ATMT Assignment 3

Topic: improving a low-resource NMT system

Deadline: Tuesday, November 5, 2024, 14:00

Important submission info:

- **Submission format: GitHub, pdf**
- Please name your PDF submission files like this: olatusername_atmt_assignmentxx.pdf, e.g. `mmuster_atmt_assignment03.pdf`
- For this assignment, you are asked to submit a report as a PDF file and your code and test set translations in a GitHub repository (with a link to the repository included in your PDF report). **NOTE:** we strongly recommend using GitHub, but if you do not want to, you can also download code repository from GitHub as a zip archive, edit it locally and then submit a zip archive of your local code repository aling with your PDF report.
- You need to submit the assignment on OLAT using the tab for assignment 3. Please hand it in on time. The submission deadline is given above. After this time, the tab will be closed. We do not consider Git commits that happened after the deadline.
- You are encouraged to work in groups of two. Please make sure every file clearly states **both** names. For pairs, only one person should submit the assignment.

1 Assignment Description

In this assignment, we focus on improving a low-resource system which serves as our baseline. In the atmt repository, you can find the checkpoints of the trained baseline in `assignments/03/baseline/checkpoints` and the data that was used for training in the folder `data/en-fr/` which has been prepared in the same way as the data used in assignment 1. Below you can see the BLEU score that this system achieved on an in-domain test set:

```
{
 "name": "BLEU",
 "score": 17.7,
 "signature": "nrefs:1|case:mixed|eff:no|tok:13a|smooth:exp|version:2.3.1",
 "verbose_score": "45.5/22.8/13.0/7.3 (BP = 1.000 ratio = 1.295 hyp_len = 5042 ref_len = 3892)",
 "nrefs": "1",
 "case": "mixed",
 "eff": "no",
 "tok": "13a",
 "smooth": "exp",
 "version": "2.3.1"
}
```

The baseline system is trained with 10,000 random sentences from the Tatoeba corpus which is a collection of sentences and their translations in many languages. We will translate from French into English. Obviously, we could improve the translation quality of our system by adding more training data. But for the sake of this assignment, these 10,000 sentences are the only parallel training material we consider (you can use additional monolingual material if you wish). Your job is to find other ways to improve the translation quality on the test set (all data is provided in `data/en-fr/raw`). In the lectures, you have already looked at some possible strategies that you can apply now:

- Translate on the level of subwords (using BPE or a related algorithm - see [5] or [2])
- Combine NMT with autoencoding of the target language (see [1])
- Tune a hyper-parameter - to simplify this process you just need to try two other values for your chosen hyper-parameter (e.g. model depth, learning rate, dropout, batch size, ... see [6])

Another strategy that has been proposed in the literature is using a lexical model [3] which is intended to counter the problem of choosing a translation that fits the context but does not reflect the actual meaning of the source sentence. This is done by allowing the model to look at the source word embeddings when producing a new translation. The paper gives you enough information to implement this in the decoder. You can implement this lexical model as one of your strategies for improving the low-resource system:

- Implement a lexical model to improve lexical choice (details in chapter 4 [3])

Online BPE-dropout [4] may be a good strategy in a low-resource scenario. During training, we re-segment the training data before every new epoch by randomly dropping out some of the merges with a given probability. In this way, we are effectively creating more training data without needing access to more parallel data. You can implement this as one of your strategies for improving the low-resource system:

- Change the code such that the training data is externally re-segmented before every new epoch and then reloaded for continued training. The implementations of BPE from [5] and [2] both support BPE-dropout.

For this assignment, choose two strategies for improving low-resource NMT that you want to test and compare to your baseline. Of course, you can also propose other ideas, ideally techniques that have proven effective in the research literature. When making your choice, think about whether you want to implement something or whether you just want to adapt the training data. In the later sections, we give you some directions for preprocessing the training data and where in the code you need to make changes for the implementation-based strategies. These are only some starting points. You need to come up with your own experiment design and evaluation of the results. As you saw in the last assignment, this requires some planning. Therefore, make sure you start early with this assignment so that you have enough time to design and run your experiments and to write the report. You have two weeks to complete the assignment. Depending on which strategies you choose, you need to train two or three different models. Details for submission are given below.

Submission: Please add a link to your GitHub repository in your PDF report (instructions on using GitHub can be found later in this document). The repository should include the test set translations (translated with your improved models) as well as any code changes and preprocessing/postprocessing scripts.[1] Your PDF report should be 2-3 pages including figures / tables and contain the following:

- A detailed description of your experiment setup (Why did you chose these two strategies? What data did you use? How did you preprocess it? What changes did you make in the code? Which hyper-parameters did you use for training your models? What training commands did you use? How did you evaluate your models? ...)
- A suitable presentation and discussion of your results compared to the baseline (table, visualization, qualitative analysis, ...)
- A final remark on what you learned in this assignment and what you would do differently next time

[1] **NOTE:** If you do not want to use GitHub, you can also download code repository as a zip archive, edit it locally and then submit a zip archive of your local code repository with your PDF report.

2 Preprocessing & Postprocessing

In the atmt repository, we've added our preprocessing script that was used to prepare the training data for the baseline system (`assignments/03/preprocess_data.sh`). You can see that we normalize, tokenize and truecase the text before we create the required training format and the vocabulary files. We use a maximum vocabulary size of 4000 (`--num-words-src 4000`) and do not make any restrictions on how many times a token needs to occur to be in the vocabulary (`--threshold-src 1`). The postprocessing script is the one you also used in the first assignment before computing the raw BLEU scores. It simply reverses the baseline preprocessing steps after translation. If you change any parts of the pre- and postprocessing, you should create your own scripts, upload them to your working GitHub repository and be sure to mention these in your PDF report.

3 Framework Code

For those who would like to implement something in this assignment, we provide some guidelines in the code. The lexical model should be implemented in the file `seq2seq/models/lstm.py`. The parts where you need to add code for the lexical model are marked with comments that start with:

```
# __LEXICAL:
```

The online BPE-dropout should be added in the file `train.py`. If you decide to implement something else, please look through the code carefully to decide where it should be added. In case you run into problems, it may be helpful to consult the PyTorch documentation.

4 Git Repository

For this assignment, you will work with your own repository on GitHub. To do this, you need a GitHub account. If you do not have one yet, you can create it here. Check out the free benefits you get as a university student[2].

Next, you need to "`fork`" the atmt repository. This will create a personal copy of the repository that you can make changes to. You should see the forked repository listed in your own repositories. Finally, to work on your own repository locally, you need to clone your repository using the steps below (check that the URL with your username is correct):

```
git clone https://github.com/your_username/atmt
```

You can update your local repository like this:

```
git pull origin main
```

[2] GitHub Education

To add your local changes to the remote repository, you can check which files you changed or newly created:

```
git status
```

After that, you need to add every file that you want to copy to the remote repository with:

```
git add file_name
```

Now, you need to confirm your changes and add a short message, e.g.:

```
git commit -m "fix some error"
```

Finally, we copy the confirmed local changes to the remote repository:

```
git push -u origin main
```

The above instructions are for public forks. This means everyone will be able see your repository and your changes. If you want your work to be private, you can follow the steps here. Remember to add `davidguzmanp` as a collaborator if you choose to work with a private fork, otherwise we will not be able to grade your assignment.

In case you need some practice with Git, here are some good resources. If you have a question that Google cannot answer, you can always post to the OLAT forum or ask in the tutorial.

Please let us know in the OLAT forum if you have problems or something is unclear.
Good luck!

References

[1] Anna Currey, Antonio Valerio Miceli Barone, and Kenneth Heafield. Copied monolingual data improves low-resource neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 148–156, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[2] Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[3] Toan Nguyen and David Chiang. Improving lexical choice in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 334–343, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[4] Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online, July 2020. Association for Computational Linguistics.

[5] 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.

[6] 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*, pages 211–221, Florence, Italy, July 2019. Association for Computational Linguistics.