Statistical Machine Translation

Ph.D. Seminar Report

by

Ananthakrishnan Ramanathan

under the guidance of

Prof. Pushpak Bhattacharyya
and
Dr. M. Sasikumar

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Mumbai

Abstract

Machine Translation (MT) refers to the use of computers for the task of translating automatically from one language to another. The differences between languages and especially the inherent ambiguity of language make MT a very difficult problem. Traditional approaches to MT have relied on humans supplying linguistic knowledge in the form of rules to transform text in one language to another. Given the vastness of language, this is a highly knowledge intensive task. Statistical MT is a radically different approach that automatically acquires knowledge from large amounts of training data. This knowledge, which is typically in the form of probabilities of various language features, is used to guide the translation process. This report provides an overview of MT techniques, and looks in detail at the basic statistical model.

Contents

Chapter 1

Machine Translation: an Overview

Machine Translation (MT) can be defined as the use of computers to automate some or all of the process of translating from one language to another. MT is an area of applied research that draws ideas and techniques from linguistics, computer science, Artificial Intelligence (AI), translation theory, and statistics. Work began in this field as early as in the late 1940s, and various approaches — some *ad hoc*, others based on elaborate theories — have been tried over the past five decades. This report discusses the statistical approach to MT, which was first suggested by Warren Weaver in 1949 [Weaver, 1949], but has found practical relevance only in the last decade or so. This approach has been made feasible by the vast advances in computer technology, in terms of speed and storage capacity, and the availability of large quantities of text data.

This chapter provides the context for the detailed discussion of Statistical MT that appears in the following chapters. In this chapter, we look at the issues in MT, and briefly describe the various approaches that have evolved over the last five decades of MT research.

1.1 Difficulties in Machine Translation

Although the ultimate goal of MT, as in AI, may be to equal the best human efforts, the current targets are much less ambitious. MT aims not to translate literary work, but technical documents, reports, instruction manuals etc. Even here, the goal usually is not fluent translation, but only correct and understandable output.

To appreciate the difficulty of MT, we will look at some examples of language features that are especially problematic from the point of view of translation. All examples in this report are with respect to translation from English to Hindi. Devanagari is represented using a roman encoding that is shown in the appendix.

1.1.1 Ambiguity

Words and phrases in one language often map to multiple words in another language. For example, in the sentence,

> *I went to the bank,*

it is not clear whether the "mound of sand" (*taTa* in Hindi) sense or the "financial institution" (*bEMka*) sense is being used. This will usually be clear from the context, but this kind of disambiguation is generally non-trivial [Nancy and Veronis, 1998].

Also, each language has its own idiomatic usages which are difficult to identify from a sentence. For example,

> *India and Pakistan have* ***broken the ice*** *finally.*

Phrasal verbs are another feature that are difficult to handle during translation. Consider the use of the phrasal verb *bring up* in the following sentences,

> *They brought up the child in luxury.* (**pAlanA**)
>
> *They brought up the table to the first floor.* (**Upara lAnA**)
>
> *They brought up the issue in the house.* (**(muddA) uThAnA**)

Yet another kind of ambiguity that is possible is structural ambiguity:

> *Flying planes can be dangerous.*

This can be translated in Hindi as either of the following two sentences.

> *HavAI jaHAja uDAnA KataranAka Ho saktA HE*
>
> *UDate Hue HavAI jaHAja KataranAka Ho sakte HEM*

depending on whether it is the planes that are dangerous or the occupation of flying them that is dangerous!

1.1.2 Structural Differences

Just as English follows a Subject-Verb-Object (SVO) ordering in sentences, each language follows a certain sentence structure. Hindi, for example, is a Subject-Object-Verb language. Apart from this basic feature, languages also differ in the structural (or syntactic) constructions that they allow and disallow. These differences have to be respected during translation.

For instance, post-modifiers in English become pre-modifiers in Hindi, as can be seen from the following pair of sentences. These sentences also illustrate the SVO and SOV sentence structure in these languages. Here, `S` is the subject of the sentence, `S_m` is the subject modifier, and similarly for the verb (`V`) and the object (`O`).

The president of America will visit the capital of Rajasthan.

```
(S)        (S_m)     (V)      (O)       (O_m)
```

amerikA ke rAXTrapati rAjasthAna kI rAjadhAnI kI sEra karenge

```
(S_m)    (S)        (O_m)     (O)         (V)
```

A complete account of the differences between English and Hindi can be found in [Dave et al., 2001].

1.1.3 Vocabulary Differences

Languages differ in the way they lexically divide the conceptual space, and sometimes no direct equivalent can be found for a particular word or phrase of one language in another.

Consider the sentence,

Tendulkar has edged the ball.

Edge as a verb has no equivalent in Hindi, and this sentence has to be translated as,

TeMDulakara ne balle ke bAHarI Hisse se geMda ko mArA HE

See [Hutchins and Somers, 1992] for more examples of vocabulary differences between languages and also other problems in MT.

1.2 Approaches to Machine Translation

We have seen in the previous section that languages differ in vocabulary and structure. MT, then, can be thought of as a process that reduces these differences to the extent possible. This perspective leads to what is known as the *transfer* model for MT.

1.2.1 The Transfer Approach

The transfer model involves three stages: *analysis*, *transfer*, and *generation*. In the analysis stage, the source language sentence is parsed, and the sentence structure and the constituents of the sentence are identified. In the transfer stage, transformations are applied to the source language parse tree to convert the structure to that of the target language. The generation stage translates the words and expresses the tense, number, gender etc. in the target language.

Figure 1.1 shows the stages in the translation of the sentence

There was a lion in the jungle.

1.1(b) shows the internal representation of this sentence, which is close to English structure, but with the *existential 'there'* removed. The transfer stage converts this according to Hindi word order (1.1(c)). Finally, the generation stage substitutes English words with corresponding Hindi words (1.1(d)).

Note that the analysis component produces a source language dependent representation, and the generation component produces the translated output from a target language dependent representation. Thus, for a multilingual MT system, a separate transfer component is required for each direction of translation for every pair of languages that the system handles. For a system that handles all combinations of n languages, n analysis components, n generation components, and $n \times (n-1)$ transfer components are required.

If the transfer stage can be done away with, say by ensuring that each analysis component produces the same language-independent representation, and that each generation component produces the translation from this very representation, then $n \times (n-1)$ translation systems can be provided by creating just n analysis components and n generation components.

This is precisely the idea behind the *interlingua* approach.

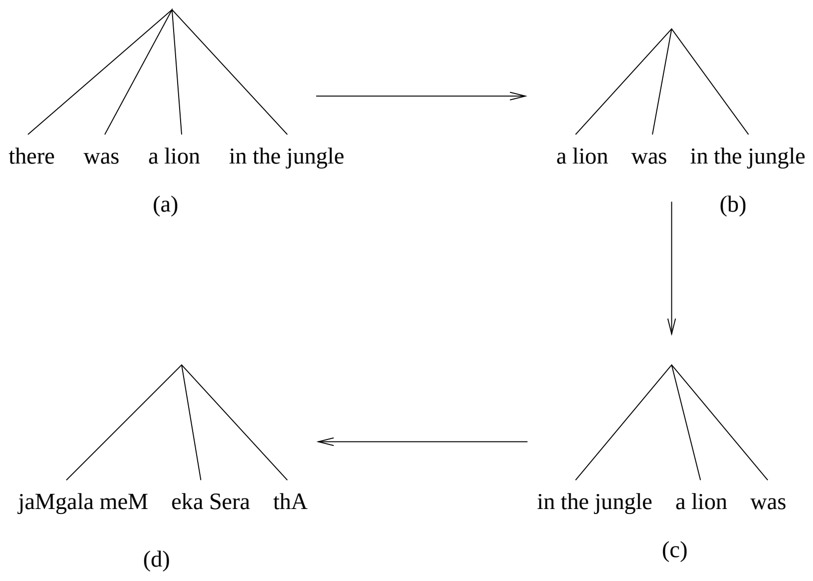

Figure 1.1: Translation using the Transfer model

1.2.2 The Interlingua Approach

The interlingua approach considers MT as a two stage process:

1. Extracting the meaning of a source language sentence in a language-independent form, and,
2. Generating a target language sentence from the meaning.

The language independent form that is used is the interlingua. This form, ideally, allows a canonical representation of any sentence, in the sense that all sentences that mean the same thing are represented in the same way, irrespective of the language.

The interlingual representation expresses the complete meaning of any sentence by using a set of universal concepts and relations. For example, one possible representation for the sentence,

The child broke the toys with a hammer,

is shown in Figure 1.2.

The concepts and relations that are used (known as the *ontology*) are the most important aspect in any interlingua-based system. The ontology should be powerful

PREDICATE	BREAK
AGENT	CHILD number: SING
THEME	TOY number: PLUR
INSTRUMENT	HAMMER number: SING
TENSE	PAST

Figure 1.2: Interlingual represention of *the child broke the toys with a hammer*

enough that all subtleties of meaning that can be expressed using any language (that the system is intended for) should be representable in the interlingua.

Figure 1.3 is the Vauquois pyramid that shows the relation between the transfer and interlingua models.

It is evident that as we move up in the triangle towards an interlingua, the burden on the analysis and generation components increases. We have to choose between various possible parses for a sentence, identify the universal concepts that the sentence refers to, and understand the relations between various concepts expressed in the sentence. But, a translation system for a specific pair of languages may not require so much analysis. This has lead many developers to use what is called the *direct* approach to MT.

1.2.3 The Direct Approach

Consider the sentence,

We will demand peace in the country.

To translate this to Hindi, we do not need to identify the thematic roles universal concepts. We just need to do morphological analysis, identify constituents, reorder them according to the constituent order in Hindi (SOV with pre-modifiers), lookup the words in an English-Hindi dictionary, and inflect the Hindi words appropriately! There seems to be more to do here, but these are operations that can usually be performed more simply and reliably. Table 1.1 shows the steps in the direct translation of this sentence.

Direct translation systems differ from transfer and interlingua systems in that they do minimal structural and semantic (meaning) analysis.

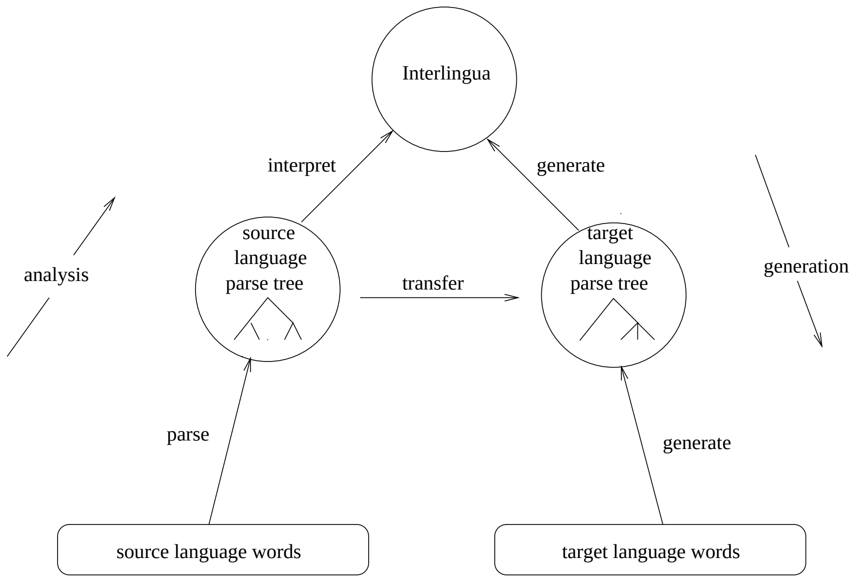

Figure 1.3: Vauquois Pyramid

Of course, the direct approach is rather *ad hoc* and not considered a good long term solution for MT. The linguistically more sophisticated interlingua and transfer methods are the way to go, albeit at a higher initial cost.

[Hutchins and Somers, 1992] and [Jurafsky and Martin,2000] contain good introductions to the transfer, interlingua, and direct approaches to MT.

English Sentence	We will demand peace in the country
Morphological Analysis	We demand FUTURE peace in country
Constituent Identification	<We> <demand FUTURE> <peace> <in country>
Reorder	<We> <in country> <peace> <demand FUTURE>
Dictionary Lookup	*Hama deSa meM SAMtI ki mAnga kara* FUTURE
Inflect	*Hama deSa meM SAMtI ki mAnga kareMge*

Table 1.1: Direct Translation: an example

1.2.4 Corpus-based Approaches

The approaches that we have seen so far, all use human-encoded linguistic knowledge to solve the translation problem. We will now look at some approaches that do not explicitly use such knowledge, but instead use a training corpus (*plur. corpora*) of already translated texts — a parallel corpus — to guide the translation process. A parallel corpus consists of two collections of documents: a source language collection, and a target language collection. Each document in the source language collection has an identified counterpart in the target language collection.

Example Based Machine Translation

This approach to MT uses a technique that human beings use very often to solve problems: they split the problem into subproblems, recall how they solved similar subproblems in the past, adapt these solutions to the new situation, and combine the solutions to solve the bigger problem.

To see how this applies to MT, consider the task of translating the sentence,

> *He bought a huge house.*

using a corpus containing these two sentence pairs:

> (1a) *He bought a pen.*
>
> (1b) *Usane eka kalama KarIdA*
>
> (2a) *All ministers have huge houses.*
>
> (2b) *saBI maMtrioM ke pAsa baHuta baDe Gara HEM*

We try to match parts of the sentence to be translated with the sentences in the corpus. We find that *He bought* matches exactly with the underlined part in (1a). *a huge house* matches with (2a) except that it is in the singular. So, we pick up these two sentence fragments, modify the second for number, combine them appropriately, and get,

Usane eka baHuta baDA Gara KarIdA.

Thus, EBMT entails three steps:

1. Matching fragments against the parallel corpus,
2. Adapting the matched fragments to the target language, and,
3. Recombining these translated fragments appropriately.

[Somers, 1999] is a survey of various EBMT techniques.

Note that even before the EBMT process can begin, corresponding sentences (translation pairs) in the parallel corpora have to be identified. Such a corpus is termed a sentence-aligned parallel corpus. Effective sentence alignment is crucial to all corpus-based MT, including statistical MT. [Gale and Church, 1993], [Kay and Roscheisen, 1993], and [Haruno and Yamazaki, 1996] describe algorithms that have achieved good results for sentence alignment.

Statistical Machine Translation

Statistical MT models take the view that every sentence in the target language is a translation of the source language sentence with some probability. The best translation, of course, is the sentence that has the highest probability. The key problems in statistical MT are: estimating the probability of a translation, and efficiently finding the sentence with the highest probability.

The rest of this report provides a detailed introduction to the basic statistical MT model.

Chapter 2

Statistical Machine Translation: an Overview

One way of thinking about MT is using the noisy channel metaphor. If we want to translate a sentence f in the source language F to a sentence e[1] in the target language E, the noisy channel model describes the situation in the following way:

We suppose that the sentence f to be translated was initially conceived in language E as some sentence e. During communication e was corrupted by the channel to f. Now, we assume that each sentence in E is a translation of f with some probability, and the sentence that we choose as the translation ($\hat{e}$) is the one that has the highest probability. In mathematical terms [Brown et al., 1990],

$$\hat{e} = \underset{e}{arg\,max}\; P(e|f) \tag{2.1}$$

Intuitively, $P(e|f)$ should depend on two factors:

1. The kind of sentences that are likely in the language E. This is known as the *language model* — $P(e)$.

2. The way sentences in E get converted to sentences in F. This is called the *translation model* — $P(f|e)$.

This intuition is borne out by applying Bayes' rule in equation 2.1:

[1]These symbols are common in statistical MT literature, because the first statistical model for translation, described by researchers at IBM, was for French (f) to English (e) translation. In this report, f is used to represent a source language sentence and e, a target language sentence

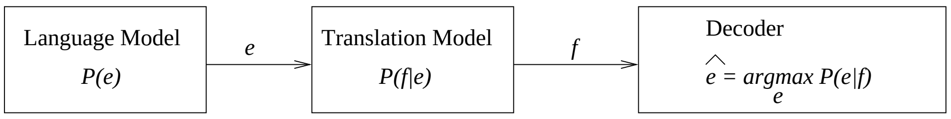

Figure 2.1: The Noisy Channel Model for Machine Translation

$$\hat{e} = \underset{e}{arg\,max}\ \frac{P(e)P(f|e)}{P(f)} \tag{2.2}$$

Since f is fixed, we omit it from the maximization and get 2.3.

$$\hat{e} = \underset{e}{arg\,max}\ P(e)P(f|e) \tag{2.3}$$

This model for MT is illustrated in Figure 2.1.

Why not calculate $P(e|f)$ directly as in 2.1, rather than break $P(e|f)$ into two terms, $P(e)$ and $P(f|e)$, using Bayes' rule. The answer has to do with the way our translation model works, that is, the way we calculate $P(e|f)$ or $P(f|e)$. Practical translation models work by giving high probabilities to $P(f|e)$ or $P(e|f)$ when the words in f are generally translations of the words in e. Also, they do not usually care about whether the words in e go together well. For example, given the sentence,

It is raining.

to be translated to Hindi, the translation model might give equal probabilities to the following three sentences,

bAriSa Ho raHI HE

Ho bAriSa raHI HE

bAriSa Ho raHA HE

This problem is circumvented if we use equation 2.3 instead of 2.1. This is because, though $P(f|e)$ would be the same for the three sentences, the language model would rule out the last two, that is, the first translation would receive a much higher value of $P(e)$ than the other two.

This leads to another perspective on the statistical MT model: the best translation is the sentence that is both faithful to the original sentence and fluent in the target language [Jurafsky and Martin,2000]. This is shown in equation 2.4. What we mean in terms of the above example is that both "*bAriSa Ho raHI HE*" and "*bAriSa Ho raHA HE*" may be considered equally faithful to "It is raining" by some translation model, but the former must prove more fluent according to the language model, and should be chosen as the correct translation.

$$\hat{e} = \underset{e}{arg\,max}\ \underbrace{P(e)}_{fluency}\ \overbrace{P(f|e)}^{faithfulness} \tag{2.4}$$

Now we have a way of finding the best translation given a set of candidate translations using 2.4, but what are these candidate translations? Unlike in our earlier supposition, we cannot practically consider each sentence in the target language. Therefore, we need a heuristic search method that can efficiently find the sentence with (close to) the highest probability.

Thus, statistical translation requires three key components:

1. Language model
2. Translation model
3. Search algorithm

We take up first two components in turn in the next couple of chapters. The last problem is the standard decoding problem in AI, and variants of the Viterbi and A^* algorithms are used in statistical MT to solve this problem.

Chapter 3

Language Modeling using N-grams

Language modeling is the task of assigning a probability to each unit of text. In the context of statistical MT, as described in the previous chapter, a unit of text is a sentence. That is, given a sentence e, our task is to compute $P(e)$.

For a sentence containing the word sequence $w_1 w_2 \dots w_n$, we can write without loss of generality,

$$P(e) = P(w_1 w_2 \dots w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1 w_2) \dots P(w_n|w_1 w_2 \dots w_{n-1}) \tag{3.1}$$

The problem here, and in fact in all language models, is that of *data sparsity.* Specifically, how do we calculate probabilities such as $P(w_n|w_1 w_2 \dots w_{n-1})$? In no corpus will we find instances of all possible sequences of n words; actually we will find only a miniscule fraction of such sequences. A word can occur in just too many contexts (history of words) for us to count off these numbers. Thus, we need to approximate the probabilities using what we can find more reliably from a corpus. N-gram models provide one way of doing this.

3.1 The N-gram approximation

In an N-gram model [Jurafsky and Martin,2000], the probability of a word given all previous words is approximated by the probability of the word given the previous N-1 words. The approximation thus works by putting all contexts that agree in the last N-1 words into one equivalence class. With $N = 2$, we have what is called the *bigram* model, and $N = 3$ gives the *trigram* model.

Though linguistically simple-minded, N-grams have been used successfully in speech recognition, spell checking, part-of-speech tagging and other tasks where language modeling is required. This is because of the ease with which such models can be built and used. More sophisticated models are possible — for example, one that gives each sentence structure a certain probability (using a probabilistic grammar). Such models, however, are much more difficult to build, basically because of the non-availability of large enough annotated corpora. N-gram models, on the other hand, can work with raw corpora, and are easier to build and use.

N-gram probabilities can be computed in a straightforward manner from a corpus. For example, bigram probabilities can be calculated as in equation 3.2.

$$P(w_n|w_{n-1}) = \frac{count(w_{n-1}w_n)}{\sum_w count(w_{n-1}w)} \tag{3.2}$$

Here $count(w_{n-1}w_n)$ denotes the number of occurrences of the the sequence $w_{n-1}w_n$. The denominator on the right hand side sums over all word w in the corpus — the number of times w_{n-1} occurs before any word. Since this is just the count of w_{n-1}, we can write 3.2 as,

$$P(w_n|w_{n-1}) = \frac{count(w_{n-1}w_n)}{count(w_{n-1})} \tag{3.3}$$

For example, to calculate the probability of the sentence, "all men are equal", we split it up as,

$$P(\mathit{all\ men\ are\ equal}) = P(\mathit{all}|\textsc{start})P(\mathit{men}|\mathit{all})P(\mathit{are}|\mathit{men})P(\mathit{equal}|\mathit{are}) \tag{3.4}$$

where START denotes the start of the sentence, and $P(\mathit{all}|\textsc{start})$ is the probability that a sentence starts with the word *all*.

Given the bigram probabilities in table 3.1, the probability of the sentence is calculated as in 3.5.

$$P(\mathit{all\ men\ are\ equal}) = 0.16 \times 0.04 \times 0.20 \times 0.08 = 0.00028 \tag{3.5}$$

Now consider assigning a probability to the sentence, "all animals are equal", assuming that the sequence "all animals" never occurs in the corpus. That is, $P(\mathit{animals}|\mathit{all}) = 0$. This means that the probability of the entire sentence is zero! Intuitively, "all animals are equal" is not such an improbable sentence, and we would like our model to give it a non-zero probability, which our bigram model fails to do.

Bigram	Probability
START all	0.16
all men	0.09
men are	0.24
are equal	0.08

Table 3.1: Bigram probabilities from a corpus

This brings us back to the problem of data sparsity that we mentioned at the beginning of this chapter. We simply cannot hope to find reliable probabilities for *all* possible bigrams from any given corpus, which is after all finite. The task of the language model, then, is not just to give probabilities to those sequences that are present in the corpus, but also to make reasonable estimates when faced with a new sequence.

Mathematically, our model in equation 3.3 uses the technique called Maximum Likelihood Estimation. This is so called because given a training corpus T, the model M that is generated is such that $P(T|M)$ is maximized. This means that the entire probability mass is distributed among sequences that occur in the training corpus, and nothing remains for unseen sequences. The task of distributing some of the probability to unseen or zero-probability sequences is called *smoothing*.

3.2 Smoothing

The simplest smoothing algorithm is *add-one*. The idea here is to simply add one to the counts of all possible sequences — those that occur and also those that do not occur — and compute the probabilities accordingly.

In terms of our bigram model, this means that the probability in equation 3.3 is now,

$$P^*(w_n|w_{n-1}) = \frac{count(w_{n-1}w_n) + 1}{count(w_{n-1}) + V} \tag{3.6}$$

where V is the number of word types in the vocabulary.

Equation 3.6 means that we consider each possible bigram to have occurred at least once in the training corpus. Sequences that do not occur in the corpus will thus have a count of one, and consequently a non-zero probability.

In practice, *add-one* is not commonly used as it ends up giving too much of the probability mass to the sequences that do not occur. This is because the number of N-gram sequences that can occur in any language is a very small fraction of the number of sequences possible (all combinations of two words).

In our earlier example with unsmoothed bigrams, our problem was that the sentence, “All animals are equal” got zero probability because the sequence “all animals” never occurred in the training corpus. This problem is solved by *add-one* smoothing because now “all animals” has a count of one and consequently some non-zero probability. Note, however, that actually impossible sequences such as “the are” and “a animals” and other possible but still unlikely sequences are also given a count of one. Thus *add-one* smoothing does not result in a realistic language model.

The problem that a smoothing algorithm solves can be paraphrased as, “estimating the probability of sequences that we will see for the first time”. One very intuitive trick is to use the number of times we saw a sequence for the first time in our training corpus to guide our computation. If the number of sequence types with w_x as the first word is $T(w_x)$, then the probability of seeing a sequence starting with w_x for the first time is,

$$\sum_{i:count(w_x w_i)=0} P^*(w_i|w_x) = \frac{T(w_x)}{count(w_x) + T(w_x)} \tag{3.7}$$

Now, if there are $Z(w_x)$ N-gram sequences starting with w_x that have a count of zero, then distributing the probability in 3.7 equally amongst them,

$$P^*(w_i|w_x) = \frac{T(w_x)}{Z(w_x)(count(w_x) + T(w_x))} if(count(w_x w_i) = 0) \tag{3.8}$$

Since we have given away some of the probability mass to zero-count sequences, we have to reduce the probability of the seen sequences accordingly:

$$P^*(w_i|w_x) = \frac{count(w_x w_i)}{count(w_x) + T(w_x))} if(count(w_x w_i) > 0) \tag{3.9}$$

The above algorithm for smoothing is called *Witten-Bell discounting* [Witten, 1991], and it has been found much better in practice than the *add-one* method. This is essentially because the count of a sequence is not arbitrarily increased by some number as in *add-one*, but only in proportion to the number of times the beginning of the sequence (the first word in bigrams) occurs.

Chapter 4

Translation Modeling

As discussed in chapter 2, the role of the translation model is to find $P(f|e)$, the probability of the source sentence f given the translated sentence e. Note that it is $P(f|e)$ that is computed by the translation model and not $P(e|f)$ (see equation 2.4 in chapter 2). The training corpus for the translation model is a sentence-aligned parallel corpus of the languages F and E.

It is obvious that we cannot compute $P(f|e)$ from counts of the sentences f and e in the parallel corpus. Again, the problem is that of data sparsity. The solution that is immediately apparent is to find (or approximate) the sentence translation probability using the translation probabilities of the words in the sentences. The word translation probabilities in turn can be found from the parallel corpus. There is, however, a glitch — the parallel corpus gives us only the sentence alignments; it does not tell us how the words in the sentences are aligned.

A word alignment between sentences tells us exactly how each word in sentence f is translated in e. Figure 4.1 shows an example alignment[1]. This alignment can also be written as $(1, (2, 3, 6), 4, 5)$, to indicate the positions in the Hindi sentence with which each word in the English sentence is aligned.

How to get the word alignment probabilities given a training corpus that is only sentence aligned? This problem is solved by using the Expectation-Maximization (EM) algorithm.

[1]The term alignment in this chapter refers to word alignment. Sentence alignment will be referred to in full.

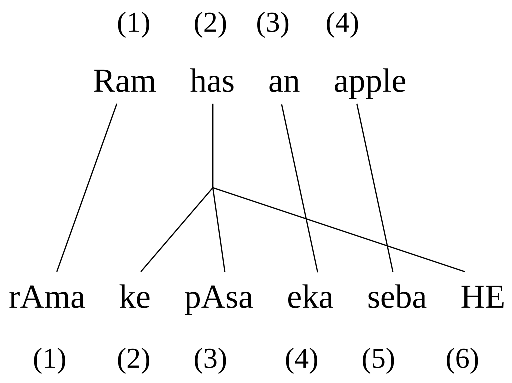

Figure 4.1: An example alignment

4.1 Expectation Maximization: The Intuition

The key intuition behind EM is this: If we know the number of times a word aligns with another in the corpus, we can calculate the word translation probabilities easily. Conversely, if we know the word translation probabilities, it should be possible to find the probability of various alignments.

Apparently we are faced with a chicken-and-egg problem! However, if we start with some uniform word translation probabilities and calculate alignment probabilities, and then use these alignment probabilities to get (hopefully) better translation probabilities, and keep on doing this, we should converge on some good values. This iterative procedure, which is called the *Expectation-Maximization* algorithm, works because words that are actually translations of each other, co-occur in the sentence-aligned corpus.

In the next section, we will formalize the above intuition. The particular translation model that we will look at is known as *IBM Model 1* [Brown et al., 1993].

4.2 IBM Model 1

The notation that is used in the following discussion is summarized in table 4.1

Before going on to the specifics of IBM model 1, it would be useful to understand translation modeling in a general way.

The probability of a sentence f being the translation of the sentence e can be written as,

f	the source language sentence
e	the target language sentence
m	length of f
l	length of e
w^f	a word in f (generally)
w_i^f	the i^{th} word in f
$w_{1,m}^f$	the sequence $w_1^f, w_2^f, \dots w_m^f$
w^e	a word in e (generally)
w_i^e	the i^{th} word in e
$w_{1,l}^e$	the sequence $w_1^e w_2^e \dots w_l^e$
a	a particular alignment between f and e
a_i	the position in e with which the i^{th} word in f is aligned
$a_{1,m}$	the sequence $a_1 a_2 \dots a_m$
Ψ	all possible alignments between f and e

Table 4.1: Notation used to describe IBM Model 1

$$P(f|e) = \sum_{a \in \Psi} P(f, a|e). \tag{4.1}$$

The right hand side in equation 4.1 sums over each way (alignment) in which f can be a translation of e.

The goal of the translation model is to maximize $P(f|e)$ over the entire training corpus. In other words, it adjusts the word translation probabilities such that the translation pairs in the training corpus receive high probabilities.

To calculate the word translation probabilities, we need to know how many times a word is aligned with another word. We would expect to count off these numbers from each sentence pair in the corpus. But, each sentence pair can be aligned in many ways, and each such alignment has some probability. So, the word-alignment counts that we get will be fractional, and we have to sum these fractional counts over each possible alignment. This requires us to find the probability of a particular alignment given a translation pair. This is given by,

$$P(a|f, e) = \frac{P(f, a|e)}{P(f|e)}. \tag{4.2}$$

Substituting from equation 4.1 into 4.2, we have,

$$P(a|f,e) = \frac{P(f,a|e)}{\sum_{a \in \Psi} P(f,a|e)} \tag{4.3}$$

Since we have expressed both $P(a|f,e)$ and $P(f|e)$ in terms of $P(f,a|e)$, we can get a relation between the word translation probabilities and the alignment probabilities by writing $P(f,a|e)$ in terms of the word translation probabilities and then maximizing $P(f|e)$.

Translation models essentially differ in the way they write $P(f,a|e)$. One general way of writing $P(f,a|e)$ is,

$$P(f,a|e) = \underbrace{P(m|e)}_{choose\ length} \prod_{j=1}^{m} \overbrace{P(a_j|a_{1,j-1}, w^f_{1,j-1}, m, e)}^{choose\ position} \underbrace{P(w^f_j|a_{1,j}, w^f_{1,j-1}, m, e)}_{choose\ word}. \tag{4.4}$$

This equation is general except that one word in f is allowed to align with at most one position in e. Words in f can also be aligned with a special *null* position in e indicating that these words have no equivalent in sentence e. An example of such words is case-markers in Hindi, which sometimes have no equivalent in English.

Equation 4.4 says that given the sentence e, we can build the sentence f in the following way:

1. Choose the length m of f
2. For each of the m word positions in f
   (a) Choose the position in e for this position in f. This depends on the positions already chosen, the words already chosen, m, and e.
   (b) Choose the word in f in this position. This depends on the positions already chosen (including the position for this word), the words already chosen, m, and e.

IBM Model 1 is derived from this by making the following simplifying assumptions:

1. $P(m|e)$ is a constant (ϵ) independent of e and m
2. A word in f has the same probability of being aligned with any position, That is,
$$P(a_j|a_{1,j-1}, w^f_{1,j-1}, m, e) = \frac{1}{l+1}.$$

3. The choice of a word depends only on the word with which it is aligned, and is independent of the words chosen so far, m, and e. That is,

$$P(w_j^f|a_{1,j}, w_{1,j-1}^f, m, e) = t(w_j^f|w_{a_j}^e),$$

   where $t(w_j^f|w_{a_j}^e)$ is the translation probability of w_j^f given $w_{a_j}^e$ — the translation probability of the word in f in the j^{th} position given the word in e with which it is aligned in alignment a.

Given these assumptions, we can write $P(f, a|e)$ in Model 1 as,

$$P(f, a|e) = \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} t(w_j^f|w_{a_j}^e). \tag{4.5}$$

Since each of the m words can be aligned with any of the $l + 1$ positions in e, $(l + 1)^m$ alignments are possible, summing over which we have,

$$P(f|e) = \frac{\epsilon}{(l+1)^m} \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} t(w_j^f|w_{a_j}^e). \tag{4.6}$$

As mentioned earlier, our goal is to maximize $P(f|e)$. This is subject to the constraint that the translation probabilities for each word in e sum to 1. That is,

$$\sum_{f} t(w^f|w^e) = 1. \tag{4.7}$$

Introducing Lagrange multipliers λ_{w^e}, we convert the constrained maximization problem above to an unconstrained maximization problem, where we maximize the expression in equation 4.8 below.

$$h(t, \lambda) = \frac{\epsilon}{(l+1)^m} \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} t(w_j^f|w_{a_j}^e) - \sum_{w^e} \lambda_{w^e} (\sum_{f} t(w^f|w^e) - 1) \tag{4.8}$$

The maximum occurs when the partial derivatives of h with respect to the components of t and λ are zero. The partial derivative with respect to λ just gives back equation 4.7. With respect to $t(w^f|w^e)$, the partial derivative is,

$$\frac{\partial h}{\partial t(w^f|w^e)} = \frac{\epsilon}{(l+1)^m}\sum_{a_1=0}^{l}\cdots\sum_{a_m=0}^{l}\sum_{j=1}^{m}\delta(w^f,w^f_j)\delta(w^e,w^e_{a_j})t(w^f|w^e)^{-1}\prod_{k=1}^{m}t(w^f_k|w^e_{a_k})-\lambda_{w^e} \tag{4.9}$$

where δ is the Kronecker delta function, which is equal to one when both its arguments are the same and zero otherwise.

To see why the derivative is so, note that we have differentiated with respect to each word translation probability, $t(w^f|w^e)$. In each alignment where w^f and w^e are aligned (this is taken care of by the Kronecker delta), the derivative consists of all translation probabilities in that alignment, except $t(w^f|w^e)$. This is acheived by multiplying all the translation probabilities in the alignment ($\prod_{k=1}^{m} t(w^f_k|w^e_{a_k})$) and dividing by the translation probability with respect to which we are differentiating ($\mathrm{t}(\mathrm{w}^f|w^e)^{-1}$).

This partial derivative will be zero when,

$$t(w^f|w^e) = \lambda_{w^e}^{-1}\frac{\epsilon}{(l+1)^m}\sum_{a_1=0}^{l}\cdots\sum_{a_m=0}^{l}\sum_{j=1}^{m}\delta(w^f,w^f_j)\delta(w^e,w^e_{a_j})\prod_{k=1}^{m}t(w^f_k|w^e_{a_k}). \tag{4.10}$$

Substituting from equation 4.5 into 4.10,

$$t(w^f|w^e) = \lambda_{w^e}^{-1}\sum_{a\in\Psi}P(f,a|e)\sum_{j=1}^{m}\delta(w^f,w^f_j)\delta(w^e,w^e_{a_j}). \tag{4.11}$$

We mentioned earlier that the word translation probabilities can be found from the fractional counts of the number of times the words are aligned with each other. Note that the counts are fractional because the alignments are not certain but probabilistic. These fractional counts can be written as,

$$c(w^f|w^e;f,e) = \sum_{a\in\Psi}P(a|f,e)\sum_{j=1}^{m}\delta(w^f,w^f_j)\delta(w^e,w^e_{a_j}). \tag{4.12}$$

This is the *count* of w^f given w^e for $(f|e)$. As expected this sums up probability of each alignment where the words co-occur.

From 4.2 and replacing λ_{w^e} by $\lambda_{w^e}P(f|e)$, we can write the intuitive relation between the word translation probabilities and the fractional counts as,

$$t(w^f|w^e) = \lambda_{w^e}^{-1}c(w^f|w^e;f,e). \tag{4.13}$$

Since our parallel corpus contains many sentence pairs, $(f^{(1)}|e^{(1)}), (f^{(2)}|e^{(2)}), \ldots,$ $(f^{(S)}|e^{(S)})$,

$$t(w^f|w^e) = \lambda_{w^e}^{-1} \sum_{s=1}^{S} c(w^f|w^e; f^{(s)}|e^{(s)}). \tag{4.14}$$

The term $\lambda_{w^e}^{-1}$ indicates that the translation probabilities should be normalized.

There is, however, another problem here. Equation 4.12 that calculates the fractional counts requires us to sum over $(l+1)^m$ alignments, which is not at all feasible. With an average sentence length of ten, we would have more than a billion alignments to go over every time! Thankfully, we can simplify this by noting that,

$$\sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} t(w_j^f|w_{a_j}^e) = \prod_{j=1}^{m} \sum_{i=0}^{l} t(w_j^f|w_i^e). \tag{4.15}$$

Substituting this into 4.6, and then evaluating the partial derivative yields,

$$c(w^f|w^e; f, e) = \frac{t(w^f|w^e)}{t(w^f|w_0^e) + \ldots + t(w^f|w_l^e)} \sum_{j=1}^{m} \delta(w^f, w_j^f) \sum_{i=0}^{l} \delta(w^e, w_i^e). \tag{4.16}$$

The number of operations required now is proportional to $(l+m)$ and not $(l+1)^m$.

Now, given a parallel corpus of aligned sentences, we proceed in the following way to estimate the translation probabilities.

1. Start with some values for the translation probabilites, $t(w^f|w^e)$.

2. Compute the (fractional) counts for word translations using 4.16.

3. Use these counts in 4.14 to re-estimate the translation probabilities.

4. Repeat the previous two steps till convergence.

This iterative use of equations 4.16 and 4.14 is the EM algorithm, as mentioned earlier.

[Brown et al., 1993] shows that any initialization (without zero probabilities) of the $t(w^f|w^e)$ parameters leads the above algorithm to converge to the maximum.

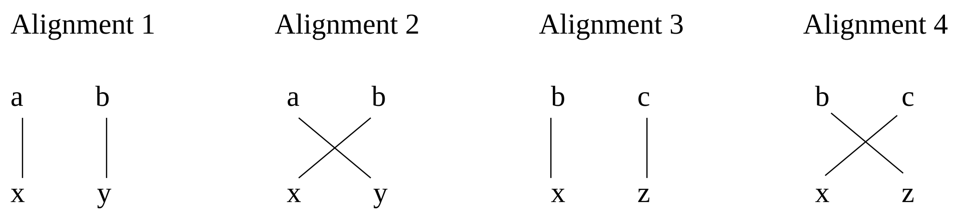

Figure 4.2: Possible word alignments in the parallel corpus

4.3 An Example

An example will help to clarify the working of the EM algorithm. Our example training corpus contains just two aligned sentence pairs: $ab \leftrightarrow xy$, and $bc \leftrightarrow xz$. Here a, b etc., are words, and ab, xy, bc, and xz are sentences. Let us also assume that there are no null-aligned words and that all word correspondences are one-to-one. Note that Model 1 does allow multiple words in f to be translations of a single word in e and also null-aligned words, but these simplifications do not hinder the understanding of the algorithm.

With these simplifications, four alignments are possible. These alignments are shown in figure 4.2.

Since b and x co-occur in the two sentences, and we would expect that b and x are translations of each other. The probabilities after a few iterations of the EM algorithm should confirm this intuition.

We kick-start the algorithm by giving uniform probabilities to all word translations:

$$t(a|x) = t(b|x) = t(c|x) = \tfrac{1}{3}$$

$$t(a|y) = t(b|y) = t(c|y) = \tfrac{1}{3}$$

$$t(a|z) = t(b|z) = t(c|z) = \tfrac{1}{3}$$

Using 4.16 to compute the fractional counts for each word alignment, we have,

$$c(a|x; ab, xy) = \tfrac{1}{2},\;\; c(a|x; bc, xz) = 0$$

$$c(a|y; ab; xy) = \tfrac{1}{2},\;\; c(a|y; bc, xz) = 0$$

$c(b|x;ab;xy) = \frac{1}{2},\;\; c(b|x;bc;xz) = \frac{1}{2}$

and so on for all word pairs.

From these counts, recomputing the translation probabilities using 4.14,

$t'(a|x) = \frac{1}{2} + 0 = \frac{1}{2},\;\; t'(a|y) = \frac{1}{2} + 0 = \frac{1}{2},\;\; t'(a|z) = 0$

$t'(b|x) = \frac{1}{2} + \frac{1}{2} = 1,\;\; t'(b|y) = \frac{1}{2} + 0 = \frac{1}{2},\;\; t'(b|z) = \frac{1}{2} + 0 = \frac{1}{2}$

$t'(c|x) = \frac{1}{2} + 0 = \frac{1}{2},\;\; t'(c|y) = 0,\;\; t'(c|z) = \frac{1}{2} + 0 = \frac{1}{2}$

Note that these probabilities are unnormalized (indicated by t'). Normalizing,

$t(a|x) = \frac{\frac{1}{2}}{\frac{1}{2}+1+\frac{1}{2}} = \frac{1}{4},\;\; t(a|y) = \frac{\frac{1}{2}}{\frac{1}{2}+\frac{1}{2}+0} = \frac{1}{2},\;\; t(a|z) = 0$

and similarly,

$t(b|x) = \frac{1}{2},\;\; t(b|y) = \frac{1}{2},\;\; t(b|z) = \frac{1}{2}$

$t(c|x) = \frac{1}{4},\;\; t(c|y) = 0,\;\; t(c|z) = \frac{1}{2}.$

We now use these new probabilities to recompute the counts, and keep doing this till convergence. After the third iteration the word translation probabilities turn out to be,

$t(a|x) = 0.15,\;\; t(a|y) = 0.64,\;\; t(a|z) = 0$

$t(b|x) = 0.70,\;\; t(b|y) = 0.36,\;\; t(b|z) = 0.36$

$t(c|x) = 0.15,\;\; t(c|y) = 0.00,\;\; t(c|z) = 0.64$

As expected, the co-occurrence of b and x leads the EM algorithm to give increasingly higher values to $t(b|x)$ in each iteration. $t(a|x)$ and $t(c|x)$ keep decreasing as a result of this. The co-occurrence of b and x also means that $t(b|y)$ and $t(b|z)$ keep decreasing, resulting in higher values for $t(a|y)$ and $t(c|z)$.

4.4 Beyond IBM Model 1

Model 1 assumes that each word in f is equally likely to go to any position in e. This also means that Model 1 might decide that all words in f are aligned with a single word (position) in e. Somehow, such alignments have to be filtered out. [Brown et al., 1993] suggests variants of Model 1, which add two more parameters: distortion and fertility.

A distortion probability looks like $d(7|1)$, which gives the probability of the 7^{th} word in f aligning with the first word in e. If such a translation is unlikely in the languages f and e, $d(7|1)$ would be low enough to filter out this alignment. A fertilily probability tells us how many words in f a word in e is likely to correspond to. For example, $(8|lion)$ is the probability that the word "lion" in e corresponds to 8 words in f — this should be a very low probability in any language pair!

The IBM models do not consider structural aspects of language, and it is suspected that these models are not good enough for structurally dissimilar languages. [Yamada and Knight, 2001] propose a syntax-based statistical translation model that includes in addition to word translation probabilities, the probabilities of nodes being reordered and words being added to particular nodes. This can capture SVO to SOV translations and case-marker additions, for example. This model worked better than the IBM models on an English-Japanese corpus.

[Och and Ney, 2001] describes a model that uses a bilingual dictionary along with the parallel corpus to improve word alignments.

These last two approaches represent an active line of work in statistical MT: incorporating linguistic knowledge in the statistical framework.

Chapter 5

Conclusion

Traditional MT techniques require large amounts of linguistic knowledge to be encoded as rules. Statistical MT provides a way of automatically finding correlations between the features of two languages from a parallel corpus, overcoming to some extent the knowledge bottleneck in MT.

The model that we have looked at in this report was entirely devoid of linguistic knowledge, but similar (non-linguistic) models have achieved encouraging results. Researchers believe that introducing linguistic knowledge can further strengthen the statistical model. Such knowledge may be in the form of morphological rules, rules about word-order, idiomatic usages, known word correspondences and so on. Intuitively, for translation between English and Hindi (or any other Indian language) such linguistic knowledge might be crucial because of the vast structural and lexical differences between the two languages.

A major drawback with the statistical model is that it presupposes the existence of a sentence-aligned parallel corpus. For the translation model to work well, the corpus has to be large enough that the model can derive reliable probabilities from it, and representative enough of the domain or sub-domain (weather forecasts, match reports, etc.) it is intended to work for. Another issue is that most evaluation of statistical MT has been with training documents that are very rigid translations of each other (parliamentary proceedings have been widely used). News articles and books, for example, are generally rather loosely translated — one sentence in the source language is often split into multiple sentences, multiple sentences are clubbed into one, and the same idea is conveyed in words that are not really exact translations of each other. In such situations, sentence-alignment itself might be a big challenge, let alone word-alignment.

Since statistical MT is in some sense word alignment (with probabilities), it can be used for lexicon acquisition also, apart from the larger goal of MT.

Statistical MT techniques have not so far been widely explored for Indian languages[1]. It would be interesting to find out to what extent these models can contribute to the huge ongoing MT efforts in the country.

[1]IBM has been working on statistical translation between English and Hindi

Appendix: Rupanthar encoding scheme for Devanagari

a	A	i	I	u	U	R	e	eh	E	o	oh	O
अ	आ	इ	ई	उ	ऊ	ऋ	ए	ऍ	ऐ	ओ	ऑ	औ
M	ah	Mh										
◌ं	◌ः	◌ँ										
k	K	g	G	Gh								
क्	ख्	ग्	घ्	ङ्								
c	C	j	J	Jh								
च्	छ्	ज्	झ्	ञ्								
T	Th	D	Dh	N								
ट्	ठ्	ड्	ढ्	ण्								
t	th	d	dh	n								
त्	थ्	द्	ध्	न्								
p	P	b	B	m								
प्	फ्	ब्	भ्	म्								
y	r	l	v	S	X	s	H					
य्	र्	ल्	व्	श्	ष्	स्	ह्					
L	x[1]	z[2]	?[3]									
ळ्	क्ष्	ज्ञ्	◌़									

---

[1] क्ष् = kX (क् + ष्)

[2] ज्ञ् = jJH (ज् + ञ्)

[3] Unknown

Bibliography

[Brown et al., 1990] Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin, A Statistical Approach to Machine Translation, *Computational Linguistics*, 16(2), pages 79–85, June 1990.

[Brown et al., 1993] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer, The Mathematics of Statistical Machine Translation: Parameter Estimation, *Computational Linguistics*, 19(2), pages 263–311, June 1993.

[Dave et al., 2001] Shachi Dave, Jignashu Parikh, and Pushpak Bhattacharyya, Interlingua-based English-Hindi Machine Translation and Language Divergence, *Journal of Machine Translation (JMT)*, 16(4), pages 251–304, 2001.

[Gale and Church, 1993] W. A. Gale and K. W. Church, A Program for Aligning Sentences in Bilingual Corpora, *Computational Linguistics*, 19(1), pages 75–102, 1993.

[Haruno and Yamazaki, 1996] Masahiko Haruno and Takefumi Yamazaki, High-Performance Bilingual Text Alignment using Statistical and Dictionary Information, *Proceedings of the 34th Conference of the Association for Computational Linguistics*, pages 131–138, 1996.

[Hutchins and Somers, 1992] W. John Hutchins and Harold L. Somers, *An Introduction to Machine Translation*, Academic Press, 1992.

[Jurafsky and Martin,2000] Daniel Jurafsky and James H. Martin, *Speech and Language Processing*, Pearson Education Inc, 2000.

[Kay and Roscheisen, 1993] Martin Kay and Martin Roscheisen, Text-Translation Alignment, *Computational Linguistics*, 19(1), pages 121–142, 1993.

[Manning and Schutze, 2000] Christopher D. Manning and Hinrich Schutze, *Foundations of Statistical Natural Language Processing*, The MIT Press, 2000.

[Nancy and Veronis, 1998] I. Nancy and J. Veronis Word Sense Disambiguation: The State of the Art *Computational Linguistics*, 24(1), 1998.

[Och and Ney, 2001] Franz Josef Och and Herman Ney, A Comparison of Alignment Models for Statistical Machine Translation, *Proceedings of the 17th Conference on Computational Linguistics*, pages 1086–1090, 2000.

[Somers, 1999] Harold L. Somers, Example-based Machine Translation, *Machine Translation*, 14, pages 113–157, 1999.

[Weaver, 1949] Warren Weaver, Translation, *Machine Translation of Languages: Fourteen Essays*, William Locke and Donald Booth (eds), pages 15–23, 1955.

[Witten, 1991] I. H. Witten and T. C. Bell, The Zero-Frequency Problem: Estimating the Probabilities of Novel Events in Adaptive Text Compression., *IEEE Transactions on Information Theory*, 37(4), pages 1085–1094, 1991.

[Yamada and Knight, 2001] Kenji Yamada and Kevin Knight, A Syntax-based Statistical Translation Model, *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*, 2001.