Assignment 5 – Alignment and Translation

CS499 - Intro to NLP
Antonis Anastasopoulos

March 2021

Due Date: 4/23/2021 (eod)
Submission instructions: see at the end of the assignment.
This assignment was created by David Chiang.

Make sure to download the assignment materials from the class page.

Intro

In 2005, a blog post[1] went viral that showed a bootleg copy of Revenge of the Sith with its Chinese version translated (apparently by low-quality machine translation) back into an English movie called Backstroke of the West. Can you do better?

Setup Download the assignment materials from the class page. It contains the following files:

- `train.zh-en`: training data (Chinese-English)
- `train.zh`: training data (Chinese side)
- `train.en`: training data (English side)
- `test.zh`: test data (Chinese side): don't peak!
- `test.en`: test data (English side): don't peak!
- `backstroke.en`: *Backstroke of the West*
- `bleu.py`: evaluation script for translation
- `translate.py`: IBM Model 1 decoder
- `lm.py`: Language Model used by translate.py

The training data is Star Wars Episodes 1, 2, and 4–6. The test data is Episode 3.

You may write code in any language you choose. You may reuse any code you've used in previous homework assignments, or even the solution or another student's code as long as you cite properly.

[1] `https://web.archive.org/web/20170115091456/http://winterson.com/2009/01/episode-iii-backstroke-of-west-redux.html`

1 These are your first steps (2 credits total)

[0.5 credit] (a) Write code to read in `train.zh-en`. It contains one sentence pair per line, with the Chinese and English versions separated by a tab. Both sides have been tokenized. Below, we assume that a fake word NULL is prepended to every English sentence, so that $e_0 = \text{NULL}$.
[1.5 credits] (a) Write code to create the data structure(s) to store the model parameters $t(f \mid e)$ and initialize them all to uniform. Important: You only need a $t(f \mid e)$ for every Chinese word f and English word e that occur in the same line. If f and e never occur in the same line, don't create $t(f \mid e)$. This will save you a lot of memory and time.

2 Join me, and I will complete your training (6 credits total)

[2 credits] (a) Write code to perform the E step. As described in the slides, for each sentence pair and for each Chinese position $j = 1, \ldots, m$ and English position $i = 0, \ldots, l$, do:

$$c(f_j, e_i) \leftarrow c(f_j, e_i) + \frac{t(f_j \mid e_i)}{\Sigma_{i'=0}^{l} t(f_j \mid e_{i'})}.$$

[2 credits] (b) Write code to perform the M step. As described in the slides, for each Chinese word type f and English word type e (including NULL), do:

$$t(f \mid e) \leftarrow \frac{c(f, e)}{\Sigma_{f'} c(f', e)}.$$

[1 credit] (c) Train the model on the training data. Report the total (natural) log-likelihood of the data after each pass through the data.

$$\text{log-likelihood} = \sum_{(\mathbf{f},\mathbf{e}) \text{in data}} \log P(\mathbf{f} \mid \mathbf{e})$$

$$P(\mathbf{f} \mid \mathbf{e}) = \frac{1}{100} \times \prod_{j=1}^{m} \frac{1}{l+1} \left(\sum_{i=0}^{l} t(f_j \mid e_i) \right).$$

It should increase every time and eventually get better than -152000.

[1 credit] (d) After training, for each English word e in {jedi, force, droid, sith, lightsabre}, for each of the five Chinese words f with the highest $t(f\ e)$, report both f and $t(f \mid e)$. The top translations should be: 绝地, 原力, 机器人, 西斯, and 光剑.

3 Now witness the power of this fully operational translation system (2 credits total)

In this part, you'll use the provided Model 1 decoder to try to translate Episode 3 better than Backstroke of the West.

[1 credit] (a) Write code to dump the word-translation probabilities in a text file, named `ttable.txt` (translation table), in the following format:

```
captain 船长 0.7648140965086824
captain 看 2.1415989262858237e-11
captain 我们 2.5532871116116466e-09
captain 搜查 2.1644056596004747e-09
captain 了 1.1592984773140098e-07
NULL 船长 3.7303483172036556e-20
NULL 是 0.04330894875722255
NULL 长官 7.859239306612376e-14
NULL ? 3.5071699071940116e-05
NULL 通知 3.890691080886856e-19
```

The order of the lines does not matter.

Translate Episode 3 (`test.zh`). The decoder should be run (in your terminal/command prompt) like this:

```
translate.py ttable.txt train.en test.zh
```

Translations are written to stdout. In your report, show the output on lines 475–492. Comment on what problems you see, and how they might be fixed.

[1 credit] (b) To evaluate translations accuracy using the BLEU metric, store the translations in a text file and run (in your terminal/command prompt):

```
bleu.py your_translations test.en
```

The score is between 0 and 1 (higher is better). What BLEU score does Backstroke of the West get? What does your system get? You must do better than Backstroke of the West.

4 [BONUS] Neural Grapheme-to-Phoneme Models (3+2 credits total)

In this section we will train a sequence-to-sequence neural model, to produce the pronunciation of French words.

The original data (available under `g2p/data/original`) list the word and the pronunciation. The pronunciation uses the International Phonetic Alphabet. The data are taken from the SIGMORPHON 2020 Shared Task on g2p conversion.[2]

[2] `https://sigmorphon.github.io/sharedtasks/2020/task1/`

The model we will train will receive a single sequence as input, consisting of the word, one character at a time. The target output will be the pronunciation (one IPA character at a time).

[2 credits] We will rely on the Joey-NMT package to implement our models. Follow the instructions here: `https://joeynmt.readthedocs.io/en/latest/index.html`. To install Joey-NMT, create a new python environment, activate it, and then download and install it following the instructions here: `https://joeynmt.readthedocs.io/en/latest/install.html#`

```
git clone https://github.com/joeynmt/joeynmt.git
cd joeynmt
pip3 install .
python3 -m unittest
```

The Joey-NMT tutorial is provides detailed instructions here: `https://joeynmt.readthedocs.io/en/latest/tutorial.html` You are already provided ready (tokenized, lowercased, proper suffix) data in the `g2p/data/converted` directory, so you can skip step 1 of the tutorial.

In step 2, create a copy of the `small.yaml` file into e.g. a `lstm.yaml` file, which we will use to provide the settings for our model. Let's change the yaml file for our purposes:

- In the data section, provide the paths (with proper prefix) to your train, dev, and test data. Keep `{src,trg}_voc_min_freq` to 0 and set the vocab limits to some large number. Keep the segmentation `level` to "word", because we already have whitespace between the characters.
- In the training section, change the epochs to 50 (to start), and set the name of the `model_dir` to a name of your choosing. If you have access to a GPU, you might want to set `use_cuda` to True. Also, change the `eval_metric` to "sequence_accuracy", so that you evaluate the model on whether it produces the correct pronunciation or not (BLEU is not appropriate for our purposes).
- In the model section there's no need to change much at this point. A small model should be able to handle this small dataset. Switch the `rnn_type` of the encoder and the decoder to "lstm" instead of "gru", use 1 encoder layer, and 1 decoder layer.
- In the testing section, set `postprocess` to False – this is only needed when we do machine translation with subword units.

Train the model and plot the learning curves. Evaluate it on the test set, and report its accuracy.

[**1 credit**] Now change the model to a transformer one (using self attention), train, and evaluate it. Does the accuracy improve compared to the LSTM-based model? Feel free to change any other parameters in your model. The best performing model will receive an extra 2 credits – make sure to also submit your system's dev/test outputs.

Submission Instructions

This assignment must be done individually. Discussing the assignment on the class forum is absolutely ok – posting the solutions is not. If you use someone else's code (e.g. taken from a book or a website), make sure to properly cite it in your report.

Please submit all of the following in a gzipped tar archive (.tgz or .tar.gz; not .zip or .rar) via Blackboard. If you're making a full submission, please name your file `gmuid-hw5.tgz` (for example, the instructor's submission would be `antonis-hw5.tgz`). If you're making a partial submission, please name your file `gmuid-hw5-part.tgz` where part is the part (1, 2, or 3) that you're submitting. **Make sure that your .tgz file expands into a self-contained folder with your gmuid in it, e.g. to `antonis-hw5` or something like that.**

Your submission should contain:

- A PDF file (`not .doc or .docx`) with your responses to the instructions/questions above. The report should include your name and gmuid, and it should be self-sufficient (as in, we shouldn't have to look elsewhere to grade you). If your name or an answer does not appear in the report, you will receive 0 credits for that part.
- All of the code that you wrote.
- A brief README file with instructions on how to build/run your code. We **will** run the code and check whether it produces all required outputs and if these outputs match the results in your report.