AER evaluation for IBM1, IBM2 and Neural IBM1

Mathijs Pieters
12369705

Janne Spijkervet
10879609

Rob Hesselink
10587454

Abstract

In this work we investigate three statistical machine translation models: IBM1, IBM2 and Neural IBM1 (NIBM1). We discuss their underlying assumptions and their generative models. We also discuss how to train the models using the Expectation-Maximisation (EM) algorithm and test their Alignment Error Rate (AER) on the Hansards corpus (English - Canadian French). We compare AER and Log Likelihood for model selection, investigate the effect of the more expressive alignment probabilities in IBM2 and compare the effects of different initialisations.

1 Introduction

Machine translation finds its origins in cryptography. While efforts were initially on syntactical machine translation, statistical methods saw an increase in popularity in the 1980s (Koehn, 2009). IBM 1-6 are prototypical statistical machine translation models. While they relied on strong assumptions that are not always consistent with natural language, they were quite successful nonetheless. For an overview of the development from statistical models to current deep learning models, we refer the reader to Srivastava et al. (Srivastava et al., 2018).

This work describes three statistical machine translation models: IBM1, IBM2 and Neural IBM1. These models make use alignment variables that indicate pairs of words that translate into each other. Since these alignments are not observed (latent), the model parameters must be estimated using the Expectation-Maximisation (EM) algorithm. Models are trained on the Hansards corpus (English - Canadian French) and are evaluated using the Alignment Error Rate (AER).

In Section 2 we describe the models, in Section 3 we present the experiments and their results. We summarise our findings and our contributions in Section 4.

2 Model

This section describes the three models investigated in this work: IBM1, IBM2 (Brown et al., 1993) and Neural IBM1. Each of these models the conditional

$$p(f, a|e) = p(a)p(f|e) \tag{1}$$

where e is the source language, f is the target language, and alignments a are latent variables that describe how f is generated from e. By marginalising over all alignments a, the translation probability $p(f|e)$ can be computed.

2.1 Probabilistic Graphical Model

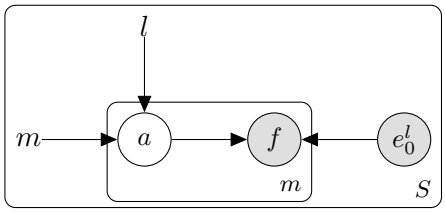

Figure 1: Generative model for IBM1, IBM2 and NIBM1

Figure 1 shows the generative model for the IBM models. From the source sentence e_0^l, a target sentence of length m is generated. The source words, combined with the alignments a iteratively generate the target sentence. The IBM models are mixture models. The alignments select the mixture components (source words), that specify the distribution that the target words are generated from. The IBM models perform lexical trans-

lation, since the alignments are independent and words are generated one by one.

Since parallel sentences of different languages need not consist of the same number of words, the source sentence is expanded with a *NULL* character at position zero. This way, additional characters in the target sentence can be generated by the *NULL* character in the source sentence. This alleviates the problem of one-to-one lexical translation somewhat.

2.2 IBM1

IBM1 assumes that alignments are independent and is described by a uniform distribution over the sentence length. Its conditionals are given by:

$$p(f, a|e) = \frac{1}{(l+1)^m} \prod_{j=1}^{m} p(f_j|e_{a_j}) \tag{2}$$

$$p(f|e) = \frac{1}{(l+1)^m} \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} p(f_j|e_{a_j}) \tag{3}$$

The distribution $p(f|e)$ is a categorical distribution, parameterised by $\theta_{e,f}$, which gives the probability that source word e translates into f. This means that

$$\sum_{f} p(f|e) = 1 \tag{4}$$

since every source word must map to a target word. The most naive way to initialise $\theta_{e,f}$ is using a uniform distribution, such that every English word has an equal probability to map to every French word in the corpus. However, if we have the train corpus available, we can determine beforehand which pairs of English-French words occur, such that we can make a more accurate estimate of $\theta_{e,f}$ where we assign a uniform probability to all pairs that exist in the corpus.

2.3 IBM2

IBM2 is very similar to IBM1, but has a more refined alignment probability. The alignment probability is a categorical distribution using parameters γ that is conditioned on the positions of the word pair and the sentence lengths. By using a more expressive distribution, it is possible to model more complex alignments. However, estimating the parameters γ for all possible combinations can be difficult for small corpora. The observation that aligned words often occur at approximately the same position in a sentence gives us a way to better generalize the alignment probabilities. In stead of calculating alignment probabilities for all possible combinations, we can use a function that maps several combinations to the same alignment probability.

In this work, we use the 'jump function', introduced by Vogel et al. (Vogel et al., 1996), leading to the conditional:

$$\begin{aligned} p(f, a|e) &= \prod_{j=1}^{M} p(a|i, j, l, m) p(f_j|e_{a_j}) \\ p(f|e) &= \prod_{j=1}^{M} \sum_{i}^{l} p(a_j = i|i, j, l, m) p(f_j|e_{a_j}) \end{aligned} \tag{5}$$

Here i and j are the positions in the source and target languages respectively and l and m their respective lengths. The alignment probability is then given by

$$\begin{aligned} p(a|i, j, l, m) &= \text{Cat}(\gamma[\text{jump}]) \\ \text{jump}(i, j, l, m) &= i - \left\lfloor \frac{j \cdot l}{m} \right\rfloor . \end{aligned} \tag{6}$$

The jump function ranges from $-L$ to L, where L is the length of the longest sentence in the source language. They are indexed by the jump function, which means that the model generates a probability for all possible forward and backward jumps. These jumps are not absolute (position 2 to position 5), but relative (a jump of a size normalised to the length of the sentence).

In this work we focus on three different methods to initialize the parameters. Whereas with IBM1 the optimization problem is convex, this is not the case for IBM2. Therefore the initial setting of the parameters is of great importance. We differentiate between uniform initialization, random initialization, and reusing the lexical parameters of the trained IBM1 model.

2.4 Neural IBM1 (NIBM1)

This model uses the conditional from equation 2, but uses a neural network to parametrise $p(f|e)$. We use Adam to optimize the objective function (Kingma and Ba, 2014) and Cross-entropy Loss as the loss function. The hyperparameters of our network are: a maximum sentence length of 30 and 64 dimensions for the word embeddings. The network consists of one Multilayer Perceptron with

two hidden layers of size 128. ReLU is used as the activation function between the layers (Glorot et al., 2011). The network is trained for 10 epochs, with a learning rate of 0.001.

2.5 Parameter Estimation

The parameters in the IBM1 and IBM2 models are estimated using the Expectation Maximization (EM) algorithm (Dempster et al., 1977). The estimation is done using an iterative computation of maximum-likelihood estimates. Each iteration of the algorithm has an expectation and a maximization step. These are repeated until it converges to a local optimum.

For IBM1, $|V_e + 1| \cdot |V_f|$ translation parameters θ are estimated. However, this is an upper bound, since many combinations of words never occur. For our data set the upper bound amounted to approximately 1.7 billion parameters, while in practice we used fewer than 1.2 million. IBM2 estimates the same number of θ's with an added $2L + 1$ γ's jump function is used. Not using the jump function yields $L^2 \times M^2$ parameters.

A disadvantage of using the EM algorithm is that it converges to a local optimum. The initialisation parameters greatly influences the convergence. For a more in depth explanation of how to specifically apply EM for IBM1 and IBM2, we refer to (Collins, 2011).

3 Experiments

3.1 Data Set

In our experiments we use the Hansard Corpus [1], consisting of English and Canadian French parallel texts, originating from the Canadian Parliament. The data set is described in Table 1.

	No. sentences	Unique tokens (e, f)
Train set	231664	36636, 46422
Test set	37	323, 345

Table 1: Metrics for the Hansards data set.

3.2 Metric & Setup

For this test set we calculate the Alignment Error Rate (AER) for the predicted alignment links, which is defined as:

$$1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|} \tag{7}$$

where A is the set of predicted labels, P is the set of probable links, and S is the set of true links. Note that in this research we do not differentiate between predicted and probable links, thus Equation 7 simplifies to

$$1 - \frac{|A \cap S|}{|A| + |S|}. \tag{8}$$

After training the models, the alignments are predicted using the Viterbi algorithm (Forney, 1973). To monitor progress, we compute the data likelihood of the train data set using the model descriptions in Section 2.

We train IBM1 for 30 epochs, and the three variants of IBM2 for 15 epochs. For the random initialized IBM2 we average the results over three experiments. In all experiments we select the model from the iteration where it achieves the optimal data log likelihood.

3.3 Results

AER Score & Model Selection

Table 2 shows the results of our experiments. With IBM1 we obtain an AER score of 0.367, all three IBM2 variants achieve a better AER score. As we can see, IBM2 generally scores better on the test set, due to its ability to expressively model the alignments. NIBM1 performs worse than both IBM1 and IBM2 with an AER score of 0.463, suggesting it is harder for this neural network's model to estimate the parameters than a generative model.

In Figure 2 we show the AER and negative log likelihood of the IBM1 models. Using the smart initialisation, we start with a slightly better model, but both models quickly converge to approximately the same result. The figure illustrates that the IBM1 model converges relatively fast, and at the end of training only minor improvements are made.

Based on AER scores, five training epochs are generally enough for the models to converge. While e.g. IBM1 and IBM2 with random initialisation show a tail with some improvements, these are relatively minor in comparison to the increase in score during the first five epochs. Using the log likelihood results in the same recommendation of five epochs. Only when initialising IBM2 using IBM1 does convergence occur within 5 epochs, which is the consequence of the translation parameters θ already having been trained on a similar likelihood objective.

[1] https://catalog.ldc.upenn.edu/LDC95T20

Model	*Negative LL*	*AER*
IBM1	15.1613	0.367
IBM2 - random	7.2197 ± 0.0007	0.335 ± 0.0013
IBM2 - uniform	7.2189	0.336
IBM2 - IBM1 init	7.2287	0.329
NIBM1	-	0.463

Table 2: Experimental results for IBM1 and the three variants of IBM2. For the random initialisation of IBM2 we denote the mean and standard deviation of the three independent experiments.

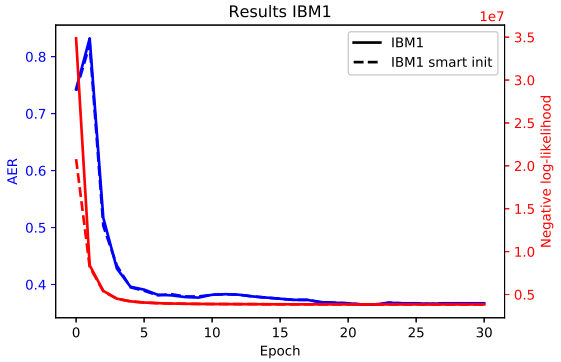

Figure 2: AER and negative log-likelihood for the IBM1 models.

Parameter Initialisation

Figures 3 and 4 show several training sessions for IBM2 with different initialisations. Both figures clearly illustrate that initialising the lexical parameters from the trained IBM1 model is an advantage, and causes faster convergence. The difference in performance between the random and uniform initialised models appears to be very minimal.

Alignment Parameters

Figure 5 shows the development of the alignment parameters during training of IBM2 with the lexical parameters initialised from IBM1. After the uniformly initialisation of the alignment parameters, the parameters quickly move towards a distribution that peaks near zero. This captures the tendency of the two languages to be quite similar in their syntax, since words that translate into each other generally occur at the same positions.

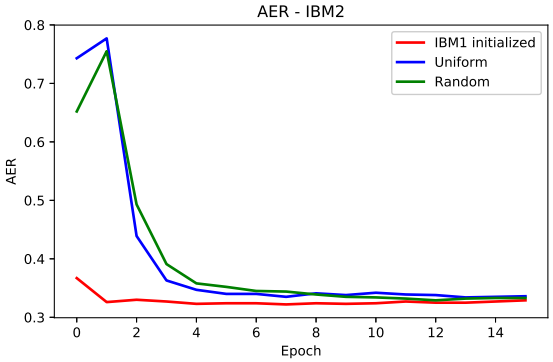

Figure 3: Alignment Error Rates of the three different IBM2 variants. For the random initialisation we show a single experiment, since the three independent experiments performed very similar.

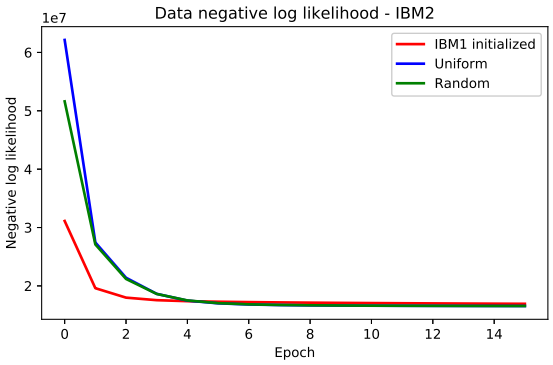

Figure 4: Data negative log likelihood of the three different IBM2 variants. For the random initialisation we show a single experiment, since the three independent experiments performed very similar.

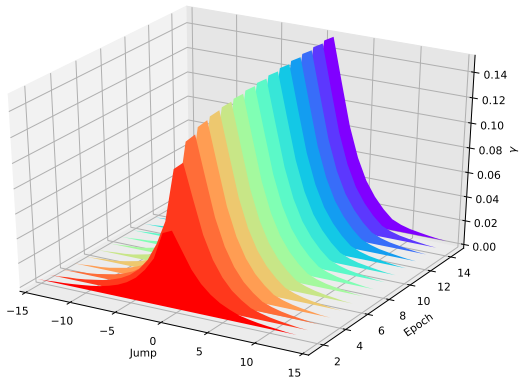

Figure 5: Variation of the gamma values (using the jump function) during training of IBM2, with lexical parameters initialized by IBM1. Iteration 1 is omitted, because the initialization is uniform.

4 Conclusion

In this research we have implemented IBM1 and IBM2, where the parameters are estimated using Expectation Maximization. For IBM1 we experimented with two variants of parameter initialisation, where smart initialisation gives slightly better results. However, since IBM1 with EM is a convex optimisation problem, they ultimately converge to the same solution. Introducing the alignment probabilities in IBM2 clearly improves the results compared to the IBM1 model. For IBM2 we tested three different initialisation methods. Both random and uniform initialisation are outperformed with respect to the AER by the IBM2 model that re-uses the lexical parameters from IBM1. We observed it is harder to estimate the parameters by a neural network than a generative model, as the NIBM1 model performed worse than both IBM1 and IBM2. More experiments should be performed to conclude statistical significance of the results and whether another neural network architecture could improve the AER score.

While the techniques discussed in this paper are outdated, and are greatly outperformed by more recent methods (e.g. neural networks) (Legrand et al., 2016), the techniques discussed in this paper remain important in the field of machine translation due to their simplicity and explicit probabilistic 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. *Computational linguistics* 19(2):263–311.

Michael Collins. 2011. Statistical machine translation: Ibm models 1 and 2. *Columbia Columbia Univ* .

Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)* 39(1):1–22.

G David Forney. 1973. The viterbi algorithm. *Proceedings of the IEEE* 61(3):268–278.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. pages 315–323.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Philipp Koehn. 2009. *Statistical machine translation*. Cambridge University Press.

Joël Legrand, Michael Auli, and Ronan Collobert. 2016. Neural network-based word alignment through score aggregation. *arXiv preprint arXiv:1606.09560* .

Siddhant Srivastava, Anupam Shukla, and Ritu Tiwari. 2018. Machine translation: From statistical to modern deep-learning practices. *arXiv preprint arXiv:1812.04238* .

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, pages 836–841.