CIS526 Homework 2: Decoding

Joe Trovato

February 21, 2015

Motivation

The default decoder implemnted a stack decoding algorithm with monotone ordering, meaning that all of the words were visited and translated in order. This method produced reasonable results but still only explored a fraction of the search space. This assignment can be reduced to a search problem: our program must search through all possible combinations and orderings of possible tranlatons for each french phrase found in the input sentence. The computations complexity of this problem is exponential with the length of the sentence. For fun, I attempted to try all possible transations and extrapolated the amount of time it would take to complete. The result was approximately 64000 years. With this in mind, the problem became more clear; how could my decoder intelligently prune out unlikely hypothesis and still search the largest portion of possibilities.

Using the textbook, I was able to implement a few ideas that were proven to improve performance, and experiment with a few other ideas. My final algorithm used future cost estiamtion and a reordering limit in addition to the given stack decoder to tranalate sentences. While the output looked reasonable, my results in terms of log probabilities did not imporve much beyond the default code.

Algorithm

My algorithm uses a stack decoder in which stacks are organized by number of foriegn words translated. I search through a variety of reorderings and use a bit vector to keep track of individual hypothesis states. Using a reordering limit and future cost estimation, I was able to get results slightly better than the default code. I eventually pruned on hypothesis according to the sum of their log probability and their future cost. This reduced a variety of errors, cropping up due to nonsensical reordering of probable phrases.

Stack Decoder

My algorithm extends the default stack decoder, whih was modeled by the following equations:
The most probable english sentence:

$$p(e|f) = argmax[p(f|e)p(e)]$$

where the language model given is:

$$p(e) = p(e_1|START) \times \prod_{j=2}^{J} p(e_j|e_{j-1}, e_{j-2}) \times p(END|e_J, e_{J-1})$$

and the likelihood is given by:

$$p(f|e) = \prod_{(i,i';j,j')} p(f_{i,i'}|e_{j,j'})$$

Reodering

My alterations for the most used this model as well. One of the first I troduced was the possibility of multiple orderings when search over all english sentences that maximize $p(f|e)p(e)$. This changes the likelihood model from $p(f|e) = p(f, a|e)p(e)$ (one alignment) to $p(f|e) = \sum_a p(f, a|e)$ (many alignments). Our new liklihood is given by the following equation. Notice that log probabilities are sued to reduce underflow.

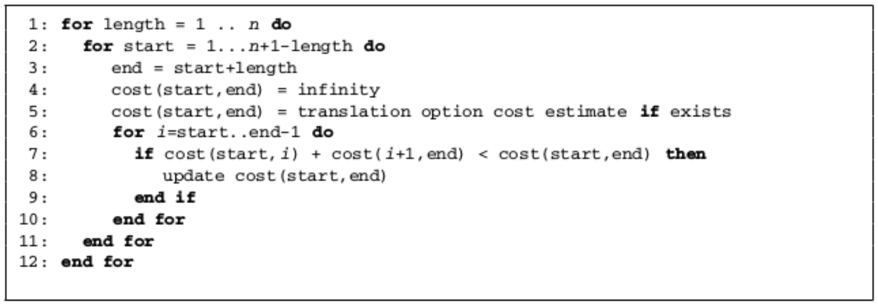
$$argmax(log(p(f, a|e)) + log(p(e)))$$

To acomplish this in code, I wrote a function that searches through all non-translated parts of the foriegn sentence and gives all possible phrases left to translate. This function essentially returns different segmentations of the input sentence. Then these possibilities are encompassed in a loop such that for every hypothesis expanded, it considers all possible phrases left to be translated. To make sure everything was being translated, I implememnted a bit vector to check if all foreign words received an equivalent translation. This in turn, results in non-monotone orderings and all possible segemntations. The final likliehood model is describe below:

$$p(f|e) = p(segmentation) \times p(reordering) \times p(phrasetranslation)$$

Future Costs

After introducing reordering, my output looked like probable phrase in a very random order. To fix this, I introduced future cost into the algorithm. I implemented the algorithm from the book as shown in Figure 1.



```
for length = 1 .. $n$ do
  for start = 1...$n$+1-length do
     end = start+length
     cost(start,end) = infinity
     cost(start,end) = translation option cost estimate if exists
     for $i$=start..end-1 do
        if cost(start,$i$) + cost($i$+1,end) < cost(start,end) then
           update cost(start,end)
        end if
     end for
  end for
end for
```

Figure 1: Future Cost estimation pseudo code from Philipp Khoen test

Ideally, the future cost of translating a sentence out of order would be higher than translating that in order, but I found that this was not true. It did however help in pruning unprobable hypotheses. To fix the reordering issue, I implemented a reordering limit.

The reordering limit immediately threw away sentences that translated word more than 5 words out of order. This was accomplished with a simple filter in the hypothesis generation section of the code.

Experimentation

After implementing all of these stack decoder add-ons, I still did not see very convincing results so I began to experiment with different parameters. First I tried to tune the pruing metric. This was orignally sorting stack hypotheses by log probabilities, but I tried it with $logprob + future_cost$, $logprob + reorder_penalty$, and $logprob + future_cost + reorder_penalty$. These gave slightly different translations but very similar model scores.I was also able to tune the word limit. Witha word limit of 5 I was able to get results slightly better thant he default. I believe this was due to getting the few situtions in which words are flipped in the foreign language as compared to english. making this limi too low gave the same results as the monotone decoder.

Results

Table 1 shows my results as my decoder evolved. One of the difficulties I had with this assignment was determining if my algorithmic changes had an affect on the quality of translations. I am not sure if the log probability based scoring gives the highest score to the best translation. I found that one of my best trails had correctly translated all of the phrases but they were often in the wrong order. A reordering limit fixed this issue but with very marginal gains in score. Similarly, increasing the stack size produced drastically better results (due to directly increasing the percentage of the search space covered), but this was limited by processing power.

It is also interesting to note that incorporating a reordering penalty produced noticable better results in the output sentences, but that this metric was not included in teh final scoring algorithm. I learned that the best translation may not give the best score. his was futher confirmed when one of my classmates manually translated all 48 input sentences and saw poor results. Similarly, before I implemented a bit vector to track traslation state, I ran a reordering algorithm on a large stack size and found that it produced very reasonable translations, but could not align 22 of 48 sentences. I look forward to the evaluation homework assignment so I can get a better understanding of some of these issues.

To summarize my table, I found that minor algorithm changes to the default decoder produced very small if any improvement in score. This could be due to failed implementation, but again, it is very difficult to discern whether your decoder is actually performing better with just the Model Score. For instance, pruning hypotheses by future cost with reordering saw no improvement over pruning by future cost without reordering. This result suggests that reordering is no doing anything or that the most probable translation are those that are already in order. Despite implementing the suggested method by the textbook, my algorithm could make siginifcant improvement over the default decoder.

Table 1: Model Score results with different decoders and stack size limits

Decoder	Stack Size	Approximate Model Score
default stack decoder	1	-1445
default stack decoder	10000	-1353
stack decoder, reordering without limit	1	-1750
stack decoder, reordering with penalty	10000	-1455
stack decoder, future cost	10000	-1353
stack decoder, reordering without limit, future cost	10000	-1353
stack decoder, reordering with limit, future cost	20000	-1340