Computational Linguistics

Assignment 5 (2022-01-04)

Winter Semester 2021/22 – Prof. Dr. Alexander Koller

Word alignments

Aligning sentences and words are central tasks in statistical machine translation (SMT). In this assignment, you get to implement a word aligner. Given pairs of aligned sentences in two languages, source and target, the goal is to align source words to their target translations. The resulting alignments might contain unaligned or multiply aligned words, i.e., the word alignments are generally `m:n`, which makes the task challenging.

This assignment is almost identical to the first assignment of a well-known online course on SMT, which is available here: `http://mt-class.org/jhu/hw1.html`. The task is to train and evaluate IBM Model 1 for word alignments. The EM algorithm for training Model 1 is explained very well in the tutorial by Adam Lopez, which is linked from mt-class and our course website.

Important: For some reason, the Lopez tutorial systematically flips the role of English and Foreign strings, compared to the lecture slides. The text of the mt-class assignment is consistent with this, but the presentation on the lecture slides is consistent with the original publication of IBM Model 1 and a lot of other material, such as Koehn's textbook on SMT. **Please implement the assignment as stated on the mt-class site**, with word translation probabilities $P(f \mid e)$ and the assumption that each Foreign word is aligned to at most one English word.

The key points are as follows:

1. Clone the repository from `https://github.com/xutaima/jhu-mt-hw`, using Git. Observe that the repository contains some Python code and a dataset of 100,000 English-French sentence pairs (`hansards.e` and `hansards.f`) in the folder `hw2`. The first 37 sentence pairs are manually aligned, and these manual alignments are encoded in the file `hansards.a`.

2. Get to know the code of the `aligner`. It provides a very simple baseline system; test it through the provided command-line interface. Determine the Alignment Error Rate (AER) of the baseline system, using the `score-alignments` script, and submit the result. Observe that it is terrible; word alignments are not an easy problem.
3. Your task is to improve over the baseline by implementing an aligner based on IBM Model 1 (see *The Challenge* section of the JHU course for a more detailed description). Your program should learn the parameters $P(f \mid e)$ of Model 1 from the given data, and then use them to compute optimal alignments. Submit the AER for your implementation. Feel free to use NLTK if you find it helpful, but note that anything in the `nltk.align` and `nltk.translate` packages is disallowed in this assignment.
4. In addition to what is required in the JHU assignment, experiment also with an off-the-shelf aligner of your choice. GIZA++ used to be the standard, but is now hard to compile. Depending on your preference of programming language, you might try MGIZA, fast_align, or the Berkeley aligner. Compare your IBM Model 1 to the implementation of Model 1 in the off-the-shelf aligner (if available) and another Model $i > 1$ of your choice.

Extra credit: Implement an aligner that improves over your implementation of IBM Model 1. Some ideas are suggested in the JHU homework assignment.

Submissions: Submit your code and document all your evaluation results. Submit at least one alignment visualization from your system in comparison to the baseline system and the off-the-shelf aligner so we can discuss it in class.

Turn in before class on 2022-01-18 on Google Classroom.