Statistical Machine Translation: IBM Models 1 and 2

Michael Collins

1 Introduction

The next few lectures of the course will be focused on machine translation, and in particular on *statistical machine translation* (SMT) systems. In this note we will focus on the *IBM translation models*, which go back to the late 1980s/early 1990s. These models were seminal, and form the basis for many SMT models now used today.

Following convention, we'll assume throughout this note that the task is to translate from *French* (the "source" language) into *English* (the "target" language). In general we will use f to refer to a sentence in French: f is a sequence of words $f_1, f_2 \dots f_m$ where m is the length of the sentence, and f_j for $j \in \{1 \dots m\}$ is the j'th word in the sentence. We will use e to refer to an English sentence: e is equal to $e_1, e_2 \dots e_l$ where l is the length of the English sentence.

In SMT systems, we assume that we have a source of example translations, $(f^{(k)}, e^{(k)})$ for $k = 1 \dots n$, where $f^{(k)}$ is the k'th French sentence in the training examples, $e^{(k)}$ is the k'th English sentence, and $e^{(k)}$ is a translation of $f^{(k)}$. Each $f^{(k)}$ is equal to $f_1^{(k)} \dots f_{m_k}^{(k)}$ where m_k is the length of the k'th French sentence. Each $e^{(k)}$ is equal to $e_1^{(k)} \dots e_{l_k}^{(k)}$ where l_k is the length of the k'th English sentence. We will estimate the parameters of our model from these training examples.

So where do we get the training examples from? It turns out that quite large corpora of translation examples are available for many language pairs. The original IBM work, which was in fact focused on translation from French to English, made use of the Canadian parliamentary proceedings (the *Hansards*): the corpus they used consisted of several million translated sentences. The *Europarl data* consists of proceedings from the European parliament, and consists of translations between several European languages. Other substantial corpora exist for Arabic-English, Chinese-English, and so on.

2 The Noisy-Channel Approach

A few lectures ago we introduced generative models, and in particular the noisy-channel approach. The IBM models are an instance of a noisy-channel model, and as such they have two components:

1. A *language model* that assigns a probability $p(e)$ for any sentence $e = e_1 \dots e_l$ in English. We will, for example, use a trigram language model for this part of the model. The parameters of the language model can potentially be estimated from very large quantities of English data.

2. A *translation model* that assigns a conditional probability $p(f|e)$ to any French/English pair of sentences. The parameters of this model will be estimated from the translation examples. The model involves two choices, both of which are conditioned on the English sentence $e_1 \dots e_l$: first, a choice of the length, m, of the French sentence; second, a choice of the m words $f_1 \dots f_m$.

Given these two components of the model, following the usual method in the noisy-channel approach, the output of the translation model on a new French sentence f is:

$$e^* = \arg\max_{e \in E} \quad p(e) \times p(f|e)$$

where E is the set of all sentences in English. Thus the score for a potential translation e is the product of two scores: first, the language-model score $p(e)$, which gives a prior distribution over which sentences are likely in English; second, the translation-model score $p(f|e)$, which indicates how likely we are to see the French sentence f as a translation of e.

Note that, as is usual in noisy-channel models, the model $p(f|e)$ appears to be "backwards": even though we are building a model for translation from French into English, we have a model of $p(f|e)$. The noisy-channel approach has used Bayes rule:

$$p(e|f) = \frac{p(e)p(f|e)}{\sum_f p(e)p(f|e)}$$

hence

$$\begin{aligned} \arg\max_{e \in E} p(e|f) &= \arg\max_{e \in E} \frac{p(e)p(f|e)}{\sum_f p(e)p(f|e)} \\ &= \arg\max_{e \in E} p(e)p(f|e) \end{aligned}$$

A major benefit of the noisy-channel approach is that it allows us to use a language model $p(e)$. This can be very useful in improving the fluency or grammaticality of the translation model's output.

The remainder of this note will be focused on the following questions:

- How can we define the translation model $p(f|e)$?
- How can we estimate the parameters of the translation model from the training examples $(f^{(k)}, e^{(k)})$ for $k = 1 \dots n$?

We will describe the IBM models—specifically, IBM models 1 and 2—for this problem. The IBM models were an early approach to SMT, and are now not widely used for translation: improved models (which we will cover in the next lecture) have been derived in recent work. However, they will be very useful to us, for the following reasons:

1. The models make direct use of the idea of *alignments*, and as a consequence allow us to recover alignments between French and English words in the training data. The resulting alignment models are of central importance in modern SMT systems.
2. The parameters of the IBM models will be estimated using the expectation maximization (EM) algorithm. The EM algorithm is widely used in statistical models for NLP and other problem domains. We will study it extensively later in the course: we use the IBM models, described here, as our first example of the algorithm.

3 The IBM Models

3.1 Alignments

We now turn to the problem of modeling the conditional probability $p(f|e)$ for any French sentence $f = f_1 \dots f_m$ paired with an English sentence $e = e_1 \dots e_l$.

Recall that $p(f|e)$ involves two choices: first, a choice of the length m of the French sentence; second, a choice of the words $f_1 \dots f_m$. We will assume that there is some distribution $p(m|l)$ that models the conditional distribution of French sentence length conditioned on the English sentence length. From now on, we take the length m to be fixed, and our focus will be on modeling the distribution

$$p(f_1 \dots f_m | e_1 \dots e_l, m)$$

i.e., the conditional probability of the words $f_1 \dots f_m$, conditioned on the English string $e_1 \dots e_l$, and the French length m.

It is very difficult to model $p(f_1 \dots f_m | e_1 \dots e_l, m)$ directly. A central idea in the IBM models was to introduce additional alignment variables to the problem. We will have alignment variables $a_1 \dots a_m$—that is, one alignment variable for each French word in the sentence—where each alignment variable can take any value in $\{0, 1, \dots, l\}$. The alignment variables will specify an alignment for each French word to some word in the English sentence.

Rather than attempting to define $p(f_1 \dots f_m | e_1 \dots e_l, m)$ directly, we will instead define a conditional distribution

$$p(f_1 \dots f_m, a_l \dots a_m | e_1 \dots e_l, m)$$

over French sequences $f_1 \dots f_m$ together with alignment variables $a_1 \dots a_m$. Having defined this model, we can then calculate $p(f_1 \dots f_m | e_1 \dots e_l, m)$ by summing over the alignment variables ("marginalizing out" the alignment variables):

$$p(f_1 \dots f_m | e_1 \dots e_l) = \sum_{a_1=0}^{l} \sum_{a_2=0}^{l} \sum_{a_3=0}^{l} \dots \sum_{a_m=0}^{l} p(f_1 \dots f_m, a_1 \dots a_m | e_1 \dots e_l)$$

We now describe the alignment variables in detail. Each alignment variable a_j specifies that the French word f_j is aligned to the English word e_{a_j}: we will see soon that intuitively, in the probabilistic model, word f_j will be generated from English word e_{a_j}. We define e_0 to be a special `NULL` word; so $a_j = 0$ specifies that word f_j is generated from the `NULL` word. We will see the role that the `NULL` symbol plays when we describe the probabilistic model.

As one example, consider a case where $l = 6$, $m = 7$, and

$e =$ *And the programme has been implemented*

$f =$ *Le programme a ete mis en application*

In this case the length of the French sentence, m, is equal to 7; hence we have alignment variables $a_1, a_2, \dots a_7$. As one alignment (which is quite plausible), we could have

$$a_1, a_2, \dots, a_7 = \langle 2, 3, 4, 5, 6, 6, 6 \rangle$$

specifying the following alignment:

Le	$\Rightarrow$	the
Programme	$\Rightarrow$	program
a	$\Rightarrow$	has
ete	$\Rightarrow$	been
mis	$\Rightarrow$	implemented
en	$\Rightarrow$	implemented
application	$\Rightarrow$	implemented

Note that each French word is aligned to exactly one English word. The alignment is many-to-one: more than one French word can be aligned to a single English word (e.g., *mis*, *en*, and *application* are all aligned to *implemented*). Some English words may be aligned to zero French words: for example the word *And* is not aligned to any French word in this example.

Note also that the model is asymmetric, in that there is no constraint that each English word is aligned to exactly one French word: each English word can be aligned to any number (zero or more) French words. We will return to this point later.

As another example alignment, we could have

$$a_1, a_2, \ldots a_7 = \langle 1, 1, 1, 1, 1, 1, 1 \rangle$$

specifying the following alignment:

Le	$\Rightarrow$	And
Programme	$\Rightarrow$	And
a	$\Rightarrow$	And
ete	$\Rightarrow$	And
mis	$\Rightarrow$	And
en	$\Rightarrow$	And
application	$\Rightarrow$	And

This is clearly not a good alignment for this example.

3.2 Alignment Models: IBM Model 2

We now describe a model for the conditional probability

$$p(f_1 \ldots f_m, a_1 \ldots a_m | e_1 \ldots e_l, m)$$

The model we describe is usually referred to as IBM model 2: we will use *IBM-M2* as shorthand for this model. Later we will describe how IBM model 1 is a special case of IBM model 2. The definition is as follows:

Definition 1 (IBM Model 2) *An IBM-M2 model consists of a finite set $\mathcal{E}$ of English words, a set $\mathcal{F}$ of French words, and integers M and L specifying the maximum length of French and English sentences respectively. The parameters of the model are as follows:*

- *$t(f|e)$ for any $f \in \mathcal{F}$, $e \in \mathcal{E} \cup \{NULL\}$. The parameter $t(f|e)$ can be interpreted as the conditional probability of generating French word f from English word e.*
- *$q(j|i,l,m)$ for any $l \in \{1 \ldots L\}$, $m \in \{1 \ldots M\}$, $i \in \{1 \ldots m\}$, $j \in \{0 \ldots l\}$. The parameter $q(j|i,l,m)$ can be interpreted as the probability of alignment variable a_i taking the value j, conditioned on the lengths l and m of the English and French sentences.*

Given these definitions, for any English sentence $e_1 \ldots e_l$ where each $e_j \in \mathcal{E}$, for each length m, we define the conditional distribution over French sentences $f_1 \ldots f_m$ and alignments $a_1 \ldots a_m$ as

$$p(f_1 \ldots f_m, a_1 \ldots a_m | e_1 \ldots e_l, m) = \prod_{i=1}^{m} q(a_i|i,l,m)t(f_i|e_{a_i})$$

Here we define e_0 to be the `NULL` *word.* □

To illustrate this definition, consider the previous example where $l = 6$, $m = 7$,

$e =$ *And the programme has been implemented*

$f =$ *Le programme a ete mis en application*

and the alignment variables are

$$a_1, a_2, \ldots a_7 = \langle 2, 3, 4, 5, 6, 6, 6 \rangle$$

specifying the following alignment:

Le	$\Rightarrow$	the
Programme	$\Rightarrow$	program
a	$\Rightarrow$	has
ete	$\Rightarrow$	been
mis	$\Rightarrow$	implemented
en	$\Rightarrow$	implemented
application	$\Rightarrow$	implemented

In this case we have

$$\begin{aligned} & p(f_1 \ldots f_m, a_1 \ldots a_m | e_1 \ldots e_l, m) \\ = \; & q(2|1,6,7) \times t(\textit{Le}|\textit{the}) \\ & \times q(3|2,6,7) \times t(\textit{Programme}|\textit{program}) \\ & \times q(4|3,6,7) \times t(\textit{a}|\textit{has}) \\ & \times q(5|4,6,7) \times t(\textit{ete}|\textit{been}) \\ & \times q(6|5,6,7) \times t(\textit{mis}|\textit{implemented}) \\ & \times q(6|6,6,7) \times t(\textit{en}|\textit{implemented}) \\ & \times q(6|7,6,7) \times t(\textit{application}|\textit{implemented}) \end{aligned}$$

Thus each French word has two associated terms: first, a choice of alignment variable, specifying which English word the word is aligned to; and second, a choice of the French word itself, based on the English word that was chosen in step 1. For example, for $f_5 = \textit{mis}$ we first choose $a_5 = 6$, with probability $q(6|5,6,7)$, and then choose the word *mis*, based on the English word $e_6 = \textit{implemented}$, with probability $t(\textit{mis}|\textit{implemented})$.

Note that the alignment parameters, $q(j|i,l,m)$ specify a different distribution $\langle q(0|i,l,m), q(1|i,l,m), \ldots, q(l|i,l,m)\rangle$ for each possible value of the tuple i,l,m, where i is the position in the French sentence, l is the length of the English sentence, and m is the length of the French sentence. This will allow us, for example, to capture the tendency for words close to the beginning of the French sentence to be translations of words close to the beginning of the English sentence.

The model is certainly rather simple and naive. However, it captures some important aspects of the data.

3.3 Independence Assumptions in IBM Model 2

We now consider the independence assumptions underlying IBM Model 2. Take L to be a random variable corresponding to the length of the English sentence; $E_1 \ldots E_l$ to be a sequence of random variables corresponding to the words in the English sentence; M to be a random variable corresponding to the length of the French sentence; and $F_1 \ldots F_m$ and $A_1 \ldots A_m$ to be sequences of French words, and alignment variables. Our goal is to build a model of

$$P(F_1 = f_1 \ldots F_m = f_m, A_1 = a_1 \ldots A_m = a_m | E_1 = e_1 \ldots E_l = e_l, L = l, M = m)$$

As a first step, we can use the chain rule of probabilities to decompose this into two terms:

$$P(F_1 = f_1 \ldots F_m = f_m, A_1 = a_1 \ldots A_m = a_m | E_1 = e_1 \ldots E_l = e_l, L = l, M =$$

$$= P(A_1 = a_1 \ldots A_m = a_m | E_1 = e_1 \ldots E_l = e_l, L = l, M = m)$$
$$\times P(F_1 = f_1 \ldots F_m = f_m | A_1 = a_1 \ldots A_m = a_m, E_1 = e_1 \ldots E_l = e_l, L = l, M$$

We'll now consider these two terms separately.

First, we make the following independence assumptions:

$$P(A_1 = a_1 \ldots A_m = a_m | E_1 = e_1 \ldots E_l = e_l, L = l, M = m)$$
$$= \prod_{i=1}^{m} P(A_i = a_i | A_1 = a_1 \ldots A_{i-1} = a_{i-1}, E_1 = e_1 \ldots E_l = e_l, L = l, M = m)$$
$$= \prod_{i=1}^{m} P(A_i = a_i | L = l, M = m)$$

The first equality is exact, by the chain rule of probabilities. The second equality corresponds to a very strong independence assumption: namely, that the distribution of the random variable A_i depends only on the values for the random variables L and M (it is independent of the English words $E_1 \ldots E_l$, and of the other alignment variables). Finally, we make the assumption that

$$P(A_i = a_i | L = l, M = m) = q(a_i | i, l, m)$$

where each $q(a_i | i, l, m)$ is a parameter of our model.

Next, we make the following assumption:

$$P(F_1 = f_1 \ldots F_m = f_m | A_1 = a_1 \ldots A_m = a_m, E_1 = e_1 \ldots E_l = e_l, L = l, M =$$
$$= \prod_{i=1}^{m} P(F_i = f_i | F_1 = f_1 \ldots F_{i-1} = f_{i-1}, A_1 = a_1 \ldots A_m = a_m, E_1 = e_1 \ldots E_l =$$
$$= \prod_{i=1}^{m} P(F_i = f_i | E_{a_i} = e_{a_i})$$

The first step is again exact, by the chain rule. In the second step, we assume that the value for F_i depends only on E_{a_i}: i.e., on the identity of the English word to which F_i is aligned. Finally, we make the assumption that for all i,

$$P(F_i = f_i | E_{a_i} = e_{a_i}) = t(f_i | e_{a_i})$$

where each $t(f_i | e_{a_i})$ is a parameter of our model.

4 Applying IBM Model 2

The next section describes a parameter estimation algorithm for IBM Model 2. Before getting to this, we first consider an important question: what is IBM Model 2 useful for?

The original motivation was the full machine translation problem. Once we have estimated parameters $q(j|i,l,m)$ and $t(f|e)$ from data, we have a distribution

$$p(f,a|e)$$

for any French sentence f, alignment sequence a, and English sentence e; from this we can derive a distribution

$$p(f|e) = \sum_a p(f,a|e)$$

Finally, assuming we have a language model $p(e)$, we can define the translation of any French sentence f to be

$$\arg\max_e p(e)p(f|e) \tag{1}$$

where the $\arg\max$ is taken over all possible English sentences. The problem of finding the $\arg\max$ in Eq. 1 is often referred to as the *decoding problem*. Solving the decoding problem is a computationally very hard problem, but various approximate methods have been derived.

In reality, however, IBM Model 2 is not a particularly good translation model. In later lectures we'll see alternative, state-of-the-art, models that are far more effective.

The IBM models are, however, still crucial in modern translation systems, for two reasons:

1. The lexical probabilities $t(f|e)$ are directly used in various translation systems.

2. Most importantly, the alignments derived using IBM models are of direct use in building modern translation systems.

Let's consider the second point in more detail. Assume that we have estimated our parameters $t(f|e)$ and $q(j|i,l,m)$ from a training corpus (using the parameter estimation algorithm described in the next section). Given any training example consisting of an English sentence e paired with a French sentence f, we can then find the most probably alignment under the model:

$$\arg\max_{a_1 \ldots a_m} p(a_1 \ldots a_m | f_1 \ldots f_m, e_1 \ldots e_l, m) \tag{2}$$

Because the model takes such as simple form, finding the solution to Eq. 2 is straightforward. In fact, a simple derivation shows that we simply define

$$a_i = \arg\max_{j \in \{0 \ldots l\}} \left(q(j|i,l,m) \times t(f_i|e_j) \right)$$

for $i = 1 \ldots m$. So for each French word i, we simply align it to the English position j which maximizes the product of two terms: first, the alignment probability $q(j|i, l, m)$; and second, the translation probability $t(f_i|e_j)$.

5 Parameter Estimation

This section describes methods for estimating the $t(f|e)$ parameters and $q(j|i, l, m)$ parameters from translation data. We consider two scenarios: first, estimation with *fully observed data*; and second, estimation with *partially observed data*. The first scenario is unrealistic, but will be a useful warm-up before we get to the second, more realistic case.

5.1 Parameter Estimation with Fully Observed Data

We now turn to the following problem: how do we estimate the parameters $t(f|e)$ and $q(j|i, l, m)$ of the model? We will assume that we have a training corpus $\{f^{(k)}, e^{(k)}\}_{k=1}^{n}$ of translations. Note however, that a crucial piece of information is missing in this data: *we do not know the underlying alignment for each training example*. In this sense we will refer to the data being only *partially observed*, because some information—i.e., the alignment for each sentence—is missing. Because of this, we will often refer to the alignment variables as being *hidden variables*. In spite of the presence of hidden variables, we will see that we can in fact estimate the parameters of the model.

Note that we could presumably employ humans to annotate data with underlying alignments (in a similar way to employing humans to annotate underlying parse trees, to form a treebank resource). However, we wish to avoid this because manual annotation of alignments would be an expensive task, taking a great deal of time for reasonable size translation corpora—moreover, each time we collect a new corpus, we would have to annotate it in this way.

In this section, as a warm-up for the case of partially-observed data, we will consider the case of *fully-observed data*, where each training example does in fact consist of a triple $(f^{(k)}, e^{(k)}, a^{(k)})$ where $f^{(k)} = f_1^{(k)} \ldots f_{m_k}^{(k)}$ is a French sentence, $e^{(k)} = e_1^{(k)} \ldots e_{l_k}^{(k)}$ is an English sentence, and $a^{(k)} = a_1^{(k)} \ldots a_{m_k}^{(k)}$ is a sequence of alignment variables. Solving this case will be be useful in developing the algorithm for partially-observed data.

The estimates for fully-observed data are simple to derive. Define $c(e, f)$ to be the number of times word e is aligned to word f in the training data, and $c(e)$ to be the number of times that e is aligned to *any* French word. In addition, define $c(j|i, l, m)$ to be the number of times we see an English sentence of length

l, and a French sentence of length m, where word i in French is aligned to word j in English. Finally, define $c(i, l, m)$ to be the number of times we see an English sentence of length l together with a French sentence of length m. Then the maximum-likelihood estimates are

$$t_{ML}(f|e) = \frac{c(e, f)}{c(e)}$$

$$q_{ML}(j|i, l, m) = \frac{c(j|i, l, m)}{c(i, l, m)}$$

Thus to estimate parameters we simply compile counts from the training corpus, then take ratios of these counts.

Figure 1 shows an algorithm for parameter estimation with fully observed data. The algorithm for partially-observed data will be a direct modification of this algorithm. The algorithm considers all possible French/English word pairs in the corpus, which could be aligned: i.e., all possible (k, i, j) tuples where $k \in \{1 \dots n\}$, $i \in \{1 \dots m_k\}$, and $j \in \{0 \dots l_k\}$. For each such pair of words, we have $a_i^{(k)} = j$ if the two words are aligned. In this case we increment the relevant $c(e, f)$, $c(e)$, $c(j|i, l, m)$ and $c(i, l, m)$ counts. If $a_i^{(k)} \neq j$ then the two words are not aligned, and no counts are incremented.

5.2 Parameter Estimation with Partially Observed Data

We now consider the case of partially-observed data, where the alignment variables $a^{(k)}$ are not observed in the training corpus. The algorithm for this case is shown in figure 2. There are two important differences for this algorithm from the algorithm in figure 1:

- The algorithm is iterative. We begin with some initial value for the t and q parameters: for example, we might initialize them to random values. At each iteration we first compile some "counts" $c(e)$, $c(e, f)$, $c(j|i, l, m)$ and $c(i, l, m)$ based on the data together with our current estimates of the parameters. We then re-estimate the parameters using these counts, and iterate.

- The counts are calculated using a similar definition to that in figure 1, but with one crucial difference: rather than defining

  $$\delta(k, i, j) = 1 \text{ if } a_i^{(k)} = j, 0 \text{ otherwise}$$

  we use the definition

  $$\delta(k, i, j) = \frac{q(j|i, l_k, m_k) t(f_i^{(k)}|e_j^{(k)})}{\sum_{j=0}^{l_k} q(j|i, l_k, m_k) t(f_i^{(k)}|e_j^{(k)})}$$

Input: A training corpus $(f^{(k)}, e^{(k)}, a^{(k)})$ for $k = 1 \ldots n$, where $f^{(k)} = f_1^{(k)} \ldots f_{m_k}^{(k)}$, $e^{(k)} = e_1^{(k)} \ldots e_{l_k}^{(k)}$, $a^{(k)} = a_1^{(k)} \ldots a_{m_k}^{(k)}$.

Algorithm:

- Set all counts $c(\ldots) = 0$
- For $k = 1 \ldots n$
  - For $i = 1 \ldots m_k$
    - For $j = 0 \ldots l_k$

$$
\begin{aligned}
c(e_j^{(k)}, f_i^{(k)}) &\leftarrow c(e_j^{(k)}, f_i^{(k)}) + \delta(k, i, j) \\
c(e_j^{(k)}) &\leftarrow c(e_j^{(k)}) + \delta(k, i, j) \\
c(j|i, l, m) &\leftarrow c(j|i, l, m) + \delta(k, i, j) \\
c(i, l, m) &\leftarrow c(i, l, m) + \delta(k, i, j)
\end{aligned}
$$

where $\delta(k, i, j) = 1$ if $a_i^{(k)} = j$, 0 otherwise.

Output:

$$t_{ML}(f|e) = \frac{c(e, f)}{c(e)} \qquad q_{ML}(j|i, l, m) = \frac{c(j|i, l, m)}{c(i, l, m)}$$

Figure 1: The parameter estimation algorithm for IBM model 2, for the case of fully-observed data.

Input: A training corpus $(f^{(k)}, e^{(k)})$ for $k = 1 \ldots n$, where $f^{(k)} = f_1^{(k)} \ldots f_{m_k}^{(k)}$, $e^{(k)} = e_1^{(k)} \ldots e_{l_k}^{(k)}$.

Initialization: Initialize $t(f|e)$ and $q(j|i,l,m)$ parameters (e.g., to random values).

Algorithm:

- For $s = 1 \ldots S$
  - Set all counts $c(\ldots) = 0$
  - For $k = 1 \ldots n$
    - For $i = 1 \ldots m_k$
      - For $j = 0 \ldots l_k$

$$
\begin{aligned}
c(e_j^{(k)}, f_i^{(k)}) &\leftarrow c(e_j^{(k)}, f_i^{(k)}) + \delta(k,i,j) \\
c(e_j^{(k)}) &\leftarrow c(e_j^{(k)}) + \delta(k,i,j) \\
c(j|i,l,m) &\leftarrow c(j|i,l,m) + \delta(k,i,j) \\
c(i,l,m) &\leftarrow c(i,l,m) + \delta(k,i,j)
\end{aligned}
$$

where

$$
\delta(k,i,j) = \frac{q(j|i,l_k,m_k)t(f_i^{(k)}|e_j^{(k)})}{\sum_{j=0}^{l_k} q(j|i,l_k,m_k)t(f_i^{(k)}|e_j^{(k)})}
$$

  - Set

$$
t(f|e) = \frac{c(e,f)}{c(e)} \qquad q(j|i,l,m) = \frac{c(j|i,l,m)}{c(i,l,m)}
$$

Output: parameters $t(f|e)$ and $q(j|i,l,m)$

Figure 2: The parameter estimation algorithm for IBM model 2, for the case of partially-observed data.

where the q and t values are our current parameter estimates.

Let's consider this last definition in more detail. We can in fact show the following identity:

$$P(A_i = j | e_1 \dots e_l, f_1 \dots f_m, m) = \frac{q(j|i,l,m)t(f_i|e_j)}{\sum_{j=0}^{l_k} q(j|i,l,m)t(f_i|e_j)}$$

where $P(A_i = j | e_1 \dots e_l, f_1 \dots f_m, m)$ is the conditional probability of the alignment variable a_i taking the value j, under the current model parameters. Thus we have effectively filled in the alignment variables probabilistically, using our current parameter estimates. This in contrast to the fully observed case, where we could simply define $\delta(k, i, j) = 1$ if $a_i^{(k)} = j$, and 0 otherwise.

As an example, consider our previous example where $l = 6$, $m = 7$, and

$e^{(k)} =$ *And the programme has been implemented*

$f^{(k)} =$ *Le programme a ete mis en application*

The value for $\delta(k, 5, 6)$ for this example would be the current model's estimate of the probability of word f_5 being aligned to word e_6 in the data. It would be calculated as

$$\delta(k, 5, 6) = \frac{q(6|5,6,7) \times t(\textit{mis}|\textit{implemented})}{\sum_{j=0}^{6} q(j|5,6,7) \times t(\textit{mis}|e_j)}$$

Thus the numerator takes into account the translation parameter $t(\textit{mis}|\textit{implemented})$ together with the alignment parameter $q(6|5,6,7)$; the denominator involves a sum over terms, where we consider each English word in turn.

The algorithm in figure 2 is an instance of the *expectation-maximization* (EM) algorithm. The EM algorithm is very widely used for parameter estimation in the case of partially-observed data. The counts $c(e)$, $c(e, f)$ and so on are referred to as *expected counts*, because they are effectively expected counts under the distribution

$$p(a_1 \dots a_m | f_1 \dots f_m, e_1 \dots e_l, m)$$

defined by the model. In the first step of each iteration we calculate the expected counts under the model. In the second step we use these expected counts to re-estimate the t and q parmeters. We iterate this two-step procedure until the parameters converge (this often happens in just a few iterations).

6 More on the EM Algorithm: Maximum-likelihood Estimation

Soon we'll trace an example run of the EM algorithm, on some simple data. But first, we'll consider the following question: how can we justify the algorithm? What are it's formal guarantees, and what function is it optimizing?

In this section we'll describe how the EM algorithm is attempting to find the maximum-likelihood estimates for the data. For this we'll need to introduce some notation, and in particular, we'll need to carefully specify what exactly is meant by maximum-likelihood estimates for IBM model 2.

First, consider the parameters of the model. There are two types of parameters: the translation parameters $t(f|e)$, and the alignment parameters $q(j|i,l,m)$. We will use t to refer to the vector of translation parameters,

$$t = \{t(f|e) : f \in F, e \in E \cup \{\texttt{NULL}\}\}$$

and q to refer to the vector of alignment parameters,

$$q = \{q(j|i,l,m) : l \in \{1 \ldots L\}, m \in \{1 \ldots M\}, j \in \{0 \ldots l\}, i \in \{1 \ldots m\}\}$$

We will use $\mathcal{T}$ to refer to the *parameter space* for the translation parameters—that is, the set of valid settings for the translation parameters, defined as follows:

$$\mathcal{T} = \{t : \forall e, f, t(f|e) \geq 0; \quad \forall e \in E \cup \{\texttt{NULL}\}, \sum_{f \in F} t(f|e) = 1\}$$

and we will use $\mathcal{Q}$ to refer to the parameter space for the alignment parameters,

$$\mathcal{Q} = \{q : \forall j, i, l, m,\ q(j|i,l,m) \geq 0; \quad \forall i, l, m,\ \sum_{j=0}^{l} q(j|i,l,m) = 1\}$$

Next, consider the probability distribution under the model. This depends on the parameter settings t and q. We will introduce notation that makes this dependence explicit. We write

$$p(f, a|e, m; t, q) = \prod_{i=1}^{m} q(a_i|i,l,m)t(f_i|e_{a_i})$$

as the conditional probability of a French sentence $f_1 \ldots f_m$, with alignment variables $a_1 \ldots a_m$, conditioned on an English sentence $e_1 \ldots e_l$, and the French sentence length m. The function $p(f, a|e, m; t, q)$ varies as the parameter vectors t

and q vary, and we make this dependence explicit by including t and q after the ";" in this expression.

As we described before, we also have the following distribution:

$$p(f|e,m;t,q) = \sum_{a \in \mathcal{A}(l,m)} p(f,a|e,m;t,q)$$

where $\mathcal{A}(l,m)$ is the set of all possible settings for the alignment variables, given that the English sentence has length l, and the French sentence has length m:

$$\mathcal{A}(l,m) = \{(a_1 \dots a_m) : a_j \in \{0 \dots l\} \text{ for } j = 1 \dots m\}$$

So $p(f|e,m;t,q)$ is the conditional probability of French sentence f, conditioned on e and m, under parameter settings t and q.

Now consider the parameter estimation problem. We have the following set-up:

- The input to the parameter estimation algorithm is a set of training examples, $(f^{(k)}, e^{(k)})$, for $k = 1 \dots n$.
- The output of the parameter estimation algorithm is a pair of parameter vectors $t \in \mathcal{T}, q \in \mathcal{Q}$.

So how should we choose the parameters t and q? We first consider a single training example, $(f^{(k)}, e^{(k)})$, for some $k \in \{1 \dots n\}$. For any parameter settings t and q, we can consider the probability

$$p(f^{(k)}|e^{(k)}, m_k; t, q)$$

under the model. As we vary the parameters t and q, this probability will vary. Intuitively, a good model would make this probability as high as possible.

Now consider the entire set of training examples. For any parameter settings t and q, we can evaluate the probability of the entire training sample, as follows:

$$\prod_{k=1}^{n} p(f^{(k)}|e^{(k)}, m_k; t, q)$$

Again, this probability varies as the paramters t and q vary; intuitively, we would like to choose parameter settings t and q which make this probability as high as possible. This leads to the following definition:

Definition 2 (Maximum-likelihood (ML) estimates for IBM model 2) *The ML estimates for IBM model 2 are*

$$(t_{ML}, q_{ML}) = \arg \max_{t \in \mathcal{T}, q \in \mathcal{Q}} L(t,q)$$

where

$$\begin{aligned} L(t,q) &= \log\left(\prod_{k=1}^{n} p(f^{(k)}|e^{(k)}, m_k; t, q)\right) \\ &= \sum_{k=1}^{n} \log p(f^{(k)}|e^{(k)}, m_k; t, q) \\ &= \sum_{k=1}^{n} \log \sum_{a \in \mathcal{A}(l_k, m_k)} p(f^{(k)}, a|e^{(k)}, m_k; t, q) \end{aligned}$$

We will refer to the function $L(t,q)$ as the log-likelihood function. □

Under this definition, the ML estimates are defined as maximizing the function

$$\log\left(\prod_{k=1}^{n} p(f^{(k)}|e^{(k)}, m_k; t, q)\right)$$

It is important to realise that this is equivalent to maximizing

$$\prod_{k=1}^{n} p(f^{(k)}|e^{(k)}, m_k; t, q)$$

because $\log$ is a monotonically increasing function, hence maximizing a function $\log f(t,q)$ is equivalent to maximizing $f(t,q)$. The $\log$ is often used because it makes some mathematical derivations more convenient.

We now consider the function $L(t,q)$ which is being optimized. This is actually a difficult function to deal with: for one thing, there is no analytical solution to the optimization problem

$$(t,q) = \arg \max_{t \in \mathcal{T}, q \in \mathcal{Q}} L(t,q) \tag{3}$$

By an "analytical" solution, we mean a simple, closed-form solution. As one example of an analytical solution, in language modeling, we found that the maximum-likelihood estimates of trigram parameters were

$$q_{ML}(w|u,v) = \frac{count(u,v,w)}{count(u,v)}$$

Unfortunately there is no similar simple expression for parameter settings that maximize the expression in Eq. 3.

A second difficulty is that $L(t,q)$ is *not* a convex function. Figure 3 shows examples of convex and non-convex functions for the simple case of a function $f(x)$ where x is a scalar value (as opposed to a vector). A convex function has a single

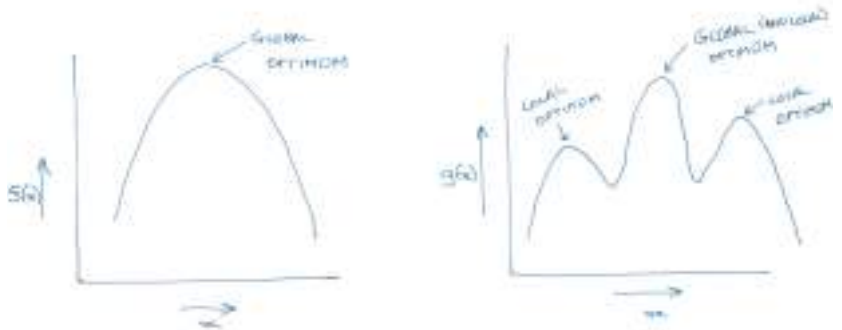

Figure 3: Examples of convex and non-convex functions in a single dimension. On the left, $f(x)$ is convex. On the right, $g(x)$ is non-convex.

global optimum, and intuitively, a simple hill-climbing algorithm will climb to this point. In contrast, the second function in figure 3 has multiple "local" optima, and intuitively a hill-climbing procedure may get stuck in a local optimum which is not the global optimum.

The formal definitions of convex and non-convex functions are beyond the scope of this note. However, in brief, there are many results showing that convex functions are "easy" to optimize (i.e., we can design efficient algorithms that find the $\arg\max$), whereas non-convex functions are generally much harder to deal with (i.e., we can often show that finding the $\arg\max$ is computationally hard, for example it is often NP-hard). In many cases, the best we can hope for is that the optimization method finds a *local* optimum of a non-convex function.

In fact, this is precisely the case for the EM algorithm for model 2. It has the following guarantees:

Theorem 1 (Convergence of the EM algorithm for IBM model 2) *We use $t^{(s)}$ and $q^{(s)}$ to refer to the parameter estimates after s iterations of the EM algorithm, and $t^{(0)}$ and $q^{(0)}$ to refer to the initial parameter estimates. Then for any $s \geq 1$, we have*

$$L(t^{(s)}, q^{(s)}) \geq L(t^{(s-1)}, q^{(s-1)}) \tag{4}$$

Furthermore, under mild conditions, in the limit as $s \rightarrow \infty$, the parameter estimates $(t^{(s)}, q^{(s)})$ converge to a local optimum of the log-likelihood function.

Later in the class we will consider the EM algorithm in much more detail:

we will show that it can be applied to a quite broad range of models in NLP, and we will describe it's theoretical properties in more detail. For now though, this convergence theorem is the most important property of the algorithm.

Eq. 4 states that the log-likelihood is strictly non-decreasing: at each iteration of the EM algorithm, it cannot decrease. However this does not rule out rather uninteresting cases, such as

$$L(t^{(s)}, q^{(s)}) = L(t^{(s-1)}, q^{(s-1)})$$

for all s. The second condition states that the method does in fact converge to a local optimum of the log-likelihood function.

One important consequence of this result is the following: *the EM algorithm for IBM model 2 may converge to different parameter estimates, depending on the initial parameter values $t^{(0)}$ and $q^{(0)}$*. This is because the algorithm may converge to a different local optimum, depending on its starting point. In practice, this means that some care is often required in initialization (i.e., choice of the initial parameter values).

7 Initialization using IBM Model 1

As described in the previous section, the EM algorithm for IBM model 2 may be sensitive to initialization: depending on the inital values, it may converge to different local optima of the log-likelihood function.

Because of this, in practice the choice of a good heuristic for parameter initialization is important. A very common method is to use IBM Model 1 for this purpose. We describe IBM Model 1, and the initialization method based on IBM Model 1, in this section.

Recall that in IBM model 2, we had parameters

$$q(j|i, l, m)$$

which are interpreted as the conditional probability of French word f_i being aligned to English word e_j, given the French length m and the English length l. In IBM Model 1, we simply assume that for all i, j, l, m,

$$q(j|i, l, m) = \frac{1}{l+1}$$

Thus there is a uniform probability distribution over all $l + 1$ possible English words (recall that the English sentence is $e_1 \ldots e_l$, and there is also the possibility that $j = 0$, indicating that the French word is aligned to $e_0 = \texttt{NULL}$.). This leads to the following definition:

Definition 3 (IBM Model 1) *An IBM-M1 model consists of a finite set $\mathcal{E}$ of English words, a set $\mathcal{F}$ of French words, and integers M and L specifying the maximum length of French and English sentences respectively. The parameters of the model are as follows:*

- *$t(f|e)$ for any $f \in \mathcal{F}$, $e \in \mathcal{E} \cup \{NULL\}$. The parameter $t(f|e)$ can be interpreted as the conditional probability of generating French word f from English word e.*

Given these definitions, for any English sentence $e_1 \dots e_l$ where each $e_j \in \mathcal{E}$, for each length m, we define the conditional distribution over French sentences $f_1 \dots f_m$ and alignments $a_1 \dots a_m$ as

$$p(f_1 \dots f_m, a_1 \dots a_m | e_1 \dots e_l, m) = \prod_{i=1}^{m} \frac{1}{(l+1)} \times t(f_i|e_{a_i}) = \frac{1}{(l+1)^m} \prod_{i=1}^{m} t(f_i|e_{a_i})$$

Here we define e_0 to be the NULL *word.* □

The parameters of IBM Model 1 can be estimated using the EM algorithm, which is very similar to the algorithm for IBM Model 2. The algorithm is shown in figure 4. The only change from the algorithm for IBM Model 2 comes from replacing

$$\delta(k,i,j) = \frac{q(j|i,l_k,m_k)t(f_i^{(k)}|e_j^{(k)})}{\sum_{j=0}^{l_k} q(j|i,l_k,m_k)t(f_i^{(k)}|e_j^{(k)})}$$

with

$$\delta(k,i,j) = \frac{\frac{1}{(l^{(k)}+1)}t(f_i^{(k)}|e_j^{(k)})}{\sum_{j=0}^{l_k} \frac{1}{(l^{(k)}+1)}t(f_i^{(k)}|e_j^{(k)})} = \frac{t(f_i^{(k)}|e_j^{(k)})}{\sum_{j=0}^{l_k} t(f_i^{(k)}|e_j^{(k)})}$$

reflecting the fact that in Model 1 we have

$$q(j|i,l_k,m_k) = \frac{1}{(l^{(k)}+1)}$$

A key property of IBM Model 1 is the following:

Proposition 1 *Under mild conditions, the EM algorithm in figure 4 converges to the global optimum of the log-likelihood function under IBM Model 1.*

Thus for IBM Model 1, we have a guarantee of convergence to the *global* optimum of the log-likelihood function. Because of this, the EM algorithm will converge to the same value, regardless of initialization. This suggests the following procedure for training the parameters of IBM Model 2:

Input: A training corpus $(f^{(k)}, e^{(k)})$ for $k = 1 \ldots n$, where $f^{(k)} = f_1^{(k)} \ldots f_{m_k}^{(k)}$, $e^{(k)} = e_1^{(k)} \ldots e_{l_k}^{(k)}$.
Initialization: Initialize $t(f|e)$ parameters (e.g., to random values).

Algorithm:

- For $t = 1 \ldots T$
  - Set all counts $c(\ldots) = 0$
  - For $k = 1 \ldots n$
    - For $i = 1 \ldots m_k$
      - For $j = 0 \ldots l_k$

$$
\begin{aligned}
c(e_j^{(k)}, f_i^{(k)}) &\leftarrow c(e_j^{(k)}, f_i^{(k)}) + \delta(k, i, j) \\
c(e_j^{(k)}) &\leftarrow c(e_j^{(k)}) + \delta(k, i, j) \\
c(j|i, l, m) &\leftarrow c(j|i, l, m) + \delta(k, i, j) \\
c(i, l, m) &\leftarrow c(i, l, m) + \delta(k, i, j)
\end{aligned}
$$

where

$$\delta(k, i, j) = \frac{t(f_i^{(k)}|e_j^{(k)})}{\sum_{j=0}^{l_k} t(f_i^{(k)}|e_j^{(k)})}$$

  - Set

$$t(f|e) = \frac{c(e, f)}{c(e)} \qquad q(j|i, l, m) = \frac{c(j|i, l, m)}{c(i, l, m)}$$

Output: parameters $t(f|e)$

Figure 4: The parameter estimation algorithm for IBM model 2, for the case of partially-observed data.

1. Estimate the t parameters using the EM algorithm for IBM Model 1, using the algorithm in figure 4.

2. Estimate parameters of IBM Model 2 using the algorithm in figure 2. To initialize this model, use: 1) the $t(f|e)$ parameters estimated under IBM Model 1, in step 1; 2) random values for the $q(j|i, l, m)$ parameters.

Intuitively, if IBM Model 1 leads to reasonable estimates for the t parameters, this method should generally perform better for IBM Model 2. This is often the case in practice.

See the lecture slides for an example of parameter estimation for IBM Model 2, using this heuristic.