Homework 2 - Report

Jim Chen

February 20, 2015

Motivation and Description of Model

For the first part of this project, I implemented the basic stack decoder with reordering and no extra prediction (future costs) or swap penalty. This is the simplest way to extend from the base code. I was able to run the dataset at a stack size of 1000, 2000 and 5000, and with varying cutoffs for probability at 10, 20, 40 choices per word. The decoder extends from the default code by not only considering lengths following the current position, but all possible sub-sections of the source sentence (foreign) that do not overlap with the current decoded bitmap.

Results

This naive implementation of the decoder actually runs slightly below baseline score, but well above the default score. I was able to get a score of -1341.08 for a stack size of 2000 and -1346.67 for a stack size of 5000. Incrementing the stack size does not seem to help much in this case, since we are still pruning a large chunk of possible translations. This default algorithm also gives rise to the problem of accepting high-probability words very early on. There are often cases where even '.' will be aligned incorrectly (not at the end of the sentence). Increasing the word cutoff position did not improve the score much either, with $2000, 40$ giving only -1339.84.

Second Attempt Motivation and Description of Model

For my second attempt, I implemented future cost for the stack decoding which also required relatively few code changes. A step to calculate future cost by dynamic programming was added before stack decoding each sentence. The future cost estimation also takes into account the language model and generates future costs for all possible phrase segments.

Results

Future cost estimation brings significant improvements to the original model, causing it to converge to a good high probability (high score) sentence very quickly. A simple $S = 100$ stack run with default $k = 10$ will get -1295.77, which brings us above the baseline. Enhancing this with more choices and a larger stack ($k = 20, S = 1000$) gives us a huge room for improvement, resulting in a score of -1251.35. Increasing the stack size further gives us even better results, at $S = 5000, k = 20$ we can get a score of -1240.27. Further scaling up the choice set we get -1229.52 for $S = 5000, k = 40$. The benefits of increasing stack size and word considerations diminish even as the sizes increase exponentially (by factors of 2). We run this with parallel processing at 16 processes each 3 sentences under parameters $S = 10000, k = 100$ however this gives us no benefits over the $5000, 40$ model, resulting in nearly exactly the same score.

A final run was done again using parallel processing at 16 processes with 3 sentences each under parameters $S = 2147482647, k = 2147483647$ (max int, first one had a typo). Of this round, only 12 out of 16 groups of sentences have finished and there is no difference between this partial output and the $5000, 40$ case. Arguably the remaning long sentences are very hard to decode and may provide the last of the possible improvements for the theoretical best limit. As of the time of writing this report, the longest sentences have run for 706 minutes (nearly 12 hours). We expect that the gains in probability improvements are not worth the extra runtime and processing power needed.