What Matters When Building Universal Multilingual Named Entity Recognition Models?

Anonymous ACL submission

Abstract

Recent progress in universal multilingual named entity recognition (NER) has been driven by multilingual transformer models, task-specific architectures, custom loss functions, and large-scale training datasets. However, despite substantial prior work, we find that many critical design decisions for such models are made without systematic justification, with individual components evaluated only in combination rather than in isolation. We argue that this lack of rigor impedes progress in the field by making it difficult to identify which choices improve multilingual generalization. In this work, we conduct extensive experiments on transformer backbones, architectures, training objectives, data composition, and threshold selection. Building on these findings, we present OTTER, a universal multilingual NER model supporting over 100 languages. OTTER achieves consistent improvements over strong multilingual NER baselines, outperforming similarly sized models by 5.3 percentage points in F1 and achieving competitive performance compared to 90× larger generative models, while being substantially more efficient. We release model checkpoints, training, and evaluation code to facilitate reproducibility and future research.

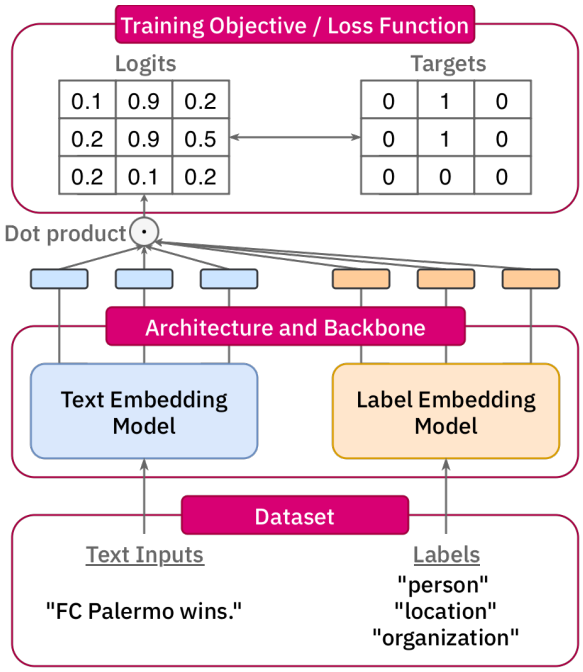

Figure 1: Overview of the design choices in universal multilingual NER systems: the dataset, backbone, architecture, and training objective. Prior work typically fixes a particular combination of these choices and studies only their joint effect. In this work, we vary each dimension in a controlled setting to isolate its impact on performance and efficiency.

1 Introduction

Multilingual named entity recognition (NER) is a token-level information extraction task (Lample et al., 2016; Akbik et al., 2018) used in applications such as knowledge graph construction (Zhong et al., 2023; Arsenyan et al., 2024), invoice parsing (Perot et al., 2024) or web search (Shachar et al., 2025). While large language models (LLMs) (Grattafiori et al., 2024; Yang et al., 2025; DeepSeek-AI et al., 2025; Team et al., 2025) are increasingly capable and improve performance on multilingual NER, their billion parameters come at a high computational cost. Instead, recent work use such models as effective teachers for training smaller, more efficient encoder-only models, as done in GLiNER (Zaratiana et al., 2024, 2025), NuNER (Bogdanov et al., 2024), or LitSet (Golde et al., 2024).

Design choices are rarely justified. However, despite this progress, many design choices in the literature are only weakly justified or briefly discussed. Because such choices vary widely across systems, it is hard to tell which ones actually drive performance, and progress becomes difficult to interpret. For instance, Zaratiana et al. (2024); Huang et al. (2022) use a cross-encoder architecture, while Zhang et al. (2023); Bogdanov et al. (2024);

Golde et al. (2024) use a bi-encoder. Likewise, `gliner-multi-v2.1`[1] uses mDeBERTa (He et al., 2023) as its backbone, whereas `gliner-x-base`[2] uses mT5 (Xue et al., 2021). To our knowledge, these choices have never been systematically ablated, so their trade-offs in compute, data efficiency, and performance remain poorly understood. In this work, we examine these core design dimensions and measure their effects on performance and efficiency.

Identifying the core design dimensions. Specifically, we identify five core dimensions along which prior work differs: *(i)* the choice between cross-encoder and bi-encoder architectures, *(ii)* the transformer backbone, *(iii)* the training objective, *(iv)* the composition of the training dataset, and *(v)* the selection of language-specific thresholds for zero-shot NER. Figure 1 illustrates these dimensions. We evaluate them within a controlled experimental setting to isolate their individual and joint effects.

Contributions. Building on these insights, we train OTTER, an optimized universal multilingual NER model. OTTER achieves strong performance across seven multilingual NER benchmarks, consistently outperforming existing models of comparable size and remaining competitive with substantially larger generative models, while also being efficient at both training and inference time. We structure the paper around the design dimensions identified above, devoting a section to each, and we release model checkpoints and code to support reproducibility and further research in multilingual NER.

We summarize our contributions as follows:

- We empirically investigate the critical design choices in universal multilingual NER, isolating the effects of architecture, transformer backbone, loss function, training data and language-specific thresholding.
- Building on these findings, we derive OTTER, a multilingual NER model supporting over 100 languages, and evaluate it zero-shot across multiple benchmarks, achieving state-of-the-art performance on many of them.
- We release the model checkpoints, training datasets, and training and evaluation code to facilitate reproducibility and further study in multilingual NER.[3]

[1] https://huggingface.co/urchade/gliner_multi-v2.1

[2] https://huggingface.co/knowledgator/gliner-x-base

2 Methodology

2.1 Dimension 1: Architecture and Backbone

At a high level, there are two main approaches for combining text inputs $\mathrm{X} = \mathrm{x}_1, \ldots, \mathrm{x}_n$ and label inputs $\mathrm{Y} = \mathrm{y}_1, \ldots, \mathrm{y}_n$. The cross-encoder approach concatenates text and label descriptions and feeds them jointly into a single transformer, allowing text tokens to directly cross-attend to label tokens (Equation (1)). This paradigm is used, for example, in GLiNER (Zaratiana et al., 2024). In contrast, bi-encoders process text and label inputs separately using two transformer encoders (Equation (2)), as in Binder (Zhang et al., 2023), and combine their representations only after encoding.

$$\mathbf{H}^{\mathrm{X}}, \mathbf{H}^{\mathrm{Y}} = f_{\mathrm{CE}}(\mathrm{X}, \mathrm{Y}), \tag{1}$$

$$\mathbf{H}^{\mathrm{X}} = f^{\mathrm{X}}_{\mathrm{BI}}(\mathrm{X}), \qquad \mathbf{H}^{\mathrm{Y}} = f^{\mathrm{Y}}_{\mathrm{BI}}(\mathrm{Y}). \tag{2}$$

The underlying encoders are typically initialized from pretrained transformer models and are trained using in-batch negatives. We note that there are also more advanced approaches to negative mining which we do not explore with this work.

Given hidden representations $\mathbf{H}^{\mathrm{X}} \in \mathbb{R}^{|\mathrm{X}| \times d}$ and $\mathbf{H}^{\mathrm{Y}} \in \mathbb{R}^{|\mathrm{Y}| \times d}$, we project token and label representations using two-layer MLPs to obtain start, end, and label embeddings (Equations (3) to (5)). Candidate spans are then represented by concatenating the corresponding start and end projections with a span-width embedding, yielding a span representation $\mathbf{k}_{i,j}$ for each span (i, j) (Equation (6)).

$$\mathbf{S} = \mathrm{MLP}_{\mathrm{START}}(\mathbf{H}^{X}), \tag{3}$$

$$\mathbf{E} = \mathrm{MLP}_{\mathrm{END}}(\mathbf{H}^{X}), \tag{4}$$

$$\mathbf{Q} = \mathrm{MLP}_{\mathrm{LABEL}}(\mathbf{H}^{L}), \tag{5}$$

$$\mathbf{k}_{i,j} = \mathrm{MLP}_{\mathrm{SPAN}}\Big(\mathbf{s}_i \oplus \mathbf{e}_j \oplus \mathbf{D}(j-i)\Big). \tag{6}$$

We then use a span representation $k_{i,j}$ to compute label-specific logits using label representations $\mathbf{Q}$ (Equation (7)).

$$\ell_{i,j,n} = \mathbf{k}_{i,j}^{\top} \mathbf{q}_n, \quad \mathbf{q}_n \in \mathbf{Q}. \tag{7}$$

Technically, we follow the ideas of GLiNER for the cross-encoder setup by introducing an additional `[LABEL]` token to obtain label representations, and we adopt the Binder formulation for the

[3] Removed for double-blind review.

bi-encoder setup. In our experiments, we sweep over both architectures using five different transformer backbones.

2.2 Dimension 2: Fine-Tuning Datasets

In parallel, several large-scale NER datasets have been released that use large language models to annotate unlabeled data, which is then used to train smaller models. These datasets typically cover large label sets following a long-tail distribution, such as PileNER (Zhou et al., 2024) or NuNER (Bogdanov et al., 2024). Rather than a fixed ontology, such label sets provide a more realistic supervision signal for generalizing to arbitrary label descriptions. In our experiments, we use datasets that vary substantially in language coverage, ranging from English-only to 91 languages. This lets us investigate whether a multilingual transformer trained only on English is sufficient for cross-lingual generalization, or whether fine-tuning on multilingual data yields additional gains.

Further, we investigate the role of thresholding in multilingual NER. Multilingual transformers tend to produce predicted spans of substantially different lengths across languages, for example shorter spans for English and longer ones for Chinese, which affects both training and prediction (cf. Section F). A single fixed decision threshold is therefore unlikely to be optimal across all languages, however, it is desirable from a usability perspective. We investigate language-specific thresholding to account for these differences.

2.3 Dimension 3: Loss Functions

The training objective determines how the model combines text and label representations, and it must handle the severe class imbalance in span-based NER, where most span–label pairs are negatives. Existing models take different approaches: Binder uses a contrastive loss to align span and label representations in a shared embedding space, whereas GLiNER applies a binary cross-entropy loss over span–label pairs. In this work, we explore a range of loss functions that all aim to improve generalization under imbalanced positive-to-negative annotations: binary cross-entropy (BCE), BCE with positive weighting, focal loss (Lin et al., 2018) with varying α and γ, contrastive loss, and dice loss (Li et al., 2020).

2.4 Modeling Without Word Segmentation

Classical named entity recognition relies on BIO tagging (Ratinov and Roth, 2009), which assigns a label to each word in the sequence. This requires the input to be split into words beforehand. Word-level tagging keeps the number of negative candidates low and simplifies training, but it has two drawbacks for our setting. First, it ties supervision to word boundaries, so the model only updates the representations at those boundaries. Second, and more importantly, splitting text into words requires an accurate word-segmentation model for each language, which is impractical when covering many languages and scripts.

To avoid this, we train directly on the raw input text and let the model handle segmentation itself, forming candidate spans over subword tokens rather than words. This increases the number of negative span candidates, but it removes the need for external, language-specific preprocessing and ensures consistent behavior across scripts. We analyze this trade-off in Section F.

3 Experiment 1: Architecture and Backbone

In this experiment, we train cross- and bi-encoder models on the PileNER dataset (English-only, Latin script) on five multilingual transformer backbones: *(i)* `xlm-roberta-base` (Conneau et al., 2020), *(ii)* `mmBERT` (Marone et al., 2025), *(iii)* `mT5-base` (Xue et al., 2021), *(iv)* `mdeberta-v3-base` (He et al., 2023) and *(v)* `rembert` (Chung et al., 2021).

Hyperparameters. We train all models for 10k steps without early stopping, using a batch size of 12 and the AdamW optimizer (Loshchilov and Hutter, 2019). Unless stated otherwise, we use a learning rate of 3×10^{-5} for all transformer backbones and MLP components, following the settings recommended in the original works. For the mT5 backbone, we use a higher learning rate of 1×10^{-3}, as suggested in the original paper. We fix the maximum sequence length to 512 tokens (1024 for mmBERT) and consider subword spans of up to length 30. We set the output dimension of all MLP projections to $d_{\text{MLP}} = 384$ and that of the span-width embedding layer to $d_{\text{width}} = 128$. For the bi-encoder setup, we use `multilingual-bert-base-uncased` (Devlin et al., 2019) as the label encoder and represent labels using the `[CLS]` token. We use standard binary

Architecture	*Backbone* mDeBERTa	mmBERT	mT5	RemBERT	XLM-R	Avg.
Bi-Encoder	0.458	**0.489**	0.425	<u>0.471</u>	0.456	0.460
Cross-Encoder	0.494	**0.529**	0.427	<u>0.528</u>	0.441	0.484
Avg.	0.476	**0.509**	0.426	<u>0.499</u>	0.448	–

Table 1: Macro-averaged F1 on the holdout validation set for each combination of architecture and transformer backbone, trained on PileNER. Within each row, the best and second-best scores are shown in bold and underlined, respectively.

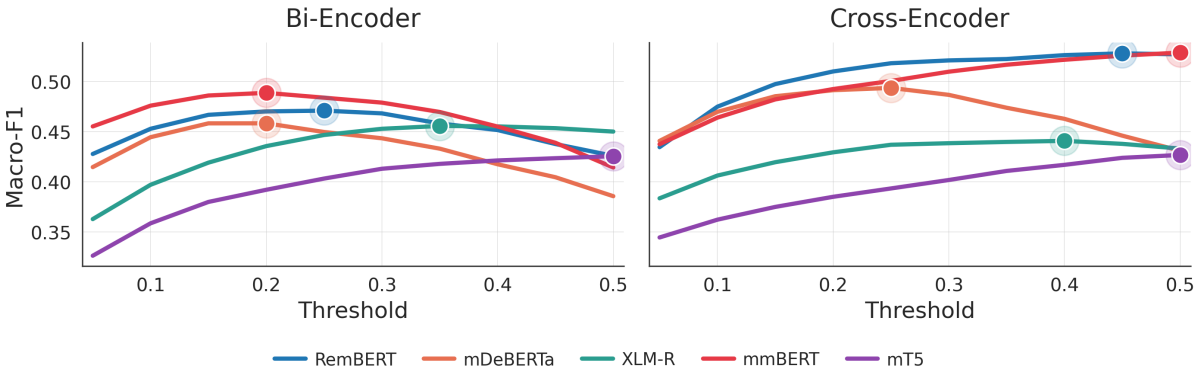

Figure 2: Macro-averaged micro-F1 scores across all backbones evaluation benchmarks with different decision thresholds t. We observe that optimal performance is dependent on the transformer backbone.

cross-entropy loss and leave the exploration of different datasets and loss functions to later sections.

For evaluation during development, we use a holdout set of eight languages (English, German, French, Spanish, Chinese, Arabic, Hindi, and Japanese), drawn from the validation splits of MultiNERD (Tedeschi and Navigli, 2022) and PAN-X (Hu et al., 2020), with 500 samples per split. We perform threshold optimization after training to study transfer across languages. While per-language thresholds can be tuned when validation data is available for each target language, this is unrealistic in a zero-shot setting, where the languages seen at inference time are not known in advance. We therefore tune a single global threshold on the holdout set and apply it across all languages, favoring robust transfer to unseen languages over per-language fitting and avoiding the need for language-specific validation data at deployment time.

Results. We report results on the validation set in Table 1, using the best global threshold for each configuration. The cross-encoder outperforms the bi-encoder on average (0.484 vs. 0.460 F1), and mmBERT is the strongest backbone overall (0.509 F1 averaged across architectures), followed closely by RemBERT (0.499 F1). For both architectures, mmBERT is the best backbone (0.489 F1 for the bi-encoder, 0.529 F1 for the cross-encoder) and RemBERT the second-best, suggesting that the relative ranking of backbones is fairly stable across architectures. Backbone choice nonetheless matters substantially: mT5 is consistently the weakest, trailing mmBERT by 0.064 F1 in the bi-encoder setting and by 0.102 F1 in the cross-encoder setting. XLM-R is the one backbone whose ranking depends on the architecture, performing reasonably in the bi-encoder setup (0.456 F1) but dropping to the lowest cross-encoder score among the strong backbones (0.441 F1).

Threshold Selection. We evaluate model performance across a range of global decision thresholds $t \in \{0.05, 0.1, 0.15, \ldots, 0.5\}$, in steps of 0.05, used to convert span–label scores into final predictions. We restrict the range to $[0.05, 0.5]$ because lower thresholds are needed to compensate for the large negative-to-positive span ratio in span-based NER. Figure 2 shows the macro-averaged micro-F1 over the eight validation splits for each backbone, with the optimal threshold per curve marked. We observe a consistent difference between the two architectures: the bi-encoder peaks at lower thresholds (roughly $t \in [0.2, 0.3]$),

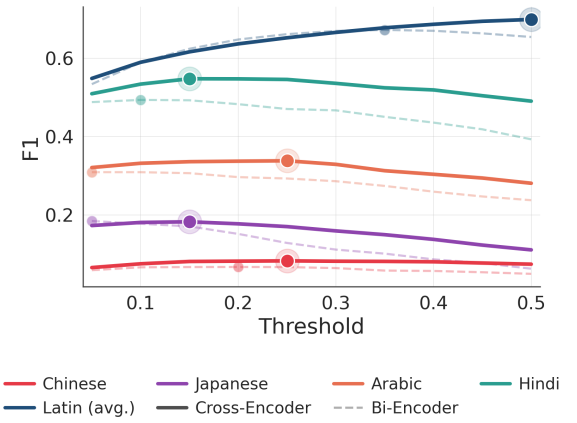

Figure 3: F1 versus decision threshold per language, macro-averaged over backbones for each architecture (cross-encoder: solid; bi-encoder: dashed). Latin (avg.) is averaged over English, German, French, and Spanish; the optimal threshold per curve is marked.

whereas the cross-encoder peaks at higher thresholds (roughly $t \in [0.4, 0.5]$). We attribute this to the way each architecture combines span and label information. The bi-encoder relies on late interaction, scoring spans and labels independently before comparing them, which tends to produce lower-confidence matches that are best captured at lower thresholds. The cross-encoder fuses span and label information early, yielding sharper, higher-confidence scores that favor higher thresholds.

Cross-encoders transfer better across scripts. To examine how thresholding interacts with different languages and scripts, we macro-average over backbones to obtain a single curve per architecture for each language, shown in Figure 3. On Latin-script languages (averaged over English, German, French, and Spanish), the two architectures behave almost identically, peaking at high thresholds. Non-Latin scripts, in contrast, peak at lower thresholds, likely because subword tokenization splits them into more fragments. The differences between architectures also emerge here: across Hindi, Arabic, Japanese, and Chinese, the cross-encoder (solid) consistently outperforms the bi-encoder (dashed), with the gap widening at higher thresholds. This suggests that early fusion of span and label information in the cross-encoder leads to better cross-lingual and cross-script transfer than the late interaction of the bi-encoder.

Performance Comparison. Using the cross- and bi-encoder with mmBERT, we compare how their cost and performance scale with the number of labels provided at inference. We extend the label set of our validation split up to 1,000 label descriptions by adding the most frequent labels from PileNER, thereby introducing noise and ambiguity, such as synonyms or more fine-grained labels. We measure micro-F1, inference latency, and FLOPs (Figure 4). We find that micro-F1 decreases for both architectures as the label set grows, indicating that larger, more confusable label sets hurt performance. The cross-encoder, however, collapses beyond 250 labels: once not all labels fit in a single forward pass, logits computed across separate batches are no longer comparable. The bi-encoder avoids this, as its scores are independent of the batch composition. In terms of cost, the bi-encoder has higher training FLOPs, but at inference its label embeddings can be cached and reused across inputs, making inference roughly linear in the label set size and far cheaper. Bi-encoder latency is nearly identical with and without caching, so we omit the training-time curve for readability.

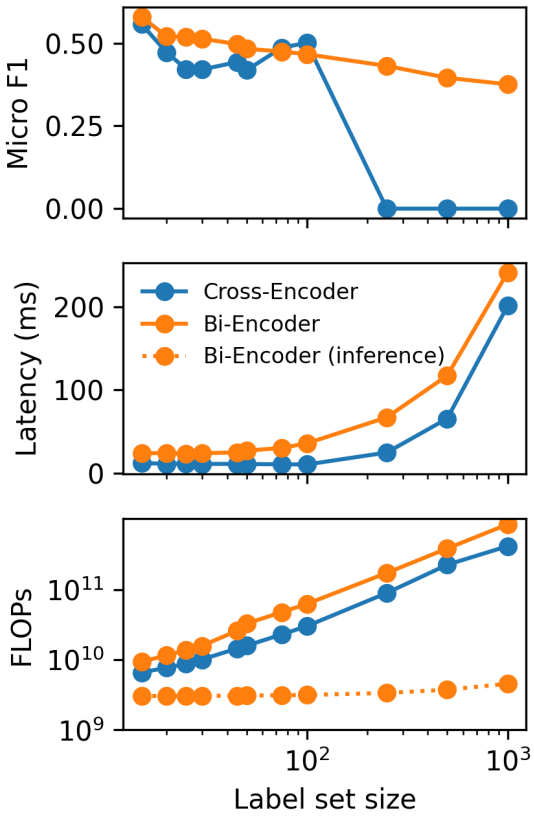

Figure 4: Micro-F1, latency, and FLOPs versus the number of inference labels for the cross- and bi-encoder (mmBERT). The dotted line shows bi-encoder inference FLOPs with cached label embeddings.

Dataset	*Bi-Encoder* RemBERT	*Bi-Encoder* mmBERT	*Cross-Encoder* RemBERT	*Cross-Encoder* mmBERT	Avg.
PileNER	0.471	0.489	0.528	**0.529**	0.504
Euro-GLiNER-x	0.488	0.575	0.534	**0.612**	0.552
FiNERweb	0.453	0.474	0.555	**0.575**	0.514

Table 2: Macro-averaged F1 across evaluation language splits per training dataset for bi-encoder and cross-encoder architectures with different transformer backbones. Best per row in **bold**, second-best underlined.

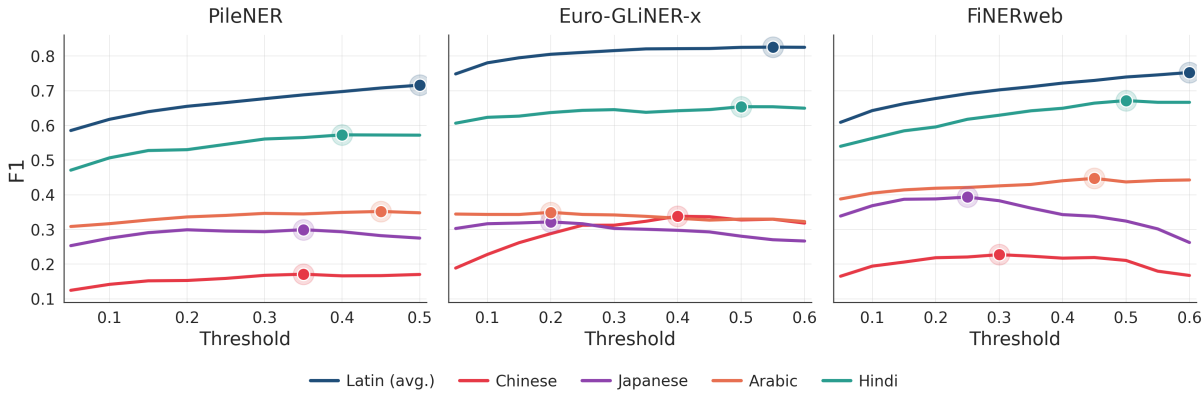

Figure 5: Macro-averaged micro-F1 scores with different decision thresholds t using different training datasets. We observe that optimal performance for each language is dependent on scripts and languages covered during training.

4 Experiment 2: Multilingual Fine-Tuning

We now analyze the effect of the fine-tuning dataset. We consider three datasets: *(i)* PileNER (Zhou et al., 2024), which is English-only; *(ii)* Euro-GLiNER-x [4], which covers 12 Indo-European languages in Latin script; and *(iii)* FiNERweb (Golde et al., 2025), which spans 91 languages and 25 scripts. All three follow a long-tail distribution of entity types. We ask whether fine-tuning a multilingual backbone on English-only data is sufficient, or whether multilingual training data is needed.

As a first indication, we tokenize each dataset with the XLM-R tokenizer and compute the fraction of subword embeddings that receive gradient updates during training (cf. Table 6). FiNERweb updates over 94% of subword embeddings, compared to only 38.55% for English-only PileNER, suggesting that much of the multilingual vocabulary is never trained under English-only fine-tuning. We otherwise reuse the experimental setup from the previous section.

Results. We report macro-averaged F1 per training dataset in Table 2. The cross-encoder outperforms the bi-encoder across all datasets and backbones, and mmBERT is the stronger backbone in every setting. Multilingual training helps overall: the best result for each architecture is obtained with a multilingual dataset rather than with PileNER. Euro-GLiNER-x is the strongest dataset on average (0.552), improving over PileNER for both architectures (e.g., cross-encoder with mmBERT: 0.529 → 0.612). FiNERweb also improves the cross-encoder over PileNER (0.529 → 0.575 for mmBERT) but, unlike Euro-GLiNER-x, does not help the bi-encoder, where it falls slightly below English-only training.

Per-language analysis. Figure 5 breaks performance down by language for the cross-encoder with mmBERT. The gains from multilingual training are uneven across scripts. On Latin-script languages, FiNERweb does not improve over the other datasets and even trails them, likely because its broad language coverage dilutes Latin-script supervision. On non-Latin scripts, however, FiNERweb already yields clear improvements, with Arabic, Hindi, and Japanese all scoring higher than under English- or Latin-only training. This indicates that broad multilingual coverage trades a small amount of Latin-script performance for substantially better transfer to underrepresented scripts.

[4] https://huggingface.co/datasets/knowledgator/gliner-multilingual-synthetic

Loss Function	*F1 Score*
Baseline (BCE)	0.575
BCE + Pos. Weight λ	
$\lambda = 10.0$	0.530
$\lambda = 100.0$	0.432
Focal Loss	
$\alpha = 0.75, \gamma = 1.0$	**0.585**
$\alpha = 0.25, \gamma = 2.0$	0.573
Contrastive Loss	
$\alpha = 0.5, \beta = 0.5$	0.046
Dice Loss	<u>0.577</u>

Table 3: Performance across evaluation datasets for different loss functions (cross-encoder, mmBERT, FiNERweb).

5 Experiment 3: Loss Functions

Using the best configuration from the previous experiments (mmBERT trained on FiNERweb), we compare loss functions: *(i)* binary cross-entropy (BCE) with optional positive upweighting, *(ii)* focal loss with varying α and γ, and *(iii)* a contrastive loss with adaptive thresholding (full definitions in Section E).

Results. We report results in Table 3. Upweighting positives does not help and instead degrades performance, mainly by shifting the decision threshold upward. Dice loss roughly matches BCE (0.577 vs. 0.575), and contrastive loss fails to train competitively in our setting (0.046), despite removing the need for a fixed threshold. Focal loss gives the best result (0.585 with α=0.75, γ=1.0), a marginal improvement over BCE. We attribute this small gain to the breadth of label descriptions in FiNERweb: with many overlapping and fine-grained labels, the down-weighting of easy negatives in focal loss provides a modest benefit over BCE.

6 Combining The Insights

We finally combine our findings to obtain OTTER, training our best architecture–backbone combination on FiNERweb for 30k steps without early stopping, using the mmBERT backbone and binary cross-entropy loss. We choose FiNERweb for its broad language and script coverage, and BCE since the more advanced loss functions yielded only marginal gains. Based on our threshold analysis, we adopt a single global threshold per architecture, $t = 0.3$ for the cross-encoder and $t = 0.2$ for the bi-encoder, choosing values at the lower end of the optimal range since non-Latin scripts peak at lower thresholds and we evaluate across many languages and splits. To situate OTTER against the current related work, we compare it to two families of baselines: general-purpose multilingual LLMs and dedicated universal NER models. The former includes GPT-5[5], Qwen3-32B (Yang et al., 2025), and Gemma3-27B (Team et al., 2025); the latter includes WikiNeural (Tedeschi et al., 2021) and two multilingual GLiNER variants (Zaratiana et al., 2024).

Evaluation Benchmarks. We use seven multilingual datasets for evaluation: DynamicNER (Luo et al., 2025), UNER (Mayhew et al., 2024), Masakhaner 2.0 (Adelani et al., 2022), MultiNERD (Tedeschi and Navigli, 2022), MultiCoNER v1 (Malmasi et al., 2022) and v2 (Fetahu et al., 2023) and PAN-X (Hu et al., 2020). We provide an overview of the number of languages and label set sizes in Table 5. As our evaluation covers 250 test splits in total, we limit each test split at 1,000 examples when it is larger. We report all results using micro-averaged F1 within each dataset across languages and a macro-averaged F1 across datasets when reporting aggregate results.

Results. We report all baselines and OTTER in Table 4. Our best cross-encoder reaches 0.484 F1, the strongest result among similarly sized task-specific models, beating GLiNER-x-base (0.431) by 5.3 pp. With per-language thresholds (OTTER*), it improves to 0.501 F1, matching Qwen3-32B (0.503) within 0.2 pp despite being roughly 90× smaller. This reinforces our finding from Section 3 that per-language thresholding matters, as tokenization varies across languages.

The cross-encoder outperforms the bi-encoder at scale (0.484 vs. 0.437 for mmBERT), unlike our earlier holdout experiments where the two were comparable. We attribute this to its early fusion of span and label information, which better captures language-specific patterns such as the varying positive-to-negative span ratio across languages.

Among LLMs, Gemma3-27B is strongest overall (0.543), ahead of Qwen3-32B (0.503), while GPT-5 is surprisingly weak (0.347). Overall, a carefully designed task-specific model matches Qwen3-32B at a fraction of the size and compute.

[5] https://platform.openai.com/docs/models/gpt-5

Model	*Datasets* Dynamic-NER	Masakha-NER	MultiCoNER v1	MultiCoNER v2	Multi-NERD	PAN-X	UNER	Avg.
LLMs								
GPT-5	0.204	0.388	0.267	0.133	0.496	0.477	0.468	0.347
Qwen3-32B	0.365	0.535	0.419	0.349	0.617	0.554	0.681	0.503
Gemma3-27B	0.434	0.594	0.428	0.373	0.646	0.584	0.742	0.543
Universal NER Models								
WikiNeural	0.001	0.307	0.097	0.013	0.652	0.403	0.506	0.283
GLiNER-multi-v2.1	0.291	0.480	0.366	0.238	0.533	0.532	0.551	0.427
GLiNER-x-base	0.187	0.559	0.329	0.210	0.582	0.509	0.644	0.431
OTTER (BI-ENC.)								
w/ RemBERT	0.166	0.541	0.351	0.156	0.595	0.508	0.704	0.432
w/ mmBERT	0.251	0.485	0.355	0.182	0.576	0.479	0.661	0.427
OTTER (CROSS-ENC.)								
w/ RemBERT	0.354	0.573	0.347	0.294	0.636	0.436	0.540	0.454
w/ mmBERT	0.382	0.511	0.358	0.254	0.638	0.535	0.713	0.484
w/ mmBERT*	0.385	0.523	0.369	0.265	0.661	0.549	0.754	0.501

Table 4: Macro-averaged F1 scores across NER benchmarks. We highlight the best average performance in bold. *Using the best performing threshold per language before aggregating.

7 Related Work

Natural Language Prompting with LLMs. The advent of increasingly capable autoregressive language models has introduced a new paradigm based on natural language prompting (Brown et al., 2020; Schick and Schütze, 2021; Min et al., 2022), effectively replacing the fixed output layer. This paradigm has been widely applied to information extraction tasks, including text classification (Halder et al., 2020; Sun et al., 2023), entity linking (Cao et al., 2021; Ding et al., 2024), and named entity recognition (Huang et al., 2022; Ashok and Lipton, 2023), as well as joint modeling across multiple tasks (Wang et al., 2023).

Knowledge Distillation from Synthetic Datasets. A major drawback of large language models is their substantial computational cost. Consequently, more recent work no longer relies on the autoregressive generation process at inference time, but instead adopts a knowledge distillation framework (Hinton et al., 2015). In this setup, LLMs serve as teachers to produce annotated datasets in a one-off process (Ye et al., 2022). This approach has been successfully applied to named entity recognition (Zhou et al., 2024; Bogdanov et al., 2024; Golde et al., 2025).

Universal NER. Recent work such as Binder (Zhang et al., 2023) and NuNER (Bogdanov et al., 2024) employs bi-encoder architectures, whereas USM (Lou et al., 2023) and GLiNER (Zaratiana et al., 2024) rely on cross-encoders. However, these models differ substantially in their loss functions, transformer backbones, and training data. In contrast, our work compares these design choices in a controlled experimental setting. Finally, our work follows the same research direction as Huang et al. (2019), but using more recent modeling approaches and training paradigms for universal NER.

8 Conclusion

In this work, we systematically explored the design space of prior approaches to universal NER. Based on these insights, we derived OTTER, a new state-of-the-art universal NER model that generalizes to more than 100 languages. Our results indicate that *(i)* multilingual training data is essential for universal NER, *(ii)* the effectiveness of a transformer backbone strongly depends on the chosen architecture, *(iii)* cross-encoders outperform bi-encoders at scale in terms of generalization, *(iv)* a simple binary cross-entropy loss is sufficient, and *(v)* threshold selection plays a critical role in universal, multilingual NER.

Limitations

Pretrained model and data constraints. Our proposed approach is limited by the availability and quality of pretrained multilingual language models and training data, as the design of OTTER is empirically derived from existing architectures and datasets. In this work, the maximum languages supported in one training dataset is 91.

Language coverage. Although we evaluate OTTER on more than 150 languages, only a subset of these languages is represented in the training data. As a result, performance may degrade for languages outside the investigated training scope, in particular for low-resource languages or scripts that are underrepresented in the pretrained models and/or the training data.

Label semantics. The evaluation benchmarks considered in this work use label sets with distinct class boundaries such as "person" and "location". We do not explicitly investigate the effect of semantic similarity between entity labels, e.g. "person" and "human", which may contribute to the observed performance differences when training with labels in the target language because we train the model to treat these labels as distinct concepts.

Threshold Selection. We find that threshold selection for zero-shot NER is highly language-dependent, and therefore select the threshold based on a hold-out validation set. Thus, the performance gap to in-domain fine-tuned models can remain substantial, as label definitions and, in particular, span boundary conventions differ across datasets. As a result, our model can serve as a suitable starting point for further fine-tuning on target datasets, where dataset-specific label and boundary definitions can be incorporated.

References

David Ifeoluwa Adelani, Graham Neubig, Sebastian Ruder, Shruti Rijhwani, Michael Beukman, Chester Palen-Michel, Constantine Lignos, Jesujoba O. Alabi, Shamsuddeen H. Muhammad, Peter Nabende, Cheikh M. Bamba Dione, Andiswa Bukula, Rooweither Mabuya, Bonaventure F. P. Dossou, Blessing Sibanda, Happy Buzaaba, Jonathan Mukiibi, Godson Kalipe, Derguene Mbaye, and 26 others. 2022. MasakhaNER 2.0: Africa-centric transfer learning for named entity recognition. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4488–4508, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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.

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, and 30 others. 2024. Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS '24, page 929–947, New York, NY, USA. Association for Computing Machinery.

Vahan Arsenyan, Spartak Bughdaryan, Fadi Shaya, Kent Wilson Small, and Davit Shahnazaryan. 2024. Large language models for biomedical knowledge graph construction: Information extraction from EMR notes. In *Proceedings of the 23rd Workshop on Biomedical Natural Language Processing*, pages 295–317, Bangkok, Thailand. Association for Computational Linguistics.

Dhananjay Ashok and Zachary C. Lipton. 2023. Promptner: Prompting for named entity recognition. *Preprint*, arXiv:2305.15444.

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.

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, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. *Preprint*, arXiv:2010.00904.

Hyung Won Chung, Thibault Fevry, Henry Tsai, Melvin Johnson, and Sebastian Ruder. 2021. Rethinking embedding coupling in pre-trained language models. In *International Conference on Learning Representations*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco

Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

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.

Yifan Ding, Qingkai Zeng, and Tim Weninger. 2024. ChatEL: Entity linking with chatbots. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 3086–3097, Torino, Italia. ELRA and ICCL.

Besnik Fetahu, Zhiyu Chen, Sudipta Kar, Oleg Rokhlenko, and Shervin Malmasi. 2023. MultiCoNER v2: a large multilingual dataset for fine-grained and noisy named entity recognition. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2027–2051, Singapore. Association for Computational Linguistics.

Jonas Golde, Patrick Haller, and Alan Akbik. 2025. Finerweb: Datasets and artifacts for scalable multilingual named entity recognition. *Preprint*, arXiv:2512.13884.

Jonas Golde, Felix Hamborg, and Alan Akbik. 2024. Large-scale label interpretation learning for few-shot named entity recognition. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2915–2930, St. Julian's, Malta. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Kishaloy Halder, Alan Akbik, Josip Krapac, and Roland Vollgraf. 2020. Task-aware representation of sentences for generic text classification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3202–3213, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. *Preprint*, arXiv:2111.09543.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *Preprint*, arXiv:1503.02531.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4411–4421. PMLR.

Xiaolei Huang, Jonathan May, and Nanyun Peng. 2019. What matters for neural cross-lingual named entity recognition: An empirical analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6395–6401, Hong Kong, China. Association for Computational Linguistics.

Yucheng Huang, Kai He, Yige Wang, Xianli Zhang, Tieliang Gong, Rui Mao, and Chen Li. 2022. COPNER: Contrastive learning with prompt guiding for few-shot named entity recognition. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 2515–2527, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. 2020. Dice loss for data-imbalanced NLP tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 465–476, Online. Association for Computational Linguistics.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2018. Focal loss for dense object detection. *Preprint*, arXiv:1708.02002.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *Preprint*, arXiv:1711.05101.

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. *Preprint*, arXiv:2301.03282.

Hanjun Luo, Yingbin Jin, Yiran Wang, Xinfeng Li, Tong Shang, Xuecheng Liu, Ruizhe Chen, Kun Wang, Hanan Salam, Qingsong Wen, and Zuozhu Liu. 2025. DynamicNER: A dynamic, multilingual, and fine-grained dataset for LLM-based named entity recognition. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 16522–16546, Suzhou, China. 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 *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3798–3809, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Marc Marone, Orion Weller, William Fleshman, Eugene Yang, Dawn Lawrie, and Benjamin Van Durme. 2025. mmbert: A modern multilingual encoder with annealed language learning. *Preprint*, arXiv:2509.06888.

Stephen Mayhew, Terra Blevins, Shuheng Liu, Marek Šuppa, Hila Gonen, Joseph Marvin Imperial, Börje F. Karlsson, Peiqin Lin, Nikola Ljubešić, LJ Miranda, Barbara Plank, Arij Riabi, and Yuval Pinter. 2024. Universal NER: A gold-standard multilingual named entity recognition benchmark. 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 4322–4337, Mexico City, Mexico. 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? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Vincent Perot, Kai Kang, Florian Luisier, Guolong Su, Xiaoyu Sun, Ramya Sree Boppana, Zilong Wang, Zifeng Wang, Jiaqi Mu, Hao Zhang, Chen-Yu Lee, and Nan Hua. 2024. LMDX: Language model-based document information extraction and localization. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 15140–15168, Bangkok, Thailand. Association for Computational Linguistics.

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.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

Or Shachar, Uri Katz, Yoav Goldberg, and Oren Glickman. 2025. NER retriever: Zero-shot named entity retrieval with type-aware embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 11175–11186, Suzhou, China. Association for Computational Linguistics.

Xiaofei Sun, Xiaoya Li, Jiwei Li, Fei Wu, Shangwei Guo, Tianwei Zhang, and Guoyin Wang. 2023. Text classification via large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8990–9005, Singapore. Association for Computational Linguistics.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Simone Tedeschi, Valentino Maiorca, Niccolò Campolungo, Francesco Cecconi, and Roberto Navigli. 2021. WikiNEuRal: Combined neural and knowledge-based silver data creation for multilingual NER. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2521–2533, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Simone Tedeschi and Roberto Navigli. 2022. MultiNERD: A multilingual, multi-genre and fine-grained dataset for named entity recognition (and disambiguation). In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 801–812, Seattle, United States. Association for Computational Linguistics.

Xiao Wang, Weikang Zhou, Can Zu, Han Xia, Tianze Chen, Yuansen Zhang, Rui Zheng, Junjie Ye, Qi Zhang, Tao Gui, Jihua Kang, Jingsheng Yang, Siyuan Li, and Chunsai Du. 2023. Instructuie: Multi-task instruction tuning for unified information extraction. *Preprint*, arXiv:2304.08085.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz,

Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. 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.

Jiacheng Ye, Jiahui Gao, Qintong Li, Hang Xu, Jiangtao Feng, Zhiyong Wu, Tao Yu, and Lingpeng Kong. 2022. ZeroGen: Efficient zero-shot learning via dataset generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11653–11669, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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

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.

Sheng Zhang, Hao Cheng, Jianfeng Gao, and Hoifung Poon. 2023. Optimizing bi-encoder for named entity recognition via contrastive learning. In *The Eleventh International Conference on Learning Representations*.

Lingfeng Zhong, Jia Wu, Qian Li, Hao Peng, and Xindong Wu. 2023. A comprehensive survey on automatic knowledge graph construction. *ACM Computing Surveys*, 56(4):1–62.

Wenxuan Zhou, Sheng Zhang, Yu Gu, Muhao Chen, and Hoifung Poon. 2024. Universalner: Targeted distillation from large language models for open named entity recognition. *Preprint*, arXiv:2308.03279.

Appendix

A Implementation and Compute

We use for all our experiments the transformers library (Wolf et al., 2020) and PyTorch (Ansel et al., 2024). We conducted all experiments on 8 NVIDIA RTX A6000 GPUs with 48GB VRAM each.

For LLMs, we use the default hyperparameters for generation and re-use the prompt and substring extraction procedure as introduced in Golde et al. (2025).

B Evaluation Benchmarks

We show an overview of all evaluation benchmarks used in Table 5.

Dataset	*# Langs*	*# Labels*
PAN-X	176	3
MasakhaNER	20	4
UNER	13	3
MultiCoNER v2	12	33
MultiCoNER v1	11	6
MultiNERD	10	15
DynamicNER	8	155

Table 5: Overview of evaluation benchmarks used in our experiments, showing the number of supported languages and number of entity types.

C Detailed Results

We show detailed results for selected experiments in Tables 9 and 10.

D Error Analysis

As we do not rely on word segmentation and instead learn entity boundaries implicitly at the embedding level, we observe negative implications for languages with productive prefixation. For example, on the Shona split of MasakhaNER, we find that our model achieves an F1 score of 0.298, which is substantially lower than comparable baselines such as GLiNER (above 0.6 F1 for both variants). By inspecting the model predictions, we observe that locative or associative prefixes attached to named entities (e.g., *eZimbabwe* “in Zimbabwe”,

Original Inputs: [“Mapurisa”, “eZimbabwe”, “Republic”, “Police”, . . .]
Gold: `[O, ORG, ORG, ORG, . . . ]`
Subword Tokenized: `[_Map, ur, isa, _e, Zimbabwe, _Republic, _Police, . . . ]`
Pred: `[O, O, O, O, ORG, ORG, ORG, ]`

Figure 6: Example of a subword-boundary error. The prefix marker in `eZimbabwe` is labeled as O (red), while the following subwords are correctly predicted as ORG (green).

paVaMugabe “at Mr. Mugabe”) lead to discrepancies between our predicted spans and the evaluation format. Considering the range of languages covered by the training data, we find that such constructions are comparatively rare, which leads the model to predict entity spans starting at the lexical stem (e.g., *Zimbabwe Republic Police*), resulting in boundary mismatches with the gold annotations (Figure 6). While this behavior does not conform to the annotation guidelines, we consider it linguistically plausible.

E Loss Function Definitions

For each candidate span $s_{i,j}$, our models predict a score for every entity label e_k. We denote by $p_{i,j,k}$ the predicted probability for span $s_{i,j}$ for label e_k. Further, let k^* denote the gold label index for span $s_{i,j}$ (or $\varnothing$ if the span is negative).

Binary cross-entropy loss. For a given span $s_{i,j}$ with gold label k^*, binary cross-entropy computes the negative log-likelihood over the model’s predicted scores for each label:

$$\ell_{\text{BCE}} = -\log p_{i,j,k^*} - \sum_{k \neq k^*} \log\big(1 - p_{i,j,k}\big). \quad (8)$$

Focal loss. For a given span $s_{i,j}$ and a label k, we define

$$p_t = \begin{cases} p_{i,j,k^*} & \text{if } k = k^*, \\ 1 - p_{i,j,k} & \text{otherwise.} \end{cases} \quad (9)$$

The focal loss is then

$$\ell_{\text{Focal}}(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log(p_t). \quad (10)$$

The focusing parameter γ controls the down-weighting of easy examples and when $\gamma = 0$, the loss reduces to classical cross-entropy. The weighting factor α balances positive and negative examples and can be set using inverse class frequencies or tuned as a hyperparameter. This loss follows the idea by Lin et al. (2018).

Contrastive loss. Following Zhang et al. (2023), we treat $(s_{i,j}, e_{k^*})$ as a positive pair and contrast it against a set of negative spans $S^-_{k^*}$ for the same label, where $\text{sim}(\cdot,\cdot) = \cos(\cdot,\cdot)/\tau$ is a temperature-scaled cosine similarity. This yields the span-based typing objective:

$$\ell_{\text{span}} = -\log \frac{\exp\big(\text{sim}(s_{i,j}, e_{k^*})\big)}{\sum_{s' \in S^-_{k^*} \cup \{s_{i,j}\}} \exp\big(\text{sim}(s', e_{k^*})\big)}. \quad (11)$$

The typing objective pushes entity spans close to their label embedding, but does not specify how close a span must be to be predicted as positive at test time. To separate entity from non-entity spans without an explicit `Outside` class, we add a dynamic thresholding objective. We use the similarity between a special threshold span $s_{0,0}$ (derived from the `[CLS]` token, which summarizes the full input) and the label embedding e_{k^*} as a per-label, input-specific threshold, and learn it jointly with the typing objective:

$$\ell_{\text{thr}} = -\log \frac{\exp\big(\text{sim}(s_{0,0}, e_{k^*})\big)}{\sum_{s' \in S^-_{k^*}} \exp\big(\text{sim}(s', e_{k^*})\big)}. \quad (12)$$

The two objectives are combined into a single contrastive loss,

$$\ell_{\text{Con}} = \alpha\, \ell_{\text{span}} + \beta\, \ell_{\text{thr}}, \quad (13)$$

where α and β weight the typing and thresholding objectives, respectively. We set $\alpha = \beta = 0.5$, giving equal weight to both. At inference, a span is predicted as positive for label k^* if $\text{sim}(s_{i,j}, e_{k^*}) > \text{sim}(s_{0,0}, e_{k^*})$, i.e. if it is more similar to the label than the learned threshold.

Dice loss. Dice loss optimizes the soft Sørensen–Dice coefficient between predicted and target spans, directly targeting overlap rather than per-pair classification (Li et al., 2020). Let $p_{i,j,k} = \sigma(\text{sim}(s_{i,j}, e_k))$ be the predicted probability that span $s_{i,j}$ has label k, and $y_{i,j,k} \in \{0, 1\}$ the corresponding target. The loss is defined as:

$$\ell_{\text{Dice}} = 1 - \frac{2 \sum_{i,j,k} p_{i,j,k}\, y_{i,j,k} + \gamma}{\sum_{i,j,k} p_{i,j,k} + \sum_{i,j,k} y_{i,j,k} + \gamma}, \quad (14)$$

where γ is a smoothing constant that stabilizes training and avoids division by zero. Because the

Dataset	*Subword Tokens With Gradients (%)*
PileNER	38.55
Euro-GLiNER-x	46.30
FiNERWeb	94.16

Table 6: Fraction of subword embeddings updated during training.

score depends on the relative overlap between positives and predictions rather than on every negative span individually, dice loss is naturally robust to the severe positive–negative imbalance in span-based NER.

F The Impact of Word-Segmented Inputs

We do not use word-segmented text when training our models, although this practice is common in prior work. Word segmentation allows masking irrelevant spans during training, for example when a span covers multiple words and each word consists of several subwords (cf. Figure 6). In such cases, the model does not need to compute loss terms for subword combinations that do not form valid spans. However, this approach requires language-specific word segmentation models. Maintaining such models for more than 100 languages introduces substantial manual overhead. We therefore omit word segmentation and instead let the model learn span boundaries implicitly. Concretely, we treat all subword combinations up to a maximum span length l as candidate spans.

To analyze the effect of this design choice, we select two whitespace-separated languages from FiNERweb (English and Swahili) and two non-whitespace-separated languages (Thai and Chinese). For each language, we subword-tokenize the data in two ways: using raw text with span labels based on character offsets, and using word-segmented inputs with span labels having word boundaries. We apply Stanza (Qi et al., 2020) for word segmentation in Chinese and Thai, and simple whitespace splitting for English and Swahili. We then remap span annotations from the character level to the token level and enumerate all valid spans up to length $l = 30$, both with and without word-segmentation constraints.

We report the resulting positive-to-negative span ratios in Figure 7. Using word-segmented inputs consistently increases the positive-to-negative ratio across all languages and tokenizers. Without word segmentation, the ratios vary substantially across languages and tokenizers, whereas using word-segmentated inputs largely aligns the ratios for English and Swahili across tokenizers. While our approach removes the need for explicit word segmentation, it requires the model to learn language-specific boundary behavior from subword representations alone. This choice leads to lower positive-to-negative ratios during training. However, our experiments indicate that the models can handle this setting.

G Train-Eval Distributional Similarity

To verify that the improvements from multilingual training cannot be explained by distributional matching between the training and evaluation data, we measure the lexical and semantic similarity between the training corpora and the evaluation benchmark. We use the English, German, and Russian evaluation splits of MultiNERD and compare each against same-language samples drawn from PileNER, Euro-GLiNER-x, and FiNERweb, using 20 subsamples of 5,000 sentences per language. We report two metrics: (i) the character 3–5-gram Jensen–Shannon divergence between the n-gram distributions, capturing lexical similarity, and (ii) the Frechet distance between sentence-embedding distributions, capturing semantic similarity. Sentence embeddings are computed with paraphrase-multilingual-MiniLM-L12-v2. For both metrics, lower values indicate greater similarity to MultiNERD. PileNER is English-only and Euro-GLiNER-x does not cover Russian, so the corresponding cells are empty.

Training Corpus	ENG	DEU	RUS
Lexical (n-gram JS divergence)			
FiNERweb	0.153	0.170	0.235
Euro-GLiNER-x	0.157	0.136	–
PileNER	0.160	–	–
Semantic (Frechet distance)			
FiNERweb	2.339	1.844	2.413
Euro-GLiNER-x	0.959	0.379	–
PileNER	3.545	–	–

Table 7: Lexical and semantic distance between each training corpus and the MultiNERD evaluation splits (English, German, Russian), averaged over 20 subsamples of 5,000 sentences per language. Lower is more similar.

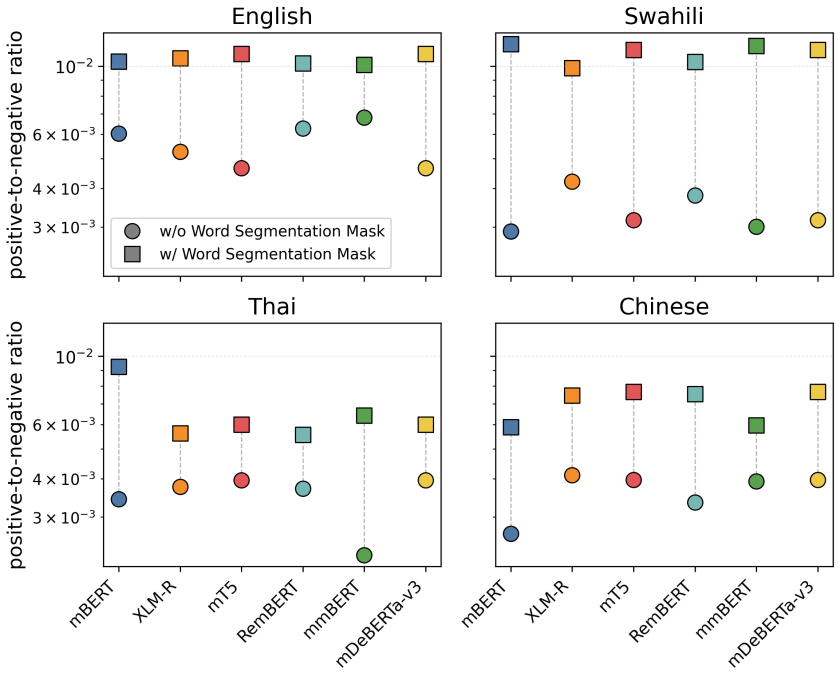

Figure 7: The impact of using pre-tokenized text for training. We can use word boundaries to exclude spans from the loss resulting the higher positive-to-negative ratios.

The results show that FiNERweb is not the closest corpus to MultiNERD on either metric. On the semantic measure, Euro-GLiNER-x is substantially closer to MultiNERD than FiNERweb for both English (0.959 vs. 2.339) and German (0.379 vs. 1.844), and lexically the three corpora are comparable, with FiNERweb only marginally closer for English. Despite this, FiNERweb yields the strongest downstream performance, particularly on non-Latin scripts. This indicates that the gains from broad multilingual training do not stem from FiNERweb being more distributionally similar to the evaluation data, but from genuinely broader language and script coverage.

H Smaller and Quantized LLM Baselines

Smaller and quantized LLMs are an important point of comparison for efficiency. In our main experiments we already use FP8-quantized versions of Qwen and Gemma. To further isolate the efficiency trade-off, we conduct an additional ablation on the English, German, and Russian splits of MultiNERD (1,000 examples each), comparing our best-performing cross-encoder against the smaller Qwen3-0.6B and Qwen3-4B variants. We measure latency per example as a single encoder forward pass for Otter and a single generation step for the autoregressive models, using batch size 1 for a fair comparison. All measurements are taken on a single H100.

Model	F1	Latency (ms)	VRAM (GB)
English			
Qwen3-0.6B	0.197	1029.9	1.25
Qwen3-4B	0.475	1111.1	7.73
Otter	**0.769**	**18.5**	1.43
German			
Qwen3-0.6B	0.241	939.6	1.20
Qwen3-4B	0.484	1316.2	7.75
Otter	**0.709**	**18.3**	1.44
Russian			
Qwen3-0.6B	0.043	1526.1	1.25
Qwen3-4B	0.274	1338.1	7.75
Otter	**0.588**	**18.3**	1.45

Table 8: F1, per-example latency (batch size 1, single H100), and VRAM on the English, German, and Russian splits of MultiNERD (1,000 examples each). Best per language in bold.

Even against smaller or quantized LLMs, the autoregressive baselines either incur substantially higher latency or achieve lower performance than the distilled encoder-only model. Otter outperforms Qwen3-4B by a wide margin in F1 on all three languages while being roughly 50–80 times faster per example and using a fraction of the memory. We do not report exact latencies for our largest baselines (about 30B parameters), since these were run through inference providers; our local FP8 mea-

surements place their per-example latency on the order of several seconds, though this could be reduced with batching. Overall, these results support our claim that distilled encoder models offer a substantially better performance-to-cost trade-off for large-scale multilingual NER.

Arc.	Dataset	τ	Dynamic-NER	Masakha-NER	MultiCoNER v1	MultiCoNER v2	Multi-NERD	PAN-X	UNER	Avg.
Bi-Encoder	PileNER	0.050	0.107	0.495	0.277	0.095	0.464	0.420	0.560	0.345
		0.100	0.113	0.514	0.289	0.095	0.534	0.418	0.614	0.368
		0.150	0.113	0.517	0.287	0.090	0.577	0.410	0.642	0.377
		0.200	0.113	0.516	0.279	0.082	0.606	0.402	0.655	0.379
		0.300	0.110	0.500	0.258	0.062	0.633	0.377	0.667	0.373
		0.400	0.099	0.470	0.233	0.038	0.637	0.352	0.662	0.356
		0.500	0.082	0.430	0.202	0.019	0.627	0.321	0.645	0.332
	Euro-GLiNER-x	0.050	0.064	0.549	0.261	0.100	0.675	0.490	0.700	0.406
		0.100	0.059	0.552	0.263	0.101	0.699	0.488	0.718	0.411
		0.150	0.053	0.553	0.260	0.100	0.710	0.484	0.724	0.412
		0.200	0.046	0.550	0.253	0.097	0.718	0.480	0.728	0.410
		0.300	0.042	0.544	0.230	0.091	0.730	0.470	0.729	0.405
		0.400	0.036	0.537	0.212	0.086	0.740	0.461	0.731	0.400
		0.500	0.034	0.529	0.198	0.079	0.750	0.449	0.735	0.396
	FiNERWeb	0.050	0.145	0.505	0.328	0.137	0.491	0.510	0.608	0.389
		0.100	0.158	0.538	0.346	0.149	0.543	0.517	0.677	0.418
		0.150	0.164	0.545	0.352	0.154	0.574	0.515	0.702	0.430
		0.200	0.166	0.541	0.351	0.156	0.595	0.508	0.704	0.432
		0.300	0.164	0.516	0.331	0.144	0.612	0.489	0.681	0.420
		0.400	0.160	0.468	0.287	0.113	0.603	0.451	0.632	0.388
		0.500	0.145	0.397	0.233	0.067	0.557	0.394	0.541	0.333
Cross-Encoder	PileNER	0.050	0.199	0.049	0.252	0.190	0.506	0.094	0.045	0.191
		0.100	0.216	0.049	0.223	0.206	0.562	0.102	0.060	0.203
		0.150	0.229	0.047	0.192	0.209	0.593	0.100	0.069	0.206
		0.200	0.230	0.044	0.161	0.207	0.612	0.098	0.071	0.203
		0.300	0.225	0.034	0.108	0.192	0.627	0.086	0.073	0.192
		0.400	0.204	0.024	0.067	0.166	0.625	0.064	0.049	0.171
		0.500	0.166	0.014	0.036	0.133	0.603	0.044	0.027	0.146
	Euro-GLiNER-x	0.050	0.086	0.453	0.226	0.097	0.674	0.466	0.641	0.378
		0.100	0.079	0.437	0.177	0.066	0.682	0.444	0.641	0.361
		0.150	0.068	0.417	0.137	0.040	0.677	0.422	0.622	0.341
		0.200	0.052	0.392	0.098	0.023	0.662	0.399	0.604	0.319
		0.300	0.032	0.337	0.038	0.006	0.608	0.343	0.546	0.273
		0.400	0.016	0.272	0.011	0.001	0.524	0.282	0.468	0.225
		0.500	0.007	0.197	0.004	0.000	0.413	0.216	0.370	0.173
	FiNERWeb	0.050	0.303	0.517	0.321	0.260	0.550	0.471	0.493	0.416
		0.100	0.324	0.551	0.340	0.276	0.584	0.473	0.524	0.439
		0.150	0.337	0.564	0.349	0.284	0.604	0.467	0.533	0.448
		0.200	0.345	0.571	0.351	0.290	0.619	0.460	0.542	0.454
		0.300	0.354	0.573	0.347	0.294	0.636	0.436	0.540	0.454
		0.400	0.361	0.563	0.334	0.290	0.649	0.405	0.515	0.446
		0.500	0.358	0.532	0.314	0.283	0.659	0.363	0.461	0.424

Table 9: Detailed results for `rembert-base`.

Arc.	Dataset	τ	Dynamic-NER	Masakha-NER	MultiCoNER v1	MultiCoNER v2	Multi-NERD	PAN-X	UNER	Avg.
Bi-Encoder	PileNER	0.050	0.150	0.427	0.289	0.129	0.497	0.451	0.508	0.350
		0.100	0.163	0.439	0.291	0.135	0.543	0.457	0.551	0.368
		0.150	0.169	0.434	0.282	0.136	0.566	0.452	0.574	0.373
		0.200	0.170	0.425	0.270	0.134	0.580	0.443	0.592	0.373
		0.300	0.168	0.399	0.236	0.125	0.586	0.421	0.618	0.365
		0.400	0.151	0.362	0.203	0.116	0.567	0.389	0.606	0.342
		0.500	0.127	0.322	0.172	0.102	0.528	0.347	0.575	0.310
	Euro-GLiNER-x	0.050	0.074	0.505	0.298	0.087	0.660	0.496	0.753	0.410
		0.100	0.057	0.502	0.268	0.076	0.680	0.481	0.765	0.404
		0.150	0.040	0.497	0.233	0.064	0.690	0.462	0.762	0.393
		0.200	0.031	0.490	0.199	0.052	0.694	0.444	0.755	0.381
		0.300	0.021	0.473	0.141	0.033	0.693	0.409	0.743	0.359
		0.400	0.014	0.448	0.096	0.022	0.684	0.373	0.715	0.336
		0.500	0.008	0.417	0.061	0.014	0.668	0.332	0.667	0.310
	FiNERWeb	0.050	0.218	0.548	0.325	0.165	0.532	0.509	0.627	0.418
		0.100	0.237	0.553	0.343	0.175	0.569	0.509	0.670	0.437
		0.150	0.245	0.523	0.353	0.180	0.581	0.497	0.675	0.436
		0.200	0.251	0.485	0.355	0.182	0.576	0.479	0.661	0.427
		0.300	0.253	0.401	0.341	0.182	0.554	0.436	0.597	0.395
		0.400	0.249	0.334	0.315	0.175	0.524	0.390	0.518	0.358
		0.500	0.227	0.273	0.275	0.158	0.483	0.335	0.418	0.310
Cross-Encoder	PileNER	0.050	0.223	0.321	0.175	0.084	0.477	0.362	0.468	0.301
		0.100	0.208	0.359	0.181	0.087	0.536	0.356	0.535	0.323
		0.150	0.174	0.351	0.175	0.082	0.562	0.338	0.545	0.318
		0.200	0.137	0.327	0.162	0.071	0.572	0.314	0.534	0.303
		0.300	0.080	0.265	0.131	0.049	0.571	0.263	0.473	0.262
		0.400	0.036	0.195	0.096	0.029	0.547	0.216	0.370	0.213
		0.500	0.015	0.125	0.062	0.015	0.503	0.167	0.244	0.162
	Euro-GLiNER-x	0.050	0.189	0.477	0.279	0.141	0.696	0.465	0.681	0.418
		0.100	0.193	0.495	0.304	0.144	0.720	0.469	0.712	0.434
		0.150	0.193	0.498	0.311	0.138	0.731	0.469	0.727	0.438
		0.200	0.186	0.497	0.309	0.129	0.739	0.470	0.732	0.437
		0.300	0.166	0.492	0.297	0.110	0.749	0.469	0.741	0.432
		0.400	0.145	0.487	0.279	0.091	0.754	0.468	0.748	0.425
		0.500	0.124	0.480	0.258	0.074	0.754	0.465	0.745	0.414
	FiNERWeb	0.050	0.269	0.516	0.311	0.216	0.541	0.509	0.573	0.419
		0.100	0.288	0.534	0.327	0.230	0.579	0.515	0.619	0.442
		0.150	0.305	0.532	0.339	0.239	0.598	0.518	0.644	0.453
		0.200	0.314	0.527	0.341	0.242	0.611	0.520	0.663	0.460
		0.300	0.325	0.511	0.338	0.233	0.627	0.516	0.678	0.461
		0.400	0.315	0.483	0.317	0.209	0.629	0.503	0.671	0.447
		0.500	0.293	0.439	0.282	0.174	0.620	0.481	0.645	0.419

Table 10: Detailed results for `mmBERT-base`.