Chapter 5

Multilingual Learning based Sequence Labeling

5.1 Introduction

POS tagging for low resource languages is challenging. Low resource languages raise a fundamental issue of machine learning (especially deep learning), namely that they lack sufficiently large datasets to train a machine learning model from scratch under complete supervision. Several approaches have been developed to learn representations for POS tagging for low resource languages. They can be broadly classified into three categories [88, 1, 25] unsupervised POS induction, cross-lingual transfer of features, or supervised induction from small labelled corpora or dictionaries. The proposed approach is based on cross-lingual transfer of features, where meta learning is utilized in a multilingual setup to learn these cross-lingual features.

Meta learning algorithms [172, 78] try to learn a parameter initialization that can be finetuned on a new task with a few gradient updates. These learned parameters store the experience collected by the model over the distribution of similar tasks. These algorithms have significantly improved results on few-shot learning for image classification in the supervised setting.

Meta learning approaches have given promising results on few shot image classifications. Finn et al. [78] have developed Model Agnostic Meta Learning (MAML), which produced promising results for few shot image classification on the Omniglot dataset [125] and the Mini-ImageNet dataset [199]. MAML is a second-order meta learning algorithm that requires the computation of the Hessian. Reptile is another meta learning algorithm, which is a first-order approximation of MAML. It has also given promising results on few-shot image classification. Building upon the success of meta learning approaches for computer vision, these algorithms are now being increasingly used in natural language processing. Deng et al. [61] have used meta learning for few-shot text classification. They used a pre-trained model like BERT [63] to learn task-agnostic contextualized features that capture linguistic information. These features were then exploited for meta learning-based model training and then tested on unseen text to perform few-shot text classification. Dou et al. [65] have applied meta learning to improve performance for low resource tasks on the General Language Understanding Evaluation benchmark [245]. They split the benchmark dataset into high resource tasks (auxiliary tasks) and low resource tasks (target tasks) and obtain significant improvement in performance on low resource tasks by training with meta learning algorithms like MAML and Reptile, which motivated us to apply meta learning algorithms to POS tagging for low resource languages.

Machine learning or Deep learning for low resource languages has always been challenging due to a lack of supervised data. Developing an automated POS tagging tool for low resource languages is essential since it is fundamental to several natural language processing tasks. Kann et al. [109] experimented with several state-of-the-art baselines for POS tagging and determined that model performance is lacking. They show that POS tagging for *truly* low resource languages is not a trivial task. Kann et al. [108] proposed a hierarchical BiLSTM based tagger with a character-based sequence to sequence model for low resource language POS tagging. They also augment their training with auxiliary tasks such as lemmatization to improve performance for low resource languages. Plank et al. [189] proposed to add an auxiliary loss function for rare words to improve performance on POS tagging. Cardenas et al. [31] proposed an unsupervised universal POS tagger and obtained state-of-the-art results on low resource languages with nearly no labelled data.

Unlike previous work, our approach uses meta learning to learn better representations for POS tagging of low resource languages

5.2 Contribution of Chapter

In this chapter, we train a multilingual POS tagger via meta learning for low resource languages. We formulate POS tagging for different languages as tasks to be fed into the meta learning algorithms. Later on, we finetune the learned model on the target task (previously unseen low resource language). This scenario address the question of *how to optimize a deep learning architecture's parameters for low resource language by utilizing annotated resources of other languages.* Meta learning enables POS taggers to learn cross-lingual representations which are transferable across languages, especially if the languages have a syntactic or semantic similarity. The model also captures language-agnostic contextual and lexical features. In my experiments we observe that the model trained with meta learning consistently surpasses a strong baseline trained from scratch on low resource languages, indicating this model's superiority over the baseline. The ablation study shows the extent to which a high resource language is required amongst the training languages. The obtained results suggest that having a high resource language helps the meta model learn better representations.

5.3 Methodology

The proposed method has two parts, the first being the POS tagging model and the second being the meta learning model. In the subsequent subsections, we describe these models.

5.3.1 Residual BiLSTM-CRF Model

I propose a novel Residual BiLSTM-CRF architecture, which closely follows the architecture proposed in Ma and Hovy [140] and Yang et al. [256], but unlike their architecture does not pass the character embeddings through the BiLSTM. An overview of the model

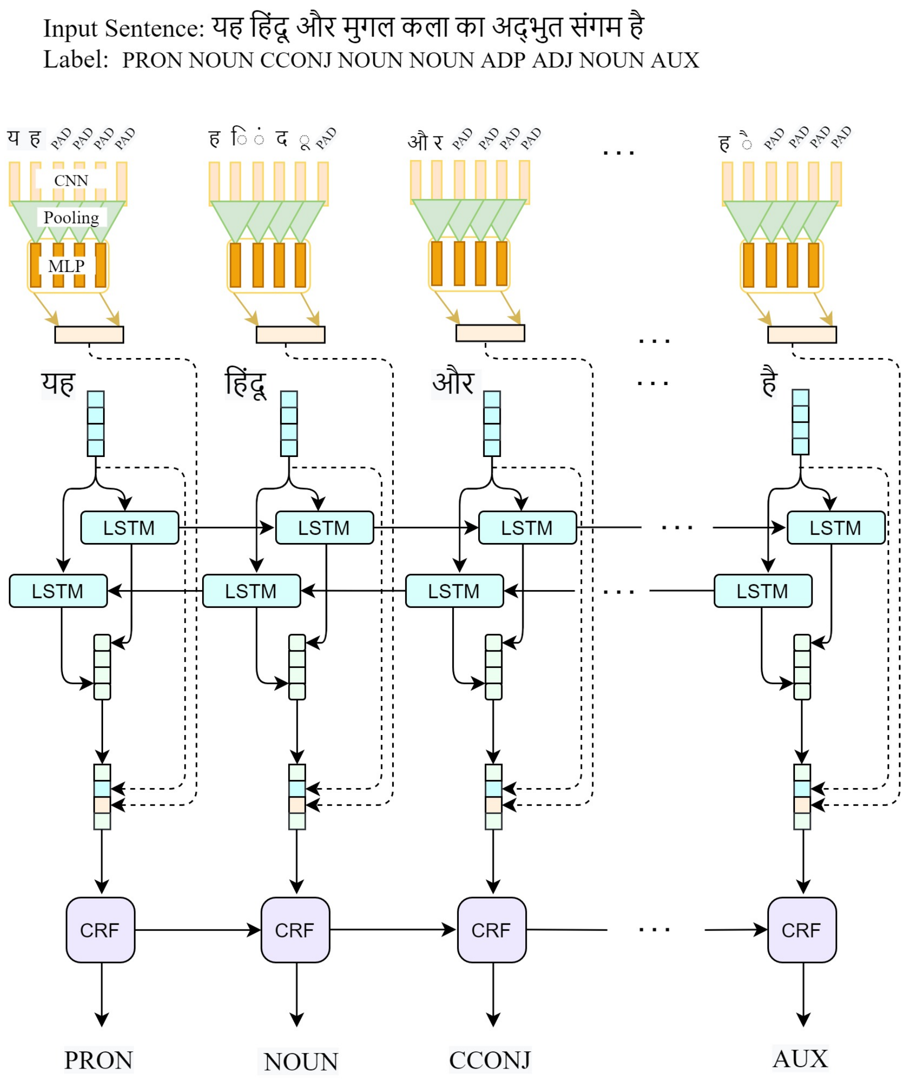
Input Sentence: यह हिंदू और मुगल कला का अद्भुत संगम है
Label: PRON NOUN CCONJ NOUN NOUN ADP ADJ NOUN AUX
CNN
Pooling
MLP
यह
हिंदू
और
है
LSTM
CRF
PRON
NOUN
CCONJ
AUX

FIGURE 5.1: Residual BiLSTM-CRF model for POS tagging

is given in Figure 5.1. Let the input sequence to be tagged be $w = \{w_1, w_2, \ldots, w_m\}$, where m is the length of the sequence. The corresponding pre-trained word embeddings are $e = \{e_1, e_2, \ldots, e_m\}$, where $e_i \in \mathbb{R}^{d_1}$, and d_1 is the size of the word embedding vector. These word embeddings are obtained using the word2vec model [155]. Embeddings for each language are trained on monolingual data and are kept fixed during training. To deal with out of vocabulary words (OOV's), character embeddings for each word have been exploited. These character embeddings are learned during the model training from scratch by a convolution neural network, similar to the one described by Kingma and Ba [116]. They are formed by applying layers of convolutional filters on the characters of the words followed by max-pooling over them.

Let the corresponding character embeddings for every input word be $c = \{c_1, c_2, \ldots, c_m\}$, where $c_i \in \mathbb{R}^{d_2}$, and d_2 is the size of the character embedding vector obtained via the character embedding model.

$$c_i = \text{CONV}(\text{char}(w_i)) \tag{5.1}$$

where CONV represents the CNN based character model and char(w_i) represents characters of w_i.

The obtained word embeddings are passed through a BiLSTM to obtain latent representations for every token in the input sequence. Let the hidden states obtained from the BiLSTM be $h = \{h_1, h_2, \ldots, h_m\}$ where $h_i \in \mathbb{R}^{2\times d_3}$, and d_3 is the size of the BiLSTM hidden state.

$$\{h_1, h_2, \ldots, h_m\} = \text{BiLSTM}(\{e_1, e_2, \ldots, e_m\}) \tag{5.2}$$

According to Yang and Xu [255], word embedding, BiLSTM representation and character embedding have been concatenated. The concatenated representation is passed through a single linear layer to obtain the emission (log) probabilities.

$$x_i = W[h_i; c_i; e_i] \tag{5.3}$$

Here, x_i represents the emission (log) probabilities. The single linear layer $W \in \mathbb{R}^{l\times(d_1+d_2+2\times d_3)}$ is a learnable projection matrix and l is the total number of POS tags. These emission probabilities are passed to the CRF for decoding the POS tag sequence. For the CRF, the probability of obtaining the POS tag sequence $y = y_1, y_2, \ldots, y_m$ conditioned on the input representation x is given by:

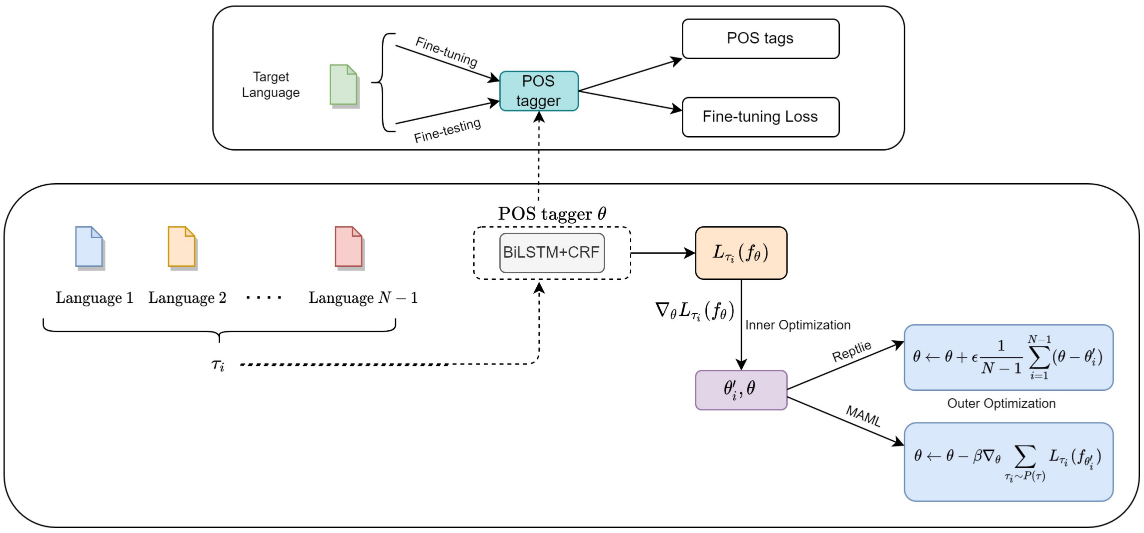

FIGURE 5.2: Meta learning based training for POS tagging

$$P(y|x) = \frac{\exp(\text{Score}(y,x))}{\sum_{y\prime} \exp(\text{Score}(y\prime,x))} \tag{5.4}$$

where

$$\text{Score}(y,x) = \sum_i x_i[y_i] + T_{y_{i-1}y_i} \tag{5.5}$$

$$L(\theta) = -\log(P(y|x)) \tag{5.6}$$

where θ represents all the model parameters. The goal of the model learning is to minimize $L(\theta)$, that is attained by two distinct meta learning algorithms.

5.3.2 Meta Learning Algorithms

Meta learning algorithms have given promising results on low resource datasets [65]. Here, we explore MAML and Reptile algorithm for meta learning which we use in these experiments to train the multilingual POS tagging model (model described in section 5.3.1), which is shown in Figure 5.2.

5.3.2.1 Model agnostic meta learning

Model Agnostic Meta Learning (MAML) and its variants learn parameters that are sensitive to changes in the task, such that the parameters can easily adapt to a previously unseen task sampled from a distribution of similar tasks $P(\tau)$ only with a few examples of the unseen task. Let f_θ represent the model function (i.e the Residual BiLSTM-CRF model) parameterized by model parameters θ. When adapting to task τ_i, the model parameters become θ'_i which are obtained via one or more steps of gradient descent on task τ_i.

$$\theta'_i = \theta - \alpha \nabla_\theta L_{\tau_i}(f_\theta) \tag{5.7}$$

Here, α is the learning rate for the gradient update (∇_θ) and L_{τ_i} is the task specific loss. The model parameters θ are trained by optimizing for the performance of $f_{\theta'_i}$ with respect to θ across all tasks sampled from $P(\tau)$. The meta learning objective can be written as:

$$\min{}_\theta \sum_{\tau_i \sim P(\tau)} L_{\tau_i}(f_{\theta'_i}) = \sum_{\tau_i \sim P(\tau)} L_{\tau_i}(f_{\theta - \alpha \nabla_\theta L_{\tau_i}(f_\theta)}) \tag{5.8}$$

The meta-optimization is performed over the model parameters θ, whereas the objective is computed using the updated model parameters θ'. The meta objective's goal is to optimize the model parameters so that the gain in accuracy is maximized on a new task when the model is finetuned with a few gradient steps for that task. The meta-optimization objective for updating θ across all the tasks is given as follows:

$$\theta = \theta \leftarrow \beta \nabla_\theta \sum_{\tau_i \sim P(\tau)} L_{\tau_i}(f_{\theta'_i}) \tag{5.9}$$

Here, β is the meta-optimization learning rate. From equation 5.9, the meta-optimization update step requires the computation of the Hessian and thus is a second-order meta learning algorithm. The inner updation step is outlined in Algorithm 5.1.

Algorithm 5.1 Model Agnostic Meta Learning (MAML)

```
Require: $p(\tau)$: distribution over tasks
Require: $\alpha$, $\beta$: learning rate hyperparameters
1: randomly initialize $\theta$
2: while not done do
3:     Sample batch of tasks $\tau_i \sim p(\tau)$
4:     for all $\tau_i$ do
5:         Evaluate $\nabla_\theta L_{\tau_i}(f_\theta)$
6:         Compute adapted parameters with gradient descent $\theta'_i = \theta - \alpha \nabla_\theta L_{\tau_i}(f_\theta)$
7:     end for
8:     Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\tau_i \sim P(\tau)} L_{\tau_i}(f_{\theta'_i})$
9: end while
```

5.3.2.2 Reptile

Reptile is a first-order meta learning algorithm. It updates the model parameters θ with the task specific parameters θ'_i obtained from equation 5.7 as follows:

$$\theta = \theta + \epsilon \frac{1}{n} \sum_{i=1}^{n} (\theta - \theta'_i) \tag{5.10}$$

where ϵ is the step size and n is the total number of tasks. The algorithm learns an initialization for parameters that adapts to a new task very fast. It tries to minimize the difference between the task-specific parameters and the model parameters to learn a parameter initialization for fine tuning, i.e., it assumes the gradient for the meta-updation step to be the difference between the task-specific parameters and the current model parameters. The one inner updation step of the Reptile algorithm is outlined in Algorithm 5.2.

Algorithm 5.2 Reptile

```
Require: $\epsilon$: step size
Require: $n$: total number of tasks
1: while not done do
2:     for $i = 1, 2 \ldots n$ do
3:         Evaluate $\nabla_\theta L_{\tau_i}(f_\theta)$
4:         Compute adapted parameters with gradient descent $\theta'_i = \theta - \alpha \nabla_\theta L_{\tau_i}(f_\theta)$
5:     end for
6:     Update $\theta = \theta + \epsilon \frac{1}{n} \sum_{i=1}^{n} (\theta - \theta'_i)$
7: end while
```

5.4 Dataset Description

The dataset used for our experiments has been borrowed from the Universal Dependencies Treebank[1]. To evaluate the model's ability to capture task-specific morphological and contextual features and cross-lingual features, a total of 9 languages have been used for the experiments. The languages and the number of examples in each of them can be found in Table 5.1.

The initial 5 languages in Table 5.1 belong to the Devanagari script and have lexical and grammatical similarities, whereas the remaining 4 languages belong to the Latin script and have linguistic similarities amongst themselves [93]. Moreover, German and Sanskrit, both have similar grammar and sentence structures [107]. Even for languages such as English, Dutch and Danish, where abundant data is available, we only sample out a few examples to test in the low resource setting. The dataset size for German and Hindi is kept relatively large to learn better character embeddings for each script. Using multiple scripts helps us understand whether meta learning can learn cross-script features.

TABLE 5.1: Data statistics

Language	**Number of samples**
Bhojpuri	357
Hindi	13304
Marathi	446
Sanskrit	230
Magahi	551
English	78
German	18434
Dutch	875
Danish	565

5.5 Experimental Details

The BiLSTM has a single layer and a hidden size of 256. The word embeddings for every language are learned from the language's monolingual corpus using the word2vec

[1]https://universaldependencies.org/

model [155]. For character embeddings, each character of every token is first represented by a 17-dimensional vector. One-dimensional convolution filters (of size 2, 2, 3, 3, each with 64 filters) are then run over the character embeddings independently (not sequentially), followed by max-pooling over them. As a result, we obtained 4 vectors of size 64 for each token, which are concatenated to form a 256 sized vector to the token. Other than the convolution layer parameters, all parameters are initialized with Xavier initialization [83]. The convolution layer parameters are randomly initialized. The value of ϵ used for all experiments is 0.1. For all experiments the value of β is 0.01 and that of α is 0.01. The inner gradients for MAML are computed using Stochastic Gradient Descent, and the meta-optimization update is done with Adam [116]. For Reptile, inner gradients are computed using Adam with a learning rate of 0.01.

I conduct experiments on few-shot learning [78] before testing the proposed model on low resource languages to verify whether meta learning is able to capture cross-lingual features. Moreover, few shot learning settings are an extreme case of low resource setting.

For few shot POS tagging we employ the N-way K-shot setting [78]. Here, N languages are used for testing and the rest are used for training. We experiment with 2-way 5-shot learning, 2-way 1-shot learning, 4-way 5-shot learning and 4-way 1-shot learning.

Training: Out of the N languages used for testing $N/2$ languages belong to Devanagari script and the other $N/2$ languages belong to Latin script. To train the multilingual meta model via MAML M languages are randomly sampled out of the $9-N$ languages available for training in step (3) of Algorithm 5.1 (i.e $\tau_i = M$). K examples from each of the M languages are used to update parameters in the inner loop in steps (5) and (6) of the algorithm. Another set of K examples are chosen from each of the M languages to update the model parameters in the meta optimization step in step (8) of the algorithm. For Reptile all the $9-N$ languages are used in step (2) of Algorithm 5.2 (i.e $n = 9-N$). K examples from each of the $9-N$ languages are used to calculate task specific parameters at step (4) of the algorithm. Unless otherwise specified $M = N$ for all our experiments.

Evaluation: K examples from each of the N test languages are given to the multilingual meta model for finetuning after being trained in the above fashion. The model now is evaluated on its ability to perform POS tagging on examples from these N test languages.

For low resource POS tagging, a multilingual meta model trained the same way as for few shot POS tagging is used. The only difference being N now becomes equal to one, i.e. all languages except the target language are used for training. The multilingual meta model is now finetuned on the low resource language's training data before evaluating it on its test data.

For every language 10% of instances are randomly chosen for testing and another 10% are chosen for validation. All languages except Hindi and German are used for the low resource language setting experimentation. A larger dataset for Hindi and German helps the model learn better representations for the two scripts, also in the results section we show that having at least one language with a large number of examples is crucial to the multilingual meta model's performance on the low resource languages. All multilingual meta models for few shot learning are trained for 600 iterations. Each few shot learning experiment is repeated five times where the $N/2$ languages from each script are resampled. We present the average accuracy and the standard deviation over the five runs. Each low resource setting finetuning experiment is run five times, and for every run the validation set and the test set are resampled. We present the average test set results for these five runs.

5.6 Results and Discussion

5.6.1 Results for Few Shot POS Tagging

The results for few shot POS tagging have been summarized in Table 5.2. MAML outperforms Reptile by nearly 20 points for the 2-way K-shot POS tagging task and outperforms Reptile by 16 points for the 4-way K-shot POS tagging task. Reptile being a first-order meta learning algorithm gives significant speedup over MAML. However, there a significant drop in performance since it does not compute the Hessian and approximates the gradient to $\theta - \theta'$ which may not be optimal.

TABLE 5.2: Results for few shot POS tagging

Meta learning algorithm	**2-way accuracy**		**4-way accuracy**	
	1-shot	**5-shot**	**1-shot**	**5-shot**
MAML	73.50 ± 2.08%	73.90 ± 0.87%	61.41 ± 2.17%	64.52 ± 0.86%
Reptile	48.14 ± 0.21%	54.70 ± 0.57%	47.23 ± 0.64%	48.59 ± 1.52%

The results obtained for few shot POS tagging with MAML in the 2-way 5-shot setting are comparable to the average accuracy[2] (86.04%) obtained by training a powerful Yang et al. [256] model[3] using NCRF++ framework[4] on the nine languages from scratch. This validates that meta learning can capture cross-lingual features even in the low resource setting, results for which are elaborated in the subsequent section.

5.6.2 Results for Low Resource POS Tagging

Since the performance of MAML for few shot POS tagging is significantly better than Reptile, unless otherwise stated all our multilingual meta models for low resource POS tagging are trained with MAML. Also, since 2-way 5-shot models perform better than the other three variants the multilingual meta model is pretrained in this setting.

TABLE 5.3: Results for low resource POS tagging

Model	**Sanskrit**	**Marathi**	**Bhojpuri**	**Magahi**	**English**	**Dutch**	**Danish**
Multi. meta model*	75.92	82.65	**89.82**	**89.28**	89.16	87.42	83.50
Multi. meta model†	**76.41**	**87.24**	89.29	88.39	**90.91**	**90.94**	84.68
Residual BiLSTM-CRF	72.92	77.39	89.18	88.39	88.94	89.30	87.84
NCRF++	74.88	87.05	87.25	84.07	90.14	90.29	**88.44**
BiLSTM-LAN	61.61	76.70	82.15	77.17	86.60	88.18	70.30

* Our proposed model pretrained irrespective to script
† Our proposed model pretrained only on the same script

Results on the test set for this section are summarized in Table 5.3. A separate Residual BiLSTM-CRF model was trained from scratch with the low resource language's dataset for comparison. For Marathi, the Residual BiLSTM-CRF attains maximum accuracy of

[2]We average out accuracy for all languages except Hindi and German since sufficiently large datasets have been used for these languages

[3]We use the default hyperparameters and model configuration settings provided by the authors for their best performing model

[4]https://github.com/jiesutd/NCRFpp

77.39%, whereas the finetuned multilingual meta model obtains an accuracy of 82.65%, which is an improvement of five points over the Residual BilSTM-CRF. It indicates that the multilingual meta model has captured task-specific information that only needs finetuning in the low resource language setting. These result also supports the argument that meta learning aids in learning cross-lingual representations. In contrast, the model trained from scratch needs more data to learn the task-specific and the language-specific features.

In case of Marathi, it being a language that uses the Devanagari script in the training set makes it easier for the multilingual meta model to adapt to the language-specific features. The performance for Marathi increases by another five points when only Devanagari script languages are used during multilingual meta model training. A similar observation is made for Sanskrit, the baseline trained from scratch gives a maximum accuracy of 72.92% whereas finetuning the multilingual meta model gives a maximum accuracy of 75.92% which is a three point increase over the model trained from scratch. Using only languages with Devanagari script further boosts the performance by one point. On the other hand, Bhojpuri and Magahi give a better improvement in performance when finetuned on the multilingual meta model trained with languages from both the scripts. This acts as empirical evidence to justify my claim that there is cross-script knowledge transfer as well.

For English and Dutch, similar to Marathi, using only languages from with the same script (i.e Latin) gives the best results. In fact, for Dutch the baseline Residual BiLSTM-CRF does better than the multilingual meta model pretrained on all languages. For Danish we observe that baseline Residual BiLSTM-CRF model does better than both the multilingual meta models. A possible reason for this could be that Danish being a north-Germanic-Scandinavian language differs in some structures from German, English and Dutch which are essential for POS tagging. Thus using these languages for pretraining adds noise from which the model is unable to recover even after finetuning.

I also compare the model against two other baselines, namely Yang et al NCRF++ [256] and the BiLSTM-LAN [52]. For both BiLSTM-LAN and NCRF++ framework we use the hyperparameters reported in the papers for the best performing models. In the case of Bhojpuri and Magahi, both, the baseline and the multilingual meta model, outperform the NCRF++ model, demonstrating the effectiveness of them. For Marathi, Sanskrit, English, Dutch and Danish, NCRF++ outperforms Residual BiLSTM-CRF model. However, in the case of Sanskrit, Marathi, English and Dutch the meta model trained with only Devanagari script languages outperforms the rest of the models demonstrating the performance gain achieved with meta learning for low resource languages. For Danish NCRF++ outperforms all the models. All our models, except the multilingual meta model pretrained on all languages for Dutch, outperform BiLSTM-LAN. This indicates that a CRF network, missing in the BiLSTM-LAN, can be crucial for POS tagging for low resource languages.

5.6.3 Ablation Study

Here we study the importance of having a high resource language in the multilingual meta model's training set. Since Marathi and Sanskrit are affected the most by meta learning we use these languages for the ablation study. We only pretrain the multilingual meta model with languages with Devanagari script since it has already been established that this variant works better for Marathi and Sanskrit. We start with 1000 examples for Hindi and increase the number of examples by 1000 for each new run of MAML algorithm. The result for this are presented in Figure 5.3 and Figure 5.4 for Sanskrit and Marathi, respectively. The performance for both the languages increases as the number of examples for Hindi are increased, but saturates after about 6000 examples. This shows that having a high resource language in the meta pretraining stage is crucial for the POS tagger's performance. The results also suggest that increasing the number of examples for the high resource language beyond a certain saturation point during the pretraining phase does not further improve the final model's performance on the low resource language.

5.6.4 Analysis

As stated earlier, MAML outperforms Reptile in all our experiments. A possible reason for this could be the inability of Reptile to capture cross-lingual features and cross-script features. The gradient direction used by Reptile may not be a good enough approximation to learn parameters for POS tagging. Reptile does not capture cross-lingual features and

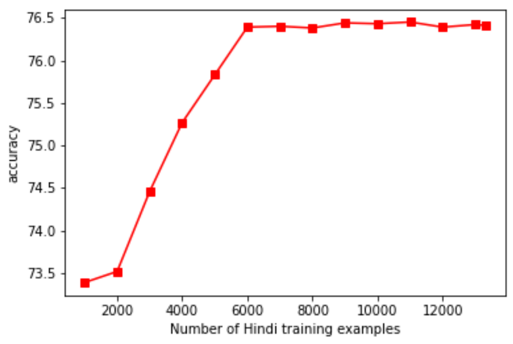

FIGURE 5.3: Sanskrit POS tagger's performance as a function of the number of Hindi training examples used in the meta pretraining stage

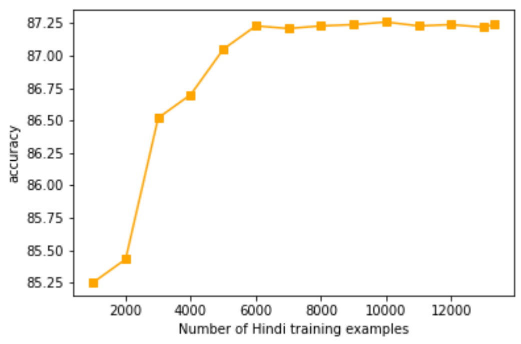

FIGURE 5.4: Marathi POS tagger's performance as a function of the number of Hindi training examples used in the meta pretraining stage

adds noise into the model parameters when finetuned with Marathi, Sanskrit, Bhojpuri and Magahi. Therefore, it does not outperform the baseline models trained from scratch.

To verify the importance of character embeddings, we experimented without including the characters embeddings. The performance of the Residual BiLSTM-CRF model and the multilingual meta model dropped by nearly 25 points on Marathi for low resource POS tagging. It indicates that character embeddings capture relevant morphological and semantic features which are not captured by the pretrained word embeddings. A similar performance drop was observed in few shot POS tagging (when tested on Marathi and Magahi for 2-way 5-shot learning with both MAML and Reptile). The concatenation of word embeddings to the BilSTM hidden states proves to be beneficial as well. The hidden states contain contextual information, and the word embeddings and the character embeddings have lexical and semantic information for each token. Moreover, the character embeddings take care of OOV's.

I also experimented by replacing the BiLSTM with a self-attentive encoder [242]. There was no significant improvement in performance in the low resource setting and for few shot POS tagging. This is probably because the sentences in the training set are relatively small in size for which the BiLSTM suffices. The ablation studies indicate that having a high resource language from the same script while training the multilingual meta model is crucial, without which there is a significant drop in performance.

5.7 Summary

This chapter explored the application of meta learning algorithms, MAML and Reptile on the proposed Residual BiLSTM-CRF model to low resource languages' POS tagging. Results obtained by MAML for 2-way few shot POS tagging and 4-way few shot POS tagging show that the idea of meta learning has immense scope for multilingual few shot learning. Based on few shot POS tagging success, it was extended to test the multilingual meta model in the low resource language setting, where it provides competitive results.