What Matters When Building Universal Multilingual Named Entity Recognition Models?

Jonas Golde **Patrick Haller** **Alan Akbik**
Humboldt Universität zu Berlin
jonas.max.golde.1@hu-berlin.de

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 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 impedes progress in the field by making it difficult to identify which choices improve multilingual generalization. In this work, we conduct an extensive empirical evaluation on transformer backbones, architectures, training objectives, data composition, and threshold selection for zero-shot, multilingual NER models. Building on these findings, we present OTTER, a small encoder model that achieves consistent improvements over strong multilingual NER baselines, outperforming similarly sized models by 5.7 percentage points in F1. Further, it remains within 1.4 points of Qwen3-32B and 5.4 points below Gemma3-27B, while being roughly 90× smaller and one to two orders of magnitude faster at inference.

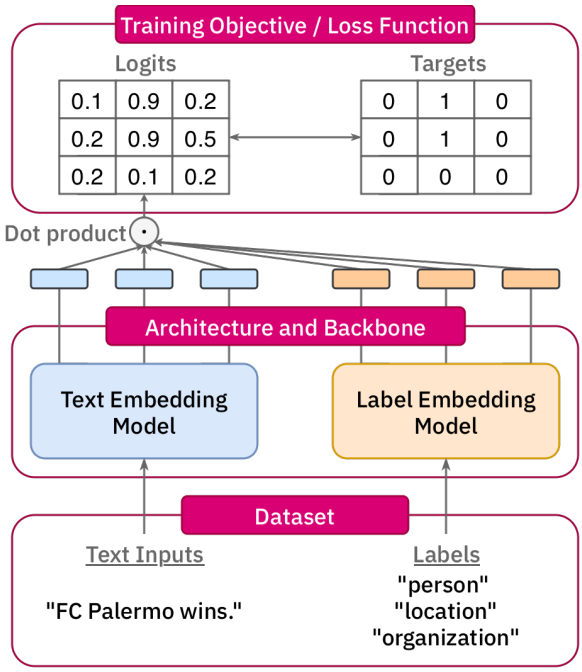

Figure 1: Overview of the design choices in universal multilingual NER systems: the dataset, backbone, architecture, and training objective. Prior work typically fixes a particular combination of these choices and studies only their joint effect. In this work, we vary each dimension in a controlled setting 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 uses such models as effective teachers for training smaller, more efficient encoder-only models, as done in GLiNER (Zaratiana et al., 2024, 2025) or NuNER (Bogdanov et al., 2024).

Design choices are rarely justified. However, despite this progress, many design choices in related work are only weakly justified or briefly discussed. Because such choices vary widely across systems, it is hard to conclude which ones actually contribute to better generalization across languages and domains. 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

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 51628–51647
October 24-29, 2026 ©2026 Association for Computational Linguistics

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 the best of 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 design dimensions and measure their effects on performance and efficiency.

Identifying the design dimensions. We investigate five core dimensions arising from prior work: *(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. In this paper, we empirically evaluate these dimensions within a controlled experimental setting to isolate their individual and joint effects.

Contributions. Building on these insights, we train OTTER, a universal multilingual NER model optimized for the long tail of languages, which we evaluate on more than 150 distinct languages. OTTER achieves strong zero-shot performance across seven benchmarks, consistently outperforming existing models of comparable size while running one to two orders of magnitude faster than comparable autoregressive baselines (Section J). We summarize our contributions as follows:

- We empirically investigate the design choices in universal multilingual NER, isolating the effects of architecture, transformer backbone, loss function, training data and language-specific thresholding within a staged, controlled comparison.
- We generally find that a simple loss function and large-scale multilingual training are sufficient to obtain strong performance. Further, we support our findings with several ablations, such as score calibration, the impact of subword fragmentation, and candidate-span imbalance.
- Based on our findings, we present OTTER, a multilingual NER model supporting over 100 languages, and evaluate it across seven benchmarks, achieving state-of-the-art zero-shot performance on many of them.
- We release the model checkpoints as well as the training and evaluation code to support reproducibility and further research using our model.[3]

2 Methodology

2.1 Dimension 1: Architecture and Backbone

At a high level, there are two main approaches for combining text inputs $\mathbf{x} = \mathrm{x}_1, \ldots, \mathrm{x}_n$ and label inputs $\mathbf{y} = \mathrm{y}_1, \ldots, \mathrm{y}_n$. The cross-encoder approach concatenates text and label descriptions and processes them jointly through a single transformer, allowing text tokens and label tokens to cross-attend to each other (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.

$$\mathbf{h}^{\mathrm{x}}, \mathbf{h}^{\mathrm{y}} = f_{\text{cross}}(\mathbf{x}, \mathbf{y}), \tag{1}$$

$$\mathbf{h}^{\mathrm{x}} = f_{\text{bi}}^{text}(\mathbf{x}), \qquad \mathbf{h}^{\mathrm{y}} = \mathrm{Pool}(f_{\text{bi}}^{labels}(\mathrm{y})). \tag{2}$$

where $\mathrm{Pool}(\cdot)$ maps the token embeddings of a label description to a single vector. For the cross-encoder setup, we follow the idea of GLiNER and introduce an additional `[LABEL]` token to obtain single-vector label representations. The underlying encoders are typically initialized from pretrained transformer models.

Given hidden representations $\mathbf{h}^{\mathrm{x}} \in \mathbb{R}^{|\mathrm{x}| \times d}$ and $\mathbf{h}^{\mathrm{y}} \in \mathbb{R}^{d}$, we project text and label representations using a two-layer MLP to obtain span-start, span-end, and label embeddings (Equations (3) to (5)). We then concatenate the corresponding start and end representations with a span-width embedding to obtain a span representation $\mathbf{k}_{i,j}$ for each span (i, j) (Equation (6)).

$$\mathbf{s} = \mathrm{MLP}_{\text{start}}(\mathbf{h}^{\mathrm{x}}), \tag{3}$$

$$\mathbf{e} = \mathrm{MLP}_{\text{end}}(\mathbf{h}^{\mathrm{x}}), \tag{4}$$

$$\mathbf{q} = \mathrm{MLP}_{\text{label}}(\mathbf{h}^{\mathrm{y}}), \tag{5}$$

$$\mathbf{k}_{i,j} = \mathrm{MLP}_{\text{span}}\big(\mathbf{s}_i \oplus \mathbf{e}_j \oplus \mathbf{d}(j-i)\big). \tag{6}$$

We then use this span representation $k_{i,j}$ to compute label-specific logits using label representations

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

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

[3] https://github.com/whoisjones/otter

$\mathbf{q}$ (Equation (7)).

$$\ell_{i,j} = \mathbf{k}_{i,j}^{\top}\mathbf{q} \tag{7}$$

This formulation extends naturally to n labels by computing $\ell_{i,j,n} = \mathbf{k}_{i,j}^{\top}\mathbf{q}_n$ for every $\mathbf{q}_n \in \mathbf{Q}$. The underlying encoders are usually initialized from pretrained transformer models and trained using in-batch negatives. We note that more advanced negative mining strategies exist that go beyond negatives occurring within the batch, which we do not explore in this work.

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. These datasets typically have large label sets following a long-tail distribution, such as PileNER (Zhou et al., 2024), NuNER (Bogdanov et al., 2024) or LitSet (Golde et al., 2024). When we train a smaller model on these datasets, the resulting supervision relies on noisy label descriptions rather than on a fixed set of classes. Since these label sets contain many synonymous and hierarchically related labels of varying granularity, the model learns to interpret the semantics of the label descriptions themselves, which is what allows the model to generalize to arbitrary, unseen labels at inference time. 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.

Multilingual tokenization. Because tokenization produces spans of different lengths across languages (e.g., shorter ones for English and longer ones for Chinese), the ratio of positive to negative span candidates varies by language, which affects the calibration of the decision threshold (cf. Section D). We deliberately use a single global threshold across all languages rather than tuning one per language, as the latter would require labeled data in each target language.

2.3 Dimension 3: Loss Functions

Given the span-label scores $\ell_{i,j,n}$, the loss function determines how errors are converted into a training signal. In our setting, it must handle the severe class imbalance in span-based NER, where most span–label pairs are negatives. Existing approaches use different loss functions: Binder uses a contrastive loss to align span and label representations in a shared embedding space, whereas GLiNER applies a binary cross-entropy loss. 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 NER 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 chunking keeps the number of negative candidates low and simplifies training, but it has two drawbacks in 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 accurate word-segmentation models 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. We quantify this trade-off, including its negative consequences, in Section D and Section K.

Further, this formulation natively supports nested NER because each candidate span–label pair is scored independently. Overlapping and nested predictions therefore require no special treatment.

3 Experiments

In the following, we evaluate each of the dimensions introduced above. We do not try every combination of the five dimensions, but instead look at one dimension at a time: we keep the best setting from the earlier steps and sweep over the next dimension.

3.1 Experiment 1: Architecture and Backbone

We start by comparing the cross-encoder and bi-encoder architectures. We train both models on

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: Averaged micro-F1 scores on the holdout validation benchmarks for each architecture–backbone pair. Within each row, the best and second-best scores are shown in bold and underlined, respectively.

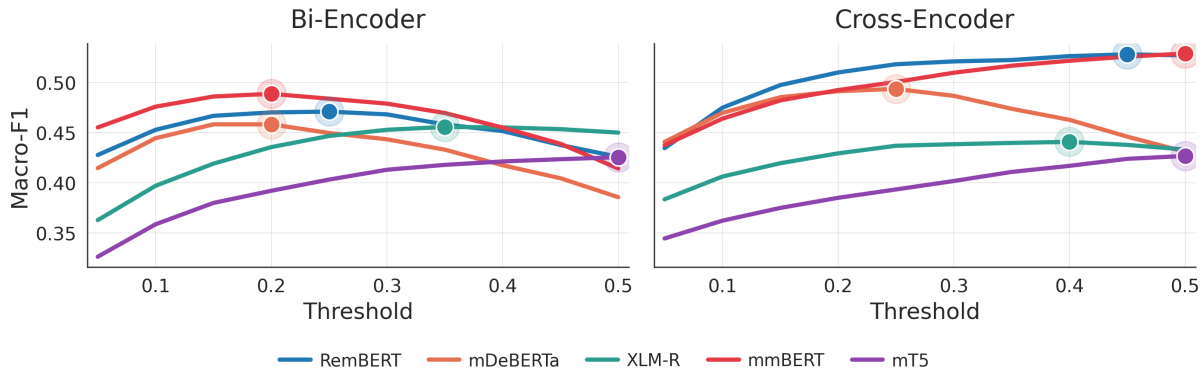

Figure 2: Average micro-F1 scores across the holdout validation benchmarks for different decision thresholds τ. We observe that the optimal threshold depends mostly on the architecture (early vs. late fusion).

the PileNER dataset (English-only, Latin script) using 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 MLPs, 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 the standard binary cross-entropy loss and leave the exploration of different datasets and loss functions to later sections.

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 optimize a single threshold τ on this holdout set and transfer this threshold across all languages in the later experiments.

Results. We report results on the validation set in Table 1, using the best global threshold for each architecture–backbone pair. The cross-encoder generally 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 a low cross-encoder score (0.441 F1).

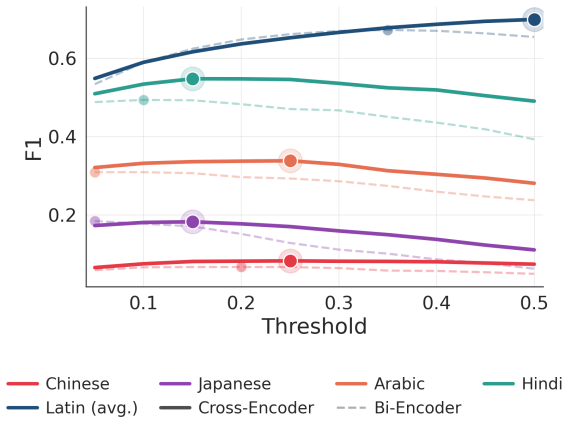

Figure 3: F1 versus decision threshold per language, macro-averaged over backbones for each architecture (cross-encoder: solid; bi-encoder: dashed). Latin (avg.) is averaged over English, German, French, and Spanish; the optimal threshold per curve is marked.

Threshold Selection. We further evaluate model performance across a range of global decision thresholds $\tau \in \{0.05, 0.1, 0.15, \ldots, 0.5\}$, in steps of 0.05, used to convert span–label scores into final predictions. Figure 2 shows the averaged micro-F1 over the eight validation splits for each backbone. We highlight the optimal threshold per curve (architecture–backbone pair). We observe a consistent difference between the two architectures: the bi-encoder peaks at lower thresholds (roughly $\tau \in [0.2, 0.3]$), whereas the cross-encoder peaks at higher thresholds (roughly $\tau \in [0.4, 0.5]$). This follows from how each architecture combines span and label information, which we verify directly by scoring each candidate span–label pair on the holdout splits and comparing the resulting distributions (Section E). The bi-encoder relies on late interaction, scoring spans and labels independently before comparing them: it assigns gold spans a mean probability of only 0.384, and just 40.5% of them exceed 0.5, so its correct matches are best captured at lower thresholds. The cross-encoder fuses the two sources of information early, yielding sharper and better-calibrated scores (0.573 mean probability, with 62.1% above 0.5) while pushing negatives lower, which favors higher thresholds.

Cross-encoders transfer better across scripts. We further examine thresholding across different languages and scripts. We show the threshold sweep from the previous experiment grouped by script in Figure 3. We observe that the two architectures behave almost identically on Latin-script languages (English, German, French, and Spanish), peaking at high thresholds. Non-Latin scripts, in contrast, peak at lower thresholds. We further find that subword fragmentation across scripts is what drives this observation: in Section F, we show that the per-language fragmentation rates correlate with the optimal threshold, with Chinese and Japanese text producing roughly 1.1 subwords per character and peaking at $\tau \in [0.25, 0.30]$, whereas Latin-script text produces 0.26–0.34 subwords per character and peaks at $\tau \in [0.45, 0.60]$.

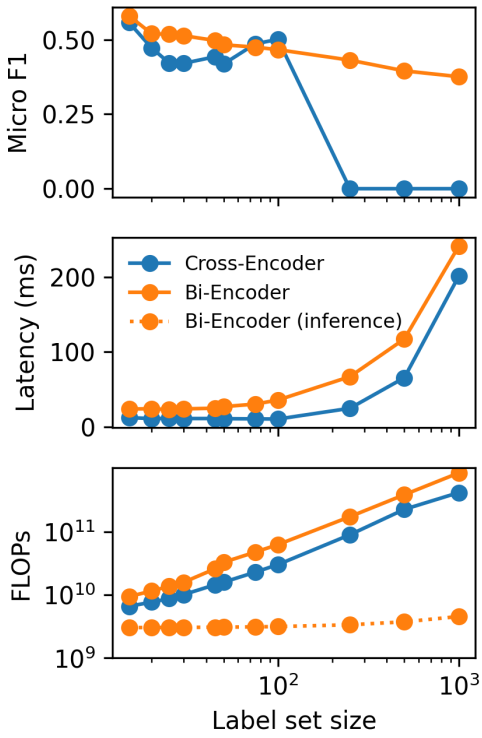

Figure 4: Micro-F1, latency, and FLOPs with increasing number of labels for cross- and bi-encoder (mmBERT). The dotted line shows bi-encoder inference FLOPs with cached label embeddings.

Efficiency Comparison. In this experiment, we compare how performance scales with the number of labels used during inference. Specifically, we use the mmBERT versions of both architectures and evaluate on the four MultiNERD validation datasets, which have 15 distinct labels. We extend this label set 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. In Figure 4, we show micro-F1, inference latency, and FLOPs as the label set grows with noisy and ambiguous labels. We find that micro-F1 decreases for both

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: Averaged micro-F1 on the validation splits for different training datasets. Within each row, the best and second-best scores are shown in **bold** and underlined, respectively.

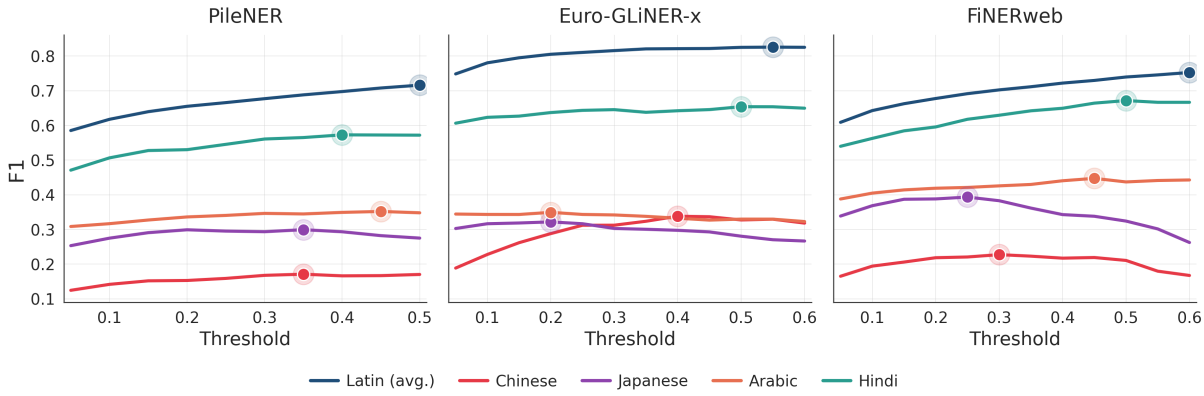

Figure 5: Macro-averaged micro-F1 scores for different decision thresholds τ and different training datasets. We observe that the optimal threshold for each language depends on the languages and scripts covered during training. We use the cross-encoder architecture with mmBERT.

architectures as the label set grows, indicating that larger, more confusable label sets decrease performance. The cross-encoder architecture collapses beyond 250 labels: once not all labels fit in a single forward pass, logits across batches become incomparable. The bi-encoder naturally overcomes this issue, as its scores are computed independently of the batch composition. In terms of cost, the bi-encoder has higher training FLOPs due to its two encoders, 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 latter curve for readability.

3.2 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. These experiments aim to answer to what extent fine-tuning a multilingual backbone on English-only data is sufficient, and whether multilingual training data is required.

Interpreting the comparison. The three corpora differ in language coverage, but also in their underlying generation procedure. Isolating coverage as a single variable is therefore not strictly possible. However, all runs use the same architecture, backbone, objective, optimizer settings, and step budget, so the comparison is at least held fixed on the training side. We further conduct a basic lexical and semantic comparison between the training and validation data in Section H to better interpret the findings of this section. Corpus size and annotation quality likely contribute to the absolute numbers, and we therefore read these results as evidence about corpora as they exist rather than about coverage in isolation. One factor we can quantify directly is how many subword embeddings each corpus actually updates during training. We tokenize each dataset with the XLM-R tokenizer and compute the fraction of subword embeddings that receive gradient updates (Section G). FiNERweb updates over 94% of all subword embeddings, compared to only 38.55% for English-only PileNER,

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

indicating that much of the multilingual vocabulary is never trained under English-only fine-tuning.

Results. We report averaged micro-F1 scores per training dataset in Table 2. The cross-encoder architecture outperforms the bi-encoder across all datasets and backbones, and mmBERT is again the strongest backbone. We find that multilingual training helps overall: under a matched budget, 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 (cross-encoder with mmBERT: 0.529 $\rightarrow$ 0.612). FiNERweb also improves the cross-encoder over PileNER (cross-encoder with mmBERT: 0.529 $\rightarrow$ 0.575) 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 down F1 by language for the cross-encoder with mmBERT. The gains from multilingual training depend on the script. On Latin-script languages, FiNERweb does not beat the other datasets and even falls slightly behind, likely because its broad coverage leaves less room for Latin-script data. On non-Latin scripts, it is clearly ahead: Arabic, Hindi, and Japanese all score higher than under English- or Latin-only training. This split is what we would expect if coverage is the deciding factor, since corpus size and annotation quality should not push Latin and non-Latin scripts in opposite directions. Broad coverage thus costs a little on Latin scripts and gains a lot on the rest.

3.3 Experiment 3: Loss Functions

In this experiment, we compare different loss functions: *(i)* binary cross-entropy (BCE) (with positive upweighting), *(ii)* focal loss with varying α and γ, *(iii)* contrastive loss and *(iv)* dice loss. We provide details on the loss formulations in Section C and use the cross-encoder with mmBERT trained on FiNERweb.

Results. We report results in Table 3. First, we observe that upweighting positives does not help, but instead degrades performance. Dice loss roughly matches BCE (0.577 vs. 0.575), and contrastive loss fails to train competitively in our setting (0.046). Focal loss gives the best holdout result (0.585 with α=0.75, γ=1.0), a marginal improvement over BCE. One hypothesis for this gain is the breadth of label descriptions in FiNERweb: with many overlapping and fine-grained labels, down-weighting easy negatives may help.

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	0.577

Table 3: Averaged micro-F1 on the validation splits for different loss functions (cross-encoder, mmBERT, FiNERweb).

4 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 because the more advanced loss functions yielded only marginal gains. We retain BCE rather than the nominally better focal loss because the two tie across the full benchmark suite (Table 4), so "optimized" here denotes the best performance–simplicity trade-off rather than the literal union of every top-scoring choice. Based on our threshold analysis, we adopt a single global threshold per architecture, $\tau = 0.3$ for the cross-encoder and $\tau = 0.2$ 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 with respect to related work, we compare it against 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 vari-

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

Model	Datasets							Avg.
	Dynamic-NER	Masakha-NER	MultiCoNER v1	MultiCoNER v2	Multi-NERD	PAN-X	UNER	
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	0.300	0.542	0.414	0.249	0.653	0.602	0.660	0.489
w/ mmBERT, focal	0.321	0.537	0.401	0.240	0.649	0.596	0.680	0.489

Table 4: Macro-averaged F1 scores across NER benchmarks. All OTTER rows use a single global threshold; unless marked otherwise they are trained with BCE, and the focal-loss row is matched to it at 30k training steps.

ants (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 to 1,000 examples when it is larger. We report all results using micro-averaged F1 across languages within each dataset, and macro-averaged F1 across datasets when reporting aggregate results.

Annotation conventions and class imbalance. These benchmarks were annotated independently and differ in label definitions and span-boundary conventions, and their positive-class rates vary by more than an order of magnitude across languages (Figure 7). This bounds what any model can score and applies equally to all systems we compare; Section K gives a concrete case in which our predictions are penalized by a boundary convention rather than being modeling errors. A systematic study of annotation noise across these benchmarks is a valuable but separate contribution.

Results. We report all baselines and OTTER in Table 4. All results for OTTER use the single global threshold described above, i.e. the deployable setting that requires no target-language validation data. Our best cross-encoder reaches 0.489 F1, the strongest result among similarly sized task-specific models, beating GLiNER-x-base (0.431) by 5.7 pp. It remains 1.4 pp behind Qwen3-32B (0.503) and 5.4 pp behind Gemma3-27B (0.543), despite being roughly 90× smaller. Training the identical configuration with focal loss instead of BCE gives the same average of 0.489, confirming that the objective is not what drives this result.

The cross-encoder outperforms the bi-encoder at scale (0.489 vs. 0.427 for mmBERT), unlike our earlier holdout experiments where the two were comparable. This is consistent with its early fusion of span and label information capturing language-specific patterns, such as the varying positive-to-negative span ratio across languages: stratifying the per-language F1 gap by the negative-to-positive candidate ratio shows that the cross-encoder advantage is largest where positive spans are dense and smallest where they are sparse (Section I).

Among LLMs, Gemma3-27B is strongest overall (0.543), ahead of Qwen3-32B (0.503), while

GPT-5 is surprisingly weak (0.347). We select Gemma3 deliberately, as it is one of the teachers used to annotate FiNERweb, which makes the comparison a measure of how much teacher performance a 90× smaller student recovers. Absolute F1, however, is only one side of the picture. At 18 ms per example and 1.4 GB of VRAM, OTTER runs 50–80× faster than a 4B-parameter baseline that it outperforms by 15–31 pp, and roughly two orders of magnitude faster than the 27–32B models leading the table (Section J). Our claim is therefore about the trade-off rather than the maximum: at the scale at which universal NER models are deployed, a task-specific model comes within a few points of models that are far more expensive to run.

5 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), who use pre-transformer architectures, train on at most 14 languages, and evaluate on 4. We train on up to 91 languages and evaluate on 250+ test splits with current universal-NER architectures, and find a picture more specific than "multilingual fine-tuning helps": it helps unevenly across scripts (Figure 5), does not help the bi-encoder at all with FiNERweb (Table 2), and backbone rankings can flip with the architecture (Table 1) — none of which follows from first principles.

6 Conclusion

In this work, we systematically explored the design space of prior approaches to universal NER under a staged, controlled protocol. 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. Under a single global threshold, the resulting model stays within a few points of generative models two orders of magnitude larger while running one to two orders of magnitude faster, which we regard as the practically relevant operating point.

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 largest number of languages covered by a single training dataset is 91.

Staged rather than exhaustive search. Fully crossing all five design dimensions would require several hundred training runs, so we proceed in stages: each dimension is varied while the choices preferred in the earlier stages are held fixed. Concretely, the loss-function comparison is run only with the cross-encoder, the mmBERT backbone, and FiNERweb, and the threshold analysis is conditioned on the corresponding checkpoints. Our conclusions are therefore statements about the configurations we tested, and a different objective could in principle interact differently with another backbone or corpus. Two observations mitigate this. First, where we do cross dimensions, the conclusions are stable: the cross-encoder is ahead of the bi-encoder for every backbone–dataset pair in Table 2, and mmBERT is the best backbone under both architectures in Table 1. Second, the one interaction that we do observe, the architecture-dependent ranking of XLM-R, is reported explicitly rather than averaged away. Still, a full factorial study remains a valuable direction for future work.

Confounds in the dataset comparison. The three fine-tuning corpora differ in language coverage but also in size, annotation pipeline, and label granularity. We hold the architecture, backbone, objective, and step budget fixed across them and rule out distributional proximity to the evaluation data as an explanation (Section H), but we cannot attribute the observed differences to multilingual coverage alone. An experiment that varies language coverage while holding the annotation pipeline fixed would be needed to establish that claim conclusively.

Annotation noise and cross-dataset scheme mismatch. NER benchmarks are known to differ in span-boundary conventions and to contain annotation noise, which affects cross-dataset zero-shot evaluation for all models we compare. Section K documents one concrete instance of this on Shona, where linguistically plausible predictions are penalized by the annotation convention rather than being modeling errors. We do not attempt to quantify how much of the remaining gap across the 250 test splits is attributable to such mismatches; doing so would require a systematic re-annotation study.

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". Since we train the model to treat such labels as distinct concepts, this may contribute to the observed performance differences when training with labels in the target language.

Threshold Selection. We find that threshold selection for zero-shot NER is highly language-dependent, and therefore select the threshold on a holdout validation set. As a consequence, the performance gap to in-domain fine-tuned models can remain substantial, since 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.

Acknowledgments

We thank all reviewers for their valuable comments. Jonas Golde is supported by the Bundesministerium für Bildung und Forschung (BMBF) as part of the project "FewTuRe" (project number 01IS24020). Alan Akbik and Patrick Haller are supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Emmy Noether grant "Eidetic Representations of Natural Language" (project number 448414230). Further, Alan Akbik is supported under Germany's Excellence Strategy "Science of Intelligence" (EXC 2002/1, project number 390523135).

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. COP-NER: 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.

A Implementation and Compute

For all our experiments, we use 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.

LLM prompting and decoding. For all generative baselines, we use each model's default generation hyperparameters as shipped with its chat template, and re-use the extraction prompt and substring-matching parser introduced by Golde et al. (2025). We prompt the model once per input sentence with the full label set of the respective benchmark and ask it to return the surface forms of all entities per label. The parser then maps each returned surface form back to character offsets in the input by exact substring matching, discarding generations that do not occur verbatim in the input. We use no few-shot examples, self-consistency, or chain-of-thought decoding, so the comparison reflects single-pass prompted extraction. We show the verbatim prompt template in Figure 6.

```
<instruction>
Given the following passage in {language}, extract all
named entities and assign each a type. Identify a
diverse and comprehensive set of entities, including
but not limited to:
- Persons, organizations, and locations;
- Technologies, programming languages, and frameworks;
- Scientific concepts, theories, and discoveries;
- Cultural references, works of art, and media;
- Products, brands, and inventions;
- Events, time periods, and dates;
- Quantities, measurements, and statistics;
- Any other meaningful or contextually significant
  named entities.
For each entity, determine the most appropriate and
specific type label. Go beyond general categories when
possible (e.g., use Quantum Theory instead of just
Scientific Concept).
</instruction>
<example>
The output should be a list of tuples in the following
format:
[("entity 1", "type of entity 1"), ...]
</example>
<input>
Passage: {input_passage}
</input>
Annotations:
```

Figure 6: Verbatim extraction prompt used for all generative baselines, following Golde et al. (2025). The placeholders `{language}` and `{input_passage}` are filled with the language and the input sentence of the respective benchmark instance.

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 the number of entity types.

B Evaluation Benchmarks

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

C 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 for $\gamma = 0$ 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 of 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 $\mathrm{sim}(\cdot,\cdot) = \cos(\cdot,\cdot)/\tau$ is a temperature-scaled cosine similarity. This yields the span-based typing objective:

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

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:

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

The two objectives are combined into a single contrastive loss,

$$\ell_{\text{Con}} = \alpha\, \ell_{\text{span}} + \beta\, \ell_{\text{thr}}, \tag{13}$$

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

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

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

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

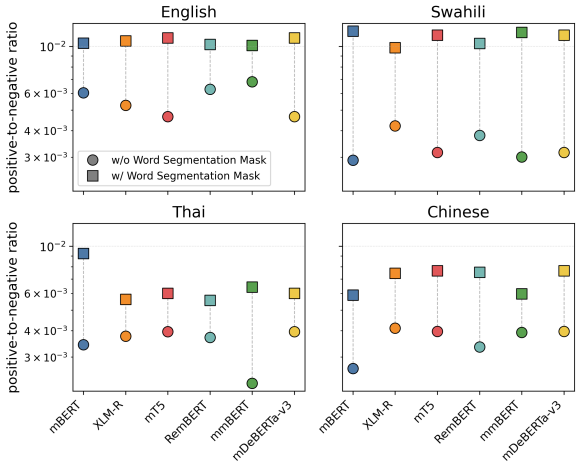

Figure 7: Positive-to-negative span ratios per language and tokenizer, with and without a word-segmentation mask. Segmentation raises the ratio in every case, so training on raw text means training under a harsher class imbalance.

D 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 8). 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 quantify what this costs us, 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 based on 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. We observe that word segmentation raises the ratio for every language and tokenizer, i.e. training on raw text means training under a harsher class imbalance. This is the trade-off we accept in exchange for not maintaining a segmentation model per language, and Section K shows a concrete case in which it hurts.

E Score Distributions and Calibration by Architecture

In Section 3.1, we observe that the bi-encoder peaks at lower decision thresholds than the cross-encoder, and we attribute this to late versus early fusion of span and label information. Here, we verify this directly. Using the trained bi- and cross-encoder checkpoints (mmBERT, PileNER), we score every candidate span–label pair on the eight holdout validation splits (roughly 3.6M candidate pairs per split), label each pair as positive or negative against the gold entities, and compare the resulting score distributions.

Metric	Bi-Enc.	Cross-Enc.
Mean prob. on gold spans	0.384	**0.573**
Gold spans with $p > 0.5$	40.5%	**62.1%**
Mean prob. on negative spans	7.2e−4	2.4e−4
Expected calibration error	5.3e−4	1.5e−4
Score entropy	2.5e−3	8.7e−4

Table 6: Score distribution and calibration statistics aggregated over the eight holdout splits. Lower is better for negative-span probability, calibration error, and entropy.

We report the resulting statistics in Table 6. The cross-encoder assigns gold-positive spans a mean probability of 0.573 against 0.384 for the bi-encoder, and 62.1% of its true positives exceed 0.5 compared to only 40.5%. It is also better calibrated (expected calibration error 1.5e−4 vs. 5.3e−4), sharper (mean binary entropy 8.7e−4 vs. 2.5e−3), and pushes negatives further down (mean probability 2.4e−4 vs. 7.2e−4). This explains the threshold behavior observed in Figure 2: the correct matches of the bi-encoder cluster below 0.5, so its optimum sits at a lower threshold, whereas those of the cross-encoder cluster near 1.0 and thus favor a higher one.

F Subword Fragmentation and Optimal Thresholds

In Section 3.1, we further observe that non-Latin scripts peak at lower thresholds, and we attribute this to subword fragmentation. To verify this, we correlate the per-language subword fragmentation rate, measured as the number of subword tokens produced per character of raw text, with the per-language optimal threshold from Figure 3.

Lang.	*Script*	*Subwords / char*	*Optimal threshold*
eng	Latin	0.263	0.60
deu	Latin	0.283	0.60
spa	Latin	0.289	0.55
fra	Latin	0.338	0.45
ara	Arabic	0.442	0.45
hin	Devanagari	0.548	0.50
jpn	CJK	1.118	0.25
zho	CJK	1.175	0.30

Table 7: Subword fragmentation rate and optimal decision threshold per holdout language, sorted by fragmentation.

We report both quantities in Table 7 and observe a monotone trend. Chinese and Japanese text is fragmented most heavily (about 1.1 subwords per character) and peaks at the lowest thresholds (0.25–0.30), while Latin-script text is fragmented least (0.26–0.34) and peaks highest (0.45–0.60). Arabic and Devanagari fall in between on both axes. This is the mechanism we appeal to in Section 3.1: more subword fragments per unit of text produce more candidate spans per entity, which spreads probability mass and thus lowers the optimal threshold.

G Gradient Updates in Multilingual Models

We show in Table 8 the percentage of subword embeddings that receive gradient updates for each of the three fine-tuning datasets.

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

Table 8: Fraction of subword embeddings updated during training.

H Train-Eval Distributional Similarity

To verify that the improvements from multilingual training cannot be explained by distributional matching between the training and the 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 Fréchet distance between the sentence-embedding distributions, capturing semantic similarity. We compute sentence embeddings 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 (Fréchet distance)			
FiNERweb	2.339	1.844	2.413
Euro-GLiNER-x	0.959	0.379	–
PileNER	3.545	–	–

Table 9: 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.

We report the results in Table 9 and find 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. We therefore conclude 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.

I Candidate-Span Imbalance and the Cross-Encoder Advantage

In Section 4, we attribute the advantage of the cross-encoder at scale to its ability to capture the varying positive-to-negative span ratio across languages. To verify this, we compute the per-language negative-to-positive candidate-span ratio on the holdout splits and correlate it with the per-language F1 gap between the cross- and the bi-encoder.

Lang.	*Script*	*neg:pos*	ΔF1
fra	Latin	56	+0.159
ara	Arabic	88	+0.162
hin	Devanagari	112	+0.217
eng	Latin	264	+0.089
deu	Latin	276	+0.121
spa	Latin	353	+0.042
jpn	CJK	473	+0.074
zho	CJK	884	+0.023

Table 10: Negative-to-positive candidate-span ratio and cross-encoder minus bi-encoder F1 per holdout language, sorted by ratio.

We report both quantities in Table 10 and observe that the cross-encoder advantage is largest where positive spans are dense (low negative-to-positive ratio) and smallest where they are sparse, which is consistent with the explanation given in Section 4. We note one caveat: the low-ratio languages here (French, Arabic, and Hindi) also tend to have shorter texts in PAN-X, so the ratio is partly correlated with sequence length. We therefore report this as supporting rather than as conclusive evidence.

J Smaller and Quantized LLM Baselines

Smaller and quantized LLMs are an important point of comparison for efficiency, and we already use FP8-quantized versions of Qwen and Gemma in

our main experiments. 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 as a single generation step for the autoregressive models, using batch size 1 for a fair comparison. We take all measurements 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 11: 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.

We report the results in Table 11. 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 we ran these through inference providers. Our local FP8 measurements place their per-example latency on the order of several seconds, although 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.

K 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", *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*) and thus results in boundary mismatches with the gold annotations (Figure 8). While this behavior does not conform to the annotation guidelines, we consider it linguistically plausible.

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 8: 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).

L Detailed Results

We show detailed results for selected experiments in Tables 12 and 13.

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 12: Detailed results for `rembert-base`.

ARC.	DATA-SET	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 13: Detailed results for `mmBERT-base`.