UXLA: A Robust Unsupervised Data Augmentation Framework for Zero-Resource Cross-Lingual NLP

M Saiful Bari [*¶], **Tasnim Mohiuddin** [*¶], **and Shafiq Joty**[¶§]
[¶] Nanyang Technological University, Singapore
[§] Salesforce Research Asia, Singapore
{bari0001@e., mohi0004@e., srjoty@}ntu.edu.sg

Abstract

Transfer learning has yielded state-of-the-art (SoTA) results in many supervised NLP tasks. However, annotated data for every target task in every target language is rare, especially for low-resource languages. We propose UXLA a novel unsupervised data augmentation framework for zero-resource transfer learning scenarios. In particular, UXLA aims to solve cross-lingual adaptation problems from a source language task distribution to an unknown target language task distribution, assuming no training label in the target language. At its core, UXLA performs simultaneous self-training with data augmentation and unsupervised sample selection. To show its effectiveness, we conduct extensive experiments on three diverse zero-resource cross-lingual transfer tasks. UXLA achieves SoTA results in all the tasks, outperforming the baselines by a good margin. With an in-depth framework dissection, we demonstrate the cumulative contributions of different components to its success.

1 Introduction

Self-supervised learning in the form of pretrained language models (LM) has been the driving force in developing state-of-the-art NLP systems in recent years. These methods typically follow two basic steps, where a *supervised* task-specific fine-tuning follows a large-scale LM pretraining (Radford et al., 2019). However, getting labeled data for every target task in every target language is difficult, especially for low-resource languages.

Recently, the *pretrain-finetune* paradigm has also been extended to multi-lingual setups to train effective multi-lingual models that can be used for zero-shot cross-lingual transfer. Jointly trained deep multi-lingual LMs like mBERT (Devlin et al., 2019) and XLM-R (Conneau et al., 2020) coupled with supervised fine-tuning in the source language have been quite successful in transferring linguistic and task knowledge from one language to another without using any task label in the target language. The joint pretraining with multiple languages allows these models to generalize across languages.

Despite their effectiveness, recent studies (Pires et al., 2019; K et al., 2020) have also highlighted one crucial limiting factor for successful cross-lingual transfer. They all agree that the cross-lingual generalization ability of the model is limited by the (lack of) structural similarity between the source and target languages. For example, for transferring mBERT from English, K et al. (2020) report about 23.6% accuracy drop in Hindi (structurally dissimilar) compared to 9% drop in Spanish (structurally similar) in cross-lingual natural language inference (XNLI). The difficulty level of transfer is further exacerbated if the (dissimilar) target language is low-resourced, as the joint pretraining step may not have seen many instances from this language in the first place. In our experiments (§3.2), in cross-lingual NER (XNER), we report F1 reductions of 28.3% in Urdu and 30.4% in Burmese for XLM-R, which is trained on a much larger multi-lingual dataset than mBERT.

One attractive way to improve cross-lingual generalization is to perform *data augmentation* (Simard et al., 1998), and train the model on examples that are similar but different from the labeled data in the source language. Formalized by the Vicinal Risk Minimization (VRM) principle (Chapelle et al., 2001), such data augmentation methods have shown impressive results in vision (Zhang et al., 2018; Berthelot et al., 2019). These methods enlarge the support of the training distribution by generating new data points from a *vicinity distribution* around each training example. For images, the vicinity of a training image can be defined by a set of operations like rotation and scaling, or by

[*]Equal contribution

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 1978–1992
August 1–6, 2021.

linear mixtures of features and labels (Zhang et al., 2018). However, when it comes to text, such *unsupervised* augmentation methods have rarely been successful. The main reason is that unlike images, linguistic units are discrete and a smooth change in their embeddings may not result in a plausible linguistic unit that has similar meanings.

In NLP, to the best of our knowledge, the most successful augmentation method has so far been back-translation (Sennrich et al., 2016) which paraphrases an input sentence through round-trip translation. However, it requires parallel data to train effective machine translation systems, acquiring which can be more expensive for low-resource languages than annotating the target language data. Furthermore, back-translation is only applicable in a supervised setup and to tasks where it is possible to find the alignments between the original labeled entities and the back-translated entities, such as in question answering (Yu et al., 2018). Other related work includes contextual augmentation (Kobayashi, 2018), conditional BERT (Wu et al., 2018) and AUG-BERT (Shi et al., 2019). These methods use a constrained augmentation that alters a pretrained LM to a label-conditional LM for a specific task. Since they rely on labels, their application is limited by the availability of enough task labels.

In this work, we propose UXLA, a robust **u**nsupervised **cross-l**ingual **a**ugmentation framework for improving cross-lingual generalization of multilingual LMs. UXLA augments data from the unlabeled training examples in the target language as well as from the virtual input samples generated from the vicinity distribution of the source and target language sentences. With the augmented data, it performs simultaneous *self-learning* with an effective *distillation strategy* to learn a strongly adapted cross-lingual model from noisy (pseudo) labels for the target language task. We propose novel ways to generate virtual sentences using a multilingual masked LM (Conneau et al., 2020), and get reliable task labels by simultaneous multilingual co-training. This co-training employs a two-stage co-distillation process to ensure robust transfer to dissimilar and/or low-resource languages.

We validate the effectiveness and robustness of UXLA by performing extensive experiments on three diverse zero-resource cross-lingual transfer tasks–XNER, XNLI, and PAWS-X, which posit different sets of challenges, and across many (14 in total) language pairs comprising languages that are similar/dissimilar/low-resourced. UXLA yields impressive results on XNER, setting SoTA in all tested languages outperforming the baselines by a good margin. The relative gains for UXLA are particularly higher for structurally dissimilar and/or low-resource languages: 28.54%, 16.05%, and 9.25% absolute improvements for Urdu, Burmese, and Arabic, respectively. For XNLI, with only 5% labeled data in the source, it gets comparable results to the baseline that uses all the labeled data, and surpasses the standard baseline by 2.55% on average when it uses all the labeled data in the source. We also have similar findings in PAWS-X. We provide a comprehensive analysis of the factors that contribute to UXLA's performance. We open-source our framework at https://ntunlpsg.github.io/project/uxla/ .

2 UXLA Framework

While recent cross-lingual transfer learning efforts have relied almost exclusively on multi-lingual pretraining and zero-shot transfer of a fine-tuned source model, we believe there is a great potential for more elaborate methods that can leverage the unlabeled data better. Motivated by this, we present UXLA, our unsupervised data augmentation framework for zero-resource cross-lingual task adaptation. Figure 1 gives an overview of UXLA.

Let $\mathcal{D}_s = (\mathcal{X}_s, \mathcal{Y}_s)$ and $\mathcal{D}_t = (\mathcal{X}_t)$ denote the training data for a source language s and a target language t, respectively. UXLA augments data from various origins at different stages of training. In the initial stage (epoch 1), it uses the augmented training samples from the target language ($\mathcal{D}'_t$) along with the original source ($\mathcal{D}_s$). In later stages (epoch 2-3), it uses vicinal sentences generated from the vicinity distribution of source and target examples: $\vartheta(\tilde{x}^s_n|x^s_n)$ and $\vartheta(\tilde{x}^t_n|x^t_n)$, where $x^s_n \sim \mathcal{X}_s$ and $x^t_n \sim \mathcal{X}_t$. It performs *self-training* on the augmented data to acquire the corresponding pseudo labels. To avoid *confirmation bias* with self-training where the model accumulates its own errors, it simultaneously trains three task models to generate *virtual* training data through data augmentation and filtering of potential label noises via multi-epoch *co-teaching* (Zhou and Li, 2005).

In each epoch, the co-teaching process first performs *co-distillation*, where two peer task models are used to select "reliable" training examples to train the third model. The selected samples with pseudo labels are then added to the target task

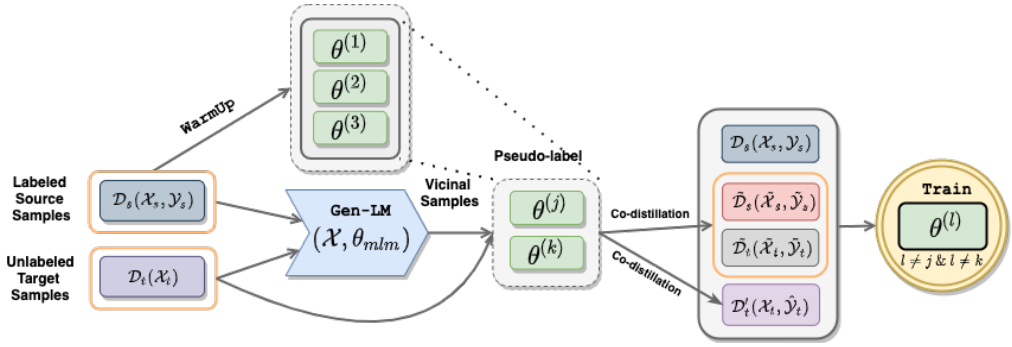


Figure 1: Training flow of UXLA. After training the base task models $\theta^{(1)}$, $\theta^{(2)}$, and $\theta^{(3)}$ on source labeled data $\mathcal{D}_s$ (**WarmUp**), we use two of them ($\theta^{(j)}$, $\theta^{(k)}$) to **pseudo-label** and **co-distill** the unlabeled target language data ($\mathcal{D}'_t$). A pretrained LM (**Gen-LM**) is used to generate new vicinal samples for both source and target languages, which are also pseudo-labeled and co-distilled using the two task models ($\theta^{(j)}$, $\theta^{(k)}$) to generate $\tilde{\mathcal{D}}_s$ and $\tilde{\mathcal{D}}_t$. The third model $\theta^{(l)}$ is then progressively trained on these datasets: $\{\mathcal{D}_s, \mathcal{D}'_t\}$ in epoch 1, $\tilde{\mathcal{D}}_t$ in epoch 2, and all in epoch 3.

model's training data by taking the agreement from the other two models, a process we refer to as *co-guessing*. The co-distillation and co-guessing mechanism ensure robustness of UXLA to out-of-domain distributions that can occur in a multilingual setup, *e.g.,* due to a structurally dissimilar and/or low-resource target language. Algorithm 1 gives a pseudocode of the overall training method. Each of the *task* models in UXLA is an instance of XLM-R fine-tuned on the source language task (e.g., English NER), whereas the pretrained masked LM parameterized by θ_{mlm} (*i.e.,* before fine-tuning) is used to define the *vicinity* distribution $\vartheta(\tilde{x}_n|x_n, \theta_{\text{mlm}})$ around each selected example x_n. In the following, we describe the steps in Algorithm 1.

2.1 Warm-up: Training Task Models

We first train three instances of the XLM-R model ($\theta^{(1)}, \theta^{(2)}, \theta^{(3)}$) with an additional task-specific linear layer on the source language (English) labeled data. Each model has the same architecture (XLM-R large) but is initialized with different random seeds. For token-level prediction tasks (*e.g.,* NER), the *token-level* representations are fed into the classification layer, whereas for sentence-level tasks (*e.g.,* XNLI), the `[CLS]` representation is used as input to the classification layer.

Training with confidence penalty Our goal is to train the task models so that they can be used reliably for self-training on a target language that is potentially dissimilar and low-resourced. In such situations, an overly confident (overfitted) model may produce more noisy pseudo labels, and the noise will then accumulate as the training progresses. Overly confident predictions may also impose difficulties on our distillation methods (§2.3) in isolating good samples from noisy ones. However, training with the standard cross-entropy (CE) loss may result in overfitted models that produce overly confident predictions (low entropy), especially when the class distribution is not balanced. We address this by adding a negative entropy term $-\mathcal{H}$ to the CE loss as follows.

$$\mathcal{L}(\theta) = \sum_{c=1}^{C} \Big[\underbrace{- y^c \log p^c_\theta(\mathbf{x})}_{\text{CE}} + \underbrace{p^c_\theta(\mathbf{x}) \log p^c_\theta(\mathbf{x})}_{-\mathcal{H}} \Big] \quad (1)$$

where $\mathbf{x}$ is the representation that goes to the output layer, and y^c and $p^c_\theta(\mathbf{x})$ are respectively the ground truth label and model predictions with respect to class c. Such regularizer of output distribution has been shown to be effective for training large models (Pereyra et al., 2017). We also report significant gains with confidence penalty in §3. Appendix B shows visualizations on why confidence penalty is helpful for distillation.

2.2 Sentence Augmentation

Our augmentated sentences come from two different sources: the *original* target language samples $\mathcal{X}_t$, and the *virtual* samples generated from the vicinity distribution of the source and target samples: $\vartheta(\tilde{x}^s_n|x^s_n, \theta_{\text{mlm}})$ and $\vartheta(\tilde{x}^t_n|x^t_n, \theta_{\text{mlm}})$ with $x^s_n \sim \mathcal{X}_s$ and $x^t_n \sim \mathcal{X}_t$. It has been shown that contextual LMs pretrained on large-scale datasets capture useful linguistic features and can be used to generate fluent grammatical texts (Hewitt and Manning, 2019). We use XLM-R masked LM (Conneau et al., 2020) as our vicinity model θ_{mlm}, which is trained on massive multilingual corpora (2.5 TB of Common-Crawl data in 100 languages). The

Algorithm 1 UXLA: a robust unsupervised data augmentation framework for cross-lingual NLP

Input: *source* (s) and *target* (t) language datasets: $\mathcal{D}_s = (\mathcal{X}_s, \mathcal{Y}_s), \mathcal{D}_t = (\mathcal{X}_t)$; task models: $\theta^{(1)}, \theta^{(2)}, \theta^{(3)}$, pre-trained masked LM θ_{mlm}, mask ratio P, diversification factor δ, sampling factor α, and distillation factor η
Output: models trained on augmented data

$\theta^{(1)}, \theta^{(2)}, \theta^{(3)} = \textsc{WarmUp}(\mathcal{D}_s, \theta^{(1)}, \theta^{(2)}, \theta^{(3)})$ ▷ warm up with conf. penalty.
for $e \in [1:3]$ **do** ▷ e denotes epoch.
  for $k \in \{1,2,3\}$ **do**
    $\mathcal{X}_t^{(k)}, \mathcal{Y}_t^{(k)} = \textsc{Distil}(\mathcal{X}_t, \eta_e, \theta^{(k)})$ ▷ infer and select tgt training data for augmentation.
    for $j \in \{1,2,3\}$ **do**
      if $k == j$ **then** Continue
      /* source language data augmentation */
      $\tilde{\mathcal{X}}_s = \textsc{Gen-LM}(\mathcal{X}_s, \theta_{\text{mlm}}, P, \delta)$ ▷ vicinal example generation.
      $\mathcal{X}_s^{(k)}, \mathcal{Y}_s^{(k)} = \textsc{Distil}(\tilde{\mathcal{X}}_s, \eta_e, \theta^{(k)})$; $\mathcal{X}_s^{(j)}, \mathcal{Y}_s^{(j)} = \textsc{Distil}(\tilde{\mathcal{X}}_s, \eta_e, \theta^{(j)})$
      $\tilde{\mathcal{D}}_s = \textsc{Agreement}\big(\mathcal{D}_s^{(k)} = (\mathcal{X}_s^{(k)}, \mathcal{Y}_s^{(k)}), \mathcal{D}_s^{(j)} = (\mathcal{X}_s^{(j)}, \mathcal{Y}_s^{(j)})\big)$
      /* target language data augmentation (no vicinity) */
      $\mathcal{X}_t^{(j)}, \mathcal{Y}_t^{(j)} = \textsc{Distil}(\mathcal{X}_t, \eta_e, \theta^{(j)})$
      $\mathcal{D}'_t = \textsc{Agreement}\big(\mathcal{D}_t^{(k)} = (\mathcal{X}_t^{(k)}, \mathcal{Y}_t^{(k)}), \mathcal{D}_t^{(j)} = (\mathcal{X}_t^{(j)}, \mathcal{Y}_t^{(j)})\big)$ ▷ see line 4
      /* target language data augmentation */
      $\tilde{\mathcal{X}}_t = \textsc{Gen-LM}(\mathcal{X}_t, \theta\text{mlm}, P, \delta)$ ▷ vicinal example generation.
      $\mathcal{X}_t^{(k)}, \mathcal{Y}_t^{(k)} = \textsc{Distil}(\tilde{\mathcal{X}}_t, \eta_e, \theta^{(k)})$; $\mathcal{X}_t^{(j)}, \mathcal{Y}_t^{(j)} = \textsc{Distil}(\tilde{\mathcal{X}}_t, \eta_e, \theta^{(j)})$
      $\tilde{\mathcal{D}}_t = \textsc{Agreement}\big(\mathcal{D}_t^{(k)} = (\mathcal{X}_t^{(k)}, \mathcal{Y}_t^{(k)}), \mathcal{D}_t^{(j)} = (\mathcal{X}_t^{(j)}, \mathcal{Y}_t^{(j)})\big)$
      /* train new models on augmented data */
      for $l \in \{1,2,3\}$ **do**
        if $l \neq j$ and $l \neq k$ **then**
          with sampling factor α, train $\theta^{(l)}$ on $\mathcal{D}$, ▷ train progressively
          where $\mathcal{D} = \{\mathcal{D}_s \mathbb{1}(e \in \{1,3\}) \cup \mathcal{D}'_t \mathbb{1}(e \in \{1,3\}) \cup \tilde{\mathcal{D}}_s \mathbb{1}(e = 3) \cup \tilde{\mathcal{D}}_t \mathbb{1}(e \in \{2,3\})\}$
Return $\{\theta^{(1)}, \theta^{(2)}, \theta^{(3)}\}$

vicinity model is a disjoint pretrained entity whose parameters are not trained on any task objective.

In order to generate samples around each *selected* example, we first randomly choose $P\%$ of the input tokens. Then we successively (one at a time) mask one of the chosen tokens and ask XLM-R masked LM to predict a token in that masked position, *i.e.,* compute $\vartheta(\tilde{x}_m|x, \theta_{\text{mlm}})$ with m being the index of the masked token. For a specific mask, we sample S candidate words from the output distribution, and generate novel sentences by following one of the two alternative approaches.

(i) Successive max In this approach, we take the most probable output token ($S = 1$) at each prediction step, $o^*_m = \arg\max_o \vartheta(\tilde{x}_m = o|x, \theta_{\text{mlm}})$. A new sentence is constructed by $P\%$ newly generated tokens. We generate δ (diversification factor) virtual samples for each original example x, by randomly masking $P\%$ tokens each time.

(ii) Successive cross In this approach, we divide each original (multi-sentence) sample x into two parts and use successive max to create two sets of augmented samples of size δ_1 and δ_2, respectively. We then take the cross of these two sets to generate $\delta_1 \times \delta_2$ augmented samples.

Augmentation of sentences through *successive max* or *cross* is carried out within the GEN-LM (generate via LM) module in Algorithm 1. For tasks involving a single sequence (*e.g.,* XNER), we directly use successive max. Pairwise tasks like XNLI and PAWS-X have pairwise dependencies: dependencies between a premise and a hypothesis in XNLI or dependencies between a sentence and its possible paraphrase in PAWS-X. To model such dependencies, we use successive cross, which uses cross-product of two successive max applied independently to each component.

2.3 Co-labeling through Co-distillation

Due to discrete nature of texts, VRM based augmentation methods that are successful for images such as MixMatch (Berthelot et al., 2019) that generates new samples and their labels as simple linear interpolation, have not been successful in NLP. The meaning of a sentence can change entirely even with minor variations in the original sentence. For example, consider the following example generated by our vicinity model.

Original: *EU rejects German call to boycott british lamb.*
Masked: *<mask> rejects german call to boycott british lamb.*
XLM-R: *Trump rejects german call to boycott british lamb.*

Here, EU is an *Organization* whereas the newly predicted word *Trump* is a *Person* (different name type). Therefore, we need to relabel the augmented

sentences no matter whether the original sentence has labels (source) or not (target). However, the relabeling process can induce noise, especially for dissimilar/low-resource languages, since the base task model may not be adapted fully in the early training stages. We propose a 2-stage sample distillation process to filter out noisy augmented data.

Stage 1: Distillation by single-model The first stage of distillation involves predictions from a single model for which we propose two alternatives:

(*i*) *Distillation by model confidence:* In this approach, we select samples based on the model's prediction confidence. This method is similar in spirit to the selection method proposed by Ruder and Plank (2018a). For sentence-level tasks (*e.g.,* XNLI), the model produces a single class distribution for each training example. In this case, the model's confidence is computed by $p^* = \max_{c\in\{1...C\}} p^c_\theta(\mathbf{x})$. For token-level sequence labeling tasks (*e.g.,* NER), the model's confidence is computed by: $p^* = \frac{1}{T}\sum_{t=1}^{T}\left\{\max_{c\in\{1...C\}} p^c_\theta(\mathbf{x}_t)\right\}$, where T is the length of the sequence. The distillation is then done by selecting the top $\eta\%$ samples with the highest confidence scores.

(*ii*) *Sample distillation by clustering:* We propose this method based on the finding that large neural models tend to learn good samples faster than noisy ones, leading to a lower loss for good samples and higher loss for noisy ones (Han et al., 2018; Arazo et al., 2019). We use a 1d two-component Gaussian Mixture Model (GMM) to model *per-sample loss distribution* and cluster the samples based on their *goodness*. GMMs provide flexibility in modeling the sharpness of a distribution and can be easily fit using *Expectation-Maximization* (EM) (See more on Appendix C). The loss is computed based on the pseudo labels predicted by the model. For each sample $\mathbf{x}$, its *goodness* probability is the posterior probability $p(z = g|\mathbf{x}, \theta_{\text{GMM}})$, where g is the component with smaller mean loss. Here, distillation hyperparameter η is the posterior probability threshold based on which samples are selected.

Stage 2: Distillation by model agreement In the second stage of distillation, we select samples by taking the agreement (co-guess) of two different peer models $\theta^{(j)}$ and $\theta^{(k)}$ to train the third $\theta^{(l)}$. Formally, $\textsc{Agreement}(\mathcal{D}^{(k)}, \mathcal{D}^{(j)}) = \{(\mathcal{X}^{(k)}, \mathcal{Y}^{(k)}) : \mathcal{Y}^{(k)} = \mathcal{Y}^{(j)}\}$ $s.t.$ $k \neq j$

2.4 Data Samples Manipulation

UXLA uses multi-epoch co-teaching. It uses $\mathcal{D}_s$ and $\mathcal{D}'_t$ in the first epoch. In epoch 2, it uses $\tilde{\mathcal{D}}_t$ (target virtual), and finally it uses all the four datasets - $\mathcal{D}_s$, $\mathcal{D}'_t$, $\tilde{\mathcal{D}}_t$, and $\tilde{\mathcal{D}}_s$ (line 22 in Algorithm 1). The datasets used at different stages can be of different sizes. For example, the number of augmented samples in $\tilde{\mathcal{D}}_s$ and $\tilde{\mathcal{D}}_t$ grow polynomially with the *successive cross* masking method. Also, the *co-distillation* produces sample sets of variable sizes. To ensure that our model does not overfit on one particular dataset, we employ a balanced sampling strategy. For N number of datasets $\{\mathcal{D}_i\}_{i=1}^N$ with probabilities, $\{p_i\}_{i=1}^N$, we define the following multinomial distribution to sample from:

$$p_i = \frac{f_i^\alpha}{\sum_{j=1}^N f_j^\alpha}, \text{ where } f_i = \frac{n_i}{\sum_{j=1}^N n_j} \quad (2)$$

where α is the sampling factor and n_i is the total number of samples in the i^{th} dataset. By tweaking α, we can control how many samples a dataset can provide in the mix.

3 Experiments

We consider three tasks in the *zero-resource* cross-lingual transfer setting. We assume labeled training data only in English, and transfer the trained model to a target language. For all experiments, we report the *mean score* of the three models that use different seeds.

3.1 Tasks & Settings

XNER: We use the standard CoNLL datasets (Sang, 2002; Sang and Meulder, 2003) for English (en), German (de), Spanish (es) and Dutch (nl). We also evaluate on Finnish (fi) and Arabic (ar) datasets collected from Bari et al. (2020). Note that Arabic is structurally different from English, and Finnish is from a different language family. To show how the models perform on extremely low-resource languages, we experiment with three structurally different languages from WikiANN (Pan et al., 2017) of different (unlabeled) training data sizes: Urdu (ur-20k training samples), Bengali (bn-10K samples), and Burmese (my-100 samples).

XNLI We use the standard dataset (Conneau et al., 2018). For a given pair of sentences, the task is to predict the entailment relationship between the two sentences, *i.e.*, whether the second sentence (*hypothesis*) is an *Entailment*, *Contradiction*, or

Model	en	es	nl	de	ar	fi
Supervised Results						
LSTM-CRF (Bari et al., 2020)	89.77	84.71	85.16	78.14	75.49	84.21
XLM-R (Conneau et al., 2020)	92.92	89.72	92.53	85.81	–	–
XLM-R (our imp.)	92.9	89.2	92.9	86.2	86.8	92.4
Zero-Resource Baseline						
$\text{mBERT}_{\text{cased}}$ (our imp.)	91.13	74.76	79.58	70.99	45.48	65.95
XLM-R (our imp.)	92.23	79.29	80.87	73.40	49.04	75.57
XLM-R (ensemble)	92.76	80.62	81.46	75.40	52.30	76.85
Our Method						
$\text{mBERT}_{\text{cased}}$ +con-penalty	90.81	75.06	79.26	72.31	47.03	66.72
XLM-R+con-penalty	92.49	80.45	81.07	73.76	49.94	76.05
UXLA	–	83.05	85.21	80.33	57.35	79.75
UXLA (ensemble)	–	**83.24**	**85.32**	**80.99**	**58.29**	**79.87**

Table 1: **F1 scores** in XNER on the datasets from CoNLL and (Bari et al., 2020). "–" represents no results were reported.

Neutral with respect to the first one (*premise*). We experiment with Spanish, German, Arabic, Swahili (sw), Hindi (hi) and Urdu.

PAWS-X The Paraphrase Adversaries from Word Scrambling Cross-lingual task (Yang et al., 2019) requires the models to determine whether two sentences are paraphrases. We evaluate on all the six (typologically distinct) languages: fr, es, de, Chinese (zh), Japanese (ja), and Korean (ko).

Evaluation setup Our goal is to adapt a task model from a source language distribution to an unknown target language distribution assuming no labeled data in the target. In this scenario, there might be two different distributional gaps: (*i*) the generalization gap for the source distribution, and (*ii*) the gap between the source and target language distribution. We wish to investigate our method in tasks that exhibit such properties. We use the standard task setting for XNER, where we take 100% samples from the datasets as they come from various domains and sizes without any specific bias.

However, both XNLI and PAWS-X training data come with machine-translated texts in target languages. Thus, the data is parallel and lacks enough diversity (source and target come from the same domain). Cross-lingual models trained in this setup may pick up distributional bias (in the label space) from the source. Artetxe et al. (2020) also argue that the translation process can induce subtle artifacts that may have a notable impact on models.

Therefore, for XNLI and PAWS-X, we experiment with two different setups. First, to ensure distributional differences and non-parallelism, we use 5% of the training data from the source language and augment a different (nonparallel) 5% data for the target language. We used a different seed each time to retrieve this 5% data. Second, to compare with previous methods, we also evaluate on the standard 100% setup. The evaluation is done on the entire test set in both setups. We will refer to these two settings as **5%** and **100%**. More details about model settings are in Appendix D.

Model	ur	bn	my
Supervised Results			
XLM-R (our-impl)	97.1	97.8	76.8
Zero-Resource Results			
XLM-R (XTREME)	56.4	78.8	54.3
XLM-R (our imp.)	56.45	78.17	54.56
UXLA	**84.99**	**82.68**	**70.61**

Table 2: XNER results on WikiANN.

3.2 Results

XNER Table 1 reports the XNER results on the datasets from CoNLL and (Bari et al., 2020), where we also evaluate an *ensemble* by averaging the probabilities from the three models. We observe that after performing *warm-up* with conf-penalty (§2.1), XLM-R performs better than mBERT on average by ∼3.8% for all the languages. UXLA gives absolute improvements of 3.76%, 4.34%, 6.94%, 8.31%, and 4.18% for *es, nl, de, ar,* and *fi*, respectively. Interestingly, it surpasses *supervised* LSTM-CRF for *nl* and *de* without using any target language labeled data. It also produces comparable results for *es*.

In Table 2, we report the results on the three *low-resource* langauges from WikiANN. From these results and the results of *ar* and *fi* in Table 1, we see that UXLA is particularly effective for languages that are structurally dissimilar and/or low-resourced, especially when the base model is weak:

Model	en	es	de	ar	sw	hi	ur
Supervised Results (TRANSLATE-TRAIN-ALL)							
XLM-R	89.1	86.6	85.7	83.1	78.0	81.6	78.1
Zero-Resource Baseline for **Full (100%) English labeled** training set							
XLM-R (XTREME)	88.7	83.7	82.5	77.2	71.2	75.6	71.7
XLM-R (our imp.)	88.87	84.34	82.78	78.44	72.08	76.40	72.10
XLM-R (ensemble)	89.24	84.73	83.27	79.06	73.17	77.23	73.07
XLM-R+con-penalty	88.83	84.30	82.86	78.20	71.83	76.24	71.62
UXLA	–	85.65	84.15	80.50	74.70	78.74	73.35
UXLA (ensemble)	–	**86.12**	**84.61**	**80.89**	**74.89**	**78.98**	**73.45**
Zero-Resource Baseline for **5% English labeled** training set							
XLM-R (our imp.)	83.08	78.48	77.54	72.04	67.3	70.41	66.72
XLM-R (ensemble)	84.65	79.56	78.38	72.22	66.93	71.00	66.79
XLM-R+con-penalty	84.24	79.23	78.47	72.43	67.72	71.08	67.63
UXLA	–	81.53	80.88	77.42	72.31	74.70	70.84
UXLA (ensemble)	–	**82.35**	**81.93**	**78.56**	**73.53**	**75.20**	**71.15**

Table 3: Results in accuracy for XNLI.

28.54%, 16.05%, and 9.25% absolute improvements for ur, my and ar, respectively.

XNLI-5% From Table 3, we see that the performance of XLM-R trained on 5% data is surprisingly good compared to the model trained on full data (see XLM-R (our imp.)), lagging by only 5.6% on average. In our single GPU implementation of XNLI, we could not reproduce the reported results of Conneau et al. (2020). However, our results resemble the reported XLM-R results of XTREME (Hu et al., 2020). We consider XTREME as our standard baseline for XNLI-100%.

We observe that with only 5% labeled data in the source, UXLA gets comparable results to the XTREME baseline that uses 100% labeled data (lagging behind by only ∼0.7% on avg.); even for *ar* and *sw*, we get 0.22% and 1.11% improvements, respectively. It surpasses the standard 5% baseline by 4.2% on average. Specifically, UXLA gets absolute improvements of 3.05%, 3.34%, 5.38%, 5.01%, 4.29%, and 4.12% for *es, de, ar, sw, hi,* and *ur*, respectively. Again, the gains are relatively higher for low-resource and/or dissimilar languages despite the base model being weak in such cases.

XNLI-100% Now, considering UXLA's performance on the full (100%) labeled source data in Table 3, we see that it achieves SoTA results for all of the languages with an absolute improvement of 2.55% on average from the XTREME baseline. Specifically, UXLA gets absolute improvements of 1.95%, 1.68%, 4.30%, 3.50%, 3.24%, and 1.65% for *es, de, ar, sw, hi,* and *ur*, respectively.

PAWS-X Similar to XNLI, we observe sizable improvements for UXLA over the baselines on PAWS-X for both 5% and 100% settings (Table 4). Specifically, in 5% setting, UXLA gets absolute gains of 5.33%, 5.94%, 5.04%, 6.85%, 7.00%, and 5.45% for *de, es, fr, ja, ko,* and *zh*, respectively, while in 100% setting, it gets 2.21%, 2.36%, 2.00%, 3.99%, 4.53%, and 4.41% improvements respectively. In general, we get an average improvements of 5.94% and 3.25% in PAWS-X-5% and PAWS-X-100% settings respectively. Moreover, our 5% setting outperforms 100% XLM-R baselines for *es, ja,* and *zh*. Interestingly, in the 100% setup, our UXLA (ensemble) achieves almost similar accuracies compared to supervised finetuning of XLM-R on all target language training dataset.

4 Analysis

In this section, we analyze UXLA by dissecting it and measuring the contribution of its *each of the components*. For this, we use the XNER task and analyze the model based on the results in Table 1.

4.1 Analysis of distillation methods

Model confidence vs. clustering We first analyze the performance of our *single-model distillation* methods (§2.3) to see which of the two alternatives works better. From Table 5, we see that both perform similarly with *model confidence* being slightly better. In our main experiments (Tables 1-4) and subsequent analysis, we use model confidence for distillation. However, we should not rule out the clustering method as it gives a more general

Model	en	de	es	fr	ja	ko	zh
Supervised Results (TRANSLATE-TRAIN-ALL)							
XLM-R (our impl.)	95.8	92.5	92.8	93.5	85.5	86.6	87.6
Zero-Resource Baseline for **Full (100%) English labeled** training set							
XLM-R (XTREME)	94.7	89.7	90.1	90.4	78.7	79.0	82.3
XLM-R (our imp.)	95.46	90.06	89.92	90.85	79.89	79.74	82.49
XLM-R (ensemble)	96.10	90.75	90.55	91.80	80.55	80.70	83.45
XLM-R+con-penalty	95.38	90.75	90.72	91.71	81.77	82.07	84.25
UXLA	–	92.27	92.28	92.85	83.88	84.27	86.90
UXLA (ensemble)	–	**92.55**	**92.35**	**93.35**	**84.30**	**84.35**	**86.95**
Zero-Resource Baseline for **5% English labeled** training set							
XLM-R (our imp.)	91.15	83.72	84.32	85.08	73.65	72.60	77.22
XLM-R (ensemble)	92.05	84.05	84.65	85.75	74.30	71.95	77.50
XLM-R+con-penalty	91.85	86.15	86.38	85.98	76.03	75.43	79.15
UXLA	–	89.05	90.27	90.12	80.50	79.60	82.65
UXLA (ensemble)	–	**89.25**	**90.85**	**90.25**	**81.15**	**80.15**	**82.90**

Table 4: Results in accuracy for PAWS-X.

solution to consider other distillation features (*e.g.,* sequence length, language) than model prediction scores, which we did not explore in this paper.

Distillation factor η We next show the results for different distillation factor (η) in Table 5. Here 100% refers to the case when no single-model distillation is done based on model confidence. We notice that the best results for each of the languages are obtained for values other than 100%, which indicates that distillation is indeed an effective step in UXLA. See Appendix B for more analysis on η.

Two-stage distillation We now validate whether the second-stage distillation (*distillation by model agreement*) is needed. In Table 5, we also compare the results with the model agreement (shown as $\cap$) to the results without using any agreement (ϕ). We observe better performance with *model agreement* in all the cases on top of the single-model distillation which validates its utility. Results with $\eta = 100, Agreement = \cap$ can be considered as the tri-training (Ruder and Plank, 2018b) baseline.

4.2 Augmentation in Stages

Figure 2 presents the effect of different types of augmented data used by different epochs in our multi-epoch co-teaching framework. We observe that in every epoch, there is a significant boost in F1 scores for each of the languages. Arabic, being structural dissimilar to English, has a lower base score, but the relative improvements brought by UXLA are higher for Arabic, especially in epoch 2

η	Agreement	es	nl	de	ar	fi
Distillation by **clustering**						
0.7	$\cap$	82.28	83.25	78.86	52.64	78.47
0.5	$\cap$	82.35	83.11	78.16	54.20	78.28
Distillation by **model confidence**						
50%	$\cap$	**82.52**	82.46	75.95	52.00	77.51
	ϕ	81.66	82.26	77.19	52.97	77.77
80%	$\cap$	82.33	**83.53**	78.50	**54.48**	78.43
	ϕ	81.61	83.03	77.08	53.31	78.34
90%	$\cap$	81.90	82.80	**79.03**	52.41	**78.66**
	ϕ	81.21	82.77	77.28	52.20	77.93
100%	$\cap$	82.50	82.35	77.06	52.58	77.51
	ϕ	81.89	82.15	76.97	52.68	78.01

Table 5: Analysis of **distillation** on XNER. Results after epoch-1 training that uses $\{\mathcal{D}_s, \mathcal{D}_t'\}$.

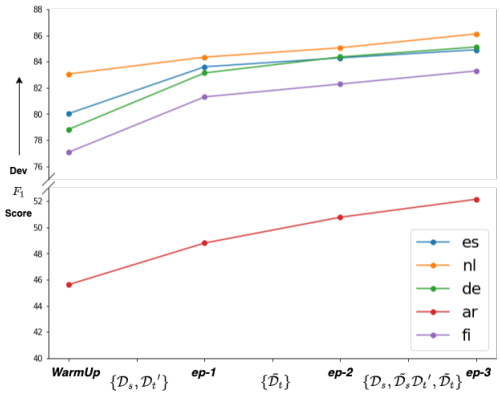


Figure 2: Validation F1 results in XNER for multi-epoch co-teaching training of UXLA.

Tgt lang	Zero shot + con-penalty	UXLA es	UXLA nl	UXLA de	UXLA ar	UXLA fi
en	**92.88**	**92.92**	**92.87**	**92.91**	**92.80**	**92.68**
es	81.42	**83.24**	82.01	77.71	80.29	81.97
nl	81.27	81.22	**85.32**	80.54	82.36	84.20
de	75.20	73.63	75.03	**80.03**	76.97	73.77
ar	50.88	52.66	53.08	52.52	**58.29**	53.80
fi	76.97	77.02	77.06	76.69	77.13	**80.11**

Table 6: **F1 scores** on XNER. Each column (*e.g.,* es) under **UXLA** represents results in all target languages for a UXLA trained with the augmented data in a specific language (*e.g.,* es). The **Zero shot+con-penalty** column represents the zero-shot results for the model after **WarmUp**.

when it gets exposed to the target language virtual data ($\tilde{\mathcal{D}}_t$) generated by the vicinity distribution.

4.3 Effect of Confidence Penalty & Ensemble

For all the three tasks, we get reasonable improvements over the baselines by training with confidence penalty (§2.1). Specifically, we get 0.56%, 0.74%, 1.89%, and 1.18% improvements in XNER, XNLI-5%, PAWS-X-5%, and PAWS-X-100% respectively (Table 1,3,4). The improvements in XNLI-100% are marginal and inconsistent, which we suspect due to the balanced class distribution.

From the results of ensemble models, we see that the ensemble boosts the baseline XLM-R. However, our regular UXLA still outperforms the ensemble baselines by a sizeable margin. Moreover, ensembling the trained models from UXLA further improves the performance. These comparisons ensure that the capability of UXLA through co-teaching and co-distillation is beyond the ensemble effect.

4.4 Robustness & Efficiency

Table 6 shows the robustness of the fine-tuned UXLA model on XNER task. After fine-tuning in a specific target language, the F1 scores in English remain almost similar (see first row). For some languages, UXLA adaptation on a different language also improves the performance. For example, Arabic gets improvements for all UXLA-adapted models (compare 50.88 with others in row 5). This indicates that augmentation of UXLA does not overfit on a target language. **More baselines, analysis and visualizations are added in Appendix.**

5 Related Work

Recent years have witnessed significant progress in learning multilingual pretrained models. Notably, mBERT (Devlin et al., 2019) extends (English) BERT by jointly training on 102 languages. XLM (Lample and Conneau, 2019) extends mBERT with a conditional LM and a translation LM (using parallel data) objectives. Conneau et al. (2020) train the largest multilingual language model XLM-R with RoBERTa (Liu et al., 2019). Wu and Dredze (2019), Keung et al. (2019), and Pires et al. (2019) evaluate zero-shot cross-lingual transferability of mBERT on several tasks and attribute its generalization capability to shared subword units. Pires et al. (2019) also found structural similarity (*e.g.,* word order) to be another important factor for successful cross-lingual transfer. K et al. (2020), however, show that the shared subword has a minimal contribution; instead, the structural similarity between languages is more crucial for effective transfer.

Older data augmentation approaches relied on distributional clusters (Täckström et al., 2012). A number of recent methods have been proposed using contextualized LMs (Kobayashi, 2018; Wu et al., 2018; Shi et al., 2019; Ding et al., 2020; Liu et al., 2021). These methods rely on labels to perform label-constrained augmentation, thus not directly comparable with ours. Also, there are fundamental differences in the way we use the pretrained LM. Unlike them our LM augmentation is purely unsupervised and we do not perform any fine-tuning of the pretrained vicinity model. This disjoint characteristic gives our framework the flexibility to replace θ_{lm} even with a better monolingual LM for a specific target language, which in turn makes UXLA extendable to utilize stronger LMs that may come in the future. In a concurrent work (Mohiuddin et al., 2021), we propose a contextualized LM based data augmentation for neural machine translation and show its advantages over traditional back-translation gaining improved performance in low-resource scenarios.

6 Conclusion

We propose a novel data augmentation framework, UXLA, for zero-resource cross-lingual task adaptation. It performs simultaneous self-training with data augmentation and unsupervised sample selection. With extensive experiments on three different cross-lingual tasks spanning many language pairs, we have demonstrated the effectiveness of UXLA. For the zero-resource XNER task, UXLA sets a new SoTA for all the tested languages. For both XNLI and PAWS-X tasks, with only 5% labeled data in the source, UXLA gets comparable results to the baseline that uses 100% labeled data.

References

Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. 2019. Unsupervised label noise modeling and loss correction. In *International Conference on Machine Learning (ICML)*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2020. Translation artifacts in cross-lingual transfer learning.

M Saiful Bari, Shafiq Joty, and Prathyusha Jwalapuram. 2020. Zero-Resource Cross-Lingual Named Entity Recognition. In *Proceedings of the 34th AAAI Conference on Artifical Intelligence*, AAAI '20, pages xx–xx, New York, USA. AAAI.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. 2019. Mixmatch: A holistic approach to semi-supervised learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 5050–5060. Curran Associates, Inc.

Olivier Chapelle, Jason Weston, Léon Bottou, and Vladimir Vapnik. 2001. Vicinal risk minimization. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 416–422. MIT Press.

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.

Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: evaluating cross-lingual sentence representations. *CoRR*, abs/1809.05053.

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.

Bosheng Ding, Linlin Liu, Lidong Bing, Canasai Kruengkrai, Thien Hai Nguyen, Shafiq Joty, Luo Si, and Chunyan Miao. 2020. DAGA: Data Augmentation with a Generation Approach for Low-resource Tagging Tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, EMNLP'20, pages 6045—-6057, Punta Cana, Dominican Republic. ACL.

Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 8527–8537. Curran Associates, Inc.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. 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 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

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 generalization. *CoRR*, abs/2003.11080.

Karthikeyan K, Zihan Wang, Stephen Mayhew, and Dan Roth. 2020. Cross-lingual ability of multilingual {bert}: An empirical study. In *International Conference on Learning Representations*.

Phillip Keung, yichao lu, and Vikas Bhardwaj. 2019. Adversarial learning with contextual embeddings for zero-resource cross-lingual classification and ner. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 452–457, New Orleans, Louisiana. Association for Computational Linguistics.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *Advances in Neural Information Processing Systems (NeurIPS)*.

Junnan Li, Richard Socher, and Steven C.H. Hoi. 2020. Dividemix: Learning with noisy labels as semi-supervised learning. In *International Conference on Learning Representations*.

Linlin Liu, Bosheng Ding, Lidong Bing, Shafiq Joty, Luo Si, and Chunyan Miao. 2021. MulDA: A Multilingual Data Augmentation Framework for Low-Resource Cross-Lingual NER. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, ACL'21, Bangkok, Thailand. ACL.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis,

Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Tasnim Mohiuddin, M Saiful Bari, and Shafiq Joty. 2021. Augvic: Exploiting bitext vicinity for low-resource nmt. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, Online. Association for Computational Linguistics.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada. Association for Computational Linguistics.

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. *CoRR*, abs/1701.06548.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Sebastian Ruder and Barbara Plank. 2018a. Strong baselines for neural semi-supervised learning under domain shift. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1044–1054, Melbourne, Australia. Association for Computational Linguistics.

Sebastian Ruder and Barbara Plank. 2018b. Strong baselines for neural semi-supervised learning under domain shift. *CoRR*, abs/1804.09530.

Erik Tjong Kim Sang. 2002. Introduction to the conll-2002 shared task: Language-independent named entity recognition. *CoRR*, cs.CL/0209010.

Erik Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *CoNLL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Linqing Shi, Danyang Liu, Gongshen Liu, and Kui Meng. 2019. Aug-bert: An efficient data augmentation algorithm for text classification. In *CSPS*.

Patrice Y. Simard, Yann A. LeCun, John S. Denker, and Bernard Victorri. 1998. *Transformation Invariance in Pattern Recognition — Tangent Distance and Tangent Propagation*, pages 239–274. Springer Berlin Heidelberg, Berlin, Heidelberg.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *The 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2012)*.

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.

Xing Wu, Shangwen Lv, Liangjun Zang, Jizhong Han, and Songlin Hu. 2018. Conditional BERT contextual augmentation. *CoRR*, abs/1812.06705.

Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. 2019. PAWS-X: A cross-lingual adversarial dataset for paraphrase identification. 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 3687–3692, Hong Kong, China. Association for Computational Linguistics.

Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. 2018. Fast and accurate reading comprehension by combining self-attention and convolution. In *International Conference on Learning Representations*.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*.

Zhi-Hua Zhou and Ming Li. 2005. Tri-training: exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17:1529–1541.

Appendix

A FAQ: Justifications for design methodology of UXLA

Here are our justifications for various design principles of the UXLA framework.

Is masked language model pre-training with cross-lingual training data from task dataset useful? In Table 7, We perform language model finetuning on XLM-R large model with multilingual sentences of NER dataset and perform adaptation with only English language. With the LM-finetuned XLM-R model, we didn't see any significant increase in cross-lingual transfer. For Spanish, Arabic language, the score even got decreased, which indicates possible over-fitting. However, robustness experiment in table 6 (see in the main paper, sec 4.4) indicates that our proposed method doesn't overfit on target language rather than augment the new knowledge base.

Model	es	nl	de	ar	fi
XLM-R	80.45	81.07	73.76	49.94	76.05
XLM-R + ens	81.42	81.27	75.20	50.93	76.97
UXLA	83.05	85.21	80.33	57.35	79.75
UXLA + ens	83.24	85.32	80.99	58.29	79.87
Finetuned XLM-R	78.11	81.61	76.33	48.04	76.63

Table 7: Some additional baseline results on XNER task. Here, ens reefers to emsemble.

Is using three models with different initialization necessary? Yes, different initialization ensures different convergence paths, which results in diversity during inference. Co-labeling (Section 3.3) utilizes this property. There could be some other ways to achieve the same thing. Our initial attempt with three different heads (sharing a backbone network) didn't work well.

Is using three epochs necessary? We utilize different types of datasets in different epochs. While pseudo-labeling may induce noise, the model's predictions for in-domain cross-lingual samples are usually better. Because of this, for a smooth transition, we apply the vicinal samples in the second epoch. Finally, inspired by the joint training of the cross-lingual language model, in the third epoch we use all four datasets. We also include the labeled source data which ensures that our model does not overfit on target distribution as well as persists the generalization capability of the source distribution.

Need for the combination of co-teaching, co-distillation and co-guessing? The combination of these helps to distill out the noisy samples better.

Efficiency of the method and expensive extra costs for large-scale pretrained models It is a common practice in model selection to train 3-5 disjoint LM-based task models (e.g., XLM-R on NER) with different random seeds and report the ensemble score or score of the best (validation set) model. In contrast, UXLA uses 3 different models and jointly trains them where the models assist each other through distillation and co-labeling. In that sense, the extra cost comes from distillation and co-labeling, which is not significant and is compensated by the significant improvements that UXLA offers.

B Visualization of confidence penalty

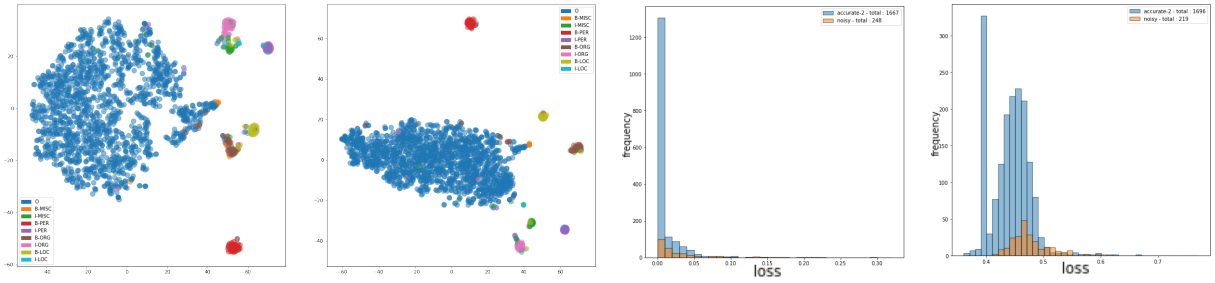
B.1 Effect of confidence penalty in classification

In Figure 3 (a-b), we present the effect of the confidence penalty (Eq. 1 in the main paper) in the target language (*Spanish*) classification on the XNER dev. data (*i.e.,* after training on English NER). We show the class distribution from the final logits (on the target language) using t-SNE plots. From the figure, it is evident that the use of confidence penalty in the warm-up step makes the model more robust to unseen out-of-distribution target language data yielding better predictions, which in turn also provides a better *prior* for self-training with pseudo labels.

B.2 Effect of confidence penalty in loss distribution

Figures 3(c) and 3(d) present the per-sample loss (*i.e.,* mean loss per sentence *w.r.t.* the pseudo labels) distribution in histogram without and with confidence penalty, respectively. Here, *accurate-2* refers to the sentences which have at most two wrong NER labels, and sentences containing more than two errors are referred to as *noisy* samples. It shows that without confidence penalty, there are many noisy samples with a small loss which is not desired. In addition to that, the figures also suggest that the confidence penalty helps to separate the clean samples from the noisy ones either by clustering or by model confidence.

Figures 4(a) and 4(b) present the loss distribution in a scatter plot by sorting the sentences based



(a) Without confidence penalty. (b) With confidence penalty. (c) Without confidence penalty. (d) With confidence penalty.

Figure 3: (a-b) Effect of training with confidence penalty in the warm-up step on target (*Spanish*) language XNER classification using t-SNE plots. From the visualization, it can be seen that the model trained with confidence penalty shows better inter-class separation which exhibits robustness of the multilingual model. (c-d) Histogram of loss distribution on target (*Spanish*) language XNER classification.

on their length in the x-axis; y-axis represents the loss. As we can see, the losses are indeed more scattered when we train the model with confidence penalty, which indicates higher per-sample entropy, as expected. Also, we can see that as the sentence length increases, there are more wrong predictions. Our distillation method should be able to distill out these noisy pseudo samples.

Finally, Figures 4(c) and 4(d) show the length distribution of all vs. the selected sentences (by *Distillation by model confidence*) without and with confidence penalty. Bari et al. (2020) shows that cross-lingual NER inference is heavily dependent on the length distribution of the samples. In general, the performance of the lower length samples is more accurate. However, if we only select the lower length samples we will easily overfit. From these plots, we observe that the confidence penalty also helps to perform a better distillation as more sentences are selected (by the distillation procedure) from the lower length distribution, while still covering the entire lengths. This shows that using the confidence penalty in training, model becomes more robust.

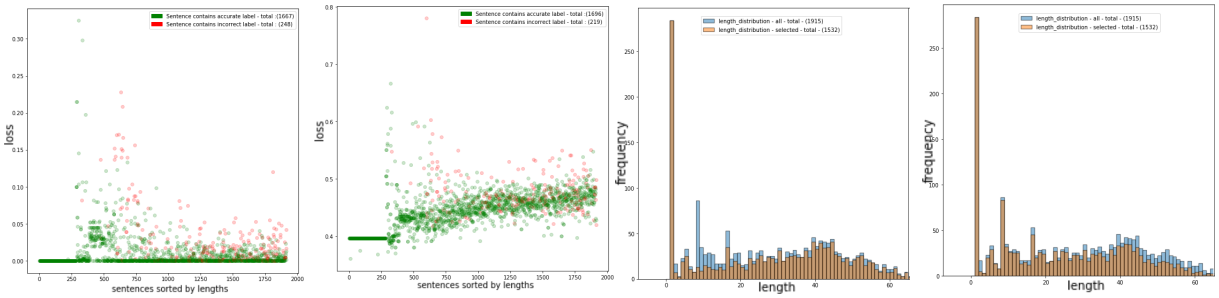
In summary, comparing the Figures 3(c-d) - 4(c-d), we can conclude that training without confidence penalty can make the model more prone to over-fitting, resulting in more noisy pseudo labels. Training with confidence penalty not only improves pseudo labeling accuracy but also helps the distillation methods to perform better noise filtering.

C Details on distillation by clustering

One limitation of the confidence-based (single-model) distillation is that it does not consider task-specific information. Apart from classifier confidence, there could be other important features that can distinguish a good sample from a noisy one. For example, for sequence labeling, *sequence length* can be an important feature as the models tend to make more mistakes (hence noisy) for longer sequences Bari et al. (2020). One might also want to consider other features like *fluency*, which can be estimated by a pre-trained conditional LM like GPT Radford et al. (2020). In the following, we introduce a clustering-based method that can consider these additional features to separate good samples from bad ones.

Here our goal is to cluster the samples based on their *goodness*. It has been shown in computer vision that deep models tend to learn good samples faster than noisy ones, leading to a lower loss for good samples and higher loss for noisy ones Han et al. (2018), Arpit et al. (2017). We propose to model *per-sample loss distribution* (along with other task-specific features) with a mixture model, which we fit using an *Expectation-Maximization* (EM) algorithm. However, contrary to those approaches which use actual (supervised) labels, we use the model predicted pseudo labels to compute the loss for the samples.

We use a two-component Gaussian Mixture Model (GMM) due to its flexibility in modeling the sharpness of a distribution Li et al. (2020a). In the following, we describe the EM training of the GMM for one feature, *i.e.,* per-sample loss, but it is trivial to extend it to consider other indicative task-specific features like sequence length or fluency score (see any textbook on machine learning).



(a) Without confidence penalty. (b) With confidence penalty. (a) Without confidence penalty. (b) With confidence penalty.

Figure 4: (a-b)Scatter plot of loss distribution on target (*Spanish*) language XNER classification. (c-d) Distribution of selected sentence lengths on target (*Spanish*) language XNER classification.

EM training for two-component GMM Let $x_i \in \mathbb{R}$ denote the loss for sample $\mathbf{x}_i$ and $z_i \in \{0, 1\}$ denote its cluster id. We can write the 1d GMM model as:

$$p(x_i|\theta, \pi) = \sum_{k=0}^{1} \mathcal{N}(x_i|\mu_k, \sigma_k)\pi_k \quad (3)$$

where $\theta_k = \{\mu_k, \sigma_k^2\}$ are the parameters of the k-th mixture component and $\pi_k = p(z_i = k)$ is the probability (weight) of the k-th component with the condition $0 \leq \pi_k \leq 1$ and $\sum_k \pi_k = 1$.

In EM, we optimize the *expected complete data* log likelihood $Q(\theta, \theta^{t-1})$ defined as:

$$\begin{aligned}
&Q(\theta, \theta^{t-1}) \\
&= \mathbb{E}(\sum_i \log[p(x_i, z_i|\theta)]) \\
&= \mathbb{E}(\sum_i \sum_k \mathbb{I}(z_i = k) \log[p(x_i|\theta_k)\pi_k]) \\
&= \sum_i \sum_k \mathbb{E}(\mathbb{I}(z_i = k)) \log[p(x_i|\theta_k)\pi_k] \\
&= \sum_i \sum_k p(z_i = k|x_i, \theta^{t-1}) \log[p(x_i|\theta_k)\pi_k] \\
&= \sum_i \sum_k r_{i,k}(\theta^{t-1}) \log p(x_i|\theta_k) + r_{i,k}(\theta^{t-1}) \log \pi_k
\end{aligned} \quad (4)$$

where $r_{i,k}(\theta^{t-1})$ is the responsibility that cluster k takes for sample $\mathbf{x}_i$, which is computed in the E-step so that we can optimize $Q(\theta, \theta^{t-1})$ (Eq. 4) in the M-step. The E-step and M-step for a 1d GMM can be written as:

E-step: Compute $r_{i,k}(\theta^{t-1}) = \frac{\mathcal{N}(x_i|\theta_k^{t-1})\pi_k^{t-1}}{\sum_k \mathcal{N}(x_i|\theta_k^{t-1})\pi_k^{t-1}}$

M-step: Optimize $Q(\theta, \theta^{t-1})$ *w.r.t.* θ and π

- $\pi_k = \frac{\sum_i r_{i,k}}{\sum_i \sum_k r_{i,k}} = \frac{1}{N}\sum_i r_{i,k}$
- $\mu_k = \frac{\sum_i r_{i,k} x_i}{\sum_i r_{i,k}}$; $\sigma_k^2 = \frac{\sum_i r_{i,k}(x_i - \mu_k)^2}{\sum_i r_{i,k}}$

Inference For a sample $\mathbf{x}$, its *goodness* probability is the posterior probability $p(z = g|\mathbf{x}, \theta)$, where $g \in \{0, 1\}$ is the component with smaller mean loss. Here, distillation hyperparameter η is the posterior probability threshold based on which samples are selected.

Relation with *distillation by model confidence* Astute readers might have already noticed that per-sample loss has a direct deterministic relation with the model confidence. Even though they are different, these two distillation methods consider the same source of information. However, as mentioned, the clustering-based method allows us to incorporate other indicative features like length, fluency, etc. For a fair comparison between the two methods, we use only the per-sample loss in our primary (single-model) distillation methods.

D Hyperparameters

We present the hyperparameter settings for XNER and XNLI tasks for the XLA framework in Table 8. In the *warm-up* step, we train and validate the task models with English data. However, for *cross-lingual adaptation*, we validate (for model selection) our model with the target language development set. We train our model with respect to the number of steps instead of the number of epochs. In the case of a given number of epochs, we convert it to a total number of steps.

We observe that *learning rate* is a crucial hyperparameter. In table 8, *lr-warm-up-steps* refer to the *warmup-step* from triangular learning rate scheduling. This hyperparameter is not to be confused with

Hyperparameter	XNER		XNLI		PAWS-X	
	Warm-up step	X-lingual adaptation	Warm-up step	X-lingual adaptation	Warm-up step	X-lingual adaptation
Training-hyperparameters						
model-type	xlm-r L	warm-up-ckpt	xlm-r L	warm-up-ckpt	xlm-r L	warm-up-ckpt
sampling-factor α	–	0.7	–	0.7	–	0.7
drop-out	0.1	0.1	0.1	0.1	0.1	0.1
max-seq-length	280	280	128	128	128	128
per-gpu-train-batch-size	4	4	16	16	16	16
grad-accumulation-steps	5	4	2	2	2	2
logging-step	50	50	50	25	50	25
learning-rate (lr)	$3e^{-5}$	$5e^{-6}$	$1e^{-6}$	$1e^{-6}$	$1e^{-6}$	$1e^{-6}$
lr-warm-up-steps	200	10% of train	10% of train	10% of train	10% of train	10% of train
weight-decay	0.01	0.01	–	–	–	–
adam-epsilon	$1e^{-8}$	$1e^{-8}$	$1e^{-8}$	$1e^{-8}$	$1e^{-8}$	$1e^{-8}$
max-grad-norm	1.0	1.0	1.0	1.0	1.0	1.0
num-of-train-epochs	–	1	–	1	–	1
UXLA-epochs	–	3	6	3	10	6
max-steps	3000	–	–	–	–	
train-data-percentage	100	100	5	5	5	5
conf-penalty	True	False	True	False	True	False
Distillation-hyperparameters						
#mixture-component	–	. 2	–	–	–	–
posterior-threshold	–	0.5	–	–	–	–
covariance-type	–	Full	–	–	–	–
distilation-factor η	–	80, 100, 100	–	50, 80, 100	–	80, 90, 80
distillation-type	–	confidence	–	confidence	–	confidence
Augmentation-hyperparameters						
do-lower-case	False	False	False	False	-	False
aug-type	–	successive-max	–	successive-cross	–	successive-cross
aug-percentage P	–	30	–	30	–	40
diversification-factor δ	–	3	–	2×2	–	2 × 2

Table 8: Hyperparameter settings for XNER, XNLI, and PAWS-X task. Total number of parameter for each of the model is 550M. We used V100 GPUs to do the experiments. Average run-time for each of the languages may differ based on total number of augmented samples. In an average, for per million augmentation requires .5-2 days based of various settings of training mechanism (ie., fp16 training, gradient accumulation etc).

Warm-up step of the UXLA framework. In our experiments, effective *batch-size* is another crucial hyperparameter that can be obtained by gradient accumulation steps. We fix the maximum sequence length to 280 for XNER and 128 tokens for XNLI. For each of the experiments, we report the average score of three task models, $\theta^{(1)}$, $\theta^{(2)}$, $\theta^{(3)}$, which are initialized with different seeds. We perform each of the experiments in a single GPU setup with *float32* precision.

E Additional Related Work

Vicinal risk minimization. One of the fundamental challenges in deep learning is to train models that generalize well to examples outside the training distribution. The widely used Empirical Risk Minimization (ERM) principle where models are trained to minimize the average training error has been shown to be insufficient to achieve generalization on distributions that differ slightly from the training data (Szegedy et al., 2014; Zhang et al., 2018). Data augmentation supported by the Vicinal Risk Minimization (VRM) principle (Chapelle et al., 2001) can be an effective choice for achieving better out-of-training generalization.

In VRM, we minimize the empirical vicinal risk defined as:

$$\mathcal{L}_v(\theta) = \frac{1}{N}\sum_{n=1}^{N} l(f_\theta(\tilde{x}_n), \tilde{y}_n) \tag{5}$$

where f_θ denotes the model parameterized by θ, and $\mathcal{D}^{\text{aug}} = \{(\tilde{x}_n, \tilde{y}_n)\}_{n=1}^{N}$ is an augmented dataset constructed by sampling the vicinal distribution $\vartheta(\tilde{x}, \tilde{y}|x_i, y_i)$ around the original training sample (x_i, y_i). Defining vicinity is however challenging as it requires to extract samples from a distribution without hurting the labels. Earlier methods apply simple rules like rotation and scaling of images (Simard et al., 1998). Recently, Zhang et al. (2018); Berthelot et al. (2019) and Li et al. (2020) show impressive results in image classification with simple linear interpolation of data. However, to our knowledge, none of these methods has so far been successful in NLP due to the discrete nature of texts.