Revisiting IBM-1 and IBM-2 Models for Machine Translation

Kai Liang
University of Amsterdam
1012 WX, Amsterdam

Billy Chan
University of Amsterdam
1012 WX, Amsterdam

Yijie Zhang
University of Amsterdam
1012 WX, Amsterdam

{kai.liang, billy.chan, yijie.zhang}@student.uva.nl

Abstract

We implement the word-based IBM-1 and the IBM-2 models for machine translation between English and French and evaluate them with respect to the log likelihood on the training set and the AER on the validation test sets. Also, the effects of different initialization methods on the convergence of the IBM-2 model is experimented and discussed. Finally, the IBM-1 and IBM-2 models are empirically compared for their AER on the test set.

1 Introduction

Though invented decades ago (Brown et al., 1993), word-based alignment models such as IBM-1 and IBM-2 remain fundamental to the core of today's statistical machine translation systems. Beginning with the lexical translation probabilities, the IBM models aim at learning and predicting the translations in a probabilistic framework, which provides reliable inferences with uncertainty.

In this study, we revisit the IBM-1 and -2 models and conduct experiments to explore the model behaviours as well as to compare the model performances. For IBM-1, the model is trained with the expectation-maximization (EM) algorithm and record the model performance with respect to the training log likelihood and the validation alignment error rate (AER). For IBM-2, due to its non-convexity nature, besides the aforementioned experiments, we also experiment the effect of different parameters initialization methods on the model performance. At the end, comparisons of the IBM 1 & 2 models are discussed in terms of model performance and behaviors.

2 Model[1]

The final implementations of the IBM-1 and IBM-2 models are done in Python 3[2], which we further elaborate in the following subsections.

2.1 Translation model

Given a parallel corpus of N paired sentences, the task of the IBM-1 and IBM-2 is to approximate the translation probabilities $P(\boldsymbol{f}|\boldsymbol{e})$ (Brown et al., 1993). For every $n \in N$, we have a French sentence f_1^I of length $|I|$, and an English sentence e_0^J of length $|J+1|$, which respectively consists of words with indices in $[1, I]$ and $[0, J]$ (0 for the prepended $null$ word to handle French words not generated by any original English word). Besides, given the assumptions that all the sentences are generated independently, and that each French word is a translated from exactly one English word (can be $null$), we supplement the models with an alignment vector $\boldsymbol{a}^I$, where each a_i represents the position of the English word the French word f_i is aligned to.

With alignment vectors as latent variables that associate the French words with their corresponding English words, IBM-1 and IBM-2 can be interpreted as a constrained mixture models with its mixture components being the English words, and constrained in the sense that the mixture components generating the French words are constrained to the English words that happen to appear in the same sentence pair.

The lexical translation probabilities $P(\boldsymbol{f}|\boldsymbol{e})$ can hence be re-expressed as follows. (Notice that the derivations below utilize the assumptions that: (1) Each French word is independent from each other

[1]The line of explanations largely follows the literature given on the syllabus and in the tutorial: (Schulz, 2017), (Lopez, n.d.)

[2]https://github.com/s2948044/Natural-Language-Processing-2

given the English sentence. (2) a_i only depends on the sentence length and the English position i. (3) f_i only depends on the aligned English word):

$$
\begin{aligned}
p(f^I, a^I|e^J) &= p(I|e^J) \cdot p(f^I, a^I|I, e^J) && (1)\\
&= p(I|e^J) \prod_{i=1}^{I} p(a_i|i, I, J) \cdot p(f_i|e_{a_i}) && (2)\\
&\propto \prod_{i=1}^{I} p(a_i|i, I, J) \cdot p(f_i|e_{a_i}) && (3)
\end{aligned}
$$

The objective of the IBM models is hence to construct a model that maximizes $p(f^I, a^I|e^J)$.

2.2 IBM-1

To attain computationally tractability, IBM-1 simplifies Equation 3 by making the assumption that the alignments are not weighted, so in other words, for each French word, the term $p(a_i|i, I, J)$ is distributed uniformly (i.e. $\frac{1}{J+1}$). Hence, we are only left with the term $p(f_i|e_{a_i})$.

2.3 EM algorithm

The EM algorithm is an iterative method to search for the Maximum Likelihood estimates of parameters in the statistical model with unobserved latent variables, which consists of two steps: the E-step and the M-step. In the E-step, updated estimate of the parameters are used to evaluate the expectation of log likelihood. Then, in the M-step, the parameters that maximizes the expected log likelihood in the E-step are re-computed. The above process repeats until some criterion of convergence are met.

For IBM-1, in E-step we first calculate the posterior distribution over all the mixture components based on the current parameters. That is, for each French word f_i, we first compute how much each mixture component is responsible for the observed outcome f_i. Recall that the probability of obtaining a mixture component given a French word is equivalently the probability of the latent variable $P(a_i = j|f_i, e_j)$. This is achieved by making use of our current estimate $P(f_i|e_j)$, and normalizing it over f_i's possible alignments:

$$P(a_i = j|f_i, e_j) = \frac{P(f_i|e_j)}{\sum_j P(f_i|e_j)}$$

By iterating through the entire corpus, we could calculate the sum of all the fractional counts of any $\{\#e, f\}$ word pair. For instance, consider an English word e and a French word f, to compute their expected co-occurrence counts, we sum over all the alignments among all sentences which connect the two in the corpus:

$$\mathbb{E}[\#ef] = \sum_i P(a_i = j|f_i = f, e_j = e)$$

Meanwhile, using this term, we could sum over all $f \in \boldsymbol{f}$ in the French vocabulary for a particular English word e to get the fractional counts of e: $\mathbb{E}[\#e]$, and complete the M-step by normalizing $\mathbb{E}[\#ef]$ with $\mathbb{E}[\#e]$. For a full update of likelihood maximization of the conditional probability $P(\boldsymbol{f}|\boldsymbol{e})$, we need to compute this for all $\{e, f\}$ pairs:

$$
\begin{aligned}
\mathbb{E}[\#e] &= \sum_{f \in \boldsymbol{f}} \mathbb{E}[\#ef]\\
P(f|e) &= \frac{\mathbb{E}[\#ef]}{\mathbb{E}[\#e]}
\end{aligned}
$$

2.4 IBM-2

IBM-2 differs from IBM-1 in that IBM-2 does not assume all alignments have equal weights, thus we compute the full-fledged likelihood in Equation 3, and the term $P(a_i|i, I, J)$ is no longer constant but also an optimized objective in EM. To avoid heavy computations, this value is approximated using the jump function (Vogel et al., 1996).

2.5 Jump function

Instead of training the full set of joint distribution for all a_i positions given different (i, I, J) values, the jump function provides an approximation of it by grouping the alignments based on how far an alignment 'jumps' from the French position to an English position, i.e. a categorical distribution of 'jumps' formulated below that exploit:

$$jump(a_i, i, I, J) = a_i - J \times (\frac{i}{I})$$

Intuitively, this function first computes the French location i relative to the French sentence length I by $\frac{i}{I}$, and then multiply the result by J to project the French word as if it were on the English sentence, after which the difference between the two positions on the English sentences are calculated.

Given the maximal length L for the English sentence in the training set, the parameters for alignments are a set of probabilities for a total of $2L+1$ distances of jumps ranging from $-L$ to $+L$.

Recall Equation 3, the EM algorithm for IBM-2 has a term $p(a_i|i,I,J) = p(jump(a_i,i,I,J))$ for every instance of $p(f_i|e_{a_i})$. Besides, for every $j \in [-L \cdots L]$, we perform the E-steps and M-steps for each of the $2L+1$ values of jumps with the probabilities $p(a_i|i,I,J)$ and $p(f_i|e_{a_i})$, where $jump(a_i|i,I,J) = j$.

2.6 Decoding

When the parameters are learned, alignments are predicted on the validation set or the test set. For this purpose, we adopt the commonly used 'Most Probable Alignment' algorithm (Lopez, n.d.) that finds the alignments of the highest probabilities:

$$\hat{\mathbf{a}} = \arg\max_{\mathbf{a}} p(\mathbf{a}|\mathbf{e},\mathbf{f})$$

To be more specific, the alignments are picked for IBM-1 and IBM-2 according to the following expressions:

$$\hat{a}_i = \begin{cases} \arg\max_{a_i} p\left(f_i|e_{a_i}\right) & \text{IBM-1} \\ \arg\max_{a_i} p(a_i|i,I,J)\left(f_i|e_{a_i}\right) & \text{IBM-2} \end{cases}$$

As we assume independence for alignments, each French word is aligned independently. Further, for each french word f_i in the sentence pairs to align, we iterate over all English words that co-occurred with f_i in the training set and predict the one with the highest posterior probability.

3 Experiments

3.1 Data

The datasets we use for the experiments are taken from the Canadian Hansards Corpus[3], which contains parallel sentence pairs between English and French. More specifically, the training set consists of 231164 pairs, the validation set consists of 37 pairs and the testing set consists of 447 pairs. Besides, ground truth alignments are provided for the validation and the testing sets.

3.2 Setup

3.2.1 IBM-1

As the optimization of the likelihood function of the IBM-1 is a convex problem (Lopez, n.d.), and given the size of the training set and the high complexity of the EM algorithm, we arbitrarily set the number of training iterations to 10, during which the evolution of training log likelihood and validation AER as a function of the iteration is recorded. Afterwards, the best models are selected based on each criterion (i.e. training log likelihood and validation AER) and tested on the testing set to obtain the AER and Viterbi alignments for each sentence pair.

[3]All test data are available from: https://uva-slpl.github.io/nlp2/projects.html

3.2.2 IBM-2

For IBM-2, due to its non-convexity, convergence depends on how the model parameters are initialized. Thus, we experiment the model with 3 different initialization methods, which are namely random (with 3 seeds), uniform and pre-trained lexical parameters from IBM-1 (with uniform jump parameters). For each initialization case, we again set the number of training iterations to 10, during which the evolution of training log likelihood and validation AER as a function of the iteration is recorded. Then, two best models are selected based on each criterion and tested on the testing set to obtain AER and Viterbi alignments for each sentence pair.

4 Results

We start by the experiment results of IBM-1. Figure 1 displays the evolution of AER on the validation set and the training log likelihood as a function of the iteration. It can be inferred from the figure that as training proceeds, both AER and training log likelihood are trending to the right directions, though AER fluctuates in the later iterations.

In terms of the model selection, as shown in Figure 1, we arrive at the same model for both metrics (i.e. AER and training log likelihood). Also, for our best model selected on both metrics, AER on test set using the gold-standard is shown in Table 1.

Table 1: The test AER of the best IBM-1 model and the best IBM-2 models

Model	Criteria	Alignment Init.	Jump Init.	Test AER
IBM-1	both	uniform	N/A	0.296
IBM-2	AER	uniform	uniform	0.274
IBM-2	likelihood	pre-trained	uniform	0.275

As for the results of IBM-2, Figure 2 shows the evolution of training log likelihood and validation AER as a function of the iteration under all the initialization cases. As can be inferred from the figure, the training log likelihood of the pre-trained

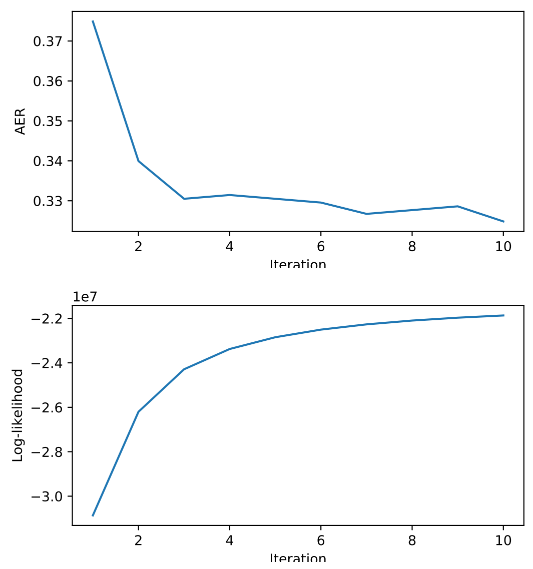

Figure 1: The evolution of training log likelihood and AER on validation data in terms of iterations for IBM-1.

initialized model performs the best, and the other models have similar performance. In terms of the validation AER, we can see that all models eventually converge to a similar level, but the uniformed initialized one as well as the pre-trained initialized one perform more stable.

As for model selection for IBM-2, as shown in Figure 2, the best model among all the cases w.r.t. training log-likelihood is the pre-trained initialized one, and the best model among all cases w.r.t. validation AER is the uniform initialized one. Then, these two selected models are tested on the testing set and the resulting AER are presented in Table 1, from which it is clear that both IBM-2 models outperform the IBM-1 model.

5 Conclusions

In this study, we implement and compare the IBM-1 and IBM-2 models for machine translation between English and French. For IBM-1, we reach the same best model with respect to both validation AER and training log likelihood, while as for IBM-2, we conclude that different initialization indeed leads to different results, and among which the uniform initialized model and the model initialized with lexical parameters from IBM-1 perform the best under the two criteria. Besides, for both the selected IBM-2 models, it is proved that they are better than the IBM-1 model in the sense that they have higher AER scores.

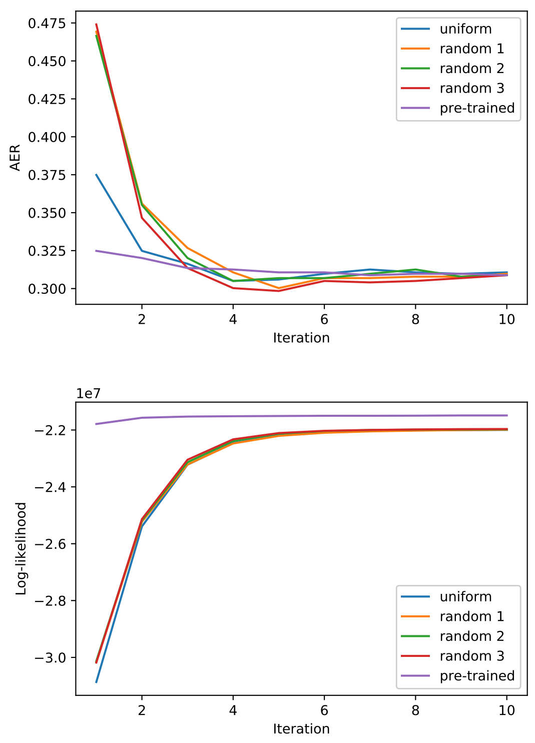

Figure 2: The evolution of training log likelihood and AER on validation data in terms of iterations for all cases of IBM-2.

However, there are also some limitations of this study. Firstly, due to time restrictions, the experiments are conducted for only 10 iterations, which might not represent the optimal performance of the models. Secondly, the cheaper parameterisation in terms of jumps is used as requested, which will weaken the real performance of the IBM-2 models.

References

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.* 19(2):263–311. http://dl.acm.org/citation.cfm?id=972470.972474.

Adam Lopez. n.d. Word alignment and the expectation-maximization algorithm.

Philip Schulz. 2017. The ibm mixture models 1 and 2 for word alignment. https://uva-slpl.github.io/nlp2/resources/papers/Schulz-IBM12-Tutorial.pdf.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 2*. Association for Computational Linguistics, Stroudsburg, PA, USA, COLING '96, pages 836–841. https://doi.org/10.3115/993268.993313.