The Devil Is in the Word Alignment Details: On Translation-Based Cross-Lingual Transfer for Token Classification Tasks

Anonymous ACL submission

Abstract

Translation-based strategies for cross-lingual transfer (XLT) such as *translate-train*—training on noisy target-language data translated from the source language—and *translate-test*—evaluating on noisy source-language data translated from the target language—are competitive XLT baselines. In XLT for token classification tasks, however, these strategies include *label projection*, the challenging step of mapping the labels from each token in the original sentence to its counterpart(s) in the translation. Although word aligners (WAs) are commonly used for label projection, their low-level design decisions have not been systematically investigated in translation-based XLT. Moreover, recent marker-based methods, which project labels by inserting tags around spans before (after) translation, claim to outperform WAs in label projection for XLT. In this work, we revisit WAs for label projection, systematically investigating the effects that low-level design decisions have on token-level XLT, namely: (i) the algorithm for projecting labels between (multi-)token spans, (ii) filtering strategy for reducing the proportion of noisy data, and (iii) pre-tokenization of the translated sentence. We find that all of these have a substantial impact on downstream XLT performance and show that, with optimal choices, WA offers XLT performance comparable to that of marker-based methods. We then introduce a new projection strategy that ensembles *translate-train* and *translate-test* predictions and show that it substantially outperforms the marker-based projection. Crucially, we show that this ensembling also reduces sensitivity to low-level WA design choices, resulting in more robust XLT for token classification tasks.

1 Introduction

In recent years, multilingual language models (mLMs) have *de facto* become the main vehicle of cross-lingual transfer (XLT): fine-tuned on labeled task data in a high-resource source language, mLMs can make predictions in target languages with few (few-shot XLT) to no (zero-shot XLT) labeled task instances (Wu and Dredze, 2019; Wang et al., 2019; Lauscher et al., 2020; Schmidt et al., 2022). While both encoder-only (Devlin et al., 2019; Conneau et al., 2020; He et al., 2023) and decoder-only (Team et al., 2024; Hui et al., 2024; Grattafiori et al., 2024) mLMs have demonstrated strong XLT performance for sequence classification tasks, in XLT for token classification tasks the comparatively smaller encoder-only mLMs, like XLM-R (Conneau et al., 2020), continue to outperform the much larger decoder LLMs (Ahuja et al., 2023; Le et al., 2024; Parekh et al., 2024).

Much of the above work points to translation-based XLT strategies——where a machine translation (MT) model is used to either (1) gather noisy target language data by translating the clean source language data prior to training, known as *translate-train* (`T-Train`) or (2) translate clean target language instances into the (noisy) source language before inference, known as *translate-test* (`T-Test`)—as competitive approaches for XLT (Hu et al., 2020; Ruder et al., 2021; Ebrahimi et al., 2022; Aggarwal et al., 2022). More elaborate translation-based XLT strategies have recently been shown to further improve the transfer performance (Artetxe et al., 2023; Ebing and Glavaš, 2024).

The effectiveness of translation-based XLT, however, has predominantly been showcased on sequence-level classification tasks (Ruder et al., 2021; Oh et al., 2022; Artetxe et al., 2023). This is in part due to the fact that translation-based XLT for *token classification tasks* entails the (difficult) step of *label projection*. Traditionally, label projection is tackled with word aligners (WAs) (Och and Ney, 2003; Dyer et al., 2013; Dou and Neubig, 2021), which map each token in the source sequence to a corresponding token in the target sequence. Recent WA work leverages contextualized embeddings from mLMs (e.g., mBERT) to produce

token alignments (Jalili Sabet et al., 2020; Dou and Neubig, 2021; Wang et al., 2022). Although WA research has a long-standing track record in NLP (Och and Ney, 2003; Dyer et al., 2013; Jalili Sabet et al., 2020; Dou and Neubig, 2021; Wang et al., 2022), standard WA evaluation protocols do not include translation-based XLT for token classification tasks. Because of this, a range of low-level design decisions related to token-level XLT using WAs—such as (i) the algorithm for projecting the labels, (ii) filtering techniques to reduce the number of noisily mapped labels, and (iii) the pre-tokenization applied to the translated target sentence before it can be aligned to the clean source sentence—remain largely uninvestigated.

In the meantime, marker-based label projection (Chen et al., 2023; Le et al., 2024) has largely replaced WA as the default approach for label projection for token-level XLT. These approaches insert tags (e.g., "[", "]") around entities of interest, either (i) before translation to preserve the markers throughout the translation process and recover the spans afterward (Chen et al., 2023), or (ii) post-translation by means of constrained decoding (Le et al., 2024). In this line of work, token-level translation-based XLT is explicitly evaluated, demonstrating strong performance for both `T-Train` and `T-Test`, rendering WA-based XLT for token classification tasks inferior (Chen et al., 2023; Le et al., 2024). While these efforts provide code and technical details for their proposed marker-based methods, they do not provide the low-level design decisions and implementation details for label projection with WAs: as such, they possibly underestimate the token-level translation-based XLT with WAs due to suboptimal design choices.

Contributions. Because of this, we **(1)** systematically investigate WA for token-level translation-based XLT. We start by evaluating the effect of low-level design decisions including (i) the exact algorithm for mapping the labels from the clean source sentence to the translated target sentence based on word alignments; (ii) filtering strategies to identify noisy label projections in the translated target sentences; and (iii) the pre-tokenization of the translated sentences, which is required to align the tokens to their counterparts in the clean source sentence. We show that these design choices can have a tremendous impact on translation-based XLT for token-level tasks. For example, using a language-specific pre-tokenizer instead of simple whitespace pre-tokenization improves performance of `T-Test` by up to 13.2% (details in Sec. 2). Generally, we find `T-Test` to be more sensitive than `T-Train` to low-level design decisions of WA. **(2)** We then extensively compare WA-based label projection—with what we found to be optimal low-level design choices—against state-of-the-art marker-based label projection methods in token-level XLT on 3 established benchmarks encompassing 35 topologically diverse languages. Contrary to prior claims (Chen et al., 2023; Le et al., 2024; Parekh et al., 2024), we show that "optimal" WA-based label projection matches or surpasses the performance of marker-based approaches in translation-based XLT on token-level tasks. **(3)** Moreover, we propose a more sophisticated method for token-level translation-based XLT with WAs based on ensembling `T-Train` and `T-Test` (Oh et al., 2022). For each token, we average the corresponding probability distributions over the labels produced by `T-Train` and `T-Test`. Our proposed ensemble (`ETT`) improves the transfer performance, substantially outperforming state-of-the-art marker-based approaches. More importantly, `ETT` drastically reduces the sensitivity of `T-Train` and `T-Test` to low-level WA design decisions. **(4)** Finally, we show that our findings hold for different choices of (i) MT model (i.e., impact of translation quality), (ii) WA model, and (iii) base mLM (i.e., encoder vs. decoder models)

2 Token-Level XLT via Word Alignment

We first detail the low-level design decisions of WA-based label projection that we investigate: span mapping, filtering strategies, and pre-tokenization, and then describe our new label projection approach that ensembles `T-Train` and `T-Test`.

2.1 Label Projection with WA: Design Choices

Span Mapping. Translation-based XLT for token-level tasks requires mapping labeled spans of tokens—named entities, answer spans in question answering, or slot-values in task-oriented dialog—from the source language input to the (spans of) tokens in the translation. Instead of simply projecting the token-level labels based on the alignments produced by a WA model, we propose a more robust, span-based label projection, as illustrated in Figure 1. We start from a pre-tokenized source sen-

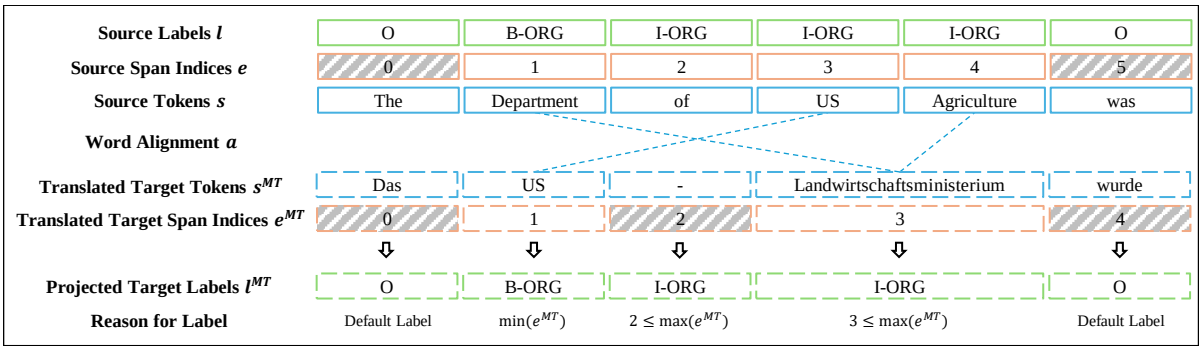

Figure 1: Schematic overview of our word alignment-based label projection for `T-Train`.

tence, i.e., the sequence of (pre-)tokens s[1] and their corresponding labels l. We then concatenate the tokens in s and translate the resulting sentence into the target language with an MT model. We then pre-tokenize the translation, obtaining the sequence of (pre-)tokens s^{MT}. We then feed s and s^{MT} into the WA, obtaining the set of alignment pairs $a_{i,j}$ (denoting that the i-th token in s is aligned to the j-th token in s^{MT}). We next carry out span-based label projection, i.e., establish the sequence of labels l^{MT}, on the basis of these alignments.

Let $e = \{m, \ldots, n\}$ be the set of token indices of one labeled span in s (e.g., "Department of US Agriculture" spanning indices 1-4 in Fig. 1). Using the alignments produced by the WA model, we then collect the set of corresponding indices $e^{MT} = \{k, \ldots, l\}$ from s^{MT} (e.g., indices $\{1, 3\}$ of the tokens "US" and "Landwirtschaftsministerium", see Fig. 1). We then project the labels based on e and e^{MT} as follows. We assume a standard BIO scheme in which the first token in a span is labeled with a different tag (`B-Tag`) from all other tokens of the span (`I-Tag`). We thus project the `B-Tag` label to the token in s^{MT} at index $min(e^{MT})$ (i.e., the smallest index among all tokens in s^{MT} that WA aligned to any of the source tokens with indices in e). We next assume that the span has to be contiguous in the translation too and assign the respective `I-Tag` to all tokens in s^{MT} between indices $min(e^{MT}) + 1$ and $max(e^{MT})$.[2]

We also experimented with the simple approach of naively projecting the labels based on the token alignments: based on the alignment $a_{i,j}$, we copy the label from the i-token in s to the j-th token in s^{MT}.[3] This simple projection strategy, however, consistently yielded worse results in our initial experiments: we thus ran the full evaluation using only our span mapping strategy described above. We believe that the simple projection along word alignments performs poorly due to frequent changes in word order between languages (as exemplarily illustrated in Figure 1). For `T-Train`, we additionally experimented with string matching (`STR-MAT`) between the source language and translated target language spans, as certain spans like dates (e.g., *1997*) or names (e.g., *Michael Jordan*) are often fully preserved in translation.[4]

The span mapping for `T-Test` follows the same procedure, only in reverse. We translate the target-language sentence with MT into the source language and then predict the labels with the fine-tuned mLM. We then run the WA and project the predicted labels to the original target-language sentence using our span-based projection approach.

Filtering Strategies. The success of the above span mapping directly depends on the quality of WA for a concrete language pair, which is affected by (i) the amount of parallel data for the pair used in WA training, (ii) the amount of monolingual data for the languages in question seen by the WA's underlying mLM in pretraining (Dou and Neubig, 2021; Wang et al., 2022), and (iii) the linguistic proximity between the two languages (and in particular whether they have similar word order). To mitigate the impact of imperfect word alignment, we propose several strategies for detecting and eliminating instances with low-quality word alignment.

Complete Source (`COM-SRC`). We test if all indices of a span $e = \{m, ..., n\}$ have an alignment in

[1] Most datasets for token classification tasks come pre-tokenized with labels assigned to these predefined (typically word-level) tokens. For a source text that is not pre-tokenized (e.g., see TyDiQA in §3), we tokenize it with the same pre-tokenization as for the translated text.

[2] If there is only one labeled span per instance, as in the case of TyDiQA (see 3), then there is no need to differentiate between `B-Tag` and `I-Tag`, so we conflate the two.

[3] If the j-th s^{MT} token is aligned to multiple tokens in s, we randomly select from which to project the label.

[4] We quantify the impact of `STR-MAT` in App G.

a, i.e., whether the corresponding tokens in s are aligned to at least one token in s^{MT}. Figure 1 illustrates an example of an incomplete source alignment: the token "of" in s is not aligned to any token in the translation s^{MT}. We assume that if e is partially unaligned, then e^{MT} is more likely to be incomplete and thus incorrect.

Complete Target (`COM-TGT`*).* The motivation for this filter is analogous to `COM-SRC`: we select only the instances for which all span tokens in the translated sentence are aligned to at least one token in the original source-language sentence. But since we do not have ground truth spans for the translation, we apply the following proxy: we retain only the instances for which the indices in e^{MT} constitute a continuous span. The example in Figure 1 does not satisfy this filter either, since $e^{MT} = \{1, 3\}$ is discontinuous.

Correct Scheme (`COR-SCH`*).* In this case we keep only the instances in which the mapped span in the target sentence adheres to the BIO scheme after projection, namely that the first token is assigned the `B-Tag` and all other tokens an `I-Tag`.

Complete Instance (`COM-INS`*).* Following Chen et al. (2023), we verify that the number and type of spans in l and l^{MT} match (e.g., if l has two spans with label *LOC* and one with label *PER* then l^{MT} must also have two *LOC* spans and one *PER* span).

We apply our filtering strategies to both `T-Train` and `T-Test`, with some exceptions: (i) we do not use `COR-SCH` for `T-Train`, as our span mapping already ensures BIO correctness; (ii) `COM-INS` cannot be applied to `T-Test` as it would require accessing gold labels. If a filter is not satisfied, in `T-Train` we simply remove the translated training instance; in `T-Test`, we do not project the labels for the corresponding span and assign the default label "O".

Pre-Tokenization. To apply word alignment for token-level tasks, both the original sentence and the translation need to be pre-tokenized. While the original sentence s is usually given in a pre-tokenized format, we still need to pre-tokenize the translation s^{MT}. We compare language-agnostic whitespace pre-tokenization (`WS-TOK`) against language-specific pre-tokenization (`SP-TOK`).[5] It is worth noting that it is more challenging to pre-tokenize target-language sentences in `T-Train` than English translations in `T-Test`.

[5]For brevity, we provide the details on the language-specific tokenizers in App. F.

2.2 Ensembling `T-Train` and `T-Test` (`ETT`)

Ensembling is an effective strategy for improving the predictions of two or more models by reducing the impact of individual errors (Wortsman et al., 2022). We next propose a translation-based strategy for token-level XLT that ensembles the predictions of `T-Train` and `T-Test` as follows. At inference time, the `T-Train` model produces class logits for each token s_i in the target-language sentence s. In contrast, the `T-Test` model outputs class logits over each token s^{MT}_j in the translated source-language sentence s^{MT}. We then use the alignments $a_{i,j}$ between s and s^{MT} produced by the WA model, and average the class logits between the aligned tokens s_i and s^{MT}_j. If a token in s^{MT}_j is not aligned to any token in s_i, we only use the `T-Train` prediction. Similarly, for tokens of spans that violate some filter (e.g., if `COM-SRC` is not satisfied), we default to the `T-Train` prediction.

In span extraction formulation of token-level tasks (i.e., no BIO scheme, see TyDiQA in §3), this approach has to be slightly modified because we do not have a vector of class logits for each token; instead, there are two logit distributions across all tokens—one for the span start and one for the span end. Here, we average the projected start/end logits produced by the `T-Test` model and the start/end logits predicted by the `T-Train` model.

3 Experimental Setup

Machine Translation. For translation, we utilize the state-of-the-art massively multilingual NLLB model with 3.3B parameters (Team et al., 2022). Following prior work (Artetxe et al., 2023; Ebing and Glavaš, 2024), we decode using beam search with a beam size of 5.

Evaluation Tasks. We evaluate on three established token classification tasks, covering both shallow understanding in short sequences (named entity recognition and slot filling) and complex reasoning over longer text (extractive QA). Our experiments span 35 diverse languages, ranging from high-resource languages, represented well in the pretraining corpus of the base mLM to low-resource languages unseen by the mLM. In all experiments, English is the source XLT language.[6]

Named Entity Recognition (NER). Our evaluation includes 18 of 20 languages from MasakhaNER 2.0 (Masakha) (Adelani et al., 2022) supported by

[6]We provide the complete list of languages in App. F.

the NLLB model used for translation. Masakha consists of underrepresented languages spoken in Sub-Saharan Africa. As source data, we use the English training (14k instances) and validation portions (3250 instances) of CoNLL (Tjong Kim Sang and De Meulder, 2003). We add a simple softmax classifier on top of the mLM to predict the class for each token.

Slot Filling (SL). We use the xSID dataset (van der Goot et al., 2021), which covers 10 diverse languages and dialects. xSID comprises only evaluation data, so we follow van der Goot et al. (2021) and use their publicly released English data for training and validation. The utterances are sourced from the Snips (Coucke et al., 2018) and Facebook (Schuster et al., 2019) SL datasets. After deduplication, we end up with over 36k instances for training and 300 for validation. As for NER, we simply add a softmax classifier on top of the mLM.

Extractive Question Answering (QA). For extractive QA, we resort to TyDiQA-GoldP (TyDiQA) (Clark et al., 2020). TyDiQA covers 8 typologically diverse languages with different scripts. We use the English training (3696 training) and validation portion (440 validation) as our source-language data. We jointly encode the question-context pair and—as common for tasks formulated as span extraction—feed the transformed sequence into a feed-forward classifier that predicts the start and end of the answer span.

Label Projection. We compare our WA label projection approaches against two state-of-the-art "marker-based" methods that tag the labeled spans and preserve the tags during translation.

Word Alignment (WA). In our main experiments, we resort to AccAlign (Wang et al., 2022), a state-of-the-art WA based on the multilingual sentence encoder LaBSE (Feng et al., 2022).[7]

EasyProject (Easy). We first compare our WA-based approach against the marker-based label projection method of Chen et al. (2023). Prior to translation, `Easy` inserts tags ("[", "]") around labeled spans (e.g., named entities). The MT is expected to preserve the tags in the translation, allowing for a trivial reconstruction of the labels. Note that `Easy` can only be used in `T-Train` and not in `T-Test`. Let t be the target-language sentence at inference time; in `T-Test`, the model will make predictions on its English translation s; `Easy` would then insert markers into s and back-translate to the target language, obtaining t'; but t' will generally differ from t, which is the actual sentence we need to label.

[7]We adopt the hyperparameters proposed by the authors.

	Masakha	xSID	Avg
Translate-Train			
NO-FILT	$65.5_{\pm 1.2}$	$\mathbf{82.8}_{\pm 0.6}$	$74.2_{\pm 1.0}$
COM-INS	$65.8_{\pm 1.3}$	$\mathbf{82.8}_{\pm 0.5}$	$\mathbf{74.3}_{\pm 1.0}$
+ COM-TGT	$66.0_{\pm 1.7}$	$82.0_{\pm 0.7}$	$74.0_{\pm 1.3}$
+ COM-TGT + COM-SRC	$\mathbf{66.6}_{\pm 1.1}$	$82.0_{\pm 0.9}$	$\mathbf{74.3}_{\pm 1.0}$
Translate-Test			
NO-FILT	$51.2_{\pm 0.4}$	$67.8_{\pm 0.4}$	$59.5_{\pm 0.4}$
COR-SCH	$58.2_{\pm 0.4}$	$74.9_{\pm 0.4}$	$66.6_{\pm 0.4}$
+ COM-TGT	$58.1_{\pm 0.5}$	$74.3_{\pm 0.5}$	$66.2_{\pm 0.5}$
+ COM-TGT + COM-SRC	$58.3_{\pm 0.5}$	$73.3_{\pm 0.5}$	$65.8_{\pm 0.5}$

Table 1: Results on the validation data for WA-based XLT with various filtering strategies. `NO-FILT` indicates that no filtering was applied. Results with XLM-R.

Codec. Our experiments further include Codec (Le et al., 2024), a label projection method that leverages constrained decoding as part of a two-step translation procedure. In the first step, the source sentence is simply translated into the target language (e.g., from English: "This is New York" to German: "Das ist New York"). Then, in step two, tags are inserted around the labeled spans in the source sentence (English: "This is [New York]") and now the marked sentence is fed again as input to the MT model: during decoding, the MT model is allowed to generate only the tokens from the translation obtained in the first step ("Das", "ist", "New", "York") or a tag ("[", "]").

Downstream Fine-Tuning. We use XLM-R Large (Conneau et al., 2020) as our base mLM. For `T-Test`, we also experiment with DeBERTaV3 Large (He et al., 2023) and LLM2Vec (BehnamGhader et al., 2024) as English-centric models. In `T-Train`, we fine-tune on both the clean English data and translated target-language data, following Ebing and Glavaš (2024) who show that this is better than training only on translations. In `T-Test`, we train the models only on the clean English data. We run all experiments with 3 random seeds and report the mean F_1 score and standard deviation. We provide full training details in App F).

4 Results and Discussion

First, using the validation portions of the respective datasets, we assess the impact of low-level WA design choices on token-level translation-based XLT

(§4.1). Based on these findings, we compare the most effective WA variant against the two state-of-the-art marker-based approaches, Easy and Codec, on test portions of all three datasets (§4.2). Finally, we provide further ablations in §4.3, analyzing the impact of the underlying mLM, WA, and MT model on the XLT performance.

4.1 WA Design Choices

Filtering Strategies. Our preliminary experiments (Table 1) reveal that instance filtering has a negligible effect on performance in T-Train: on average (between Masakha and xSID) none of the filtering strategies yields gains (i.e., over NO-FILT). This finding is positive, as finding an optimal filtering strategy for T-Train is costly: it requires (re-)training language-specific models for every change in the filtering strategy. In stark contrast, filtering, based on the correct scheme (COR-SCH), brings a substantial (+7.1% over NO-FILT) for T-Test. We find COR-SCH to be the most successful strategy for T-Test—adding additional filters (COM-TGT and/or COM-SRC) does not bring gains.[8]

Pre-Tokenization. Figure 2 summarizes the results for different pre-tokenization approaches. The results mirror the filtering findings: the pre-tokenization strategy has (1) little impact on the T-Train performance (language agnostic whitespace tokenization, WS-TOK, is marginally better for Masakha and xSID and language-specific tokenization, SP-TOK, brings marginal gains on TyDiQA); (2) much larger impact on T-Test: SP-TOK outperforms WS-TOK by 13.2% on Masakha, 5.9% on xSID and 3.2% on TyDiQA. Our findings are in line with prior work (Artetxe et al., 2023; Ebing and Glavaš, 2024), which showed that T-Test is more affected by translation quality than T-Test: our results extend this finding also to filtering and pre-tokenization strategies.

4.2 Main Results

We now compare the optimal WA-based configurations for T-Train and T-Test, respectively, against the state-of-the-art marker-based label projection methods Easy and Codec.

Translate-Train. We first note that T-Train, regardless of the label projection strategy (WA, Easy, or Codec), substantially outperforms zero-shot XLT with the mLM (e.g., by 14.2% on Masakha for WA-based projection). Contrary to the results of prior work that reported translation-based XLT with WAs inferior to Easy (Chen et al., 2023) and Codec (Le et al., 2024), we find that—when optimally configured—WA yields competitive performance: On TyDiQA and xSID, optimal WA-based T-Train lags marker-based transfer by less than 1%; and on Masakha WA-based transfer even slightly outperforms both Easy and Codec.

Translate-Test. Irrespective of the label projection approach, T-Test outperforms zero-shot XLT on Masakha and xSID (as well as T-Train on Masakha). On TydiQA, however, T-Test (with WA) yields substantially lower performance than T-Train; we did not evaluate Codec on TyDiQA for reasons provided in App. D. Again we ob-

[8] As filters available for the span extraction formulation of the task differ, we present the results for TyDiQA in App. G.

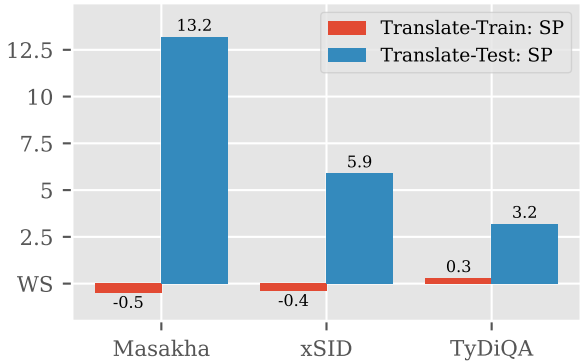

Figure 2: Transfer performance with WA for language-specific (SP) pre-tokenization relative to whitespace (WS) pre-tokenization. Results with XLM-R.

		Masakha	xSID	TyDiQA	Avg
Zero-Shot					
	X	$52.9_{\pm 1.8}$	$76.8_{\pm 1.4}$	$74.0_{\pm 1.0}$	$67.9_{\pm 1.5}$
Translate-Train					
Easy	X	$66.0_{\pm 0.9}$	$83.6_{\pm 0.9}$	$75.5_{\pm 1.0}$	$75.0_{\pm 0.8}$
Codec	X	$66.9_{\pm 1.6}$	$83.4_{\pm 0.8}$	-	-
WA	X	$67.1_{\pm 1.1}$	$82.7_{\pm 0.8}$	$75.2_{\pm 1.0}$	$75.0_{\pm 1.0}$
Translate-Test					
Codec	X	$72.0_{\pm 0.5}$	$79.4_{\pm 0.3}$	-	-
Codec	D	$72.4_{\pm 0.4}$	$79.5_{\pm 0.4}$	-	-
WA	X	$72.5_{\pm 0.5}$	$80.2_{\pm 0.3}$	$63.8_{\pm 1.1}$	$72.2_{\pm 0.8}$
WA	D	$\mathbf{72.9}_{\pm 0.4}$	$80.2_{\pm 0.4}$	$67.6_{\pm 1.0}$	$73.6_{\pm 0.7}$
Ensemble-Train-Test					
Easy + WA	X/D	$71.7_{\pm 0.7}$	$\mathbf{83.8}_{\pm 0.7}$	$\mathbf{76.3}_{\pm 0.9}$	$77.3_{\pm 0.7}$
Codec + WA	X/D	$72.3_{\pm 0.7}$	$82.8_{\pm 0.7}$	-	-
WA + WA	X/D	$72.6_{\pm 0.6}$	$83.4_{\pm 0.9}$	$76.2_{\pm 0.9}$	$\mathbf{77.4}_{\pm 0.8}$

Table 2: Main results for translation-based XLT for token-level tasks. Results with XLM-R (X) and DeBERTa (D).

		Masakha	xSID	TyDiQA	Avg
Zero-Shot					
	X	$52.9_{\pm 1.8}$	$76.8_{\pm 14}$	$74.0_{\pm 1.0}$	$67.9_{\pm 1.5}$
Translate-Test					
WS-TOK	D	$60.1_{\pm 0.3}$	$73.3_{\pm 0.4}$	$64.0_{\pm 0.8}$	$65.8_{\pm 0.5}$
SP-TOK	D	$\mathbf{72.9}_{\pm 0.4}$	$80.2_{\pm 0.4}$	$67.6_{\pm 1.0}$	$73.6_{\pm 0.7}$
Ensemble-Train-Test					
WS-TOK	X/D	$72.0_{\pm 0.7}$	$81.2_{\pm 1.0}$	$75.8_{\pm 0.9}$	$76.3_{\pm 0.9}$
SP-TOK	X/D	$72.6_{\pm 0.6}$	$\mathbf{83.4}_{\pm 0.9}$	$\mathbf{76.2}_{\pm 0.9}$	$\mathbf{77.4}_{\pm 0.8}$

Table 3: Results for translation-based XLT utilizing different pre-tokenizations for T-Test—whitespace (WS-TOK) and language-specific (SP-TOK). Results with XLM-R (X) and DeBERTa (D).

serve that the "optimal" WA-based label projection matches (in fact, slightly surpasses) the performance of the marker-based Codec on Masakha and xSID. This is encouraging because the label projection with Codec—due to its two-step translation procedure—is computationally much more expensive (i.e., slower) than WA. Further, consistent with findings of Artetxe et al. (2023) for sentence-level tasks, we observe that models solely trained on English (i.e., DeBERTa) offer gains over comparable mLMs (i.e., XLM-R). On TyDiQa, DeBERTa outperforms XLM-R by 3.8%, but the two offer comparable performance on Masakha and xSID. We speculate that this is because NER and slot labeling do not require advanced language understanding abilities and thus the monolingual English ability of an mLM suffices for these tasks.

Ensemble-Train-Test. On average, our proposed ensemble ETT improves over T-Train and T-Test by 2.4% and 3.8%, respectively. ETT even improves translation-based performance on TyDiQA, where T-Test substantially trails zero-shot XLT. We summarize our observations as follows: (i) in scenarios where T-Train performs better than T-Test, ETT achieves additional gains over T-Train by leveraging the complementary strengths of T-Test; (ii) in scenarios where T-Train performance is worse than T-Test, utilizing ETT does not harm because it results in similar performance as T-Test.

Robustness via Ensembling. Our preliminary studies on WA-related low-level design choices (§4.1) revealed notable performance variation, especially for T-Test. We now show that our proposed ensemble ETT not only improves performance over both T-Train and T-Test but also makes the performance much less sensitive to design details of WA. Table 3 compares T-Test and ETT with WA-based label projection for the two pre-tokenization strategies (WS-TOK and SP-TOK). For ETT, we modify the pre-tokenization only for the T-Test part of the ensemble and keep the T-Train pre-tokenization unchanged. For T-Test, WS-TOK underperforms SP-TOK by 7.8% on average (and even trails zero-shot XLT by 2.1%). In contrast, ETT almost completely closes the gap between the two (WS-TOK is behind SP-TOK by only 1.1%), making the choice of pre-tokenizer much less consequential for the final performance.

4.3 Further Findings

LLMs as Encoders. Prior work rendered decoder-only LLMs inferior to smaller encoder-only models (Ahuja et al., 2023; Le et al., 2024; Dukić and Snajder, 2024), but more recent efforts suggest that autoregressive LLMs can be post-hoc turned into competitive bidirectional encoders (BehnamGhader et al., 2024; Wang et al., 2024). We thus compare a state-of-the-art decoder-turned-encoder LLM2Vec (based on Llama-3-8B) (BehnamGhader et al., 2024) against the encoder models in translation-based XLT for token classification. Following original work, we add a linear classifier with dropout on top of LLM2Vec, fine-tuning only the classifier. Figure 3 summarizes the results.[9] We observe that much smaller DeBERTa is superior to LLM2Vec in T-Test (e.g., +11.4% on Masakha) and that T-Test with LLM2Vec even trails zero-shot transfer with XLM-R on xSID. In ETT (with XLM-R-based T-Train component), LLM2Vec becomes much more competitive and lags DeBERTa by only 2% on average, which further emphasizes the robustness that our ensembling (ETT) brings to translation-based XLT for token-level tasks.

Choice of Word Aligner. We next ablate the impact of the WA model on downstream transfer performance. We compare the widely used Awesome (Dou and Neubig, 2021; van der Goot et al., 2021; Chen et al., 2023; Le et al., 2024), based on mBERT, with the more recent AccAlign (Wang et al., 2022), which resorts to the multilingual sentence encoder LaBSE. Both WAs are released in the vanilla and fine-tuned variants. For the latter, the underlying mLM is explicitly fine-tuned with word-alignment objectives on parallel data (Dou and Neubig, 2021; Wang et al., 2022).

[9] We did not run LLM2Vec experiments on TyDiQA as LLMs yield better performance by solving QA generatively. We provide detailed LLM2Vec results in App. I

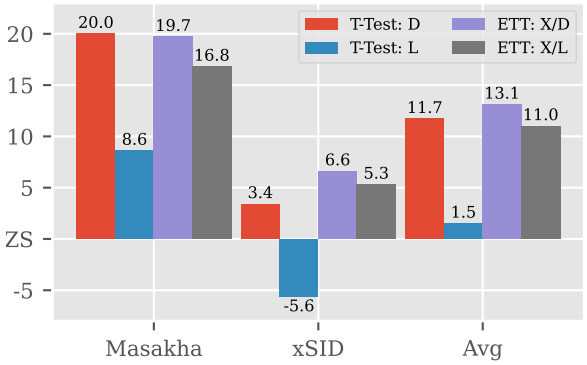

Figure 3: Results for translation-based XLT with LLM2Vec (L) vs. DeBERTa (D), relative to zero-shot XLT performance with XLM-R (X).

	Masakha	xSID	TyDiQA	Avg
Zero-Shot				
	$52.9_{\pm 1.8}$	$76.8_{\pm 1.4}$	$74.0_{\pm 1.0}$	$67.9_{\pm 1.5}$
Translate-Train				
AccAlign	$67.1_{\pm 1.2}$	$82.7_{\pm 0.8}$	$\mathbf{75.2}_{\pm 1.0}$	$\mathbf{75.0}_{\pm 1.0}$
AccAlign$_{noft}$	$66.7_{\pm 1.1}$	$\mathbf{82.9}_{\pm 0.5}$	$75.0_{\pm 1.1}$	$74.9_{\pm 1.0}$
Awesome$_{noft}$	$64.4_{\pm 1.3}$	$79.8_{\pm 0.8}$	$73.8_{\pm 1.4}$	$72.7_{\pm 1.2}$
Translate-Test				
AccAlign	$\mathbf{72.5}_{\pm 0.5}$	$80.2_{\pm 0.3}$	$63.8_{\pm 1.1}$	$72.2_{\pm 0.8}$
AccAlign$_{noft}$	$70.5_{\pm 0.5}$	$79.7_{\pm 0.3}$	$62.1_{\pm 1.1}$	$70.8_{\pm 0.7}$
Awesome$_{noft}$	$65.3_{\pm 0.4}$	$74.8_{\pm 0.3}$	$62.8_{\pm 1.1}$	$67.6_{\pm 0.7}$

Table 4: Comparison of translation-based XLT with different WAs. Results with XLM-R; $noft$ denotes vanilla WAs, without WA-specific fine-tuning.

Table 4 shows the results of the WA comparison. Without WA-specific fine-tuning, AccAlign outperforms Awesome by 2.2% for `T-Train` and 3.2% for `T-Test`, respectively. The results are mixed w.r.t. explicit WA fine-tuning: the fine-tuned AccAlign yields virtually no gains in `T-Train`, it does bring small performance boost (1.4%) in `T-Test`. This is in line with findings from Chen et al. (2023), who report similar behavior for Awesome. We hypothesize that the limited size and language diversity of WA fine-tuning limits the generalization to a broader set of (low-resource) languages, as evaluated in our work.

Translation Quality. Commercial MT models are typically considered to produce superior translation quality compared to their publicly available counterparts. To evaluate the impact of the MT model on token-level translation-based XLT, we generate translations using Google Translate (GT), which serves as a representative example of a commercial MT model. We report results for `T-Test` and `ETT` only (Table 5) as prior work already demonstrated that translation quality has a less pronounced impact on `T-Train` (Artetxe et al., 2023; Ebing and Glavaš, 2024). For `T-Test`, we find that GT outperforms NLLB by 1.7% on average. Nevertheless, the gains obtained by a more powerful MT model still trail the performance improvements introduced by using our ensemble (`ETT`) with NLLB only. `ETT`—regardless the MT model—outperforms `T-Test` with GT by more than 2%. Additionally, the difference in `ETT` performance between GT and NLLB is negligible (0.2%), which once more points to the robustness that `ETT` brings to XLT for token classification tasks.

		Masakha	xSID	TyDiQA	Avg
Zero-Shot					
	X	$50.8_{\pm 1.2}$	$76.8_{\pm 1.4}$	$74.0_{\pm 1.0}$	$67.2_{\pm 1.2}$
Translate-Test					
NLLB	D	$73.4_{\pm 0.5}$	$80.2_{\pm 0.4}$	$67.6_{\pm 1.0}$	$73.7_{\pm 0.7}$
GT	D	$\mathbf{75.0}_{\pm 0.4}$	$81.2_{\pm 0.3}$	$70.1_{\pm 1.2}$	$75.4_{\pm 0.8}$
Ensemble-Train-Test					
NLLB	X/D	$73.0_{\pm 0.6}$	$\mathbf{83.4}_{\pm 0.9}$	$76.2_{\pm 0.9}$	$77.5_{\pm 0.8}$
GT	X/D	$73.0_{\pm 0.6}$	$83.2_{\pm 0.9}$	$\mathbf{76.9}_{\pm 0.7}$	$\mathbf{77.7}_{\pm 0.8}$

Table 5: Results for translation-based XLT utilizing different translation models for `T-Test`—NLLB and Google Translate (GT). Results with XLM-R (X) and DeBERTa (D).

5 Conclusion

In this work, we thoroughly investigated the role of word aligners (WAs) in translation-based cross-lingual transfer for token classification tasks. Our experimentation on three established benchmarks covering 35 languages, revealed that low-level design decisions related to label projection via WA can have a substantial effect on translation-based XLT strategies, in particular translate-test. We then show that "optimal" WA-based label projection can match or even surpass the transfer performance of recent marker-based approaches (Chen et al., 2023; Le et al., 2024), contrary to their findings. Further, we proposed a more sophisticated WA-based transfer approach that ensembles predictions of translate-train and translate-test. We demonstrated that the proposed ensemble not only substantially increases transfer performance but also reduces the sensitivity of transfer performance to low-level design decisions of WA-based label projection.

6 Limitations

We focused on systematically exploring the design choices relevant for translation-based XLT using WA. However, our study is limited by the prevalent practice of creating new evaluation datasets by translating the data from an existing high-resource language to the desired (new) language. This applies to xSID and some languages of Masakha. The resulting data may contain distinct characteristics that stem from the translation process often referred to as *translationese*. Prior work (Artetxe et al., 2020) stated that translation-based XLT strategies might lead to the exploitation of translationese, slightly overestimating the true performance.

References

David Adelani, Graham Neubig, Sebastian Ruder, Shruti Rijhwani, Michael Beukman, Chester Palen-Michel, Constantine Lignos, Jesujoba Alabi, Shamsuddeen 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, Amelia Taylor, Fatoumata Kabore, Chris Chinenye Emezue, Anuoluwapo Aremu, Perez Ogayo, Catherine Gitau, Edwin Munkoh-Buabeng, Victoire Memdjokam Koagne, Allahsera Auguste Tapo, Tebogo Macucwa, Vukosi Marivate, Mboning Tchiaze Elvis, Tajuddeen Gwadabe, Tosin Adewumi, Orevaoghene Ahia, and Joyce Nakatumba-Nabende. 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.

Divyanshu Aggarwal, Vivek Gupta, and Anoop Kunchukuttan. 2022. IndicXNLI: Evaluating multilingual inference for Indian languages. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10994–11006, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Kabir Ahuja, Harshita Diddee, Rishav Hada, Millicent Ochieng, Krithika Ramesh, Prachi Jain, Akshay Nambi, Tanuja Ganu, Sameer Segal, Mohamed Ahmed, Kalika Bali, and Sunayana Sitaram. 2023. MEGA: Multilingual evaluation of generative AI. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4232–4267, Singapore. Association for Computational Linguistics.

Mikel Artetxe, Vedanuj Goswami, Shruti Bhosale, Angela Fan, and Luke Zettlemoyer. 2023. Revisiting machine translation for cross-lingual classification. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6489–6499, Singapore. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2020. Translation artifacts in cross-lingual transfer learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7674–7684, Online. Association for Computational Linguistics.

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. Llm2vec: Large language models are secretly powerful text encoders. *arXiv preprint arXiv:2404.05961*.

Yang Chen, Chao Jiang, Alan Ritter, and Wei Xu. 2023. Frustratingly easy label projection for cross-lingual transfer. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5775–5796, Toronto, Canada. Association for Computational Linguistics.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470.

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.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *Preprint*, arXiv:1805.10190.

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.

Zi-Yi Dou and Graham Neubig. 2021. Word alignment by fine-tuning embeddings on parallel corpora. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2112–2128, Online. Association for Computational Linguistics.

David Dukić and Jan Snajder. 2024. Looking right is sometimes right: Investigating the capabilities of decoder-only LLMs for sequence labeling. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 14168–14181, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Benedikt Ebing and Goran Glavaš. 2024. To translate or not to translate: A systematic investigation of translation-based cross-lingual transfer to low-resource languages. 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 5325–5344, Mexico City, Mexico. Association for Computational Linguistics.

Abteen Ebrahimi, Manuel Mager, Arturo Oncevay, Vishrav Chaudhary, Luis Chiruzzo, Angela Fan, John Ortega, Ricardo Ramos, Annette Rios, Ivan Vladimir Meza Ruiz, Gustavo Giménez-Lugo, Elisabeth Mager, Graham Neubig, Alexis Palmer, Rolando Coto-Solano, Thang Vu, and Katharina Kann. 2022. AmericasNLI: Evaluating zero-shot natural language understanding of pretrained multilingual models in truly low-resource languages. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6279–6299, Dublin, Ireland. Association for Computational Linguistics.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2022. Language-agnostic BERT sentence embedding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 878–891, Dublin, Ireland. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, and et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

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

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.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. 2024. Qwen2.5-coder technical report. *Preprint*, arXiv:2409.12186.

Masoud Jalili Sabet, Philipp Dufter, François Yvon, and Hinrich Schütze. 2020. SimAlign: High quality word alignments without parallel training data using static and contextualized embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1627–1643, Online. Association for Computational Linguistics.

Anne Lauscher, Vinit Ravishankar, Ivan Vulić, and Goran Glavaš. 2020. From zero to hero: On the limitations of zero-shot language transfer with multilingual Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4483–4499, Online. Association for Computational Linguistics.

Duong Minh Le, Yang Chen, Alan Ritter, and Wei Xu. 2024. Constrained decoding for cross-lingual label projection. *Preprint*, arXiv:2402.03131.

Benjamin Minixhofer, Jonas Pfeiffer, and Ivan Vulić. 2023. Where's the point? self-supervised multilingual punctuation-agnostic sentence segmentation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7215–7235, Toronto, Canada. Association for Computational Linguistics.

Minh Van Nguyen, Viet Dac Lai, Amir Pouran Ben Veyseh, and Thien Huu Nguyen. 2021. Trankit: A lightweight transformer-based toolkit for multilingual natural language processing. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 80–90, Online. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Jaehoon Oh, Jongwoo Ko, and Se-Young Yun. 2022. Synergy with translation artifacts for training and inference in multilingual tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6747–6754, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Tanmay Parekh, I-Hung Hsu, Kuan-Hao Huang, Kai-Wei Chang, and Nanyun Peng. 2024. Contextual label projection for cross-lingual structured prediction. 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 5738–5757,

Mexico City, Mexico. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Evgeniia Razumovskaia, Ivan Vulić, and Anna Korhonen. 2023. Transfer-free data-efficient multilingual slot labeling. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6041–6055, Singapore. Association for Computational Linguistics.

Sebastian Ruder, Noah Constant, Jan Botha, Aditya Siddhant, Orhan Firat, Jinlan Fu, Pengfei Liu, Junjie Hu, Dan Garrette, Graham Neubig, and Melvin Johnson. 2021. XTREME-R: Towards more challenging and nuanced multilingual evaluation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10215–10245, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Fabian David Schmidt, Ivan Vulić, and Goran Glavaš. 2022. Don't stop fine-tuning: On training regimes for few-shot cross-lingual transfer with multilingual language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10725–10742, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Sebastian Schuster, Sonal Gupta, Rushin Shah, and Mike Lewis. 2019. Cross-lingual transfer learning for multilingual task oriented dialog. 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 3795–3805, Minneapolis, Minnesota. Association for Computational Linguistics.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, and et al. 2024. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. No language left behind: Scaling human-centered machine translation. *Preprint*, arXiv:2207.04672.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Rob van der Goot, Ibrahim Sharaf, Aizhan Imankulova, Ahmet Üstün, Marija Stepanović, Alan Ramponi, Siti Oryza Khairunnisa, Mamoru Komachi, and Barbara Plank. 2021. From masked language modeling to translation: Non-English auxiliary tasks improve zero-shot spoken language understanding. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2479–2497, Online. Association for Computational Linguistics.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11897–11916, Bangkok, Thailand. Association for Computational Linguistics.

Weikang Wang, Guanhua Chen, Hanqing Wang, Yue Han, and Yun Chen. 2022. Multilingual sentence transformer as a multilingual word aligner. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2952–2963, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zihan Wang, Stephen Mayhew, Dan Roth, et al. 2019. Cross-lingual ability of multilingual bert: An empirical study. *arXiv preprint arXiv:1912.07840*.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, pages 23965–23998. PMLR.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. 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 833–844, Hong Kong, China. Association for Computational Linguistics.

A Translation Data

For TyDiQa (Clark et al., 2020), we translate the questions and the context independently. To achieve higher translation quality, we split each context into sentences prior to translation and merge them back together afterwards. We utilize wtpsplit (Minixhofer et al., 2023) for sentence segmentation. For Masakha (Adelani et al., 2022) and xSID (van der Goot et al., 2021), we join the pre-tokenized input on white space before translation. We deviate for the Chinese data in xSID, where we merge neighboring Chinese tokens without white space. Additionally, the dialect *South Tyrol* (de-st) in xSID is not supported by NLLB. We translate the dialect pretending it to be German (i.e., using the German language code) as it is closely related to the latter. Further, the Serbian (sr) data in xSID is written in Latin script, whereas NLLB only supports the Cyrillic script. We accessed all datasets through the Hugging Face datasets library and ensured compliance with the licenses.

B Word Alignment

For our main experiments, we use the neural word aligner AccAlign (Wang et al., 2022), accessed through the following repository: https://github.com/sufenlp/AccAlign. Additionally, we employ Awesome (Dou and Neubig, 2021) with the code provided in the following repository: https://github.com/neulab/awesome-align. We follow the hyperparameter configuration proposed by the authors. We ensure compliance with the license for Awesome (BSD 3-Clause). We could not find licensing information for AccAlign.

C Easy

The code and data of `Easy` is released under the MIT license. We used the publicly released data for Masakha and TyDiQA. For xSID, we produced our own translated data by adopting the existing code. We followed their implementation for Masakha closely. `Easy` (Chen et al., 2023) requires fine-tuning NLLB on preserving inserted markers (i.e., preserving "[" and "]" around entity mentions). Hence, we leverage the publicly released 3.3B parameter checkpoint from Chen et al. (2023) for translation. We accessed it through the Hugging Face transformers library.

D Codec

The authors of Codec did not release the translated data but published the source code instead. We created our own translated data for Masakha following their implementation. Further, we extended their implementation to produce the translated data for xSID. We adhered to the hyperparameters in their repository and followed the existing implementation closely. We did not extend their work to TyDiQA. First, since TyDiQA does not follow the BIO scheme, adjusting the existing code base would have been an excessive effort. Second, we already observed lengthy decoding times for the constrained decoding step on Masakha. Considering that the input for TyDiQA is significantly longer, we did not further pursue an implementation for TyDiQA. The translations for Codec are obtained using standard (i.e., non fine-tuned) NLLB. However, the constrained decoding (i.e., inserting the markers post-translation) requires a fine-tuned NLLB that is able to preserve/insert markers. For constrained decoding, we follow Le et al. (2024) using the fine-tuned 600M parameter version of NLLB released by Chen et al. (2023). We could not find licensing information for Codec.

E Implementation Details: TyDiQa

Before translation, we preprocessed the TyDiQA dataset by removing duplicated whitespaces and ensuring that a whitespace follows every sentence boundary. We adjusted the start index of the answer spans accordingly. For the evaluation metric, we follow the F1 implementation used by the SQuAD (Rajpurkar et al., 2016) dataset. Before the metric computation, the script removes Latin punctuation, we extend it to also remove language-specific punctuation (e.g., for Arabic or Bengali). For Korean, we additionally remove particles from the answer spans. Particles are suffixes that follow a noun or pronoun, which are usually not part of the minimal answer span in TyDiQA. We also apply these steps to our baselines, ensuring a fair comparison. For downstream evaluation on the target languages, we use the publicly released validation sets as our test data and randomly sample 10% of instances from the target language training data as our new validation sets. Additionally, we feed the evaluation data pre-tokenized to foster consistent word alignments for `T-Test` and `ETT`.

	Masakha	xSID	TyDiQA
Task	NER	SL	QA
Epochs	10	10	3
Eff. Batch Size	32	32	16
Learning Rate	1e-5	1e-5	1e-5
Weight Decay	0.01	0.01	0.01

Table 6: Hyperparameters for downstream fine-tuning.

F Detailed Experimental Setup

Table 6 outlines the hyperparameters for downstream fine-tuning of our utilized tasks. Alongside, we implement a linear schedule of 10% warm-up and decay and employ mixed precision. In case we can not fit the desired batch size, we utilize gradient accumulation. For the LLM2Vec experiments, we deviate from this setting as we only fine-tune the classifier. Following BehnamGhader et al. (2024), we change the learning rate to 5e-4. We evaluate models at the last checkpoint of training. We use the seqeval F1 implementation for NER and SL, and the squad F1 implementation for QA, accessed through the Hugging Face evaluate library. Further, we access our downstream models—XLM-RoBERTa (`Large`), DeBERTa V3 (`Large`) and LLM2Vec Llama 3B Instruct MNTP—through the Hugginface transformers library. All translations were run on a single A100 with 40GB VRAM, and all downstream training and evaluation runs were completed on a single V100 with 32GB VRAM. We estimate that the GPU time accumulates to 4000 hours across all translations and downstream fine-tunings.

Languages.

MasakhaNER2.0. Our experiments cover the 18 out of 20 languages that are supported by NLLB: Bambara (bam), Ewé (ewe), Fon (fon), Hausa (hau), Igbo (ibo), Kinyarwanda (kin), Luganda (lug), Luo (luo), Mossi (most), Chichewa (nya), chiShona (sna), Kiswahili (saw), Setswana (tsn), Akan/Twi (twi), Wolof (wol), isiXhosa (xho), Yorùbá (yor), and isiZulu (zul).

xSID. We evaluate 11 languages all covered by NLLB: Arabic (ar), Danish (da), German (de), South-Tyrolean (de-st), Indonesian (id), Italian (it), Kazakh (kk), Dutch (nl), Serbian (sr), Turkish (tr), and Chinese (zh). Following Razumovskaia et al. (2023), we excluded Japanese from the evaluation because it only has half of the validation and test instances and spans only a fraction of entities compared to the other languages.

TydiQA-GoldP. Our evaluation spans the 8 languages included in TyDiQA: Arabic (ar), Bengali (bn), Finnish (fi), Indonesian (id), Korean (ko), Russian (ru), Swahili (sw), Telugu (te).

Filtering Strategy. We use a greedy approach to explore the various design options (§4.1). We start with the selection of the filtering strategy, followed by our pre-tokenization experiments. For the exploration of the filtering strategy, we apply whitespace tokenization and do not use `STR-MAT`.

Pre-Tokenization. For the per-tokenization experiments, we filter the translated training data based on `COM-INS`, `COM-SRC`, and `COM-TGT` for xSID and Masakha, and based on `COM-TGT` for TyDiQA. Additionally, we use `STR-MAT` on TyDiQA. For `T-Test`, we apply `COR-SCH` for Masakha and xSID, and `NO-FILT` for TyDiQA. Language-specific tokenization is done with the MosesTokenizer from the Sacremoses library (https://github.com/hplt-project/sacremoses) for Masakha and xSID, except for Chinese, where we use jieba (https://github.com/fxsjy/jieba). Both are released under the MIT license. For TyDiQA, we utilize trankit (Nguyen et al., 2021) to pre-tokenize the target language data and the MosesTokenizer for the source language data (i.e., English). As Bengali and Swahili are not supported by trankit, we fallback to whitespace pre-tokenization for these languages.

Main Results and Further Findings. As suggested by the findings of our preliminary experiments, we apply whitespace pre-tokenization for Masakha and xSID, except for Chinese, where we use language-specific tokenization. For TyDiQA, language-specific tokenization is applied to all languages for the translated training data (`T-Train`). For `T-Test`, we use language-specific tokenization for all tasks. We utilize the same filtering as for the pre-tokenization experiments.

G Detailed Results: Filtering Strategies

	bam	ewe	fon	hau	ibo	kin	lug	luo	mos	nya	sna	swa	tsn	twi	wol	xho	yor	zul	Avg
Translate-Train																			
NO-FILT	43.8	78.4	72.9	66.7	62.9	70.2	79.6	69.7	58.9	65.2	72.5	84.4	69.1	57.3	63.3	63.2	33.7	67.1	65.5
COM-INS	45.5	78.4	76.0	66.5	62.8	69.5	79.4	70.2	59.9	66.1	73.1	84.1	69.0	56.6	65.8	62.8	32.9	66.5	65.8
COM-INS + COM-TGT	50.9	77.8	76.0	66.5	62.5	69.0	80.4	70.4	60.4	65.4	73.0	83.6	68.9	55.9	67.0	63.1	33.2	64.7	66.0
COM-INS + COM-TGT + COM-SRC	51.2	78.8	76.5	66.5	63.7	69.7	80.0	70.0	60.3	66.2	73.1	83.6	68.2	60.8	67.3	63.4	32.6	66.3	66.6
STR-MAT + COM-INS + COM-TGT + COM-SRC	48.9	78.9	77.5	67.1	61.1	69.4	78.7	70.4	61.4	65.9	73.3	83.4	68.5	59.1	67.2	63.5	32.7	66.0	66.3
Translate-Test																			
NO-FILT	27.5	58.3	39.6	58.9	49.9	61.1	67.3	56.0	38.0	59.6	57.8	64.1	57.7	53.4	48.1	39.4	33.7	51.5	51.2
COR-SCH	37.9	69.0	58.4	61.9	57.4	67.2	71.0	62.0	46.6	64.8	61.0	67.6	63.3	61.9	55.9	43.3	43.1	56.3	58.2
COM-TGT	34.6	62.6	50.7	60.4	51.6	63.0	68.8	59.4	43.2	61.1	58.1	65.0	60.8	59.4	52.1	40.3	40.3	52.2	54.6
COM-SRC	29.5	57.4	40.6	59.7	50.1	60.9	67.6	57.1	36.6	59.8	57.9	63.5	57.1	55.0	49.4	39.4	33.6	51.7	51.5
COR-SCH + COM-TGT	37.0	67.8	52.8	63.1	57.7	66.9	71.7	62.8	45.3	65.1	61.1	68.2	64.3	61.4	57.3	43.7	43.5	56.4	58.1
COR-SCH + COM-TGT + COM-SRC	39.3	66.7	53.6	63.7	58.0	66.8	71.8	63.5	43.7	65.2	61.2	67.4	63.4	63.0	58.0	43.7	43.5	56.6	58.3

Table 7: Results for translation-based XLT evaluated on the Masakha validation data utilizing different filtering strategies. `NO-FILT` indicates that no filtering strategy was applied. We use XLM-R.

	ar	da	de	de-st	id	it	kk	nl	sr	tr	zh	Avg
Translate-Train												
NO-FILT	85.4	81.8	88.3	60.3	86.2	89.8	70.5	94.1	85.7	85.9	-	82.8
COM-INS	85.1	82.4	88.7	58.3	86.0	90.1	70.7	93.3	84.6	88.0	-	82.7
COM-INS + COM-TGT	84.9	81.9	87.8	59.3	85.3	86.7	69.8	90.9	86.6	86.6	-	82.0
COM-INS + COM-TGT + COM-SRC	86.3	81.3	86.7	58.8	85.0	88.1	69.6	91.3	86.2	86.5	-	82.0
STR-MAT + COM-INS + COM-TGT + COM-SRC	85.6	81.4	87.6	57.0	85.3	87.6	69.7	91.4	85.3	86.2	-	81.7
Translate-Test												
NO-FILT	69.8	74.7	75.0	54.4	68.0	80.1	49.8	82.4	72.3	61.1	58.7	67.8
COR-SCH	73.7	78.2	82.6	57.4	73.6	83.4	61.7	86.5	75.9	75.3	75.3	74.9
COM-TGT	70.7	75.0	80.4	56.4	69.4	79.3	59.8	82.0	73.2	73.4	74.6	72.2
COM-SRC	68.9	73.5	74.1	54.3	66.4	79.1	48.6	81.1	70.7	59.8	58.2	66.8
COR-SCH + COM-TGT	74.1	77.9	81.7	57.7	74.1	82.2	60.0	84.7	76.3	74.0	74.8	74.3
COR-SCH + COM-TGT + COM-SRC	73.3	76.8	81.0	57.4	72.5	81.6	59.2	83.9	74.6	71.9	73.7	73.3

Table 8: Results for translation-based XLT evaluated on the xSID validation data utilizing different filtering strategies. `NO-FILT` indicates that no filtering strategy was applied. We use XLM-R. For `T-Train`, we excluded Chinese (zh) since experiments were run with whitespace pre-tokenization (`WS-TOK`).

	ar	bn	fi	id	ko	ru	sw	te	Avg
Translate-Train									
NO-FILT	71.1	64.4	73.8	78.4	54.4	67.1	70.7	64.6	68.1
COM-TGT	70.1	68.9	72.9	78.0	58.2	66.6	70.6	68.2	69.2
COM-TGT + COM-SRC	68.1	69.4	71.0	76.7	57.2	66.3	68.8	68.9	68.3
STR-MAT + COM-TGT	69.6	69.2	73.0	78.4	56.1	66.8	70.8	69.8	69.2
Translate-Test									
NO-FILT	62.1	50.7	62.6	69.7	39.4	59.8	63.2	51.0	57.3
COM-SRC	51.5	51.4	53.3	63.8	39.2	57.6	61.8	53.0	53.9
COM-TGT	54.2	49.7	55.2	63.8	39.2	57.6	61.8	53.0	54.3
COM-TGT + COM-SRC	47.5	48.9	51.8	57.0	38.8	55.3	59.4	51.9	51.3

Table 9: Results for translation-based XLT evaluated on the TyDiQA validation data utilizing different filtering strategies. `NO-FILT` indicates that no filtering strategy was applied. We use XLM-R.

H Detailed Results: Pre-Tokenization

	bam	ewe	fon	hau	ibo	kin	lug	luo	mos	nya	sna	swa	tsn	twi	wol	xho	yor	zul	Avg
Translate-Train																			
WS-TOK	51.2	78.8	76.5	66.5	63.7	69.7	80.0	70.0	60.3	66.2	73.1	83.6	68.2	60.8	67.3	63.4	32.6	66.3	66.6
SP-TOK	49.4	77.9	75.3	66.7	58.4	68.8	80.2	70.7	61.3	65.9	73.7	83.4	69.8	59.2	66.9	63.8	32.3	66.8	66.1
Translate-Test																			
WS-TOK	37.9	69.0	58.4	61.9	57.4	67.2	71.0	62.0	46.6	64.8	61.0	67.6	63.3	61.9	55.9	43.3	43.1	56.3	58.2
SP-TOK	51.3	82.7	75.0	67.0	71.8	78.9	86.9	76.6	55.4	76.0	82.2	82.0	76.3	69.1	66.1	63.0	54.6	69.9	71.4

Table 10: Results for translation-based XLT evaluated on the Masakha validation data utilizing different pre-tokenization strategies. We use XLM-R.

	ar	da	de	de-st	id	it	kk	nl	sr	tr	zh	Avg
Translate-Train												
WS-TOK	86.3	81.3	86.7	58.8	85.0	88.1	69.6	91.3	86.2	86.5	-	82.0
SP-TOK	86.4	81.4	87.8	57.4	85.8	87.3	69.7	90.8	83.8	86.3	86.2*	81.6
Translate-Test												
WS-TOK	73.7	78.2	82.6	57.4	73.6	83.4	61.7	86.5	75.9	75.3	75.3	74.9
SP-TOK	79.0	81.4	89.6	62.9	78.6	89.6	69.8	92.8	80.4	81.6	82.8	80.8

Table 11: Results for translation-based XLT evaluated on the xSID validation data utilizing different pre-tokenization strategies. We use XLM-R. Results marked with * are excluded from the average.

	ar	bn	fi	id	ko	ru	sw	te	Avg
Translate-Train									
WS-TOK	69.6	69.2	73.0	78.4	56.1	66.8	70.8	69.8	69.2
SP-TOK	70.0	69.2	73.9	77.0	57.3	68.2	70.8	69.9	69.5
Translate-Test									
WS-TOK	62.1	50.7	62.6	69.7	39.4	59.8	63.2	51.0	57.3
SP-TOK	64.6	50.6	65.6	72.1	47.6	62.8	63.1	57.4	60.5

Table 12: Results for translation-based XLT evaluated on the TyDiQA validation data utilizing different pre-tokenization strategies. We use XLM-R.

I Detailed Results: Main Results and Further Findings

				bam	ewe	fon	hau	ibo	kin	lug	luo	mos	nya	sna	swa	tsn	twi	wol	xho	yor	zul	Avg
ZS				43.4	72.8	61.0	73.5	49.9	46.3	64.9	55.0	56.1	51.1	34.4	88.1	51.5	49.5	56.2	22.2	35.1	41.5	52.9
Translate-Train																						
AccAlign	X	WS-TOK	NLLB	49.2	74.1	72.1	73.1	72.2	58.6	76.4	63.5	58.2	66.2	70.9	83.2	76.5	64.1	63.9	69.6	40.2	75.6	67.1
AccAlign†	X	WS-TOK	NLLB	53.3	74.0	71.3	73.6	71.0	58.8	75.5	64.2	55.6	67.6	68.7	83.6	76.1	64.5	61.9	69.4	39.3	72.6	66.7
Awesome†	X	WS-TOK	NLLB	51.3	73.8	65.6	73.6	70.0	56.7	74.4	64.6	50.8	67.3	68.4	82.2	75.3	62.4	58.9	61.0	38.4	64.7	64.4
Easy	X	-	NLLB	54.2	75.4	71.1	73.0	64.6	66.3	77.5	63.8	51.3	68.3	57.2	84.1	74.7	63.7	63.3	71.3	37.0	70.6	66.0
Codec	X	-	NLLB	51.2	74.1	68.9	73.4	65.5	64.7	75.4	64.7	53.9	68.3	70.9	84.2	73.5	65.2	65.6	70.2	39.4	75.3	66.9
Translate-Test																						
AccAlign	X	SP-TOK	NLLB	55.0	79.2	72.6	74.1	74.3	70.5	84.0	73.4	52.9	78.8	81.3	83.2	79.4	70.3	66.2	73.0	58.0	78.7	72.5
AccAlign†	X	SP-TOK	NLLB	54.3	76.4	69.2	73.5	72.7	69.7	82.9	71.4	48.3	77.8	80.1	81.8	79.6	70.8	63.0	71.3	49.5	77.3	70.5
Awesome†	X	SP-TOK	NLLB	46.5	72.5	58.8	69.5	75.6	65.0	81.7	72.3	42.6	78.6	66.4	80.0	78.5	69.2	53.6	52.3	47.8	64.5	65.3
AccAlign	D	SP-TOK	NLLB	54.7	79.4	73.5	74.6	75.5	71.3	84.0	75.0	52.7	79.4	81.8	83.8	79.6	70.6	66.4	73.2	57.6	78.7	72.9
AccAlign	X	WS-TOK	NLLB	45.3	69.7	55.9	63.4	60.6	58.6	72.9	60.4	41.0	68.3	60.9	68.3	63.9	61.2	55.4	62.0	46.4	62.3	59.8
AccAlign	D	WS-TOK	NLLB	45.0	69.9	56.3	63.9	61.5	59.1	72.6	61.8	40.8	68.9	61.5	68.9	64.4	61.5	56.0	61.9	46.3	62.0	60.1
AccAlign	X	SP-TOK	GT	61.0	79.3	-	73.4	78.1	71.8	-	-	-	-	83.5	85.2	-	71.6	-	75.1	63.7	78.5	-
AccAlign	D	SP-TOK	GT	60.4	79.6	-	74.1	79.2	72.4	-	-	-	-	84.0	85.9	-	73.2	-	75.2	62.3	78.8	-
AccAlign	ShL	SP-TOK	NLLB	50.0	66.9	62.6	61.9	59.1	58.7	67.1	61.6	43.5	70.7	64.4	71.8	70.0	59.4	57.3	59.4	44.9	60.8	60.6
AccAlign	L	SP-TOK	NLLB	47.8	69.2	62.0	63.5	59.4	58.0	70.3	62.3	44.3	70.1	68.1	73.6	69.9	58.8	57.6	59.8	46.4	66.3	61.5
Codec	X	-	NLLB	54.5	78.8	67.4	72.9	72.8	77.6	83.6	72.8	49.4	78.1	79.3	82.2	79.2	72.5	67.3	72.5	58.4	77.1	72.0
Codec	D	-	NLLB	54.3	79.1	68.0	73.3	73.9	78.2	83.5	74.2	48.8	79.0	79.8	82.9	79.3	73.1	67.8	72.6	58.0	77.0	72.4
Ensembling-Translate-Train																						
AccAlign + AccAlign	X/X	WS-TOK/WS-TOK	NLLB	57.3	79.1	74.2	72.6	77.7	63.5	81.8	69.7	59.6	75.0	75.6	83.7	78.5	66.3	67.6	72.1	52.7	78.7	71.4
AccAlign + AccAlign	X/X	WS-TOK/SP-TOK	NLLB	57.3	78.5	75.6	72.8	79.0	64.1	82.5	70.3	60.2	75.3	77.1	84.2	78.9	66.8	68.0	72.4	53.5	78.8	72.0
AccAlign + AccAlign	X/D	WS-TOK/WS-TOK	NLLB	57.3	79.5	75.3	73.3	79.1	64.2	81.7	71.4	59.3	75.6	76.4	83.9	79.3	67.5	68.4	72.8	53.0	79.1	72.0
AccAlign + AccAlign	X/D	WS-TOK/SP-TOK	NLLB	57.1	79.1	76.4	73.5	80.6	64.7	82.7	71.9	60.2	76.0	77.5	84.2	79.9	68.3	68.8	73.0	53.9	79.0	72.6
AccAlign + AccAlign	X/D	WS-TOK/SP-TOK	GT	61.0	79.0	-	73.7	81.1	65.0	-	-	-	-	78.5	85.0	-	67.8	-	73.9	58.2	80.0	-
AccAlign + AccAlign	X/ShL	WS-TOK/SP-TOK	NLLB	55.6	76.5	73.4	73.0	75.7	60.5	78.9	66.4	58.6	71.3	73.1	83.3	78.1	64.3	66.5	70.5	46.3	76.3	69.3
AccAlign + AccAlign	X/L	WS-TOK/SP-TOK	NLLB	54.1	77.2	74.3	72.9	74.4	60.5	80.1	67.2	58.4	72.2	74.1	83.1	78.5	64.1	67.2	70.7	47.3	77.9	69.7
Easy + AccAlign	X/D	-/SP-TOK	NLLB	58.1	79.2	74.1	73.3	76.4	64.5	83.2	71.1	56.3	77.8	72.6	85.2	78.7	68.6	68.3	73.7	53.7	76.3	71.7
Codec + AccAlign	X/D	-/SP-TOK	NLLB	55.5	79.2	75.0	73.5	77.7	63.5	82.4	72.2	57.8	77.4	77.6	85.0	78.0	69.3	69.4	74.1	54.3	79.1	72.3

Table 13: Main results for translation-based XLT evaluated on Masakha using different WAs, pre-tokenizations, and MT models. We use XLM-R (X), DeBERTa (D), LLM2Vec Sheared-Llama 1.3B (ShL), and LLM2Vec LLama 3 8B (L). WAs marked with † are not fine-tuned. Results for languages indicated with - are not supported by the MT model.

				ar	da	de	de-st	id	it	ja	kk	nl	sr	tr	zh	Avg
ZS				71.5	85.6	80.8	43.9	86.8	88.2	53.9	80.8	88.8	79.0	81.5	57.4	76.8
Translate-Train																
AccAlign	X	WS-TOK	NLLB	82.6	76.0	86.1	62.2	87.4	88.1	68.1	86.0	85.5	85.0	85.3	85.1	82.7
AccAlign†	X	WS-TOK	NLLB	81.8	76.4	87.7	63.8	82.9	87.8	57.1	85.7	85.4	86.6	87.2	86.7	82.9
Awesome†	X	WS-TOK	NLLB	79.1	77.1	85.6	61.3	82.7	87.3	48.8	74.3	85.8	84.5	77.7	82.4	79.8
Easy	X	-	NLLB	83.0	84.0	89.4	62.2	86.3	87.5	26.9	89.2	88.3	81.4	86.3	80.5	83.4
Codec	X	-	NLLB	81.9	84.6	88.7	62.5	89.8	88.5	51.9	85.1	89.9	82.7	81.2	84.4	83.6
Translate-Test																
AccAlign	X	SP-TOK	NLLB	78.8	76.0	86.4	61.1	78.9	88.0	43.7	82.5	87.4	79.7	81.3	82.3	80.2
AccAlign†	X	SP-TOK	NLLB	77.9	75.4	84.9	59.8	79.4	85.7	40.3	82.2	86.7	79.9	82.0	82.5	79.7
Awesome†	X	SP-TOK	NLLB	73.8	75.0	84.8	59.4	71.0	84.7	35.3	63.1	87.1	77.0	70.9	76.4	74.8
AccAlign	D	SP-TOK	NLLB	79.3	75.8	85.7	59.4	80.1	88.6	43.2	82.6	86.7	80.1	82.1	82.1	80.2
AccAlign	X	WS-TOK	NLLB	45.3	69.7	55.9	63.4	60.6	58.6	72.9	60.4	41.0	68.3	60.9	68.3	59.8
AccAlign	D	WS-TOK	NLLB	73.0	70.5	79.2	56.0	72.2	81.3	37.6	74.4	82.0	73.9	72.9	72.7	73.5
AccAlign	X	SP-TOK	GT	80.6	76.4	86.5	61.9	78.6	88.2	40.0	83.8	87.0	82.0	81.6	86.0	81.1
AccAlign	D	SP-TOK	GT	80.5	76.4	85.5	59.6	79.5	89.3	39.6	84.0	87.0	83.3	82.4	85.6	81.2
AccAlign	ShL	SP-TOK	NLLB	68.4	68.3	76.8	51.6	68.6	77.7	41.1	72.2	78.1	70.7	72.2	73.3	70.7
AccAlign	L	SP-TOK	NLLB	68.9	69.6	77.1	52.3	70.2	77.8	40.9	71.8	79.0	71.5	71.5	74.0	71.2
Codec	X	-	NLLB	79.0	81.9	86.1	60.4	84.8	88.4	68.1	83.0	86.5	72.4	83.6	67.0	79.4
Codec	D	-	NLLB	79.9	81.8	85.5	58.8	85.8	89.0	67.3	83.2	86.0	72.9	84.2	67.5	79.5
Ensembling-Translate-Train																
AccAlign + AccAlign	X/X	WS-TOK/WS-TOK	NLLB	80.9	75.1	85.9	63.1	88.1	89.2	59.4	81.4	86.3	82.7	81.5	80.5	81.3
AccAlign + AccAlign	X/X	WS-TOK/SP-TOK	NLLB	82.0	76.3	88.2	64.6	88.7	89.6	63.7	86.5	86.4	83.0	85.0	85.0	83.2
AccAlign + AccAlign	X/D	WS-TOK/WS-TOK	NLLB	81.2	75.0	85.9	61.8	88.1	89.9	58.5	81.1	86.6	82.6	81.7	79.7	81.2
AccAlign + AccAlign	X/D	WS-TOK/SP-TOK	NLLB	82.4	76.2	88.2	64.3	89.0	90.0	62.6	86.6	87.3	83.1	85.3	84.8	83.4
AccAlign + AccAlign	X/D	WS-TOK/SP-TOK	GT	82.4	75.8	87.2	64.4	87.7	89.8	61.7	86.5	86.3	84.4	84.2	86.4	83.2
AccAlign + AccAlign	X/ShL	WS-TOK/SP-TOK	NLLB	81.2	76.2	87.4	63.6	87.7	89.6	65.5	86.0	86.1	83.7	84.7	84.5	82.8
AccAlign + AccAlign	X/L	WS-TOK/SP-TOK	NLLB	80.0	76.4	87.1	62.4	86.2	88.7	63.2	85.5	86.1	83.3	83.5	84.2	82.1
Easy + AccAlign	X/D	-/SP-TOK	NLLB	81.1	81.2	89.7	64.4	89.0	90.6	45.6	85.7	90.4	82.7	82.4	84.5	83.8
Codec + AccAlign	X/D	-/SP-TOK	NLLB	82.5	75.3	89.1	62.8	87.6	88.7	28.0	87.1	89.2	81.8	85.3	81.3	82.8

Table 14: Main results for translation-based XLT evaluated on xSID using different WAs, pre-tokenizations, and MT models. We use XLM-R (X), DeBERTa (D), LLM2Vec Sheared-Llama 1.3B (ShL), and LLM2Vec LLama 3 8B (L). WAs marked with † are not fine-tuned. For `T-Train`, we pre-tokenize Chinese (zh) with `SP-TOK`.

				ar	bn	fi	id	ko	ru	sw	te	Avg
ZS				77.1	70.4	75.3	79.8	68.0	70.1	72.5	78.6	74.0
Translate-Train												
AccAlign	X	SP-TOK	NLLB	76.6	72.0	76.9	81.1	68.4	73.3	74.4	79.0	75.2
AccAlign†	X	SP-TOK	NLLB	76.8	73.0	76.0	80.8	67.4	72.4	74.8	79.1	75.0
Awesome†	X	SP-TOK	NLLB	74.7	71.8	75.6	79.5	64.3	73.1	73.7	78.2	73.8
Easy	X	-	NLLB	76.6	74.8	76.0	79.8	72.0	71.9	74.8	78.4	75.5
Translate-Test												
AccAlign	X	SP-TOK	NLLB	70.5	55.9	68.3	74.6	48.5	69.0	65.2	58.4	63.8
AccAlign†	X	SP-TOK	NLLB	67.6	53.9	67.8	73.8	44.3	67.6	64.6	57.1	62.1
Awesome†	X	SP-TOK	NLLB	69.5	54.9	68.2	73.4	45.4	68.6	64.5	57.6	62.8
AccAlign	D	SP-TOK	NLLB	72.2	59.8	71.1	77.5	49.9	72.5	72.8	65.1	67.6
AccAlign	X	WS-TOK	NLLB	68.1	55.9	65.4	72.2	34.9	64.8	65.3	51.7	59.8
AccAlign	D	WS-TOK	NLLB	69.6	59.8	67.7	76.1	37.8	68.1	72.8	60.2	64.0
AccAlign	X	SP-TOK	GT	74.3	55.7	69.9	75.2	52.9	69.4	68.2	69.1	66.8
AccAlign	D	SP-TOK	GT	76.7	59.4	73.1	78.3	53.5	72.9	76.5	70.5	70.1
Ensembling-Translate-Train												
AccAlign	X/X	SP-TOK/WS-TOK	NLLB	76.3	71.4	76.7	80.6	65.1	72.2	74.1	78.0	74.3
AccAlign	X/X	SP-TOK/SP-TOK	NLLB	76.8	71.6	75.6	79.9	68.8	74.3	74.1	78.5	74.9
AccAlign	X/D	SP-TOK/WS-TOK	NLLB	76.6	73.8	77.9	82.3	64.7	74.5	77.2	79.2	75.8
AccAlign	X/D	SP-TOK/SP-TOK	NLLB	78.3	73.8	77.4	82.6	65.4	75.9	77.2	78.9	76.2
Easy + AccAlign	X/D	-/SP-TOK	NLLB	78.9	75.0	77.9	81.5	66.2	75.8	76.6	78.5	76.3

Table 15: Main results for translation-based XLT evaluated on TyDiQA using different WAs, pre-tokenizations, and MT models. We use XLM-R (X) and DeBERTa (D). WAs marked with † are not fine-tuned.