Assignment 5: Machine Translation

1 IBM Model 1

1.1 Description of IBM Model 1

The IBM models are an instance of a noisy-channel model. IBM Model 1 is a word alignment model, which uses an Expectation Maximization Algorithm to compute the lexical translation probabilities in parallel text.

Let $e_1 ... e_l$ and $f_1 ... f_m$ be the English sentence and foreign sentence we are doing alignment. Let l and m be their length respectively. we have latent variable $a = a_1, \ldots, a_m$, each a_i ranging over $0, \ldots, l$ to represent the alignment. Then the probability of sentence e with length m To be tranlated to f using alignment a is:

$$p(f_1 ... f_m, a_1 ... a_m | e_1 ... e_l, m) = (\frac{1}{l+1})^m \prod_{i=1}^{m} t(f_i | e_{a_i})$$

$t(f_i | e_{a_i})$ can be interpreted as the conditional probability of generating a foreign word f from an English word e (or from NULL).

The **limitations** of IBM model 1 is that it is weak to reordering. We can see that the above probability is calculated only using $t(f|e)$, therefore it is possible that there are two alignments with the same $t(f|e)$. Also this model does not take the relative postions of words into consideration.

1.2 Description of EM Algorithm

In this section, we use the Expectation Maximization Algorithm(EM Algorithm) to estimate each $t(f|e)$. The EM Algorithm iterates through the parallel corpus repeatedly. For the k-th sentence pair and each index i in the foreign sentence $f^{(k)}$ and j in the English sentence $e^{(k)}$, let l_k be the length of the k-th English sentence, we define:

$$\delta(k, i, j) = \frac{t(f_i^{(k)} | e_j^{(k)})}{\sum_{j=0}^{l_k} t(f_i^{(k)} | e_j^{(k)})}$$

Then we use the δ value to update the expected counts for the current iteration:

$$c(e_j^{(k)}, f_i^{(k)}) = c(e_j^{(k)}, f_i^{(k)}) + \delta(k, i, j)$$
$$c(e_j^{(k)}) = c(e_j^{(k)}) + \delta(k, i, j)$$

Then at the end of each iteration, we update each $t(f|e)$:

$$t(f|e) = \frac{c(e, f)}{c(e)}$$

For the **strengths** of EM algorithm, it is iterative and at each iteration we compute the counts based on the data and current parameter estimates. For the **weaknesses** of EM algorithm, it will converge to a local maximum of log-likelihood function, so sometime we cannot obtain the global maximum. Also as this is an iterative algorithm, the running time will be too long.

1.3 Method Overview

In my implementation, I set $t(f|e)$ to the uniform distribution over all foreign words that could be aligned to e in the corpus, which is $\frac{1}{n(e)}$. Then I make the EM algorithm to iterate 5 times and use the `pickle` package in python to save the t Parameters for IBM model 2. Then the program will use the t Parameters to find the best alignment with the highest $t(f|e)$ score for each foreign word f_i over all english words in the sentence:

$$a_i = arg\,max_{j \in 0 \ldots l} t(f_i|e_j)$$

Then the program will save the result to dev.out, and I use `eval_alignments.py` to estimate it.

1.4 Results

Type	Total	Precision	Recall	F1-Score
Total	5920	0.413	0.427	0.420

1.5 Discussion

Iterations	1	2	3	4	5
F1-Score	0.214	0.380	0.408	0.416	0.420

We can see that as the iterations become bigger, the F1-Score become larger. However, as the number of iterations increases, the increase in the score decreases, this is because the model parameters are converging to the local optimum.

2 IBM Model 2

2.1 Description of IBM Model 2

The difference between IBM Model 2 and Model 1 is that in Model 2 we introduce alignment parameters $q(i|j, l, m)$, which is the probability that that jth French word is connected to ith English word, given sentence lengths of e and f are l and m respectively. Then probability of sentence e with length m To be tranlated to f using alignment a is:

$$p(f_1 \ldots f_m, a_1 \ldots a_m | e_1 \ldots e_l, m) = \prod_{i=1}^{m} q(a_j|j, l, m) t(f_i|e_{a_i})$$

Then we show how to compute δ in EM algorithm:

$$\delta(k, i, j) = \frac{q(j|j, l_k, m_k) t(f_i^{(k)}|e_j^{(k)})}{\sum_{j=0}^{l_k} q(j|j, l_k, m_k) t(f_i^{(k)}|e_j^{(k)})}$$

Also we need to update $q(j|i,l,m)$ at the end of each iteration:

$$q(j|i,l,m) = \frac{c(j|i,l,m)}{c(i,l,m)}$$

IBM Model 2 is better than IBM Model 1 because it add the alignment parameters $q(i|j,l,m)$, which can represent the probability that that jth French word is connected to ith English word, given sentence lengths of e and f are l and m respectively, thus it solve the weak reordering problem in Model 2.

For the **limitations** of IBM Model 2, there are still some noisy alignments which can influence the result. Also IBM Model 2 still assume that the alignments are many to one, which is not realistic.

2.2 Method Overview

In my implementation, I first load the t parameters computed from IBM Model 1, then initialize the q paramters to $\frac{1}{l+1}$, which is the uniform distribution over all j for each i, l, and m. Then I run 5 iteration for the program, and the program will use the t Parameters and q parameters to find the best alignment with the highest score for each foreign word f_i over all english words in the sentence:

$$a_i = arg\, max_{j\in 0\ldots l} q(j|i,l,m)t(f_i|e_j)$$

Then the program will save the result to dev.out, and I use `eval_alignments.py` to estimate it.

2.3 Results

Type	Total	Precision	Recall	F1-Score
Total	5920	0.442	0.456	0.449

2.4 Discussion

Iterations	1	2	3	4	5
F1-Score	0.439	0.442	0.446	0.446	0.449

We can see that as the iterations become bigger, the F1-Score become larger. We at lower iteration, IBM Model 2 can get a better result than IBM Model 1, thus we can conclude that the IBM Model 2 converges rapidly.

A correctly aligned example using IBM Model 2 is shown below:

	espero	que	el	nuevo	parlamento	y	la	nueva	comisión	puedan	hacer	uso	de	ellas
i														
hope	●	●												
that														
the			●											
new				●										
parliament					●									
and						●								
the							●							
new								●						
comission									●					
will														
make											●			
use										●		●		
of													●	
Them														●

A misaligned example is shown below:

	creo	que	a	la	larga	debemos	llegar	a	ello
i									
believe	●								
we			●	●					
should		●				●			
achieve					●		●	●	●
this									
in									
time									

From the above two examples we can see that IBM Model is good at solving the tranlation problem when the meaning and structure of the sentence is simple. From the examples, we can see that the model can handle many-to-one alignment very well. I think the reason why the model misaligned in the second example is that IBM Model 2 still assume that the alignments are many to one, which is not realistic.