Investigating Techniques for Improving NMT Systems for Low Resource Languages

Stanford CS224N Custom Project

Alex Lee
Department of Computer Science
Stanford University
leealex@stanford.edu

Pranav Vaid
Department of Computer Science
Stanford University
pvaid@stanford.edu

Abstract

Neural Machine Translation (NMT) has become the standard for Machine Translation tasks, however, they encounter many technical challenges when training in low resource language pairs. In this paper, we investigate how different subword and word representations, as well as different data augmentation techniques can improve NMT performance on low resource languages. For our baseline, we train an encoder-decoder based seq2seq NMT model on a scarce Nepali-English dataset. Then, we compare different subword and word representations, such as Byte Pair Encoding (BPE) and a reduced vocab set. Finally, we augment our training data with backtranslation of monolingual data, transfer learning from Hindi, and noisy data. In addition, we propose a new variant of backtranslation for low-resource NMT that exceeds performance of traditional backtranslation methods. We find that BPE was the best performing subword representation. For data augmentation, we find that transfer learning and noisy data gives reliable improvements, yet back translation requires careful management of noise levels. By utilizing our novel variant of backtranslation alongside BPE and auxiliary data methods in combined models, we are able to increase in-domain performance by +4.55 BLEU and out-of-domain performance by +3.93 BLEU compared to the baseline.

1 Introduction

In recent years, Neural Machine Translation (NMT) has very quickly been established as the dominant approach for translation tasks, with variants of NMT achieving state-of-the-art results across tasks in WMT19 [1]. However, the most effective NMT systems are data hungry, requiring a parallel corpus with tens of millions of sentences in order to perform well. In low resource scenarios with only tens of thousands of parallel sentences, state-of-the-art NMT systems have drastically lower performance and adapting NMT system performance in these scenarios is often considered one of the major challenges in Neural Machine Translation today [2].

Improvement of low resource NMT performance is particularly impactful due to its practical implications. Most of the 7,000+ languages in the world are considered to be low resource, are spoken by significant fraction of the world population, and usually belong to underprivileged groups [3]. A third of languages existing today are endangered with less than 1,000 speakers remaining, and one language dies every 2 weeks [4]. Improving low resource NMT systems allows for accessible communication with these languages, offers a method of preserving the myriad of endangered languages for future generations, and provides underprivileged groups access to advanced NLP technologies.

In this work, we explore improvements that can be made to NMT systems so that they perform better in low resource settings. We select Nepali, a language with minimal parallel data available, and have access to just 65k Nepali-English parallel sentences, 150k noisy Nepali-English parallel sentences, and 334k Nepali monolingual sentences.

In our investigation, we focus on two areas: how to best represent low resource languages to improve translation performance, and how to most effectively augment low resource languages training data with monolingual data, data from similar languages and noisy data. We begin by comparing different vocab representations and tokenization schemes, namely a reduced vocab model and Byte Pair Encoding (BPE). Then, we apply transfer learning using a model trained on Hindi-English data. We also perform data augmentation through back translation on monolingual Nepali and English datasets and a noisy dataset. Finally, we combine successful improvements to create a final complete model with optimal tokenization and data augmentation.

2 Related Work

Multiple methods have been proposed to tackle low resource NMT in past literature. A common approach is domain adaption, in which a model is pretrained on high-resource domains and then finetuned on a new low-resource domain. Transfer learning is one such domain adaptation, in which a model is trained on a high-resource parent language pair prior to being trained on a low-resource child language pair, so that the model may apply learnings from the parent language to the child language [5]. Additionally, some have applied both supervised and unsupervised meta-learning successfully in low resource situations, an approach where the model utilizes information from a high-resource domain to learn an optimal parameter initialization for fast adapation to a low-resource domain [6][7]. Source or target side monolingual data is much more common than parallel data, and semi-supervised and unsupervised methods have been proposed to take advantage of this. Backtranslation is one such method that has been successful in a wide variety of NMT tasks, in which synthetic source data is generated from target side monolingual data [6][7]. Self-training is used similarly, instead generating synthetic target data from source side monolingual data [8][9]. Model based approaches, such as hyperparameters modifications to adapt models to low-resource situations have also been successfully used to low-resource boost performance in supervised approaches [10].

Although many methods have been proposed, few of these methods have been comprehensively applied to low resource languages. Instead, most of the discussed research focus on these methods in isolation, or only applied to out-of-domain high-resource languages or simulated low-resource language scenarios (when high-resource languages are sampled to simulate low-resource environments). We aim to investigate the effect of tokenization modifications, backtranslation and data augmentation on training in Nepali-English, a true low resource language pair.

3 Approach

Training difficulties for low resource languages typically manifest in two ways: a limited set of words in the vocabulary, and a lack of training data. We aim to address both problems through investigating different word and subword representations and data augmentation techniques, respectively.

3.1 Baseline

For our baseline model, we use an encoder-decoder seq2seq model based on the OpenNMT library [11], consulting the sample NMT code in the iPythonNotebook Library [12]. For our encoder, we use a bidirectional LSTM model. When the Nepali word sequence is fed into the encoder, it is first converted into embedding vectors, then fed through the bidirectional LSTM. For the decoder, we use a input feed decoder described in [13] with a unidirectional LSTM and global attention. As an input feed decoder, each attentional vector is fed back into the attention layer as part of the input for the next attention vector. The attentional layer then uses a softmax function to generate probability outputs. Figure 3 in A.1 provides a visual of our model.

For tokenization, our baseline model uses a simple word based encoding system. We generated our vocab set for the baseline model using the training set and validation set for the baseline model. In particular, we used the validation set in vocab generation to prevent the model from being unable to validate due to an inability to recognize words and sentences in the validation set.

3.2 Word and Subword Representation

3.2.1 Reduced Vocab Size

We used OpenNMT to construct our vocab set. To reduce the vocab size, we set up a vocab frequency threshold for the vocab to be included in the final dataset. In particular, we require a word to occur at least three times in order to be included in the Nepali and English vocabs. This reduces our source

vocab from around 78k tokens to 28k tokens and our target vocab from around 52k tokens to 20k tokens. We hope that with less tokens, the model will focus on learning more from tokens with higher frequencies, instead of trying to learn on tokens that only appear once.

3.2.2 Byte Pair Encoding (BPE)

For BPE, we used the OpenNMT tokenizer. As Nepali is a morphologically rich language, we believe that using a modified vocab can help the model learn better. Also, the lack of training data for low resource languages means that the vocabulary set generated from available sentence pairs will be very limited, so subword vocabs would allow the model to generate new words from elements of existing words and adapt to the open vocabulary of the testing data.

We used an implementation of BPE as according to Sennrich's original paper [14]. BPE begins by separating the entire datasets into characters. Then, the BPE algorithm continually merges the most common subword pairs, starting by merging the characters. The merging stops when the BPE model reaches the desired amount of subwords, which in our case is 3k tokens. We tokenize our training and validation data into subword units, and then train our models on this tokenized data. To retrieve model predictions for evaluation, we detokenize the model translations.

3.3 Transfer Learning

Transfer learning for low resource machine translation has been studied before and we implement a similar procedure as [5]. We first train our model on a Hindi-English translation task, and use the model parameters learned during this training as the initialization parameters for training on our Nepali-English corpus. We do not freeze any parameters, so that all parameters are retrained.

3.4 Data Augmentation

3.4.1 Backtranslation

We implement multiple variants of backtranslation to create synthetic parallel datasets. Our *vanilla backtranslation* (V-BT) approach utilizes the method described in [6]: we train a backward English-Nepali model and use the model to generate synthetic Nepali sentences from a monolingual English corpus. We then concatenate our synthetic generated parallel corpus with the original parallel corpus, and use the combined corpus to train our final model. Our *iterative-backtranslation* (I-BT) approach utilizes the method described in [7]: we train a forward Nepali-English model and use it to generate synthetic English sentences from a Nepali monolingual corpus. We then combine this synthetic corpus with our original corpus, to train a backward English-Nepali model, and use this model to generate synthetic Nepali sentences from a English monolingual corpus. The outputted synthetic dataset from this step only is combined with the original parallel corpus, and used to train our final model. We refer readers to [7] for a visual of this process.

We utilize the above two backtranslation variants with two decoding variants while translating from the backtranslation models. In *vanilla decoding*, we generate the output word by selecting the word with highest probability conditioned on the previously outputted words (equation 1). In *sampling decoding* we utilize a decoding scheme presented in [15], where we generate diverse pseudo-parallel sentences by randomly sampling the output word based on the word probability distribution, conditioned on the previously outputted words (equation 2).

$$\hat{y}_t = \arg\max_{y_t} \Pr(y_t|\boldsymbol{y}_{<t}, \boldsymbol{x}) \quad (1)$$

Vanilla Decoding

$$\hat{y}_t \sim \frac{\Pr(y_t|\boldsymbol{y}_{<t}, \boldsymbol{x})^{\frac{1}{\tau}}}{\sum_{y'} \Pr(y'|\boldsymbol{y}_{<t}, \boldsymbol{x})^{\frac{1}{\tau}}} \quad (2)$$

Sampling Decoding

$\hat{y}_t$, $\boldsymbol{y}_{<t}$, and $\boldsymbol{x}$ denote the outputted word at time t, word sequence history until time t, and input sequence respectively. τ represents the temperature parameter, used to to control the output diversity. We use a temperature value of 1.0.

From our experiments, we note that with successive iterations of backtranslation, vanilla decoding provides asympotically smaller but consistent increases in performance while sampling decoding provides a large increase in performance followed by a large decrease in performance. We thus propose and utilize a novel backtranslation variant combining the two decoding methods with iterative-backtranslation, in which we conduct iterative-backtranslation but use vanilla decoding when generating synthetic data from the forward English-Nepali model and use sampling decoding when generating synthetic data from the backward Nepali-English model. We refer to this as *diversified*

iterative-backtranslation or DI-BT for short, and a visual is pictured in Figure 1. To the best of our knowledge, this backtranslation variant has not been used before. The intuition behind this is that vanilla decoding will augment existing patterns in the dataset, helping the model reinforce knowledge of these, and that sampling decoding inserts diversity and noise into the dataset, so doing this for only the last step will help the model generalize to new patterns without overfitting to noise. Lastly, we take inspiration from [15] and generate two synthetic sentences per monolingual sentence during the DI-BT sampling decoding step, to further insert diversity into the dataset. This method is referred to as *double diversified iterative-backtranslation* (double DI-BT).

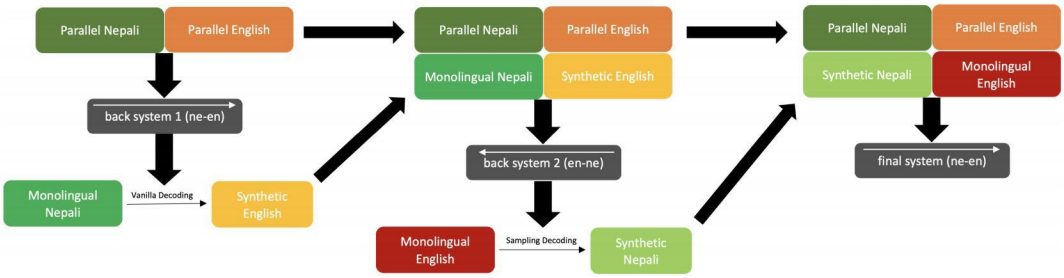

Figure 1: The flow of Diversified Iterative-Backtranslation

3.4.2 Utilization of noisy datasets

As discussed in 4.1, we access a noisy GNOME dataset. Many low resource language pairs have noisy datasets, so we explore the possibilty of cleaning the dataset. We take inspiration from [16], implementing heuristics to identify high quality sentences, but adapt our filters to patterns we observe in the GNOME dataset. We rely on the following 4 filters we consider to contribute to noisy data:

1. **Non-translated sentences:** Filters partially/non-translated Nepali sentences by removing sentence pairs that contain English characters in the Nepali sentence, since Nepali uses devanagari script.

2. **Presence of code:** Validates whether a sentence pair contains computer code. We remove sentence pairs containing any of the following characters in the Nepali translation: "*;%[]/:_~".

3. **Sentence fragments:** Filters segment fragments by removing sentence pairs where the English translation doesn't terminate with one of the following punctuation marks: ".?!".

4. **Minimum Sentence Length:** Identifies sentences most likely to resemble regular Nepali sentences. We remove any sentences with less than 6 words.

We experiment with augmenting our original dataset with the filtered GNOME dataset, containing 20K sentences, and compare it to performance when augmenting the original dataset with 150K unfiltered random sentences sampled from the full GNOME dataset. The amount of sentences filtered out by each of the above filters and examples of these sentences are detailed in Appendix D.

3.4.3 Dictionary

To perform out of domain translation, it is important for the model to learn more vocabulary. By pretraining our model on a bilingual Nepali-English dictionary, we hope the model will be able to learn a wide vocabulary from the dictionary, both to understand words from the Nepali input sequence and to translate Nepali sentences to Engish with the most relevant English word. We experiment with two variants of a dictionary, one containing pairings of Nepali words with their top English definition and another containing pairings of Nepali words with all of their possible English definitions.

3.5 Combined Model

After observing the methods with the largest improvements on validation sets, we train final models combining these methods, to explore whether individual performance improvements can be stacked together for a large performance improvement in a combined model. We utilize byte pair encoding (BPE), transfer learning on Hindi-English, the 150K unfiltered sentences from the GNOME dataset, and diversified-iterative-backtranslation. The flow of the fully combined model is pictured in Figure 2. For some low resource languages, transfer learning is impractical due to possible parent languages also being low resource, a condition which we simulate in one of our combined models by using just the combination of DI-BT and the unfiltered GNOME dataset to boost performance.

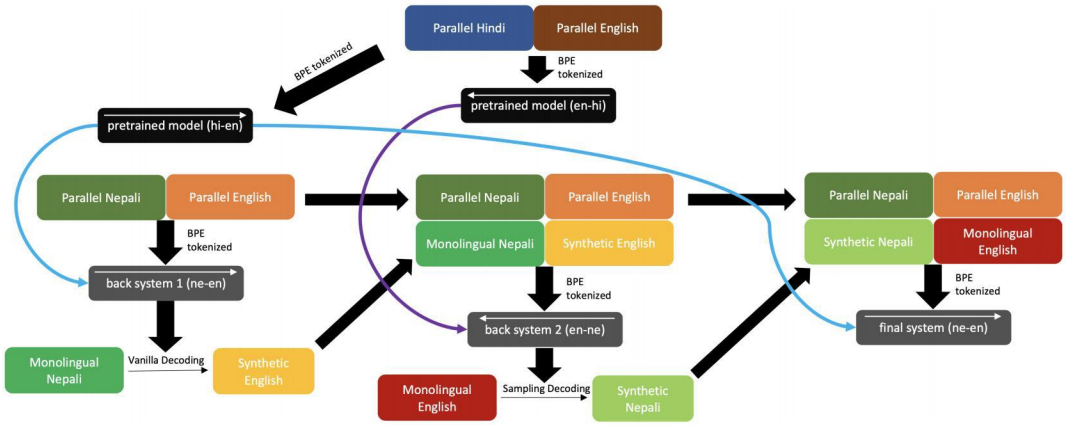

Figure 2: The flow of our fully combined NMT system

4 Experiments

4.1 Data

Dataset	Language Pair	Format	Sentences	Task
Bible	Nepali-English	parallel	61,809	Core Corpus
Penn Tree Bank	Nepali-English	parallel	4,279	Core Corpus
FLoRes	Nepali-English	parallel	2,924	Test Set
IIT Bombay Corpus	Hindi-English	parallel	1,609,682	Transfer Learning
Wikipedia (en)	English	monolingual	280,162	Backtranslation
Wikipedia (ne)	Nepali	monolingual	334,487	Backtranslation
GNOME/KDE/Ubuntu	Nepali-English	noisy parallel	494,994	Auxiliary Data
Dictionary (all pairs)	Nepali-English	dictionary	17,885 (word pairs)	Auxiliary Data
Dictionary (top pairs)	Nepali-English	dictionary	9,865 (word pairs)	Auxiliary Data

Table 1: Datasets utilized for experiments

We are using Nepali-English parallel data acquired from various sources, as described in Table 4.1, to train and evaluate our model on Nepali to English translations. We combine the Bible [17] and Penn Tree Bank [18] into a combined corpus, and split this corpus randomly to create our training set with 60K sentences, validation set with 3K sentences, and test set of 3K sentences to evaluate in-domain performance. We also use the FLoRes dataset [19], drawn from Wikipedia, as a test set to evaluate our models out-of-domain performance. Our transfer learning experiments utilize 1.5M sentences from the IIT Bombay corpus [20], with the rest of the corpus used as a validation set. Our backtranslation experiments utilize monolingual English and Nepali Wikipedia data [21]. These monolingual corpuses don't include any sentences from the FLoRes dataset. We also access the GNOME/KDE/Ubuntu corpus [22], which we consider to be noisy due to the heavy presence of code, non-translated words, and incomplete sentence fragments. We use this to explore methods of utilizing noisy datasets to improve model performance. Finally, to explore expansion of model vocabulary we utilize two Nepali-English dictionaries, extracted from the same source dictionary [23]. The "all pairs" dictionary includes pairings of Nepali words with all of their possible English definitions, while the "top pairs" dictionary only utilizes the strongest English definition for each Nepali word.

4.2 Evaluation method

We use the BLEU score as our evaluation metric, because it is simple and effective at understanding translation quality. Evaluation was done on a test set of 3K parallel Nepali-English sentences on the FLoRes dataset drawn from Wikipedia to test out-of-domain performance, and 3K parallel Nepali-English sentences drawn from the Bible and Penn Tree Bank to evaluate in-domain performance.

4.3 Experimental details

We relied on past research to decide on appropriate settings for our encoder-decoder model [2]. We decided to use a embedding size of 512 and hidden layer size of 1024. For both our encoders and decoders, we used 2 layers, as according to the paper we reviewed. We used an encoder and decoder dropout of 0.2 and an attentional dropout of 0.1. We used the Adam optimizer for our model, a

learning rate of 0.01 and a max gradient normalization of 2. We used an early stopping mechanism to decide when to stop: we evaluate the model every 500 to 1000 iterations and early stopping happens when accuracy and perplexity doesn't improve in 2000 iterations.

4.4 Results

4.4.1 Performance of Word and Subword Representation

Model Number	Model Description	BLEU	
		In Domain	Out of Domain
0	Baseline	16.10	0.43
1	0 + Reduced Vocab Size	16.84	0.29
2	0 + Byte Pair Encoding (BPE)	**16.89**	**0.81**

Table 2: Performance of models with different word and subword encodings

Reduced vocab size and BPE both achieve some success over the baseline model. For the reduced vocab size, there was some improvements for the in-domain score, but a decrease for the out-domain score. This is expected, as a decrease in vocab size would improve learning for the in domain translation, but would also reduce the words the model understands in the out-of-domain data. Training on the BPE tokenized training data boosted both in domain and out domain performance. The out domain performance improvement suggests that BPE tokenization provides a greater vocabulary set by allowing the model to generate and learn new words. On the other hand, the in-domain improvement suggests that BPE tokenization allows the model to learn subword tokens across different words, providing more training data for each subword, as compared to words.

4.4.2 Performance of Data Augmentation methods via Auxiliary Data

Model Number	Model Description	BLEU	
		In Domain	Out of Domain
0	Baseline	16.10	0.43
3	0 + Hindi Transfer Learning	**18.49**	**1.04**
4	0 + Dictionary (all pair)	13.77	0.41
5	0 + Dictionary (top pairs)	9.00	0.41
6	0 + GNOME (unfiltered)	17.62	0.74
7	0 + GNOME (filtered)	14.88	0.53

Table 3: Performance of models utilizing auxiliary data

Among auxiliary data methods, transfer learning our model with Hindi resulted in the largest in-domain and out-of-domain scores, which we expected due to the morphological similarity that Hindi has with Nepali, and the large size of the Hindi-English corpus. We were very surprised that pretraining our model on both variants of the dictionary degraded performance, and it's worth noting that the all pairs dictionary does significantly better than the top pairs dictionary. Training on the unfiltered GNOME dataset boosted our performance compared to the baseline, while training on the smaller filtered GNOME dataset surprisingly resulted in a lower performance. This suggests that our model is resilient to a noisy input, and can extract meaningful insights from it, and also that the filters we utilized on the GNOME dataset were not good heuristics for identifying high quality sentences.

4.4.3 Performance of Backtranslation

Model Number	Model Description	BLEU	
		In Domain	Out of Domain
0	Baseline	16.10	0.43
8	0 + V-BT w/ vanilla decoding	16.96	0.36
9	0 + I-BT w/ vanilla decoding	17.47	0.34
10	0 + V-BT w/ sampling decoding	18.66	0.31
11	0 + I-BT w/ sampling decoding	16.52	0.37
12	0 + DI-BT	**19.14**	**0.48**
13	0 + double DI-BT	18.50	0.26

Table 4: Performance of systems utilizing various backtranslation variants

Backtranslation appears to be a very effective strategy for low resource translation, with all but one of the variants increasing in-domain scores. However, all but one of the variants degrades performance on out-of-domain scores, which is unsurprising since backtranslation augments the dataset with synthetic

sentences from the model, which are likely to resemble in-domain sentences. Interestingly, I-BT with vanilla decoding increases scores compared to V-BT with vanilla decoding, but I-BT with sampling decoding decreases scores compared to V-BT with sampling decoding. We discuss reasons for this in the Analysis section. Our proposed DI-BT exceeds both traditional backtranslation methods, with DI-BT giving us the highest in-domain and out-of-domain score across all backtranslation methods, and double DI-BT provides results comparable to traditional backtranslation methods. DI-BT also produces the highest in-domain scores among all of our approaches in this paper, with only model 15, which still utilizes DI-BT (but in combination with noisy GNOME data), scoring higher.

4.4.4 Performance of Combined Models

Model Number	Model Description	BLEU	
		In Domain	Out of Domain
0	Baseline	16.10	0.43
14	Partially Combined, no DI-BT (0 + 2 + 3 + 6)	18.08	3.85
15	DI-BT and GNOME only (0 + 6 + 12)	**20.65**	0.52
16	Fully Combined (0 + 2 + 3 + 6 + 12)	18.97	**4.36**

Table 5: Performance of different combined models

All of our combined models produce a significant improvement over the baseline, which was expected. Its worth noting that models 14 and 16 have a lower performance on in-domain performance than their best component method, but significantly outperform their component methods on the out-of-domain set (with the fully combined model performing best). Model 15 significantly outperforms its component methods on the in-domain test set, but underperforms the best component method on the out-of-domain test set. This suggests that improvements among individual methods do not stack cleanly on each other when combined, and the interaction of methods with each other plays a very significant role. Also, the best combination of methods for low resource translation is dependent on the target task: in-domain and out-of-domain performance are not correlated.

5 Analysis

5.1 Word and Subword Representations

Inspection of the translated outputs for reduced vocab size and BPE shows us the different levels of quality and characteristics of each tokenization technique. The BPE sentence in Appendix B provides a translation that captures some of the meaning of the gold standard, mainly the concepts of "heart", "you" and "open". However, they are not organized in the same order as the gold standard. Instead of "our mouth is open", the BPE translation has "our hearts is open". This may be because the same sentence can have more tokens when represented as subwords than as words, and thus, it can be more difficult for the model to replicate an entire sequence. Interestingly, BPE generates the word "Corinthites", as an equivalent translation to "Corinthians". It shows that while BPE translations are able to generate new words, the new words may not be a valid English word. For the reduced vocab size example in Appendix B, we can see that the reduced vocab size can easily generate <unk> (unknown) tokens, as it encounters a Nepali or English word that is not in its vocabulary.

5.2 Data Augmentation via Auxiliary Data Methods

The improvements made by data augmentation method are most evident from observing out-of-domain translations. As data augmentation methods via auxiliary data provide a new domain of data besides the core Bible and Penn Tree Bank dataset, out-of-domain translations rely on this auxiliary data the most. The key towards choosing the right auxiliary data is to find data that has the desired domain for the translation task. We can observe the difference from the example in Appendix C.

While the dictionary task provides an expanded vocabulary, the fact that each dictionary entry is just a single word means that it doesn't help the model to learn the overall structure of a sentence. Evidently, in the Dict translation example in Appendix C, the sentence is fairly fragmented. The GNOME dataset has a domain in computer programming, hence, when it encounters a Nepali word that should translate to report or objective report, it would tend to translate it into "bug report" such as in the example. Finally, the Hindi dataset is a general domain dataset with a large vocabulary. The expanded vocabulary from Hindi required us to put a threshold on vocabulary to prevent it from overloading the embeddings. Thus, we can see that there's an <unk> token in the translation in the example. However, beyond the <unk>, the Hindu translation performs the best, being able to

translate the word "report". Furthermore, as a general domain dataset, it also helps in establishing a basic Nepali/Hindu language model. This is evident in how the model has been able to understand synonyms by capturing the meaning of "prepare" by using a similar word "plans".

5.3 Backtranslation

Backtranslation was our most effective method explored, with every variant providing an increase in score compared to the baseline. Synthetic sentences generated by vanilla decoding augment pre-existing patterns in the dataset, helping the model learn these patterns better. This explains the consistent increase in score between V-BT and I-BT variants, since further iterations of backtranslation augment these patterns further and increase model performance. Synthetic sentences generated by sampling decoding inserts slightly new and diverse patterns into the dataset, by virtue of the random sampling, which we hypothesize helps the model learn these new patterns and generalize better.

However, random sampling can also result in overly noisy datasets. One way of evaluating the level of noise is by measuring the number of <unk> tokens in each translation. For the double iteration backtranslation, repeating sampling decoding twice leads to a lot of nonsensical and noisy sentences in the dataset: likely because the model has very limited training data to start off with, and so is unable to generate sentences that make sense after two iterations of randomness in decoding have been utilized in sequence. In fact, the translation after two iterations of randomness has almost 50% more <unk> tokens compared to the translation with no iterations of random samples. We believe DI-BT works so well in this low resource scenario because it is able to create a large synthetic corpus with existing patterns by utilizing vanilla decoding with back system 1, so that back system 2's knowledge of these patterns is reinforced. Back system 2 then is able to generate diverse outputs utilizing sampling generation that are more sensible due to the reinforced knowledge of correct patterns, and these diverse outputs help the final model accurately generalize to new patterns while maintaining accuracy on original patterns. Double DI-BT performs slightly worse because doubling the amount of sampled sentences results in too much noise, raising the issues mentioned before.

5.4 Combined Model

Each of our combined models is designed for specific real life low resource language scenario. Model 14 corresponds to a low resource language with no monolingual data, model 15 corresponds to a low resource language whose similar languages are also low resource, and model 16 corresponds to a low resource language that is has a high resource parent language and monolingual data.

From the results, we learn that, as expected, low resource languages with high resource parent languages can achieve good out-of-domain performance after pretraining, simply by applying the model for the high resource language to the low resource language. In the context of our paper, Hindu and Nepali share the same script and many words. However, for an improvement in in-domain translation performance, monolingual data for back translation becomes more important in order to provide more augmented and diversified patterns in the training data.

6 Conclusion

In this paper, we explored multiple improvements that could be applied to low resource Neural Machine Translation systems. We experiment with multiple approaches to word representations, data augmentation, backtranslation, and combinations of the aforementioned methods. Additionally, we propose a new variant of backtranslation to increase model performance and find it to be more effective on our low resource language pair than traditional backtranslation methods. We find that BPE, transfer learning, noisy parallel data, and our proposed diversified-iterative-backtranslation (DI-BT) provide the largest individual improvements. We also find that a system combining of DI-BT and noisy data provide the largest in-domain improvements, and a system combining BPE, Transfer Learning, noisy data, and DI-BT provide the largest out-of-domain improvements.

Due to time and compute constraints, we are only able to test our methods in Nepali-English translation, and further work is required to determine whether improvements found in this paper generalize to all low resource language pairs. We were also unable to explore many other proposed methods for low-resource NMT, such as meta-learning, different model structures, or data augmentation with dictionaries. However, we hope that our initial results provide a good direction for promising areas that should be explored in future work.

7 Additional Information

- In-Class Mentor: John Hewitt
- External Mentors: Leah Brickson and Ryan Burke
- External Collaborators (if you have any): None
- Sharing project: None

References

[1] Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, Florence, Italy, August 2019. Association for Computational Linguistics.

[2] Philipp Koehn and Rebecca Knowles. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver, August 2017. Association for Computational Linguistics.

[3] Surafel M. Lakew, Matteo Negri, and Marco Turchi. Low resource neural machine translation: A benchmark for five african languages, 2020.

[4] Nina Strochlic. Saving the world's dying and disappearing languages, Feb 2021.

[5] Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 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, November 2016. Association for Computational Linguistics.

[6] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. *CoRR*, abs/1511.06709, 2015.

[7] Vu Cong Duy Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 18–24, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[8] Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. Revisiting self-training for neural sequence generation. *CoRR*, abs/1909.13788, 2019.

[9] Jiajun Zhang and Chengqing Zong. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545, Austin, Texas, November 2016. Association for Computational Linguistics.

[10] Rico Sennrich and Biao Zhang. Revisiting low-resource neural machine translation: A case study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[11] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[12] Opennmt ipythonnotebook library. `https://opennmt.net/OpenNMT-py/examples/Library.html`. Accessed February, 2021.

[13] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. 2015.

[14] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

[15] Kenji Imamura, Atsushi Fujita, and Eiichiro Sumita. Enhancement of encoder and attention using target monolingual corpora in neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 55–63, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[16] Mārcis Pinnis. Tilde's parallel corpus filtering methods for WMT 2018. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 939–945, Belgium, Brussels, October 2018. Association for Computational Linguistics.

[17] A multilingual parallel corpus created from translations of the bible. `https://github.com/christos-c/bible-corpus/`. Accessed February, 2021.

[18] Penn tree bank in english and nepali. `https://dl.fbaipublicfiles.com/fairseq/data/nepali-penn-treebank.en.patch` and `https://dl.fbaipublicfiles.com/fairseq/data/nepali-penn-treebank.ne.patch`. Accessed February, 2021.

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

[20] Iit bombay english-hindi corpus. `https://www.cfilt.iitb.ac.in/~parallelcorp/iitb_en_hi_parallel/`. Accessed March, 2021.

[21] Monolingual english and nepali corpus. `https://dl.fbaipublicfiles.com/fairseq/data/wikipedia.ne_filtered.gz` and `https://dl.fbaipublicfiles.com/fairseq/data/wikipedia.en_filtered.gz`. Accessed February, 2021.

[22] Gnome/kde/ubuntu handbook from opus. `https://opus.nlpl.eu`. Accessed February, 2021.

[23] Nepali english dictionary via tacl data release. `http://www.seas.upenn.edu/~nlp/resources/TACL-data-release/dictionaries.tar.gz`. Accessed February, 2021.

A Additional Figures

A.1 Baseline Model

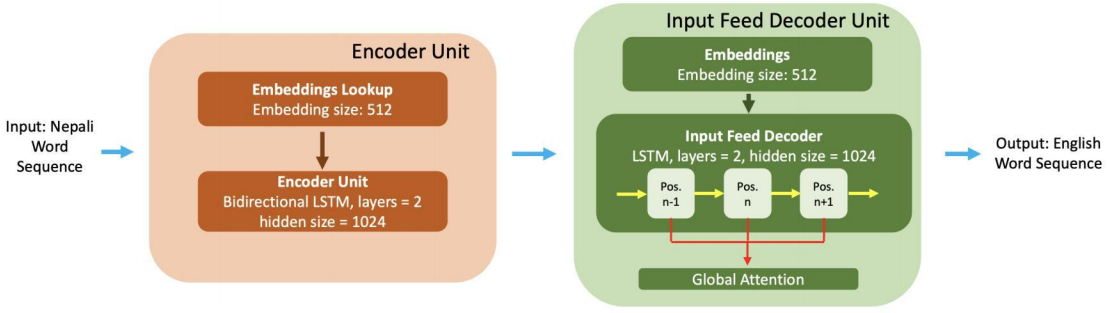

Figure 3: Baseline Model

B Word and Subword Representation Translation Examples

Model Type	Prediction Sentence 1
Gold Standard Sentence	***Our mouth is open to you, Corinthians. Our heart is enlarged.***
BPE	*O ye Corinthites, our heart is with you, and our hearts is open.*
Reduced Vocab Size	*And now we say unto you, We are our <unk> our heart is enlarged.*

C Data Augmentation Translation Examples

Model Type	Prediction Sentence 2
Gold Standard Sentence	***It is used to prepare the objective report of the study.***
Dict	*According to the Upjohn's to use it for a price.*
Gnome	*Let them be given to the bug report to the proud.*
Hindi	*<unk> purpose plans to file the reports.*

D Filter statistics and example sentences in GNOME Dataset

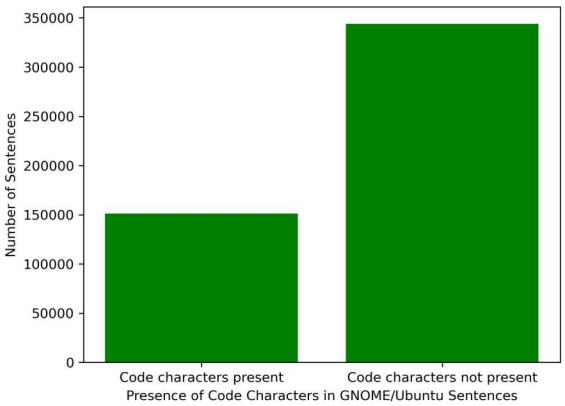

Figure 4: Distribution of code characters in GNOME

English Sentence	***Nepali Parallel Translation***
Move %s onto %s.	*%s लाई %s मा सार्नुहोस्*
Base Card: ~a	*आधार तास: ~a*
Add <dir> to the list of directories to search for source files	*स्रोत फाइल खोज्नका लागि डाइरेक्टरीको सूचीमा <dir> थप्नुहोस् ।*
Enter <size>,<assoc>,<line_size>:	*प्रविष्टि गर्नुहोस् <size>,<assoc>,<line_size>:*
Call __libc_freeres() at exit before checking for memory leaks	*बाहिरिदा स्मृति चुहावट जाँच गर्नु अगाडी Call __libc_freeres()*

Figure 5: Examples of code character sentence pairs in GNOME

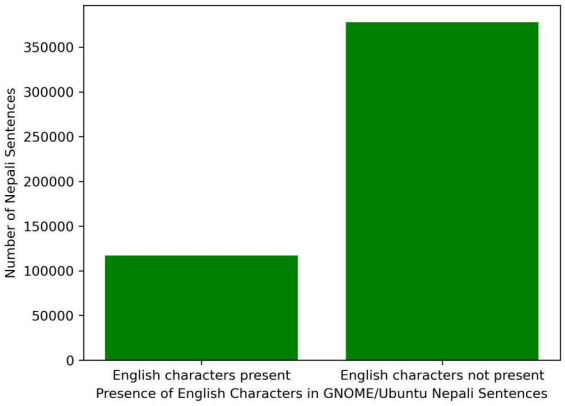

Figure 6: Distribution of english characters in Nepali sentences in GNOME

English Sentence	***Nepali Parallel Translation***
_Hint	*सङ्केत_New" is for the menu item 'Game->New', implies "New Game*
_New	*नयाँ_New Game*
_Redo Move	*चाललाई रिडू गर्नुहोस्Reset*
Waste	*foundationslot hint*
You should have received a copy of the GNU General Public License along with this program. If not, see .	*तपाईंले %s को साथमा जीएनयूको साधारण सार्वजनिक इजाजतपत्रको प्रतिलिपि प्राप्त गरेको हुनुपर्छ; यदि नभएमा, निः शुल्क सफ्टवेयर प्रतिष्ठान, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA मा पत्र लेख्नुहोस्slot type*

Figure 7: Examples of non/semi-translated sentence pairs in GNOME

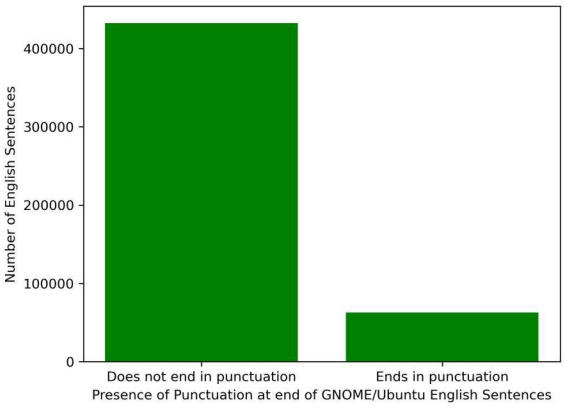

Figure 8: Distribution of punctuated English sentences in GNOME

English Sentence	***Nepali Parallel Translation***
Same suit	*खेल सुरु गर्नुहोस्*
Will O The Wisp	*विल ओ द विस्प*
When without a stapler, a staple and a ruler will work	*स्टेप्लर नभएको बेलामा स्टेपल र रुलरले काम गर्छ*

Figure 9: Examples of low quality sentences and sentence fragments in GNOME

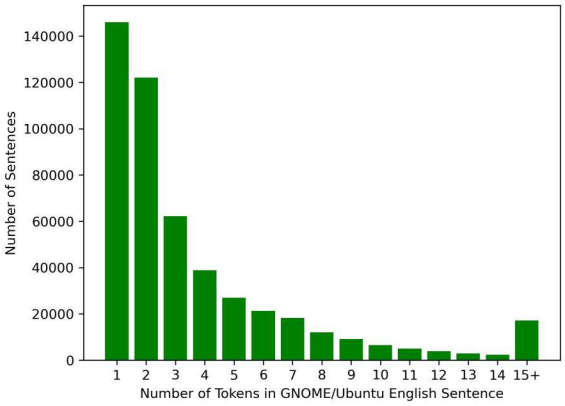

Figure 10: Distribution of English sentence lengths in GNOME