Homework 2: Decoding

Roger Que

2014–03–06

This submission implements a phrase-based stack decoder in Python. It supports arbitrary reordering of source phrases, with an optional limit set by the user on how many words of the source sentence may be skipped at any given point. Hypotheses are divided into stacks by the number of completed words, and future costs are estimated based on the translation and language model scores of each span [1]. Histogram pruning is used to only extend a constant number of the best hypotheses in each stack. Hypotheses that share the same language model state (i.e., end with the same target language 3-gram) are combined, with the hypothesis with the higher probability taking precedence.

One quirk of the future cost implementation is that it only takes these costs into account for hypothesis recombination, and not at the pruning step. Although this is not justified theoretically, this empirically gave better model scores than when future costs were also taken into account during pruning, as shown in Table 1. This may be explained by the fact that combination of adjacent segment scores in the initial computation of future costs is performed by simply taking their sum. In turn, this causes consistent overestimation of the language model score, as the probabilities of n-grams that overlap the boundary between the two segments are never computed. Only hypotheses with the same terminating LM state compete against each other in recombination, while pruning compares all hypotheses with the same number of completed words, and thus this inaccuracy would more heavily affect the latter process. In any case, both implementations of future cost estimation give decodings with better model score than performing no estimation at all.

Future costs used...	**Log-prob.**
Nowhere	–1341.18
In recombination	**–1289.35**
In recombination & pruning	–1316.51

Table 1: Effects of future cost prediction on corpus log-probability, for number of translations per phrase $k = 100$, stack size limit $s = 200$, and reordering limit $r = 3$.

The effects of the decoder's translation table and stack size limits are summarized in Table 2. As intuitively expected, larger stack sizes always yield equal or better model scores, as fewer hypotheses are pruned. However, increasing the number of translations per phrase gives mixed results. Although there is an obvious benefit to considering 10 translations over just a single one, further bumping the number to 100 actually causes performance to decrease in some cases. This indicates that some low-probability translations lead to hypotheses that appear better for some prefix of a sentence, but end up having a worse score overall.

Changing the reordering limit has results similar to changing the stack size, as shown in Figure 1. Although increasing the limit causes a corresponding increase in model score up to 4 words, further increases have a negative impact as the decoder tends towards simply outputting all of the high-probability segments first.

		Translations per phrase		
		1	**10**	**100**
Stack size	**1**	–1700.33	–1646.97	–1646.91
	25	–1369.91	–1310.66	–1320.42
	50	–1369.91	–1298.37	–1302.51
	100	–1369.91	–1291.52	–1296.66
	200	–1369.91	–1290.17	**–1289.35**

Table 2: Corpus log-probability of decodings given various limits on the number of translations per phrase k and the stack size s, with the reordering limit $r = 3$.

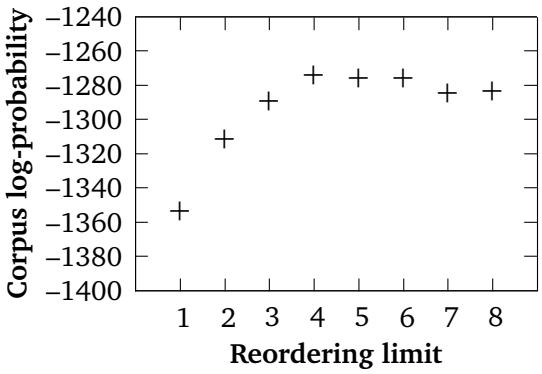

Figure 1: Effects of the reordering limit r on corpus log-probability, with $k = 100$ and $s = 200$.

References

[1] Philipp Koehn. *Statistical machine translation*. Cambridge University Press, 2010.