Homework #1
Machine Translation
600.468
Spring 2015

Sumit Pawar - (spawar3@jhu.edu)

February 17, 2015

1 IBM Model 1

Initially we try to implement the basic IBM Model 1 for alignment of the given French word corpus against the generated English words. IBM Model 1 is a probabilistic model that generates each word of the foreign sentence 'f' independently, conditioned on some word in the English sentence. We will use expectation maximization (EM) to align the text.

We can define the alignment as :

$$p(a|e,f) = \frac{p(e,a|f)}{p(e|f)}$$

We factorize over the alignment a:

$$p(e|f) = \sum_a p(e,a|f)$$

$$= \frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} \sum_{j=0}^{l_f} t(e_j|f_i)$$

For the **E-STEP** we have ($Koehn4.2.3$):

$$p(a|e,f) = \frac{\frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})}{\frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} \sum_{i=0}^{l_f} t(e_j|f_i)}$$

$$= \prod_{j=1}^{l_e} \frac{t(e_j|f_{a(j)})}{\sum_{i=0}^{l_f} t(e_j|f_i)}$$

We have collected counts for the word translation over all possible alignments, weighted by probability. Let c be the count function for the pair(e,f), that a particular word f translates to the word e:

$$c(e|f;\textbf{e,f}) = \sum_a p(a|\textbf{e,F}) \sum_{j=1}^{l_e} \delta(e,e_j)\delta(f,f_{a(j)})$$

Where the Kronecker delta function $\delta(x,y)$ is 1 if $x == y$ and 0 otherwise. On substituting we get:

$$c(e|f;\textbf{e,f}) = \frac{t(e|f)}{\sum_{i=0}^{l_f} t(e|f_l)} \sum_{j=1}^{l_f} \delta(e,e_j) \sum_{i=0}^{l_f} \delta(f,f_l)$$

Therefore for the **M-STEP** we have ($Koehn4.2.3$):

$$t(e\perp; \textbf{e,f}) = \frac{\sum_{\textbf{(e,f)}} c(e|f; \textbf{e,f})}{\sum_e \sum_{\textbf{(e,f)}} c(e|f; \textbf{e,f})}$$

The code is in JAVA. Here is the pseudo code for the basic initial model:

Algorithm 1 BASIC-EM_IBM_MODEL_1

```
initialize t(e—f) uniformly
while repeat until convergence do
    // initialize
    count(e|f) = 0 for all e,f
    total(f) = 0 for all f
    for each sentence pair (e,f) do
        // compute normalization
        for each word e in e do
            s-total(e) = 0
            for each word f in f do
                s − total(e) += t(e|f)
            end for
        end for
        // Collect counts
        for each words e in e do
            for each word f in f do
                count(e|f) += t(e|f) / (s−total(e))
                total(f) += t(e|f) / (s−total(e))
            end for
        end for
    end for
    // estimate probabilities
    for each foreign word f in f do
        for each english word e in e do
            t(e|f) = count(e|f) / total(f)
        end for
    end for
end while
```

On running this iteratively and applying to the corpus of French text we get a decent alignment of English text with a total score (error) of around 0.40. See IBMModel__1.java for details

2 IBM Model 1 with Word Frequency Estimates

This is very similar to the above model but instead of uniformly initializing the initial parameters, we derive these from the model and then run EM. Therefore the initialization function will look as follows:

Algorithm 2 INITIALIZEFROMDATA()

1: **for** each word pair (e,f) **do**
2: $count(e, f) += 1$
3: $count(e) += 1$
4: $count(f) += 1$
5: **end for**

3 IBM Model 1 with NULL alignment

We can re-consider the above model and think about NULL alignments mode mathematically. From the given corpus if we want to determine the average amount of null alignments and fit that into our model we can make the following mathematical considerations:

if($l_e > l_f$) // if length(English e) $>$ length(French f)

$$p\_null += \frac{(l_e - l_f)}{l_e}$$

else 0.

This is the estimate that for a given pair of English and French sentences, if the English sentence is longer than the French sentence, then some of the English words will have NULL French alignment.

Alternatively, we can initially assign some random values to p_null and then fine-tune it depending on the results observed.

Now, the new **E-Step** formulas change as follows:

$$p(I = e_i|len_f, len_e, f_j) = \frac{(1 - pNULL) \cdot t(f_j|e_i)}{pNULL \cdot t(f_j|NULL) \quad + \quad (1 - pNULL) \sum_{e_{i'}} t(f_j|e_{i'})}$$

And,

$$p(I = NULL\_POS|len_f, len_e, j) = \frac{pNULL \cdot t(f_j|NULL\_POS)}{pNULL \cdot t(f_j|NULL) \quad + \quad (1 - pNULL) \sum_{e_{i'}} t(f|e_{i'})}$$

Consecutively the **M-Step** becomes:

$$t(f|e) = \frac{C(e, f)}{\sum_{f'} C(e, f')}$$

Or,

$$t(e|f) = \frac{C(f, e)}{\sum_{e'} C(f, e')}$$

We can simply plug in these formulations in place of the original one's in the above algorithm.

This eventually gives us a sufficient gain and score (error) of 0.32 on the test data, which is above the required baseline.

4 IBM Model 1 with bucket considerations for punctuations and transition to IBM Model 2

There are still errors and considerable scope for improvement. Therefore, let's try transitioning into IBM Model 2. This requires a change to $a(a_j|j, len_f, len_e)$ which will make a_j no longer independent of j. Here we create buckets and define probabilities for the buckets. These will also include normalization.

$$a(a_j|j, len_f, len_e) = p(bucket(\delta(a_j, len_e, len_f))) = p(bucket(a_j - j \cdot \frac{len_e}{len_f}))$$

Substituting this change in the existing model, we can achieve a slightly better score (error) of 0.31 on the test data.