Chapter 2: Word Alignment

In this chapter and the next we examine how much can be accomplished in MT, working just at the level of words. We build up the framework for word level alignment, leading to IBM models of SMT in the next chapter. Learning bilingual word mappings from parallel corpora is the focus of this chapter.

In the previous chapter we saw that translation is a mapping or transformational process. Since 1990s, borrowing techniques from speech, initiated by the IBM group, statistical machine translation slowly came to dominate MT. The basic idea in SMT is to *teach* a machine how to translate, through a large number of examples of translation. These examples should contain at least the following basic information:

a. Translation of words: what do the words map to; could be to more than one word in either direction
b. Movement of translated words to their correct positions in the target sentence; called *alignment*

For illustration, we take an English-Hindi translation pair:

2.1.E: Peter slept early today

2.1.H: पीटर आज जल्दी सोया

2.1.HT: piitar aaj jaldii soyaa

2.1.HG: Peter today early slept

Note: Throughout the book we follow a three part convention for expressing non-English text: the language (*e.g., H*, for Hindi), its transliteration (*HT*) and its gloss (*HG*). Thus in the above example, 2.1.H is the first Hindi string of the chapter, 2.1.HT is its transliteration from Hindi and 2.1.HG is the corresponding gloss (word to word translation) from Hindi.

In the parallel sentence pair 2.1, "piitar" maps to "Peter" (1→1), "aaj" to "today" (2→4), "jaldii" to "early" (3→3) and "soyaa" to "slept" (4→2). The mappings in parenthesis indicate alignment: *a(1)=1, a(2)=4, a(3)=3, a(4)=2*: the 1st word in the Hindi sentence maps to the 1st word in the English sentence, the 2nd word to the 4th, the 3rd to the 3rd and the 4th to the 2nd. The translation task is almost over when words are translated and the translated words are positioned. We say *almost*, because an important task may still be pending. A word from either side of the translation can map to more than one word or even to nothing on the other side. For example:

Bengali Sentence	**English Sentence**
2.2.B: সরোবরে লাল লাল ফুল প্রস্ফুটিত 2.2.BT: sarobare laal laal phul prosphutita 2.2.BG: in_the_lake red red flowers on_bloom 2.2.E: Red flowers are on bloom in the lake	sarobare → in the lake laal laal → red phul → flowers prosphutita → are on bloom

Table 2.1: Bengali-English parallel sentence

2.2.B and 2.2.E are parallel Bengali-English sentences, 2.2.BT the transliteration of the Bengali sentence and 2.2..BG the word to word English gloss. The column right shows many-to-one mappings from Bengali to English (*laal laal → red*) and also the one-to-many ones (*prosphutita*

→ *are on bloom, sarobare* → *in the lake*). This many-to-many alignment is the foundation for the so called *phrase based SMT* and will be discussed in chapter 3.

If we do not allow many-to-many mappings, we have to accept existence of *null mappings*. An example is in "Peter went to school" → "piitar paathshaalaa gayaa", where

to → *null*

i.e., 'to' maps to nothing in Hindi. Similarly, in "Peter hit hard" → "piitar ne jor se maaraa",

ne → *null; se* → *null*

i.e., 'ne' and 'se' map to nothing in English. Were we to allow many-to-many mappings, we could have had

Peter → *piitar ne; hard* → *jor se*

The phenomenon of one word mapping to multiple words on the other side is called "fertility" which we discuss in the next chapter on IBM models of word alignment.

We summarize. For translation to happen, one must put in place processes for:

A. Word translation

B. Translation alignment

C. Word fertility management or phrase alignment management

In this chapter we concentrate on a kind of alignment that is restricted, but is fundamental to understanding alignment, none the less. We presume *one to one* mapping, *i.e.*, any word from either side of the translation pair maps to **at least one and at most one** word on the other side. Language pairs like Hindi-Urdu, Hindi-Pujabi, Hindi-Gujrati, Assamese-Bengali, Spanish-Catalan *etc.* largely obey this property.

We note a key point: steps (A) and (B) above are mutually supportive; knowing one, one knows the other. Given the parallel sentence pair and word translations, one can get the alignments. Similarly, given the parallel sentence pair and the alignments, one can get the translations. Such two way absence of knowledge is symptomatic of application of Expectation Maximization (EM).

2.1 A combinatorial argument

Translation, alignment, fertility- whatever, the information has to emerge from data which in our case is parallel corpora. The bottom line is **counts**, *i.e.*, how often a phenomenon manifests in the data. For the evidence to be substantial and believable, we need sentence after sentence of parallel corpora. Let us reflect on what is needed for discovering alignment, given the parallel corpora.

If there are M words in the sentence S from language L_1 and N words in the parallel sentence T from L_2, then there are 2^{MN} possible alignments. Each of the M words in S can map to any subset

of *N* words in *T*. This makes for 2^N mappings per word in *S*. Hence the total number of possible alignments is 2^N x 2^N x 2^N x … *M times, i.e.,* 2^{MN}. Only one of these alignments is the desired one, meaning wise. For example, in parallel sentence pair 2.1.E, *Peter→piitar, slept→soyaa, early→jaldii, today→aaj* is the correct alignment.

> **Exercise 2.1**: Can you think an example from any language wherein there are more than one "meaningful" alignment? If you cannot think of one, argue why it is impossible to have more than one meaningful alignment.

2.1.1 Necessary and sufficient conditions for deterministic alignment in case of one-to-one word mapping

To concretize our ideas on alignment, we build the concepts in steps. We repeat that in this chapter we study the simplest case of alignment, *viz.*, given a parallel sentence pair *S* ←→ *T*, each word from *S* maps to at least one and at most one word in *T*. In such a situation, to finalize any alignment, two translation pairs are needed. One **introduces** the mappings and the other **singles out** one of these mappings. For example, the translation pair "*Peter slept early today*" and "*piitar aaj jaldii soyaa*" introduces the possibility of "*early→jaldii*" mapping. The certainty for this mapping can come in one of the following two forms:

(i) We have a translation pair containing "*early→jaldii*" and **none** of the other words in the translation pair. We will call this situation *one-same-rest-changed.*

(ii) We have a translation pair containing **all** words in the translation pair, except "*early→jaldii*". We will call this situation *one-changed-rest-same.*

From the above description we obtain the insight that:

word alignment requires evidence of:

a translation pair to introduce the POSSIBILITY of a mapping.

then, another pair to establish with CERTAINTY the mapping.

The above argument leads to the observation that: *If M valid bilingual mappings exist in a translation pair then an additional M-1 pairs of translations will decide these mappings with certainty.*

After *(M-1)* words have been aligned, the last mapping does not need a parallel pair, hence the need for *(M-1)* translation pairs.

2.1.1 A naïve estimate for corpora requirement

Based on the above observation, let us try to get a rough (and naïve) estimate of how many sentences will be needed for alignment. Suppose we are creating an SMT system between two languages L_1 and L_2. We assume no a-priori linguistic or world knowledge, *i.e.*, no meanings or grammatical properties of any words (*i.e.*, noun, verb, adjective, adverb, preposition *etc.*), phrases or sentences. Each language is assumed to have a vocabulary of 100,000 words. This repository can give rise to about 500,000 word forms (table 2.2), through various morphological processes; for example 'go' gives rise to 'go', 'went', 'gone' and 'going'.

Words of Language L_1	***Words of Language L_2***
U_1	V_1

U_2	V_2
U_3	V_3
U_4	V_4
…	…
…	…
U_{499999}	V_{499999}
U_{500000}	V_{500000}

Table 2.2: word lists of languages L_1 and L_2; we need to compute mapping probability

Let us assume 1 word from either language can on an average map to 5 words in the other language. This can happen due to:

(i) Synonymy on the target side (e.g., "*to go*" in English translating to "*jaanaa*", "*gaman karnaa*", "*chalnaa*" *etc.* in Hindi), a phenomenon called lexical choice or register

(ii) Polysemy on the source side (e.g., "*to go*" translating to "*ho jaanaa*" as in "*her face went red in anger*"→"*usakaa cheharaa gusse se laal ho gayaa*")

(iii) Syncretism ("went" translating to "*gayaa*"[1], "*gayii*"[2], or "*gaye*"[3]).

So, on an average we need to discover 5 translation correspondences per word. Each correspondence will need a sentence to introduce it and another to isolate it.

[1] Masculine Gender, 1st or 3rd person, singular number, past tense, non-progressive aspect, declarative mood

[2] Feminine Gender, 1st or 3rd person, singular number, past tense, non-progressive aspect, declarative mood

[3] Masculine or Feminine Gender, 1st, 2nd or 3rd person, Plural number, past tense, non-progressive aspect, declarative mood

2.1.1.1 One-changed-rest-same

Let us get an insight into the one-changed-rest-same alignment situation with a best case example:

Pair-1: Peter slept piitar soyaa; 2 correspondences introduced Pair-2: Robin slept rabin soyaa; *Peter*→*piitar, Robin*→*rabin, slept*→ *soya* resolved

3 correspondences got resolved by 2 sentences. Since we have to resolve (5 x 500,000) correspondences, we will need (5 x 500,000 x 2/3) or approximately 1.7 million parallel sentences *all of which are 2-word sentences*! This is a very artificial situation. For example, adverbs are most of the time in company of at least one noun and one verb.

Sentences come in all shapes and sizes. Modern day studies put the average sentence length as 14.3 for English[4]. Thus sentences introduce 10-15 correspondences and singling out each correspondence with complete certainty will need additional about that many sentences. As *M* in *C=(5 x 500K x (M-1)/M)* increases, so does *C* which is the number of parallel sentences required. *M* is the sentence length. The corpora requirement is upper bounded by *(5 x 500K)*, as $Lt_{M\to\infty}\frac{M-1}{M}=1$.

We take another example:

[4] http://en.wikipedia.org/wiki/Sentence_(linguistics)

Pair-1: Peter slept early today
piitar aaj jaldii soyaa*;* 4 correspondences introduced
Pair-2: Robin slept early today
rabin aaj jaldii soyaa; *Peter* and *Robin* resolved
Pair-3: Robin ate early today
rabin aaj jaldii khaayaa; *slept* and *ate* resolved
Pair-4: Robin ate late today
Rabin aaj der_se khaayaa; *today*, *early* and *late* resolved

As this example shows, 7 correspondences got resolved by 4 sentence pairs.

Generalizing, if originally *M* correspondences are introduced by a sentence, then by the "one-changed-rest-same" method, we can resolve *(2M-1)* correspondences through *M* parallel sentence pairs. This is because each new sentence helps resolve two correspondences and the last correspondence does not need a sentence. Thus the corpora requirement *C* is upper bounded by

$$C = \frac{P.D.M}{2M-1}$$

where, *D* is the lexicon size and *P* is the average number of correspondences per word on either side of the bilingual lexicon and *M* is the average sentence length. Assuming *D= 500,000*, *P= 5* and *M= 10*, we get C≈ 1,300,000. As $M \rightarrow \infty$, $C \rightarrow P.D/2$.

2.1.1.2 One-same-rest-changed

Now we discuss the other extreme, *viz.*, "one-same-rest-changed" situation:

Pair-1: Peter slept early today

piitar aaj jaldii soyaa*;* 4 correspondences introduced

Pair-2: Peter ate late yesterday

piitar kal der_se khaayaa; *Peter* resolved (use pairs 1 & 2);

3 new correspondences introduced

Pair-3: Robin slept at home

rabin ghar meM soyaa; *slept* resolved (1 & 3);

3 new correspondences

Pair-4: Reaching early he waited

jaldii pahuMchkar usne intajaar_kiyaa; *today, early* resolved;

3 new correspondences

(start resolving pair-2)

Pair-5: She ate a banana

use_ne eka kela khaayaa; *ate* resolved (2 & 5);

3 new correspondences

Pair-6: They all arrived late

ve sab der_se pahuMche; *late* , *yesterday* resolved (2 & 6);

3 new correspondences

(resolve pair-3)

Pair-7: Robin plays excellent chess

rabin utkrisht shatranj kheltaa_hai; Robin resolved (3 & 8);

3 new correspondences

…

and so on

Picturization and generalization will help. Suppose our starting sentence has M words. So there are *M* valid correspondences. Initially, *(M-1)* parallel sentence pairs will resolve these M correspondences. We keep sentence length fixed at *M* for all sentences.

Refer Figure 2.1 below. Pair P_1 introduces *M* correspondences. These are resolved by *(M-1)* pairs of sentences, P_{11}, P_{12}, P_{13}, … , $P_{1,(M-1)}$. These new set of pairs each introduces *(M-1)* new correspondences. These new correspondences from the 1st pair (say) are resolved by *(M-2)* pairs of sentences, P_{111}, P_{112}, P_{113}, … , $P_{11,(M-2)}$. This continues, giving rise to a computation tree of which figure 2.1 is a small part.

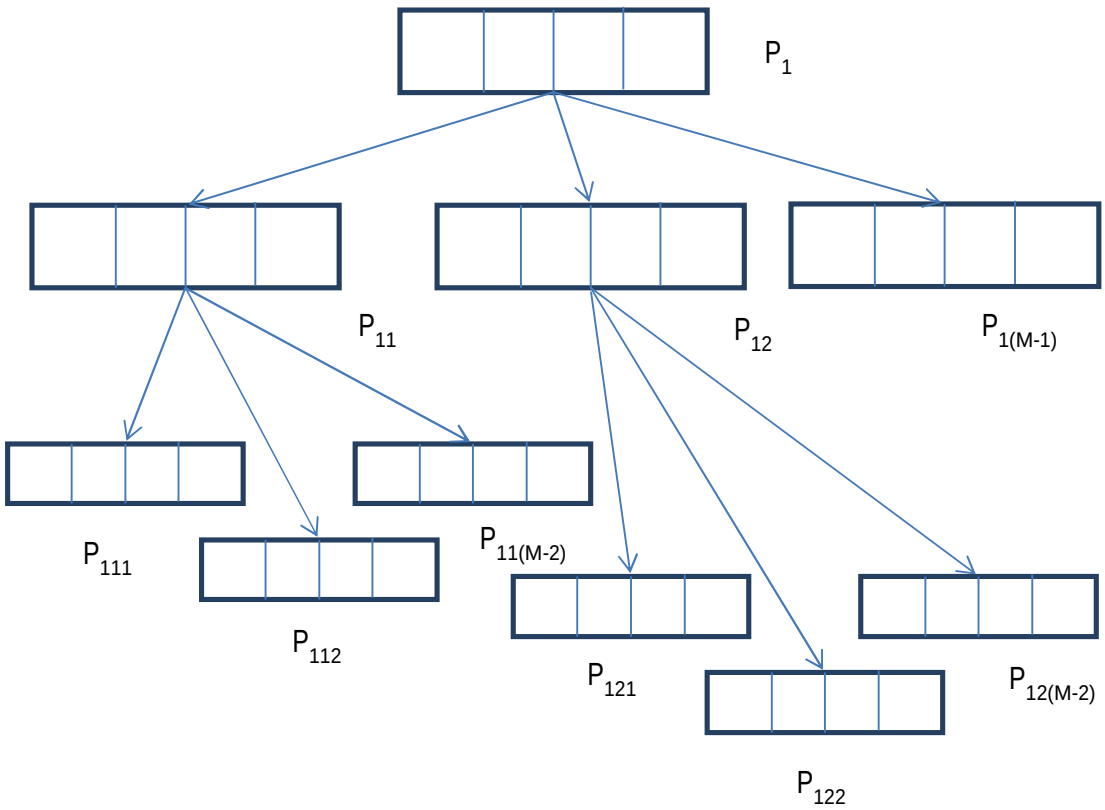

Figure 2.1: Partial tree: resolving correspondences with "one-same-rest-changed" method

We can compute the number of correspondences and corpora requirement as follows by referring to the levels of the tree:

Level	**# sentences pairs**	**#correspondences resolved (of previous level)**
0	1	Not applicable
1	M-1	M
2	(M-1).(M-2)	(M-1).(M-1)
3	$(M-1).(M-2)^2$	(M-1).(M-2).(M-1)
4	$(M-1).(M-2)^3$	$(M-1).(M-2)^2.(M-1)$
…	…	…
L	$(M-1)(M-2)^{L-1}$	$(M-1)(M-2)^{(L-2)}(M-1)$
Total	$1+(M-1)\frac{((M-1)^L-1)}{M-3}$	$M+(M-1)^2\frac{((M-2)^{L-1}-1)}{M-3}$

Table 2.3: Formulae for #sentence pairs and #correspondences for "one-same-rest-changed" situation

So per $M+(M-1)^2\frac{((M-2)^{L-1}-1)}{M-3}$ correspondences we need $1+(M-1)\frac{((M-1)^L-1)}{M-3}$ sentence pairs.

Let us get an estimate of the numbers involved. Our running example of lexicon size *D* is 500,000 (table 2.2). Suppose M=10. Number of correspondences is upper bounded by *P.D* which for us is 2,500,000. Now we have:

$$M + (M-1)^2 \frac{((M-2)^{L-1}-1)}{M-3} \leq P \bullet D$$

$$i.e., 10 + 81 \bullet \frac{8^{L-1}-1}{7} \leq 2500000$$

Neglecting terms,

$$8^L \leq \frac{7 * 2500000 * 8}{81}$$

$$i.e., \quad L \leq \frac{\log_2(1750000)}{3} \cong 7$$

This means that in about 7 levels we will cover all the correspondences, *provided we choose the sentences in the one-same-rest-changed scheme.* This is extreme form of active learning and is too idealistic to be applicable. But still this gives some insight and valuable figures.

Plugging in *L=7* in the formula for number of sentence pairs (table 2.3)

$$S=1+(10-1)[(10-2)^7-1)/(10-3)]=1+9.[8^7/7] \approx (9/7).8^7$$

These many parallel sentences are required to resolve $(8^7 x 81)/(7x8)$ or $(10/7).8^7$ correspondences. So in the setting discussed, approximately **per 10 correspondences 9 parallel sentences are required.** That is, in the current setting about 2.25M parallel sentences are needed. The "one-changed-rest-same" method requires in the limit about half the size of the corpora required by "one-same-rest-changed" method.

The reader might have noted that at the heart of this calculation is the assumption of careful choice of parallel sentence. This is never realistic. In practice, SMT systems pick up corpora from multilingual sources like Wikipedia, e-newspapers and so on. Also, teams of translators are employed to produce parallel translations. Thus, without control on corpora, intuitively speaking the corpora requirement will only go up.

In well-known SMT systems today (Google translate, Bing etc.), 10 to 15 million parallel sentences is common. Rule based systems, on the other hand, come with close to 100,000 rules (Systran system). These rules require many person years to build. Who said MT systems are easy to construct?

The moot point, though, is *if* we sacrifice 'certainty' and are prepared to work with most probable correspondences, what is the parallel corpora requirement? At the current state of knowledge, no theoretical limits are known.

> **Exercise 2.1** (research question): proceeding in the same line as above, can you work out the combinatorics of the corpora requirement when the setting is probabilistic?

The landscape of languages is complex, with unique requirements for translation of individual language pairs. Some languages are morphologically rich, *e.g.*, Dravidian languages, Finish, Hungarian *etc*. Their translation to a language with relatively simpler morphology benefits greatly from accurate morphology analysis on the source side. In the reverse direction of translation, morphology generation is a challenge (for example, translation from English to Finnish).

Morphology can help reduce resource requirement for SMT. *Factor based* SMT was introduced in 2007, wherein lemma and morphological features are inserted on parallel corpora. It was shown that for a given level of accuracy, morphology brings down the parallel corpora

requirement. Conversely, given a fixed amount of parallel corpora, accuracy level goes up if morph analysis is applied.

The fact that morphology could help can easily be seen from the discussions in section 2.1 on combinatorics of alignment. A vocabulary size of 100,000 lemma inflated to a lexicon size of 500,000. One reason for this inflation is inflexional morphology, as in 'go' undergoing inflection as 'going', 'gone', 'went'. Morphological analysers can look past inflected forms. The number of correspondences reduces considerably, if we aim to set up lemma mappings and not word mappings.

2.2 Deeper look at one-to-one alignment

Let us remember the basic equation of SMT

$$\hat{e} = \arg\max_{e}(P(e \mid f)) = \arg\max_{e}(P(e).P(f \mid e))$$

where, e is a sentence from the sentence repository E of language L_1 (say, English) and f is from the sentence repository F of language L_2 (the 'foreign' language). The *argmax* expression is broken down into two parts: **language model** *P(e)* and **translation model** *P(f|e)*. The term *P(e)* is computed from the product of *N*-grams obtained from corpora of L_1. On the other hand *P(f|e)* would have been the product of word translations conditioned on input words, but for the influence of alignment.

2.2.1 Drawing parallel with Part of Speech Tagging

Noisy channel modeling of SMT has similarity with POS tagging:

$$\hat{T} = \arg\max_{T}(P(T \mid W)) = \arg\max_{T}(P(T).P(W \mid T))$$

where $T^{\wedge}$ is the best possible tag sequence, T is the POS tag sequence for the input word sequence W. $P(T)$ is broken into product of tag N-grams (e.g., $P(t_i|t_{i-1})$ for $N=2$) and $P(W|T)$ into product of lexical probabilities $P(w_i|t_i)$ at each position i. A trellis of tags on the input words is set up and Viterbi algorithm is applied to compute the best possible tag sequence (figure 2.2).

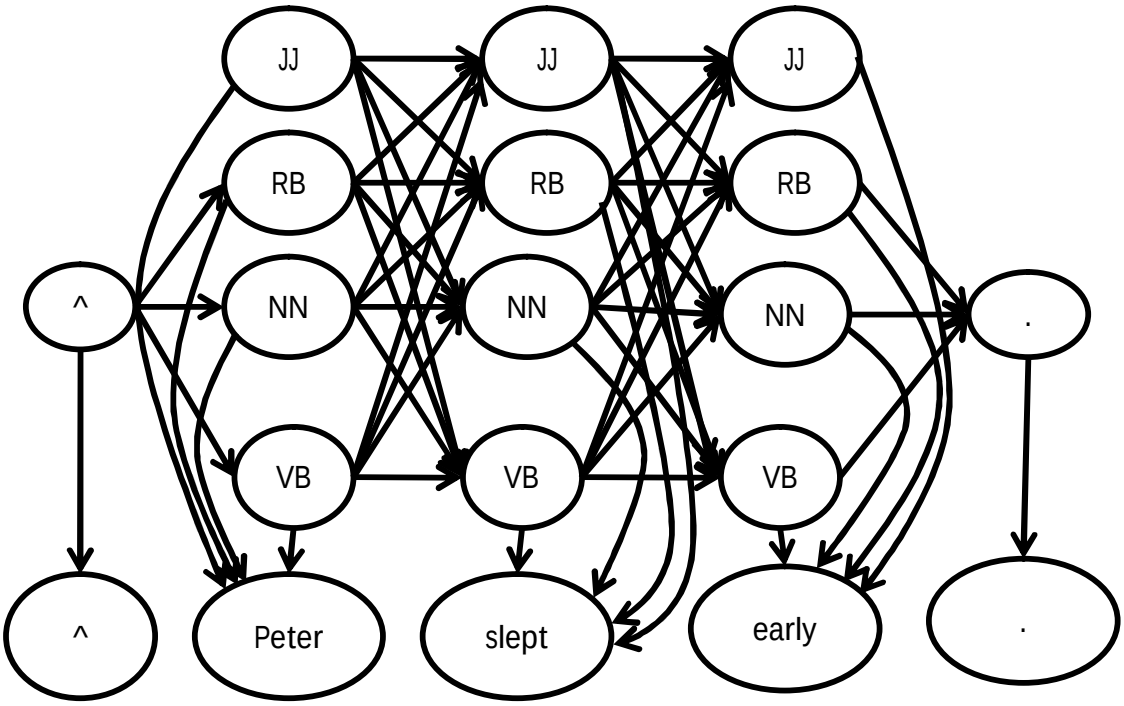

Figure 2.2: Trellis for POS tag (N-noun, V-verb and A-Adverb)

The sentence is "Peter slept early" with POS tags as *NN (noun), VB (verb)* and *RB (adverb)*. The POS trellis is drawn. The trellis describes an HMM of order 2. Viterbi algorithm run on the trellis produces the best tag sequence in $O(LT^2)$ time, where L is the length of input sentence and T the number of states (tags). The parameters are *P(VB|NN), P(RB|VB) etc.* and *P('Peter'|NN), P('slept'|RB) etc*. These parameters are learnt from POS tagged corpus:

Man/NNP is/VBZ known/VBN by/IN the/DT company/NN he/PRP keeps/VBZ ./.

Or equivalently as,

Man is known by the company he keeps .	NNP VBZ VBN IN DT NN PRP VBZ .

which can be looked upon as parallel 'corpora' with the sentence being the 'sentence of language L_1' and the tag sequence being the 'sentence in language L_2'. The crucial point is that there is one to one correspondence between the word sequence and the tag sequence. The key point to note is: *In POS tagging, tags and words are in direct positional correspondence*.

An analogous situation can be conceived of for MT. We can replace tags T by words V_E of English, and words W with the foreign language sentence f (figure 2.3). Hypothetically speaking, like in POS tagging, we can run Viterbi algorithm on the trellis to find the best path from the 'source' to the 'target', i.e., from '^' to '.'. Like in POS tagging, we compute product of transition probabilities (language model) and observation probabilities (translation model). At each stage of the trellis we keep track of the maximum over all possible states in the previous stage, maintaining a back pointer to recover the best path (English translation) when the algorithm stops.

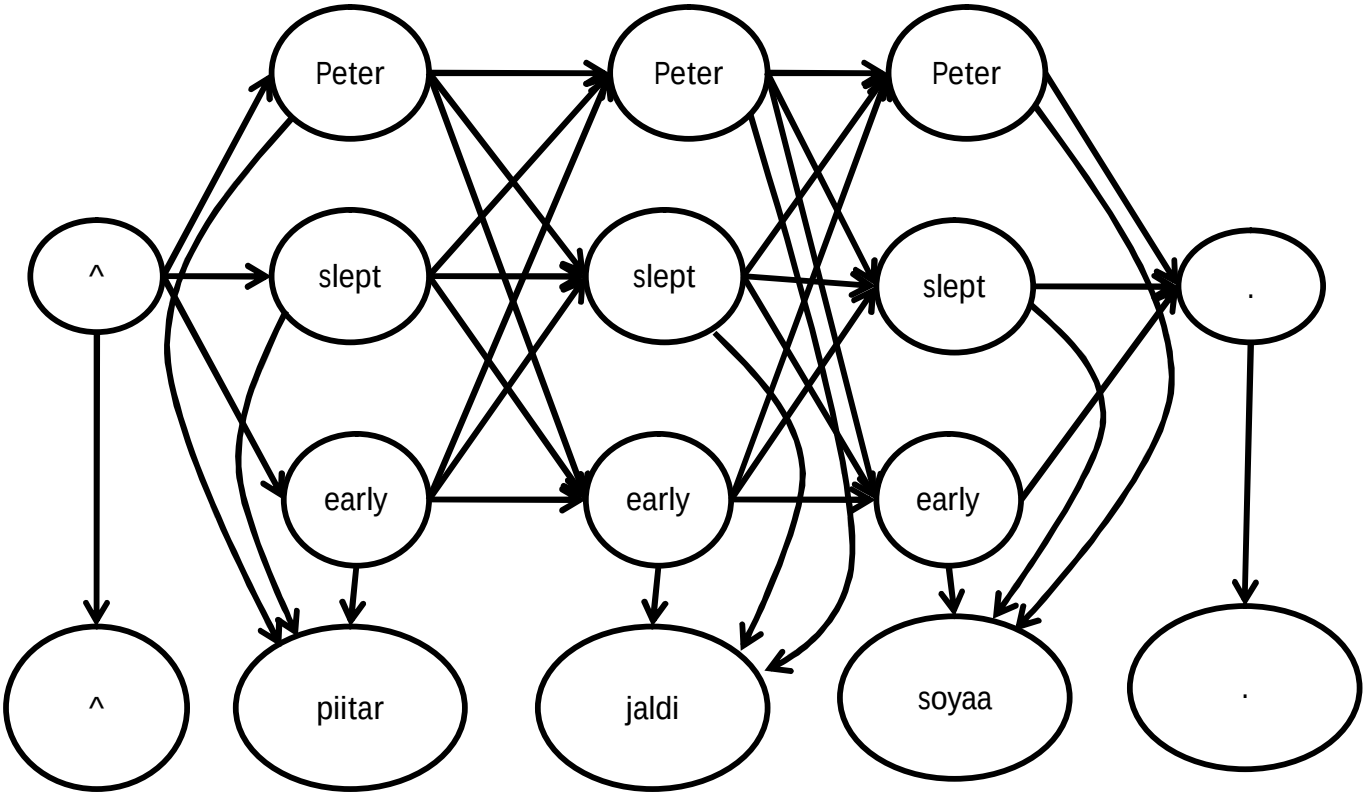

Figure 2.3: Trellis of English words for the Hindi sentence "piitar jaldii soyaa"

Language model probabilities come from the monolingual corpus of English. These are bigram probabilities like *P ('slept'|'Peter')*.

$$P('slept'|'Peter') = \frac{count(<'Peter','slept'>)}{count('Peter')}$$

i.e., the ratio of the number of times 'slept' follows 'Peter' and the number of times 'Peter' appears in the corpus.

Translation model probabilities have to be computed from parallel corpora. In POS tagging, we have a matrix of *tags* vs. *words* and their counts/probabilities:

words → ↓ tags	Apple	Bat	…	Web	…
NN	*P('apple'|N)*				
JJ					
VB		*P('bat'|VB)*			
RB					
…					

Table 2.4: lexical probability table in POS tagging

Each cell in the matrix records *P(w|t)*, the probability of word *w* given tag *t*, e.g., *Pr('slept'|verb)*.

$$P('slept'|\, verb) = \frac{count(verb, 'slept')}{count(verb)}$$

That is, divide the count of the times a verb manifests as 'slept' by total count of verbs in the corpus. Such a count can be obtained and trusted, since the tags are right ON the words, *i.e.*, in one to one positional correspondence with the words.

An identical procedure will not in general work for SMT. We can create a matrix of V_E *x* V_F, where V_E is the vocabulary of English and V_F is the vocabulary of the foreign language.

V_F → ↓ V_E	…	Piitar	…	jaldii	…	soyaa	…
…							
Peter		*P('piitar'| 'Peter')*					

…							
Slept						*P(‘soyaa| ‘slept’)*	
…							
Early				*P(‘jaldii’| ’early)*			
…							

Table 2.5: V_E *x* V_F matrix

How should we compute these probabilities? For example, if we say,

$$\Pr('soyaa' | slept) = \frac{count('soyaa', 'slept')}{count('slept')}$$

how do we get *count(‘soyaa’, ‘slept)*? Where in the text should look to get the count of joint occurrence of ‘soyaa’ and ‘slept’? The answer is: *increment a counter whenever ‘soyaa’ and ‘slept’ co occur in a parallel sentence pair.*

Thus both in POS tagging and in MT, we count co-occurrence. For POS tag the text span is a single position; increment *count(word, tag)* if *word* and *tag* co occur at the same position. For MT, the text span could be the *sentence*.

Here arises the need for alignment. From parallel sentence pairs like “Peter slept early ←→ piitar jaldii soya” we will see *Peter→piitar* is possible and get its counts too. However, we will *not have confidence* in these counts, since *Peter→jaldii* and *Peter→soyaa*, are also possible. From the discussion in section 2.1 (combinatorics of alignment), we know we will need additional

examples mentioning *Peter→piitar, early→jaldii, slept→soyaa* to reinforce the strength of these mappings.

A point to note: if alignment of each English word is known with certainty, we can consider the probability values in the $(V_E \times V_F)$ table as final. Then the translation problem is reduced to HMM-Viterbi like situation. Of course, the key assumption in this case is *in a sentence every English word is mappable to at least and at most one foreign word* just like a tag is mappable to at least and at most one word in a sentence.

How good is this assumption of one source word per one target word? For familialy very close languages this is a good assumption. For example, Hindi-Gujarati, Assamese-Bangla, Spanish-Catalan *etc.* are likely to follow this one to one correspondence. Translation between such languages can be looked upon as taking place at the base of the Vauquois Triangle.

The reader will now ask: for such familialy close languages why not adopt a word substitution strategy, *i.e.*, for every word in source language substitute a word from the target language? This will not work, since disambiguation will still be required. POS tagging, we remember, is a disambiguation task. A word can have multiple tags depending on the context:

Children went to *play_VB* in the city garden.

Children went to the *play_NN* in the city hall.

Play is POS disambiguated by the preceding words 'to' and 'the'. HMM-Viterbi framework disambiguates using tag bigram and lexical probabilities.

For translation between close languages too we need context, because words can have multiple translations. Target language bigram probabilities and target→source translation probabilities facilitate disambiguation.

Now that we need to fill the V_E x V_F table with evidence from multiple parallel sentences, we proceed to illustrate the computation followed by the theory behind this computation.

2.3 Heuristics based computation of the V_E X V_F table

English Sentence	French Sentence
2.4.E: *three rabbits* a b 2.5.E: *rabbits of Grenoble* b c d	2.4.F: *trois lapins* w x 2.F.5: *lapins de Grenoble* x y z

Table 2.6: Two English-French parallel sentence pairs

We discuss a simple method of filling the translation probability table with the parallel corpora of two sentence pairs (Table 2.6). "three rabbits ←→ trois lapins" is the first parallel sentence pair and "rabbits of Grenoble ←→ lapins de Grenoble" is the second. Note that "rabbits ←→ lapins" alignment is repeated in the two pairs. This will show up in the alignment probabilities. The probability *P(lapins|rabbits)* is:

$$\frac{1+1}{(1+1)+(1+1+1)} = \frac{2}{5} = 0.4$$

The numerator is the number of possibilities of "rabbit ←→ lapins" mappings in the whole corpus, which is 2 from the 2 sentence pairs. The denominator is the total number of possible "rabbit ←→ x" mappings for any word '*x*' in the parallel corpus. In the first parallel sentence pair "rabbits" can map to any of "trois" and "lapins. In the second sentence it can map to any of "lapins", "de" and "Grenoble". Thus we get the probability value as 2/5=0.4. The complete probability table is shown below (table 2.7):

V_F → V_E ↓	*Trois*	*lapins*	*De*	*Grenoble*
Three	½	1/2	0	0
Rabbits	1/5	2/5	1/5	1/5
Of	0	1/3	1/3	1/3
Grenoble	0	1/3	1/3	1/3

Table 2. 7: Alignment probabilities found by simple counting heuristic; average entropy=1.53

We will use entropy as the figure of merit. Each row is a probability distribution. We compute the average entropy of the probability distribution in the alignment table 2.7:

Row-1 entropy:

$$[-(1/2*\log_2(1/2))-(1/2*\log_2(1/2))]=1$$

Row-2 entropy:

$[-(1/5)*log_2(1/5)-(2/5)*log_2(2/5)-(1/5)*log_2(1/5)-$

$(1/5)*log_2(1/5)]=0.2*(2.32+2*1.32+2.32+2.32)= (1.6+0.32)=1.92$

Row-3 entropy:

$[-(1/3*log_2(1/3))-(1/3*log_2(1/3))-(1/3*log_2(1/3))]= -log_2(1/3)=1.6$

Row-3 entropy:

$[-(1/3)*log_2(1/3)-(1/3)*log_2(1/3)-(1/3)*log_2(1/3)]= -log_2(1/3)=1.6$

Average entropy = (1+1.92+1.6+1.6)/4=1.53

2.4 Iterative (EM based) computation of the V_E x V_F table

The heuristic alignment described above gives a way of fixing the translation probabilities. But it is *ad hoc*, and we do not know how good the word alignments found are. The only constraint satisfied is that row values sum to 1, *i.e.*, form a probability distribution. Since word alignment and word translation are mutually supportive (knowledge of one helps know the other), a principled algorithm can be used to find both. This is the expectation maximization algorithm grounded on the principle of maximum data likelihood estimate. The procedure is an iterative one, alternating between computing the probability of word alignments (M-step) and getting the expected count of these alignments (E-step).

Initialization and Iteration-1 of EM: We start with uniform probability values for the alignments. In absence of any information other than the parallel corpora, all alignments are equally likely. *Three*, for example, can map to any one of *trois*, *lapins*, *de* and *Grenoble* with equal probability. So:

$V_F \rightarrow$ $V_E \downarrow$	*Trois*	*lapins*	*de*	*Grenoble*
Three	¼	1/4	1/4	¼
Rabbits	¼	1/4	1/4	¼
Of	¼	1/4	1/4	¼
Grenoble	¼	1/4	1/4	¼

Table 2.8: Initial alignment probabilities; Average entropy=2

From this we get the "expected counts" (explained in the theory in section 2.4) of alignments. For the expected count of *three* → *trois* alignment from the first parallel sentence, we note the count of *three* and *trois* in the sentence pair and weigh the count by current *Pr(trois|three)* normalized.

$$expected\_count\ [three \leftrightarrow trois;\ (three\ rabbits) \leftrightarrow (trois\ lapins)]$$

$$= \frac{\Pr(trois \mid three)}{\Pr(trois \mid three) + \Pr(lapins \mid three)} * (\#three) * (\#trois)$$
$$= \frac{1/4}{1/4 + 1/4} * 1 * 1$$
$$= 1/2$$

Three and *trois* appear once each in the first parallel sentence pair. The current *Pr(troi|three)* value is ¼. In the parallel sentence pair "three rabbits ↔ trois lapins", *three* can map to *lapins* also. Hence the weightage factor is ½.

Carrying out the above procedure, we get two "count" tables for the two sentences. This completes iteration-1.

Three rabbits ←→ ***trois lapins***	*trois*	*Lapins*	*de*	*Grenoble*
Three	1/2	½	0	0
Rabbits	1/2	½	0	0
of	0	0	0	0
Grenoble	0	0	0	0

Tables 2.9: Expected counts of mappings in "three rabbits ←→ troi lapins"

Rabbits of Grenoble ←→ ***Lapins de Grenoble***	*trois*	*lapins*	*de*	*Grenoble*
Three	0	0	0	0
Rabbits	0	1/3	1/3	1/3
of	0	1/3	1/3	1/3
Grenoble	0	1/3	1/3	1/3

Tables 2.10: Expected counts of mappings in "rabbits of grenoble ←→ lapins de Grenoble"

Iteration-2: From these expected counts we get the "revised" probabilities *Pr(trois|three)*, *Pr(lapins|three)*, *Pr(lapins|rabbit)* *etc*. Since the current value of *Pr(lapins|rabbit)* is ¼=0.25, it will be interesting to see what it gets updated to:

$$P_{revised}(lapins \mid rabbit)$$
$$= \frac{count(rabbits \rightarrow lapins\ in\ the\ corpus)}{count(rabbits \rightarrow anything\_else\ in\ the\ corpus)}$$
$$= \frac{1/2 + 1/3}{(1/2 + 1/2) + (1/3 + 1/3 + 1/3)}$$
$$= \frac{5}{12} \approx 0.4$$

The numerator is the total "count" in the corpus of *rabbits* ←→*lapins* which from tables 2.9 and 2.10 is (1/2+1/3). The denominator is the total count of *rabbits* ←→ *X* mapping, where X is any other word. That count from the first parallel sentence is (1/2+1/2) and from the second sentence is (1/3+1/3+1/3).

Thus after the first iteration the probability *P(lapins|rabbit)* seems to be moving in the right direction. The value has increased from 0.25 to 0.4.

V_F → V_E ↓	*Trois*	*lapins*	*de*	*Grenoble*
Three	½	1/2	0	0
Rabbits	¼	5/12	1/6	1/6

Of	0	1/3	1/3	1/3
Grenoble	0	1/3	1/3	1/3

Table 2.11: Revised alignment probabilities after iteration-1; average entropy= 1.9

We revise the counts now:

Three rabbits ←→ ***trois lapins***	*trois*	*lapins*	*de*	*Grenoble*
Three	1/2	½	0	0
Rabbits	1/2	½	0	0
of	0	0	0	0
Grenoble	0	0	0	0

Table 2.12: revised expected counts of mappings in "three rabbits ←→ troi lapins"

Rabbits of Grenoble ←→ ***Lapins de Grenoble***	*trois*	*lapins*	*de*	*Grenoble*
Three	0	0	0	0

Rabbits	0	5/12	1/6	1/6
of	0	1/3	1/3	1/3
Grenoble	0	1/3	1/3	1/3

Table 2.13: revised expected counts of mappings in "rabbits of grenoble ←→ lapins de Grenoble"

Iteration-3: The alignment probabilities will get revised as:

V_F → V_E ↓	*Trois*	*lapins*	*de*	*Grenoble*
Three	½	1/2	0	0
Rabbits	2/7	11/21	2/21	2/21
Of	0	1/3	1/3	1/3
Grenoble	0	1/3	1/3	1/3

Table 2.14: Revised alignment probabilities after iteration-2; average entropy= 1.4

Thus *P(lapins|rabit)* value goes on increasing (now it is about 0.5) and the average entropy value is already less than the average entropy value with heuristic alignment. Thus we are progressing towards a better probability distribution.

Exercise 2.1: complete the interactions and see what values the 'rabbit row' in the probability table converges to.

Exercise 2.2: Write MATLAB code to program the above procedure.

Exercise 2.3: You will note that the "Rabbit" example is a case of "one-same-rest-changed" parallel corpora; *rabbits* ←→ *lapins* is unchanged. It is possible to know with certainty this mapping. Verify that the "rabbit" row keeps boosting *P(lapins|rabbits)* to 1, while suppressing the other values.

The illustration above shows the following two formulae to be computed:

$$C(e_i \leftrightarrow f_j; e \leftrightarrow f) = \frac{\Pr(f_j \mid e_i)}{\sum_x \Pr(x \mid e_i)} * (\#e_i \in e) * (\#f_j \in f) \qquad -E-step$$

where, *C* is the expected count of $e_i \leftarrow\rightarrow f_j$ mapping in the context of the parallel corpus $e \leftarrow\rightarrow f$. $(\#e_i \varepsilon e)$ is the number of times e_i occurs in e; similarly for f_j. The other formula is:

$$\Pr(f_j \mid e_i) = \frac{\sum_s C(e_i \leftrightarrow f_j; e^s \leftrightarrow f^s)}{\sum_s \sum_x C(e_i \leftrightarrow x; e^s \leftrightarrow f^s)} \qquad -M-step$$

where, *s* refers to a particular sentence pair. $P(f_j|e_i)$ is calculated from the ratio of "count"s (which can be fractional as per the E-step) of $e_i \leftarrow\rightarrow f_j$ mappings in all parallel sentence pairs and the "count" of mappings of $e_i \leftarrow\rightarrow x$, where *x* is any word in all the parallel sentences.

It can be proved that after every-iteration of E-step and M-step, the likelihood of the data which in this case is the parallel corpora, increases monotonically. An equivalent way of describing improvement is the progressive decrease in entropy. Thus the iterative procedure is a greedy

procedure. It could have got stuck in local minimum, but for the fact that the data likelihood expression is a convex one and guarantees global minimum. We now proceed to give the mathematics behind our EM base alignment algorithm.

2.5 Mathematics of Alignment

We do a build up to the formulae based on EM that we have used. The fundamental methodology is:

(a) A data or observation likelihood expression is set up, assuming an appropriate distribution.

(b) The parameters of the distribution are the quantities of interest (in our case alignment probabilities).

(c) Hidden variables are assumed for mathematical simplicity of the likelihood expression.

(d) The parameters and the hidden variables are estimated iteratively, starting with initial values of parameters.

(e) The iteration step to estimate the parameters is called the ***M-step*** or maximization step.

(f) The iteration step to estimate the hidden variables is called the **E-step** or expectation step.

This is the essence of the **EM** algorithm. We keep in mind, we need to estimate the alignment probabilities. Below we present a number of problems that illustrate the key concepts of data likelihood, parameters and hidden variables. In general, if X is an observations sequence $<x_1, x_2, x_3,\ldots, x_{N-1}, x_N>$ of length N generated from a distribution Ψ with parameters θ, then the data likelihood D is given as:

$$D = P(X;\theta)$$
$$\textit{also written as } P_\theta(X)$$

That is, D is the joint probability of the observation sequence. In general, the form and nature of the distribution are pre supposed. This is the *inductive bias* of the learning situation. The problem of learning BOTH the distribution AND its parameters is an unsolved problem.

There are two basic methodologies for estimating θ:

1. Find θ by **MLE**. That is, find parameters that will maximize the likelihood of the data or observation.
2. Find θ by **MAXENT**. That is, find parameters that will maximize the entropy of the distribution Ψ subject to the constraint that the expected values of *features* of the observation as per Ψ must match the actual values of the features.

2.5.1 A few illustrative problems to clarify application of EM

Situation-1: throw of a single coin: the parameter is the probability p of getting *head* in a single toss. Let N be the number of tosses. Then the data or observation likelihood is

$$D = \prod_{i=1}^{N} p^{x_i}(1-p)^{1-x_i}, \text{ s.t. } x_i = 1 \text{ or } 0, \text{ and } 0 \le p \le 1$$

where, x_i is an indicator variable assuming values 1 or 0 depending on the i^{th} observation being *head* or *tail*. Since there are N identically, independently distributed (*i.i.d.*) observations, D is the product of probabilities of individual observations each of which is a Bernoulli trial. Since exponents are difficult to manipulate mathematically, we take log of D, also called log likelihood of data and maximize wrt. p. This yields

$$p = \frac{\sum_{i=1}^{N} x_i}{N}$$

If *M* is the number of heads out of *N*, then $p = M/N$, since M= $\sum_{i=1}^{N} x_i$. M and N are called sufficient statistics, since these observations are sufficient to estimate *p*.

Throw of two coins: now there are 3 parameters, *viz.*, probabilities p_1 and p_2 of heads of the two coins and the probability *p* of choosing the first coin (automatically, *1-p* is the probability of choosing the second coin). Here too we have *N* tosses and observations of heads and tails. Only, we do not know which observation comes from which coin. An indicator variable z_i is introduced to capture coin choice ($z_i=1$ if coin_1 is chosen, else 0). This variable is hidden, *i.e.*, we do not know its values. However, without it the likelihood expression would have been very cumbersome.

$$\textit{Data Likelihood},\ D = P_\theta(X)$$
$$\textit{and},$$
$$P_\theta(X,Z) = \prod_{i=1}^{N} \left(p p_1^{x_i} (1-p_1)^{1-x_i} \right)^{z_i} \left((1-p) p_2^{x_i} (1-p_2)^{1-x_i} \right)^{1-z_i},$$
$$\text{s.t. } x_i = 1 \text{ or } 0,\ z_i = 1 \text{ or } 0,\ \theta = < p, p_1, p_2 >$$

The choice of each coin is a Bernoulli trial. After a choice is made, we have another Bernoulli trial for the outcome of the toss. For mathematical convenience we work with expectation of log likelihood:

$$E(LL(D)) = \sum_{i=1}^{N} [E(z_i)(\log p + x_i \log p_1 + (1-x_i)\log(1-p_1)) + (1-E(z_i))(\log(1-p) + x_i \log p_2 + (1-x_i)\log(1-p_2))]$$
$$X := < x_1, x_2, x_3, ..., x_{N-1}, x_N > \ \textit{and}\ \ Z := < z_1, z_2, z_3, ..., z_{N-1}, z_N >$$

How has z_i turned into expectation of z_i, $E(z_i)$? The reasoning is as follows. We can introduce the hidden variable Z only through marginalization:

$$P_\theta(X) = \sum_Z P_\theta(X, Z)$$

We would like to work with $log(P_\theta(X))$. However, then we will have a Σ expression inside *log*. This is cumbersome. Fortunately *log* is a concave function, so that

$$\log(\sum_{i=1}^{n} \lambda_i y_i) \geq \sum_{i=1}^{n} \lambda_i \log(y_i), \ \ with \ \sum_{i=1}^{n} \lambda_i = 1$$

This is the application of Jensen's in-equality. Thus log can move inside past Σ, provided we can find appropriate λs. In this case λs are constructed from $P_\theta(X,Z)$ at particular values of Z and θ. A bit of manipulation with these probability values leads to:

$$\begin{aligned} LL(D) &= \log likelihoood \ \ of \ \ data \\ &= \log(P_\theta(X)) \\ &= \log(\sum_Z P_\theta(X, Z)) \\ &\geq E_{Z,\theta}(\log(P(X, Z))) \end{aligned}$$

where $E_{Z,\theta}(log(P(X,Z))$ is the expectation of log likelihood of the data at particular values of Z and θ. This gives rise to the above *log likelihood* expression (θ is dropped, being clear from the context). Differentiating wrt p, p_1 and p_2 and equating to 0, we get:

$$p_1 = \frac{\sum_{i=1}^{N} E(z_i) x_i}{\sum_{i=1}^{N} E(z_i)}$$

$$p_2 = \frac{M - \sum_{i=1}^{N} E(z_i) x_i}{N - \sum_{i=1}^{N} E(z_i)}, \ M = \text{observed no. of heads}$$

$$p = \frac{\sum_{i=1}^{N} E(z_i)x_i}{N}$$

$$E(z_i)$$
$$= P(z_i = 1 \mid x = x_i) = P(z_i = 1).P(x = x_i \mid z_i = 1) / P(x = x_i)$$
$$= \frac{PP_1^{x_i}(1-P_1)^{(1-x_i)}}{PP_1^{x_i}(1-P_1)^{(1-x_i)} + (1-P)P_2^{x_i}(1-P_2)^{(1-x_i)}}$$

Generalization: **throw of more than 1 'something', where that 'something' has more than 1 outcome:** this gives rise to a multinomial which is extremely useful in many NLP and ML situations:

- Observation sequence: *N* observations, with each observation having *L* outcomes such that anyone of the *L* outcomes is possible for each of the observation, *i.e.*, $\sum_{k=1}^{L} x_{ik} = 1$, since each x_{ik}=1/0 and one and only one of them is 1.

$$D: (x_{11}, x_{12}, x_{13}, ..., x_{1L}), (x_{21}, x_{22}, x_{23}, ..., x_{2L}), ..., (x_{N1}, x_{N2}, x_{N3}, ..., x_{NL})$$

- Hidden variable for *M* sources

$$Z: (z_{11}, z_{12}, z_{13}, ..., z_{1M}), (z_{21}, z_{22}, z_{23}, ..., z_{2M}), ..., (z_{N1}, z_{N2}, z_{N3}, ..., z_{NM})$$

  Since one and only one source is chosen for each observation, *i.e.*, $\sum_{j=1}^{M} z_{ij}$, since each z_{ij}= 1/0 and one and only one of them is 1.

- Parameters θ:

  π_j: Probability of choosing source j

  p_{jk}: Probability of observing k^{th} outcome from j^{th} source

Assuming the observations to be *i.i.d.* (independently and identically distributed, in this case - multinomial distribution), the likelihood, the log likelihood and the expectation of the log likelihood are respectively:

$$L(D;\theta) = \prod_{i=1}^{N} \prod_{j=1}^{M} \left(\pi_j \prod_{k=1}^{L} (p_{jk}^{x_{ik}})\right)^{z_{ij}} \quad (2.1)$$

$$LL(D;\theta) = \sum_{i=1}^{N} \sum_{j=1}^{M} z_{ij} \left(\pi_j + \sum_{k=1}^{L} (x_{ik} \log p_{jk})\right) \quad (2.2)$$

$$E(LL(D;\theta)) = \sum_{i=1}^{N} \sum_{j=1}^{M} E(z_{ij}) \left(\pi_j + \sum_{k=1}^{L} (x_{ik} \log p_{jk})\right) \quad (2.3)$$

Maximize (2.3) subject to the constraints:

$$\sum_{j=1}^{|M} \pi_j = 1 \quad (2.4)$$

$$\sum_{k=1}^{L} P_{jk} = 1, \; j = 1,...,M \quad (2.5)$$

(2.4) is true since one of the sources is certain to be chosen. (2.5) is true, since given a source, one of the outcomes is certain.

Introducing one Langrangian for (2.4) and *M* Langrangians for each of the M sources as per (2.5), the dual of (2.3) to be optimized is:

$$Q(D;\theta) = \sum_{i=1}^{N} \sum_{j=1}^{M} E(z_{ij}) \left(\pi_j + \sum_{k=1}^{L} (x_{ik} \log p_{jk})\right) - \alpha\left(\sum_{j=1}^{M} \pi_j - 1\right) - \sum_{j=1}^{M} \beta_j \left(\sum_{k=1}^{L} P_{jk} - 1\right) \quad (2.6)$$

Differentiating Q w.r.t. π_j and then equating with 0,

$$\frac{\partial Q}{\partial \pi_j} = \sum_{i=1}^{N} \frac{E(z_{ij})}{\pi_j} - \alpha = 0$$

So,

$$\alpha \times \pi_j = \sum_{i=1}^{N} E(z_{ij})$$

Therefore,

$$\alpha = \sum_{j=1}^{M} \sum_{i=1}^{N} E(z_{ij}) \quad \textit{from } (2.3)$$

Giving,

$$\pi_j = \frac{\sum_{i=1}^{N} E(z_{ij})}{\sum_{j=1}^{M} \sum_{i=1}^{N} E(z_{ij})} \qquad (2.7)$$

Now, differentiating equation 4 w.r.t. p_{jk} and equating to 0,

$$\frac{\partial Q}{\partial p_{jk}} = \sum_{i=1}^{N} \frac{E(z_{ij}) x_{ik}}{p_{jk}} - \beta_j = 0$$

So,

$$\beta_j \times p_{jk} = \sum_{i=1}^{N} E(z_{ij}) x_{ik}$$

Therefore,

$$\beta_j = \sum_{k=1}^{L} \sum_{i=1}^{N} E(z_{ij}) x_{ik} \quad \textit{from } (2.4)$$

Now,

$$p_{jk} = \frac{\sum_{i=1}^{N} E(z_{ij}) x_{ik}}{\sum_{k=1}^{L} \sum_{i=1}^{N} E(z_{ij}) x_{ik}}$$

$$= \frac{\sum_{i=1}^{N} E(z_{ij}) x_{ik}}{\sum_{i=1}^{N} E(z_{ij}) \sum_{k=1}^{L} x_{ik}}$$

But,

$$\sum_{k=1}^{L} x_{ik} = 1,$$

giving,

$$p_{jk} = \frac{\sum_{i=1}^{N} E(z_{ij}) x_{ik}}{\sum_{i=1}^{N} E(z_{ij})} \qquad (2.8)$$

Now, we need to get an expression for hidden variable $E(z_{ij})$ in terms of parameters, θ: π_j $(j=1,2,\ldots,M)$ and p_{jk} $(k=1,2,\ldots,L)$. We would like to find an expression for $E(z_{ij})$= Probability that the j^{th} source was chosen given the i^{th} observation:

$$E(z_{ij}) = P(source = j \mid obs = i)$$
$$= \frac{P(source = j,\ obs = i)}{P(obs = i)}$$
$$= \frac{P(source = j,\ obs = i)}{\sum_{j=1}^{M} P(source = j,\ obs = i)}$$
$$= \frac{\pi_j \prod_{k=1}^{L} (p_{jk}^{x_{ik}})}{\sum_{j=1}^{M} \pi_j \prod_{k=1}^{L} (p_{jk}^{x_{ik}})} \qquad (2.9)$$

To summarize the EM algorithm- initialize values of parameters θ randomly and iterate through the following steps:

E-Step:

$$E(z_{ij}) = \frac{\pi_j \prod_{k=1}^{L} (p_{jk}^{x_{ik}})}{\sum_{j=1}^{M} \pi_j \prod_{k=1}^{L} (p_{jk}^{x_{ik}})}$$

M-Step:

$$\pi_j = \frac{\sum_{i=1}^{N} E(z_{ij})}{\sum_{j=1}^{M} \sum_{i=1}^{N} E(z_{ij})}$$

$$p_{jk} = \frac{\sum_{i=1}^{N} E(z_{ij}) x_{ik}}{\sum_{i=1}^{N} E(z_{ij})}$$

2.5.2 Derivation of alignment probabilities

Key Notations:

English vocabulary: V_E

French vocabulary: V_F

No. of sentence pairs (observations): S

Data D which consists of S pairs of parallel sentences looks like:

$e_1^1, e_2^1, e_3^1, ..., e_{l_1}^1 \Leftrightarrow f_1^1, f_2^1, f_3^1, ..., f_{m_1}^1$ - (pair-1: $\boldsymbol{E^1}$, $\boldsymbol{F^1}$)

$e_1^2, e_2^2, e_3^2, ..., e_{l_1}^2 \Leftrightarrow f_1^2, f_2^2, f_3^2, ..., f_{m_2}^2$ - (pair-2: $\boldsymbol{E^2}$, $\boldsymbol{F^2}$)

…

$e_1^s, e_2^s, e_3^s, ..., e_{l_1}^s \Leftrightarrow f_1^s, f_2^s, f_3^s, ..., f_{m_s}^s$ - (s^{th} pair: $\boldsymbol{E^s}$, $\boldsymbol{F^s}$)

…

$e_1^S, e_2^S, e_3^S, ..., e_{l_1}^S \Leftrightarrow f_1^S, f_2^S, f_3^S, ..., f_{m_S}^S$ - (last pair, S^{th} pair: $\boldsymbol{E^S}$, $\boldsymbol{F^S}$)

No. words on English side in s^{th} sentence: l_s

No. words on French side in s^{th} sentence: m_s

$index_E(e_i)$=Index of English word e_i in English vocabulary/dictionary

$index_F(f_j)$ =Index of French word f_j in French vocabulary/dictionary

Hidden Variables (A; the alignment variables):

Total no. of hidden variables= $\sum_{s=1}^{S} l^S m^S$, where each hidden variable is as follows:

$a_{pq}^s = 1$, if in s^{th} sentence, p^{th} English word is mapped to q^{th} French word

=0, otherwise

Parameters (θ) :

Total no. of parameters= $|V_E| \times |V_F|$, where each parameter is as follows:

P_{ij}= Probability that i^{th} word in English dictionary is mapped to j^{th} word in foreign language dictionary, and this is the parameter set $\boldsymbol{\theta}$

Data Likelihood *L(D; θ)*= $\prod_{s=1}^{S} P(f^s \mid e^s)$

Data Likelihood *L(D; θ)*, marginalized over *A* :

$$L(D;\theta) = \sum_A L(D, A;\theta)$$

and,

$$L(D, A;\theta) = \prod_{s=1}^{S} \prod_{m=1}^{m^s} \prod_{l=1}^{l^s} (P_{index_E(e_p^s), index_F(f_q^s)})^{a_{pq}^s}$$

Marginalized Data Log-Likelihood *LL(D,A; θ)* :

$$LL(D, A;\theta) = \sum_{s=1}^{S} \sum_{q=1}^{m^s} \sum_{p=1}^{l^s} \log(P_{index_E(e_p^s), index_F(f_q^s)})^{a_{pq}^s}$$

Expectation of data log likelihood *E(LL(D; Θ))* :

$$E(LL(D, A;\theta)) = \sum_{s=1}^{S} \sum_{q=1}^{m^s} \sum_{p=1}^{l^s} E(a_{pq}^s) \log(P_{index_E(e_p^s), index_F(f_q^s)})$$

Need to find parameter values such that *E(LL(D;θ)* is maximized w.r.t. the following $|V_E|$ constraints:

$$(\sum_{j=1}^{|V_F|} P_{ij} = 1),\ \forall i$$

Introduce Lagrangian for the constraint. Let the Langrange multiplier corresponding to the i^{th} English word's constraint be λ_i.

$$E(LL(D, A;\theta)) = \sum_{s=1}^{S} \sum_{q=1}^{m^s} \sum_{p=1}^{l^s} E(a_{pq}^s) \log(P_{index_E(e_p^s), index_F(f_q^s)}) - \sum_{i=1}^{|V_E|} \lambda_i (\sum_{j=1}^{|V_F|} P_{ij} = 1)$$

Differentiating the above w.r.t. P_{ij}

$$\sum_{s=1}^{S} \sum_{q=1}^{m^s} \sum_{p=1}^{l^s} \delta_{index_E(e_p^s), i} \delta_{index_F(f_q^s), j} \left(\frac{E(a_{pq}^s)}{P_{ij}} \right) - \lambda_i = 0$$

where δ_{ij} is the *Kronecker delta* and is equal to 1 or 0 depending on whether $i=j$ or not, respectively. Now, $P_{i,j}$ can be expressed as,

$$P_{ij} = \frac{1}{\lambda_i} \sum_{s=1}^{S} \sum_{q=1}^{m^s} \sum_{p=1}^{l^s} \delta_{index_E(e_p^s),i} \delta_{index_F(f_q^s),j} E(a_{pq}^s)$$

We know from the constraint that $\sum_{j=1}^{|V_F|} P_{ij} = 1$. Therefore,

$$\sum_{j=1}^{|V_F|} P_{ij} = 1 = \sum_{j=1}^{|V_F|} \frac{1}{\lambda_i} \sum_{s=1}^{S} \sum_{q=1}^{m^s} \sum_{p=1}^{l^s} \delta_{index_E(e_p^s),i} \delta_{index_F(f_q^s),j} E(a_{pq}^s)$$

Therefore,

$$\lambda_i = \sum_{j=1}^{|V_F|} \sum_{s=1}^{S} \sum_{q=1}^{m^s} \sum_{p=1}^{l^s} \delta_{index_E(e_p^s),i} \delta_{index_F(f_q^s),j} E(a_{pq}^s)$$

Then final expression for $P_{i,j}$ is,

$$P_{ij} = \frac{\sum_{s=1}^{S} \sum_{q=1}^{m^s} \sum_{p=1}^{l^s} \delta_{index_E(e_p^s),i} \delta_{index_F(f_q^s),j} E(a_{pq}^s)}{\sum_{j=1}^{|V_F|} \sum_{s=1}^{S} \sum_{q=1}^{m^s} \sum_{p=1}^{l^s} \delta_{index_E(e_p^s),i} \delta_{index_F(f_q^s),j} E(a_{pq}^s)} \qquad -M-step$$

Now we examine the E-step. $E(a_{pq}^s)$ is the expectation that given the s^{th} parallel sentence the p^{th} word from the English side aligns with the q^{th} word in the foreign side. By definition expectation of a random variable is the sum of product of the values of the *r.v.* and the corresponding probability. The random variable a_{pq}^s takes values 0 or 1. Hence,

$$E(a_{pq}^s) = P(a_{pq}^s \mid e^s, f^s) = P(e_p^s \leftrightarrow f_q^s \mid e^s \leftrightarrow f^s)$$
$$= \frac{P(a_{pq}^s, e^s, f^s)}{P(e^s, f^s)}$$
$$= \frac{P(a_{pq}^s, e^s, f^s)}{\sum_x P(a_{px}^s, e^s, f^s)}$$
$$= \frac{P_{index_E(e_p^s),index_F(f_q^s)}}{\sum_{x=1}^{m^s} P_{index_E(e_p^s),index_F(f_x^s)}} \qquad -E-step$$

where, $P_{index_E(e_p^s),index_F(f_q^s)}$ is the probability of mapping between the dictionary entry that matches e_p^s and the dictionary entry that matches f_q^s. Hence, the EM procedure can be summarized as,

M-Step:

$$P_{ij} = \frac{\sum_{s=1}^{S}\sum_{q=1}^{m^s}\sum_{p=1}^{l^s}\delta_{index_E(e_p^s),i}\,\delta_{index_F(f_q^s),j}\,E(a_{pq}^s)}{\sum_{j=1}^{|V_F|}\sum_{s=1}^{S}\sum_{q=1}^{m^s}\sum_{p=1}^{l^s}\delta_{index_E(e_p^s),i}\,\delta_{index_F(f_q^s),j}\,E(a_{pq}^s)}$$

E-Step:

$$E(a_{pq}^s) = \frac{P_{index_E(e_p^s),index_F(f_q^s)}}{\sum_{x=1}^{m^s} P_{index_E(e_p^s),index_F(f_x^s)}}$$

The alignment algorithm will first initialize the P_{ij} values. From these it will get the z_{pq}^s values for $\forall s, p, q$ through the E-step. This will then update the P_{ij} values through the M-step. These new values of P_{ij}s will be used to get new E_{pq}^s values. Such alternations between E-step and M-Step will continue, eventually converging to stable P_{ij} values. These values will correspond to the global minimum, since the data likelihood expression is convex.

2.6 Complexity considerations

Let us get a feel for the complexity of the above EM algorithm, storage and time wise. Our running example of the dictionary contains 500,000 entries. Suppose there are 1 million parallel sentences with 10 words per sentence.

2.6.1 Storage

V_E *x* V_F matrix has 500,000 x 500,000 or 25 x 10^{10} or 250 billion cells. Each cell stores a probability value which is a real number between 0 and 1. If double precision floating point representation is used, each cell will need 16 bytes of storage. The matrix, therefore, would occupy 250 billion x 16 bytes, or 4GB of main memory.

But V_E *x* V_F matrix is extremely sparse. Each word form in V_F on an average maps to 5 word forms on the other side (assumption in section 2.1). Suppose we use adjacency list representation of V_E *x* V_F matrix (Figure 2.4). The example shows 5 common translations of the English word "house". The five translations are: "makaan" meaning "leaving quarters", "raashi" meaning "zodiac", "thietar" meaning "theater", "parivaar" meaning "family" and "sadan" meaning "legislature". "house" thus points to 5 cells each containing a 3-tuple:

$<index_F, address, probability>$

where,

$index_F$: 8 byte integer- the index of the foreign word (e.g. of "makaan")

address: 16 byte pointer to next cell (e.g., to "raashi")

prob: 16 byte double precision float value denoting probability of matching (e.g., *P(makaan|house)*)

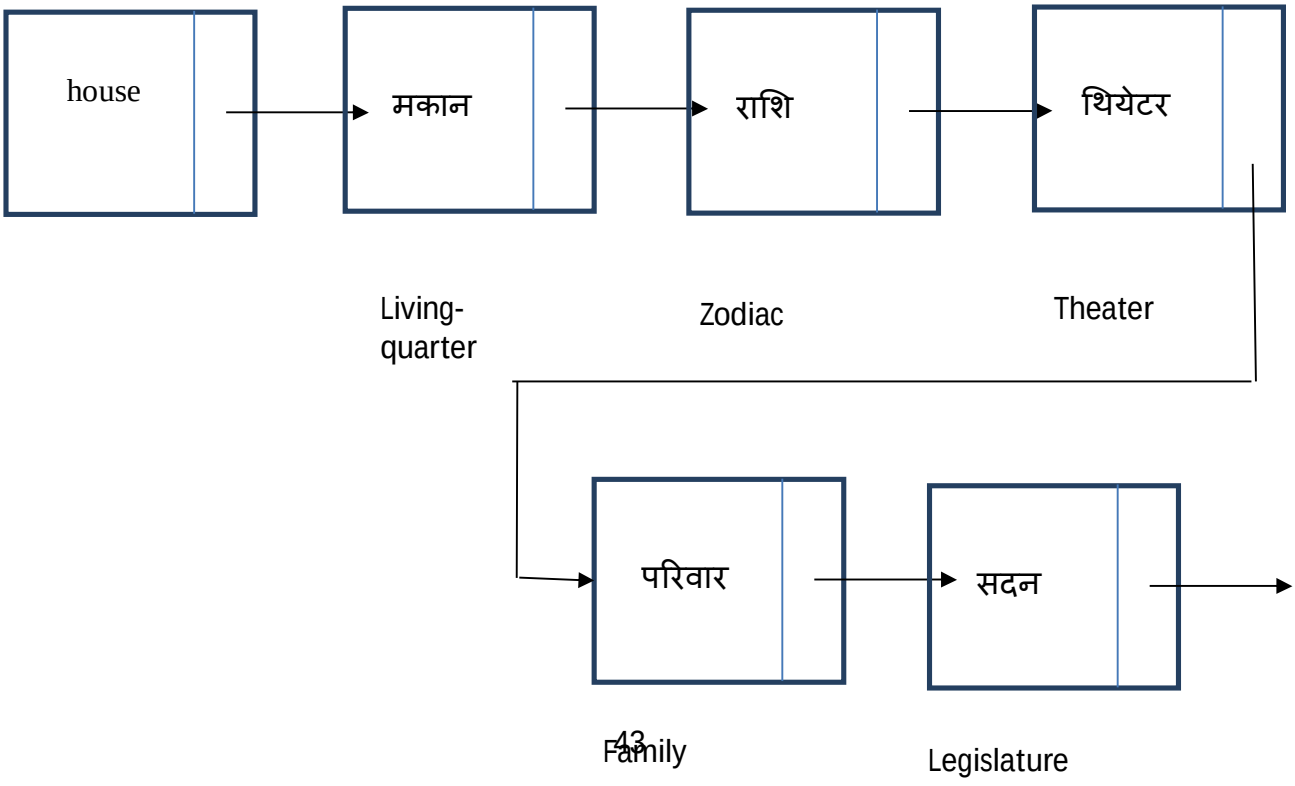

Figure 2.4: adjacency list representation of V_E x V_F matrix

Then each cell needs 8+16+16 bytes= 40 bytes. So 500,000 English words will point to a storage space of (500,000 words x 5 translations/word x 40 bytes/cell)= 10 million bytes or 10 MB of storage. 500,000 English words and 500,000 foreign words need storage for themselves. Assuming average word length of 10 characters and each character requiring a byte (actually foreign language words need more than a byte in Unicode representation), we need 500,000 words x 10 characters/word x 1 byte/character x 2 languages= 10 million bytes or 10 MB. So in total we need about 20 MB of storage for the V_E x V_F matrix represented using adjacency list.

This is a dramatic reduction from 4 GB which we calculated before. However, 4 GB of main memory is easily manageable these days even on desktops. Still it is an insightful exercise studying the storage requirement and ways of economizing it in any computational situation.

Now we turn our attention to the storage requirement for per sentence "count" tables. There are 1 million sentence pairs. Each sentence has 10 words on an average. Here we will use a 10 x 10 matrix for storing the "count" values which are real numbers. With 16 bytes per cell, we will need 1 million sentence-pairs x 100 cells/pair x 16 bytes/cell or 1.6 GB of main memory.

The above method of analysing storage requirement is applicable in other situations. We will see in chapter 4 on Phrase Based SMT that the phrase table learnt from the parallel corpora stores mappings between "phrases" of the two languages. The phrases are not linguistic phrases, but

contiguous sequences of words. Now an *N-word* sentence can give rise to $O(N^2)$ phrases. We use the storage lean adjacency list representation. Assuming average sentence length of 10 words and 5 mappings per phrase on an average, the $V_{eng\text{-}phrase}$ *x* $V_{foreign\text{-}phrase}$ table's storage requirement will shoot up to *1 million sentences x 100 phrases/sentence x 5 mappings/phrase x 40 bytes/mapping* = 20 GB of storage.

SMT indeed is storage intensive. The above ball park estimates vindicate this folklore.

2.6.2 Time

After initializing the 2.5 million V_E x V_F values (5 translations per word for 500,000 dictionary entries), we invoke the E-step. Suppose there are 1 million parallel sentences with 10 words per sentence. Then there are 1,000,000 x 10 x 10 or 10^8 or 100 million z^s_{pq} values. In each E-step these many values are updated (1 million tables with 100 values in each). Consequent updates are posted in the V_E *x* V_F table in the M-step. E and M-steps alternate until convergence. The whole algorithm, of course, is amenable to parallelization through Map-Reduce framework.

> **Exercise 2.4** (systems question): Think of a parallel implementation of the above EM algorithm, for example in MAP-REDUCE. Detail out the process along with hardware requirement and time complexity.

2.7 EM: Study of progress in parameter values

We will get an intuitive feel for the progression of bilingual mapping probabilities in the EM algorithm[5]. As has been emphasized in section 2.1, two example sentences are required for alignment decision: one to introduce the possibility of mapping and another to make it certain. The second sentence can be of the type *one-same-rest-changed* or *one-changed-rest-same*.

2.7.1 Necessity of at least two sentences:

What happens to the probability values when the required at least two sentences are not given? See the parallel sentences in table 2.15. None of the English and French words occur more than once.

English sentence	**French sentence**
I eat	*Je mange*
She dances	*Elle danse*

Table 2.15: parallel sentence with no common words

In the EM algorithm, the probability values get updated in the first iteration. After this they do not change any more, pegging the entropy at 1.0 (table 2.12).

$V_F \rightarrow$ $V_E \downarrow$	*Je*	*Mange*	*Elle*	*Danse*
I	0.5	0.5	0	0

[5] Thanks to Aditya Joshi, PhD student, CSE for writing a scilab program and documenting these observations.

Eat	0.5	0.5	0	0
She	0	0	0.5	0.5
Dances	0	0	0.5	0.5

Table 2. 16: Alignment probabilities frozen at 0.5; average entropy=1.0

2.7.2 One-same-rest-changed situation

We consider again the two parallel sentences "three rabbits ←→ troi lapins" and "rabbits of Grenoble ←→ lapins de Grenoble". The word "rabbits" remains the same in the two parallel sentences, while other words change. Figure 2.5 below shows the monotonic decrease in average entropy. The average entropy cannot go to 0, because the uncertainty of mappings of words other than "rabbits ←→ lapins" cannot be resolved given the current examples.

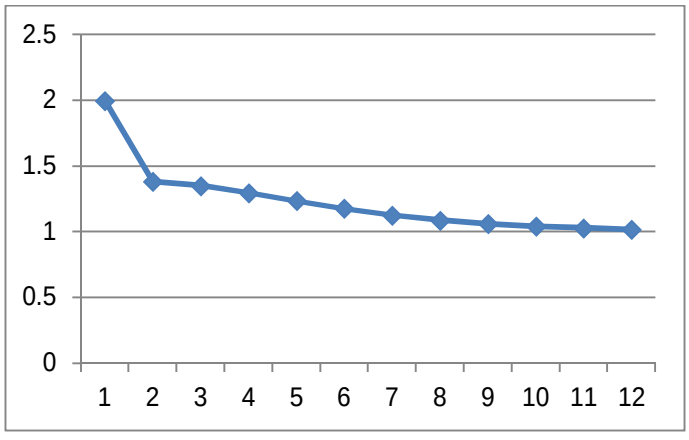

Figure 2.5: X-axis: #iterations, Y-axis: average entropy; average entropy decreases monotonically.

However, the certainty of "rabbits ←→ lapins" mapping goes on increasing.

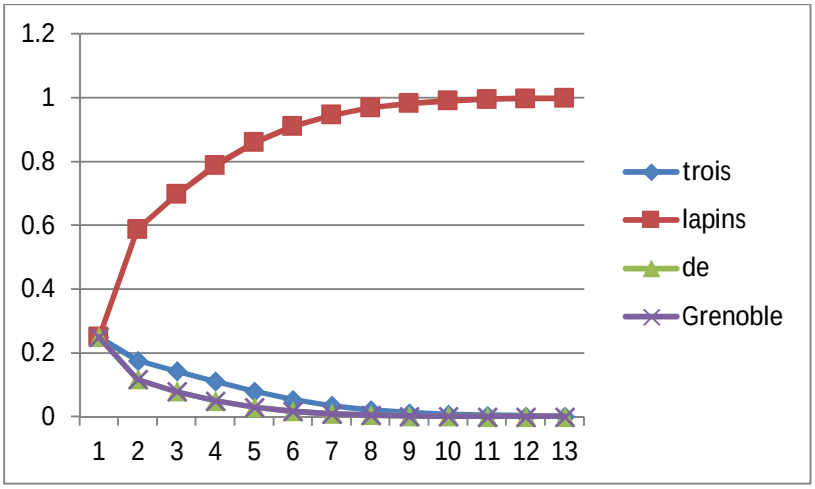

Figure 2.6: X-axis: #iterations; Y-axis: *P(x|rabbit)*, where *x=trois/lapins/de/grenoble*

At the same time the probabilities of other mappings go on decreasing and reach 0.

2.7.3 One-changed-rest-same situation:

English sentence	**French sentence**
Three white rabbits	*Trois lapins blancs*
Three white swans	*Trois cygnes blancs*

Table 2.17: parallel corpora of type one-changed-rest-same

In the above parallel corpora (table 2.17) of two sentence pairs, there is only one change: "rabbits" and "swans". Now the average entropy decreases, but at a slower rate (figure 2.7). At the end of iteration 20, the average entropy is still 1.1855 which is higher than the average entropy for one-same-rest-changed situation at the end of 10 iterations.

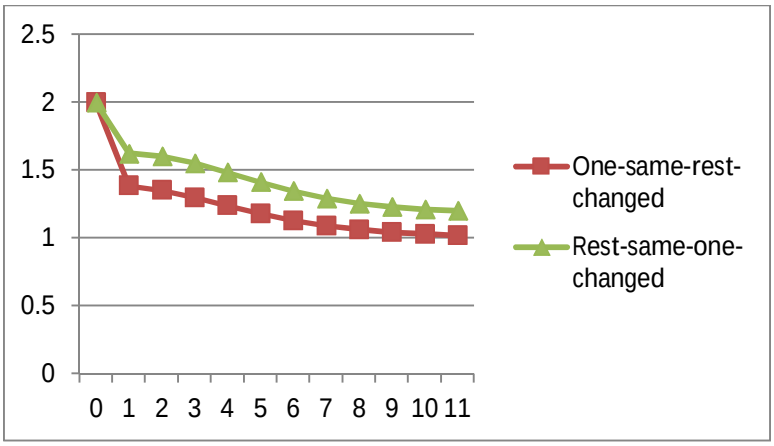

Figure 2.7: decrease in average entropy for one–changed-rest-same situation

This conforms to intuition. In the one-changed-rest-same situation, the uncertainty is higher than that in one-same-rest-changed situation, because the second sentence is almost the same as the first one, except for a single word.

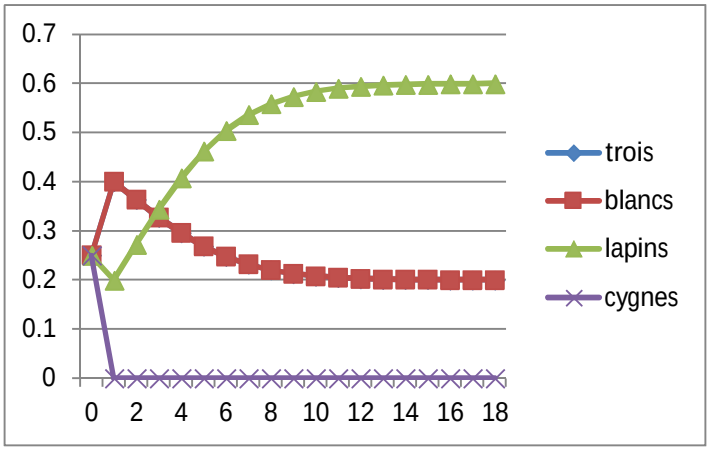

Figure 2.8: X-axis: #iterations, Y-axis: *P(x|rabbits)*, where *x= troi/blancs/lapins/cygnes*

Figure 2.8 shows the way *P(trois/rabbits)*, *P(blancs/rabbits)*, *P(lapins/rabbits)* and *P(cygnes/rabbits)* change with the number of iterations. Interesting observations from figure 2.8 are:

1. Probability of "cygnes ←→ rabbits" drops to 0, since there are no co-occurrences.

2. *P(troi|rabbits)* curve coincides with *P(blancs|rabbist)* curve, since there is nothing to distinguish between the two quantities.
3. At the end of iteration 1, *P(lapins|rabbits)* is lower than either of *P(troi|rabbits)* and *P(blancs|rabbist)*, because of higher frequencies of "trois" and "blancs". But the former soon catches up and wins. *P(troi|rabbits)* and *P(blancs|rabbits)* starves for lack of additional evidence. *P(lapins|rabbits)* soon wins because of higher count in the corpora.
4. But the rate of increase of *P(lapins|rabbits)* is lower; even at the end of iteration 18, the probability is still 0.6.

Chapter Summary

In this chapter, we laid out a simple framework for understanding word alignment. We first define alignment, mentioning its types: *one-to-one (position preserving)*, *one-to-one (non position-preserving)*, *one-to-many*, *many-to-one* and *null*. We then focus on a specific type of alignment that is one-to-one and not necessarily position preserving. This simple situation is very good for understanding the one essential point of alignment computation: the *circularity* in detecting word alignment and word translation, and the way out of the conundrum through expectation maximization (*EM*). Many real and important translation phenomena are kept out of consideration deliberately, so as to keep the above essential point in focus. This discussion formed the introductory part of the chapter.

In section 2.1, we point out the obvious requirement of detecting alignment: at least two sentence pairs are required: *one to introduce the* ***possibility*** *of a particular alignment and another to*

establish its ***certainty***. The second sentence must be of one of the two types: *one-same-rest-changed* and *one-changed-rest-same*. Note, we do not concern ourselves with probability in this section. We calculate (loose) upper bounds on corpora requirement in terms of dictionary size, average number of word mappings and average sentence length. We note that that for a bilingual word list 500,000 words on either side, the corpora requirement is 1.5 million to 2 million, assuming an average sentence length of 10 words and 5 possible mappings per word. This size of corpora requirement is realistic. We get a feel for the grape vine "millions of sentences of parallel corpora are required in SMT".

Section 2.2 proceeds from POS tagging- the representative statistical NLP problem- to an understanding of the fundamentals in SMT. Attention is first drawn to the similarity between statistical POS tagging and statistical MT. Both can use generative framework with an n-gram component (tag modeling/language modeling) and a noisy channel component (lexical mapping/translation mapping). It is in the computation of the mapping component that the two problems differ. While POS tagging can exploit the one to one and positional correspondence between tags and words to compute $P(W|T)$, translation cannot in general do so for $P(F|E)$. The reason is alignment; we do not know which word from the sentence S in L_1 maps to which word in the parallel sentence T in L_2. As a way out, we first give a simple heuristic for alignment computation: use the proportion of translation co-occurrences in parallel sentences computed over the whole corpora. Average entropy of $P(f_j|e_i)$, $i=1\ldots|V_E|$, $j=1\ldots|V_F|$ over the whole $V_E \times V_F$ table is used as the figure of merit. This quantity is not necessarily the lowest when this heuristic is used. We resort to EM which iteratively brings down average entropy, leading to a global minimum for $P(f_j|e_i)$ probabilities, the best that can be done with the given data.

To prepare the ground for the theory of expectation maximization as applied to alignment, we give an example in section 2.4 with two pairs if parallel sentences. The aim is to elucidate the iterative nature of the algorithm, with back and forth movement between the computation of alignment probability on one hand (M-step) and the computation of expected count of such alignments per sentence pair, on the other (E-step). The expected count is used to compute the new alignment probability which in turn is used to update the expected count. The algorithm converges finding the best possible probability values in the cells of the V_E *x* V_F matrix, the parameters of interest in the translation model.

In section 2.5 we establish the theory of the EM based word alignment. We do a build up to the final formula through a series of probabilistic modeling situations: *one coin toss*, *two coins toss*, *multiple outcomes from multiple sources* and finally *alignment modeling*. We emphasize that the probabilities of interest in all these situations are parameters of appropriate probability distributions which model the observation or data likelihood. The parameters found are such that the data likelihood is maximized. As an aside, we mention that the alternative path to parameter estimation could have been entropy maximization, which paradigm is the so called *discriminative modeling*, largely used in phrase based SMT.

The data likelihood expressions in all the above cases are cumbersome without the hidden variable. In alignment, we are actually interested in word translation probabilities, globally. The "within sentence" alignments are hidden. But once these hidden alignment probabilities are

assumed, the data likelihood is expressed elegantly through a multinomial involving hidden alignment probabilities in sentence pairs and global word translation probabilities. With the standard procedure of maximizing the data likelihood *wrt.* parameters and hidden variables, we arrive at expectation and maximization expressions that are consistent with the E and M step expressions illustrated in section 2.4.

Section 2.6 discusses complexity considerations of the EM algorithm. The aim is not to derive lower and upper bounds, but to get a feel for the storage and time requirement. Our numerical figures are the running values of 500,000 word forms in the bilingual word lists, 1 million parallel corpora and 10 words on the average per sentence. The storage is required by the V_E x V_F table of word form storing M-step values, and the "count" tables of parallel sentences storing E-step values. Storage lean data structures like adjacency list can reduce the memory requirement drastically. Still, approximately 2 GB of storage is needed in the setting under consideration. However, more than the actual figures, the method of calculation is instructive. In particular, it turns out that phrase based SMT will need approximately 20 GB of storage in the running example situation.

Section 2.7 gives an experimental feel for the EM algorithm. Of particular interest is the progression of probability values of word alignment. First we observe that if we give parallel sentences where no words are common, the probability values settle to uniform distribution sentence wise, and no further improvement in the values takes place. The *one-same-rest-changed* situation brings down the entropy quickly, wherein the mapping with initial advantage quickly wins into certainty (probability=1.0). The other situation of *one-changed-rest-same* has a slower

rate of entropy decrease and slower increase of the winner probability. Recall from 2.1.1 that *one-same-rest-changed* situation needs more corpora than that needed by *one-changed-rest-same* situation. This is an interesting point. The uncertainty reduction is quicker, but the resource requirement is larger.

Further reading

Though the material on word alignment in the current chapter is easily accessible, for those who are historically inclined and/or are interested in knowing current developments, a reading list is suggested. The reading list is also a help towards how to sequence the study.

It must be apparent that the context for word alignment is statistical MT for which the must-reads first include the two classics:

> Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer & Paul S. Roossin: A statistical approach to machine translation. *Computational Linguistics* 16 (2), pp. 79-85, 1990.

> 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.

Text books on Natural Language Processing now include material on SMT wherein word alignment invariably figures:

Daniel S. Jurafsky and James H. Martin. 2008. Speech and Language Processing. Prentice Hall (chapter 25, sections 5, 6 and 7).

Christopher D. Manning and Hinrich Schutze. 1999. Foundations of Statistical Natural Language Processing. MIT Press (Chapter 13, section 2).

The most referred to text book on SMT:

Philipp Koehn. 2010. Statistical Machine Translation. Cambridge University Press.

has chapter 4 section 5 devoted to word alignment.

We started discussing alignment fundamentals appealing to intuitions in statistical POS tagging for which the two mentioned text books on NLP are excellent resources:

Jurafsky and Martin, 2008 (chapter 5)

Manning and Schutze, 1999 (chapter 10)

Statistical POS tagging typically uses Hidden Markov Model (HMM) for which the classic paper is

L. R. Rabiner and B. H. Juang. 1986. An introduction to hidden markov models. IEEE ASSP Magazine, pages 4–16.

The motivation of first thoroughly understanding one-to-one word alignment comes from

I.D. Melamed. 2000. Models of Translational Equivalence among Words. Computational Linguistics, 26(2): 221-249.

Expectation Maximization formed the algorithmic cornerstone for this chapter. Classic treatises on EM are:

A.P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society: Series B, 39(1):1–38, November 1977.

Geoffrey McLachlan and Thriyambakam Krishnan. 1996. The EM Algorithm and Extensions. John Wiley & Sons, New York.

The tutorial on EM by Sean Borman is also a very lucid exposition.