Programming Assignment 4: Machine Translation

Shihan Ran, A53313589
Fall 2019, CSE 256: Statistical NLP

I. IBM Model 1

A. Description of IBM Model 1

The IBM Models are an instance of a noisy-channel model, and they have two components:

- *A Language Model* that assigns a probability $p(e)$ for any sentence $e = e_1 \dots e_l$ in English. We can use any language models we've learned before, for example, a trigram model. The parameters of the language model can potentially be estimated from very large quantities of English data.
- *A Translation Model* that assigns a conditional probability $p(s|e)$ to any Spanish/English pair of sentences. The parameters of this model will be estimated from the translation examples.

*1) **What are IBM Models used for?**:* Our goal is to model the conditional probability $p(s|e) = p(s_1, \dots, s_m|e_1, \dots, e_l, m)$ where $s_1, \dots, s_m$ is the foreign sentence and $e_1, \dots, e_l$ is the English sentence. The IBM Models make direct use of the idea of *alignments*, and the resulting alignment models are of central importance in modern Machine Translation systems. The parameters of the IBM Models will be estimated using the expectation maximization (EM) algorithm.

The Models define a conditional distribution

$$p(s|e) = p(s_1, \dots, s_m, a_1, \dots, a_m|e_1, \dots, e_l, m)$$

where $a_1, \dots, a_m$ is the alignment of foreign sentence with words $s_1, \dots, s_m$. IBM Model 1 uses only translation parameters $t(s|e)$, which are interpreted as the conditional probability of generating Spanish word s from English word e.

*2) **Limitations of IBM Model 1**:* Each Spanish word is aligned to exactly one English word, which means the alignment is many-to-one. Some English words may be aligned to zero Spanish words. From our previous definition, IBM Model 1 only uses translation parameters $t(s|e)$, which results in the limited knowledge of the model for other information such as length of Spanish and English sentences, relative positions of Spanish words and English words.

B. Description of EM Algorithm

The estimates for fully-observed data are simple to derive. However, sometimes we will need to find parameters under many circumstances that data are incomplete. The EM algorithm is an efficient iterative method to calculate the maximum likelihood estimate when some of the data are missing or hidden.

*1) **Pros**:* The EM algorithm is iterative and always improves a parameter's estimation through its process. It could be applied even when part of the data are incomplete. It is able to guess and estimate a set of parameters for your model under many situations.

*2) **Cons**:* We begin with some random initial values for the parameters. Hence, the algorithm may end up stuck in a local maximum instead of the optimal global maximum. Also, the EM algorithm can be very slow sometimes.

C. Method Overview

*1) **High-Level Description of Implementation**:* My implementation of IBM Model 1 includes two parts: training and testing. The training part is done by running 5 iterations of EM Algorithm. More details can be found in Algorithm 1. The testing part includes reading parameters as well as testing corpora and assigning alignment to each sentences pair with the highest $t(s|e)$ score, i.e. $a_i = \underset{j \in 0 \dots l}{\arg\max} t\,(s_i|e_j)$.

D. Results

The result of my implementation matches the expected F1-Score. Details can be found in Table

Algorithm 1: The parameter estimation algorithm for IBM Model 1 for partially-observed data

Input: A training corpus $(s^{(k)}, e^{(k)})$ for $k = 1 \dots n$, where $s^{(k)} = s_1^{(k)} \dots s_{m_k}^{(k)}$, $e^{(k)} = e_1^{(k)} \dots e_{l_k}^{(k)}$. An integer $N = 5$ specifying the number of iterations of training.

Initialization $t(s|e) = \frac{1}{n(e)}$, where $n(e)$ is defined as the number of different words that occur in any translation of a sentence containing e;

for $step = 1$ to N **do**

  Set all counts $c(\dots) = 0$;

  for $k = 1$ to n **do**

    for $i = 1$ to m_k **do**

      for $j = 0$ to l_k **do**

        $c\left(e_j^{(k)}, s_i^{(k)}\right) \leftarrow c\left(e_j^{(k)}, s_i^{(k)}\right) + \delta(k,i,j),\ c\left(e_j^{(k)}\right) \leftarrow c\left(e_j^{(k)}\right) + \delta(k,i,j)$;

        $c\left(j|i, l_k, m_k\right) \leftarrow c\left(j|i, l_k, m_k\right) + \delta(k,i,j),\ c\left(i, l_k, m_k\right) \leftarrow c\left(i, l_k, m_k\right) + \delta(k,i,j)$;

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

  Set $t(s|e) = \frac{c(e,s)}{c(e)}$;

Output: parameters $t(s|e)$.

TABLE I: Statistical Information About Corpus

	Training Corpus	Dev Corpus	Dev Key
Total	5401	200	5921

TABLE II: Performances of IBM Model 1

	Precision	Recall	F1-Score
Total	0.413	0.427	0.420

I and II.

E. *Discussions*

The performances through the training process are given by the Table III. We can see from the table the following facts:

- The F1-Score grows gradually as the iteration increases. This reasonable according to our previous analysis since the EM algorithm always improves a parameter's estimation through its process.
- However, there is an interesting phenomenon that the growth rate is mostly always decreasing as the iteration goes. From iteration 1 to iteration 2 the F1-Score increased by 66%, whereas from iteration 4 to iteration 5 it only increased by 0.72%.

TABLE III: Performances of IBM Model 1

Iterations	Precision	Recall	F1-Score	Growth Rate
1	0.222	0.230	0.226	-
2	0.370	0.382	0.376	**66.4%**
3	0.402	0.415	0.408	8.5%
4	0.410	0.424	0.417	2.2%
5	0.413	0.427	0.420	0.72%
6	0.418	0.431	0.425	1.2%
7	0.420	0.434	0.427	0.47%
8	0.422	0.436	0.429	0.46%
9	0.422	0.436	0.429	0%
10	**0.424**	**0.438**	**0.431**	0.46%

II. IBM Model 2

A. *Description of IBM Model 2*

*1) **Comparison with IBM Model 1**:* The IBM Model 2 is very similar to IBM Model 1 except for it extends the implementation of the EM algorithm and has a set of new parameters. The main additional step is adapting the delta function to include $q(j|i,l,m)$ which represents the probability that j-th Spanish word is connected to the i-th English word given sentence length of e and s are 1 and m respectively.

IBM Model 2 outperforms IBM Model 1 because of the alignment parameter q. It can model the translation of a source sentence word in position i to a target language sentence word in posi-

Algorithm 2: The parameter estimation algorithm for IBM Model 2 for partially-observed data

Input: A training corpus $(s^{(k)}, e^{(k)})$ for $k = 1 \dots n$, where $s^{(k)} = s_1^{(k)} \dots s_{m_k}^{(k)}$, $e^{(k)} = e_1^{(k)} \dots e_{l_k}^{(k)}$. An integer $N = 5$ specifying the number of iterations of training.

Initialization $t(s|e)$ as the last set of parameters (after 5 iterations) produced by IBM Model 1;

Initialization $q(j|i,l,m) = \frac{1}{l+1}$;

for $step = 1$ to N **do**

  Set all counts $c(\dots) = 0$;

  for $k = 1$ to n **do**

    for $i = 1$ to m_k **do**

      for $j = 0$ to l_k **do**

        $c\left(e_j^{(k)}, s_i^{(k)}\right) \leftarrow c\left(e_j^{(k)}, s_i^{(k)}\right) + \delta(k,i,j),\ c\left(e_j^{(k)}\right) \leftarrow c\left(e_j^{(k)}\right) + \delta(k,i,j)$;

        $c\left(j|i,l_k,m_k\right) \leftarrow c\left(j|i,l_k,m_k\right) + \delta(k,i,j),\ c\left(i,l_k,m_k\right) \leftarrow c\left(i,l_k,m_k\right) + \delta(k,i,j)$;

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

  Set $t(s|e) = \frac{c(e,s)}{c(e)}$, $q(j|i,l,m) = \frac{c(j|i,l,m)}{c(i,l,m)}$;

Output: parameters $t(s|e)$.

tion j using the alignment probability distribution.

*2) **Limitations of IBM Model 2**:* IBM Model 2 still uses only word-to-word translations between languages where as in reality many translations could only be achieved if group of words (phrase) are translated together. Also it sometimes maps an English (target language) word too many times to other Spanish words (source language).

B. Method Overview

*1) **High-Level Description of Implementation**:* My implementation of IBM Model 2 includes two parts: training and testing. The training part is done by running 5 iterations of EM Algorithm. More details can be found in Algorithm 2. The testing part includes reading parameters as well as testing corpora and assigning alignment to each sentences pair with the highest $q * t$ score, i.e. $a_i = \underset{j \in 0 \dots l}{\arg\max} q(j|i,l,m)t\left(s_i|e_j\right)$.

C. Results

The result of my implementation matches the expected F1-Score. Details can be found in Table IV.

D. Discussions

*1) **Performances throughout iterations**:* The performances through the training process are given by the Table V. We can see from the table the following facts:

- The performances of IBM Model 2 is a clear improvement compared with IBM Model 1. The first iteration gives a 4% improvement over the final F1-Score of IBM Model 1. This is reasonable since our IBM Model 2's parameter $t(s|e)$ is initialized using the last set of parameters (after 5 iterations) produced by

TABLE IV: Performances of IBM Model 2

	Precision	Recall	F1-Score
Total	0.442	0.456	0.449

TABLE V: Performances of IBM Model 2

IBM Model 1	Precision	Recall	F1-Score	Growth Rate
	0.413	0.427	0.420	-
Iterations	**Precision**	**Recall**	**F1-Score**	**Growth Rate**
1	0.430	0.444	0.437	**4.0%**
2	0.436	0.450	0.443	1.4%
3	0.439	0.454	0.446	0.68%
4	0.438	0.453	0.445	-0.22%
5	0.442	0.456	0.449	0.90%
6	**0.443**	**0.457**	**0.450**	0.22%
7	0.443	0.457	0.450	0.0%
8	0.442	0.456	0.449	-0.22%
9	0.442	0.456	0.449	0.0%
10	0.442	0.457	0.449	0.0%

IBM Model 1 and IBM Model 2 introduced the alignment parameters q.

- Again, there is an interesting phenomenon that the growth rate is mostly always decreasing as the iteration goes. From iteration 2 to 5, the magnitude of improvement gradually decreases and even has a negative growth in iteration 4. But iteration 5 witnesses another improvement.
- There is a bottleneck in the performance of IBM Model 2. After 6 iterations, the performances almost arrives at its peak and will no longer increase.

*2) **Examples of alignments**:* There are two misaligned word pairs in the figure 1. "que" is misaligned to "hope" and "puedan" is misaligned to "use". Other than these two pairs, other word pairs are correctly aligned with each other. The accuracy is high enough to put it under the correctly aligned sentence category. The reason why this pair of sentences are aligned most correctly is that the sentences are in relatively simple grammatical structures. The alignments of these word pairs are one-to-one and the mapping of words is straight forward. Also, there are no complex phrases in the sentences.

IBM Model 2 lacks the ability to translate phrases. It only uses lexical parameters and distortion parameters to calculate the alignments. Therefore, when sentence pairs do not have complicated phrases or sentence structure, the performance of the model tends to be high.

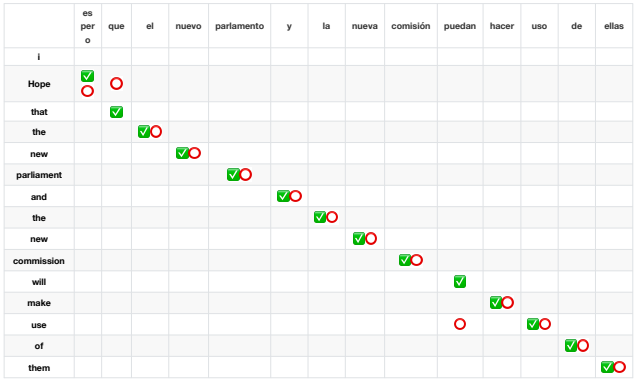

Fig. 1: An example of correctly aligned sentence. The red circle sign represents alignments generated by IBM Model 2 while the green tick sign represents the golden alignments for the sentence pair.

In figure 2, a misaligned example is shown. Different from the correctly aligned sentence, this sentence has a phrase "no reason not to" and its counterpart in Spanish "no razón para no". The Model clearly failed to catch the meaning of this phrase. This reveals the weakness of the model, which is not able to process phrases correctly.

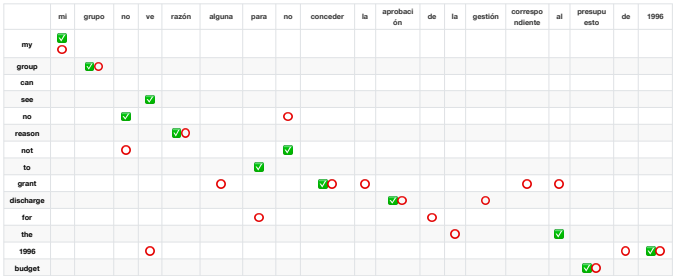

Fig. 2: An example of misaligned sentence.

III. Growing Alignments

A. Overview

*1) **Intuition**:* As we noted above, the gold alignments allow English words to be aligned with multiple Spanish words. The method is to train IBM Model 2 to calculate both $p(s|e)$ and $p(e|s)$ as a starting point. Take the alignments given by two sets of parameters help us to evaluate the alignments produced by IBM Model 2 in two directions: from English to Spanish and from Spanish to English.

*2) **High-Level Description of Implementation**:* We first train the IBM Model 2 for $p(s|e)$ and use the parameters in the model to produce the most likely alignment for each (e, s) pair. Then gain, we train IBM Model 2 for $p(e|s)$ and produce most likely alignments for each (s, e) pair.

The heuristic method used in the implementation starts with the intersection of the two sets of alignments, and grow the alignments accordingly. Any alignment point in the union of $p(s|e)$ and $p(e|s)$ could be a candidate when growing. One alignment point is added each time, and that one alignment point should be only chosen from those pairs who are currently without alignment assigned. To grow the alignments, word pairs without assigned alignment that are close to those who have been assigned would be explored first. After the initial intersection has stopped growing, we

now turn to other alignment points who are not neighbors of these points in the alignments.

B. Results and Discussions

TABLE VI: Performances of IBM Model 2

	Precision	**Recall**	**F1-Score**
IBM Model 2	0.442	0.456	0.449
Intersection	**0.823**	0.270	0.407
Union	0.320	**0.538**	0.401
Intersection + Union	0.662	0.369	**0.474**

As we can see from Table ref tab: growingalignment, it makes sense that the precision of the intersection model is the highest and the recall of the union model is the highest. Because the intersection operation will filter out those alignments with the highest confidence from both directions (Spanish to English and English to Spanish) and the union operation will include as many correct alignments as possible. When you implement the heuristic method to combine intersection and union wisely, then you'll get a not bad precision, a not bad recall, and the highest F1-Score.