Machine Translation — HW 4

Jeremy Silver

Thurs. 4/9/15

For this assignment, I implemented Och's MERT procedure as outlined in slides 30-38 of the "Tuning" lecture notes. Here is how I computed the line intersection points:

1. For some translation e_i, let v_j be its features, and let w_j be the corresponding weights. Let λ be the designated w_j for which we are doing the line search. Then the feature score is $\lambda a+b$, where $a = v_j$ and $b = \sum_{j' \neq j} w_{j'} v_{j'}$. In this way, each of the translations e_i corresponds to a line with slope a_i and y-intercept b_i.

2. For each sentence, the threshold points for BLEU testing are at the intersection points of the upper envelope of this set of lines. This is because the region between each intersection point must result in the same best-ranked translation for every sentence. We assemble the union of threshold points for all sentences.

   (a) To find the threshold points, we follow the procedure given in the pseudocode, first finding the steepest-descent line. To the left of the leftmost intersection point, this line's feature score is highest. To do this, we first sort the lines by increasing slope. We know that the next intersection must always occur with a line whose slope is bigger, otherwise the first line would fail to be in the upper envelope to the left of this intersection point. So we compute the intersection points with all lines with bigger slope. Given lines $a_1 x + b_1$ and $a_2 x + b_2$, the intersection point is at $x = \frac{b_2 - b_1}{a_1 - a_2}$, assuming $a_1 \neq a_2$ (i.e. parallel lines, which we reject if the y-intercept is smaller). Taking the earliest intersection point, we can go from one line to the next until there are no more intersections.

3. Fixing all weights but λ, we take BLEU scores for λ in each of the threshold intervals, and set λ to be the one that gives the best score.

4. We iterate this over all parameters and repeat that several times until the score increase is below a certain threshold.

As per the suggestions on the website, I added two features to the feature scoring model. The first is simply a count of words in the sentence, and the second is the number of untranslated words in the sentence. The latter I assume is equivalent to the number of words that contain a non-ASCII character. This is likely to be quite accurate because a sizable portion of the Russian alphabet is not ASCII. These features are simple to compute but may be somewhat rudimentary. For instance, we may want to normalize the word count somehow, or express it relative to the length of the original Russian sentence.

Without the added features, I was able to bring the BLEU score from 0.274 (with the default weights) to 0.283 (implemented in `mert.py`). With the added features, I brought the BLEU score up to 0.292 (implemented in `mert2.py`).

One thing I did not try but that may be relevant is starting the optimization procedure with different initial weights. Hopefully the MERT procedure itself will search around enough of the space that it won't make much difference, but the search is local, so it is probably still sensitive to initial conditions.