Machine Translation: IBM Models I & II

1. Introduction

The IBM Models I & II were groundbreaking models for machine translation, utilizing a supervised learning approach to the problem. The larger approach was the use of a noisy channel model to decode English corpora from foreign ones. The IBM Models were to act as the channel models in this approach with other language models acting as the source model for English words. The goal then is for the models to estimate $P(f|e)$, the probability of a foreign sentence given an English one. Producing a foreign sentence translation from this model alone then amounts to solving for $\max_f P(f|e)$.

Predicting $P(f|e)$ is inherently a challenging task for maximum likelihood estimation. The variations of all possible sentences leave us with incredibly sparse data. Further, breaking it down to the product of individual word pairing will definitely fail as other languages represent the same concepts in various different sentence lengths and orderings. Thus, the models incorporate more data into their analysis by also modeling the index mappings of foreign words to the words they are to have originated from. The *alignments, a*, are incorporated by marginalizing over all possibilities as such $P(f|e) = \sum_a P(f, a|e)$.

Thus, the models seek to estimate $P(f_1 \dots f_m, a_1 \dots a_m | e_1 \dots e_l)$, with the assumption foreign words map to only one word in the translated sentence. The models assume that f_i and a_i are conditionally independent of everything else given its aligned word e_{ai}, its index and the sentence lengths (m&l). This simplifies the joint conditional probability to $\prod_{i=1}^{m} P(f_i, a_i | e_{a_i}, i, l, m)$.

To further disentangle the prior probabilities of f_i and a_i, The models assume the independencies $(f_i \perp\!\!\!\perp a_i, i, l, m | e_{a_i})$ and $(a_i \perp\!\!\!\perp e_{a_i} | i, l, m)$, Essentially, the probability of the foreign word is dependent only on the word it is aligned to and the probability of the alignment between English/foreign sentence indices is only a factor of the indices and the sentence lengths. It is a large simplifying assumption, that is mitigated by the fact that the probability of the foreign word is still dependent on the quality of an alignment via its aligned word, but it nicely simplifies the joint conditional probability into an easier to quantify product: $\prod_{i=1}^{m} P(f_i | e_{a_i}) P(a_i | i, l, m)$.

The goal of this paper is to use the EM algorithm to estimate probabilities $P(f_i | e_{a_i})$ & $P(a_i | i, l, m)$, from parallel corpora of 5041 English and translated Spanish sentences, without providing alignments in the training data.

We will not be testing the models' ability at sentence translations, but instead utilize those estimated probabilities to test its ability at predicting alignments given parallel Spanish and English dev corpora, by solving for $\operatorname{argmax}_a P(a | f, e)$, maximizing the normalized probability of the foreign sentence and alignment given the English sentence out of all possible alignments, $\operatorname{argmax}_a \frac{P(f, a|e)}{\sum_a P(f, a|e)}$. Also, in recognizing that words in a foreign sentence can be structurally necessary but not directly mapped to any English counterpart, we have introduced a special NONE symbol and prepended it to every English sentence to allow our model to train for that possibility.

2. IBM Model I

2.1 IBM Model I Description

What sets IBM Model I apart from the general outline above is that it skips the EM Algorithm estimation of $P(a_i | i, l, m)$ and just sets the value to $\frac{1}{(l+1)}$. Essentially, it assumes it is equally likely that the Spanish word maps to all words in the English sentence or NONE (Hence, the +1).

As previously mentioned, the models were used for machine translation. The limitations therein, specifically for Model I, is that the model makes very large simplifying assumptions. First, it assumes there is only one English

word that a foreign one can possibly align to, but in reality, a word can map to many things in its foreign translation. For instance, German words are often compounds of many individual ideas that are only singularly expressed in English. Therefore, it requires many alignments. Second, it makes the assumption that the likelihood of a word is influenced only by the word it's aligned to. In fact, words can be influenced by many others in a sentence, even with multiple alignments. For example, the Spanish language uses grammatical gendering, requiring the knowledge of a subject's gender to influence the gendering of other nouns and adjectives. Further, words with many meanings in one language require context to decipher in another, including the use of phrases.

Lastly, IBM Model I specifically assumes the alignments of two parallel indices are uniformly likely for all possible pairings on a given sentence. This is almost certainly not the case, as languages produce consistent structure that places translated words in consistent distributions throughout a sentence. Further, many languages have similar roots and thus have similar structure that place aligned words at neighboring indices. Compound sentences in these languages will almost certainly disjoint the possible indices that many words can have, altering the distributions that Model II naively assumes as uniform.

2.2 Description of EM Algorithm

The EM algorithm estimate probabilities $P(f_i|e_{a_i})$ and $P(a_i|i,l,m)$ as parameters $t(f_i|e_{a_i})$ and $q(a|i,l,m)$. It does so by initially randomizing the parameters. This allows for non-zero values that would not allow the algorithm to update its estimations. It then uses a hill-climbing method to iteratively updates the parameters to optimize for maximizing the data likelihood, in our case the probability of the Spanish corpus given an English one, $\max_{t,q} \prod_{(f,e)} P(f|e) = \max_{t,q} \prod_{(f,e)} \sum_a \prod_{i=1}^{m} t(f_i|e_{a_i}) q(a_i|i,l,m)$. This is often formally optimizing the log of the likelihood function.

The EM Algorithm works similarly to an ML Estimation algorithm that collects counts of a word, alignment, sentence length and index groupings and then estimates the parameter probabilities as the relative frequency. However, this of course requires complete alignment data in the training set.

For the EM Algorithm, it takes a similar approach at each iteration, however instead of checking all possible variations of $(f,e), f_i, a_i$, to count training corpus instances of (f_i, e_{a_i}) & (a_i, i, l, m), it instead adds the normalized probability $P(f_i, a_i|e_{a_i}, i, l, m)$ out of all possible English word alignments in the sentence, $\frac{q(a_i \mid i,l,m) t(f_i|e_{a_i})}{\sum_{j=o}^{l} q(j|i,l,m) t(f_i|e_j)}$. Thereby essentially weighting the count of a possible grouping by the likelihood its alignment would be seen in a sentence given the current parameters.

At the end of each iteration, the parameters are updated by the relative frequency of the weighted counts. Over various iterations, the parameters will self-correct towards more accurate values as overestimated parameters are diminished due to infrequent sightings and underestimated parameters are boosted by frequent sightings. Further, better-estimated alignment probabilities will be boosted by and dimmish the value of worse alignment probabilities within its sentence.

The algorithm's strength is not needing complete data to create parameter estimations, it can estimate them from the training data. A weakness is the increased computational complexity of iterating over all possible alignments in a sentence. Further, the likelihood function is not concave and thus the EM Algorithm might get stuck at a local maximum, which is why good parameter initialization is key.

2.3 Method Overview

As previously discussed, Model I only estimates the parameters $t(f_i|e_{a_i})$, which we will initialize as $\frac{1}{n(e_{a_i})}$, where $n(e) = \#Unique\ Words\ seen\ in\ a\ parallel\ sentence\ to\ e$. That is, our initial guess is that a foreign word's chance of being aligned with an English word is the same chance as a word from a uniform distribution of all foreign words in translations of sentences containing the English word. Now, for a machine translation objective, this might be a poor starting choice for this parameter, as there are many foreign words likely not seen with many English ones and could create non-normalized conditional probabilities within some sentences. However, for our objective of predicting alignments for already translated sentences, it makes more sense, because we will be conditioning foreign words on English words that are, by construction, seen in their translated sentences.

Our implementation of this algorithm is in Python 3.8. The initialization of parameters proves to be a challenging one, as the variations of f_i, e_{a_i} pairing are quite large and demanding on memory. In that regard, we will be utilizing a *defaultdict* data structure, a mapping dictionary that returns a default value for inputs not yet mapped. We will be using 2 layers of such mappings. To return $t(f_i|e_{a_i})$ from f_i, e_{a_i}, get $map2 = map1(e_{a_i})$ and $map2(f) = t(f_i|e_{a_i})$. Initially, $map1$ will be set to a return a map that returns 0 by default for all e_{a_i}. Then $n(e_{a_i})$ will be calculated for all e_{a_i} and for each, $map1(e_{a_i})$ will be set to a map that returns $\frac{1}{n(e_{a_i})}$ by default.

Aside from that, the E.M. algorithm is implemented mostly as described. For our experiment we will be iterating 5 times. Naturally, the parameters $t(f_i|e_{a_i})$ were only estimated after each iteration for words f_i, e_{a_i} that were seen in parallel sentences. No issues were seen during implementation, apart from memory ones described above and some haphazard refactoring.

2.4 Results

To test the distributions estimated by the EM Algorithm, we will be estimating the alignments for Spanish and English sentences in parallel dev corpora. For a Spanish sentence $f_1 \dots f_m$ and an English sentence $e_1 \dots e_l$, we will be solving for $\underset{a_1 \dots a_m}{\operatorname{argmax}} P(f_1 \dots f_m, a_1 \dots a_m | e_1 \dots e_l)$ which in this context simplifies to individually solving $\underset{a_i}{\operatorname{argmax}}\ t(f_i|e_{a_i})$ for each a_i. The alignment predictions are tested against a manual aligning (sans null alignments). The results of which are shown above.

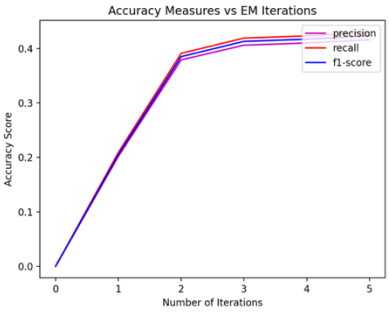
Precision	Recall	F1-Score
0.416	0.430	0.423

2.5 Discussion

To see more precisely how the EM algorithm improves the parameter estimations, below we have shown the results at each iteration, including the initialization. We have plotted these changes, noting the specific value amount the F1 score has changed since the previous iteration.

Interestingly, the improvement from the first two iterations is roughly the same amount, producing an almost linear trend. Then, as the EM Algorithm nears its local optima, the accuracy improvements in parameter estimations diminish rapidly as the EM Algorithm nears its final prediction. The trends are the same for every measure of accuracy.

Iterations	Precision	Recall	F1	Change
0	0.000	0.000	0.000	-----
1	0.202	0.209	0.205	0.205
2	0.379	0.391	0.385	0.18
3	0.406	0.419	0.413	0.028
4	0.410	0.423	0.417	0.004
5	0.416	0.430	0.423	0.006



3. IBM Model 2

Description of IBM Model 2

The IBM 2 makes improvements upon Model 1 by not making the assumption for $P(a_i|i,l,m)$ that all possible alignments (disregarding the words) are uniformly likely by setting $P(a_i|i,l,m) = \frac{1}{l+1}$. Instead, IBM 2 asserts that such a blind alignment isn't uniform, but dependent on the foreign word's index, and the lengths of the sentences. It seeks to estimate that probability through the EM algorithm as well as the parameter $q(a_i|i,l,m)$.

Thus, this model is an improvement on Model 1, because it adds more complexity and context to the analysis of language translations that will give it a more accurate understanding of the interplay of language.

The limitations of Model II, apart from the ones described for Model I, is that it assumes the alignment of words between translated sentences is only a factor of the general likelihood for a word to map onto another given the relative position of the words in the sentences and the length of the sentence. In reality, languages are quite complex and not dependent on this specific relation. Further, the addition of adjectives, qualifiers, and compound sentences can significantly throw off the indices and lengths in these calculations. Even using the

indices relative to one of the words could possibly make more sense rather than the absolute indices, but alas that is not taken into consideration and thus the alignments captured are vulnerable to significant noise.

Method Overview

We could initialize the parameter $t(f|e)$ as we had before, but since we have gotten better data on its likely values, we will be reusing the parameter estimations collected by Model I. To initialize $q(a_i|i,l,m)$ we will be setting it to $q(a_i|i,l,m) = \frac{1}{(l+1)}$. To avoid possible memory issues, we will be utilizing a similar default mapping structure as was presented with Model I. We will also be running the EM Algorithm for 5 iterations.

Much of the structure of the EM Algorithm and model was shared with Model I and thus reused. Therefore, little changed except the counts update in the EM algorithm and the estimation from those counts. There were no issues for this implementation, although countermeasures were still employed for memory issues. Again, unnecessary, haphazard refactoring did prove to cause some issues.

Results

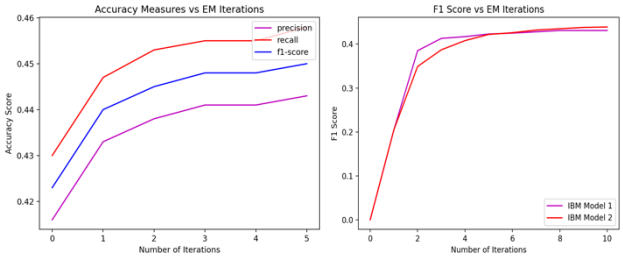
Alignment predictions were then made on the dev set and tested as previously discussed. Although results to the right show Model II beating Model I's measures, it's not really comparable as Model II was initialized with Model I's parameters.

Precision	Recall	F1
0.443	0.458	0.450

Discussion

To see more precisely how the EM algorithm improves the parameter estimations, to the right we have shown the results at each iteration, including the initialization. We have plotted these changes, noting the specific value amount the F1 score has changed since the previous iteration. We can see the EM iteration improvements have considerably less slope than Model I's. This is likely due to the initializations being a lot closer to the EM Algorithm's final predictions. Yet, continuing off of Model I's work, the iterations increase the slope of improvement from where Model I left off, suggesting this is a better model. It's also interesting to note that the pace of change is not monastically decreasing. Therefore, it might be difficult to know when the algorithm nears its optima. Also, to the right, we can see a comparison of F1 Scores over 10 iterations between the models, with the same initialization. Surprisingly, Model I improves much faster initially, although Model II overtakes it in performance eventually however marginally.

Iterations	Precision	Recall	F1	Change
0	0.416	0.430	0.423	-----
1	0.433	0.447	0.440	0.017
2	0.438	0.453	0.445	0.005
3	0.441	0.455	0.448	0.003
4	0.441	0.455	0.448	0.0
5	0.443	0.458	0.450	0.002



On the right, we can see two different sets of parallel sentences from the dev corpus. For the sentence "One issue that separates us is the civil war in Chechnya", IBM Model II correctly predicted the alignments. Whereas with the sentence "No statistical data exists," the alignments were not correctly predicted. We can see the alignment relationships marked on the matrix with an "A" for the actual alignments and a "P" for incorrectly predicted alignments. Interestingly, the correctly aligned sentences are of the same length, and the incorrect ones are not. This could potentially be the cause, as parallel sentences of the same length are more likely to contain relatively similar sentence structuring. Further, the incorrectly aligned sentences differ more in the possible index differences between aligned words, which could also hamper efforts to estimate the pairings, as probabilities of similar indices outweigh the likely probabilities of direct word translations.