A4 CSE 156 Report

David Dai
A13665770
Jun. 10/2019

Section 1. IBM Model 1

1. Description of IBM Model 1

a. Model Description

IBM Models are usually called IBM alignment model. As the name suggested, IBM models are a statistical machine translation to train an alignment model. And in this section, I implemented the first IBM model for machine translation. The goal is to create an alignment between an English Corpus and a Spanish Corpus.

b. Formula Description

The IBM Model 1 formula is provided below.
The overall goal is to create a probabilistic distribution of $p(f_1 ... f_m \mid e_1 ... e_l, m)$.
f is a foreign word, which is Spanish in this case. e is an English word and m is the length of the foreign sentence.
To achieve the goal, we need to manipulate $t(f \mid e)$ which can be interpreted as the conditional probability of generating a foreign, Spanish, word from an English word.
We initialize $t(f \mid e) = \frac{1}{n(e)}$ where $n(e)$ is the different number of words encountered during the translation of a sentence containing the English word e. Then
Then, $t(f \mid e)$ will be updated using the EM algorithm and hoping that it will converge.
More EM algorithm will be explained in the next section.
Lastly, we will run 5 iterations of EM algorithm attempting to reach the $argmax_{j \in \{0 \ldots l\}} t(f_i \mid e_j)$ for each alignment a_i. In the end, after all the iterations, the alignment will be generated.

c. IBM Model 1 Limitations

1. IBM Model 1 requires to have the terms have the same kinds of reordering. It also has a limitation that adding and dropping words for alignment should be the same across the model. In reality, however, IBM Model 1's presumptions are rather naive. There are many translations where reordering, adding or dropping words have different cases.
2. The other problem with IBM Model 1 is the fertility problem. In IBM Model 1, it has a tendency that the input words and the output words have a one-to-one relationship. However, for many foreign languages, it is not the case. Some words might produce multiple words and some might even be dropped.

2. Description of EM Algorithm

a. EM Algorithm Description

The EM algorithm can be simplified into three steps

$$\delta(k,i,j) = \frac{t\left(f_i^{(k)} | e_j^{(k)}\right)}{\sum_{j=0}^{l_k} t\left(f_i^{(k)} | e_j^{(k)}\right)} \quad \rightarrow \quad c\left(e_j^{(k)}, f_i^{(k)}\right) \leftarrow c\left(e_j^{(k)}, f_i^{(k)}\right) + \delta(k,i,j) \quad c\left(e_j^{(k)}\right) \leftarrow c\left(e_j^{(k)}\right) + \delta(k,i,j) \quad \rightarrow \quad t(f|e) = \frac{c(e,f)}{c(e)}$$

The EM algorithm is an iterative algorithm attempting to converge to find the Maximum Likelihood Estimation from the 'corpus.en' and 'corpus.es'. The EM iteration alternates between the calculated expected value and the maximized value updated along each step of the iteration.

b. Strengths and Weaknesses

1. Strengths:
    a. The EM algorithm can converge closer to the ideal Maximum Likelihood Estimate through each iteration; thus, if there are infinite computing power and infinite iterations, the EM algorithm may get infinitely close to convergence
    b. It requires limited, actually a few, parameters constraints only
    c. It is not super computational heavy since it does not requires gradient computing or an optimizer.
2. Weaknesses
    a. This algorithm converges really really slow.
    b. It will not reach the ideal Maximum Likelihood Estimate.
    c. It is hard to implement compared to numerical optimizations

3. Method Overview

The IBM Model 1 high-level implementation will be described below.
Reading the corpus requires to remove all the phrases that are empty. The program will store the words, phrases in separate lists and it will also store the words' index in sets.
Initializing the $t(f \mid e)$ by looping through all the word pairs of English and Spanish and counting each appearance in a set.
Using the EM algorithm to update the $t(f \mid e)$ value requires two functions, $delta$, and t. The $delta$ function could be viewed as the expected counts for the current iteration. Meanwhile, t is simply for updating the $t(f \mid e)$ value.
Then, it will store the $t(f \mid e)$ which an numpy matrix into a file.
For the evaluation part, the program will read the development data corpus and generate its $t(f \mid e)$ and write the alignment to a file for evaluation.
(P.S. full implementation in IBM_Model_1.py)

4. Results

Expected F1-Score 0.420

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

5. Discussions

Iteration 1 out of 5

Type	Total	Precision	Recall	F1-Score
total	5920	0.211	0.217	0.214

Iteration 2 out of 5

Type	Total	Precision	Recall	F1-Score
total	5920	0.375	0.387	0.380

Iteration 3 out of 5

Type	Total	Precision	Recall	F1-Score
total	5920	0.402	0.415	0.408

Iteration 4 out of 5

Type	Total	Precision	Recall	F1-Score
total	5920	0.410	0.423	0.416

Iteration 5 out of 5

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

As the table is shown above, there is a positive correlation between the number of iterations and precision; and there is also a positive correlation between the number of iterations and the F1-Score. More iterations mean there will be a higher precision and higher F1-Score; however, the increasing amount of both the precision and F1-Score decreases as the iterations increase. Rephrasing the above sentence, as more words converge to the $argamx$, the smaller the precision and F1-Score will change.

Section 2. Semi-supervised Learning

1. Description of IBM Model 2

a. Formula Description

The IBM Model 2 formula is provided below.
The overall goal is the same as IBM Model 1 which is to create a probabilistic distribution of $p(f_1 ... f_m \mid e_1 ... e_l, m)$.
f is a foreign word, which is Spanish in this case. e is an English word and m is the length of the foreign sentence.
To achieve the goal, we need to manipulate $t(f \mid e)$ which can be interpreted as the conditional probability of generating a foreign, Spanish, word from an English word. And we also need to manipulate $q(j \mid i, l, m)$ where j, i are the position of the corresponding English word and Spanish word in the sentence. l is the length of the English sentence.
We initialize $t(f \mid e) = \frac{1}{n(e)}$ where $n(e)$ is the different number of words encountered during the translation of a sentence containing the English word e.
We then initialize $q(j \mid i, l, m) = \frac{1}{l+1}$
Then, $t(f \mid e)$ will be updated using the EM algorithm and hoping that it will converge. And along with the EM algorithm, we will update the $t(f \mid e)$ and $q(j \mid i, l, m)$ value. Lastly, we will run 5 iterations of EM algorithm attempting to reach the $argmax_{j \in \{0 \ldots l\}} q(j \mid i, l, m)\, t(f_i \mid e_j)$ for each alignment a_i. In the end, after all the iterations, the alignment will be generated.

b. Two Models Comparison

As I have described in the first section about the weaknesses of IBM Model 1, IBM Model 1 has a hard time manipulating dropping words, adding words or reordering the words. IBM Model 2, comparing to Model 1, introduces a new parameter $q(j \mid i, l, m)$, the probability of the alignment distribution. The goal is to have a better result regarding the

reordering of the words. Thus, IBM Model 2 tends to generate better results than IBM Model 1.

c. **IBM Model 2 Limitations**
IBM Model 2 's new parameter $q(j \mid i, l, m)$ is only designated to fix the re-ordering problem of IBM Model 1. It still cannot fix the problem of the tendency to generate results, which have the same amount of words as the input language. The IBM Model 2 still has a problem dealing with adding or dropping words. And unluckily, IBM Model 2 increase the complexity of the model which takes even more steps to reach closer to the Maximum Likelihood Estimates.

2. Method Overview

The IBM Model 2 high-level implementation will be described below.
Reading the corpus requires to remove all the phrases that are empty. The program will store the words, phrases in separate lists and it will also store the words' index in sets.
Initializing the $t(f \mid e)$ by looping through all the word pairs of English and Spanish and counting each appearance in a set.
Initializing the $q(j \mid i, l, m)$ which is counting the alignment probability by looping through the English and Spanish data and store them in a set.
Loading the $t(f \mid e)$ which is generated after 5 iterations of IBM Model 1 from a file.
Using the EM algorithm to update the $t(f \mid e)$ and the $q(j \mid i, l, m)$ value requires three functions, $delta$, t, q. The $delta$ function could be viewed as the expected counts for the current iteration. Meanwhile, t is simply for updating the $t(f \mid e)$ value and q is updating the corresponding $q(j \mid i, l, m)$.
For the evaluation part, the program will read the development data corpus and generate its $q(j \mid i, l, m)$ $t(f \mid e)$ and write the alignment to a file for evaluation.
(P.S. full implementation in IBM_Model_2.py)

3. Results

Expected F1-Score: 0.450

Type	Total	Precision	Recall	F1-Score
total	5920	0.443	0.457	0.450

4. Discussion

Iteration 1 out of 5

Type	Total	Precision	Recall	F1-Score
total	5920	0.433	0.447	0.440

Iteration 2 out of 5

Type	Total	Precision	Recall	F1-Score
total	5920	0.436	0.450	0.444

Iteration 3 out of 5

Type	Total	Precision	Recall	F1-Score

total	5920	0.439	0.453	0.447

Iteration 4 out of 5

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

Iteration 5 out of 5

Type	Total	Precision	Recall	F1-Score
total	5920	0.443	0.457	0.450

As the table is shown above, there is a positive correlation between the number of iterations and precision; and there is also a positive correlation between the number of iterations and the F1-Score. More iterations mean there will be a higher precision and higher F1-Score; however, the increasing amount of both the precision and F1-Score decreases as the iterations increase. Rephrasing the above sentence, as more words converge to the $argamx$, the smaller the precision and F1-Score will change.

Comparing the results to IBM Model 1, IBM Model 2 has higher precision and F1-Score.

For the examples, I could not find sentences that are exactly correct, the following examples for all incorrect, but close to be aligned to the key. I think the one way we can improve the accuracy to have a exact correct sentence translation is by having a larger training set with more data. And there are many problems with the reordering of the words and the fertility problem regarding the model.

sixthly: hygiene.
en sexto lugar: educación sanitaria.

dev.key	out.key
9 1 1 9 1 2 9 1 3 9 2 4 9 3 6 9 3 5 9 4 7	9 1 2 9 1 3 9 2 4 9 3 1 9 3 5 9 3 6 9 4 7
En sexto lugar:	Sanitaria educacion.
sexto lugar:	educación sanitaria.

i have always had confidence in this .
siempre he tenido confianza .

dev.key	out.key
11 2 2 11 3 1 11 4 3	11 1 2 11 3 1 11 4 3

11 5 4 11 8 5	11 5 4 11 8 5
(space) he	siempretenido confianza
he (space)	siempre he tenido confianza

5. Critical Thinking

IBM Model 1 and 2 both have problems regarding dropping and adding words. Both of the models tend to have a fertility problem as well. Thus, to further improve the results, I have the following two suggestions:

1. I have a proposal to solve the fertility problem. In IBM Model 1 and 2, both cases are reading the words for words; however, what if we decided to read them as trigrams with * for NULL. For an English sentence f = a b c, where a, b, c are separate words. Word b will create abc, *bc, ab*, *b*, four trigrams. And for a Spanish sentence f' = a' b' c', it will also create three trigrams of a'b'c', *b'c', a'b'* and *b'*. Then, we run IBM model 2. I think this way it can reduce some of the one-to-one problems, where the input words are equal to the output words amount.
2. The other solution I think is introducing HMM, Hidden Markov Model along with the IBM Models. This way, with grammar being introduced, can improve the reordering problem that IBM Model 2 only partially alleviated. And create the alignment according to that. For the translation sage, we can use RNN machine learning afterward to check for sentence structure and whether it makes sense or not.