Lexicon based Fine-tuning of Multilingual Language Models for Sentiment Analysis of Low-resource Languages

Anonymous ACL submission

Abstract
Massively multilingual language models (MMLM) such as mBERT and XLM-R have shown good cross-lingual transferability. However, they are not specifically trained to capture cross-lingual signals with respect to sentiment words. In this paper, we use a sentiment lexicon of a high-resource language in order to generate an intermediate fine-tuning task for the MMLM, when fine-tuning it for a low-resource sentiment classification task. We show that such a fine-tuning task improves the mapping between similar sentiment words in different languages and improves the sentiment classification task of the low-resource language.

1 Introduction
Massively multilingual language models (MMLMs) such as mBERT (Devlin et al., 2019) and XLM-R (Conneau et al., 2020) have shown very promising results in text classification, even for tasks in low-resource language (LRL) settings (Hu et al., 2020). However, even when fine-tuned with the same dataset sizes, results for languages that are well-represented in the models are superior to that of the languages that have a lower representation (Wu and Dredze, 2020). Moreover, the amount of fine-tuning data used in down-stream tasks is also a deciding factor (Wu and Dredze, 2020; Doddapaneni et al., 2021).

Sentiment lexicons have been used as additional knowledge sources to enhance sentiment classification capabilities of deep learning models such as CNNs (Shin et al., 2017) and RNNs (Kumar et al., 2018; Margatina et al., 2019; Ma et al., 2018). They have also been used to improve sentiment-aware representations in simple Transformers (Zhong et al., 2019) and monolingual pre-trained models (Ke et al., 2020a; Suresh and Ong).

However, the use of lexicon-based data augmentation is not always a viable option for LRLs, as many such languages may not have quality, manually curated lexicons.

This paper explores the possibility of using a high-quality sentiment lexicon belonging to a high-resource language (HRL) to improve MMLM performance in sentiment classification for LRLs. We introduce two following lexicon-based intermediate fine-tuning methods that use a HRL lexicon.

- We translate the terms in the HRL lexicon to the LRL using a publicly available machine translation system. The (possibly noisy) translations are paired with original lexicon terms in such a manner to create positive and negative samples, considering their valence.

- We prepend a set of phrases (hereafter referred to as Auxiliary Phrases (APs)) from the original HRL lexicon to each training data sample of the target LRL dataset. This augmented dataset creates a binary classification task; AP and data sample having the same sentiment is a positive instance, else, a negative instance.

In both methods proposed, after the intermediate fine-tuning step, the model is further fine-tuned with the 3-class classification dataset of the target language. The aim of both these intermediate fine-tuning methods is to provide an additional cross-lingual signal to the MMLM, on the relationship between sentiment words belonging to different languages. Our method can be considered analogous to Ke et al. (2020a)’s objective to include external knowledge, but we use lexicon-based fine-tuning in contrast to their pre-training. For both the methods, we assume that there exists a sentiment lexicon for a HRL that has valence scores for each lexicon term. For English, there are many such lexicons (Hutto and Gilbert, 2014; Mohammad, 2018; Nielsen, 2011). Our methods resemble...
a Continual Learning paradigm (Parisi et al., 2018; Ke et al., 2021, 2020b). We experiment with three LRLs; Sinhala, Tamil and Bengali, together with an English sentiment classification task. We observe improved results for sentiment analysis with our second method but with varying gains across languages. We show that the quality of the HRL lexicon is more important than the relatedness between the HRLs and LRLs. Interestingly, using a HRL lexicon is better than using a noisy lexicon from the same language, when creating APs. Our code will be publicly released.

2 Method

In our two methods, we create an intermediate fine-tuning task for the MMLM. This fine-tuned model is further fine-tuned with the original target language data. For the final fine-tuning, we use a randomly initialized 3-class classifier head (similar to the supervised fine-tuning we use in our baselines). The encoder layer is initialized with the encoder layer weights of the output model from the intermediate fine-tuning step.

2.1 Intermediate Fine-tuning with Bilingual Sentiment Word Phrases

Our first method is simple and straightforward. We use a sentiment lexicon from a HRL, where each term has a valence score. Valence score can be used as a measure of the sentiment of a word, where positiveness of a word increases as the valence score gets nearer to 1 and negativeness increases when the valence score gets nearer to 0 (Mohammad, 2018; Mehrabian, 1996).

We create a set of phrases that contains HRL terms and their corresponding translations. Each of these phrases gets labelled as 1, as they carry terms bearing a similar sentiment (e.g. - For Tamil; good Nalla[SEP]affection Pâcam[EOS]; <label=1>)1. We create another set of phrases labelled as 0, by combining original lexical terms with translated terms having a different sentiment (e.g. - good Nalla[SEP]toxic Naccu[EOS]; <label=0>, where English terms are paired up with a Tamil term with a dissimilar sentiment). We use the created phrases in a binary classification task and fine-tune the MMLM.

[1] We use transliterated Tamil words here to avoid font issues and to improve readability. But we use the words in their respective scripts in experiments. Translations are shown in Figure 4 in Appendix A

Figure 1: Creating APs from the lexicon. A neutral AP is considered as an example. Dotted boxes connected by blue arrows show example instances in respective steps. Dotted arrows represent inputs from lexicons/datasets.

2.2 Intermediate Fine-tuning with Augmented LRL Data

In method 2, we use a data augmentation method to prepare data for the intermediate fine-tuning task. First, APs are created considering the valence scores of the HRL lexicon (Figure 1). Then these APs are prepended to the original data samples of the target language. The MMLM is fine-tuned with this augmented data. Finally, the augmented dataset is discarded and fine-tuning continues only with the LRL dataset (see Figure 2).

We select sentiment words from the HRL lexicon as described in the first method. They are then converted into phrases by considering all the permutations of the selected words. To select the best APs, we use a separate MMLM fine-tuned on a 3-class (positive, negative, neutral) sentiment classification dataset of the same HRL. As shown in Figure 1, we filter the best AP(s) by feeding them to this fine-tuned MMLM and taking the phrases that give the highest positive output logit value. Each AP has a specific sentiment based on the words they contain. Our APs resemble a structure similar to “Universal Adversarial Triggers” (Wallace et al., 2019); however, we use sentiment words from a lexicon to create the APs whereas Wallace et al. (2019) create trigger phrases with a refined subset of the model vocabulary (with no reference to sentiment words).

An augmented dataset is created by prepending an AP to a data sample in the target language.

[2] We create the required fine-tuned model with our English dataset but it could be a different model fine-tuned on the same 3 classes. This fine-tuning is a one-time task.

[3] From experiments, we found that prepending provides
When the AP and the target language data sample have a similar sentiment, the augmented sample is labelled as 1, and 0 otherwise. An example using Sinhala is shown in Figure 2. There, terms in the AP contain neutral sentiments (i.e.-valence scores in (0.4,0.6) interval), which means the AP bears a neutral sentiment. The Sinhala phrases are translated as; “There’s more here than we know” and “This work should be given maximum punishment” which are labelled as neutral and negative (respectively) in the original dataset.

3 Experiments

3.1 Datasets and Lexicons

For English, we use the US Twitter Airline Sentiment dataset, which is a publicly available English 3-class sentiment analysis dataset. We use a 4-class (positive, negative, neutral, conflict) Sinhala sentiment dataset (Senevirathne et al., 2020) which consists of news comments extracted from news web sites, and remove the conflict class for our experiments. For Tamil and Bengali we use datasets released by Hande et al. (2021) and Islam et al. (2020), respectively (the Tamil dataset consists of code-mixed data samples as well).

We use the VAD sentiment lexicon (Mohammad, 2018) primarily as our sentiment word lexicon, which contains valence, dominance and arousal scores for a set of 20k English words and their translations for 102 other languages. We also experiment with VADER (Hutto and Gilbert, 2014) as well, which consists of 7520 sentiment words (including emojis) and their valence scores.

3.2 Training setup & Baseline

We perform a vanilla fine-tuning for multi-class classification on the XLM-R-base model for each target language on their respective sentiment dataset to get the baselines. We also experimented on zero-shot performance for each language dataset on XLM-R-base model fine-tuned on our English dataset. In method 1, we create a 26k binary dataset using all the sentiment words in the VAD lexicon, with 4 terms per data sample. For the next intermediate fine-tuning task, we prepend 50% (to have balanced number of data points in the two classes) of the original training sentences with APs having the same sentiment and the rest with APs having dissimilar sentiments. We average the results across 3 randomly initialized runs and report the macro averages of the F1 scores. (hyperparameters are given in Table 3 in Appendix A).

3.3 Results and Discussion

First method yields lower results than the baseline (macro-F1 69.33%) for Sinhala and (61.08%) for Tamil, thus we did not use this method for further experiments. In Table 1 a clear performance gain is visible for our second method, where English and Sinhala datasets show the highest gains with 2 APs (created with English terms) used per sentiment class. The following ablation experiments were carried out on Sinhala. We use 1 AP per class for the experiments shown in Table 2 as proper translations are not available for some terms. To observe the effect of the language (specifically, language relatedness) of the lexicon used to create APs, we create APs in different languages. We select XLM-R as our MMLM since it supports all the languages considered in our experiments.
created sentiment lexicons for Hindi, Tamil, and Bengali using the VAD English lexicon. Results are reported in experiment 1 of Table 2. Although Hindi and Bengali belong to the same language family as Sinhala, and Tamil is geographically co-located with Sinhala, the results are low compared to the English lexicon. We believe this is due to the higher representation of English in XLM-R compared to other languages (Hu et al., 2020). While translating APs to other languages translation errors can occur and the noisiness of these translations can be another reason.

To determine the importance of the valence score for creating APs, we create random APs by using randomly picked words from each valence score interval where we observe a result drop (experiment 2 in Table 2). Experiment 3 in Table 2 shows the results for the two different lexicons we used (VADER and VAD), where VAD performs well with the possible reason being that it contains more words, especially belonging to the neutral class.

To identify positive, neutral, negative words in the lexicon (as mentioned in section 2.2), we first manually define valence score intervals. We verify this manual selection by providing a set of APs in English to an English fine-tuned model and observing that the model predicts the expected sentiment classes (an example is shown in Appendix A).

We also conducted experiments to determine the optimal number of lexicon terms in an AP, and the number of APs used per sentiment class. Figure 4 in Appendix A shows a result drop as the amount of words per AP is increased, due to over-fitting of the model during fine-tuning. We do not experiment beyond 8 words per AP as it takes an excessive amount of time to process the permutations.7

With the proposed two intermediate fine-tuning methods, we expect to provide an additional cross-lingual signal to the MMLM via the APs. In order to verify it, we analyse the embeddings of individual sentiment words from Sinhala and English, with/without the intermediate fine-tuning step in method 2. We manually pick English and Sinhala positive/negative sentiment words. The visualization (process details are in Appendix A) in Figure 3 shows that Sinhala and English words with similar sentiment have been grouped much closer after the intermediate fine-tuning step, especially negative words.8 The circle markers in Figure 3 shows embeddings from a vanilla fine-tuned model (on Sinhala dataset) and triangle markers show embeddings from our approach. Red is for negative words and blue is for positive.

### 4 Conclusion

We proposed two cross-lingual fine-tuning methods on MMLMs for sentiment analysis of LRLs, based on a sentiment lexicon of a HRL. Out of these, fine-tuning on augmented data created from the HRL lexicon yielded noticeable improvements over vanilla fine-tuning. We showed that this result gain is due to the intermediate fine-tuning technique providing an additional cross-lingual signal to the MMLM to learn similarity between sentiment words belonging to different languages. Our solution was tested only for languages included in XLM-R. In future we will consider other models, and languages not included in them.

---

7 We use a single shared GPU for all experiments

8 Due to font issues, we show transliterated words in the graph, but use the words in their actual script for experiments

<table>
<thead>
<tr>
<th>AP attribute</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline - vanilla fine-tuning</td>
<td>69.61</td>
</tr>
<tr>
<td>1. APs in different languages</td>
<td></td>
</tr>
<tr>
<td>English</td>
<td>70.56</td>
</tr>
<tr>
<td>Sinhala</td>
<td>69.66</td>
</tr>
<tr>
<td>Tamil</td>
<td>70.28</td>
</tr>
<tr>
<td>Bengali</td>
<td>69.16</td>
</tr>
<tr>
<td>Hindi</td>
<td>69.73</td>
</tr>
<tr>
<td>2. Randomly selected APs</td>
<td>67.66</td>
</tr>
<tr>
<td>3. APs created with different lexicons</td>
<td></td>
</tr>
<tr>
<td>VAD Sentiment Lexicon</td>
<td>70.56</td>
</tr>
<tr>
<td>VADER</td>
<td>69.99</td>
</tr>
</tbody>
</table>

Table 2: Results for experiments with varying attributes of the APs for Sinhala sentiment dataset.
5 Ethical considerations

All the datasets and lexicons used in our experiments are either licensed under MIT license, NonCommercial-ShareAlike (CC BY-NC-SA 4.0) license or mentioned as free for research purposes. We utilize them solely for research work mentioned in this paper. Furthermore, we present a few examples for APs in Appendix A where negative APs may include derogatory terms. We do not intend any offense by them.

References


Varsha Suresh and Desmond C Ong. Using knowledge-embedded attention to augment pre-trained language models for fine-grained emotion recognition. In 2021 9th International Conference on Affective Computing and Intelligent Interaction (ACII), pages 1–8. IEEE.


A Appendix

Visualization of word embeddings

We look on how XLM-R word embeddings of several sentiment words change when our method is used. We choose positive and negative words in Sinhala and English, which are also present in our training data and the VAD lexicon. We perform a dimensionality reduction using Truncated Singular Value Decomposition (Truncated SVD) followed by t-SNE (van der Maaten and Hinton, 2008) to get 2D representations of original XLM-R embeddings for the words. For dimensionality reduction, we set a fixed random state (We use Scikit-Learn’s implementation) and try with different perplexity values for t-SNE in [1, 50] interval and choose the visualization producing the lowest Kullback-Leibler (KL) divergence after 1000 iterations (perplexity=10). We used the [CLS] token’s representation as the word vector. We select 16 words in both English and Sinhala and the transliterations/translations of selected Sinhala words are shown in Figure 6.

Table 3 shows hyperparameters and dataset sizes used for each baseline experiment in different languages. Epochs separated by a comma are for the intermediate fine-tuning task and the final 3-class classification task respectively. We use less epochs for Tamil than other languages, as we observe higher epochs tend to overfit Tamil dataset using our method. We use AdamW (Loshchilov and Hutter, 2018) for all fine-tuning tasks.

An example for selecting an AP based on the logit values

We choose the 3 most positive words from lexicon (e.g.- VADER); magnificently, ilu, aml and create permutations from them. The permutations

Table 3: Parameters used for each dataset for getting baseline results

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Train/Test</th>
<th>Epochs</th>
<th>Learning rate</th>
<th>Batch size</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>13176/1464</td>
<td>4, 3</td>
<td>5e-6</td>
<td>16</td>
</tr>
<tr>
<td>Sinhala</td>
<td>11833/1314</td>
<td>4, 3</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>Tamil</td>
<td>15694/1743</td>
<td>3, 2</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>Bengali</td>
<td>14853/3000</td>
<td>5, 4</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
</tbody>
</table>

Figure 4: macro-F1 score with varying number of APs per class and no. words per AP

---

9https://scikit-learn.org
are then fed into a fine-tuned model and the best is selected by the highest logit value output for positive sentiment class prediction. In this example, we expect negative, neutral and positive predictions at indexes 1, 2 and 3 respectively from the model output array. Hence, here we choose the 4th permutation in the list.

1. aml magnificently ilu: \[-2.0061314, -1.4377168, 3.1962798\]
2. aml ilu magnificently: \[-1.8522748, -1.5239806, 3.1405883\]
3. magnificently aml ilu: \[-2.0096264, -1.4296048, 3.1805775\]
4. magnificently ilu aml: \[-1.9999465, -1.4706941, 3.1985717\]
5. ilu aml magnificently: \[-1.6413125, -1.6105448, 3.0310764\]
6. ilu magnificently aml: \[-1.9787084, -1.4326444, 3.1722727\]

APs (top two in each sentiment class) created using the two lexicons

**VAD lexicon**
- Positive - very positive magnificent love happy, joyful greatness happiest happier
- Neutral - aardvark bluff bookseller token, mushroom rigging bowler sifting
- Negative - shit suffering died toxic, decayed pain murderer chaos

**VADER**
- Positive - magnificently ilu aml, euphoria ecstasy hearts sweetheart
- Neutral - borer skeptics %
- Negative - slavery raping rapist, murder rape kill terrorist

**Computational Resources**
For all our experiments, we used XLM-R-base model which contains 270M parameters. We utilized a single shared GPU (Nvidia Quadro RTX 6000 24GB). On average, it consumes ~0.4 hours for one randomly initialized run in an experiment.

- Affection – Pācam – பாசம்
- Good – Nalla – நல்லே
- Toxic – Naccu – தாக்கு

Figure 5: Translations of the Tamil words used for the example in Section 2.1

- hoňda – ගෝන්ඩා – Good
- vaťinā – ඒටිනා – Valuable
- kætayi – බටයි – Ugly
- naraka – නරක – Bad

Figure 6: Transliterations/Translations for Sinhala words used in Figure 3

- There is more here than we know - api dannavaṭa vadā deyak metana tiyanaṇa – අපි දැන්වනට සාදා දේයක මෙටන ස්ටියනා
- This work should be given maximum punishment - mē vādēṭa uparima daṇḍuvaṁ denna ōnē – මේ හදුවට අපරිම ආද්යමකදේන් වයන්නේ