600.468 Machine Translation **Spring 2015**
Juan Marron

Reranking

1 Simplex Algorithm

As a first step towards improving the baseline reranker, I decided to implement the simplex algorithm. The algorithm works by creating three random weight vectors in our three dimensional space. For each weight, we compute bleu scores over the entire set of sentences and then rank the weights in order from best to worst. Then, we start to converge. I used midpoints between (best, good) and (best, worst) in order to narrow down the triangle to a reasonable level and better search the three dimensional space. I ran this algorithm a couple of times. Sometimes, it was better than baseline, by up to 1.3 BLEU points (I got it to .2865, up from the baseline of .2735). However, other times it was at about .266, which is lower than the baseline. I can attribute these inconsistent results to the fact that the initial vector is random. If I got lucky and picked a good initial vector, then perhaps I would get a better BLEU score. If, however, I picked a vector very far from the optimal vector in the 3 dimensional space, it'd be unlikely to reach a similar point within the number of iterations it took to converge on a value. I saw that about 5-10 iterations of the algorithm led to good scores (and the triangle mostly converged).

2 Word Count

I added a feature to my algorithm that determined the weight that word count would have on the score of a particular hypothesis. I realized that if a sentence was longer, then it likely was a better translation since that's a good indicator that all the components of the source sentence were translated into the target language. I set this weight to be about 1.1. I experimented with adding it to the simplex algorithm and letting it choose what the best weight was, but it did not converge as well as I hoped and actually yielded worse results. Using this feature led to a very good improvement in the BLEU score. My maximum, out of five trials, was .2903 (a substantial improvement over the baseline) with the following weight vector:

$$\theta = [p(e) : -1.03515625, p\_lex(f|e) : -0.55859375, 'p(e|f)' : -0.98828125, 'wordcount' : 1.1]$$

It was also much more consistent than the regular simplex algorithm, which was convenient.

3 Identifying Untranslated Words

I decided to experiment further and identify words that had not been translated. This was easy to do by checking the characters in the hypotheses that were in cyrillic instead of in the english alphabet, since all the words in Russian are written in cyrillic. If there were untranslated words, then the sentence would receive an added penalty. This brought up the score a tiny bit, but an improvement nonetheless. It went from a maximum of .2903 with the word count scoring to a .2924 under the following weight vector:

$$\theta = ['p(e)' : -0.9609375, 'p\_lex(f|e)' : -0.5390625, 'p(e|f)' : -0.921875, 'word\_count' : 1.1]$$

4 "Gaming" the Algorithm

I realized that since the simplex algorithm is so dependent on the starting weights, I could run the algorithm a couple of times, pick the best weight vectors that were generated during those runtimes, and then re-insert them into the algorithm and try again. This is in a sense, what the simplex algorithm is already doing but now the initial weights are not random at all, but instead are more or less guaranteed to result in a better bleu score because we have a general sense of where in the space we should be searching for the optimal weights. Using this system, I changed the initial weights of the first vector (just the first vector because otherwise it does not converge very well since it would already start in a very restricted space). This improved my final BLEU score to .2925, under the following weight vector:

$$\theta = [\prime p(e)\prime : -0.94588, \prime p\_lex(f|e)\prime : -0.53068, \prime p(e|f)\prime : -0.92312, \prime word\_count\prime : 1.1]$$

5 Final Score

My final score, considering the simplex algorithm, word count and penalizing untranslated words, came out to be a maximum of .2925.