Experiment Setup

In this study, I explored two strategies to improve neural machine translation performance over the baseline: Byte Pair Encoding (BPE) and tuning the hyper-parameter "batch size."

Github rep: https://github.com/qing-dai/atmt_2023
Most of my files can be found in the main branch. The rep is private, and the invitation to https://github.com/arnisafazla has been sent.

Strategy 1: Tuning Batch Size

In the baseline model, the batch size is 1, I want to try different values of this hyper parameter to see the impact. For the rest of the hyper parameters, no change. I took advantage of the check point to get better result. The data I used is the "prepared" files. I trained the model in Colab, and saved the command files in .ipynb file.

I tried two different values 128 and 1280.
The two command files are batch_size_128.ipynb and batch_size_1280.ipynb

When the batch size is 128,

INFO: Epoch 099: valid_loss 2.04 | num_tokens 9.14 | batch_size 500 | valid_perplexity 7.7
INFO: Epoch 100: loss 1.27 | lr 0.0003 | num_tokens 9.586 | batch_size 126.6 | grad_norm 5.476 | clip 0.5823
INFO: Epoch 100: valid_loss 2.04 | num_tokens 9.14 | batch_size 500 | valid_perplexity 7.72
INFO: Epoch 101: loss 1.265 | lr 0.0003 | num_tokens 9.586 | batch_size 126.6 | grad_norm 5.35 | clip 0.5696
INFO: Epoch 101: valid_loss 2.04 | num_tokens 9.14 | batch_size 500 | valid_perplexity 7.71
INFO: No validation set improvements observed for 3 epochs. Early stop!

When the batch size is 1280,

INFO: Epoch 175: valid_loss 2.06 | num_tokens 9.14 | batch_size 500 | valid_perplexity 7.83
INFO: Epoch 176: loss 1.41 | lr 0.0003 | num_tokens 9.271 | batch_size 1250 | grad_norm 1.639 | clip 0
INFO: Epoch 176: valid_loss 2.06 | num_tokens 9.14 | batch_size 500 | valid_perplexity 7.83
INFO: Epoch 177: loss 1.409 | lr 0.0003 | num_tokens 9.271 | batch_size 1250 | grad_norm 1.638 | clip 0
INFO: Epoch 177: valid_loss 2.06 | num_tokens 9.14 | batch_size 500 | valid_perplexity 7.83
INFO: No validation set improvements observed for 3 epochs. Early stop!

Batch size	BLEU
Baseline (1)	17.7
128	27.1
1280	26.2

The translated test files are

128: translations_128.txt, translations_128.p.txt (post-processed)
1280: translations_1280.txt, translations_1280.p.txt (post-processed)

Leveraging the baseline checkpoint naturally led to improved BLEU scores for our new models. However, increasing the batch size beyond a certain point did not yield further improvements. A comparison of translations reveals that a batch size of 128 offers better quality than 1280. The latter appears to overfit, resulting in more errors, whereas the former shows an enhanced ability to capture patterns more effectively than the baseline. This suggests that the model with a batch size of 128 benefits from the baseline checkpoint, whereas the 1280 model may miss some optimal learning opportunities.

e.g.:
Original: "He abuses his authority."
Baseline: "He's trying to his mother of his mother."
Batch size 128: "He's trying to his authority."
Batch size 1280: "He's stalling his mother."

Original: Tom is smarter than me.
Baseline: Tom is more than me than me.
Batch size 128: Tom is taller than me.
Batch size 1280: Tom is more than me.

Additionally, both models adeptly handle common phrases like “Thank you” and “I’m happy,” hinting at the models' proficiency with frequent patterns. This observation suggests that while larger batch sizes don't necessarily enhance translation quality, there is merit in exploring smaller sizes to optimize performance. The chosen values for batch size 128 and 1280 do not conclusively establish the former as ideal, but they do indicate that smaller batch sizes could lead to better results.

Strategy 2: BPE

The baseline model's limitation to complete words prompted the use of BPE, which subdivides words into frequent subunits or subwords, potentially refining translation quality. Applying BPE to the preprocessed data—already normalized, tokenized, and truecased—followed the best practices for languages sharing an alphabet, as detailed in the subword-nmt GitHub repository. This approach ensures segmentation consistency by learning BPE on concatenated French and English training texts.

I set the vocabulary threshold to one, adjusting for the modest training data size, which rendered larger thresholds ineffective. Consequently, the final vocabularies comprised 5,629 English and 6,776 French subwords, exceeding the baseline model's vocabulary size.

All BPE commands are documented in "BPE_commands." The resulting BPE tokenized files reside in atmt_2023/data/en-fr/new_BPE. To binarize these files, I modified preprocess.py, setting the source and target language word thresholds to one. The binarized BPE data, stored in atmt_2023/data/en-fr/BPE_final, served as input for subsequent model training.

For training, I mirrored the baseline's learning rate and batch size parameters: a learning rate of 0.0003 and a batch size of one. The training command used was: python3 train.py --save-dir assignments/03/baseline/checkpoints --restore-file None --data data/en-fr/BPE_final/ --lr 0.0003 --batch-size 1

INFO: Epoch 032: loss 2.331 | lr 0.0003 | num_tokens 9.47 | batch_size 1 | grad_norm 52.05 | clip 0.9992
INFO: Epoch 032: valid_loss 3.36 | num_tokens 9.83 | batch_size 500 | valid_perplexity 28.7
INFO: Epoch 033: loss 2.324 | lr 0.0003 | num_tokens 9.47 | batch_size 1 | grad_norm 52.72 | clip 0.9988
INFO: Epoch 033: valid_loss 3.35 | num_tokens 9.83 | batch_size 500 | valid_perplexity 28.5
INFO: Epoch 034: loss 2.301 | lr 0.0003 | num_tokens 9.47 | batch_size 1 | grad_norm 51.86 | clip 0.9984
INFO: Epoch 034: valid_loss 3.34 | num_tokens 9.83 | batch_size 500 | valid_perplexity 28.3
INFO: No validation set improvements observed for 3 epochs. Early stop!

Model	BLEU
Baseline	17.7
BPE	12.9

The translated test files:
test_translations_BPE_final.txt, test_translations_BPE_final.p.txt (post-processed)

Without a checkpoint to build upon, surpassing the baseline BLEU score was challenging using the BPE model. Nonetheless, BPE's translations were comprehensive, unlike some omissions noted in the baseline. This suggests that subword dictionaries help retain original content.

e.g.
Original: I'm happy because you're here.
Baseline: Blank.
BPE: I'm glad, because you're happy, because you're here.

Both BPE and baseline models exhibited redundancy, with shorter sentences generally yielding more coherent translations—a finding consistent with existing research indicating the complexity of translating longer sentences.

Deciding which model outperforms can be complex; while both capture certain linguistic patterns, grammatical errors and nonsensical outputs are still present.

e.g.
Original: I know that Tom doesn't like Mary.
Baseline: I know Tom doesn't know Mary's don't like Mary.
BPE: I know Tom doesn't like Mary don't like Mary.

Remarks

Throughout this assignment, the iterative process of BPE data processing has deepened my understanding of the NMT pipeline. Engaging with the nuances of model training, particularly the impact of batch size, has revealed the interconnected stages from initial training through to translation, post-processing, and BLEU score evaluation. Although the majority of my effort was invested in understanding the provided code files, BPE data handling and model training, the experience has significantly boosted my confidence in managing and training NMT models in the future.

In reflection, with more time and resources, experimenting BPE training with a reduced learning rate could potentially yield better results, considering the finer granularity of subwords. Adjusting other hyper-parameters, starting from a BPE checkpoint, may also enhance BLEU scores like what happened in the first strategy. Moreover, implementing BPE dropout could be beneficial as it can introduce variability in the way words are tokenized during training, effectively argumenting the training data and thus dealing with rare words or out-of-vocabulary terms.