ATMT Assignment 3 - Github

Sam Wallace

October 2024

1 Introduction

Neural Machine Translation (NMT) has become an essential tool for translating languages, especially in low-resource settings where traditional methods face challenges due to limited data. This assignment focuses on improving NMT from English to French, experimenting with Byte Pair Encoding (BPE) and hyperparameter tuning to enhance translation quality. BPE, a subword segmentation approach, is widely used in NMT to handle rare words and improve vocabulary coverage, allowing models to generate translations more effectively by splitting words into smaller, more manageable units.

Alongside BPE, hyperparameter tuning—particularly in this discussion, learning rate adjustments—plays a significant role in model performance. The learning rate dictates how the model updates its parameters, influencing convergence and accuracy. This project explores the impact of different learning rates on model stability and BLEU scores, aiming to identify an optimal balance that can achieve higher translation quality. Through these experiments, this assignment seeks to understand how both BPE encoding and careful tuning of hyperparameters can contribute to more robust and accurate NMT models.

To ease the navigation of my Github Repository, below I have included paths to various items of interest.

1.1 Checkpoints

Model checkpoints can be found under the respective paths

- **assignments/03/baseline_lr_0001** - Model with learning rate 0.0001
- **assignments/03/baseline_lr_001** - Model with learning rate 0.001
- **assignments/03/baselineBPE** - BPE train
- **assignments/03/baseline** - Original model

1.2 Translations

Translations can be found under the respective paths

- **assignments/03/translations_lr_00001.p.txt** - Smallest learning rate model
- **assignments/03/translations_lr_001.p.txt** - Largest learning rate model
- **assignments/03/translations_BPE.p.txt** - BPE translations

1.3 Data

BPE model has its own data due to the decomposing of words into subwords.

data/en-fr-BPE

2 BPE Encoding

I decided to peruse BPE encoding [1] for my first experiment in improving the translation accuracy on low resource NMT. To achieve this I completed three stages

1. Modify the preprocess.sh script to include BPE
2. Train a new model on these translations
3. Evaluate the performance vs non-BPE encoded text

2.1 Obtaining the BPE

To do this I utilised the python-package subword-nmt, this python package provides an easy interface to apply BPE encoding to word segments, below is a short example usage of the package taken from github [2]

To apply byte pair encoding to word segmentation, invoke these commands:

```
subword-nmt learn-bpe -s {num_operations} < {train_file} > {codes_file}
subword-nmt apply-bpe -c {codes_file} < {test_file} > {out_file}
```

To segment rare words into character n-grams, do the following:

```
subword-nmt get-vocab --train_file {train_file} --vocab_file {vocab_file}
subword-nmt segment-char-ngrams --vocab {vocab_file} -n {order} --shortlist {size} < {test_fil
```

The original segmentation can be restored with a simple replacement:

```
sed -r 's/(@@ )|(@@ ?$)//g'
```

The specific modifications I made where in the preproccess.sh file as following

```
# Train BPE on concatenated source and target data (we use 10,000 merge operations here; adjust as needed)
cat $data/preprocessed/train.$src $data/preprocessed/train.$tgt | subword-nmt learn-bpe -s 10000 > $data/preprocessed/bpe.codes

# Apply BPE to each data split (train, valid, test, tiny_train)
for split in train valid test tiny_train; do
    subword-nmt apply-bpe -c $data/preprocessed/bpe.codes < $data/preprocessed/$split.$src > $data/preprocessed/$split.bpe.$src
    subword-nmt apply-bpe -c $data/preprocessed/bpe.codes < $data/preprocessed/$split.$tgt > $data/preprocessed/$split.bpe.$tgt
done
```

2.1.1 Training the Model

The model was trained as in previous assignments using the standard command

```
❯ python train.py --data data/en-fr/prepared_BPE --source-lang fr --target-lang en --save-dir assignments/03/baselineBPE/checkpoints
```

2.2 Evaluation

```
❯ cat assignments/03/translations_BPE.p.txt | sacrebleu data/en-fr-BPE/raw/test.en
{
 "name": "BLEU",
 "score": 5.0,
 "signature": "nrefs:1|case:mixed|eff:no|tok:13a|smooth:exp|version:2.4.3",
 "verbose_score": "31.5/8.2/2.8/0.9 (BP = 1.000 ratio = 1.373 hyp_len = 5345 ref_len = 3892)",
 "nrefs": "1",
 "case": "mixed",
 "eff": "no",
 "tok": "13a",
 "smooth": "exp",
 "version": "2.4.3"
}
```

Clearly, this experiment was a failure, a BLEU score of 5.0 represents a major decrease in quality from the original 17.7 provided. I hypothesis the reason for this failing was due to some errors in the construction of dictionaries for the translation phase of the program. I believe that the construction of dicts for the newly created BPE pairs was in some faculties lacking and hence the translations are unrecognizable.

Further investigation below shows that a lot of the translations produced by the model after post-processing are whilst still consisting of a majority of English which suggests the model can at least model English language lack any particular meaning.

```
1  Come on the people to be who.
2  I'm father you had be past in the sleep of the sleep.
3  I'm the teacher?
4  He's three to his Japan in his sleep.
5  He is us than than! The hospital! Hospital.
6  I'd like to be so long I'd like to be long.
7  I don't hope day cats to hope cats.
```

3 Hyper-parameter Tuning

Hyper-parameter tuning is a critical step in optimizing neural machine translation (NMT) systems, significantly impacting their performance. Among the various hyper parameters, the learning rate plays a pivotal role in determining how quickly or slowly a model learns during training. It controls the size of the steps taken toward a minimum of the loss function, influencing the convergence speed and the stability of the training process.

For this reason I have chosen to experiment with results obtained by using different learning rates specifically I have trained two models on

- lr = 0.0001
- lr = 0.001

Currently the model is trained on a lr of 0.0003

```
parser.add_argument('--lr', default=0.0003, type=float, help='learning rate')
```

To achieve this I ran the following training commands

```
❯ python train.py --data data/en-fr/prepared --source-lang fr --target-lang en --lr 0.0001 --save-dir assignments/03/baseline_lr_0001
❯ python train.py --data data/en-fr/prepared --source-lang fr --target-lang en --lr 0.001 --save-dir assignments/03/baseline_lr_001
```

3.1 Evaluation

This section was more successful than the attempted BPE above,

Model	BLEU-Score
LR-0.0003 (Original)	17.7
LR-0.001	12.3
LR-0.0001	13.7

Certainly, the original model outperforms both retrained model with a different learning parameter. However I am interested to note that the model with the lower learning rate was able to outperform the one with the larger learning rate. In this case, I believe the lower learning rate may have allowed the model to converge more effectively by taking smaller, more precise steps during gradient descent. This often leads to better alignment between the source and target languages, as the model can capture subtleties in linguistic structure and semantic relationships without over-adjusting weights too quickly, which can happen with a higher learning rate.

To further, explore this avenue I would like to re-run the experiment, whilst also tracking how the learning rate has evolved and producing a graph to show that a lr of 0.001 is too high for this task and demonstrate its erratic nature.

4 Final Remark

Whilst, lacking clear improvements utilizing the techniques I attempted I still believe that with further exploration BPE can be shown to be a more effective tool for increasing BLEU scores and hence translation accuracy. To do this I would re-run the experiment but be more careful with the dictionary command and also spend more time familiarizing myself with the python package that I used above to generate the BPE encodings. In the case of hyper parameter tuning, it is already a well established fact that the learning rate can greatly impact performance [3]. So in further experimentation it would be interesting to perhaps experiment more parameters specifically drop out to see how we can

make models more robust to over fitting. Perhaps by including a large corpus of in-domain tasks and train two models, one utilizing dropout and one not and then testing on some out of domain corpus to see how the dropout model is more robust to unseen types of translations. I.e. this could be training on a conversational dataset and then testing on something with more formal language such as business documents.

References

[1] R. Sennrich, B. Haddow, and A. 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.

[2] R. Senrich. Subword-nmt.

[3] Y. Wu, L. Liu, J. Bae, K.-H. Chow, A. Iyengar, C. Pu, W. Wei, L. Yu, and Q. Zhang. Demystifying learning rate policies for high accuracy training of deep neural networks, 2019.