MT Exercise 1

Topic: Evaluation of MT Systems and Preprocessing

Submission: Tuesday, March 18 2025, 14:00

Submission Instructions:

- **Submission file format: zip**
- Please follow our file naming convention: olatusername_mt_exercise_xx.zip, for instance `mmuster_mt_exercise_01.zip`
- For this exercise you will submit a Python script and a PDF file. Submit both in the same zip folder.
- Please submit via the exercise submodule on OLAT. Submission is open only until Tuesday, 14:00.
- This is a training exercise and will *not* be graded. The goal is to get you acquainted with the format of the exercises and give you a chance to test your own understanding of the content so far.

1 Automatic Evaluation[1]

You now know that BLEU (Bilingual Evaluation Understudy) is by far the most popular and most widely used automatic evaluation metric. In this exercise you will write a script that computes the BLEU score of a translated text. You can assume that there will always be one single reference translation only. Follow these steps:

1. Write a function to open a text file and read all sentences into memory, as strings. Then the function should tokenize all sentences and return them. For tokenization you can use Sacremoses, see below for an example call.

2. Write a function that computes ngram precision (p) of a translated text (hypothesis) and a reference, given a particular ngram order. Implement the formula shown in the lecture and do not forget about clipping:

$$p = \frac{correct}{hyp\ length} \tag{1}$$

3. Write a function that computes ngram precision of a list of translated texts (hypotheses) and references, given a particular ngram order (using the function from 2.).

4. Write a function that computes the final BLEU score for a given set of hypotheses and references. Use the function from 3. to compute the ngram precisions of different orders. Compute the geometric mean (P) from all ngram precisions using the formula from the lecture slides (see below). For this exercise, set the weights λ to 1 and the maximum ngram order N to 4.

$$P = (\prod_{n=1}^{N} \lambda_n\ p_n)^{\frac{1}{N}} \tag{2}$$

   Also, the function should compute the brevity penalty (BP) using the formula shown below. The next section shows how to exponentiate numbers in Python.

$$BP = \min(1.0, \exp(1 - \frac{ref\ length}{hyp\ length})) \tag{3}$$

   Finally the function should combine the intermediate results above into the final BLEU score and return it. The correct formula is:

$$BLEU = BP * P \tag{4}$$

5. Write a `main` function that reads in two text files that are specified as arguments on the command line (using the function from 1.). The tokenized sentences should be given to the function that computes BLEU, and your program should output the final BLEU score as a single number.

[1] Based on an exercise by Chantal Amrhein. Thanks Chantal!

You can test your implementation with the two sample input files `test_reference.txt` and `test_hypothesis.txt` – they are included in the exercise folder. To validate your implementation, compare against `SacreBLEU`[2]:

```
cat [hypothesis] | sacrebleu [reference]
```

Your script should output very similar BLEU scores. To also check whether clipping works as intended: use the files `test_clipping_reference.txt` and `test_clipping_hypothesis.txt`, and print the individual ngram precisions to debug. In this case, the precision of ngrams of order 1 should be $\frac{1}{7}$, and 0 for all other ngram orders.

Sample usage of the Moses tokenizer, Sacremoses[3]:

```python
from sacremoses import MosesTokenizer

mt = MosesTokenizer(lang='en')

sentence = "This is a house."
tokenized = mt.tokenize(sentence)

print(tokenized) # prints ['This', 'is', 'a', 'house', '.']
```

Sample usage of exponentiation in Python:

```python
import numpy as np

exp_of_1 = np.exp(1)

print(exp_of_1) # prints 2.71828182846
```

[2] https://pypi.org/project/sacrebleu
[3] https://pypi.org/project/sacremoses

2 Impact of Postprocessing on Translation Quality

In this exercise, you are going to investigate how BLEU scores are influenced by different methods of postprocessing. The NMT toolkit we are working with is JoeyNMT[4], a tool built with Pytorch, for educational purposes. For now, you do not have to train your own model: you can download one that was already trained for you.

2.1 Installing JoeyNMT

Make sure virtualenv and Python3 are installed on your system, by trying those commands:

```
virtualenv # if this command fails: pip3 install virtualenv
python3 # if this command fails, install Python3 for your OS
```

Clone a convenient repository we created for you, in the desired place:

```
git clone https://github.com/marpng/joeynmt-toy-mode
cd joeynmt-toy-models
```

Execute this script to create a virtualenv:

```
./scripts/make_virtualenv.sh
```

Activate this virtualenv:

```
source venvs/torch3/bin/activate
```

You will now install several pieces of software inside your virtual environment, among which `joeynmt`. Due to some recent updates in several Python packages, directly installing `joeynmt` will cause problems later on. You therefore need to install some packages manually first. Execute the following commands:

```
pip uninstall setuptools
pip install setuptools==59.5.0
pip install torchtext
```

Download and install software by running the following script (this includes scripts for preprocessing, Python packages such as torch and joeynmt):

```
./scripts/download_install_packages.sh
```

Those scripts take care of a lot of details for you, make sure to study them closely and understand all the steps they take.

[4] `https://github.com/joeynmt/joeynmt`

2.2 Downloading a Trained JoeyNMT Model

We will use a pretrained JoeyNMT transformer model for German→English found here:

`https://github.com/joeynmt/joeynmt#iwslt14-deen`

This script explains what the individual steps are and in which order they were executed to reproduce this trained model. Have a look at all of those commands.

`https://github.com/joeynmt/joeynmt/blob/main/scripts/get_iwslt14_bpe.sh`

Then download the model tar ball, and unpack:

```
wget https://cl.uni-heidelberg.de/statnlpgroup/joeynmt2/transformer_iwslt14_deen_bpe.tar.gz
tar -xzvf transformer_iwslt14_deen_bpe.tar.gz
```

Some logistics to rename and move files to a reasonable folder:

```
# only run this if extracting the tar ball was successful
rm transformer_iwslt14_deen_bpe.tar.gz
mkdir models
mv transformer_iwslt14_deen_bpe models/
```

Take care not to move files around as they might be referenced in the scripts or configs. Be sure to change those corresponding paths as well if you really want to move files to a different directory.

2.3 Translating with a Trained Model

For testing translation, you will be using this raw test set pair:

```
data/test.raw.de
data/test.raw.en
```

These files are not preprocessed, because JoeyNMT has built in pre-tokenizers, tokenizers, casing systems as well as a function to learn subword tokenization (BPE). You will learn about these functions in more detail in the future.
For now it is important to understand the basic series of steps necessary for translation:

```
raw -> tokenized -> truecased -> byte-pair encoding (BPE)
```

Consistency and reproducibility is paramount here. All of these steps should be undone in the same way after translation. In our case JoeyNMT handles this automatically, so testing the output hypothesis translation against the reference is possible without further postprocessing. This is not the always the case, so special attention has to be given to the pre- and postprocessing of your data.

2.4 Investigate impact of postprocessing on BLEU

First, we will translate our test file `data/test.raw.de` to English. Then, you will investigate the following question: how does postprocessing influence BLEU scores? The motivation for this is that reporting BLEU scores is not entirely standardized:

- some people report BLEU on detokenized text, others on tokenized text
- if text is still tokenized, different people use different tokenizers
- some people report BLEU on lowercased text – others dont

This creates some unintended differences between evaluations (and opportunities for light cheating) in research papers and it is not obvious how those choices influence the result.

Compute BLEU scores with SacreBLEU[5] (already installed if you followed the instructions above), using different versions of the translation (called `hyp`) and the reference (called `ref`). Take a look at `scripts/evaluate.sh` to help you get started.

Details about SacreBLEU implementation

In order to avoid tokenization differences, SacreBLEU expects the hypothesis and reference to be detokenized, and by default applies its own internal tokenization before computing BLEU. To turn off any internal tokenization, use:

```
cat [hyp] | sacrebleu --tokenize none [ref]
```

Likewise, by default SacreBLEU computes *cased* BLEU, meaning that casing matters for computing ngram precisions. You can make the tool lowercase the texts as follows:

```
cat [hyp] | sacrebleu -lc [ref]
```

Retokenize differently

Instead of SacreBLEU's internal tokenization, you could use any other tokenization method. Have a look at the `evaluate.sh` script for an example on how to use the Moses tokenizer. Try also what happens when you tokenize the hypothesis and reference differently, which is generally considered very bad practice. You might try the one found in:

```
tools/moses-scripts/scripts/tokenizer/tokenizer_PTB.perl
```

You can also write a simple one yourself or search for another one online. Be sure to mention your choice in the submission.

Investigate how those choices impact BLEU and fill in the table below by postprocessing the files in different ways. Make sure to compare like with like, for instance: if the hypothesis is tokenized, the reference you compare to must also be tokenized (except for the deliberately differently tokenized task of course). You are free to modify the scripts we provided, write new scripts or run the commands directly, but make sure to document your choice accordingly. In

Postprocessing steps (for hyp *and* ref!)	**sacreBLEU settings**	**BLEU**
Full		
Full but lowercased	`-lc`	
Tokenize with Moses tokenizer instead	`--tokenize none`	
Tokenize `hyp` and `ref` differently	`--tokenize none`	

the end, you should fill in BLEU scores in the following table:

Summarize your findings by answering the following questions:

1. Which method achieves the highest BLEU scores? Why?
2. Following up on the last question, what are some flaws in the translation setup we used here? How could it be improved?
3. How might the results have looked like if we used the opposite translation direction? Try to support your thoughts with good reasons.

[5] `https://github.com/mjpost/sacreBLEU`

3 Bonus Task : Automatic vs. Manual Evaluation[6]

As explained in the lecture, it is in some cases more insightful to evaluate translations *manually*. Therefore, in this exercise you will compare several translation systems manually. The exercise folder contains an article from `Die Zeit Online` that was published in German[7] and English[8]. Also, we have included the automatic translations of the article by Google Translate, DeepL and Microsoft Translator (Bing). Work through the following exercises:

1. Use your Python script from Part 1 of this exercise to compute BLEU scores for all translations. Then create a table as follows and insert your calculated scores:

	DE→EN	EN→DE
Google Translate		
DeepL		
Microsoft Translator		

2. For two systems and one translation direction of your choice, have a closer look. Compare their BLEU scores with each other and with the reference translation. Explain which translation is better in your eyes and whether your assessment is reflected in the BLEU scores.
   Pro tip: compare the translations with the interactive BLEU tool from Tilde[9]. To use it you would need to upload the source text, reference translation and the automatic translations.

Submission: A Python script that implements the functions explained above and a PDF document with your answers

In case of problems or if exercises are unclear please post in our OLAT forum.
Good luck!

[6] Based on an exercise created by Mathias Müller. Thanks Mathias!

[7] `https://www.zeit.de/digital/internet/2019-01/datenschutz-katarina-barley-facebook-werbung-personalisierung-mark-zuckerberg`

[8] `https://www.zeit.de/digital/internet/2019-01/privacy-katarina-barley-data-protection-facebook-ad-targeting-mark-zuckerberg`

[9] `https://www.letsmt.eu/Bleu.aspx`