Transfer Learning for Low Resource Translation on the FLoRes Dataset

Anonymous ACL-IJCNLP submission

Abstract

In recent years, neural machine translation models have had some success at the task of machine translation, with the most successful models even achieving comparable results to more traditional machine translation models. However, most of these models rely on a large amount of parallel training data, which is not available for the vast majority of languages in the world. In order to combat this difficulty, we apply transfer learning, in which a suitable model is trained on a high-resource language pair, before its parameters are used to initialize a model on a low-resource language pair, which is then trained, in the hope that some of the learned aspects of the two "parent" languages can be applied to improve translation quality between the low-resource "child" languages.

We train a parent model on the relatively large IIT Bombay Hindi-English parallel corpus. The model is a GRU-based encoder-decoder model, with a Bahdanau attention mechanism. This model is then used to intialize an identically-structured model for either Nepali-English or Sinhala-English translation. These the FLORES dataset, a corpus of a relatively small amount of data for translation between either Nepali or Sinhala and English. This dataset was created with the intent of being used for new work in low-resource machine translation. We do not achieve state-of-the-art results, but our findings are indicative of some of the behaviors and challenges of transfer learning and low-resource machine translation.

1 Introduction

Machine translation is a prominent problem in natural language processing, and one of the approaches to this problem is neural machine translation. In particular, the encoder-decoder framework has been very successful with the BERT model reaching a GLUE score of 80.5%. However, one of the most important factors in the framework's and in general neural machine translation's success is having large amounts of data, and this leads to the problem of low-resource machine translation.

As seen in Zoph et al., vanilla neural machine translation even with attention does much worse than string-to-tree statistical method on low-resource problems, but when used with the transfer learning method introduced in the paper, the NMT methods improve significantly and do almost as well if not better than the statistical methods.

The transfer learning method used in the Zoph et al. paper starts by training a parent model on a high resource language pair, in the paper's case French-English, and then using the trained model's parameters to initialize a child model which trains on the low resource language pair with the same target language. However, the child model does not train all parameters, but instead fixes any parameters that seem to be useful in general and not only to the parent model.

Due to how transfer learning works, the embeddings of the target language remain the same, but the words of the child source language are randomly mapped to parent source language embeddings. These randomly mapped embeddings are also modified over the training process.

In this report, we will be exploring using transfer learning for low resource machine translation on the FLORES dataset which contains parallel data for Nepali-English and Sinhala-English.

2 Background and Related Works

Here we cover the background of the problem and methods used that we will be discussing for the rest of the paper as well as any related works.

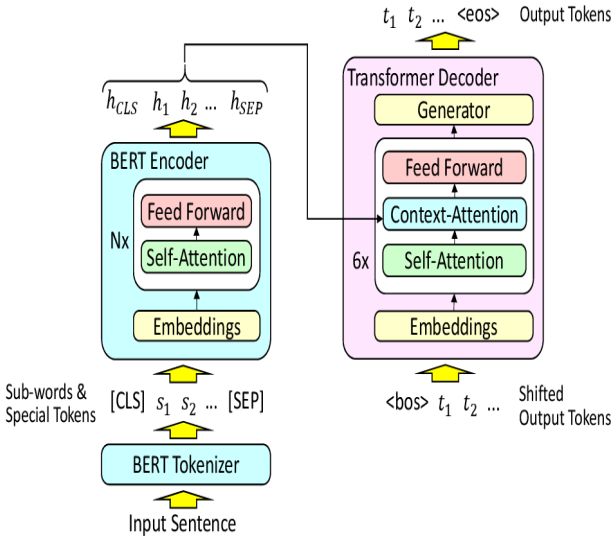

Figure 1: Machine Translation Architecture with BERT

2.1 Low-Resource Machine Translation

Before discussing low-resource machine translation, we first want to formulate the traditional neural-based machine translation task. Given a sequence of input

$$(x_1, x_2, x_3, \ldots, x_n)$$

. , where x_i is word in the source language for $i \in [1, n]$.
We want to generate another sequence of outputs

$$(y_1, y_2, y_3, \ldots, y_m)$$

, where y_i is word in the target language and m not necessarily equals n.
The loss function L is computed using cross-entropy loss.

$$L = -\sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j}\log(p_{i,j})$$

where $y_{i,j}$ represents the label of the ith word in the true output sequence, and $p_{i,j}$ represents the predicted probability of the ith word being the jth word in the whole vocabulary.
For the output of the neural based model, we generate a sequence of probability distribution of the output, and then decode the probability based on different decoding methods, such as greedy decode and beam search decode.

Whiuencle people have genher qerally been successful building models for machine translation (both rule-based and neural), the construction of adequate models, especially in the neural machine translation case, relies on a large amount of data. Neural models struggle with learning adequate representations of the rules necessary to translate between languages, and with smaller amounts of data there is a large risk of overfitting. Due to the large number of human languages, most of which are not spoken by hundreds of millions of people, translation between the majority of pairs of languages can be classified as low-resource. As such, low-resource machine translation represents a fairly significant problem in improving global human communication.

Due to the lack of verifiable translations to or from less commonly spoken languages, it is somewhat difficult to find even relatively small datasets of supervised examples. Recently, a dataset of parallel examples between four low-resource languages and English was created with the intent of fostering research in low-resource machine translation (Guzmán et al., 2019). Creating translation models using only these datasets is quite challenging due to their size, so we seek to achieve better results by training models on larger datasets between high-resource languages and applying a powerful tool known as transfer learning to get effective models for translating between the low resource languages.

2.2 Transfer Learning

Humans tend to learn new tasks more effectively when they are related to previously encountered tasks as we are able to recognize similarities between the tasks and apply relevant experience, which is the intuition that forms the basis of transfer learning in a machine learning context (Torrey and Shavlik, 2009). Transfer learning techniques have shown promise in applications such as document classification and speech processing (Wang and Zheng, 2015), especially in reducing computational cost and data requirements (Kunze et al., 2017).
Traditionally, transfer learning could utilize our domain knowledge of connections between different machine learning tasks, and generalize one model to a different task by constraining the parameters to be similar with the parameters in the original model through loss function.

$$L' = L + \lambda\sum_{i}(w_i' - w_i)^2$$

, where L represents the training loss of the child model without regularization, and w_i' and w_i represent the corresponding parameter in child model and the parent model.

By adding regularization, we can limit the search space of our child model, so it would take less time and data to make our child model converge.

Zoph et al. were the first to apply transfer learning to low-resource machine translation, and found significant improvements to BLEU score for Hausa-English, Turkish-English, Uzbek-English, and Urdu-English, all using a French-English parent model (Zoph et al., 2016). Their findings not only confirmed the viability of transfer learning approaches for low-resource translation, but also suggested the idea that choosing "better" parent languages can improve results for low-resource target languages.

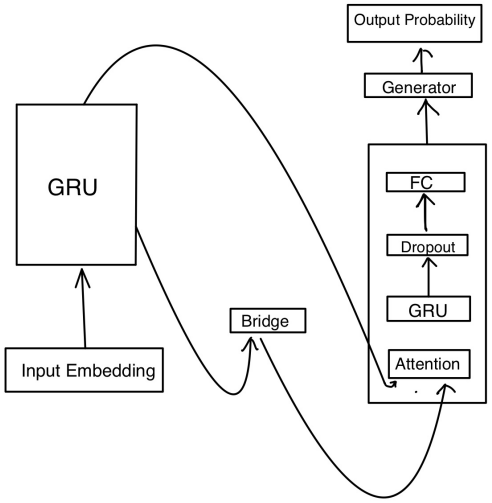

Figure 2: Model Architecture

3 Experimentation Overview

Now we will go into the details of the design of our datasets and model as well as explain our experimentation procedure.

3.1 Preprocessing

Hindi-English parallel data (approximately 1.5 million training examples) was obtained from the IIT Bombay Corpus, while Nepali-English and Sinhala-English parallel data were obtained from the FLoRes dataset. Each parallel data set was first tokenized (word tokens) by splitting the text by spaces and also separating punctuation from any word. Vocabularies were constructed based on word frequencies in the training sets: for example, our Hindi vocabulary is composed of the words which appear greater than (25?) times in the Hindi training examples, yielding a vocabulary of approximate size 30,000. To trim down the datasets and vocabulary into more manageable sizes, all sentences with more than 50 tokens were removed and less common words in the vocabulary had all their occurrences replaced with `<unk>` markers.

3.2 Model Architecture

We use the bidirectional Encoder-Decoder framework with attention for both the parent and child models in our transfer learning approach to machine translation. Each word in the source and target language is mapped to a word embedding of length 256. The encoder then takes these embeddings as input to a 2 layer bidirectional GRU with a hidden size of 256. Then, the final encoder vector is used to initialize the hidden state for the decoder, which is a 2 layer unidirectional GRU with a hidden size of 256. The decoder predicts words in the target language one step at a time.

3.3 Training

The model is trained for 10 epochs with a batch size of 128. While training, we use the ground truth target words at each time step as input to the decoder step, instead of using the output from the previous decoder step. This is called teacher forcing.

We observe that the validation perplexity does not consistently decrease over the epochs. In spite of this, we later found that the BLEU score increased over the epochs. We noticed that the particular validation set we used was small in size, which could have led to more variance in perplexity. Regardless, the translated examples increased in quality over the epochs, so we still felt confident in the model's training.

3.4 Decoding

Once the model is trained, its outputs must be decoded and converted into predicted word sequences from the target language. We introduce two schemes to do so: a "greedy" decoder and a "beam search" decoder. The greedy decoder simply selects the word with maximum probability predicted by the decoder at each time step, while the beam search decoder keeps the top k candidate words at each time step, then continues the search

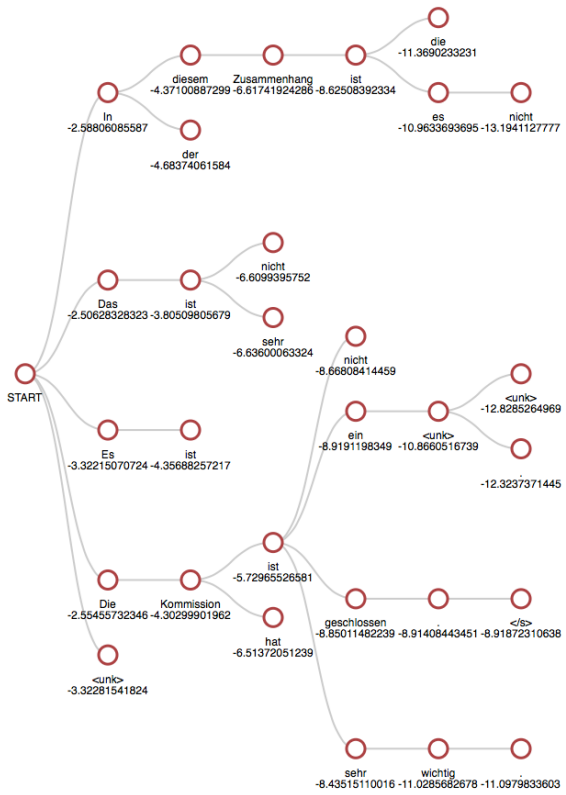

Figure 3: Beam search visualization (OpenNMT docs)

along each of the candidate "beams," accumulating scores across time steps for each path along the search space. A visualization of the beam search algorithm can be seen in Figure 2.

Each scheme presents advantages and disadvantages. The greedy approach is simple and computationally efficient, but can miss a global optimum sequence that has its probability concentrated toward its later words, since it can be "hidden" behind a low-probability word in the beginning of the sequence that will not be greedily chosen. Conversely, beam search is more likely to find the highest probability sequence because it considers more candidates, but it comes at a cost of decoding speed, especially as the number of beams k increases. let n be the longest length of our output sequence. The time complexity for greedy decoding algorithm is $O(n^2)$, because finding the most probable index should also take linear time. For the beam decoding algorithm, it would take $O(n^2k)$ time to run with using efficient implementation like quicksort. Therefore, when k gets really large, the time required for beam search will be much higher. The space complexity for beam search is also $O(nk)$, since each time we are strong k most probable solutions. Therefore, beam search decoding is not the best option for inference, but during training, we don't need to worry too much about the space and time it takes. We have implemented both the greedy decoder and the beam search decoder (for which we use $k = 4$ and $k = 10$) in order to compare their performances.

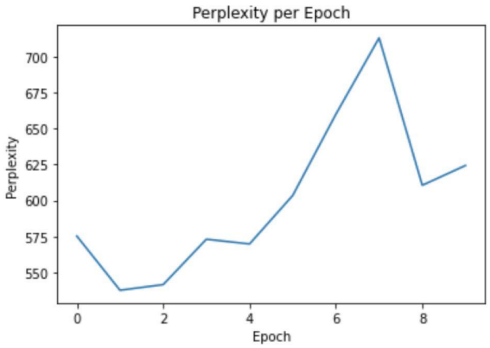

Figure 4: Parent perplexity over training

3.5 Evaluation

For evaluation, we use the BLEU score, a standard metric for evaluating the quality of machine translations, and the metric of choice for the FLoRes paper we obtained our low-resource parallel data from. BLEU assesses candidate translations in comparison to a set of reference translations using a modified form of precision.

3.6 Computation Resource

The only computation resource we used is Google Colab, using their provided GPUs. For our parent model, it takes around 12 hours to train the model with the entire dataset for 10 epochs. Training our child models takes much less time, because the training dataset for those low-resource language models is much smaller. The parent model also gives great initialized parameters for the child model, and the child model, if the child and parent language pairs are similar enough, could inherit domain knowledge, enabling it to be trained in a shorter time.

4 Results

4.1 Parent Model

After training the parent model for 10 epochs, we attain a BLEU score of 9.94 using beam decode with 10 beams, which is significantly worse than the BLEU score of 22.4 for the French-English parent model used in (Zoph et al., 2016). However,

Decoding Scheme	BLEU (Hindi-English)
Greedy	7.4
Beam search (4 beams)	8.14
Beam search (10 beams)	9.94

Figure 5: Hindi-English BLEU, various decoders

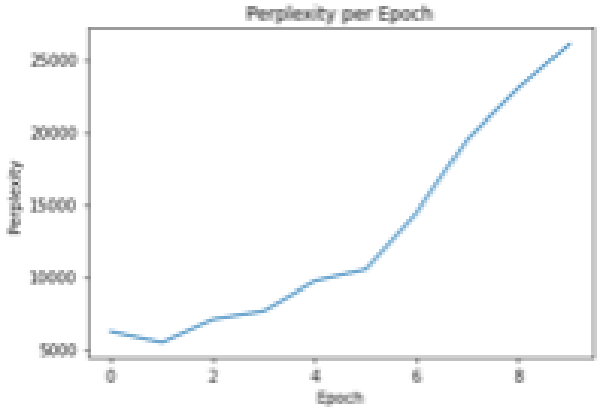

Figure 6: Sinhala child model perplexity

it is still significantly better than the supervised model's BLEU scores in (Guzmán et al., 2019) for Nepali-English and Sinhala-English (without Hindi parallel data), so it suffices for our experimentations as a parent model. We also computed BLEU using greedy decode and beam decode with 4 beams, both of which were lower than the 9.94 we attained with 10 beams.

This BLEU score is not the highest, especially compared to state-of-the-art machine translation methods, but training an encoder-decoder model by hand was a good exercise, as opposed to simply downloading pretrained weights from another source to initialize the child model with.

4.2 Nepali Child Model

The Nepali child model has the same encoder-decoder architecture as the parent model, but with the source (Nepali) words mapped to random Hindi embeddings, and the model parameters initialized using the weights from the Hindi-English parent model. Then, certain parameters are "frozen" so as to not be updated during training. The full results of trying to freeze various sets of parameters can be seen in Figure 8.

Frozen Parameters	Sinhala Test BLEU (Greedy, Beam)
Target Input Embeddings + Target Output Embeddings	0.7, 0.47
Target Output Embeddings	0.53, 0.266

Figure 7: Test BLEU under various frozen parameters and decoders

4.3 Sinhala Child Model

Similarly to the Nepali child model, the Sinhala child model has the same architecture as the parent except with Sinhala words being mapped to random Hindi embeddings and certain parameters being frozen.

Seeing that greedy decode performs better than beam decode on the Sinhala dataset likely means that the model failed to learn most of the language as if it did learn a significant portion of the language, doing a beam search should allow the decoder to provide better, more coherent results. There are many possible factors contributing to this the first being the dataset size. Compared to Nepali or Hindi, Sinhala had about 10 times less the amount of data which makes it difficult for the model to learn successfully. Another factor is that Sinhala has a much smaller vocabulary size which would likely make it more difficult to translate it to a larger target vocabulary. The last major possible factor we could see affecting these results is the similarity between the parent and child model, and as Sinhala is not that similar to Hindi especially compared to Nepali, it would intuitively make sense that Sinhala does worse when we use transfer learning. We further discuss the choice of the parent language in a later section.

4.4 Freezing Parameters

For each of the child models we tried out the freezing the two sets of parameters that did best in the (Zoph et al., 2016) paper as well as one other, and it seems that freezing only the target output embeddings did not do as well as the other two, and it makes sense that freezing only the final step and retraining the rest takes away most of the benefits of transfer learning. However, not retraining at all creates a model that simply does not fit the new

	Supervised		Unsupervised		Semi-supervised				Weakly supervised
		+mult.		+ mult.	it. 1	it. 2	it 1. + mult.	it 2. + mult.	
English–Nepali	4.3	6.9	0.1	8.3	6.8	6.8	8.8	8.8	5.8
Nepali–English	7.6	14.2	0.5	18.8	12.7	15.1	19.8	21.5	9.6
English–Sinhala	1.2	-	0.1	-	5.2	6.5	-	-	3.1
Sinhala–English	7.2	-	0.1	-	12.0	15.1	-	-	10.9

Figure 8: Benchmark Results

dataset, so the results of our experiments and of the (Zoph et al., 2016) paper makes intuitive sense. For our models, of the sets of parameters we tried for freezing, freezing target attention, target input and output embeddings did the best for Nepali, while for Sinhala, freezing just the target input and output embeddings did best.

4.5 Benchmark

We can compare our results with the results from the original Flores paper. The original paper achieved SOTA performance on machine translation task between the low-resource language pairs, Nepali-English and Sinhala-English. Even though we used the same dataset as in the Flores paper, our results are significantly worse. This is in part due to the fact that our parent model is not as effective as the one they were able to train. Without the same computation resources available, it was difficult to get a comparable parent model. Therefore, it's challenging to achieve a robust machine translation model between the child languages and English, given the shallower model architecture we use and the amount of computation resource we have.

For the original Flores paper, it takes nearly 25 hours with 8 Tesla V100 GPUs. This enabled them to use a much deeper model architecture then we were able to work with. In particular, the numbers of encoding and decoding layers they used are significantly more than ours, and their embedding space dimension is also much higher.

5 Discussion

5.1 Effect of Parent Language

One of the predictions in the (Zoph et al., 2016) paper was that for models where the parent language and the child language are similar, freezing more of the parameters would lead to better performance. In our experiments, we have such a pair of languages: Hindi and Nepali. These two share many similarities, and in fact, the (Guzmán et al., 2019) paper uses Hindi parallel data jointly with the Nepali data during training and this greatly improved the BLEU score. We also have a pair that are not as similar: Hindi and Sinhala. By comparing the effects of freezing more parameters on both, we can get some small empirical evidence to explore this claim.

Frozen Parameters	**Test BLEU (Nepali, Sinhala)**
Target Attention + Target Input Embeddings + Target Output Embeddings	1.14, 0.315
Target Input Embeddings + Target Output Embeddings	1.07, 0.47
Target Output Embeddings	0.875, 0.266

Figure 9: Test BLEU under various frozen parameters

As expected for Nepali, as the set of parameters that are frozen grows, the BLEU score increases as it is very similar to the parent language. However, with Sinhala freezing more parameters does not seem to help as past freezing the target input and output embeddings, the performance seems to decrease. However, it improves from freezing just the target output embeddings to freezing both the input and output embeddings, but this is likely due to the effects of transfer learning rather than any relationship between the parent and child language (although it is hard to draw any conclusion with the poor performance of the model).

5.2 Domain Mismatch

The creators of the FLoRes dataset acknowledged the existence of domain mismatch as a potential problem (Guzmán et al., 2019). Namely, the training examples and test examples have distinct domain-specific tendencies, meaning that a model trained on the training examples might still perform poorly on the test set simply due to a failure to learn about the test set's domain-specific qualities. Guzmán et al. demonstrated this by restricting the Sinhala-English training and test sets to a particular domain, and observed significantly better performance within that domain.

By choosing English as the target language, we are able to glean information about the model's translations to analyze this phenomenon. Consider

the following example outputs from our Sinhala-English child model.

- Example 1
  - Trg: While six people died when building the ship , one person died when it was about to launch the ship by falling a beam on to the head .
  - Pred: To customize <unk> contents of the titlebar and the contents of the screen saver , you can customize <unk> <unk> <unk> .
- Example 2
  - Trg: In the ship if it was filled the maximum amount it can carry these boats would only suffice for one-third of that whole amount .
  - Pred: This is the user interface <unk> which is not supported by the <unk> version of this box . This is a list of user interface manager .
- Example 3
  - Trg: Some bodies that sank along with the ship had been ravaged and dispersed by the ocean waves , and so could not be collected at all .
  - Pred: Could not enable <unk> notifications when <unk> <unk> is not found . This will not be able to read the <unk> <unk> .

We observe that the model's predictions are all related to user interfaces and their components, which is peculiar but a likely effect of the domain specificity hinted at by Guzmán et al. In these examples, the test set target "domain" is a ship-building disaster, and it seems that the model best approximates this "domain" using the user interface "domain" it learned during training.

We also hypothesize that the small size of the Sinhala-English dataset exacerbates this issue, as domain specificity is more heavily concentrated. The Sinhala-English dataset has only about 90,000 training examples, compared to 1.5 million training examples for the Hindi-English data obtained from the IIT Bombay Corpus.

This highlights a prevailing problem in low-resource machine translation: not only is the quantity of training examples low, this small quantity also contributes to a higher likelihood of domain specificity and mismatch. For less common languages, dataset quality is as much of an issue of dataset quantity.

6 Conclusion

6.1 Future Work and Extensions

Because of the limited time and computation resources we have, we have thought of some extensions of our work that might significantly improve the performance of the models.

Currently, the child model and the parent model share the same architecture, and the weights of parameters of the parent model serve as the initial parameters for the child model. Even though the similar language pair could benefit significantly from inheriting trained parameters, the child model could forget the knowledge it acquired from the parent model during the training process. As the experimentation results from Section 4.4 have shown, when more parameters are frozen and the parent and child language are similar enough, the child model tends to perform better. Therefore, it's reasonable to hypothesize that if we freeze the whole model, the child model could still well despite not training on the child language at all. The (Guzmán et al., 2019) paper supports this as they use Hindi training examples in the Nepali model and it actually improves the BLEU score, and it seems that simply using a model trained on Hindi data could potentially translate Nepali as well. Then, through adding a fully connected layer before the input of the child model, we aim to preprocess the child language, such as Nepali, to imitate the language structure of the parent language, Hindi, and then pass the preprocessed inputs to the original model, and this preprocessing would provide an easy way to transform the parent model to account for the new child language without changing the original model, and this could potentially work better than the method of transfer learning tried in (Zoph et al., 2016) or in our current project.

Another extension to explore is noise filtering or SVD truncating on weight matrices as that is something we did not try in this project. We could try to take the truncated SVD of weight matrices W and use those to generate a new weight matrix W^* which is the same idea as LSA but used in a different situation, but we hope to achieve the same effect of reducing features but maintaining the structure.

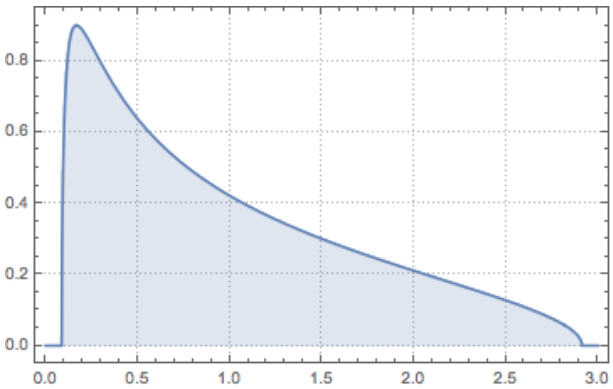

Figure 10: Machenko-Pastur Distribution

$$W = U\Sigma V^\top$$
$$W^* = U_t \Sigma_t V_t^\top$$

The other idea we could try is filtering out random noise and this comes from the idea that weight matrices are randomly initialized so one would expect them to contain a lot of random noise even after training. Then, when we do transfer learning to make sure that the learned structure stands out more, we could filter out noise from a weight matrix W by taking $W^\top W$ and taking its eigendecomposition, and finding the portion of the eigenvalues that best fits the Marchenko-Pastur distribution and removing the portion of the eigendecomposition corresponding to those eigenvalues as the Marchenko-Pastur distribution is what we would expect if W was purely random and then we could rescale the resulting matrix. This idea has been used to filter random noise in finance and provided an improvement (Laloux et al., 2000), and it seems possible that it could work in language translation as well. It would also be possible to use these two ideas in conjunction. Our motivation for this is that doing so would keep the more prominent features and make any truly learned feature more prominent, and one possibility is that these large features are the ones that hold more general linguistic information and creates a more generalized model. This could be seen as an approach similar to regularization, but uses very different ideas which leads us to believe that it could possible work despite (Zoph et al., 2016) claiming that regularization does not improve the model more than parameter freezing.

One idea that we believe would very likely improve the transfer learning method is using subword tokens. This was not done for our current model because of limitations, but for parent-child

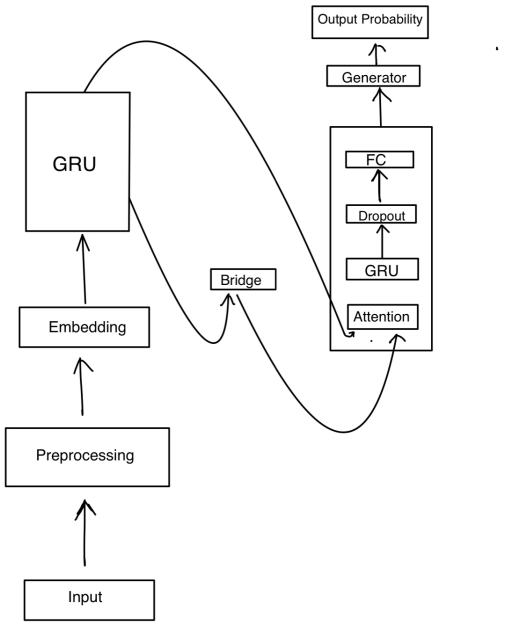

Figure 11: Child Model with Preprocess

language pairs that share the same script, using subword tokens would likely provide much more information and structure, and the information and structure found from sub-word tokens is more likely to transfer over, especially since we could use the exact same tokens in the child language.

We've also thought about using a pretrained model for learning Hindi embeddings. Generally, using pretrained model could significantly improve the efficiency and the performance of shallow models that don't have enough expressiveness. Therefore, using a pretrained model for Hindi might improve our performance, because the parent model will have better generalization power.

6.2 Takeaway

Transfer learning relies on significant domain knowledge. In order to create a good machine translation model for a low-resource language, we first need to identify whether there exists a successful machine translation model for a similar language to the target language. Sufficient domain knowledge in linguistics can help researchers identify similar languages, allowing for improved results. In addition, knowledge about the ways in which two languages are similar can help give

better intuition of which layers of the child model to freeze and inherit the parameters from the parent.

Freezing parameters is a powerful technique to reinforce the learning of child model from the parent model. However, deciding which layers to freeze is a delicate task which requires both domain knowledge and empirical exhaustive attempts.

Machine translation for low resource languages is an important task, because there is a descent amount of people in our world who do not speak mainstream languages with sophisticated translation models. Therefore, this research area deserves more attention to create effective solutions that could potentially benefit a lot of people.

References

Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato. 2019. The flores evaluation datasets for low-resource machine translation: Nepali–english and sinhala–english. pages 6098–6111.

Julius Kunze, Louis Kirsch, Ilia Kurenkov, Andreas Krug, Jens Johannsmeier, and Sebastian Stober. 2017. Transfer learning for speech recognition on a budget.

Laurent Laloux, Pierre Cizeau, Marc Potters, and Jean-Philippe Bouchaud. 2000. Random matrix theory and financial correlations. *International Journal of Theoretical and Applied Finance*.

L. Torrey and J. Shavlik. 2009. Transfer learning. *Handbook of Research on Machine Learning Applications*.

Dong Wang and Thomas Fang Zheng. 2015. Transfer learning for speech and language processing.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, Austin, Texas. Association for Computational Linguistics.

A Code

The code used for this project is at https://colab.research.google.com/drive/1QdEvvtXXH1K3gP4DNBZL553U5xgOF6zt?authuser=1#scrollTo=MufRoTQL3tuU