The IBM Mixture Models 1 and 2 for Word Alignment

Philip Schulz
P.Schulz@uva.nl

last modified: March 31, 2017

Abstract

This is a tutorial on the IBM models 1 and 2 for word alignment. In contrast to many other presentations, I motivate the models from a mixture model rather than from a translation perspective. This view makes it easier to derive the EM algorithms for learning and to understand why the likelihood function of the models usually has multiple optima.

1 Introduction

The IBM models for word alignment were introduced in Brown et al. (1993). They were originally designed to handle the translation task. However, after the advent of phrase-based machine translation models (Och and Ney, 2004; Koehn et al., 2003), they are now solely used for word alignment.

There are 5 IBM models. The first two are tractable, meaning that the inference tasks necessary for learning can be performed exactly in a reasonable (polynomial) amount of time. Inference in IBM models 3 to 5 is intractable and can therefore not be performed exactly. Moreover, IBM models 3 and 4 are deficient in the sense that they assign positive probability to events that are known to be impossible in naturally observed data. In statistical terms, these models are mis-specified. Here, we only present IBM models 1 and 2.

The general problem that the IBM models try to solve is to account for a collection of sentence-aligned text, known as a parallel corpus. Such a corpus is a collection of sentence pairs from two languages. The sentences that form a pair are known to be translations of each other. In line with the presentation in Brown et al. (1993), I will generically call the two languages in the corpus English and French.

To formalize the problem, we introduce random variables (RVs) E and F over the English and French vocabulary. There are in fact several such RVs, one for each word position in the corpus. We also assume that the sth English sentence contains s_l word positions and its French translation contains s_m word positions. Under the assumption that all sentence pairs are independent, a probabilistic model of a parallel corpus C with $|C|$ sentence pairs can then generally be formulated as

$$P(C) = \prod_{s=1}^{|C|} P(e_1^{s_l}, f_1^{s_m}) \; . \tag{1}$$

I will use the notation e_1^l throughout to denote a vector of outcomes $(e_1, \ldots, e_l)$. Furthermore, since the sentence pairs within C are generally assumed independent, I will limit the following exposition to one sentence pair only for the sake of clarity. In doing so, I will drop the index s for the sentence pair. Hence, the sentences are simply of lengths l and m.

The IBM models decompose the joint probability of a sentence pair according to the chain rule.

$$P(e_1^l, f_1^m) = \overbrace{P(e_1^l)}^{\text{language model}} \times \underbrace{P(f_1^m|e_1^l)}_{\text{translation model}} \tag{2}$$

The first factor in Equation (2) is called a language model as it models the monolingual probability of the English sentence. There are various language models available such as ngram models (Chen and Goodman, 1999), log-linear models (Rosenfeld, 1996) and neural network models (Bengio et al., 2003; Mikolov et al., 2010) to only name a few. In our presentation of the IBM models, we will stay agnostic as to what kind of language model we use. Any probabilistic language model will do. Our main focus is going to be on the second factor of Equation (2) which is the translation model.

The translation model assigns a probability to French sentences given an English sentence. The crucial observation of Brown et al. (1993) was that each French word will usually be translated into exactly one word in English. Of course, there are many counterexamples to this. However, assuming that each French word is generated from only one English word allows us to formulate a very local generative process. If we add in the assumption that all French words are conditionally independent (given the English sentence), we arrive at surprisingly simple graphical models. The models are so simple in fact that inference can be done exactly in them. This property turns out to be extremely helpful when learning the parameters of IBM models 1 and 2.

Before I describe these models and derive their learning algorithm, we quickly review mixture models. As it turns out, IBM models 1 and 2 are just slightly awkward *contstrained* mixture models.

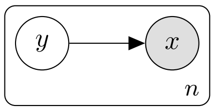

Figure 1: Graphical depiction of a general mixture model where the mixture components are assumed conditionally independent.

2 Mixture Models

A mixture model consists of k mixture components, each of which defines a distribution over the data space $\mathcal{X}$ which is the same for all components. The mixture model also defines a distribution over components. Since each component can potentially specialize its distribution on a subset of the data space, a mixture model can provide a tighter fit to the data than a single mixture component alone. The probability that a mixture model with k components assigns to n i.i.d. data point is

$$P(x_1^n) = \prod_{i=1}^{n} \sum_{j=1}^{k} P(x_i, y_i = j) = \prod_{i=1}^{n} \sum_{j=1}^{k} P(y_i = j) P(x_i | y_i = j) \ . \tag{3}$$

where we have introduced a set of random variables Y_i which range over the k mixture components.

It is common to interpret mixture models as missing data models. Concretely, we assume our data actually consists of pairs (x, y) where x is always observed and y is always missing. The inference task is then to infer a distribution over possible y_i for each x_i.

Non-identifiability It is well known that different parameter settings in a mixture model can lead to the same probability distribution. This is often called the labelling problem in machine learning. In statistics we say that the model is not identifiable. A model is said to be identifiable if each parameter setting gives rise to a different probability distribution. This is not the case for mixture models. Let us quickly illustrate why.

We assume a mixture model with two components c_1, c_2 which is used to model data from $\mathcal{X} = \{a, b\}$. We further set the distribution over components to $P(c_1) = P(c_2) = 0.5$. Let us assume that $P(a|c_1) = 0.2$ and $P(b|c_1) = 0.8$. Let us also assume that $P(a|c_2) = 0.7$ and $P(b|c_2) = 0.3$. It is then easy to see that no matter what our observed data actually looks like, we will assign it the same probability when using the component distributions just described or when we simply exchange them. By that I mean that we use component distributions $P(a|c_1) = 0.7$ and $P(b|c_1) = 0.3$ and $P(a|c_2) = 0.2$ and $P(b|c_2) = 0.8$. The probability of the data will be the same under both parameter settings.

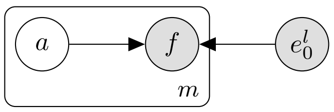

Figure 2: Graphical depiction of IBM models 1 and 2. They are both constrained mixture models. We do not show the French sentence length as a variable and rather take it as given since the model never learns parameters for it anyway.

This has an important consequence for the likelihood function of our model. Let us assume that it has a maximum at one of the two parameter configurations presented above. Then it will also have a maximum at the other one. This has major implications for any parameter estimation method based on the maximum likelihood principle. We will necessarily use the estimate from one of the two maxima, without knowing which one performs better empirically. If we use a hill-climbing parameter estimation method, as we are forced to do when doing maximum likelihood estimation in models with unobserved variables, the maximum that we arrive at in the end will crucially depend on the point at which we started the optimisation. We will see these problems come to bear shortly in the context of the IBM models.

3 IBM Model 1

Before we take a look at the actual IBM model 1, let us first formulate the generative process that underlies this model. Recall that we assume that we are given a language model. Thus, we assume that the English sentence has already been produced for us. Therefore, we only need to formulate a generative process for the French sentence conditioned on the English sentence. The process works as follows:

1. Choose the French sentence length m based on the English sentence length l.

2. For each French position j, choose the English position a_j that it is generated from.

3. For each French position j, choose a French word based on the English word in position a_j.

This generative story introduces a new variable A_j for each French position. That variable is nothing but an indicator for the mixture component that the French word in position j is generated from. The mixture components are English words. We are thus dealing with a mixture model, albeit a peculiar one. In normal mixture models, all mixture components can be assigned to a given data point. In the IBM models, only the mixture components (English words) that are present in the English sentence can be used. This drastically reduces the possible choice of mixture components for a given French word. We are thus dealing with what I will call a *constrained* mixture model, where the choice of mixture components is constrained by the English sentence. Compare Figures 1 and 2 to understand the difference graphically.

Another non-standard aspect of the IBM models is that A_j does not point to the mixture component directly but rather to the position in the English sentence that contains the mixture component. This also tells us something about the nature of the alignment variable. It is a variable that ranges over positions in the English sentence.

At this point we should clarify some terminology: The assignment of mixture components to French words for the entire French sentence is usually called an *alignment*. Each individual assignment for each French word is called an *alignment link*. Such an alignment link contains two pieces of information: the position of the French word (through the index j on the variable A_j) and the position of the English word (through the assignment $a_j = i$). We can thus alternatively view an alignment link as a pair (j, i) where j is the French position and i is the English position. The entire alignment can be written as a vector $(a_1, \ldots, a_m)$ (this is handy when implementing the IBM models).

The IBM models also introduce one slight modification to the English sentence. They add a hypothetical NULL word to each English sentence in position 0. Why would they do that? Well, consider the German sentence

$$Ich\ gehe\ nach\ Hause$$

and its English translation

$$I\ go\ home\ .$$

The word-to-word translations are I=Ich, go=gehe, home=Hause but there is no translation for the German preposition "nach". If the German sentence is to be generated from the English one, however, we need to assign some mixture component to "nach". Choosing an actual English word would be non-sensical since we just stated that "nach" does not have a translation in the English sentence. Brown et al. (1993) therefore introduced the NULL word[1]. It is a generic mixture component that generates untranslatable words.

With these preliminaries out of the way, let us actually formulate the IBM alignment models. Notice that they model a joint distribution of the French sentence length, an alignment (assignment of mixture components) and the choice of French words, all of which are conditioned on the English

[1]Notice that our language model does not need to worry about the NULL word. It is assumed to always be present in position 0 of every English sentence. Thus, we do not need to model it probabilistically. If we did model it probabilistically, we'd trivially have $P(e_0 = NULL) = 1$.

sentence. There are some important independence assumptions: first, we assume that the alignment links are conditionally independent given the French sentence length m and the English sentence. Second, we assume that the French words are conditionally independent given the English sentence and their alignment links. Finally, we assume that French sentence lengths are uniformly distributed, meaning that there is a constant probability for all possible French sentence lengths.

$$P(f_1^m, a_1^m, m|e_0^l) = P(m|e_0^l) \times \prod_{j=1}^{m} P(a_j|m, e_0^l) \times P(f_j|e_0^l, a_j) \tag{4}$$

$$= P(m|l) \times \prod_{j=1}^{m} P(a_j|m, l) \times P(f_j|e_{a_j}) \tag{5}$$

$$\propto \prod_{j=1}^{m} P(a_j|m, l) \times P(f_j|e_{a_j}) \tag{6}$$

The proportionality in Equation (6) follows from the fact that $P(m|l)$ is a constant. Notice also that f_j depends on the English sentence only through the English word it is aligned to (the word in position a_j, denoted by e_{a_j}). Furthermore, the alignment links depend on the English sentence only through its length l.

IBM1 is a particularly simple version of this general model. It assumes that $P(a_j|m, l)$ is uniform, i.e. constant, as well. Thus its probability mass function can be written as

$$P(f_1^m, a_1^m, m|e_0^l) \propto \prod_{j=1}^{m} P(f_j|e_{a_j}) \ . \tag{7}$$

4 Parameter Estimation for IBM1

We now turn to learning the parameters of IBM1. As we see in Equation 7, the only probability mass functions whose parameters we need to learn are those of the mixture components (the English words). The mixture weights $P(a_j|m, l)$ are assumed to be fixed uniformly. We will assume that each mixture component follows a categorical distribution. Thus, the probability that each mixture component e_i assigns to a sequence of French words is given by

$$P(f_1^m|e_i) = \prod_{j=1}^{m} P(f_j|e_i) = \prod_{j=1}^{m} \theta_{f_j|e_i} \tag{8}$$

where $\theta_{f_j|e_i}$ is simply the probability of word f_j under mixture component e_i. It is this probability that we want to estimate. We will do so using maximum likelihood estimation (MLE).

Recall that the maximum likelihood estimate for a conditional categorical distribution in the case of *fully observed data* is simply

$$\theta^{(MLE)}_{x|y} = \frac{\#xy}{\#y} \tag{9}$$

where $\#xy$ is the number of times we have seen outcome x paired with mixture component y in our data and $\#y$ is the number of times y was observed with any data point. Notice that $\#y = \sum_x \#xy$. Since $\#xy$ is the only information required from our data to arrive at the MLE, we call it a *sufficient statistic*.

Here is the trouble with mixture models: as I pointed out at the end of Section 2, mixture models are models of *partially* observed data. In particular, the mixture component responsible for each data point is hidden. However, to do MLE for the mixture component parameters, we need the sufficient statistics (counts in case of the categorical distribution). In particular, we need counts of how often each mixture component co-occurs with each observed outcome. Where do we get these from?

One popular idea in machine learning is to simply use the expected counts of mixture components under some auxiliary distribution. This is a way of completing the data. However, instead of making "hard" decisions about which mixture component to assign to which data point, we instead assign mixture components partially according to their probability under the auxiliary distribution. The MLE for that probabilistically completed data then is

$$\theta^{(MLE)}_{x|y} = \frac{E[\#xy]}{E[\#y]} \tag{10}$$

where the expectation is taken with respect to the auxiliary distribution. This of course begs the question of what a "good" auxiliary distribution is. It can be shown (Neal and Hinton, 1999) that the best possible auxiliary distribution under this framework is $P(y|x)$, the posterior distribution of the hidden variables.

This general procedure can be broken down into two steps. First, there is the E(xpectation) step in which we compute the expected sufficient statistics (the numerator of Equation (10)). Second, there is the M(aximisation) step in which we find a maximum likelihood estimate based on the expected statistics found in the E-step. This procedure is called the Expectation-Maximisation (EM) algorithm. If applied repeatedly, it is guaranteed to find a local maximum in the likelihood function.

EM for IBM1 Let us now apply the EM algorithm to parameter estimation in IBM1. First, we need to compute the expected sufficient statistics. To do so, we need to find the posterior distribution over mixture components, which for French word j is

$$P(a_j = i|f_j, e_0^l) = P(a_j = i|f_j, e_i) = \frac{P(f_j|e_i)}{\sum_{i=0}^{l} P(f_j|e_i)} \ . \tag{11}$$

This may be slightly confusing. After all, the mixture components are English words but we are computing a distribution over English positions. Notice however, that each English position is uniquely associated with the English word in that position. Thus, we are in fact computing a distribution over *available* mixture components.

Next, we need to compute the expected co-occurrence count for an English word e and a french word f. To do so, we simply sum over all French word positions[2].

$$E[\#ef] = \sum_{j=1}^{m} P(a_j = i|f_j = f, e_i = e) \tag{12}$$

This completes the E-step. We now need to perform the M-step. This is straightforwardly done by filling substituting f for x and e for y in Equation (10).

Issues with non-identifiability In the case of IBM1, where the mixture weights are fixed and uniform, EM is guaranteed to arrive at a global maximum. However, as pointed out in Section 2, there may be many global maxima. For example, assume that the compound “ice cream” occurs several times in a parallel corpus. Assume further that neither of the two words occurs without the other. If we get MLE parameters from EM, we are guaranteed to get another MLE estimate (not found by EM) by simply exchanging the component distributions of “ice” and “cream”. Which of the several possible MLEs the EM algorithm finds depends on the starting parameters we provide it with. In practice, one usually starts from uniform parameters. Note, however, that Toutanova and Galley (2011) have shown that other initialisations may be better.

5 IBM Model 2

IBM2 is an extension of IBM1 that also learns the mixture weights. That means that the weights are not fixed and uniform anymore during training. We thus need a new likelihood function for this model. We will take the

[2]The reason we can do this is because French words in different positions together with their generating mixture components are assumed to be independent. This means that the posterior factorises in the same way as the likelihood in Equation 8. Notice also that if we are dealing with a full parallel corpus, the summation is done over all French positions in the corpus.

one from Equation (6). We are going to model the probability of an alignment link as the conditional probability of generating French position j from English position i.

$$P(a_j = i|m, l) = P(i|j, m, l) \tag{13}$$

Other formulations are possible but this is the one originally chosen by Brown et al. (1993). It is crucial that we condition on m and l in order to know which positions are available in the first place. The full probability mass function for IBM2 is

$$P(f_1^m, a_1^m, m|e_0^l) \propto \prod_{j=1}^{m} P(i|j, m, l) \times P(f_j|e_i) \tag{14}$$

where we have again dropped the constant term for the length probability.

6 Parameter Estimation for IBM2

The posterior for IBM2 simply contains an additional factor that stems from the non-uniform mixture weights introduced in Equation (14).

$$P(a_j = i|f_j, e_0^l, m) = P(a_j = i|f_j, e_i, m, l) = \frac{P(i|j, m, l) \times P(f_j|e_i)}{\sum_{i=0}^{l} P(i|j, m, l) \times P(f_j|e_i)} \tag{15}$$

In addition to the expected sufficient statistics for the lexical translations, we now also need expected sufficient statistics for the alignment parameters. We condition these expectations on the sentence lengths because the alignment parameters vary with these lengths. For each sentence pair, those expected sufficient statistics then are

$$E[\#ij|m, l] = P(a_j = i|f_j, e_i, m, l) \ . \tag{16}$$

As with lexical translations, we need to sum these expected sufficient statistics over the entire parallel corpus if we are dealing with more than one sentence.

Issues with non-identifiability As we have seen with IBM1, there may be several maxima in the likelihood function of the IBM models. This problem escalates in IBM2. First of all, the mixture weights are not fixed anymore, adding several new parameters to the model. Second, because of the asymmetric mixture weights, most maxima are now local. As in IBM model 1, exchanging the component distributions of "ice" and "cream" will again yield a maximum in the likelihood function. However, because the mixture weights are not uniform, one of the two maxima will have a lower likelihood value than the other. Moreover, neither of them is guaranteed to be a global maximum. This is because the model needs to find a good balance between the mixture weights (which are independent of lexical content and only depend on positions) and the component distributions (which are entirely lexical). Changing the weights may induce a change in the component distributions and the other way around. This can lead to complex interactions between the model parameters.

An additional worry is that EM is only guaranteed to find a *local* maximum which may be far from globally optimal. In practice, one initialises the component distributions of IBM2 (i.e. its translation parameters) with IBM1 estimates. The alignment distributions are initialised uniformly. Notice that this implies that we first have to train IBM1 before proceeding to training IBM2.

7 Decoding

Recall that we do not use the IBM models for translation but only for word alignment. Thus, after having trained the models, we need to decide on one specific alignment (this process is usually referred to as *decoding*). We will pick the alignment that has the highest posterior probability. This is often called the Viterbi alignment. The posterior probabilities for alignment links are given in Equations (11) and (15) for IBM models 1 and 2, respectively.

Notice that because we have assumed conditional independence of the alignment links, the problem of maximising the probability of an alignment conveniently factorises over the individual alignment links. In other words: to find the best alignment for an entire sentence we only need to find the best alignment link for each French word. This can be formalised as

$$\max_{a_1^m} P(a_1^m|f_1^m, e_0^l) = \prod_{j=1}^{m} \max_{a_j} P(a_j|f_j, e_{a_j}, m, l) \ . \tag{17}$$

Outputting the final alignment is achieved by returning

$$\arg\max_{a_1^m} P(a_1^m|f_1^m, e_0^l) \ .$$

Finally, recall that there is a NULL word that produces words that do not have a lexical translation. Whenever the NULL word (or, equivalently, the 0^{th} position) is the most probable component, we leave the French word in question unaligned.

8 Further Reading

The original text by (Brown et al., 1993) is fairly long and involved and you should probably read it only after you have completed your first course on machine translation. Michael Collins offers an excellent tutorial on his website that focuses on the algorithmic description of the EM learning algorithm for the IBM models and also provides some pseudo-code. You'll definitely want to look at this before you start implementing the models yourself.

References

Yoshua Bengio, Rjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

Peter F. Brown, Vincent J.Della Pietra, Stephen A. Della Pietra, and Robert. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19:263–311, 1993.

Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4): 359–393, 1999.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics, 2003.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, 2010.

Radford M. Neal and Geoffrey E. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In Michael I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. MIT Press, 1999.

Franz Josef Och and Hermann Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449, 2004. URL http://dx.doi.org/10.1162/0891201042544884.

Ronald Rosenfeld. A maximum entropy approach to adaptive statistical language modeling. *Computer, Speech and Language*, 10:187–228, 1996.

Kristina Toutanova and Michel Galley. Why initialization matters for ibm model 1: Multiple optima and non-strict convexity. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 461–466. Association for Computational Linguistics, 2011. URL http://aclweb.org/anthology/P11-2081.