multiconer: A large-scale multilingual dataset for complex named entity recognition. In *International Conference on Computational Linguistics*.

Andrei Mikheev, Marc Moens, and Claire Grover. 1999. Named entity recognition without gazetteers. In *Conference of the European Chapter of the Association for Computational Linguistics*.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? *ArXiv*, abs/2202.12837.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2021. Cross-task generalization via natural language crowdsourcing instructions. In *Annual Meeting of the Association for Computational Linguistics*.

David Nadeau, Peter D. Turney, and Stan Matwin. 2006. Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. In *Canadian Conference on AI*.

OpenAI. 2023. Gpt-4 technical report.

Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. 2023. Is chatgpt a general-purpose natural language processing task solver? *ArXiv*, abs/2302.06476.

Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Lev-Arie Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Conference on Computational Natural Language Learning*.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, I. Evtimov, Joanna Bitton, Manish P Bhatt, Cristian Cantón Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre D'efossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2023. Code llama: Open foundation models for code. *ArXiv*, abs/2308.12950.

Oscar Sainz, Iker García-Ferrero, Rodrigo Agerri, Oier Lopez de Lacalle, German Rigau, and Eneko Agirre. 2023. Gollie: Annotation guidelines improve zero-shot information-extraction. *ArXiv*, abs/2310.03668.

Sunita Sarawagi and William W. Cohen. 2004. Semi-markov conditional random fields for information extraction. In *Neural Information Processing Systems*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971.

Xiao Wang, Wei Zhou, Can Zu, Han Xia, Tianze Chen, Yuan Zhang, Rui Zheng, Junjie Ye, Qi Zhang, Tao Gui, Jihua Kang, J. Yang, Siyuan Li, and Chunsai Du. 2023. Instructuie: Multi-task instruction tuning for unified information extraction. *ArXiv*, abs/2304.08085.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Maitreya Patel, Kuntal Kumar Pal, M. Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Shailaja Keyur Sampat, Savan Doshi, Siddharth Deepak Mishra, Sujan Reddy, Sumanta Patro, Tanay Dixit, Xudong Shen, Chitta Baral, Yejin Choi, Noah A. Smith, Hannaneh Hajishirzi, and Daniel Khashabi. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *Conference on Empirical Methods in Natural Language Processing*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Huai hsin Chi, F. Xia, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903.

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.

Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. 2021. A unified generative framework for various ner subtasks.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Neural Information Processing Systems*.

Junjie Ye, Xuanting Chen, Nuo Xu, Can Zu, Zekai Shao, Shichun Liu, Yuhan Cui, Zeyang Zhou, Chao Gong, Yang Shen, Jie Zhou, Siming Chen, Tao Gui, Qi Zhang, and Xuanjing Huang. 2023. A comprehensive capability analysis of gpt-3 and gpt-3.5 series models. *ArXiv*, abs/2303.10420.

Norshuhani Zamin and Alan Oxley. 2011. Building a corpus-derived gazetteer for named entity recognition. In *International Conference on Software Engineering and Computer Systems*.

Urchade Zaratiana, Nadi Tomeh, Niama Elkhbir, Pierre Holat, and Thierry Charnois. 2023. Filtered semi-markov CRF.

Urchade Zaratiana, Nadi Tomeh, Pierre Holat, and Thierry Charnois. 2022. Named entity recognition as structured span prediction. In *Proceedings of the Workshop on Unimodal and Multimodal Induction of Linguistic Structures (UM-IoS)*, pages 1–10, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Wenxuan Zhou, Sheng Zhang, Yu Gu, Muhao Chen, and Hoifung Poon. 2023. Universalner: Targeted distillation from large language models for open named entity recognition.