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, the low-level design decisions for applying them to translation-based XLT have not been systematically investigated. Moreover, recent marker-based methods, which project labeled spans by inserting tags around them before (or after) translation, claim to outperform WAs in label projection for XLT. In this work, we revisit WAs for label projection, systematically investigating the effects of low-level design decisions on token-level XLT: (i) the algorithm for projecting labels between (multi-)token spans, (ii) filtering strategies to reduce the number of noisily mapped labels, and (iii) the pre-tokenization of the translated sentences. We find that all of these substantially impact translation-based XLT performance and show that, with optimized choices, XLT with WA offers performance at least comparable to that of marker-based methods. We then introduce a new projection strategy that ensembles *translate-train* and *translate-test* predictions and demonstrate that it substantially outperforms the marker-based projection. Crucially, we show that our proposed 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 mLMs (Ahuja et al., 2023; Le et al., 2024; Parekh et al., 2024).

Much of the above work highlights translation-based strategies as competitive approaches for XLT, where a machine translation (MT) model is used to either (1) generate noisy target language data by translating the original source language data before training, known as *translate-train* (`T-Train`), or (2) translate original target language instances into the (noisy) source language before inference, known as *translate-test* (`T-Test`) (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 se-

quence. 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 exact algorithm for projecting the labels, (ii) filtering techniques to reduce the number of noisily mapped labels, and (iii) the process of inducing word boundaries (i.e. pre-tokenization) to the translated sentence before it can be aligned to the tokens in the original 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 labeled spans 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). This line of work explicitly evaluates token-level translation-based XLT, demonstrating strong performance for both `T-Train` and `T-Test`. Furthermore, it renders WA-based XLT for token classification tasks inferior (Chen et al., 2023; Le et al., 2024). While these efforts provide the technical details for their proposed marker-based methods, they do not lay out the low-level design details for label projection with WAs. Therefore, we argue that they possibly underestimate 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 covering (i) the exact algorithm for mapping the labels between (multi-)token spans from the source sentence to the target sentence based on word alignments; (ii) filtering strategies to identify incomplete labeled span alignments, and (iii) the process of inducing word boundaries (i.e. pre-tokenization) to the translated sentence required to align the tokens to their counterparts in the original sentence. We demonstrate that these design choices can substantially impact translation-based XLT performance for token-level tasks. For example, using a language-specific pre-tokenizer instead of simple whitespace pre-tokenization improves the performance of `T-Test` by up to 12.6%. Overall, we find `T-Test` to be more sensitive to low-level design decisions of WA than `T-Train`. **(2)** We then extensively compare WA-based label projection—with the identified well-performing low-level design choices—against state-of-the-art marker-based label projection methods in token-level XLT on two established benchmarks encompassing 29 diverse languages. Contrary to prior claims (Chen et al., 2023; Le et al., 2024; Parekh et al., 2024), we find that optimized 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 probability distributions over the labels produced by `T-Train` and `T-Test`. Our 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 MT models (i.e., translation quality), WA models, and base LMs (i.e., encoder vs. decoder models).

2 Token-Level XLT via Word Alignment

We first detail our pipeline for label projection with WA focusing on the low-level design choices we investigate: mapping labeled spans, filtering strategies, and pre-tokenization. We then describe our new translation-based XLT approach for token-level tasks that ensembles `T-Train` and `T-Test`.

2.1 Label Projection with WA: Design Choices

Translation-based XLT entails two paradigms: (1) creating noisy target language training data by translating it from the original source language instances (`T-Train`) and (2) running inference on noisy source language instances translated from the original target language data (`T-Test`). For token-level tasks, label projection is required for both. We illustrate our pipeline for label projection with WA in Figure 1, comprising the following steps:

1. Translating and Pre-Tokenizing (§2.1.3). We start from an original source language training in-

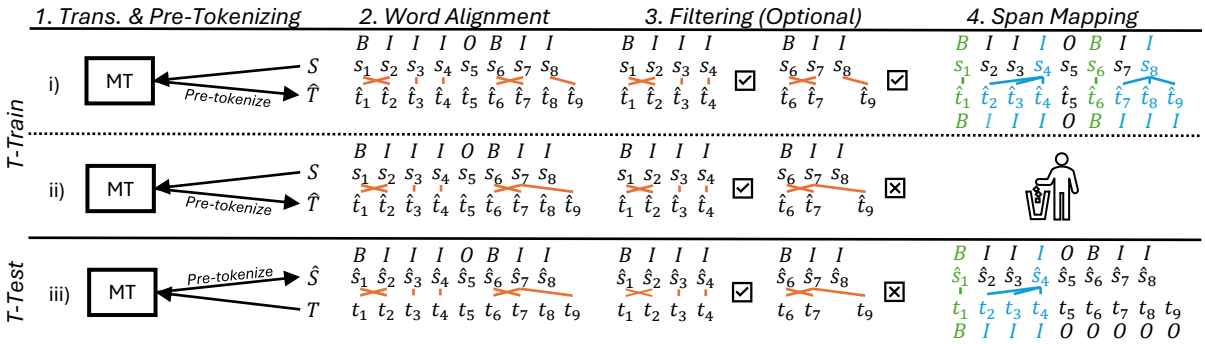

Figure 1: Schematic overview of our pipeline for label projection using word aligners. From top to bottom, we display three cases—two for T-Train and one for T-Test: (i) a successful creation of a translated target language training instance $\hat{T}$ from the original source language training instance S, (ii) a discarded translated target language training instance $\hat{T}$ due to incomplete alignments detected by our filtering strategies, and (iii) a partial label projection from a translated source language test instance $\hat{S}$ to the original target language test instance T due to incomplete alignments detected by our filtering strategies. All cases exemplarily apply the *Complete Source* filter.

stance S in T-Train (or an original target language test instance T in T-Test). Next, we translate the instance to the target language for T-Train (or source language for T-Test). We then pre-tokenize the translated sentence $\hat{T}$ (or $\hat{S}$) for word alignment (i.e., we induce word boundaries).

2. Word Alignment. We produce the word alignments between S and $\hat{T}$ for T-Train (or $\hat{S}$ and T for T-Test). In the case of T-Test, we first gather the label predictions of the mLM on $\hat{S}$.

3. Filtering (§2.1.2). Based on the word alignment, we collect the labeled spans of tokens that need to be mapped from the source to the target sentence. Optionally, we apply our proposed filtering strategies to identify incomplete alignments between labeled spans of S and the corresponding tokens in $\hat{T}$ for T-Train ($\hat{S}$ and T for T-Test). For T-Train, if our filters are met for any labeled span, we discard the training instance $\hat{T}$ (case two in Figure 1). For T-Test, we modify the procedure as we cannot discard test instances at inference time. Instead, for an incomplete labeled span, we do not map the corresponding labels from $\hat{S}$ to T and assign the default label "O" (case three in Figure 1).

4. Span Mapping (§2.1.1). Last, we apply our span mapping algorithm to project the labeled spans from S to $\hat{T}$ for T-Train (or $\hat{S}$ to T for T-Test).

2.1.1 Span Mapping

Translation-based XLT for token-level tasks requires mapping labeled spans of tokens from the source language to the target language sentence. Instead of simply projecting the token-level labels based on the word alignments produced in step 2 of our pipeline, we propose a more robust, span-based label projection, as illustrated in step 4 of Figure 1.

For T-Train, given a labeled span L in the source sentence S (e.g., $L = \{s_1, \ldots, s_4\}$ in Figure 1) and the corresponding candidate labeled span $\hat{L}$ in the translated sentence $\hat{T}$ (e.g., $\hat{L} = \{\hat{t}_1, \ldots, \hat{t}_4\}$ in Figure 1) , we map the labels as follows. We assume a standard BIO scheme in which the first token in a span is labeled with a different tag (B-Tag) than the remaining span (I-Tag). We then map the label of the first token in L to the first token in $\hat{L}$. Next, we map the label of the last token in L to all tokens in $\hat{T}$ between the first and the last token of $\hat{L}$ (inclusively). The procedure ensures that the mapped labeled spans in $\hat{T}$ are continuous and comply with the BIO scheme.

The span mapping for T-Test follows the same procedure. However, instead of mapping the gold labels, from the original source sentence S to the translated target sentence $\hat{T}$, we now map the predictions of the mLM from the translated source sentence $\hat{S}$ to the original target sentence T.

We also experimented with the simple approach of naively projecting the labels directly based on the word alignments as produced in step 2. 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.

2.1.2 Filtering Strategies

The success of the above span mapping directly depends on the quality of WA for a concrete lan-

guage 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 filtering alignments of labeled spans with low quality.

Complete Source (`COMP-SRC`*).* We test if all tokens of a span $L = \{s_m, ..., s_n\}$ in the source language sentence—S for `T-Train` and $\hat{S}$ for `T-Test`—have an alignment, i.e., whether the corresponding tokens in S (or $\hat{S}$) are aligned to at least one token in $\hat{T}$ (or T). Figure 1 illustrates an example of an incomplete source alignment: in the second and third cases, the token s_8 in S (or $\hat{s_8}$ in $\hat{S}$) is not aligned to any token in $\hat{T}$ (or T). We assume that if L is partially unaligned, we miss information from the source language span. Hence, $\hat{L}$ is more likely to be incomplete and thus incorrect.

Complete Target (`COMP-TGT`*).* The motivation for this filter is analogous to `COMP-SRC`: we select only the labeled span alignments for which all span tokens in the target language sentence—$\hat{T}$ for `T-Train` and T for `T-Test`—are aligned to at least one token in the source language sentence. But since we do not have ground truth spans for the target language sentence, we apply the following proxy: we retain only the instances for which the tokens in $\hat{L}$ form a continuous span. Case two and three in Figure 1 do not satisfy this filter, since $\{\hat{t}_6, \hat{t}_7, \hat{t}_9\}$ (or $\{t_6, t_7, t_9\}$) are discontinuous. We assume that if $\hat{L}$ is partially unaligned, we risk adding additional information to the labeled span in the target language sentence that did not exist in the source language (e.g., by including a target language token distant from the rest of the span).

Complete Instance (`COMP-INS`*).* Following Chen et al. (2023), we verify that the number and type of labeled spans in S and $\hat{T}$ match (e.g., if S has two spans with label *LOC* and one with label *PER* then $\hat{T}$ must also have two *LOC* spans and one *PER* span). This filter can only be applied to `T-Train` as it needs access to the gold labels of S.

Restricted Target (`RSTR-TGT`*).* This filter is specifically designed for `T-Test`. Preliminary experiments showed that single token labeled spans in $\hat{S}$ are often aligned to multiple discontinuous target tokens in T that have a comparatively large number of unaligned tokens in between. On the one hand, applying our span mapping algorithm without filtering is too coarse-grained—such that it might map a single token labeled span from $\hat{S}$ to a much longer sequence in T—on the other hand, applying `COMP-TGT` is too strict—such that it does not map the single token label span from $\hat{S}$ at all. In summary, the former reduces precision while the latter degrades recall. `RSTR-TGT` aims at the trade-off between no filtering and `COMP-TGT`. It maps the single token label span from $\hat{S}$ only to the first token in $\hat{L}$ and assigns the default label "O" to all other tokens. This way, we still map the single token label spans from $\hat{S}$ but limit their length to a single token in T.

No Filtering (`NO-FILT`*).* We additionally compare our filtering strategies against a naive baseline: we omit the filtering step in our label projection pipeline and immediately apply our span mapping algorithm after we obtain the word alignments.

2.1.3 Pre-Tokenization

Word alignment for token-level tasks requires the original sentence and the translation to contain word boundaries (i.e., to be pre-tokenized). While the original sentence S (or T) is usually given in a pre-tokenized format, we still need to pre-tokenize the translation $\hat{T}$ (or $\hat{S}$). We compare language-agnostic whitespace pre-tokenization (`WS-TOK`) against language-specific pre-tokenization (`LS-TOK`).[1] It is worth noting that it is more challenging to pre-tokenize the translated target language sentences $\hat{T}$ in `T-Train` than the English translations $\hat{S}$ in `T-Test`.

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

We next detail our proposed translation-based strategy for token-level XLT that ensembles the predictions of `T-Train` and `T-Test` (Oh et al., 2022). At inference time, the `T-Train` model produces class logits for each token t_i in the target language sentence T. In contrast, the `T-Test` model outputs class logits over each token $\hat{s}_j$ in the translated source language sentence $\hat{S}$. We then run our WA-based pipeline for label projection to map the `T-Test` predictions from $\hat{S}$ to T. Instead of mapping the predicted class labels for each token, we now map the class logits. Finally, we average the class logits for each token in T obtained from

[1] We provide the details on the tokenizers in App. E.

T-Train and T-Test. If we cannot project the logits for a token in T-Test—because of a missing alignment or because a filter prevents the mapping of a labeled span—we only use the T-Train prediction. Importantly, our proposed ensembling is not restricted to a WA-based T-Train component and can also be combined with marker-based methods.

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 two established token classification tasks: named entity recognition and slot labeling. Our experiments span 29 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. We use English as our source language.[2]

Named Entity Recognition (NER). Our evaluation includes 18 of 20 languages from MasakhaNER 2.0 (Masakha) (Adelani et al., 2022) supported by the NLLB model used for translation. Masakha consists of underserved languages from 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 softmax classifier on top of the mLM to predict the class for each token.

Slot Labeling (SL). We use the xSID dataset (van der Goot et al., 2021), which covers 11 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 add a softmax classifier on top of the mLM.

Label Projection. We compare our WA-based label projection approach 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 WA that is based on the multilingual sentence encoder LaBSE (Feng et al., 2022).

EasyProject (Easy). We compare our WA-based approach against the marker-based label projection method of Chen et al. (2023). Before translation, Easy inserts tags ("[", "]") around labeled spans. The MT model is expected to preserve the tags during translation, enabling the reconstruction of the labels afterward. 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 $\hat{S}$; Easy would then insert markers into $\hat{S}$ and back-translate to the target language, obtaining $\hat{\hat{T}}$; but $\hat{\hat{T}}$ will generally differ from T, which is the actual sentence we need to label.

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 the second step, tags are inserted around the labeled spans in the source sentence (English: "This is [New York]"). The marked sentence is fed again as input to the MT model: during decoding, the MT model is now constrained 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 encoder-turned-decoder Llama 3 8B (LLM2Vec) (BehnamGhader et al., 2024) as English-centric models. In T-Train, we fine-tune on both the original 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 original 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 Appendix E.

4 Results and Discussion

Starting from the validation portions of our datasets, we assess the impact of low-level design choices related to using WA for token-level translation-based XLT (§4.1). Based on these findings, we compare

[2] We provide a complete list of target languages in App. E.

	Masakha	xSID	Avg
Translate-Train			
NO-FILT	$65.5_{\pm1.2}$	$82.8_{\pm0.6}$	$74.2_{\pm1.0}$
COMP-INS	$65.8_{\pm1.3}$	$82.8_{\pm0.5}$	$74.3_{\pm1.0}$
+ COMP-TGT	$66.0_{\pm1.7}$	$82.0_{\pm0.7}$	$74.0_{\pm1.3}$
+ COMP-TGT + COMP-SRC	$66.6_{\pm1.1}$	$82.0_{\pm0.9}$	$74.3_{\pm1.0}$
Translate-Test			
NO-FILT	$46.3_{\pm0.4}$	$68.2_{\pm0.4}$	$57.2_{\pm0.4}$
RSTR-TGT	$57.9_{\pm0.5}$	$74.8_{\pm0.4}$	$66.4_{\pm0.5}$
+ COMP-TGT	$57.8_{\pm0.5}$	$74.3_{\pm0.4}$	$66.0_{\pm0.5}$
+ COMP-SRC	$58.6_{\pm0.5}$	$73.9_{\pm0.5}$	$66.2_{\pm0.5}$
+ COMP-TGT + COMP-SRC	$58.0_{\pm0.5}$	$73.2_{\pm0.4}$	$65.6_{\pm0.5}$

Table 1: Results on the validation data for WA-based XLT with various filtering strategies. Results with XLM-R and whitespace pre-tokenization (WS-TOK).

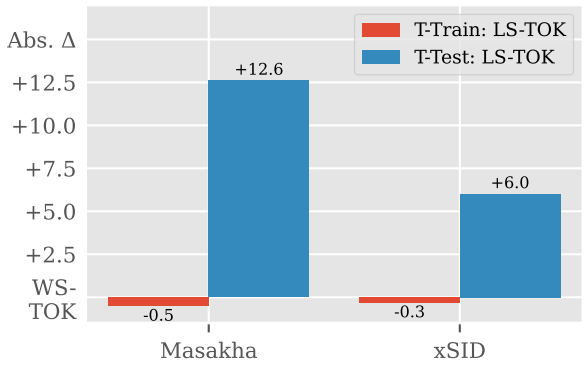

Figure 2: Transfer performance with WA for language-specific pre-tokenization (LS-TOK) relative to whitespace pre-tokenization (WS-TOK). Results with XLM-R, COMP-INS+COMP-TGT+COMP-SRC filtering for T-Train and RSTR-TGT filtering for T-Test.

WA—applying the on average best-performing pre-tokenization and filtering strategies—against two state-of-the-art marker-based methods, Easy and Codec, on the test portions of our datasets (§4.2). Last, we provide further ablations in §4.3, analyzing the impact of decoder-turned encoder LLMs, WA, and MT model on the XLT performance.

4.1 WA Design Choices

Filtering Strategies. Our preliminary experiments (Table 1) reveal that filtering has a negligible effect on performance in T-Train: on average none of the filtering strategies yields substantial gains. Still, COMP-INS and COMP-INS+COMP-TGT+COMP-SRC perform the best on average. This finding is positive, as searching for 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, applying the RSTR-TGT filter results in a substantial gain (+9.2% over NO-FILT) for T-Test. On average, we find RSTR-TGT to be the most successful strategy for T-Test. Adding additional filtering (+COMP-SRC) only results in marginal gains for Masakha.

Pre-Tokenization. Figure 2 shows the results for various 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 pre-tokenization (WS-TOK) is marginally better than language-specific pre-tokenization (LS-TOK)— and (2) a substantial impact on T-Test—LS-TOK outperforms WS-TOK by 12.6% on Masakha and 6.0% on xSID. Our findings add to prior work (Artetxe et al., 2023; Ebing and Glavaš, 2024), which showed that T-Test is more affected by translation quality than T-Train: our results extend this finding to filtering and pre-tokenization strategies.

4.2 Main Results

We now compare our optimized WA-based configurations for T-Train and T-Test against the state-of-the-art marker-based label projection methods Easy and Codec. For the remainder of experiments, we apply language-agnostic whitespace pre-tokenization (WS-TOK) and COMP-INS+COMP-TGT+COMP-SRC filtering for T-Train and language-specific pre-tokenization (LS-TOK) and RSTR-TGT filtering for T-Test.

Translate-Train. We find T-Train, regardless of the label projection strategy (WA, Easy, or Codec), to substantially outperform 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 demonstrate that—when reasonably configured—WA yields competitive performance: on xSID, 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 (WA, Easy, or Codec), T-Test outperforms zero-shot XLT (and T-Train on Masakha). Again we observe that optimized WA-based label projection matches and even slightly surpasses the performance of the marker-based Codec. This is encouraging because the label projection with Codec—due to its two-step translation procedure—is computationally more expensive (i.e., slower)

		Masakha	xSID	Avg
Zero-Shot				
	X	$52.9_{\pm 1.8}$	$76.8_{\pm 1.4}$	$64.9_{\pm 1.6}$
Translate-Train				
Easy	X	$66.0_{\pm 0.9}$	$83.6_{\pm 0.9}$	$74.8_{\pm 0.9}$
Codec	X	$66.9_{\pm 1.6}$	$83.4_{\pm 0.8}$	$75.2_{\pm 1.3}$
WA	X	$67.1_{\pm 1.2}$	$82.7_{\pm 0.9}$	$74.9_{\pm 1.0}$
Translate-Test				
Codec	X	$72.0_{\pm 0.5}$	$79.4_{\pm 0.3}$	$75.7_{\pm 0.4}$
Codec	D	$72.4_{\pm 0.4}$	$79.5_{\pm 0.4}$	$76.0_{\pm 0.4}$
WA	X	$72.3_{\pm 0.5}$	$80.2_{\pm 0.3}$	$76.3_{\pm 0.4}$
WA	D	$\mathbf{72.7}_{\pm 0.4}$	$80.2_{\pm 0.4}$	$76.5_{\pm 0.4}$
Ensemble-Train-Test				
Easy + WA	X + D	$71.7_{\pm 0.7}$	$\mathbf{83.8}_{\pm 0.8}$	$77.7_{\pm 0.7}$
Codec + WA	X + D	$72.3_{\pm 0.7}$	$82.8_{\pm 0.8}$	$77.5_{\pm 0.7}$
WA + WA	X + D	$72.6_{\pm 0.6}$	$83.4_{\pm 0.9}$	$\mathbf{78.0}_{\pm 0.8}$

Table 2: Main results for translation-based XLT for token-level tasks. Results with XLM-R (X) and DeBERTa (D). We denote the ensembles as $\star + \bullet$, where $\star$ refers to the `T-Train` and $\bullet$ to the `T-Test` component.

		Masakha	xSID	Avg
Translate-Test				
WS-TOK	D	$59.8_{\pm 0.3}$	$73.2_{\pm 0.4}$	$66.6_{\pm 0.4}$
LS-TOK	D	$\mathbf{72.7}_{\pm 0.4}$	$80.2_{\pm 0.4}$	$76.5_{\pm 0.4}$
Ensemble-Train-Test				
WS- + WS-TOK	X + D	$72.0_{\pm 0.8}$	$81.2_{\pm 1.0}$	$76.6_{\pm 0.9}$
WS- + LS-TOK	X + D	$72.6_{\pm 0.6}$	$\mathbf{83.4}_{\pm 0.9}$	$\mathbf{78.0}_{\pm 0.8}$

Table 3: Robustness results for `ETT` utilizing different pre-tokenizations for the `T-Test` component: whitespace (WS-TOK) and language-specific (LS-TOK). Results with XLM-R (X) and DeBERTa (D). We denote the ensembles as $\star + \bullet$, where $\star$ refers to the `T-Train` and $\bullet$ to the `T-Test` component.

than WA. Further, we observe that models solely trained on English (i.e., DeBERTa) only offer marginal gains over comparable mLMs (i.e., XLM-R). We speculate that this is because NER and SL 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 3.1% and 1.5%, respectively. 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 reduces sensitivity 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 `LS-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. We observe that for `T-Test`, `WS-TOK` underperforms `LS-TOK` by 9.9% on average. `ETT`, however, almost completely closes the gap between the two, with `WS-TOK` trailing `LS-TOK` by only 1.4%, making the choice of the 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 for token-level tasks (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 evaluate a state-of-the-art decoder-turned-encoder Llama 3 8B (LLM2Vec) (BehnamGhader et al., 2024) in translation-based XLT for token classification. Following the original work, we add a linear classifier with dropout on top of LLM2Vec, fine-tuning only the classifier. Figure 3 summarizes the results. We observe that much smaller DeBERTa is superior to LLM2Vec in `T-Test` (i.e., +12.4% on Masakha and 9.9% on xSID) and that `T-Test` with LLM2Vec even trails zero-shot transfer with XLM-R on xSID. In `ETT` with an XLM-R-based `T-Train` component, LLM2Vec becomes much more competitive and lags DeBERTa by only 1.5% on average, further emphasizing the robustness that our ETT ensembling 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 sen-

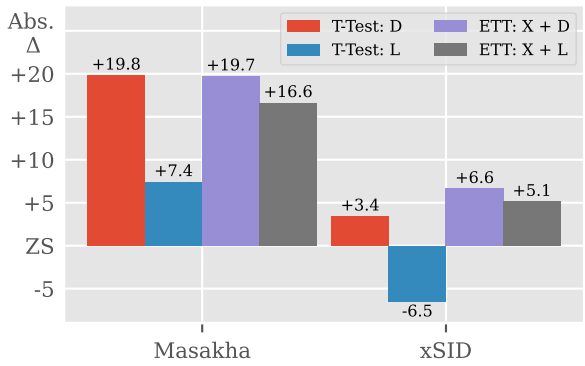

Figure 3: Results for translation-based XLT with LLM2Vec (L) vs. DeBERTa (D), relative to zero-shot XLT performance with XLM-R (X). We denote the ensembles as ⋆ + •, where ⋆ refers to the `T-Train` and • to the `T-Test` component.

	Masakha	xSID	Avg
Translate-Train			
AccAlign	$67.1_{\pm 1.2}$	$82.7_{\pm 0.8}$	$75.0_{\pm 1.0}$
AccAlgin$_{noft}$	$66.7_{\pm 1.1}$	$\mathbf{82.9}_{\pm 0.5}$	$74.9_{\pm 1.0}$
Awesome$_{noft}$	$64.4_{\pm 1.3}$	$79.8_{\pm 0.8}$	$72.7_{\pm 1.2}$
Translate-Test			
AccAlign	$\mathbf{72.3}_{\pm 0.5}$	$80.2_{\pm 0.3}$	$\mathbf{76.3}_{\pm 0.4}$
AccAlgin$_{noft}$	$70.4_{\pm 0.5}$	$79.6_{\pm 0.3}$	$75.0_{\pm 0.4}$
Awesome$_{noft}$	$65.1_{\pm 0.4}$	$74.8_{\pm 0.3}$	$70.0_{\pm 0.4}$

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

tence encoder LaBSE. Both WAs were released in 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). Table 4 shows the results of the WA comparison. Without WA-specific fine-tuning, AccAlign outperforms Awesome by 2.2% for `T-Train` and 5.0% 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`, but it does bring a small performance boost (1.3%) 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

		Masakha	xSID	Avg
Translate-Test				
NLLB	D	$72.6_{\pm 0.4}$	$80.2_{\pm 0.4}$	$76.4_{\pm 0.4}$
GT	D	$\mathbf{74.9}_{\pm 0.4}$	$81.2_{\pm 0.4}$	$\mathbf{78.1}_{\pm 0.4}$
Ensemble-Train-Test				
NLLB + NLLB	X + D	$71.9_{\pm 0.6}$	$\mathbf{83.4}_{\pm 0.9}$	$77.7_{\pm 0.8}$
NLLB + GT	X + D	$73.0_{\pm 0.6}$	$83.2_{\pm 0.9}$	$\mathbf{78.1}_{\pm 0.8}$

Table 5: Results for translation-based XLT using different MT models for `T-Test`: NLLB and Google Translate (GT). Results with XLM-R (X) and DeBERTa (D). We denote the ensembles as ⋆ + •, where ⋆ refers to the `T-Train` and • to the `T-Test` component. Scores for Masakha differ as GT does not support all languages.

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 are on par with the performance improvements introduced by using our ensemble (`ETT`) with NLLB only. Additionally, the difference in `ETT` performance between GT and NLLB is negligible (0.4%), which once more points to the robustness that `ETT` brings to XLT for token classification tasks.

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 evaluation on two established benchmarks covering 29 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 showed that an optimized application of 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 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.

For the proposed filtering strategies, we strove for simple and task-agnostic designs. We are aware that the presented filtering strategies are only an excerpt from the endless search space of possible options. However, the contribution of our work does not focus on fully exploring the design space of filtering strategies but demonstrating that filtering strategies for WAs may have a substantial impact on the XLT performance. Further, we show that our proposed ensemble successfully mitigates the observed performance variations making suboptimal filtering strategies less influential for the final XLT performance.

Our span mapping approach assumes that the labeled spans projected from the source language sentence S to the target language sentence $\hat{T}$ for `T-Train` (or $\hat{S}$ to T for `T-Test`) are continuous—such that a single labeled span from the source language sentence may not be split into multiple labeled spans in the target language sentence. This is an (overly) simplifying assumption that might not hold for all pairs of source-to-target (or target-to-source) translations. In our evaluation, the assumption does not hold for some German and Dutch examples in the xSID dataset because the languages allow for sentence final verbs. Therefore, a single slot may be split into discontinuous labeled spans during translation. Nevertheless, our empirical evaluation shows strong results even for German and Dutch indicating that the appearance of discontinuous labeled spans is not too common.

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.

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 pre-training 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.

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.

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*.

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 Masakha (Adelani et al., 2022) and xSID (van der Goot et al., 2021), we concatenated the pre-tokenized input on whitespace before translation. We deviate from this for the Chinese data in xSID, where we merge Chinese tokens without whitespace. 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 was only trained in Serbian Cyrillic script. We accessed all datasets through the Hugging Face 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. 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 for preserving inserted tags (i.e., preserving "[" and "]" around labeled spans). Hence, we leverage their publicly released 3.3B parameter checkpoint from Chen et al. (2023) for translation. We accessed it through the Hugging Face 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. The translations for Codec are obtained using vanilla (i.e., non fine-tuned) NLLB. However, the constrained decoding (i.e., inserting the tags post-translation) requires a fine-tuned NLLB that is able to preserve/insert tags. Therefore, 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 Detailed Experimental Setup

We train both tasks (NER and SL) for 10 epochs using an effective batch size of 32. In case we can not fit the desired batch size, we utilize gradient accumulation. The learning rate is set to $1e^{-5}$ with a weight decay of 0.01. We implement a linear schedule of 10% warm-up and employ mixed precision. For the LLM2Vec experiments, we deviate from this setting as we only fine-tune the classifier. Following BehnamGhader et al. (2024), we set the learning rate to $5e^{-4}$. We evaluate models at the last checkpoint of training. We use the seqeval F1 implementation accessed through the Hugging Face library. Further, we access our downstream models—XLM-RoBERTa Large, DeBERTaV3 Large, and LLM2Vec Llama 3 8B Instruct MNTP—through the Hugging Face 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 the GPU time to 4000 hours across all translations and downstream fine-tunings.

Languages.

MasakhaNER2.0. Our experiments cover 18 out of 20 languages that are supported by NLLB. Note that Google Translate (GT) does not support all 18 languages. Following, we mark the 11 languages that are supported by GT with an additional asterisk: 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)*, Yorùrbá (yor)*, and isiZulu (zul)*.

xSID. We evaluate 11 languages all covered by NLLB and GT: 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 vali-

dation and test instances and spans only a fraction of entities compared to the other languages.

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 pre-tokenization (`WS-TOK`).

Pre-Tokenization. For the per-tokenization experiments, we filter the translated training data based on `COMP-INS+COMP-SRC+COMP-TGT`. For `T-Test`, we apply `RSTR-TGT`. Language-specific tokenization (`LS-TOK`) is done with the MosesTokenizer from the Sacremoses library (https://github.com/hplt-project/sacremoses), except for Chinese, where we use jieba (https://github.com/fxsjy/jieba). Both tokenizers are released under the MIT license.

Main Results and Further Findings. As suggested by the findings of our preliminary experiments, we apply whitespace pre-tokenization (`WS-TOK`) for `T-Train`, except for Chinese in xSID, where we use language-specific tokenization (`LS-TOK`). For `T-Test`, we use language-specific pre-tokenization (`LS-TOK`). We utilize the same filtering as for the pre-tokenization experiments: `COMP-INS+COMP-SRC+COMP-TGT` for `T-Train` and `RSTR-TGT` for `T-Test`.

F 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
COMP-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
COMP-INS + COMP-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
COMP-INS + COMP-TGT + COMP-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
Translate-Test																			
NO-FILT	26.2	49.0	40.0	55.5	43.3	55.7	58.6	52.9	36.6	51.9	52.6	55.3	52.6	54.3	42.9	30.9	29.0	45.7	46.3
RSTR-TGT	37.6	68.9	58.0	61.7	55.5	67.1	70.9	61.6	46.4	64.7	60.9	67.4	63.2	61.6	55.6	43.1	43.0	55.5	57.9
COMP-TGT	32.5	52.0	47.2	56.9	44.5	56.8	59.7	55.2	40.6	52.9	53.0	55.7	54.1	59.6	46.0	31.5	32.8	46.2	48.7
COMP-SRC	28.9	47.7	41.5	56.3	43.4	55.6	58.8	54.0	35.1	52.1	52.7	54.5	51.8	56.4	44.2	30.8	28.7	45.9	46.6
COMP-TGT + COMP-SRC	34.4	50.5	47.9	57.4	44.6	56.7	59.8	55.7	38.8	53.0	53.0	54.8	53.1	61.1	46.5	31.4	32.7	46.4	48.8
RSTR-TGT + COMP-TGT	36.8	67.7	52.4	62.9	55.9	66.8	71.6	62.5	45.0	65.0	61.0	68.0	64.2	61.1	57.0	43.6	43.4	55.6	57.8
RSTR-TGT + COMP-SRC	41.7	68.2	60.5	62.6	55.8	67.0	71.2	63.0	45.4	65.0	61.1	66.8	62.6	64.0	57.5	43.1	42.9	55.7	58.6
RSTR-TGT + COMP-TGT + COMP-SRC	39.1	66.6	53.2	63.5	56.1	66.7	71.7	63.1	43.4	65.1	61.1	67.2	63.3	62.7	57.8	43.5	43.4	55.8	58.0

Table 6: Results for translation-based XLT evaluated on the Masakha validation data utilizing different filtering strategies. 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
COMP-INS	85.1	82.4	88.7	58.3	86.0	90.1	70.7	93.3	84.6	88.0	-	82.7
COMP-INS + COMP-TGT	84.9	81.9	87.8	59.3	85.3	86.7	69.8	90.9	86.6	86.6	-	82.0
COMP-INS + COMP-TGT + COMP-SRC	86.3	81.3	86.7	58.8	85.0	88.1	69.6	91.3	86.2	86.5	-	82.0
Translate-Test												
NO-FILT	68.1	75.6	74.4	51.9	73.1	76.6	51.7	80.2	69.9	66.6	61.6	68.2
RSTR-TGT	73.6	78.1	82.7	57.3	73.5	83.4	61.7	86.4	75.9	75.1	75.4	74.8
COMP-TGT	68.4	75.5	79.4	53.8	74.3	75.5	59.8	79.8	70.6	74.9	74.6	71.5
COMP-SRC	67.2	74.3	73.4	51.7	71.5	75.5	50.4	78.9	68.2	65.2	61.0	67.0
COMP-TGT + COMP-SRC	67.4	74.3	78.8	53.4	72.8	74.8	59.0	78.9	68.7	72.8	73.5	70.4
RSTR-TGT + COMP-TGT	74.1	77.9	81.7	57.6	74.0	82.1	60.1	84.6	76.3	73.8	74.9	74.3
RSTR-TGT + COMP-SRC	72.9	77.0	82.0	57.2	71.9	82.5	60.8	85.2	74.3	73.9	75.1	73.9
RSTR-TGT + COMP-TGT + COMP-SRC	73.3	76.7	81.1	57.3	72.4	81.6	59.3	83.8	74.5	71.7	73.8	73.2

Table 7: Results for translation-based XLT evaluated on the xSID validation data utilizing different filtering strategies. We use XLM-R. For `T-Train`, we excluded Chinese (zh) since experiments were run with whitespace pre-tokenization (`WS-TOK`).

G 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
LS-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	37.6	68.9	58.0	61.7	55.5	67.1	70.9	61.6	46.4	64.7	60.9	67.4	63.2	61.6	55.6	43.1	43.0	55.5	57.9
LS-TOK	50.8	82.5	74.8	67.0	71.5	78.8	86.8	76.2	55.2	75.8	82.0	81.9	76.1	68.8	66.1	62.9	54.7	69.4	71.2

Table 8: 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
LS-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	73.6	78.1	82.7	57.3	73.5	83.4	61.7	86.4	75.9	75.1	75.4	74.8
LS-TOK	79.1	81.3	89.7	62.7	78.7	89.6	69.9	92.9	80.4	81.7	82.7	80.8

Table 9: 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.

H 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	LS-TOK	NLLB	54.7	79.1	72.4	74.0	73.9	70.4	83.8	73.3	52.7	78.6	81.2	83.1	79.2	70.0	66.1	72.8	57.8	78.5	72.3
AccAlign†	X	LS-TOK	NLLB	54.1	76.3	69.1	73.4	72.4	69.6	82.7	71.4	48.2	77.6	80.0	81.7	79.4	70.5	62.8	71.1	49.4	77.0	70.4
Awesome†	X	LS-TOK	NLLB	46.3	72.4	58.6	69.4	75.3	65.0	81.6	72.1	42.4	78.4	66.3	80.0	78.3	68.9	53.4	52.1	47.7	64.3	65.1
AccAlign	D	LS-TOK	NLLB	54.3	79.2	73.3	74.5	75.1	71.2	84.0	75.0	52.5	79.1	81.6	83.7	79.3	70.2	66.3	72.9	57.6	78.2	72.7
AccAlign	X	WS-TOK	NLLB	44.6	69.6	55.8	63.3	59.1	58.4	72.9	60.2	40.7	68.2	60.8	68.3	63.7	60.9	55.2	61.8	46.2	62.1	59.5
AccAlign	D	WS-TOK	NLLB	44.3	69.7	56.2	63.8	59.9	58.9	72.6	61.6	40.4	68.7	61.4	68.8	64.3	61.3	55.8	61.7	46.2	61.7	59.8
AccAlign	X	LS-TOK	GT	60.9	79.3	-	73.4	78.0	71.7	-	-	-	-	83.4	85.2	-	71.3	-	75.0	63.7	78.2	-
AccAlign	D	LS-TOK	GT	60.2	79.4	-	74.1	79.1	72.3	-	-	-	-	83.9	85.9	-	72.9	-	75.1	62.2	78.4	-
AccAlign	ShL	LS-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	LS-TOK	NLLB	46.2	68.1	61.0	62.1	58.0	56.9	69.9	60.9	43.0	69.0	67.8	72.5	66.7	56.4	56.9	59.3	45.5	65.5	60.3
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- + WS-TOK	NLLB	57.0	79.1	74.3	72.6	77.6	63.5	81.8	69.8	59.4	74.9	75.6	83.8	78.4	66.3	67.6	72.0	52.6	78.7	71.4
AccAlign + AccAlign	X + X	WS- + LS-TOK	NLLB	57.3	78.5	75.7	72.8	79.0	64.1	82.5	70.3	60.2	75.3	77.2	84.3	78.9	66.7	68.0	72.4	53.5	78.8	72.0
AccAlign + AccAlign	X + D	WS- + WS-TOK	NLLB	56.9	79.5	75.2	73.3	79.0	64.2	81.7	71.5	59.3	75.6	76.4	83.9	79.3	67.5	68.3	72.8	53.0	79.1	72.0
AccAlign + AccAlign	X + D	WS- + LS-TOK	NLLB	57.0	79.2	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	72.9	53.9	79.1	72.6
AccAlign + AccAlign	X + D	WS- + LS-TOK	GT	61.0	79.0	-	73.6	81.1	64.9	-	-	-	-	78.5	85.0	-	67.8	-	74.0	58.2	80.0	-
AccAlign + AccAlign	X + ShL	WS- + LS-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- + LS-TOK	NLLB	53.9	77.2	74.2	72.9	74.2	60.5	80.0	67.1	58.2	72.1	74.1	83.1	78.0	63.8	67.2	70.5	47.2	77.5	69.5
Easy + AccAlign	X + D	- + LS-TOK	NLLB	58.0	79.2	74.0	73.3	76.4	64.5	83.2	71.1	56.3	77.8	72.6	85.2	78.7	68.6	68.3	73.8	53.6	76.3	71.7
Codec + AccAlign	X + D	- + LS-TOK	NLLB	55.4	79.2	74.9	73.5	77.7	63.4	82.4	72.2	57.9	77.4	77.6	85.0	77.9	69.3	69.4	74.0	54.3	79.1	72.3

Table 10: 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). We denote the ensembles as $\star + \bullet$, where $\star$ refers to the `T-Train` and $\bullet$ to the `T-Test` component. 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	kk	nl	sr	tr	zh	Avg
ZS				71.5	85.6	80.8	43.9	86.8	88.2	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	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	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	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	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	85.1	89.9	82.7	81.2	84.4	83.6
Translate-Test															
AccAlign	X	LS-TOK	NLLB	78.8	76.0	86.3	60.9	78.8	88.1	82.5	87.5	79.8	81.3	82.3	80.2
AccAlign†	X	LS-TOK	NLLB	77.8	75.4	84.8	59.6	79.3	85.7	82.1	86.8	80.0	82.0	82.6	79.6
Awesome†	X	LS-TOK	NLLB	73.8	74.9	84.8	59.2	71.0	84.8	63.0	87.2	77.1	70.8	76.4	74.8
AccAlign	D	LS-TOK	NLLB	79.3	75.8	85.6	59.1	80.0	88.7	82.5	86.7	80.0	82.2	82.1	80.2
AccAlign	X	WS-TOK	NLLB	72.9	70.6	79.1	55.7	72.2	81.2	74.2	82.0	73.8	72.9	72.6	73.4
AccAlign	D	WS-TOK	NLLB	73.4	70.2	77.9	53.6	73.3	82.2	74.2	80.7	73.8	73.4	72.5	73.2
AccAlign	X	LS-TOK	GT	80.5	76.5	86.4	61.8	78.6	88.2	83.8	87.0	82.2	81.6	86.2	81.2
AccAlign	D	LS-TOK	GT	80.5	76.5	85.6	59.3	79.6	89.3	84.0	87.1	83.4	82.4	85.7	81.2
AccAlign	ShL	LS-TOK	NLLB	68.4	68.3	76.8	51.6	68.6	77.7	72.2	78.1	70.7	72.2	73.3	70.7
AccAlign	L	LS-TOK	NLLB	68.6	69.2	76.1	50.1	69.9	76.9	70.4	77.8	70.8	71.1	72.8	70.3
Codec	X	LS-TOK	NLLB	79.0	81.9	86.1	60.4	84.8	88.4	83.0	86.5	72.4	83.6	67.0	79.4
Codec	D	LS-TOK	NLLB	79.9	81.8	85.5	58.8	85.8	89.0	83.2	86.0	72.9	84.2	67.5	79.5
Ensembling-Translate-Train															
AccAlign + AccAlign	X + X	WS- + WS-TOK	NLLB	80.9	75.1	85.9	63.1	88.1	89.2	81.4	86.3	82.7	81.5	80.4	81.3
AccAlign + AccAlign	X + X	WS- + LS-TOK	NLLB	82.0	76.3	88.2	64.5	88.7	89.6	86.5	86.4	82.9	85.0	85.0	83.2
AccAlign + AccAlign	X + D	WS- + WS-TOK	NLLB	81.2	75.0	85.8	61.7	88.2	89.9	81.1	86.6	82.6	81.7	79.6	81.2
AccAlign + AccAlign	X + D	WS- + LS-TOK	NLLB	82.4	76.1	88.2	64.2	89.0	90.0	86.6	87.2	83.1	85.3	84.8	83.4
AccAlign + AccAlign	X + D	WS- + LS-TOK	GT	82.4	75.8	87.1	64.3	87.7	89.8	86.5	86.3	84.4	84.2	86.4	83.2
AccAlign + AccAlign	X + ShL	WS- + LS-TOK	NLLB	81.2	76.2	87.4	63.6	87.7	89.6	86.0	86.1	83.7	84.7	84.5	82.8
AccAlign + AccAlign	X + L	WS- + LS-TOK	NLLB	79.7	76.4	87.3	61.8	86.1	88.4	85.4	86.1	82.9	83.3	84.0	81.9
Easy + AccAlign	X + D	- + LS-TOK	NLLB	81.1	81.2	89.7	64.4	89.0	90.6	85.7	90.4	82.7	82.4	84.5	83.8
Codec + AccAlign	X + D	- + LS-TOK	NLLB	82.5	75.3	89.1	62.7	87.6	88.7	87.1	89.2	81.8	85.3	81.3	82.8

Table 11: 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). We denote the ensembles as $\star + \bullet$, where $\star$ refers to the `T-Train` and $\bullet$ to the `T-Test` component. WAs marked with † are not fine-tuned. For `T-Train`, we pre-tokenize Chinese (zh) with `LS-TOK`.