CSE256: Statistical NLP Assignment 4

Pranav Verma

November 27, 2019

1 IBM model 1

1.1 Description of IBM model 1

IBM model 1 was designed as an extension of the noisy channel model for translation. A noisy channel model has two components: $p(e)$, the language model, which models the source language; and $p(f|e)$, the translation model. The IBM model 1 models the translation aspect of the noisy channel model. It models $p(f_1...f_m|e_1...e_l)$, by finding the best possible alignments for each word in the foreign language sentence, $f_i^{(k)}$. It gets $p(f_1, ..., f_m|e_1, ..., e_l)$ by computing the maximum likelihood estimates of probabilities $t(f_i|e_j)$, which is then used to find alignments $a_1, ..., a_m$ which maximize the probability $p(f_1, ..., f_m, a_1, ..., a_m|e_1, ..., e_l)$.
IBM Model 1 has some severe limitations, chief of which is that each word in the source sentence can only translate to one word in the target sentence. This is not the case in real world, and thus this model is severely limited. Also, the position of the target word in target language is independent of the location of source word in the source sentence, and also independent of the positions of other words in the sentences. It does not consider the fact that for some translations, words which appear after the other in the source translations will also appear after each other in the target sentence. It will treat this language pair translation the same as it would treat language pairs in which the words appear in reverse order in the target sentence.

1.2 Description of EM algorithm

The Expectation Maximization (EM) algorithm is used to find out the maximum likelihood estimates of the unobserved data ($t(f_i|e_j)$) using an iterative procedure. In the expectation step, it calculates the likelihood of finding i^{th} foreign language word and j^{th} language word together using the current estimates of $t(f_i|e_j)$ over the whole model, and then uses these estimates to compute the parameter $t(f_i|e_j)$ which would maximize the maximum likelihood value computed in the E step.
The EM algorithm is a good way to estimate the latent variables values. However, there is a possibility that the parameters found are not the globally optimum parameters (i.e. better values of these parameters may exist). The performance is dependent on the starting point of the algorithm, and is therefore not guarenteed to yield the best results.

1.3 Method Overview

To implement IBM model 1, we stored the values $t(f|e)$ in a dictionary, for each English word e and Spanish word f in the corresponding sentences in the English and Spanish corpora. Similarly, the values $c(e, f), c(e), c(j|i, l_k, m_k), c(i, l_k, m_k)$ were all stored in dictionaries, with tuples of e, f, i, j, l_k, m_k as keys. This allowed us to store the parameters as sparsely as possible, saving a lot of memory.
$t(f|e)$ was initialised with $\frac{1}{n(e)+1}$ where $n(e)$ was the frequency count of the word e in the English corpus.
Then, the EM algorithm was a straight-forward procedure, where we started by setting $c = 0$ for all keys. Then we incremented the counts of the c dictionaries by $\delta(k, i, j)$, which was calculated using our current estimates of $t(f|e)$, for all e,f in all the sentence pairs in the English and Spanish corpora. Then, using these counts c, we updated the values of $t(f|e)$ for all English words e and Spanish words f in the sentence pairs of the respective corpora. Repeating the above procedure 5 times yielded a good approximation of the value $t(f|e)$
To get the alignments for i^{th} Spanish word f_i^k in k^{th} sentence, we simply found the location of j^{th} English word e_j^k in the corresponding k^{th} English sentence which yielded the maximum value of $t(f|e)$. The alignment was finally represented as $[k \quad j \quad i]$.

1.4 Results

By performing 5 iteration of the EM algorithm for the IBM model 1, as described above, we get the following values:

Type	Total	Precision	Recall	F1-Score
total	5920	0.412	0.426	0.419

Table 1: Performance of IBM model 1 on dev data

1.5 Discussion

As a function of iterations, we see that the model iteratively learns optimal alignments. This is reflected in the increasing F1 scores:

Iteration	Type	Total	Precision	Recall	F1-Score
1	total	5920	0.222	0.230	0.226
2	total	5920	0.369	0.381	0.375
3	total	5920	0.401	0.414	0.407
4	total	5920	0.408	0.422	0.415
5	total	5920	0.412	0.426	0.419

Table 2: Performance of IBM model 1 on dev data as a function of number of iterations of the EM algorithm

2 IBM model 2

2.1 Description of IBM model 2

IBM model 2 was designed to be an improvement over model 1. It models the $p(f|e)$ using two parameters, $q(j|i, l, m)$ and $t(f|e)$. The first parameter describes the likelihood that the j^{th} word in source sentence of length l aligns to the i^{th} word in the target sentence of length m. This will allow us, for example, to capture the tendency for words close to the beginning of the foreign sentence to be translations of words close to the beginning of the source language sentence.
However, there's one key limitation in translation that even IBM model 2 fails to address: the source word can translate to zero or one or more than one words in the foreign language. It does not model these tendencies in its translation model at all.

2.2 Method Overview

The IBM model 2 was very similar to the IBM model 1 described above. We started with $t(f|e)$ as estimated from 5 iterations of EM algorithm in IBM model 1. We also kept track of another parameter $q(j|i, l_k, m_k)$, using a dictionary, with the tuple (i, j, l_k, m_k) as keys for the dictionary. We initialized it using $q(j\Big|i, l_k, m_k) = \frac{1}{l_k+1}$, where l_k was the length of k^{th} English sentence.
Then, as we did in the case of IBM model 1, we initialized $c = 0$ for all keys, and for all words in each sentence in the corpora, updated it using

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

using the current estimates of $t(f_i^{(k)}\Big|e_j^{(k)})$ and $q(j\Big|i, l_k, m_k)$ for each Englis word e_j^k, and spanish word f_i^k in k^{th} sentence pairs in the corpora.
Using these updated estimates of c, we updated our estimates of q and t, using:

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

and

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

5 iterations of this EM algorithm were run to get estimates of t and q. Using these estimates, we predicted the alignments for i^{th} in k^{th} sentence by finding the j^{th} word such that the value of $q(j|i, l, m)t(f_i|e_j)$ was maximum. The alignment was then recorded as the tuple $[k \quad j \quad i]$

2.3 Results

Using the algorithm described above, and running for five iterations, we get the following results: We note that the final F1 score after 5 iterations was 0.449, a clear improvement over IBM model 1.

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

Table 3: Performance of IBM model 2 on the dev data set after 5 iterations of EM algorithm

2.4 Discussion

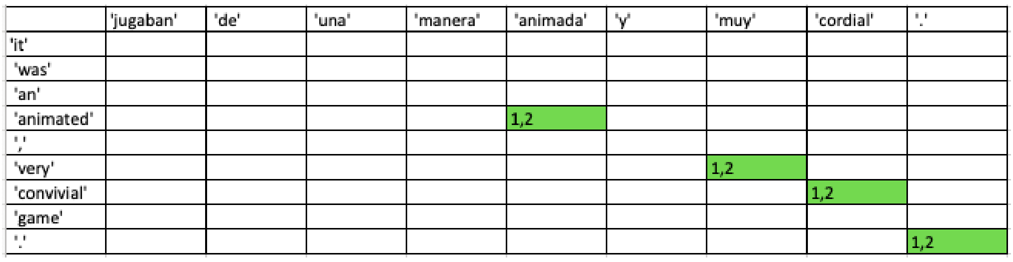
We present below the performance of our model at the end of each iteration. We see that the model generally performs better each iteration, as it updates the t and q parameters iteratively on the whole dataset.

Iteration	Type	Total	Precision	Recall	F1-Score
1	total	5920	0.431	0.445	0.438
2	total	5920	0.436	0.451	0.443
3	total	5920	0.439	0.454	0.446
4	total	5920	0.438	0.453	0.445
5	total	5920	0.442	0.456	0.449

Table 4: Performance of IBM model 2 on the dev data set as a function of number of iterations of EM algorithm run

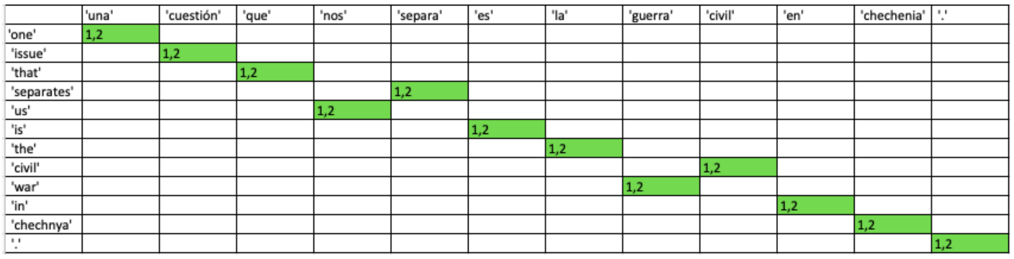
2.4.1 Aligned and Misaligned sentences

We present the alignment matrices for aligned and misaligned cases. Green cells highlight alignments predicted by both the gold standard and our model. Gold cells represent alignments from the gold standard missed by our model, and red cells represent incorrect alignments predicted by our model.



	'jugaban'	'de'	'una'	'manera'	'animada'	'y'	'muy'	'cordial'	'.'
'it'									
'was'									
'an'									
'animated'					1,2				
','									
'very'							1,2		
'convivial'								1,2	
'game'									
'.'									1,2

(a) Sentence 1



	'una'	'cuestión'	'que'	'nos'	'separa'	'es'	'la'	'guerra'	'civil'	'en'	'chechenia'	'.'
'one'	1,2											
'issue'		1,2										
'that'			1,2									
'separates'					1,2							
'us'				1,2								
'is'						1,2						
'the'							1,2					
'civil'									1,2			
'war'								1,2				
'in'										1,2		
'chechnya'											1,2	
'.'												1,2

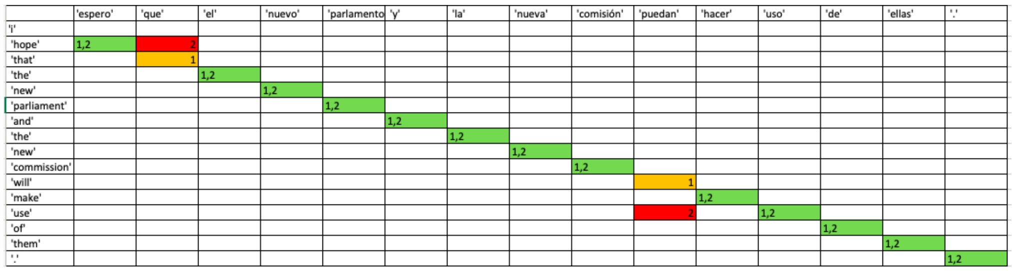
(b) Sentence 2

Figure 1: Correctly Aligned sentences

3 Growing Alignments

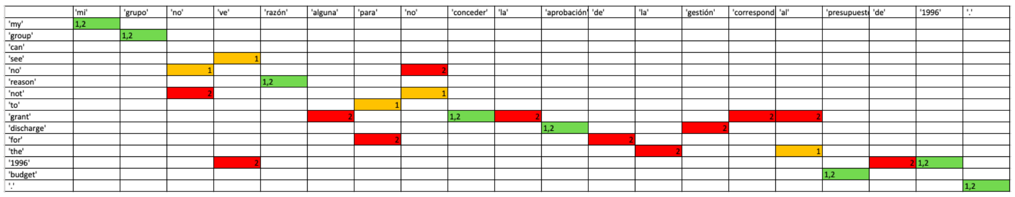
3.1 Method Overview

We started with the alignments which we got from the IBM model 2 described above. Then we found the alignments by going in the reverse direction: English to foreign. Then, we took the alignments which were predicted by both these methods (calling these the intersection of the two alignment sets), and all the alignments by any of these two methods (calling these the union of the two alignment sets). We start from the intersection set, and iteratively add the alignments from the union set which are not in the intersection set. We do this by generating the alignment matrix for each sentence. We put alignments in the intersection as 1, and alignments in union as -1.
Our heuristic is to add neighbouring alignments, i.e. if (i, j) is not currently in the alignments, but one of its neighbours are,



	'espero'	'que'	'el'	'nuevo'	'parlamento	'y'	'la'	'nueva'	'comisión'	'puedan'	'hacer'	'uso'	'de'	'ellas'	'.'
'i'															
'hope'	1,2	2													
'that'		1													
'the'			1,2												
'new'				1,2											
'parliament'					1,2										
'and'						1,2									
'the'							1,2								
'new'								1,2							
'commission'									1,2						
'will'										1					
'make'											1,2				
'use'										2		1,2			
'of'													1,2		
'them'														1,2	
'.'															1,2

(a) Sentence 1



	'mi'	'grupo'	'no'	've'	'razón'	'alguna'	'para'	'no'	'conceder'	'la'	'aprobación'	'de'	'la'	'gestión'	'correspond	'al'	'presupuest	'de'	'1996'	'.'
'my'	1,2																			
'group'		1,2																		
'can'																				
'see'				1																
'no'			1					2												
'reason'					1,2															
'not'			2					1												
'to'							1													
'grant'						2			1,2	2					2	2				
'discharge'											1,2			2						
'for'							2					2								
'the'													2			1				
'1996'				2														2	1,2	
'budget'																	1,2			
'.'																				1,2

(b) Sentence 2

Figure 2: Incorrectly Aligned sentences

then we add (i, j) into the alignment set. In the alignment matrix, it is equivalent to turing all -1's which have 1 as their immediate neighbours into +1's, and adding those alignments to the intersection set.
We do this procedure over all sentences, and repeat the whole process till we have no more alignments to add this way. We present our results below.

3.2 Results

The intersection alignment set had the following scores:

Type	Total	Precision	Recall	F1-Score
total	5920	0.775	0.352	0.485

Table 5: Performance of Intersection alignments

The union alignment set had the following scores:

Type	Total	Precision	Recall	F1-Score
total	5920	0.363	0.565	0.442

Table 6: Performance of Union alignments

We note that intersection alignment set has a high precision score, while union set has a high recall score. Combining the alignments from these two allow us to get alignments which balance the precision and recall, and achieve an overall higher F1 score than both:
We note that as we add more and more of alignments, we are able to achieve a better F1 score.

3.3 Another heuristic towards growing alignments

We tried another heuristic to grow the alignments, similar to the one described in section 4.5 of https://www.aclweb.org/anthology/N03 1017.pdf. For each sentence, we built the alignment matrix, and we looked at words which were unaligned in the intersection set, and added alignments from the union set if they existed. We did this for each sentence in the sentence pairs of the corpus. After one complete pass over the entire corpus, we evaluated the model's performance:
We see that our model performs better than IBM model 2, and improves over the intersection alignment set.

Iteration	Type	Total	Precision	Recall	F1-Score
1	total	5920	0.673	0.453	0.541
2	total	5920	0.662	0.464	0.545
3	total	5920	0.659	0.466	0.546
4	total	5920	0.658	0.467	0.547
5	total	5920	0.658	0.468	0.547
6	total	5920	0.658	0.468	0.547

Table 7: Growing Alignments: Performance as we add more neigbouring alignments from union of alignments to intersection, over multiple iterations

Type	Total	Precision	Recall	F1-Score
total	5920	0.558	0.440	0.492

Table 8: Performance using the new heuristic, after one complete pass over the entire corpus