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 advances in multilingual transformer models and task-specific architectures, loss functions, and training datasets. Despite substantial prior work, we find that many critical design decisions for such models are made without systematic justification, with architectural components, training objectives, and data sources evaluated only in combination rather than in isolation. We argue that these decisions impede progress in the field by making it difficult to identify which choices improve model performance. In this work, we conduct extensive experiments around architectures, transformer backbones, training objectives, and data composition across a wide range of languages. Based on these insights, we introduce OTTER, a universal multilingual NER model supporting over 100 languages. OTTER achieves consistent improvements over strong multilingual NER baselines, outperforming GLiNER-x-base by 5.3pp in F1 and achieves competitive performance compared to large generative models such as Qwen3-32B, while being substantially more efficient. We release model checkpoints, training and evaluation code to facilitate reproducibility and future research.

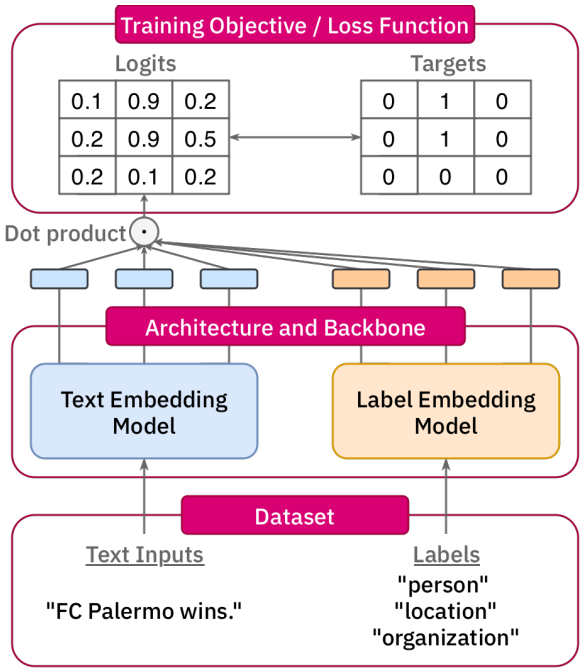

Figure 1: Overview of the design choices in universal multilingual NER systems. Prior work typically combines a choice of dataset, architecture, backbone, and training objective, only investigating their joint effects. In this work, we systematically combine these dimensions in a controlled setting to study their impact on performance and efficiency.

1 Introduction

Multilingual named entity recognition (NER) is a widely used information extraction task that identifies named entities such as "person" or "location" in text (Lample et al., 2016; Akbik et al., 2018) and is used in applications such as knowledge graph construction (Zhong et al., 2023; Arsenyan et al., 2024), automatic invoice extraction (Perot et al., 2024), and large-scale document indexing and search (Shachar et al., 2025). The development of increasingly capable large language models (LLMs) (Grattafiori et al., 2024; Yang et al., 2025; DeepSeek-AI et al., 2025; Team et al., 2025) also achieves improved performance on the task of multilingual NER but, more importantly, we can utilize such models as effective teachers to train smaller and more effective encoder-only models as done in UniNER (Zhou et al., 2024), GLiNER (Zaratiana et al., 2024, 2025), NuNER (Bogdanov et al., 2024) or LitSet (Golde et al., 2024). However, despite progress in this area, we find many design choices in the literature are only weakly justified experimentally or briefly discussed.

We argue that performance improvements arising from such heterogeneous experimental setups are difficult to interpret, making it harder for the community to achieve meaningful and grounded progress. For instance, Zaratiana et al. (2024); Huang et al. (2022) employ a cross-encoder architecture whereas Zhang et al. (2023); Bogdanov et al. (2024); Golde et al. (2024) use a bi-encoder architecture. Similarly, `gliner-multi-v2.1` [1] uses mDeBERTa (He et al., 2023) as the backbone transformer model (Vaswani et al., 2023) whereas `gliner-x-base` [2] employs mT5 (Xue et al., 2021). To the best of our knowledge, these choices have never been systematically ablated, and trade-offs in terms of compute, data efficiency and performance remain poorly understood. Thus, in this work, we examine a set of such core design dimensions and quantify their effects on performance and efficiency.

Specifically, we identify four core dimensions along which prior work differs: *(i)* the choice between cross-encoder and bi-encoder architectures, *(ii)* the selection of the transformer backbone, *(iii)* the training objective or loss function, and *(iv)* the composition of the training dataset. We also show these dimensions in Figure 1. We systematically combine and evaluate these design choices within a controlled experimental setting to isolate their individual and joint effects. We conduct our analysis in three stages: first, we assess the impact of architectural choices and backbone models; second, we train on different training datasets covering a different number of languages; third, we ablate over a range of loss functions. Notably, we find that the transformer-choice is architecture-dependent as well as that a training dataset covering more languages is key to good performance. We further find that a simply using a binary cross-entropy loss performs best in our evaluation setting.

Building on these insights, we train an optimized universal multilingual entity recognition model (OTTER). We achieve strong performance across seven multilingual NER benchmarks and consistently outperform existing multilingual NER models of comparable size and remains competitive with substantially larger generative models. Further, OTTER remains competitive in efficiency at both training and inference time. We release model checkpoints and code to support reproducibility and further research in multilingual NER.

We summarize our contributions as follows:

- We empirically investigate and identify critical design choices in universal multilingual named entity recognition, examining the effects of architectures, transformer backbones, loss functions, and training data.
- Based on these findings, we derive OTTER, a multilingual NER model that supports over 100 languages, and zero-shot evaluate it across multiple multilingual benchmarks, achieving state-of-the-art performance in 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]

2 Exploring The Design Choices

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

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

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

[3] Inserted after review.

span-width embedding, yielding a span representation $\mathbf{k}_{i,j}$ for each span (i, j) (Equation (6)).

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

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

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

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

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

Dimension 2: Fine-Tuning Datasets. In parallel, several large-scale NER datasets have been released by using large language models to annotate unlabeled data, which is subsequently used to train smaller models. These datasets typically cover large label sets following a long-tail distribution, which can serve as a realistic supervision signal to generalize to arbitrary label descriptions rather than a fixed ontology such as PileNER (Lou et al., 2023) or NuNER (Bogdanov et al., 2024). In our experiments, we use datasets that vary substantially in language coverage, ranging from English-only to 91 languages. With this comparison, we investigate whether a multilingual transformer trained only on English data is sufficient for cross-lingual generalization, or whether fine-tuning on data covering multiple languages yields additional gains.

Dimension 3: Loss Functions. The training objective reflects how the model combines text and label representations and must handle a severe class imbalance in span-based NER, where most span–label pairs are negatives. To do so, 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 the use of various loss functions from classical binary cross-entropy loss to focal loss (Lin et al., 2018) with varying values of α and γ.

Modeling Without Word Segmentation. Named entity recognition traditionally uses BIO tagging schemes (Ratinov and Roth, 2009), where labels are assigned to word-segmented sequences. While this representation reduces the number of negative span candidates and simplifies the training process, it constrains supervision to word-level boundaries and thus only updates representations corresponding to these boundaries. Further, as we are interested in training a model for many languages and scripts, this would require accurate word segmentation models for each language.

To address this limitation, we train our models directly on the input texts and shift the burden of word segmentation to the model itself by computing candidate spans over subword tokens. While training on subword spans increases the number of negative span candidates compared to word-segmented sequences, it removes the need for external, language-specific preprocessing at inference time and ensures consistent behavior across scripts. We also refer to Section D to illustrate this trade-off.

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

2.1 Experimental Setup

Evaluation Benchmarks. We use seven multilingual, human-annotated dataset 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 1. 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.

Hyperparameters. We train all models with

Architecture	Backbone					Avg.
	mDeBERTa	mmBERT	mT5	RemBERT	XLM-R	
Bi-Encoder	0.330	<u>0.373</u>	0.293	**0.379**	0.349	0.345
Cross-Encoder	<u>0.363</u>	0.323	0.357	0.206	**0.370**	0.324
Avg.	0.347	0.348	0.325	0.292	0.359	–

Table 2: Macro-averaged performance across all evaluation benchmarks for different architectures and transformer backbones. We highlight best and second-best scores per architecture in bold and underlined, respectively.

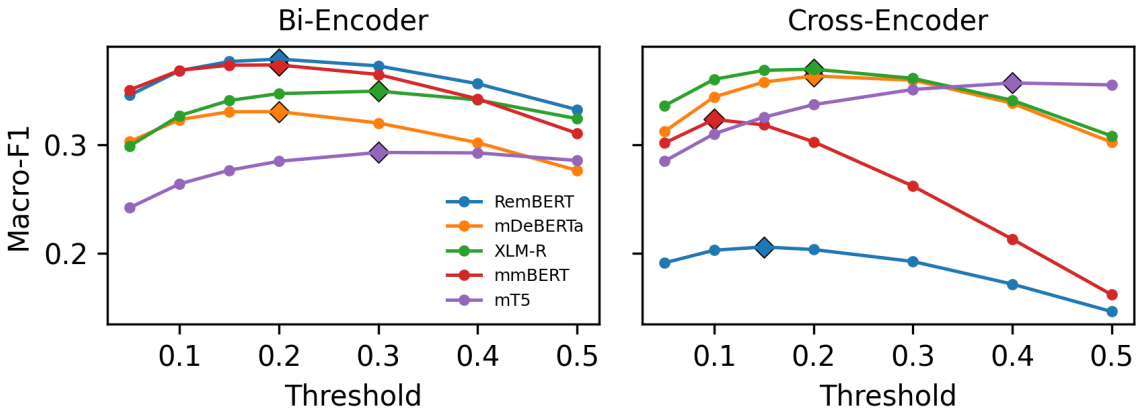

Figure 2: Macro-averaged micro-F1 scores across all backbones evaluation benchmarks with different decision thresholds t. We observe that optimal performance is dependent on the transformer backbone.

a batch size of 12 using 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 output dimension of all MLP projections to $d_{\text{MLP}} = 384$ and 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.

3 Results

3.1 Dimension 1: Architecture and Model Backbones

In this experiment, we train cross- and bi-encoder models on the PileNER dataset for 30k steps 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) (Xue et al., 2021), *(iv)* `mdeberta-v3-base` (He et al., 2023) and *(v)* `rembert` (Chung et al., 2021). We apply early stopping with a patience of 3 based on performance on a held-out validation split of 500 samples. Further, we use standard binary cross-entropy loss and leave the exploration of different datasets and loss functions to later sections.

Results. We report results for all evaluation benchmarks in Table 2. Overall, we observe that the best-performing configurations across architectures achieve nearly identical performance, with the bi-encoder performing best with Rembert (0.379 F1) and the cross-encoder with XLM-R (0.370 F1), indicating that optimal backbone choice is architecture-dependent. Using a suboptimal backbone may lead to noticeable performance drops. For example, we observe 0.330 F1 for the bi-encoder with mDeBERTa, while the cross-encoder achieves 0.323 F1 with mmBERT. We further observe that RemBERT behaves very differently depending on the architecture: it performs competitively in the bi-encoder setting but performance degrades substantially in the cross-encoder setup to 0.206 F1.

We further note that early stopping is triggered across all configurations, and none of the models are trained for the full 30k steps. While we use early stopping based on in-domain validation performance, this criterion may not align with the test distribution. We therefore disable early stopping in later experiments and train models for more steps.

Threshold Selection. We evaluate model performance across a range of decision thresholds $t \in \{0.05, 0.1, 0.15, 0.2, 0.3, 0.4, 0.5\}$ used to convert span–label scores into final predictions. We show results across all evaluation benchmarks in Figure 2. For the bi-encoder, we observe that the best performance is achieved with thresholds in the range $t \in [0.2, 0.3]$. For the cross-encoder, the optimal threshold varies across transformer backbones, with mmBERT performing best at $\tau = 0.1$ and mT5 at $\tau = 0.4$. This indicates that threshold behavior is more stable for the bi-encoder architecture, whereas cross-encoders require backbone-specific threshold tuning.

Performance Comparison. We now use the best-performing configuration for each architecture, we compare their computational requirements during training and inference as the number of labels increases. For this experiment, we select 1,000 examples from the MultiNERD benchmark and extend the label set up to 1,000 labels by adding the most frequent labels from PileNER. For each configuration, we measure micro F1, inference latency, and FLOPs and show the results in Figure 3.

Overall, we observe a consistent decrease in micro-F1 as the label set size increases for both architectures, suggesting that larger and potentially more similar label sets negatively affect inference performance. Further, the cross-encoder performance collapses when the label set exceeds 250 labels as it is no longer possible to include all labels in a single forward pass. Since the cross-encoder learns to discriminate among in-batch labels, logits computed across different batches are not directly comparable.

In terms of computational cost, the bi-encoder shows higher training FLOPs. During inference, however, the computational cost becomes approximately linear, since label embeddings can be cached and reused across inputs which substantially reduces the effective computation required by the bi-encoder. We observe only marginal differences in inference latency between the bi-encoder training and inference and thus omit it from the plot for readability.

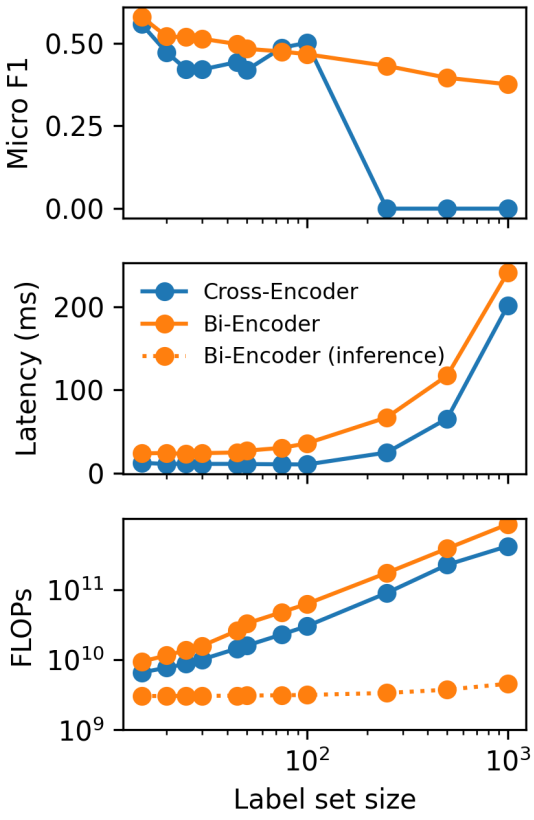

Figure 3: Micro F1, latency, and FLOPs as the number of labels increases.

3.2 Dimension 2: Dataset

We now analyze the effect of the fine-tuning dataset. We consider three datasets for fine-tuning: *(i)* PileNER (Zhou et al., 2024), which contains English-only data; *(ii)* Euro-GLiNER-x, which covers 12 Indo-European languages in Latin script; *(iii)* FiNERweb (Golde et al., 2025), which spans 91 languages and 25 scripts.

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

Table 3: Fraction of subword embeddings updated during training.

Since all datasets follow a long-tail distribution of entity types, we aim to understand to what extent multilingual training data is required, or whether fine-tuning a multilingual backbone on English-only data is sufficient. To illustrate the idea, we tok-

Dataset	Bi-Encoder			Cross-Encoder			Avg.
	XLM-R	RemBERT	mmBERT	XLM-R	RemBERT	mmBERT	
PileNER	0.349	0.379	0.373	0.370	0.206	0.323	0.333
Euro-GLiNER-x	0.375	0.412	0.410	0.383	0.361	0.438	0.397
FiNERweb	**0.394**	**0.432**	**0.437**	**0.420**	**0.454**	**0.461**	**0.433**

Table 4: Macro-averaged performance across datasets for bi-encoder and cross-encoder architectures with different transformer backbones training on PileNER, Euro-GLiNER-x and FiNERweb.

enized each dataset using the XLM-R tokenizer and compute the fraction of subword embeddings that would receive gradient updates during training. Table 3 shows the results where we observe that training on FiNERweb updates more than 94% of subword embeddings, while training on the English-only PileNER updates only 38.55%. We re-use the experimental setup from previous section.

Results. We report results in Table 4 and further refer to the full experimental results in Appendix B. We first observe a consistent trend that increasing the language diversity of the training data improves performance across both architectures and transformer backbones. Training on FiNERweb yields performance improvements in all settings, e.g., +4.5pp F1 over PileNER for the bi-encoder and +5.0pp F1 for the cross-encoder when using XLM-R. The results obtained with Euro-GLiNER-x further support this observation, as training on 12 languages already leads to consistent improvements over English-only training. Overall, we find that multilingual training consistently improves downstream performance, with FiNERweb providing gains of up to 10.0 F1.

Interestingly, we find that RemBERT and mmBERT perform best for both architectures. In particular, the cross-encoder with RemBERT shows an improvement of +24.8 F1 compared to training on English-only data, with similar trends observed for mmBERT.

3.3 Dimension 3: Loss Functions

Using the best-performing configuration from previous experiments (mmBERT backbone trained on FiNERweb), we now evaluate different loss functions: *(i)* binary cross-entropy and upweighting positives, *(ii)* focal loss with varying α and γ, and *(iii)* a contrastive loss with adaptive thresholding. We provide full definitions in Appendix C.

Results. We present results for training both architectures, on mmBERT and FiNERweb, using different loss functions and configurations in Table 5.

Loss	Architecture	
	Bi-Enc.	Cross-Enc.
Baseline (BCE)	0.437	0.461
BCE + Pos. Weight λ		
$\lambda = 10.0$	0.413	0.363
$\lambda = 100.0$	0.297	0.268
Focal Loss		
$\alpha = 0.25, \gamma = 0.0$	0.407	0.437
$\alpha = 0.50, \gamma = 0.0$	0.401	0.438
$\alpha = 0.75, \gamma = 0.0$	0.415	0.356
$\alpha = 0.50, \gamma = 1.0$	0.422	0.314
$\alpha = 0.75, \gamma = 2.0$	0.401	0.435
Contrastive Loss		
$\alpha = 0.30, \beta = 0.3$	0.395	0.430
$\alpha = 0.55, \beta = 0.5$	0.417	0.435
$\alpha = 0.70, \beta = 0.7$	0.409	0.392

Table 5: Performance across evaluation datasets for different loss functions.

First, we observe that simply upweighting positive labels does not improve performance over the BCE baseline. Instead, we observe that it mainly shifts the prediction threshold up. While focal and contrastive losses achieve comparable performance, BCE consistently performs better in both cases (+1.5pp F1 for the bi-encoder and +2.3pp F1 for the cross-encoder). We further observe two outliers when using focal loss for the cross-encoder ($\alpha = 0.75, \gamma = 0.0$ and $\alpha = 0.50, \gamma = 1.0$), which we attribute to early stopping after 3000 steps. This suggests that focal loss may converge more slowly and require longer training than BCE. For contrastive loss, we find that a balanced weighting between typing and thresholding objectives performs best for both architectures. Even though we can avoid using a fixed threshold, contrastive loss remains worse compared to simple BCE in our setting.

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	0.237	0.553	0.343	0.175	0.569	0.509	0.670	0.437
w/ mmBERT-100k	0.281	0.481	0.335	0.238	0.544	0.505	0.592	0.425
Otter (Cross-Enc.)								
w/ RemBERT	0.354	0.573	0.347	0.294	0.636	0.436	0.540	0.454
w/ mmBERT	0.325	0.511	0.338	0.233	0.627	0.516	0.678	<u>0.461</u>
w/ mmBERT-100k	0.382	0.511	0.358	0.254	0.638	0.535	0.713	**0.484**
w/ mmBERT-100k*	0.385	0.523	0.369	0.265	0.661	0.549	0.754	0.501

Table 6: Macro-averaged F1 scores across NER benchmarks. We highlight the best average performance in bold. *Using the best performing threshold per language before aggregating.

4 Combining The Insights

We finally combine these findings and scale up training to obtain Otter, which is trained on FiNERweb for 100k steps without early stopping, using the mmBERT backbone and binary cross-entropy loss. As baselines, we include multilingual LLMs (GPT-5[4], Qwen3-32B (Yang et al., 2025), and Gemma3-27B (Team et al., 2025)), WikiNeural (Tedeschi et al., 2021), and two multilingual variants of GLiNER (Zaratiana et al., 2024).

Results. We report results for all baselines and Otter in Table 6. We first observe that the initially trained bi-encoder and cross-encoder models from previous experiments achieve 0.437 and 0.461 F1, respectively, outperforming the similarly sized GLiNER-x-base baseline by up to 3.0 pp F1 on average. We further find that extending training to 100k steps without early stopping yields additional improvements in the cross-encoder setting of +2.3pp F1, whereas we do not observe similar gains for bi-encoders. Further, we report results for Otter using the best-performing threshold per language per dataset rather than a fixed one, showing an additional +1.5pp F1 in the cross-encoder setting and closing the gap to 0.3pp F1 compared to Qwen3-32B, despite being approximately 90× smaller. This observation confirms the findings in Section 3.1 that universal multilingual NER requires careful per-language threshold selection, as underlying tokenization can vary substantially across languages.

While our initial results indicated the bi-encoders perform similarly to cross-encoder, we now observe cross-encoders clearly outperforming bi-encoders. We attribute this difference to architectural properties, as cross-encoders may better capture language-specific patterns, such as language-dependent ratios between positive and negative spans. We also observe that cross-encoders generally outperform their bi-encoder counterparts, although the performance gap remains comparatively small.

Among large language model baselines, GPT-5 shows weak performance across datasets, whereas Gemma3-27B achieves the strongest overall results, outperforming all other evaluated models. Qwen3-32B also performs competitively, but remains below Gemma3-27B on average. Overall, these re-

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

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 4: 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).

sults indicate that while large language models can achieve strong multilingual NER performance, carefully trained task-specific models are able to close much of the performance gap with substantially lower computational cost.

Negative Findings. 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*), resulting in boundary mismatches with the gold annotations (Figure 4). While this behavior does not conform to the annotation guidelines, we consider it linguistically plausible.

5 Related Work

Named entity recognition using machine learning approaches has been widely studied, traditionally relying on a fixed output layer to compute a probability distribution over a predefined label set (Huang et al., 2015; Lample et al., 2016; Akbik et al., 2018).

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; Golde et al., 2023). 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.

6 Conclusion

In this work, we systematically explored the design space of prior approaches to universal NER by comparing architectures, transformer backbones, training datasets, and loss functions within a unified experimental setup. 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 generally outperform bi-encoders 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 that performs best on average across languages. Nevertheless, 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, Patrick Haller, Felix Hamborg, Julian Risch, and Alan Akbik. 2023. Fabricator: An open source toolkit for generating labeled training data with teacher LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 1–11, Singapore. Association for Computational Linguistics.

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.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *Preprint*, arXiv:1508.01991.

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.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need. *Preprint*, arXiv:1706.03762.

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.

B Detailed Results

We show detailed results for all combinations of our experiments in Tables 8 to 12. We further show all detailed results for our loss function experiments in Tables 13 to 16.

C Loss Functions

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

Architecture	Training Labels	Evaluation Labels	
		English	Translated
Bi-Encoder	English	**0.432**	0.323
	Target-language	0.278	0.306
Cross-Encoder	English	**0.410**	0.337
	Target-language	0.267	0.302

Table 7: Macro-averaged performance over all languages under different combinations of translated training and evaluation data for bi-encoder and cross-encoder architectures. Best results per architecture are highlighted in bold.

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.

Contrastive loss. For contrastive training, 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. Let $\mathrm{sim}(\cdot,\cdot)$ denote a similarity function. We define then the contrastive objective as:

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

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 4). 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 5. 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.

E Training On Translated Labels

We now analyze the effect of using translated label descriptions during training and evaluation. To this end, we train a bi-encoder and cross-encoder on the FiNERweb variant in which label descriptions are provided in the respective target languages. For evaluation, we additionally translate the label sets of all benchmarks into their target languages using the Google Translate API[5]. For languages not supported, we retain the original English label descriptions.

We show results in Table 7. We observe the best performance when if models are trained on English label descriptions and evaluation is also done in English, achieving 0.432 F1 for the bi-encoder and 0.410 F1 for the cross-encoder. However, when

[5] https://cloud.google.com/translate/docs

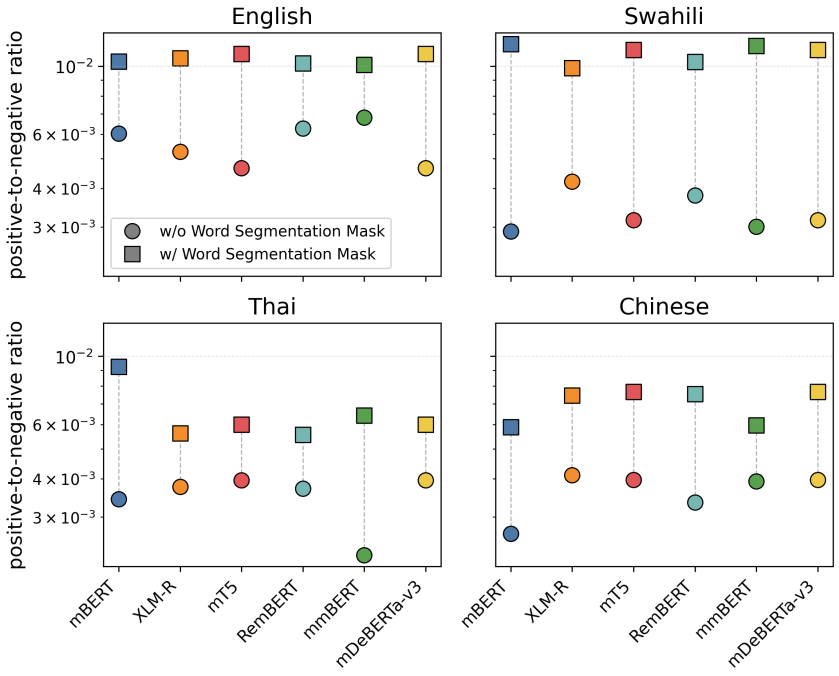

Figure 5: The impact of using pre-tokenized text for training. We can use word boundaries to exclude spans from the loss resulting the higher positive-to-negative ratios.

evaluating these models on translated label descriptions, we observe substantial performance drops for both architectures (up to 0.154 F1 for the bi-Encoder and up to 0.143 F1 for the cross-encoder). Further, we find that training on translated label descriptions improves robustness to translated evaluation, increasing performance to 0.306 F1 for the bi-encoder and 0.302 F1 for the cross-encoder.

Overall, these results show the trade-off between maximizing performance on English label descriptions and improving robustness under translated evaluation settings. This observation is consistent with prior findings by Golde et al. (2025). Addressing this trade-off likely requires more translation-aware training objectives, which we leave for future work. In the rest of this paper, we therefore focus on experiments using English label descriptions.

Arc.	Dataset	τ	Dynamic-NER	Masakha-NER	MultiCoNER v1	MultiCoNER v2	Multi-NERD	PAN-X	UNER	Avg.
Bi-Encoder	PileNER	0.050	0.083	0.384	0.252	0.090	0.420	0.409	0.451	0.299
		0.100	0.091	0.406	0.274	0.092	0.489	0.422	0.511	0.327
		0.150	0.094	0.416	0.276	0.092	0.533	0.426	0.546	0.341
		0.200	0.092	0.420	0.265	0.088	0.561	0.425	0.577	0.347
		0.300	0.088	0.418	0.237	0.081	0.591	0.421	0.609	0.349
		0.400	0.074	0.409	0.203	0.071	0.600	0.412	0.620	0.341
		0.500	0.049	0.393	0.173	0.055	0.592	0.389	0.617	0.324
	Euro-GLiNER-x	0.050	0.042	0.464	0.243	0.074	0.617	0.471	0.682	0.371
		0.100	0.043	0.465	0.243	0.074	0.639	0.469	0.695	0.375
		0.150	0.040	0.459	0.235	0.071	0.654	0.466	0.693	0.374
		0.200	0.037	0.450	0.223	0.068	0.667	0.458	0.687	0.370
		0.300	0.026	0.431	0.199	0.060	0.683	0.440	0.657	0.356
		0.400	0.019	0.406	0.174	0.052	0.680	0.416	0.629	0.340
		0.500	0.015	0.373	0.147	0.042	0.670	0.388	0.596	0.319
	FiNERWeb	0.050	0.174	0.494	0.296	0.138	0.494	0.483	0.563	0.377
		0.100	0.189	0.510	0.304	0.148	0.524	0.484	0.599	0.394
		0.150	0.198	0.495	0.305	0.155	0.519	0.468	0.591	0.390
		0.200	0.206	0.459	0.300	0.159	0.507	0.451	0.558	0.377
		0.300	0.213	0.389	0.281	0.160	0.480	0.407	0.493	0.346
		0.400	0.214	0.326	0.246	0.149	0.447	0.365	0.425	0.310
		0.500	0.208	0.275	0.210	0.119	0.401	0.315	0.317	0.264
Cross-Encoder	PileNER	0.050	0.166	0.385	0.263	0.144	0.475	0.438	0.480	0.336
		0.100	0.178	0.423	0.264	0.147	0.525	0.451	0.534	0.360
		0.150	0.174	0.434	0.257	0.143	0.556	0.454	0.561	0.368
		0.200	0.156	0.439	0.247	0.135	0.579	0.451	0.579	0.370
		0.300	0.114	0.431	0.226	0.111	0.610	0.433	0.600	0.361
		0.400	0.073	0.409	0.205	0.081	0.628	0.405	0.584	0.341
		0.500	0.024	0.370	0.179	0.053	0.634	0.363	0.533	0.308
	Euro-GLiNER-x	0.050	0.073	0.432	0.255	0.083	0.665	0.456	0.562	0.361
		0.100	0.067	0.452	0.267	0.075	0.691	0.462	0.602	0.374
		0.150	0.063	0.459	0.266	0.069	0.708	0.463	0.623	0.379
		0.200	0.059	0.463	0.261	0.062	0.719	0.464	0.642	0.382
		0.300	0.049	0.466	0.250	0.052	0.735	0.467	0.660	0.383
		0.400	0.036	0.463	0.235	0.040	0.743	0.465	0.669	0.379
		0.500	0.024	0.456	0.221	0.030	0.745	0.464	0.672	0.373
	FiNERWeb	0.050	0.199	0.475	0.279	0.185	0.509	0.463	0.547	0.380
		0.100	0.215	0.489	0.292	0.199	0.542	0.470	0.588	0.399
		0.150	0.225	0.494	0.297	0.208	0.562	0.473	0.610	0.410
		0.200	0.234	0.495	0.300	0.214	0.577	0.474	0.624	0.417
		0.300	0.244	0.488	0.293	0.217	0.599	0.472	0.625	0.420
		0.400	0.240	0.464	0.277	0.210	0.613	0.464	0.614	0.412
		0.500	0.225	0.424	0.256	0.185	0.618	0.450	0.587	0.392

Table 8: Detailed results for `xlm-roberta-base`.

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.	Data-set	τ	Dynamic-NER	Masakha-NER	MultiCoNER v1	MultiCoNER v2	Multi-NERD	PAN-X	UNER	Avg.
Bi-Encoder	PileNER	0.050	0.081	0.335	0.167	0.061	0.379	0.314	0.357	0.242
		0.100	0.086	0.363	0.172	0.064	0.432	0.327	0.402	0.264
		0.150	0.091	0.377	0.174	0.064	0.467	0.334	0.428	0.276
		0.200	0.095	0.385	0.169	0.063	0.492	0.337	0.451	0.285
		0.300	0.093	0.391	0.158	0.060	0.526	0.336	0.485	0.293
		0.400	0.085	0.388	0.142	0.053	0.543	0.328	0.507	0.292
		0.500	0.067	0.377	0.130	0.045	0.548	0.314	0.517	0.285
	Euro-GLiNER-x	0.050	0.025	0.391	0.118	0.030	0.541	0.347	0.475	0.275
		0.100	0.026	0.395	0.106	0.029	0.574	0.344	0.506	0.283
		0.150	0.025	0.393	0.097	0.028	0.593	0.341	0.518	0.285
		0.200	0.023	0.391	0.090	0.026	0.606	0.338	0.528	0.286
		0.300	0.023	0.384	0.080	0.023	0.620	0.330	0.539	0.286
		0.400	0.020	0.375	0.071	0.018	0.623	0.323	0.543	0.282
		0.500	0.018	0.362	0.064	0.015	0.620	0.315	0.540	0.276
	FiNERWeb	0.050	0.107	0.455	0.197	0.082	0.463	0.405	0.512	0.317
		0.100	0.117	0.451	0.197	0.087	0.501	0.404	0.536	0.328
		0.150	0.118	0.429	0.188	0.086	0.504	0.394	0.522	0.320
		0.200	0.121	0.395	0.172	0.082	0.490	0.378	0.500	0.305
		0.300	0.119	0.324	0.142	0.069	0.441	0.336	0.427	0.265
		0.400	0.099	0.254	0.109	0.051	0.367	0.287	0.357	0.218
		0.500	0.073	0.205	0.081	0.028	0.293	0.244	0.293	0.174
Cross-Encoder	PileNER	0.050	0.143	0.343	0.216	0.132	0.402	0.372	0.384	0.285
		0.100	0.154	0.390	0.231	0.142	0.437	0.380	0.435	0.310
		0.150	0.164	0.417	0.238	0.148	0.462	0.382	0.467	0.325
		0.200	0.172	0.435	0.239	0.150	0.483	0.383	0.496	0.337
		0.300	0.183	0.454	0.236	0.149	0.519	0.382	0.533	0.351
		0.400	0.175	0.462	0.230	0.140	0.548	0.378	0.565	0.357
		0.500	0.161	0.458	0.215	0.122	0.569	0.368	0.591	0.355
	Euro-GLiNER-x	0.050	0.026	0.372	0.180	0.028	0.551	0.369	0.406	0.276
		0.100	0.019	0.393	0.178	0.030	0.584	0.375	0.441	0.289
		0.150	0.016	0.401	0.169	0.031	0.602	0.377	0.460	0.294
		0.200	0.010	0.405	0.160	0.031	0.615	0.379	0.473	0.296
		0.300	0.007	0.408	0.148	0.027	0.630	0.381	0.491	0.299
		0.400	0.004	0.406	0.137	0.021	0.635	0.381	0.509	0.299
		0.500	0.003	0.403	0.127	0.015	0.633	0.381	0.520	0.297
	FiNERWeb	0.050	0.127	0.409	0.196	0.107	0.421	0.396	0.452	0.301
		0.100	0.136	0.437	0.209	0.117	0.459	0.406	0.497	0.323
		0.150	0.144	0.450	0.214	0.122	0.481	0.412	0.522	0.335
		0.200	0.150	0.457	0.215	0.123	0.500	0.414	0.544	0.343
		0.300	0.158	0.463	0.209	0.116	0.527	0.412	0.563	0.350
		0.400	0.161	0.454	0.201	0.101	0.542	0.405	0.569	0.348
		0.500	0.158	0.437	0.188	0.082	0.543	0.395	0.556	0.337

Table 10: Detailed results for `mT5-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 11: Detailed results for `mmBERT-base`.

Arc.	Dataset	τ	Dynamic-NER	Masakha-NER	MultiCoNER v1	MultiCoNER v2	Multi-NERD	PAN-X	UNER	Avg.
Bi-Encoder	PileNER	0.050	0.070	0.416	0.229	0.063	0.417	0.409	0.514	0.303
		0.100	0.078	0.439	0.226	0.065	0.481	0.402	0.570	0.323
		0.150	0.084	0.446	0.212	0.064	0.517	0.387	0.602	0.330
		0.200	0.086	0.446	0.196	0.061	0.538	0.370	0.615	0.330
		0.300	0.077	0.437	0.163	0.052	0.549	0.330	0.631	0.320
		0.400	0.066	0.418	0.136	0.041	0.538	0.286	0.627	0.302
		0.500	0.044	0.391	0.113	0.032	0.514	0.238	0.601	0.276
	Euro-GLiNER-x	0.050	0.037	0.493	0.185	0.078	0.664	0.495	0.708	0.380
		0.100	0.037	0.484	0.177	0.079	0.671	0.487	0.709	0.378
		0.150	0.038	0.474	0.171	0.078	0.667	0.478	0.703	0.372
		0.200	0.038	0.463	0.161	0.075	0.664	0.472	0.699	0.367
		0.300	0.033	0.442	0.144	0.070	0.657	0.456	0.688	0.356
		0.400	0.028	0.421	0.123	0.063	0.653	0.439	0.680	0.344
		0.500	0.027	0.394	0.102	0.057	0.645	0.418	0.666	0.330
	FiNERWeb	0.050	0.134	0.515	0.272	0.110	0.462	0.498	0.602	0.371
		0.100	0.144	0.526	0.281	0.117	0.497	0.500	0.648	0.388
		0.150	0.150	0.513	0.285	0.122	0.515	0.491	0.649	0.389
		0.200	0.155	0.483	0.286	0.125	0.523	0.477	0.627	0.382
		0.300	0.158	0.417	0.276	0.125	0.499	0.429	0.550	0.350
		0.400	0.158	0.356	0.252	0.119	0.462	0.374	0.471	0.313
		0.500	0.151	0.292	0.218	0.107	0.407	0.314	0.404	0.270
Cross-Encoder	PileNER	0.050	0.177	0.369	0.223	0.142	0.434	0.423	0.418	0.312
		0.100	0.197	0.407	0.242	0.147	0.485	0.438	0.491	0.344
		0.150	0.192	0.424	0.251	0.143	0.517	0.442	0.534	0.358
		0.200	0.175	0.434	0.251	0.133	0.540	0.439	0.571	0.363
		0.300	0.136	0.437	0.241	0.104	0.567	0.421	0.610	0.359
		0.400	0.084	0.428	0.220	0.074	0.575	0.378	0.608	0.338
		0.500	0.051	0.402	0.188	0.047	0.563	0.313	0.551	0.302
	Euro-GLiNER-x	0.050	0.144	0.443	0.279	0.054	0.618	0.482	0.632	0.379
		0.100	0.148	0.450	0.284	0.055	0.645	0.484	0.649	0.388
		0.150	0.138	0.453	0.282	0.053	0.662	0.485	0.654	0.390
		0.200	0.123	0.453	0.277	0.050	0.674	0.484	0.659	0.389
		0.300	0.090	0.452	0.262	0.043	0.688	0.483	0.661	0.383
		0.400	0.055	0.445	0.243	0.034	0.691	0.480	0.660	0.373
		0.500	0.031	0.436	0.223	0.024	0.687	0.475	0.653	0.361
	FiNERWeb	0.050	0.250	0.445	0.285	0.191	0.488	0.472	0.495	0.375
		0.100	0.265	0.480	0.297	0.205	0.525	0.481	0.547	0.400
		0.150	0.273	0.491	0.302	0.209	0.547	0.483	0.575	0.411
		0.200	0.275	0.495	0.301	0.210	0.561	0.481	0.589	0.416
		0.300	0.270	0.490	0.285	0.201	0.578	0.470	0.603	0.414
		0.400	0.258	0.471	0.267	0.183	0.579	0.454	0.600	0.402
		0.500	0.236	0.444	0.241	0.154	0.569	0.429	0.572	0.378

Table 12: Detailed results for `mdeberta-v3-base`.

Arc.	Config	τ	Dynamic-NER	Masakha-NER	MultiCoNER v1	MultiCoNER v2	Multi-NERD	PAN-X	UNER	Avg.
Bi-Encoder	$\lambda = 10.0$	0.05	0.142	0.362	0.264	0.101	0.419	0.445	0.458	0.313
		0.1	0.153	0.410	0.282	0.111	0.460	0.461	0.516	0.342
		0.15	0.160	0.434	0.294	0.118	0.484	0.471	0.553	0.359
		0.2	0.166	0.452	0.303	0.123	0.502	0.477	0.585	0.373
		0.3	0.178	0.473	0.316	0.132	0.534	0.485	0.622	0.391
		0.4	0.187	0.485	0.326	0.140	0.560	0.489	0.652	0.405
		0.5	0.196	0.488	0.332	0.147	0.573	0.489	0.667	0.413
	$\lambda = 100.0$	0.05	0.078	0.164	0.210	0.074	0.331	0.346	0.276	0.211
		0.1	0.085	0.196	0.220	0.079	0.350	0.366	0.314	0.230
		0.15	0.088	0.221	0.227	0.082	0.363	0.379	0.341	0.243
		0.2	0.091	0.243	0.232	0.085	0.372	0.387	0.360	0.253
		0.3	0.094	0.282	0.240	0.089	0.388	0.401	0.394	0.270
		0.4	0.097	0.316	0.247	0.092	0.401	0.412	0.419	0.283
		0.5	0.100	0.348	0.255	0.095	0.413	0.421	0.446	0.297
Cross-Encoder	$\lambda = 10.0$	0.05	0.119	0.330	0.191	0.121	0.387	0.365	0.425	0.277
		0.1	0.136	0.372	0.207	0.136	0.416	0.382	0.476	0.304
		0.15	0.147	0.394	0.219	0.146	0.435	0.392	0.506	0.320
		0.2	0.151	0.410	0.227	0.154	0.452	0.399	0.531	0.332
		0.3	0.148	0.429	0.240	0.165	0.481	0.408	0.565	0.348
		0.4	0.129	0.444	0.252	0.171	0.507	0.415	0.590	0.358
		0.5	0.087	0.455	0.264	0.168	0.531	0.422	0.617	0.363
	$\lambda = 100.0$	0.05	0.048	0.170	0.145	0.054	0.290	0.315	0.304	0.190
		0.1	0.052	0.200	0.155	0.058	0.312	0.331	0.340	0.207
		0.15	0.054	0.221	0.161	0.060	0.325	0.340	0.363	0.218
		0.2	0.056	0.238	0.167	0.062	0.336	0.348	0.383	0.227
		0.3	0.060	0.265	0.177	0.065	0.354	0.359	0.413	0.242
		0.4	0.064	0.288	0.187	0.068	0.370	0.368	0.441	0.255
		0.5	0.068	0.310	0.197	0.072	0.385	0.377	0.468	0.268

Table 13: Detailed results for loss function ablation using BCE loss but up-weighting the loss of positive examples by λ.

Arc.	Dataset	τ	Dynamic-NER	Masakha-NER	MultiCoNER v1	MultiCoNER v2	Multi-NERD	PAN-X	UNER	Avg.
Bi-Encoder	$\alpha = 0.25$, $\gamma = 0.0$	0.05	0.206	0.474	0.323	0.134	0.560	0.481	0.674	0.407
		0.1	0.205	0.416	0.272	0.127	0.515	0.440	0.594	0.367
		0.15	0.189	0.336	0.211	0.115	0.449	0.389	0.490	0.311
		0.2	0.173	0.273	0.171	0.094	0.398	0.342	0.420	0.267
		0.3	0.126	0.207	0.127	0.052	0.339	0.282	0.315	0.207
		0.4	0.068	0.162	0.092	0.025	0.288	0.240	0.232	0.158
		0.5	0.025	0.117	0.051	0.017	0.229	0.200	0.170	0.115
	$\alpha = 0.50$, $\gamma = 0.0$	0.05	0.187	0.482	0.314	0.119	0.530	0.476	0.675	0.398
		0.1	0.204	0.466	0.327	0.128	0.546	0.464	0.669	0.401
		0.15	0.213	0.432	0.315	0.132	0.533	0.444	0.618	0.384
		0.2	0.211	0.396	0.286	0.131	0.512	0.419	0.583	0.363
		0.3	0.203	0.330	0.224	0.119	0.459	0.373	0.500	0.316
		0.4	0.180	0.270	0.166	0.094	0.397	0.328	0.404	0.263
		0.5	0.136	0.211	0.125	0.069	0.340	0.295	0.308	0.212
	$\alpha = 0.75$, $\gamma = 0.0$	0.05	0.167	0.460	0.298	0.119	0.492	0.470	0.619	0.375
		0.1	0.182	0.487	0.320	0.131	0.541	0.479	0.672	0.402
		0.15	0.192	0.496	0.330	0.138	0.566	0.481	0.691	0.414
		0.2	0.202	0.495	0.337	0.144	0.576	0.478	0.678	0.415
		0.3	0.216	0.467	0.329	0.148	0.555	0.463	0.636	0.402
		0.4	0.223	0.410	0.305	0.145	0.504	0.435	0.582	0.372
		0.5	0.221	0.352	0.270	0.135	0.452	0.392	0.518	0.334
	$\alpha = 0.5$, $\gamma = 1.0$	0.05	0.147	0.370	0.278	0.106	0.410	0.444	0.495	0.322
		0.1	0.172	0.465	0.316	0.126	0.482	0.475	0.610	0.378
		0.15	0.192	0.499	0.334	0.136	0.532	0.486	0.675	0.408
		0.2	0.206	0.509	0.341	0.142	0.562	0.491	0.705	0.422
		0.3	0.218	0.458	0.307	0.146	0.531	0.462	0.645	0.396
		0.4	0.188	0.334	0.222	0.128	0.429	0.386	0.483	0.310
		0.5	0.119	0.225	0.148	0.094	0.342	0.305	0.327	0.223
	$\alpha = 0.75$, $\gamma = 2.0$	0.05	0.044	0.129	0.141	0.034	0.252	0.296	0.230	0.161
		0.1	0.058	0.200	0.169	0.046	0.315	0.362	0.312	0.209
		0.15	0.065	0.279	0.189	0.052	0.355	0.397	0.378	0.245
		0.2	0.073	0.351	0.207	0.058	0.395	0.421	0.440	0.278
		0.3	0.088	0.439	0.240	0.071	0.466	0.448	0.551	0.329
		0.4	0.108	0.479	0.273	0.082	0.543	0.467	0.647	0.371
		0.5	0.124	0.499	0.293	0.087	0.614	0.480	0.712	0.401

Table 14: Detailed results for loss function ablation using focal loss with parameters α and γ for the bi-encoder architecture.

Arc.	Dataset	τ	Dynamic-NER	Masakha-NER	MultiCoNER v1	MultiCoNER v2	Multi-NERD	PAN-X	UNER	Avg.
Cross-Encoder	$\alpha = 0.25$, $\gamma = 0.0$	0.05	0.247	0.509	0.312	0.197	0.594	0.488	0.621	0.424
		0.1	0.232	0.511	0.332	0.174	0.638	0.500	0.668	0.437
		0.15	0.206	0.494	0.335	0.147	0.654	0.505	0.682	0.432
		0.2	0.174	0.466	0.325	0.124	0.658	0.505	0.690	0.420
		0.3	0.111	0.408	0.285	0.089	0.649	0.488	0.663	0.385
		0.4	0.075	0.335	0.229	0.063	0.618	0.457	0.592	0.338
		0.5	0.046	0.248	0.157	0.042	0.564	0.407	0.529	0.285
	$\alpha = 0.50$, $\gamma = 0.0$	0.05	0.224	0.445	0.282	0.180	0.471	0.488	0.511	0.371
		0.1	0.247	0.486	0.302	0.196	0.518	0.502	0.563	0.402
		0.15	0.250	0.504	0.315	0.205	0.550	0.510	0.595	0.418
		0.2	0.248	0.513	0.324	0.207	0.574	0.515	0.614	0.428
		0.3	0.235	0.520	0.337	0.193	0.611	0.522	0.651	0.438
		0.4	0.192	0.514	0.344	0.164	0.633	0.526	0.665	0.434
		0.5	0.144	0.497	0.337	0.129	0.647	0.523	0.677	0.422
	$\alpha = 0.75$, $\gamma = 0.0$	0.05	0.096	0.419	0.227	0.072	0.445	0.411	0.501	0.310
		0.1	0.097	0.440	0.243	0.077	0.492	0.424	0.546	0.331
		0.15	0.097	0.452	0.254	0.073	0.524	0.432	0.573	0.343
		0.2	0.092	0.460	0.261	0.065	0.546	0.437	0.592	0.350
		0.3	0.073	0.470	0.270	0.051	0.573	0.443	0.611	0.356
		0.4	0.045	0.474	0.267	0.041	0.587	0.448	0.627	0.356
		0.5	0.021	0.472	0.255	0.030	0.587	0.448	0.626	0.348
	$\alpha = 0.5$, $\gamma = 1.0$	0.05	0.049	0.261	0.168	0.026	0.377	0.349	0.316	0.221
		0.1	0.063	0.357	0.196	0.032	0.461	0.387	0.410	0.272
		0.15	0.052	0.400	0.218	0.029	0.524	0.408	0.479	0.301
		0.2	0.016	0.423	0.226	0.019	0.559	0.422	0.534	0.314
		0.3	0.001	0.437	0.210	0.004	0.517	0.432	0.585	0.312
		0.4	0.000	0.397	0.172	0.000	0.400	0.414	0.550	0.276
		0.5	0.000	0.296	0.134	0.000	0.330	0.356	0.441	0.223
	$\alpha = 0.75$, $\gamma = 2.0$	0.05	0.108	0.106	0.175	0.138	0.292	0.319	0.207	0.192
		0.1	0.141	0.154	0.203	0.174	0.352	0.366	0.295	0.241
		0.15	0.157	0.212	0.221	0.193	0.387	0.397	0.364	0.276
		0.2	0.172	0.284	0.237	0.210	0.417	0.423	0.421	0.309
		0.3	0.208	0.416	0.267	0.241	0.476	0.460	0.517	0.369
		0.4	0.236	0.480	0.295	0.275	0.539	0.483	0.600	0.415
		0.5	0.188	0.489	0.320	0.278	0.608	0.495	0.667	0.435

Table 15: Detailed results for loss function ablation using focal loss with parameters α and γ for the cross-encoder architecture.

Arc.	Config	Dynamic-NER	Masakha-NER	MultiCoNER v1	MultiCoNER v2	Multi-NERD	PAN-X	UNER	Avg.
Bi-Encoder	$\alpha = 0.3, \beta = 0.7$	0.196	0.435	0.322	0.113	0.556	0.478	0.664	0.395
	$\alpha = 0.5, \beta = 0.5$	0.180	0.501	0.329	0.129	0.592	0.502	0.688	0.417
	$\alpha = 0.7, \beta = 0.3$	0.098	0.516	0.318	0.118	0.631	0.480	0.701	0.409
Cross-Encoder	$\alpha = 0.3, \beta = 0.7$	0.240	0.447	0.351	0.241	0.635	0.372	0.724	0.430
	$\alpha = 0.5, \beta = 0.5$	0.292	0.511	0.379	0.223	0.600	0.387	0.655	0.435
	$\alpha = 0.7, \beta = 0.3$	0.227	0.445	0.337	0.188	0.587	0.355	0.604	0.392

Table 16: Detailed results for loss function ablation using a contrastive loss with different parameter α (typing loss) and β (thresholding loss).