Natural Language Processing 2: project 2

Fabio Curi and Mirthe van Diepen
fabio.curi.paixao@student.uva.nl, mmvandiepen@gmail.com

Abstract

This study applies two well-known translation models, namely IBM1 and IBM2, to train bilingual corpora and assign alignments to possible translations. Apart from the classical Expectation-Maximization training, a Variational Bayes model was also implemented for the first model. The performance of both models were indicated through both the perplexity of the train data and the alignment error-rate of the validation data.

Introduction

Within the context of translation models, the Expectation-Maximization training of bilingual corpora has been introduced by Brown et al. (1993), which enables a proper parameter estimation for five IBM models. In this study, only the two first models are explored. IBM model 1 (IBM1) is a rather straight-forward algorithm which updates the translation probabilities between two words accordingly to the occurrence and co-occurrence of both target and source language words within the bilingual phrase pairs. IBM model 2 (IBM2) is an extension of the first model; it differs in the alignments probabilities between two words. Additionally, a variational inference for Bayesian IBM1 (IBM1VB) was carried out, which adds an element which counts how many times words in bilingual data actually co-occur, unlike the two first approaches which actually update probability values only through the update of the conditional probabilities themselves.

First the models are defined. Then the evaluation measures are described. Thereafter, the experiments are clarified. Finally the results are stated and discussed.

Models

The Expectation-Maximization training for IBM1 and IBM2 involves an initialization of the translation and alignment probabilities of all word pairs of the two languages corpora. Then, a certain number of iterations is set to run the algorithm. In theory, the algorithm should be run until the perplexity values converge. However, due to RAM memory issues, this procedure had to be adapted through setting a fixed number of iterations to ten.

An additional smoothing has been enabled in the algorithm which follows the suggestion made by (Moore, 2004) to handle uncommon words. This procedure can also be carried out by simply replacing the uncommon words by a tag (namely, *UNK*). The selected words to be replaced by this tag were those which were only found less than five times in the corpora.

Furthermore, as detailedly explained by Collins (2011), an initial NULL-valued word is added to the beginning of each English sentence, suggesting that the alignment $a_j = 0$ specifies that word f_j is generated from the NULL word. This is an important step in all models since this model provides one-to-one alignments between the two corpora, enabling that words in the target sentence which are not directly translation from source sentence can still carry an alignment within it.

Finally, the source and target languages in this study were respectively English and French. Given that the aim is to train a translation model, probabilities are calculated of a French sentence $\boldsymbol{f}$ given an English sentence $\boldsymbol{e}$.

IBM1. The Expectation-Maximization (EM) training of IBM1 sets a uniform initialization for the lexicon parameters $t(f_i \mid e_i)$ between words in the two corpora, i.e. $t(f_i \mid e_i)$ is initialized with $1/|V_{\mathcal{F}}|$ where $V_{\mathcal{F}}$ is the French vocabulary. The

translation probability of $\boldsymbol{f}$ given $\boldsymbol{e}$ is

$$p(\boldsymbol{f} \mid \boldsymbol{e}) = \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} t(f_j \mid e_i).$$

This algorithm works accordingly to the following steps: *(i)* For every English word, the sum of probabilities of French words which are possible translations is computed; *(ii)* The normalized total probability that a given French word f is a translation of an English word e is computed; *(iii)* Estimation of the new lexicon parameters for all word pairs. In this step the t values are updated.

Note that for the model using *k*-smoothing, the smoothing is enabled in step (iii), and experiments were carried out with $k = 5$.

IBM1VB. For IBM1VB we use the same notations as in (Schulz, 2017). First the values of the parameters $\lambda_{f|e}$ are initialized with 0.5. Then EM algorithm is used to update the parameters $\theta_{f|e}$ and $\lambda_{f|e}$. The E-step consist of the update rule:

$$\hat{\theta}_{f|e} = \exp\left(\Psi(\lambda_{f|e}) - \Psi\left(\sum_{f'} \lambda_{f|e}\right)\right)$$

and the M-step consists of the update rule:

$$\hat{\lambda}_{f|e} = \alpha + \mathbb{E}[\#(e \to f)] = \alpha + \frac{\#(e \to f)\hat{\theta}_{f|e}}{\sum_{e' \in \boldsymbol{e}} \hat{\theta}_{f|e'}}$$

where the hyperparameter $\alpha = 0.001$ (as instructed during the lecture).

IBM2. This model extends IBM1 such that the word positions in the sentences are considered. For this model the alignment probabilities are introduced Brown et al. (1993). This model is more sentitive for initialization; it may converge to different local optima of the log-likelihood function. Hence, different initializations were introduced: uniform, random and using the translations probabilities of IBM1. To update the alignment and translation probabilities of the model the EM algorithm described in (Collins, 2011) was used. The translation probability of $\boldsymbol{f}$ given $\boldsymbol{e}$ is:

$$p(\boldsymbol{f} \mid \boldsymbol{e}) = \epsilon \prod_{j=1}^{m} \sum_{i=0}^{l} t(f_j \mid e_i) A(i \mid j, m, l)$$

where $A(i \mid j, l, m)$ is alignment probability of word on position i of the source sentence given word on position j of the target sentence, with l is the length of $\boldsymbol{e}$ and m is the length of $\boldsymbol{f}$. For this model ϵ is set to 0.1 and the unknown word tags are used.

Evaluation

The guarantee that the EM training of both IBM models has been properly carried out is indicated through the convergence of the perplexity values, meaning that the values do not decrease significantly after a certain iteration. The perplexity is dependent on the translation probability, which is expected to converge after some iterations. One challenge while calculating the perplexity values for each iteration was dealing with long sentences. The $p(\boldsymbol{f} \mid \boldsymbol{e})$ values for these sentences were sometimes rendering minus infinity, which is expected in the multiplication of translation probabilities which are already low. To avoid an undefined perplexity, all values in this report were calculating ignoring these extreme scenarios. For IBM1VB a special case of the perplexity is used to evaluate, namely ELBO (see (Schulz, 2017) for more details for computing ELBO).

The choice for the final alignments were done accordingly to Brown et al. (1993): choose the alignments which render the highest probabilities when multiplying sequent alignments among words in the two sentences. This method is similar to a Viterbi algorithm to calculate the path with highest profitability. It differs from this last given that there are no initial or final probability matrices for the alignments, nor a transition matrix. All that is provided to calculate a proper output for Viterbi is the emission translation probability $p(f_i \mid e_i)$ of word f_i of the target sentence given word e_i of the source sentence. Once the final alignments are chosen, the accuracy is measured through the alignment error-rate (AER) in regard to a provided golden standard with the respective "sure" and "possible" alignments between words. AER is a commonly used metric for assessing sentence alignments. It combines precision and recall metrics together such that a perfect alignment must have all of the sure alignments and may have some possible alignments.

Experiments

Several experiments were done to tune the parameters for the training. For IBM1 the different values (0.1 and 0.05) for ϵ were explored. Further-

more, the decision between applying k-smoothing or using the tag *UNK* was also used to create different IBM1's. For IBM2, the different initializations were used to define the models.

Data

Parallel corpora in both English and French were present in the training, validation and test datasets. Translation is here treated as a supervised learning problem in which sets of pairs $(\boldsymbol{e}, \boldsymbol{f})$ are instances of the data, and the AER serves as a validation tool for the trained alignments. The English corpus had a total vocabulary of size 36635, and the French one had 46421 words in its vocabulary. After replacement by UNK, roughly 43% and 41% of the English and French vocabularies were seen less than five times in the corpus.

Results

IBM1: EM. The preliminary aim in the implementation of the EM algorithm for IBM1 is to estimate the parameters: ϵ, and the k-smoothing or using the tag *UNK*. In Table 1 the different AER scores of the IBM1 are shown.

	***k*-smoothing**	**UNK**
$\boldsymbol{\epsilon = 0.1}$	0.554	0.295
$\boldsymbol{\epsilon = 0.05}$	0.554	0.295

Table 1: AER of IBM1 after ten iterations.

After a run of 10 iterations, the algorithm finished its training. Figures 1 and 2 show the evolution of perplexity and AER during the iterations. We see that both perplexity and AER are invariant under different values for ϵ, which makes sense since the perplexity and AER do not depend on ϵ when deriving the equations of these measures. From both figures it is clear that the model using the tags *UNK* is outperforming the model using k-smoothing.

In Figure 3 the lexicon parameter of the pairs house-maison and house-demain are shown in a plot starting after the first iteration. We can see that for the models with k-smoothing, the difference of the lexicon parameters between the two pairs stays small. However, the difference of the lexicon parameters between the two pairs for the model using the tag *UNK* is increasing.

IBM1: Variational Bayes. The training of IBM1VB took considerably longer, specially during the E-step. We speculate that this might be due to the overload of the operations followed by the calculations of $\theta^{\text{new}}_{f|e}$. Please find on Table 2 the AER values for certain iterations of the model. The AER values do decrease, as expected.

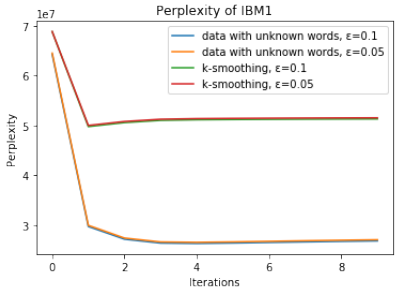

Figure 1: The perplexity of the four IBM1's.

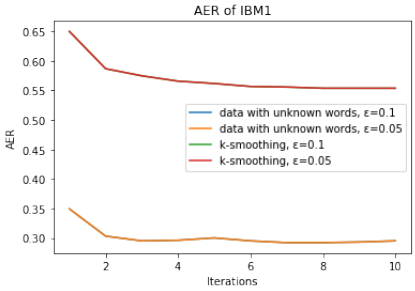

Figure 2: The AER of the four IBM1's.

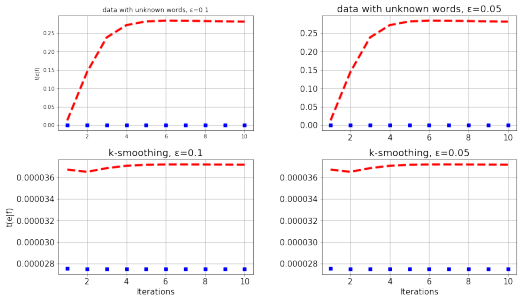

Figure 3: The lexicon parameters of $t(maison \mid house)$ in red and $t(demain \mid house)$, in blue.

	i=2	**i=5**
AER	0.552	0.484

Table 2: AER of IBM1VB after five iterations.

Given the expensive computational power demanded, the algorithm was set to run for five iterations only.

IBM2. Different initializations were implemented in IBM2; see Table 3 for the AER scores of the models. It is shown that the models using the pre-trained lexical parameters are outperform-

ing the other models. Also, the model using the uniform initializations are either outperforming or equal to those using random initialization. However, the difference is not significant (< 0.05).

	uniform A	**random A**
uniform/random t	0.291	0.299
IBM1 t	0.266	0.266

Table 3: AER of IBM2 after ten iterations.

In Figures 4 and 5, the perplexity and AER over the ten iterations of training the IBM2 models are shown. Both perplexity and AER from the models with a uniform initialization are performing similar to the models with a random initialization. The models using pre-trained lexical parameters are converging faster than the other models in both perplexity and AER. All models run for IBM2 had as hyperparameter $\epsilon = 0.1$ fixed. This was the chosen value from IBM1 given that no major discrepancies were found in comparison to a value of 0.05.

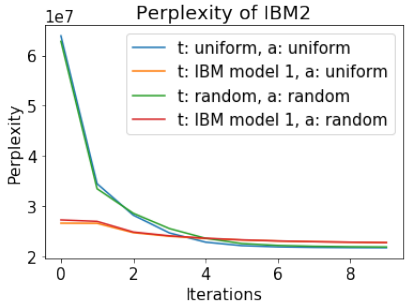

Figure 4: The perplexity of the four IBM2's.

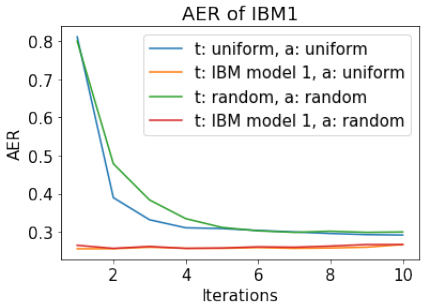

Figure 5: The AER of the four IBM2's.

Conclusion

In this study, IBM1 with EM algorithm, IBM1 with variational Bayes and IBM2 with EM algorithm were introduced. There were some experiments done for fine-tuning these models using evaluation measures, namely perplexity and AER. For IBM1, different values for ϵ, as well as the choice between k-smoothing and using the tag *UNK*, were explored. For IBM2, the following different initializations for t and A were implemented: random, uniform, and inputing the t from IBM1. In the case of IBM2VB, only one model was trained.

The choice of ϵ for IBM1 did not effect the perplexity and AER, hence the evaluation measures used in this study were not able to fine-tune ϵ. However, the use of the *UNK* word tag is clearly outperforming the models using k-smoothing.

For IBM2, there was no significant difference between all models. Nevertheless, the models using pre-trained lexicon parameters converged faster than the other models. Indeed, the choice of the initialization for IBM2 seems to be a key element in making the EM training converge. Within its implementation, using the output from IBM1 seems to be a wiser choice than either uniformly or randomly generating the initial probabilities. This might be due to the fact that once already-trained probabilities are fed into IBM2, this last algorithm explores a more defined region concentrated around the highest probabilities found so far, rendering lower perplexity and AER values. However, we believe that a proper initialization choice depends on the size of the bilingual corpora, given that feeding already-trained probabilities into the second model can be time-demanding, depending on the implementation. Moreover, the uniform initializations are much faster than the random initialization to create.

Comparing all models, we may observe there is no significant difference between the best performing IBM1 and the best performing IBM2 for ten iterations. However, training IBM1 is much faster so it is outperforming IBM2 in terms of time. The model IBM1VB is significant worser than IBM1 and IBM2 this may be due to training the model for less than ten iterations.

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* .

Robert C Moore. 2004. Improving IBM Word-Alignment Model 1. *In Proceedings of the ACL.* .

Philip Schulz. 2017. Bayesian IBM Model 1 .

Appendices

Code

Please find the code at: Github.

NAACL files

The following NAACL files are provided alongside the present paper:

1. IBM1 EM: $\epsilon = 0.1$ with *UNK* word tag
2. IBM1 VB: after one iteration
3. IBM2 EM: uniform A and pre-trained t