Data Augmentation for Low-resource Neural Machine Translation: A Systematic Analysis

Zhiqiang Shi *
King's College London
London, UK
z.q.shizhiqiang@gmail.com

Abstract

As an effective way to address data scarcity problem, data augmentation has received significant interest in low-resource neural machine translation, while the latter has the potential to reduce digital divide and benefit out of domain translation. However, the existing works mainly focus on how to generate the synthetic data, while the synthetic data quality and the way we use the synthetic data also matter. In this paper, we give a systematic analysis of data augmentation for low-resource neural machine translation that encompasses all the three aspects. We show that with careful control of the synthetic data quality and the way we use the synthetic data, the performance can be greatly boosted even with the same method to generate the synthetic data.

1 Introduction

Machine translation is referred to as using machines to translate between different languages. Low-resource machine translation can help to reduce digital divide by translating the large amount of information available in some high-resource languages to lower-resource languages. Also, out of domain translation tasks, including those in medical domain can benefit from the advancements in low-resource machine translation. However, data scarcity poses significant challenges to low-resource machine translation.

As an effective method to address lack of data, data augmentation has received great interest in both computer vision (Oquab et al., 2014), (Cubuk et al., 2019) and natural language processing (Sennrich et al., 2016a), (Wang et al., 2018). Several methods such as back-translation (Sennrich et al., 2016a), switchout (Wang et al., 2018) have been developed to augment the training data of machine translation. Additional resources such as additional data (Sennrich et al., 2016a), (Yang et al., 2020) or dictionaries (Song et al., 2019), (Jones et al., 2023) have also been utilised to facilitate data augmentation within the context of machine translation.

The core of data augmentation is the synthetic data. In our opinion, there are three aspects that are important to the synthetic data: how to generate the synthetic data, the synthetic data quality and how to use the synthetic data. Most of the existing works only focus on how to generate the synthetic data, the synthetic data is then mixed with the original data to train the final model, which prevents us from truly understanding the mechanism of data augmentation for neural machine translation. Therefore, in this paper, we aim to give a systemic analysis of data augmentation for low-resource neural machine translation which covers all the three aspects. Based on our findings, we also propose some methods that can significantly boost the performance of data augmentation for low-resource machine translation.

2 Related Work

Neural machine translation (Sutskever et al., 2014), (Bahdanau et al., 2016), (Vaswani et al., 2017) has greatly improved the performance of machine translation recently. However, these methods require large amount of training data, while data scarcity is a common problem in low-resource problem. As an effective way to address lack of data, data augmentation has been utilised in both computer vision (Oquab et al., 2014), (Cubuk et al., 2019) and natural language processing (Li et al., 2022). For neural machine translation, the existing work can be categorise into word based methods and translation based methods depending on whether an additional translation model is used to generate the synthetic data. In word based methods, the words or word

*This work was initiated at the University of Edinburgh and completed independently after the author's departure, and before the author joined King's College London. The author is currently affiliated with King's College London, but the research was not supported by King's College London.

Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 510–522

embeddings of source or target sentences are replaced. Several examples include switchout (Wang et al., 2018), replacing common words with rare words (Fadaee et al., 2017) and soft contextual (Gao et al., 2019).

Back-translation (Sennrich et al., 2016a), (Edunov et al., 2018) utilise an additional translation model to generate the synthetic data. Various tricks such as adding a special tags to indicate whether the data is synthetic data or natural data (Bahdanau et al., 2016) have also been proposed. The codeswitching method is used early in fully supervised neural machine translation (Song et al., 2019) where some source tokens are replaced by pre-specified translation in target language and a pointer network is used. More recent work using codeswitching makes use of additional monolingual or bilingual data where the model is pretrained on these large amount of additional data (Yang et al., 2020), (Jones et al., 2023). Apart from traditional translation methods, codeswitching is also used in prompting methods with in context learning (Ghazvininejad et al., 2023). In (Ghazvininejad et al., 2023), the researchers show improvement when the prompts contain the translation of source tokens in target language.

3 Experiment Setup

We use WMT17 news translation task dataset (Bojar et al., 2017) for our experiments. We remove sentences that are longer than 250 tokens and sentence pairs that source and target sentence ratio is larger than 1.5. This results in 2M sentences for training dataset, 20K sentences for development dataset and 3K for test dataset. We use Moses tokenizer (Koehn et al., 2007) to tokenize all the datasets and then use BPE (Sennrich et al., 2016b) with 40K source and target joint vocabulary. The data processing procedure follows (Edunov et al., 2018).

We subsample the training set to 200K to simulate the low-resource situation, we choose 200K because this size is widely used for low-resource machine translation. We further subsample the 200K training set exponentially following the work in (Edunov et al., 2018) which increases the dataset size exponentially and floor the size for easier implementation, this results in 10K, 20k, 50K, 100K and 200K for our experiments. For the monolingual data, because we only use a small fraction of the original 2M training dataset as our training set, we directly sample the monolingual data from the remaining part of the 2M training dataset. We randomly sample 600K target sentences from the remaining part of the 2M training dataset which excludes the 200K training set we use in the paper. Because our monolingual target data is from the original 2M training dataset, we have the corresponding source translations. This can help us to better analyse the synthetic data generated by back-translation because we have the ground truth reference for it. We will describe how we use the source reference of the monolingual data to evaluate the quality of the synthetic data in Section 5. We use both BLEU score (Papineni et al., 2002) and COMET score (Rei et al., 2020) to measure the translation quality. For COMET score, we use `wmt22-comet-da` for evaluation. The COMET scores we report in the paper are multiplied by 100 for better visualisation. Unless explicitly stated, we use the 100K bilingual dataset for most of our experiments.

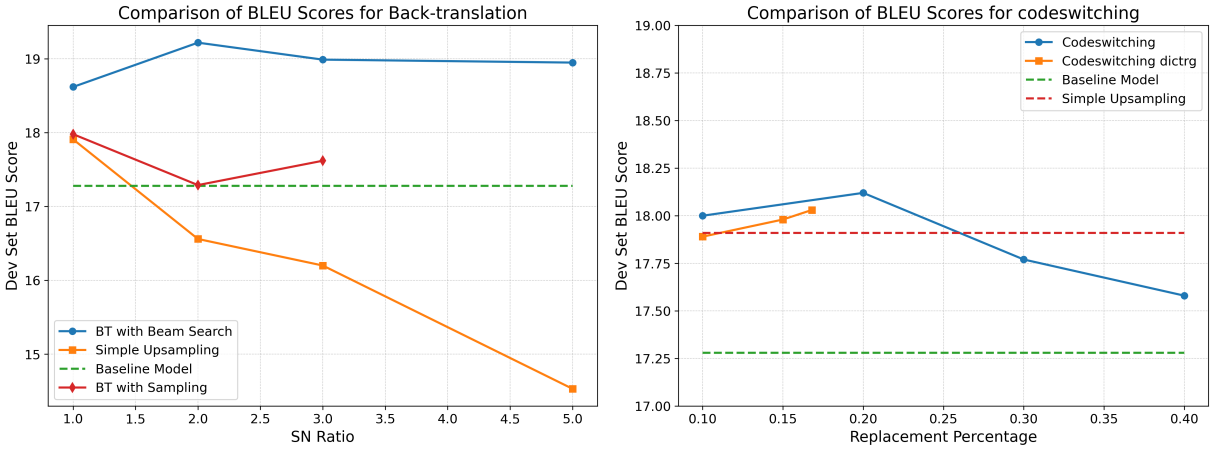
We use the transformer base model in (Vaswani et al., 2017) as our translation model which has 6 transformer layers in both encoder and decoder. We use the same learning rate schedule as in (Vaswani et al., 2017) and a warmup step 4000 and peak learning rate is 0.001. We use the same hyperparameters for all the experiments. The experiments are conducted on an Nvidia 24GB A10 GPU with 200 GPU hours.

4 Generating the Synthetic Data

In this section, we describe the data augmentation methods we use in this project. The existing works for machine translation data augmentation can be divided into translation based and replacement based methods. Therefore, in this project, we implement back-translation and codeswitching as representative methods.

For codeswitching, the existing works (Yang et al., 2020), (Jones et al., 2023) use large amount of additional bilingual data and pre-train the model on the corrupted data. However, the additional bilingual datasets are not available for low-resource languages. Therefore, in this project, we implement a very simple codeswitching method, in which we generate the synthetic data by replacing the source words with their corresponding target translation based on a bilingual dictionary [1]. For

[1] https://github.com/facebookresearch/MUSE##ground-truth-bilingual-dictionaries



(a) (b)

Figure 1: The experiment results for back-translation and codeswitching with different synthetic data generation strategies. Figure 1a: The experiment results for back-translation. *SN ratio: synthetic and natural data ratio, BT: back-translation.* Figure 1b: The experiment results for codeswitching. *codeswitching dictrg: codeswitching with target translation.* The COMET scores are reported in Appendix A.

source words that have multiply target translations, we random choose one translation. In order to test the influence of this random selection, we also implement another method: codeswitching with target translation, in which we replace the source word with the translation that appears in the target sentence. See Appendix A for more details about our codeswitching methods.

We use the same procedure for back-translation as in (Sennrich et al., 2016a). However, different from (Sennrich et al., 2016a), we experiment with different methods to generate the synthetic data, including beam search and `topk` sampling (Edunov et al., 2018).

For beam search, at each time step, we keep `beam size` hypotheses with the highest probabilities, which will result in `beam size` hypotheses after decoding. We then select the hypothesis with the highest probability as our generation results. We use `beam size` 5 for our project. For `topk` sampling, at each time step, we select k outputs with the highest probabilities and normalize this distribution. We then select the output by sampling from these k outputs according to the normalized distribution. Because existing work (Edunov et al., 2018) shows different k gives similar results, we use k equals to 5 for our project.

We also use simple upsampling to investigate the performance gain is because of the data augmentation methods we use, not just because we have more training data due to the additional synthetic data. In simple upsampling, we directly upsample the natural data so that the final training data we have is equal to that used in the corresponding data augmentation methods.

Figure 1 shows the experiment results of our data augmentation methods on the 100K bilingual dataset, we first generate the synthetic data using the methods we describe above, then mix the synthetic data and the natural data to train the final translation model and evaluate the BLEU score in the development set. Because of the restriction of codeswitching with target translation, the maximum replacement percentage for this method is 16.8%. The results indicate that for both back-translation and codeswitching, the performance greatly depends on the synthetic data generation strategies and their hyperparameters, with inappropriate generation strategies and hyperparameters, the results are even worse than simple upsampling and the model without any data augmentation methods. For example, BT with sampleing and SN ratio 2 underperforms compared to baseline model, codeswitching with large replacement percentage underperforms compared to simple upsampling This highlights the urgent need of analysing the generated synthetic data.

5 The Synthetic Data Quality

5.1 Back-translation

In this section, we will give a systematic analysis of the generated synthetic data using back-translation. For the evaluation dataset we use in this section,

dataset size	perplexity beam	perplexity sampling	cosine similarity beam	cosine similarity sampling	BLEU score beam	BLEU score sampling	final BLEU beam	final BLEU sampling
10k	159.59	267.23	0.3054	0.3161	3.88	1.21	2.25	1.74
50k	389.81	410.41	0.5974	0.4883	17.05	4.84	16.08	12.15
100k	383.26	379.85	0.6739	0.5529	23.81	7.54	18.62	17.98
200k	357.63	210.08	0.6994	0.5524	27.6	5.17	21.99	-

Table 1: Perplexity and BLEU score of the synthetic data generated using beam search or sampling. The perplexity for the monolingual data is 42.53. *beam search and sampling indicate the strategy used to generate the synthetic data, perplexity and BLEU score means perplexity and BLEU score of the synthetic data, final BLEU score: the BLEU score of the final translation model.*

dataset size	hypo to refe beam	hypo to refe sampling	refe to hypo beam	refe to hypo sampling	both directions beam	both directions sampling
10k	14.9%	15.64%	8.17%	13.74%	6.73%	10.9%
50k	22.96%	12%	20.92%	12.44%	18.37%	8%
100k	19.59%	13.17%	18.04%	13.66%	14.43%	10.24%
200k	25.82%	21.67%	27.7%	21.18%	22.54%	16.25%

Table 2: The entailment percentage of the synthetic data which we use to evaluate adequacy of the synthetic data. *hypo: hypothesis (the synthetic data), refe: reference.*

we randomly sample 2K sentences from the monolingual target data stated in Section 3. Because the monolingual target data we use in this project is from the original 2M training data, for each sentence in the monolingual target data we use, we have its corresponding source translation. In other words, for the 2K evaluation dataset we use in this section, we have both source sentences and target sentences. Because of this, we can compute BLEU score of the synthetic data against the source reference to evaluate the translation quality of the synthetic data, which contains both fluency and adequacy (Papineni et al., 2002). The evaluation procedure we use is: We first train a translation model from target language to source language on different natural bilingual datasets sizes (10K, 50K, 100K, 200K), we then use this translation model to generate synthetic data for the 2K evaluation dataset using beam search or sampling, after we obtain the synthetic data, we can compute perplexity (Jelinek et al., 1977) to evaluate fluency of the synthetic data and BLEU score to evaluate translation quality which includes fluency and adequacy.

For the perplexity, we randomly sample 100K monolingual source sentences from our monolingual data and fine-tune a GPT2 small model (Radford et al., 2019). We evaluate every 500 steps and stop the training when the performance on development set does not improve for more than 3 evaluations. We then use our fine-tuned model to compute the mean perplexity of the synthetic data. Because we observe some synthetic data have very large perplexity, we use isolation forest (Liu et al., 2008) to filter out the outliers (10% of the data is removed). For the adequacy, we use a sentence BERT model [2] to compute the cosine similarity between the generated synthetic data and the reference data.

The experiment results are shown in Table 1. From the experiment results, we can see that across all the dataset sizes, back-translation with beam search gives higher BLEU score than back-translation with sampling. The BLEU score of the synthetic data shows similar trend as the final BLEU score, which indicates the synthetic quality is important for the final translation quality. However, there are not too many correlations between perplexity and the BLEU score. Sampling can generate more fluent synthetic data when the dataset size is large, but it still underperforms compared to beam search for the final performance. Different from perplexity, cosine similarity shows strong correlation with the synthetic data BLEU score and the final BLEU score, which highlights the adequacy of the synthetic data is important for the final translation quality.

To better support our conclusion, we conduct another set of experiments, in which we use an-

[2]https://www.modelscope.cn/models/deepset/sentence_bert

other method to evaluate adequacy of the synthetic data. We cast the adequacy evaluation as a natural language inference (NLI) problem as we suppose for a good translation, the hypothesis should entail the reference and the reference should also entail the hypothesis. In other words, NLI task can help to capture the semantic relation between 2 sentences, which can be helpful for evaluating adequacy which reflects the semantic coverage of hypothesis over reference. We use a BERT (Devlin et al., 2019) model and fine-tune it on MNLI natural language inference dataset [3] as our evaluation model. We use entailment percentage as our evaluation metric, $entailment\ percentage = \frac{entailment\ sentences}{total\ sentences}$. For each synthetic dataset, we compute entailment percentage for hypothesis to reference (whether hypothesis entails reference) and reference to hypothesis (whether reference entails hypothesis), we also report entailment percentage hypo to refe and refe to hypo in which the synthetic sentence passes both entailment from hypothesis to reference and reference to hypothesis.

The experiment results are shown in Table 2. From the experiment results, we can see that when the bilingual dataset is too small such as 10K, the translation model from target language to source language which is used to generate synthetic data is too weak, so the entailment percentage for both beam search and sampling is very low. When the natural bilingual dataset size increases, the entailment percentage for both beam search and sampling increases. However, more importantly to our conclusions is that when the natural bilingual dataset sizes are not too small (50K, 100K, 200K in the experiments), the entailment percentage for beam search is higher than sampling. This indicates that the synthetic data generated by beam search is more adequate than it generated by sampling, which further supports our conclusion that adequacy is more important than fluency for the synthetic data.

Based on our findings, we hypothesize the performance of back-translation can be improved if we can select more adequate or fluent synthetic data. To test this hypothesis, we conduct another set of experiments. We use the 10K bilingual dataset and train a model from target side to source side, we then generate 100k synthetic data using this model. We use 3 different methods to select different amount of synthetic data and mix it with the natural data to train the final model: random selection (simulate the original back-translation), select the most adequate data, select the most fluent data. The adequacy and fluency are evaluated using the methods introduced in the previous paragraphs.

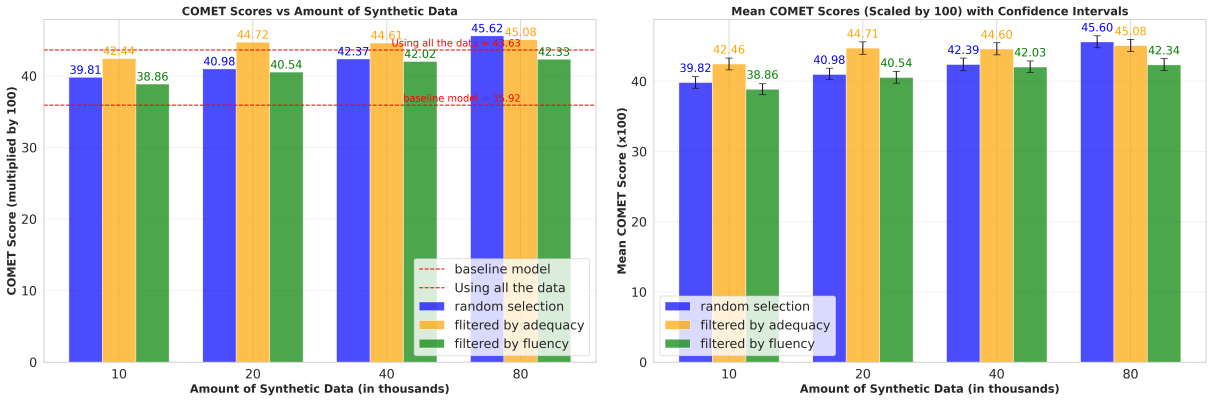
The results are shown in Figure 2, Figure 2a shows the COMET scores of different methods across different synthetic data sizes. To provide statistical significance, we also use bootstrap resampling (Dixon, 2006) to provide 95% confidence intervals, the mean COMET scores of different samples and 95% confidence intervals are shown in Figure 2b. The results show that at most synthetic data sizes, filtering the data by adequacy can improve the performance of back-translation with statistical significance. An important thing to note is that the performance of back-translation is peaked at a certain point and then decreases with the increase of the synthetic data, as suggested in Figure 1 and the results in this section (using all the synthetic data underperforms using only 80K synthetic data), however, when we filter the data by adequacy, we can achieve better performance with less training data for the final translation model. When the synthetic data size is 80K, random selection achieves slightly higher COMET score, but its confidence interval overlaps with that of filtered by adequacy, therefore, we can only conclude they achieve similar results. This is normal because we almost select all the training data (80% training data is selected). The results also indicate that filtering the data by fluency cannot improve the performance, and it may even cause the performance degradation, which is in line with our findings in the previous paragraphs.

Conclusions: For low-resource back-translation, the synthetic data quality plays an important role in the final performance, where the adequacy of the synthetic data is much more important than the fluency. The performance of back-translation can be improved if we can filter the synthetic data based on their adequacy. In this section, the method we evaluate adequacy requires reference for the synthetic data which is not available in real situation. However, it is possible to evaluate adequacy without the reference, for example, using a multilingual language model, we leave systematically evaluate adequacy without reference for future work.

5.2 Codeswitching

We can see that in Figure 1, with the same replacement percentage, codeswitching with target trans-

[3] https://huggingface.co/datasets/glue



(a) (b)

Figure 2: The experiments for filtering the synthetic data by adequacy or fluency. The natural bilingual dataset size is 10K for these experiments. Figure 2a: The COMET scores of different methods. Figure 2b: Mean COMET scores with confidence intervals, the mean and confidence intervals are computed using bootstrap resampling (Dixon, 2006)

replacement percentage	codeswitching	codeswitching dictrg
0.1	15.49%	46.91%
0.15	-	46.97%
0.168	-	46.84%
0.2	15.56%	-
0.3	15.61%	-
0.4	15.55%	-

Table 3: The frequent token percentage for codeswitching and codeswitching with target translation. *codeswitching dictrg: codeswitching with target translation.* The codeswitching with target translation method has much higher frequent token percentage.

lation gives worse performance. We hypothesize it is because replacing source tokens randomly from multiple translations can provide stronger training signal and help to increase the model's generality as the model has to distinguish among different meanings, also, by replacing source tokens randomly from translations, it will not be dominated by the frequent tokens in the target sentences as the replacement procedure of codeswitching with target translation we use cannot make sure that we can replace the source tokens with the correct translations.

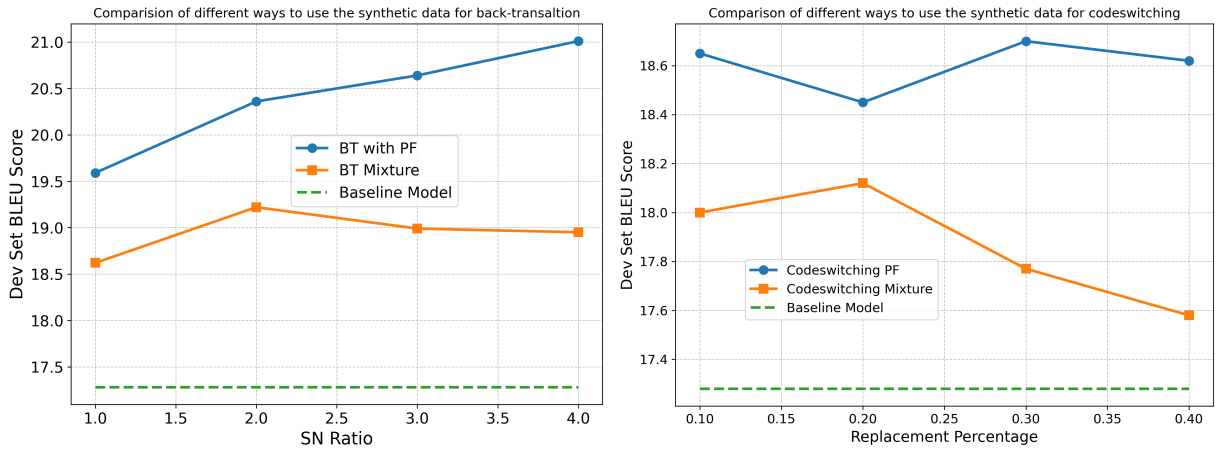
In order to test this hypothesis, we compute how many target translations that replace the original source tokens are frequent tokens in the target side of the bilingual training data. We treat the token in the target side as frequent token if its frequency is larger than 10K in the target side of the 100K bilingual training data. This only results in around 10 tokens in target language. We then compute frequent token percentage as $frequent\ token\ percentage = \frac{frequent\ tokens}{total\ target\ tokens\ used\ to\ replace\ source\ tokens}$. The results are shown in Table 3, from the results we can see that for both methods, the frequent token percentage is similar across different replacement percentages, which is expected because the replacement procedure we use is random regardless of different replacement percentages. However, for codeswitching with target translation, the frequent token percentage is much larger, which indicates that a large fraction of target tokens that used to replace the source tokens are frequent tokens in target side of the bilingual training data. This can help to support our hypothesis that the performance of codeswitching with target translation is worse is because it is dominated by the frequent tokens in target side of the bilingual training data.

Conclusion: For codeswitching, it is important to introduce randomness to the replaced tokens. It can be beneficial to systematically analyse the quality of the synthetic data generated by codeswitching using the similar methods as for back-translation. However, because of computation limitation, we leave it for future work.

6 Using the Synthetic Data

6.1 Back-translation

Figure 1 shows that using more monolingual data causes performance degradation of back-translation. In this section, we will systematically analyse the reasons of it and describe how we can increase the performance with more monolingual data.



(a) (b)

Figure 3: The experiments for back-translation and codeswitching with pre-training and fine-tuning. *PF: pre-training and fine-tuning, SN ratio: synthetic data and natural data ratio, mixture means the model is trained on the mixture of the synthetic data and natural data.* Figure 3a: The experiments for back-translation with pre-training and fine-tuning. Figure 3b: The experiments for codeswitching with pre-training and fine-tuning. The COMET scores are reported in Appendix A.

We first experiment whether the way we use the synthetic data can influence the final performance. In this section, we use a novel method to use the synthetic data, in which we pre-train the final translation model on the augmented synthetic data and then fine-tune it on the natural data. The methods we generate the synthetic data are the same as stated in Section 4, the only difference is the way we use the synthetic data.

The experiment results are shown in Figure 3a, the natural dataset size is 100K. The results indicate that different from traditional way of using the synthetic data, this pre-training and fine-tuning method help model benefit from more synthetic data as the performance increases with the increase of SN ratio. To understand why this method can help to increase the performance, we conduct evaluation to the intermediate pre-trained models which are pre-trained on the synthetic data. We use the similar methods as in Section 5 to evaluate fluency and adequacy separately. For the fluency, we randomly sample 100k data from our monolingual data and fine-tune a German GPT2 model [4], we evaluate every 500 steps and stop the training when the performance does not improve for more than 3 evaluations. For the adequacy, we use a German BERT model [5] to compute the cosine similarity between the model output and the reference on the development set.

SN ratio	perplexity	cosine similarity
1.0	242.59	0.8711
2.0	192.25	0.8819
3.0	169.07	0.8791
5.0	109.58	0.8783

Table 4: Perplexity and cosine similarity of the pre-trained models for back-translation. The perplexity for the monolingual German data is 61.61.

The results are shown in Table 4. The cosine similarity of the pre-trained models shows similar trend as the final performance of traditional back-translation where the model is trained on the mixture of the natural data and synthetic data. This can help to explain why the performance of traditional back-translation peaks at a certain point and then decreases, because more monolingual data hurts adequacy and as we stated in Section 5, adequacy plays an important role for the synthetic data quality and the final translation quality. However, the perplexity decreases with more synthetic data, which means that our pre-trained models can generate more fluent sentences in target language when pre-trained using more synthetic data (which indicates more monolingual target data). This also helps to explain why the model can benefit from more synthetic data using pre-training and fine-tuning method.

[4] https://www.modelscope.cn/models/dbmdz/german-gpt2

[5] https://www.modelscope.cn/models/dbmdz/bert-base-german-cased

SN ratio	perplexity	cosine similarity	final COMET score
1.0	145.93	0.8973	76.28
2.0	140.14	0.8992	76.72
3.0	120.58	0.8956	76.75
5.0	121.10	0.9011	76.59

Table 5: The experiment results for codeswitching pre-training and fine-tuning with different SN ratio. The perplexity and cosine similarity are evaluate on the pre-trained model, the COMET score is the final score on development set. *SN ratio: synthetic and natural data ratio.*

Conclusions: The way we use the synthetic data is important for low-resource back-translation, pre-training the model on the synthetic data and fine-tuning it on the natural data can significantly increase the performance. Compared to adequacy, fluency is more important for the pre-trained model.

6.2 Codeswitching

Following our work for back-translation with pre-training and fine-tuning, we use the similar method for codeswitching. The results are shown in Figure 3b. Different from back-translation, the amount of the synthetic data for these experiments is the same (the SN ratio is 1.0), however, pre-training and fine-tuning is still beneficial as the performance is increased for all the replacement percentages. Moreover, it can increase the robustness of codeswithcing against its hyperparameters, for example, we can still get reasonable results when the replacement percentage is as large as 40%.

To test whether codeswitching also benefits from more synthetic data using pre-training and fine-tuning, we increase the SN ratio and pre-train the model on the synthetic data generated using codeswitching. We use the similar evaluation methods as in pre-training and fine-tuning for back-translation. The perplexity, cosine similarity and final COMET score on development set is reported in Table 5. The results indicate that the performance is increased when the SN ratio increases from 1.0 to 2.0, but larger SN ratios give similar results.

We hypothesize it is because the pre-training synthetic data for codeswitching lacks diversity. As shown in Appendix Table 8, for each source sentence, different replacements will have the same target sentence, so the unique target sentences will

SN ratio	codeswitching PF	codeswitching PF + target DA
1.0	76.28	72.00
3.0	76.75	76.20
5.0	76.59	76.99

Table 6: The COMET score of codeswithcing PF and codeswitching + target DA. *PF: pretraining and fine-tuning, DA: data augmentation.*

equal to the natural dataset regardless of the synthetic and natural data ratio. In order to see the diversity of the pre-training data more precisely, we compute the unique n-gram percentage, as shown in Appendix A Table 9, Table 10 respectively. The unique n-gram percentage is computed as the amount of the unique n-grams in the pre-training data divided by the amount of total n-grams (which contains the same n-grams) in the pre-training data, $unique\ n-gram\ percentage = \frac{uniquen-grams}{totaln-grams}$. We compute unigram, bigram, trigram for both source side and target side of the pre-training data. The results show that the decrease speed of the unique n-gram percentage for codeswitching is significantly larger than back-translation, especially on larger n-grams, the target side falls dramatically when the dataset size increases.

To test this hypothesis, we perform some simple data augmentation on the target side for the codeswitching pre-training data. More specifically, we randomly replace 5% tokens with other tokens in target language and randomly remove 5% tokens. The results are shown in Table 6. When the SN ratio is small, the data augmentation on target side will only hurts the performance, however, with target side data augmentation, the final performance shows similar trend as back-translation pre-training and fine-tuning, the model can benefit from more synthetic pre-trained data. As shown in Table 6, with target DA, the performance increases with the increase of SN ratio, and it finally outperforms codeswitching PF.

It further raises some research questions: Table 4 shows that more fluent pre-trained model gives better final result, will it be better to corrupt only part of the target data so that we can increase the diversity of the pre-trained data and the model can still learn from natural target data during pre-trianing? What is the trend of the final performance if we further increase the SN ratio? We corrupt 10% target tokens in our experiments, how this percentage

methods	BLEU score	COMET score
baseline model	13.67	61.06
back-translation	14.48	64.80
back-translation PF	**17.03**	**65.50**
codeswitching	14.20	62.52
codeswitching PF	14.63	62.58

Table 7: Final results on test set. *PF: pre-training and fine-tuning.*

will influence the performance? Because of computation limitation, we leave these research questions for future work.

Conclusions: Codeswitching also benefits from pre-training and fine-tuning, but it is important to make sure the pre-training data is diverse.

7 Results on Test Set

In this section, we report our final results on WMT test set as we stated in Section 3. We use the best hyperparameters for each method based on development set BLEU score from the previous experiments. All the models are trained using the 100K training dataset. The results are shown in Table 7. From the results, we can see that codeswitching can effectively improve low-resource machine translation without any additional data. And pre-training and fine-tuning, which utilises different way to use the synthetic data, greatly boosts the performance. It is important to note that codeswitching PF can achieve higher BLEU score than traditional back-translation without the need of any additional bilingual or monolingual data, and optimising the usage of the synthetic data using pre-training and fine-tuning can obtain +2.55 BLEU score compared to traditional back-translation technique. These results highlight the importance of analysing the synthetic data.

8 Conclusions

In this paper, we presented a systematic analysis of data augmentation strategies for low-resource neural machine translation, focusing not only on how synthetic data is generated, but also on its quality and usage. Our findings highlight that the synthetic data quality plays a critical role in data augmentation methods. For example, filtering the synthetic data based on adequacy can significantly improve the performance of back-translation, and introducing randomness in token replacement in codeswitching can be beneficial.

Furthermore, we demonstrated that the way synthetic data is used has a substantial impact on performance. Our novel method that utilises pre-training and fine-tuning for data augmentation enhances both back-translation and codeswitching methods, with fluency and the diversity of the pre-training data being more important during pre-training.

Our final results on test set have confirmed the importance of the quality and usage of the synthetic data. For example, pre-training can help codeswitching obtain higher BLEU score than traditional back-translation. And optimising back-tradition can obtain +2.55 BLEU score. These highlight the importance of analysing the synthetic data.

9 Limitations

In this paper, we give a systematic analysis of data augmentation for low-resource machine translation. As shown in (Edunov et al., 2018), data augmentation is also an effective way to increase the performance when the natural dataset is large. It can be helpful to give a systematic analysis when we have large amount of training data. Moreover, we only analyse bilingual machine translation, it is helpful to extend our analyses to multilingual machine translation and unsupervised machine translation.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2016. Neural machine translation by jointly learning to align and translate. *Preprint*, arXiv:1409.0473.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214, Copenhagen, Denmark. Association for Computational Linguistics.

Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. 2019. Autoaugment: Learning augmentation policies from data. *Preprint*, arXiv:1805.09501.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of*

the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Philip M. Dixon. 2006. *Bootstrap Resampling*. John Wiley Sons, Ltd.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics.

Fei Gao, Jinhua Zhu, Lijun Wu, Yingce Xia, Tao Qin, Xueqi Cheng, Wengang Zhou, and Tie-Yan Liu. 2019. Soft contextual data augmentation for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5539–5544, Florence, Italy. Association for Computational Linguistics.

Marjan Ghazvininejad, Hila Gonen, and Luke Zettlemoyer. 2023. Dictionary-based phrase-level prompting of large language models for machine translation. *Preprint*, arXiv:2302.07856.

Fred Jelinek, Robert L Mercer, Lalit R Bahl, and James K Baker. 1977. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63.

Alex Jones, Isaac Caswell, Ishank Saxena, and Orhan Firat. 2023. Bilex rx: Lexical data augmentation for massively multilingual machine translation. *Preprint*, arXiv:2303.15265.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Bohan Li, Yutai Hou, and Wanxiang Che. 2022. Data augmentation approaches in natural language processing: A survey. *AI Open*, 3:71–90.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422.

Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. 2014. Learning and transferring mid-level image representations using convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1717–1724.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

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

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

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

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. 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. Association for Computational Linguistics.

Kai Song, Yue Zhang, Heng Yu, Weihua Luo, Kun Wang, and Min Zhang. 2019. Code-switching for enhancing NMT with pre-specified translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 449–459, Minneapolis, Minnesota. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. 2018. SwitchOut: an efficient data augmentation

algorithm for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 856–861, Brussels, Belgium. Association for Computational Linguistics.

Zhen Yang, Bojie Hu, Ambyera Han, Shen Huang, and Qi Ju. 2020. CSP:code-switching pre-training for neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2624–2636, Online. Association for Computational Linguistics.

A Appendix

we make use of the natural bilingual dataset and an additional bilingual dictionary [6] from source language to target language. We replace the source words with their corresponding translation in target language. Let p denote the replacement percentage in source sentences, $p = \frac{replaced\ tokens}{total\ tokens\ in\ source\ sentences}$. Because the dictionary we use in the project cannot cover all the source tokens, let p_{dic} denote the percentage of the source tokens covered by the dictionary, $p_{dic} = \frac{source\ tokens\ covered\ by\ the\ dictionary}{total\ tokens\ in\ source\ tokens}$. For each token in source sentences, we randomly choose it with probability ϵ, if the token is chosen, we look it up in the dictionary, if it appears, we randomly choose one translation from multiple translations and then replace the source token with the chosen translation, if it does not appear in the dictionary, we do not make any replacements. Because the dictionary cannot cover all the source tokens, in order to reach the replacement percentage we want, we use $\epsilon = \frac{p}{p_{dic}}$.

The method we mention above does not consider the target sentences when generating the synthetic data. In order to examine the effect of this factor, we consider another method codeswitching with target translation in which we take the target sentences into account when generating the synthetic data. Let p_{dictrg} denote the percentage of the source tokens covered by the dictionary and one of the translations in the dictionary also appears in the corresponding target sentences. For each token in the source sentences, we randomly choose it with probability ϵ_{trg}, if the token is chosen, we look it up in the dictionary, if it appears and one of its translations also appears in the corresponding target sentences, we replace it with the translation that appears in the target sentences, if more than one translations appear in the target sentences, we choose the first one, otherwise, we do not make any replacements. In order to reach the replacement percentage we want, we use $\epsilon_{trg} = \frac{p}{p_{dictrg}}$.

In this work, we utilized the WMT17 dataset, the pre-trained GPT-2 model, and the pre-trained BERT model. We outline the licensing terms and conditions for each resource to ensure transparency and compliance with their respective usage policies.

The WMT17 dataset, provided as part of the Workshop on Machine Translation (WMT) shared task, is publicly available for research purposes. The dataset is distributed under terms that allow non-commercial use, as specified in the shared task documentation. Researchers are encouraged to use the dataset for benchmarking and evaluation in machine translation tasks.

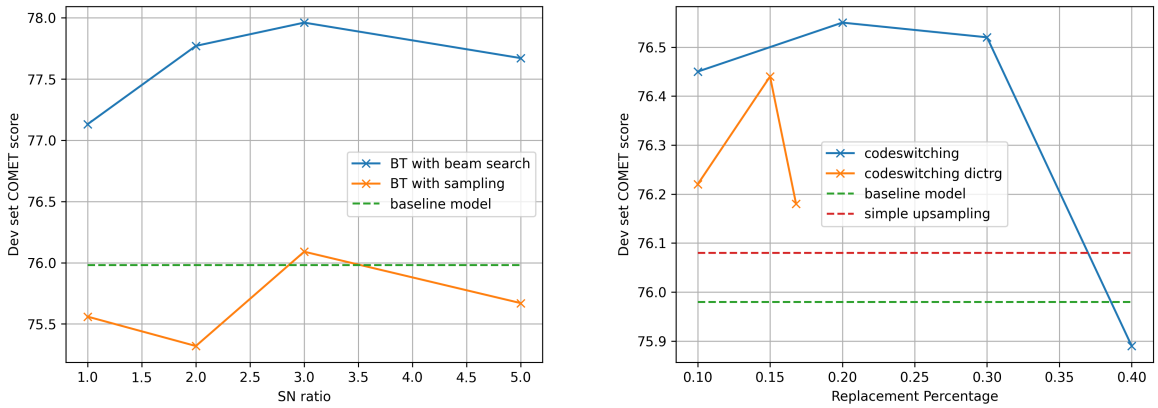
The GPT-2 model is released under an open-source license. The model and its associated code are available on GitHub under the MIT license, which permits unrestricted use, modification, and distribution, provided that proper attribution is given to the original authors. This permissive license allows for both research and commercial applications, making GPT-2 a widely adopted resource in natural language processing.

The BERT model is distributed under the Apache 2.0 license. This license allows users to freely use, modify, and distribute the model, provided that proper attribution is given to the original authors. Additionally, the Apache 2.0 license includes a patent grant, which protects users from patent claims by contributors to the model. This makes BERT a robust and legally safe choice for research and development.

[6] https://github.com/facebookresearch/MUSE##ground-truth-bilingual-dictionaries

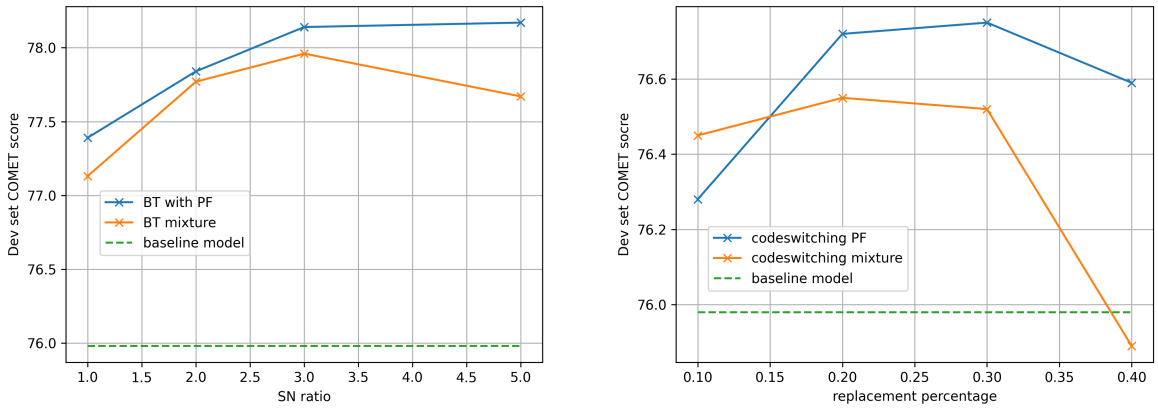
original source sentence	x1 x2 x3 x4
original target sentence	y1 y2 y3 y4
augmented source sentences	xy1 x2 x3 x4; x1 xy2 x3 x4; x1 x2 xy3 x4
augmented target sentences	y1 y2 y3 y4; y1 y2 y3 y4; y1 y2 y3 y4

Table 8: Illustration of the pre-training synthetic data for codeswitching. *xi: source tokens, yi: target tokens, xyi: one of the translations of the source tokens in target language.*



(a) (b)

Figure 4: The experiment results for back-translation and codeswitching with different synthetic data generation strategies. Figure 4a: The experiment results for back-translation. *SN ratio: synthetic and natural data ratio, BT: back-translation.* Figure 4b: The experiment results for codeswitching. *codeswitching dictrg: codeswitching with target translation.*



(a) (b)

Figure 5: The experiments for back-translation and codeswitching with pre-training and fine-tuning. *PF: pre-training and fine-tuning, SN ratio: synthetic data and natural data ratio, mixture means the model is trained on the mixture of the synthetic data and natural data.* Figure 5a: The experiments for back-translation with pre-training and fine-tuning. Figure 5b: The experiments for codeswitching with pre-training and fine-tuning.

pre-training data	unigram source	unigram target	bigram source	bigram target	trigram source	trigram target
100k	1.15%	1.26%	39.56%	39.13%	81.41%	79.71%
200k	0.57%	6.28%	27.28%	19.56%	65.19%	39.85%
300k	0.38%	0.42%	21.77%	13.04%	56.79%	26.57%
500k	0.3%	0.25%	16.11%	7.82%	46.92%	15.94%

Table 9: The unique n-gram percentage for codeswitching pre-trained on synthetic data generated using only the natural bilingual data and a bilingual dictionary.

pre-training data	unigram source	unigram target	bigram source	bigram target	trigram source	trigram target
100k	1.12%	1.34%	26.6%	36.92%	60.88%	74.33%
200k	0.6%	0.7%	20.95%	29.53%	54.93%	68.16%
300k	0.41%	0.48%	17.96%	25.61%	51.02%	64.08%
500k	0.26%	0.29%	14.7%	21.16%	46.18%	58.64%

Table 10: The unique n-gram percentage for back-translation with pre-training and fine-tuning.