# Ghoti Word Aligner

## Algorithm outline

The algorithm in `aligner.py` is a "bidirectional" IBM Model 1 with one null word per sentence and *add-n* smoothing (Moore). Here are some details:

First, we start with exactly the same pseudocode as the baseline algorithm described on course webpage, including the easy initialization, EM training, and the decoding. Then we add the following improvements:

1. For each source sentence ($e$), we add a null word. The null word added to every sentence is the same. The initialization and the training process are the same (i.e. the null word is treated just as a normal word). And in the decoding process, if the best alignment of a word in the target sentence ($f$) is a null word, we ignore this target word and go on to the next (i.e. assume this target word has no alignment).

2. When calculating the expected count, we use the so-called "add-$n$" smoothing described in Moore's paper. The modification is minor:

$$t_k(f|e) = \frac{count(f, e) + n}{count(e) + n|V|}$$

where $n$ is a small positive constant and $|V|$ is the size of vocabulary of the target language corpus. The meaning and rationale are explained in details in the paper. The parameter $n$ we chose after experiments is 0.01.

3. "Bi-directional" means that we run the same algorithm (with all the modifications above) twice; the second time we flip the target and the source languages, i.e., to calculate $t(e|f)$. In the decoding phase, for each pair of sentence we get two best alignments, $f$ to $e$ and $e$ to $f$, based on $t(f|e)$ and $t(e|f)$ respectively. Finally, we only output the intersection of the two sets of alignments. This greatly reduces false alignments (at the cost of losing some correct alignments) and improves the AER overall.

## Other experimented approaches

We also tried IBM Model 2 and HMM, but due to limited time because of midterms, we were not able to finish them.

## Acknowledgement


- Anoop Sarkar, the course web page
- Robert C. MOORE, Improving IBM Word-Alignment Model 1