Gaussian Alignments in Statistical Translation Models

by

Ali Mohammad

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Masters of Science in Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2006

Author .....
Department of Electrical Engineering and Computer Science
January 9, 2006

Certified by .....
Michael J. Collins
Associate Professor
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Graduate Head

Gaussian Alignments in Statistical Translation Models

by

Ali Mohammad

Submitted to the Department of Electrical Engineering and Computer Science
on January 9, 2006, in partial fulfillment of the
requirements for the degree of
Masters of Science in Computer Science

Abstract

Machine translation software has been under development almost since the birth of the electronic computer. Current state-of-the-art methods use statistical techniques to learn how to translate from one natural language to another from a corpus of hand-translated text. The success of these techniques comes from two factors: a simple statistical model and vast training data sets. The standard agenda for improving such models is to enable it to model greater complexity; however, it is a byword within the machine learning community that added complexity must be supported with more training data. Given that current models already require huge amounts of data, our agenda is instead to simplify current models before adding extensions. We present one such simplification, which results in fewer than 10% as many alignment model parameters and produces results competitive with the original model. An unexpected benefit of this technique is that it naturally gives a measure for how difficult it is to translate from one language to another given a data set. Next, we present one suggestion for adding complexity to model new behavior.

Thesis Supervisor: Michael J. Collins
Title: Assosciate Professor

Acknowledgments

I would like to begin by thanking my advisor, Mike Collins, with whom I have had many fruitful discussions and who has been very patient and supportive during this time.

I would also like to thank Jason Rennie and Sam Roweis, who have helped me out of tough spots and told me when I was trying to do uncomputable things in polynomial time (which is apparently very impossible).

Many thanks go to my family for being very patient with me during this time. Of course, my parents, Ahlam and Hassan, deserve thanks for their encouragement and support. My little sister Asma deserves thanks for the many hours she spent translating German and evaluating alignments. Finally, my older sister Duaa deserves thanks for dealing with home issues, helping with laundry, cooking, understanding when I would disappear for several days to my office, and, most of all, for dealing with my crankiness with infinite patience these past months.

My (current and former) office-mates also deserve thanks for dealing with my excessively loud music, many phone calls, and general good humor: Yuan Shen, Vineet Sinha, Alex Gruenstein, Harr Chen, Meg Aycinena, Percy Liang, Terry Koo, and Angelina Lee.

I would also like to thank Jeremy Fineman for encouraging me to keep going to the gym while working on my thesis to make use of my frustration and to avoid gaining a thesis gut of truly prodigious girth, and also for making fun of me when I wouldn't go. It is thanks to him that I gained only twenty pounds during this time.

Thanks go to the entire gang at CSAIL, but particularly to Federico Mora and Daniel Loreto, who passed many a night at lab with me sharing stories, singing songs, and finding out which restaurants would deliver after two in the morning (pretty much just Sicilia's, though 7-11 is 24 hours and just down the street).

Finally, I would like to remember my extended family in Iraq, who have had to deal with much greater burdens than a thesis, and have done so bravely and patiently for the past sixteen years.

Contents

1 Introduction **9**

1.1 Motivation 9

1.2 Background 9

1.2.1 Five IBM Models 11

1.2.2 Phrase-Based Models 14

2 One-Dimensional Gaussian Alignments **17**

2.1 Motivation 17

2.2 Algorithm 21

2.3 Implementation and Evaluation 26

3 Multivariate Gaussian Alignments **30**

3.1 Motivation 30

3.2 Algorithm 31

3.2.1 Sampling 32

3.2.2 Sampling from Weighted Multivariate Gaussians 35

3.3 Implementation and Evaluation 38

4 Conclusions and Future Work **42**

A EM Algorithm Reference **44**

A.1 Definitions 44

A.2 Lemmas 44

A.3 EM is Nondecreasing 45

A.4 EM on Multinomials 46
A.5 Parametrizing Multinomials 47
A.6 EM on IBM2+1dG 48

List of Figures

1-1 Typical states in the Köhn decoder. In the original state, depicted above, the decoder has hypothesised that the French phrase f_3 f_4 corresponds with the English phrase e_1 e_2 with score S. This state is updated by adding that the phrase f_5 f_6 corresponds to the English phrase e_3 e_4 with score terms corresponding to this assignment (Score($e_3, e_4; f_5, f_6$)), log likelihood terms corresponding to the language model, and a score for the phrase reordering. 16

2-1 The length of English sentences drawn from the English translation of the proceedings of the European Parliament appears to have a Gaussian distribution. 18

2-2 (a) The length of English sentences whose German translations are twenty-five words long appears to have a Gaussian distribution. (b) The length of sentences in English and German appears to have a bivariate Gaussian distribution. All things are indeed Gaussian. . . . 19

2-3 Our motivation: The indices of aligned words appear to be (roughly) Gaussian. At the very least, it seems like a safe approximation. . . . 20

2-4 The Vanilla Gaussian Algorithm. 23

2-5 The performance of the alignments induced by the one-dimensional gaussian model is similar to those induced by the IBM-2 model. This graph shows the BLEU metric of the two models when applied to the EUROPARL training data and standard test set for the German-English language pair. 28

3-1 The covariance matrix. Graph (a) shows the actual values of the cells. Graph (b) shows the values versus the distance between the cell indices. Clearly values off the tridiagonal are nearly zero. 39

3-2 Alignments of four German-English sentence pairs. The dotted lines indicate alignments common to the output of both the univariate and multivariate models; the dashed line is only in the univariate model and the solid line is only in the multivariate model. (a) Improved pronoun alignment. Due to the covariance of neighboring words, the multivariate model aligns the German pronoun "sie" correctly. (b) Improved multiple particle alignment. Multiple instances of a particle (or similar particles) are aligned correctly in the multi-variate model despite the presence of multiple high-scoring candidate words in the English sentence, which confuses the univariate Gaussian model. (c) Article misalignment. Due to the frequency with which German articles appear without English analogs, German articles sometimes align to the English word that corresponds to the German word that it modifies even if there is also a corresponding English article. (d) Infinitive misalignment. It is not clear why this happens, though we imagine that frequently occurring verbs (such as "habe") occur so frequently that apparently pronouns are likely candidates for translation in the translation model. Since the last word in a German sentence often aligns to a word far from the second-to-last word, "habe" tried to align to a word away from the word that "angesprochen" aligned to. 41

List of Tables

2.1 BLEU score versus Normalized Language Cross-Variance (NLCV). We expect that translation quality will decrease as cross-variance increases; furthermore, we expect that BLEU scores increase with increasing translation quality. Thus, we expect that BLEU scores decrease with increasing cross-variance. This graph suggests this expected trend. The BLEU scores here were taken from [5]; our metric was measured on 10,000 sentence pairs in each case taken from the EUROPARL corpus, the same corpus used to train and test the phrase-based model in [5]. The values in the table are NLCV·10^5/BLEU·10^4. 29

3.1 Results of the double-blind trial on alignments generated by 101 sentence pairs using 1dG (the one-dimensional Gaussian technique of the last chapter) and MdG (the multi-variate Gaussian technique of this chapter) models trained on 100,000 sentence pairs of the EUROPARL corpus. 38

Chapter 1

Introduction

1.1 Motivation

Machine translation could use a lot of work. Much progress has been made in the last fifteen years; we have moved all the way from massive and complex rule-based systems to (relatively) simple statistical systems based on large amounts of data. Much research since the initial IBM models has been on adding complexity in exchange for increased accuracy. Unfortunately, it is a game of diminishing returns and the later IBM models are already somewhat complex.

Therefore, we will begin our exploration by adding complexity to the early IBM models according to a different agenda.

1.2 Background

Let us state the machine translation problem: our goal is to translate French sentences to English sentences. We will always denote sentences in the source language by $\mathbf{f}$ and sentences in the target language by $\mathbf{e}$. m is the number of words in the source sentence $\mathbf{f}$ and ℓ is the number of words in the target sentence $\mathbf{e}$. Now, in Bayesian terms, given a French sentence $\mathbf{f}$, we wish to find the English sentence $\mathbf{e}$ that maximizes $\Pr(\mathbf{e} \mid \mathbf{f})$, in effect imagining that French sentences are generated by some unknown

transformation on English sentences. Using Bayes' Law, we write:

$$\begin{aligned}\arg\max_{\mathbf{e}} \Pr(\mathbf{e} \mid \mathbf{f}) &= \arg\max_{\mathbf{e}} \frac{\Pr(\mathbf{f} \mid \mathbf{e}) \Pr(\mathbf{e})}{\Pr(\mathbf{f})} \\ &= \arg\max_{\mathbf{e}} \Pr(\mathbf{f} \mid \mathbf{e}) \Pr(\mathbf{e}) .\end{aligned}$$

The $\Pr(\mathbf{f})$ term can be ignored since $\mathbf{f}$ is constant. The first term, $\Pr(\mathbf{f} \mid \mathbf{e})$, is called the "translation model" and the second, $\Pr(\mathbf{e})$, is called the "language model." Our work focuses on improving the translation model, because one has to start somewhere. It is just good to know that our translation model does not have to worry about assigning low probabilities to English sentences that look like they could be translations of $\mathbf{f}$ but don't really look like they could be English sentences; a good language model can make up for some deficiencies in the translation model. One other advantage of factoring the model in this way is that the language model can be trained on very large unlabeled (i.e., untranslated) data sets in the target language (note, however, that if we do this, justification by way of Bayes' law is no longer technically valid; we instead say that $P(\mathbf{f} \mid \mathbf{e})$ and $P(\mathbf{e})$ approximate their values in the limit of infinite data and, in that limit, Bayes' law is again valid).

In this work, we use a trigram language model; that is, we assume that the procedure that produces English sentences is a Markov process with a history of two words: the probability of an English sentence $\mathbf{e}$ is broken down like this:

$$\Pr(\mathbf{e}) = \Pr(e_1, e_2, \cdots, e_\ell) = \prod_{i=1}^{\ell+1} \Pr(e_i \mid e_{i-1}, e_{i-2}) ,$$

where $e_{\ell+1}$ is a stop symbol implicitly included at the end of every English sentence. This is a raw trigram model. To learn the parameters, we can simply count the number of times each trigram appears in a corpus:

$$\Pr(e \mid e_2, e_1) = \frac{count(e_1, e_2, e)}{count(e_1, e_2)}$$

where $count(\cdots)$ denotes the number of times the words $\cdots$ appear together in the

corpus in the given order. One problem with this model is that it will assign a zero probability to any sentence that has a trigram that was never seen in the corpus. To fix this, one uses a *smoothed* trigram model [3]:

$$\Pr(e \mid e_2, e_1) = \alpha_t \Pr_t(e \mid e_2, e_1) + \alpha_b \Pr_b(e \mid e_2) + \alpha_m \Pr_m(e)$$

where $\alpha_t + \alpha_b + \alpha_m = 1$ (and the αs are nonnegative) and $\Pr_t(\cdot)$, $\Pr_b(\cdot)$, and $\Pr_m(\cdot)$ denote trigram, bigram, and unigram probabilities, respectively.

1.2.1 Five IBM Models

In a seminal 1993 paper, Brown *et al* introduced a set of five machine translation systems based on simple statistical models (under the "noisy-channel" framework described above) and large parallel corpora. The later models are significantly more complex than the earlier models, with each subsequent model corresponding to an increase in complexity and improved accuracy (but with diminishing returns in the later models) [1].

We assume that we have at our disposal a corpus of N pairs of sentences $(\mathbf{e}^{(1)}, \mathbf{f}^{(1)})$, $(\mathbf{e}^{(2)}, \mathbf{f}^{(2)})$, ..., $(\mathbf{e}^{(N)}, \mathbf{f}^{(N)})$.

In this document, our analysis will not extend beyond the first and second IBM Models, so we will limit our discussion of the later models to a brief overview.

Model 1

We begin by describing an idea fundamental to both Model 1 and Model 2: an *alignment* between a pair of sentences $\mathbf{f}$ and $\mathbf{e}$ is an ordered set $a_1, a_2, \cdots, a_m \in \{0, 1, \ldots, \ell\}$. The French word f_j is said to be *aligned* to the English word e_{a_j} (where e_0 denotes a fake word "NULL" that is used to explain function words that may not have an obvious analog in the English sentence). Notice that we don't demand that English words are aligned to French words; a single English word could be used to explain an entire French sentence (a developed model would declare such an alignment as very improbable, however).

Model 1 makes the following assumptions/approximations:

- All alignments are equally likely.
- All French sentence lengths m are equally likely (we will ignore the obvious problem that there are infinitely many French sentence lengths[1]; if it bothers you, you can assume that someone gives you the length or that there are only finitely many possible French sentence lengths, which is true in practice, anyway). We will generally omit this term.
- Each word is translated independently of the other words.

These assumptions sound ridiculous, but it is important to start with a tractable model. It is also important to remember that our language model will clean up output problems: we can reasonably expect short-range alignment problems and, to some extent, poor grammar to be dealt with there.

Ultimately, we obtain the following formulation:

$$\begin{aligned}
\Pr(\mathbf{f} \mid \mathbf{e}) &= \frac{1}{(\ell+1)^m} \sum_{\mathbf{a}} \prod_{j=1}^{m} \Pr(f_j \mid e_{a_j}) \\
&= \frac{1}{(\ell+1)^m} \sum_{a_1=0}^{\ell} \sum_{a_2=0}^{\ell} \cdots \sum_{a_m=0}^{\ell} \prod_{j=1}^{m} \Pr(f_j \mid e_{a_j}) \\
&= \frac{1}{(\ell+1)^m} \prod_{j=1}^{m} \sum_{a_j=0}^{\ell} \Pr(f_j \mid e_{a_j}).
\end{aligned}$$

Here, a_j is the index of the English word that is *aligned* to the jth French word; Model 1's parameters are the translation probabilities $P(f \mid e)$ (the probability that the French word f was generated by the English word e). Model 1 is an excellent candidate for optimization by EM; it is convex and has only one local maximum (outside of saddle points due to symmetry) so given a random starting point, it will always converge to the same, optimum translation table. Some more math gives us

[1]There really are. As proof, I present a regular expression that matches an infinite number of grammatical French sentences: *Je suis un très* grand singe.* Obvious analogs exist in other languages.

the following update rules:

$$T'(f \mid e) \;=\; \frac{1}{Z_T(e)} \sum_{\substack{i,j,k \\ \mathbf{e}_i^{(k)}=e, \mathbf{f}_j^{(k)}=f}} \frac{T(\mathbf{f}_j^{(k)} \mid \mathbf{e}_i^{(k)})}{\sum_{i'=0}^{\ell} T(\mathbf{f}_j^{(k)}, \mathbf{e}_{i'}^{(k)})},$$

where Z_T is a normalization constant.

Model 2

In Model 2, we wish to relax Model 1's assumption that all alignments are equally likely. However, we will assume for simplicity that the words all "move" independently; that is, which English word a French word is aligned to is independent of the alignment of the remaining words. Here is the formulation of Model 2:

$$\begin{aligned} \Pr(\mathbf{f} \mid \mathbf{e}) \;&=\; \sum_{\mathbf{a}} \prod_{j=1}^{m} \Pr(f_j \mid e_{a_j}) \Pr(a_j \mid j, \ell, m) \\ &=\; \prod_{j=1}^{m} \sum_{a_j=0}^{\ell} \Pr(f_j \mid e_{a_j}) \Pr(a_j \mid j, \ell, m). \end{aligned}$$

Here, in addition to the translation probabilities $\Pr(f \mid e)$ Model 2 inherits from Model 1, we find alignment probabilities, $\Pr(a_j \mid j, \ell, m)$ (the probability that, given a French sentence of length m that is the translation of an English sentence of length ℓ, the jth French word was generated by the a_jth English word). Model 2 is not as good a candidate for EM as Model 1 was; it is riddled with saddle points and local maxima. Typically, one initializes the translation parameters by training Model 1 before training Model 2, whence we use the following update rules:

$$T'(f \mid e) \;=\; \frac{1}{Z_T(e)} \sum_{\substack{i,j,k \\ \mathbf{e}_i^{(k)}=e, \mathbf{f}_j^{(k)}=f}} \frac{D(a_j = i \mid j, \ell, m) T(\mathbf{f}_j^{(k)} \mid \mathbf{e}_i^{(k)})}{\sum_{i'=0}^{\ell} D(a_j = i' \mid j, \ell, m) T(\mathbf{f}_j^{(k)}, \mathbf{e}_{i'}^{(k)})}$$

$$D'(a_j = i \mid j, \ell, m) \;=\; \frac{1}{Z_D(j, \ell, m)} \sum_{\substack{k \\ |\mathbf{e}^{(k)}|=\ell, |\mathbf{f}^{(k)}|=m}} \frac{D(a_j = i \mid j, \ell, m) T(\mathbf{f}_j^{(k)} \mid \mathbf{e}_i^{(k)})}{\sum_{i'=0}^{\ell} D(a_j = i' \mid j, \ell, m) T(\mathbf{f}_j^{(k)}, \mathbf{e}_{i'}^{(k)})},$$

where Z_T and Z_D are normalization constants.

Models 3, 4, and 5

IBM Model 3 introduces fertility parameters, modelling the number of French words a single English word generates. IBM Model 4 introduces distortion parameters to the alignment models as a way of encouraging words to move in groups. Both of these models are formulated "deficiently": that is, they assign probability mass to impossible French sentences (four-word French sentences without a third word, for instance); Model 5 is the non-deficient version of Model 4. Since this makes little empirical difference and is a great computational burden, Model 5 is rarely used in practice [8].

1.2.2 Phrase-Based Models

The primary unit of information in all of the systems we have described up to this point is the word; in phrase-based systems, the primary unit of information is the phrase, a collection of (lexically) consecutive words and the lexical entry in a phrase-based system is a triple containing a source phrase $f_1 \dots f_n$, a target phrase $e_1 \cdots e_m$, and a score $s \in [0, 1]$. That is, instead of considering probabilites of word-to-word translations and word-movement, a phrase-based system will deal with probabilities of phrase-to-phrase translations and phrase-movement. There is a great deal of evidence to suggest that machine translation systems generally experience a performance boost by making this change.

Some phrase-based models, such as those in [7], simply introduce mechanisms for phrase-to-phrase translations and invent policies to assign probability mass to phrase-to-phrase translations. Others, such as those in [6], build a dictionary of phrases from other information sources. Our experiments are centered on the Köhn system, as it achieves state-of-the-art performance.

There are a number of ways one can build phrase dictionaries depending on the data available. Phrases can be built from word-based alignments (such as those gen-

erated by the IBM Models). If syntactic information is available, it can be used to restrict our attention to syntactic phrases; although it seems that syntactic phrases may be more useful, experiment suggests that phrases that are not syntactically motivated are, generally, just as useful. Furthermore, even weighting syntactic phrases produces virtually no improvement at best and is sometimes harmful. Phrase dictionaries can also be built from phrase-aligned data generated from phrase-based systems. Again, we can place more confidence in these phrases if we wish, but generally, the lesson from the experiments with syntactic phrases is applicable: it is better in practice to simply consider as many phrases as possible than to restrict our knowledge to satisfy any bias we may have [5].

Experiments by Köhn *et al* show that simple heuristic methods based on word-based alignments from the IBM models generate state-of-the-art translations. To generate a phrase dictionary, he begins by observing that the IBM models are not symmetric; the alignments generated by a model trained to translate from French to English can be different from alignments generated by a model trained to translate from English to French (in fact, it is often impossible for alignments generated in one direction to match those generated in the other direction, due to inherent restrictions of the IBM models). Köhn's method begins by considering the intersection of the two alignments as a starting point for the phrases it must generate; that is, it begins by suggesting that words that are aligned in both models are probably related. Next, Köhn uses a growing technique to induce the phrase dictionary; phrase dictionaries generated in this fashion tend to be very large because of the generality of this technique; however, the method naturally also generates scores and generates many very low-scoring phrase pairs.

Decoding is done using an algorithm described in [4]. The output sentence is generated left-to-right in the form of partial translations which are formed using a beam-search algorithm including scores for the phrase translations and for the language model, rewards for increased sentence length, and penalties for phrase reordering. Typical states are depicted in Figure 1-1.

The two factors that govern the quality of translations generated using this tech-

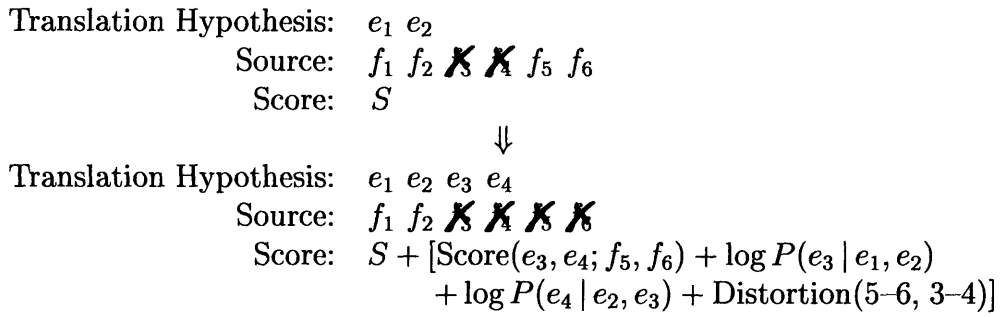


Figure 1-1: Typical states in the Köhn decoder. In the original state, depicted above, the decoder has hypothesised that the French phrase f_3 f_4 corresponds with the English phrase e_1 e_2 with score S. This state is updated by adding that the phrase f_5 f_6 corresponds to the English phrase e_3 e_4 with score terms corresponding to this assignment ($\text{Score}(e_3, e_4; f_5, f_6)$), log likelihood terms corresponding to the language model, and a score for the phrase reordering.

nique are the quantity and quality of the alignments it is fed during training. In practice, translation quality from this method is significantly better than any of the IBM Models. Surprisingly, it does almost as well with IBM Model 2 alignments as with IBM Model 4 alignments.

Chapter 2

One-Dimensional Gaussian Alignments

2.1 Motivation

Assuming that a random variable is Gaussian is a natural choice when the distribution is unknown, because Gaussians have many nice properties. In particular, they are an especially good choice for use in EM algorithms since the maximum likelihood estimates for a Gaussian can be written in closed form. Furthermore, assuming that a variable is Gaussian is often a good approximation due to the Central Limit Theorem, which states that a sum of independent and identically distributed variables with finite mean and variance tends to be Gaussian as the number of addends approaches infinity, and particularly due to the fuzzy Central Limit Theorem, which states that data influenced by many independent sources of noise are roughly normally distributed [13].

For instance, the number of words in English sentences in the EUROPARL corpus, depicted in Figure 2-1, is roughly Gaussian. One could explain this based on that fuzzy Central Limit Theorem by imagining a number of independent sources of noise that would influence the length of an English sentence, including such things as the connotation of certain phrases (which influences the author's choice and thereby influences the length of the sentence) and the author's desire to be precise. In fact,

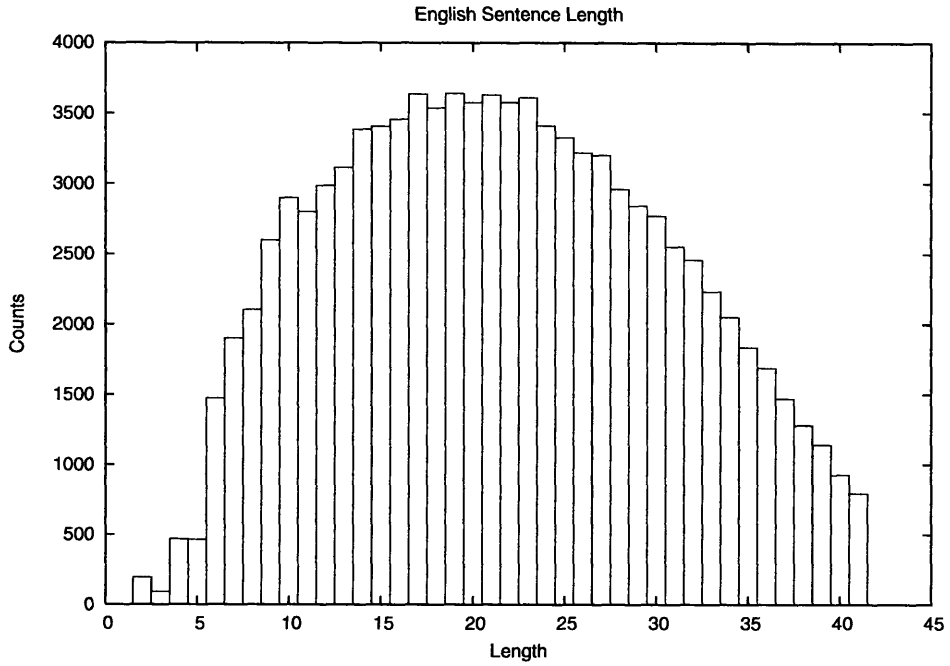
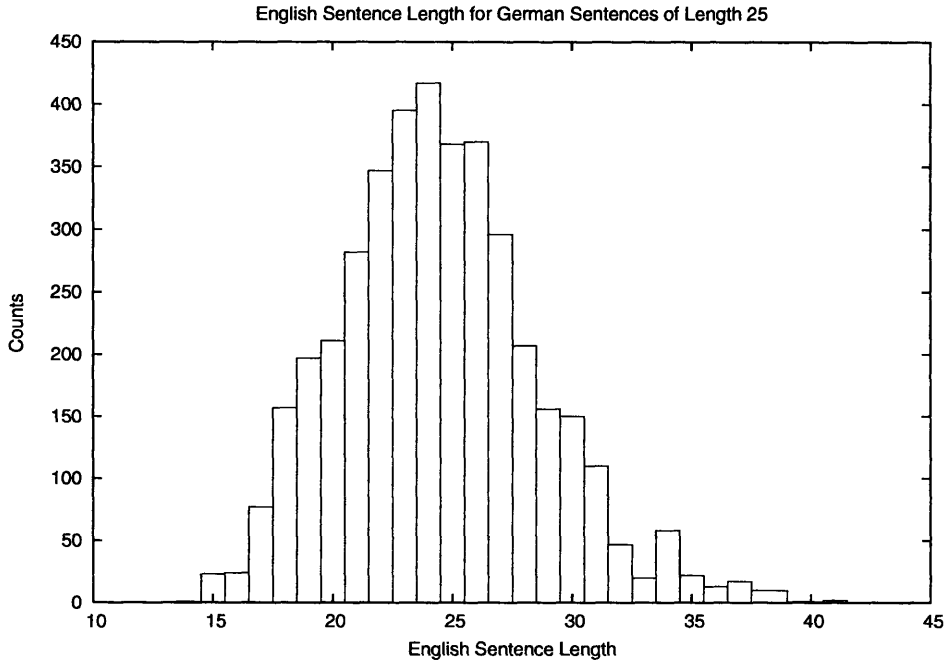


Figure 2-1: The length of English sentences drawn from the English translation of the proceedings of the European Parliament appears to have a Gaussian distribution.

the distribution of English sentence lengths for a fixed German sentence length also appears to be Gaussian (see Figure 2-2). Most relevant to us, however, is that when one trains an IBM2 model to translate from German to English, the distribution of the index of the word in a, say, 25-word English sentence that the 13th German word of a 25-word German sentence is aligned to also suggests the Gaussian shape, as Figure 2-3 shows. This can again be argued using the fuzzy Central Limit Theorem: all else equal, we imagine that a word is most likely to remain in the same relative spot within a sentence; for each transformation that would move it to one side, we imagine there is another transformation that is likely to move it to the other side. Ultimately, the reasons for a translator's choices are innumerable and will be written off as noise here.

One might argue that the best way to model a random variable of unknown distribution is by simply modelling a probability for each of its possible values. This technique is certainly flexible; unfortunately, it is a byword in the machine learning community that one must pay for added flexibility with more training data to avoid over-fitting. Given that even the most sophisticated machine translation models are

(a)



(b)

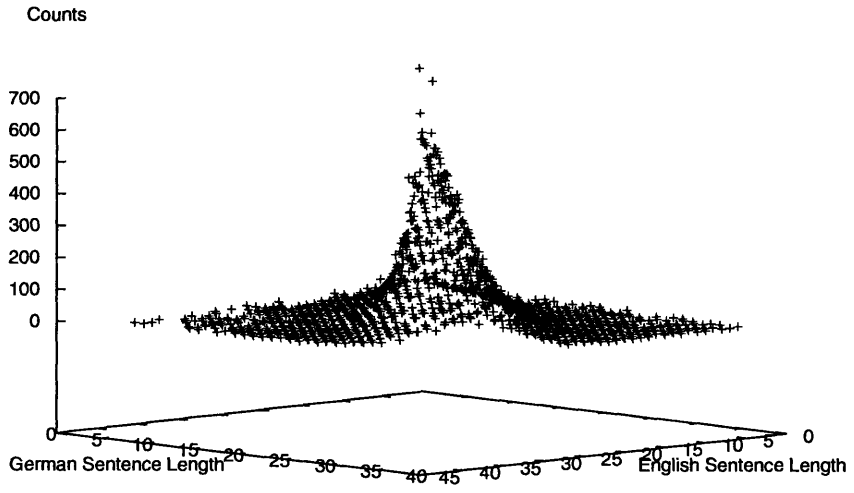


Figure 2-2: (a) The length of English sentences whose German translations are twenty-five words long appears to have a Gaussian distribution. (b) The length of sentences in English and German appears to have a bivariate Gaussian distribution. All things are indeed Gaussian.

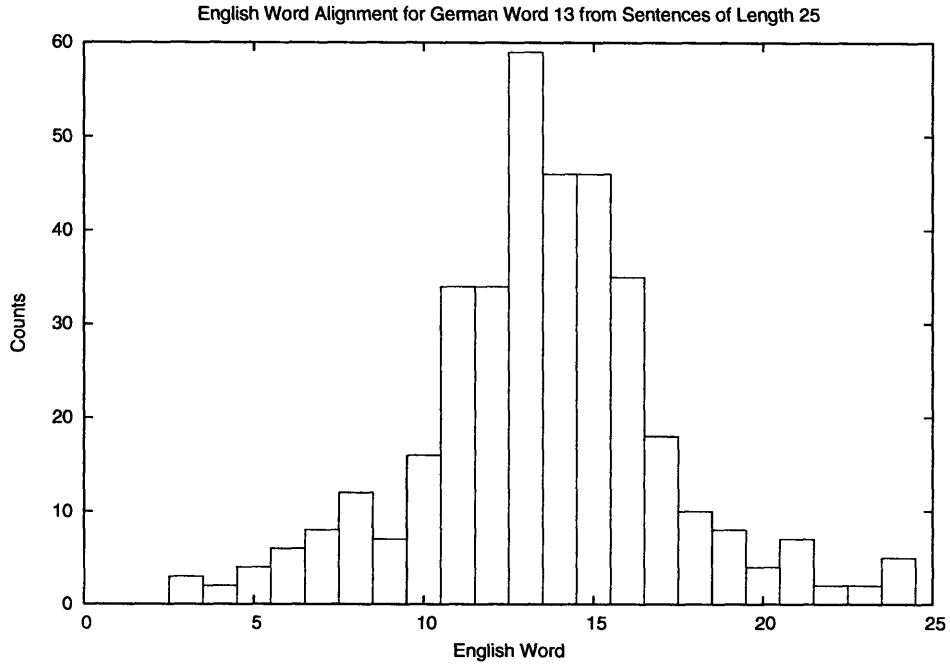


Figure 2-3: Our motivation: The indices of aligned words appear to be (roughly) Gaussian. At the very least, it seems like a safe approximation.

far from perfect and that they demand massive amounts of training data, our approach is to begin by further constraining existing models instead of creating more flexible ones.

In accordance to this intuition and our guess that alignments look Gaussian, instead of modelling every possible value for each alignment as a separate probability, we will model the alignment probabilities by a single Gaussian. The mathematical formulation for IBM Model 2 is changed by the addition of the second line:

$$\begin{aligned} P(\mathbf{f}, \vec{a} \,|\, \mathbf{e}) &= \prod_{j=1}^{m} T(\mathbf{f}_j \,|\, \mathbf{e}_{a_j}) D(a_j \,|\, j, \ell, m) \\ D(a_j \,|\, j, \ell, m) &= \frac{f_{\mathcal{N}}(a_j \,|\, \mu_{j,\ell,m}, \sigma_{j,\ell,m})}{\sum_{i=0}^{\ell} f_{\mathcal{N}}(i \,|\, \mu_{j,\ell,m}, \sigma_{j,\ell,m})}, \end{aligned}$$

where $f_{\mathcal{N}}(\cdot \,|\, \mu, \sigma)$ denotes the density of the gaussian with mean μ and standard deviation σ, $1/(\sqrt{2\pi}\sigma) \cdot \exp\left((\mu - \cdot)^2/(2\sigma^2)\right)$. That is to say, we replace $D(a_j \,|\, j, \ell, m)$ for $a_j = 1, \ldots, \ell$ with $\mu_{j,\ell,m}$ and $\sigma_{j,\ell,m}$ as our parameters. This typically results in fewer than 10% as many alignment parameters as IBM Model 2.

2.2 Algorithm

IBM Model 2 is a multinomial model; consequently, the EM updates are very easy to compute. The type of extension we are discussing ("parametrizing the parameters") corresponds to a simple addition to the algorithm in this case. The full derivation is discussed in the appendix.

Let's begin by casting the model in the standard form: each observation consists of a sentence pair $(\mathbf{e}, \mathbf{f})$ generated from a hidden alignment $\vec{a} \in \{0, \ldots, \ell\}^m$. The IBM Model 2 probability is:

$$\begin{aligned} P(\mathbf{f}, \vec{a} \mid \mathbf{e}; T, D) &= \prod_{j=1}^{m} T(\mathbf{f}_j \mid \mathbf{e}_{a_j}) D(a_j \mid j, \ell, m) \\ &= \prod_{f,e} T(f \mid e)^{\text{Count}_{f,e}(\mathbf{e},\mathbf{f},\vec{a})} \times \prod_{i,j,\ell,m} D(i \mid j, \ell, m)^{\text{Count}_{i,j,\ell,m}(\mathbf{e},\mathbf{f},\vec{a})} \end{aligned}$$

where

$$\begin{aligned} \text{Count}_{f,e}(\mathbf{e}, \mathbf{f}, \vec{a}) &:= \sum_{j=1}^{m} \delta(f, \mathbf{f}_j) \delta(e, \mathbf{e}_{a_j}) \\ \text{Count}_{i,j,\ell,m}(\mathbf{e}, \mathbf{f}, \vec{a}) &:= \delta(\ell, |\mathbf{e}|) \delta(m, |\mathbf{f}|) \delta(a_j, i), \end{aligned}$$

and $\delta(\cdot, \cdot)$ denotes the Kronecker delta function (1 if the two parameters are equal, and 0 otherwise). The probability model we propose is:

$$\begin{aligned} P(\mathbf{f}, \vec{a} \mid \mathbf{e}; T, \mu, \sigma) &= \prod_{f,e} T(f \mid e)^{\text{Count}_{f,e}(\mathbf{e},\mathbf{f},\vec{a})} \times \prod_{j,\ell,m} N(j, \ell, m)^{\text{Count}_{0,j,\ell,m}(\mathbf{e},\mathbf{f},\vec{a})} \\ &\quad \times \prod_{i,j,\ell,m} \left((1 - N(j, \ell, m)) \cdot f_{\mathcal{N}}(i \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m})\right)^{\text{Count}_{i,j,\ell,m}(\mathbf{e},\mathbf{f},\vec{a})}. \end{aligned}$$

Here, $N(j, \ell, m)$ is the probability that the jth French word aligns to the NULL English word (we do not wish this probability to be modeled by some slot in the Gaussian, because no position corresponds logically to the NULL word; thus we separate it in this fashion). Note that the $f_{\mathcal{N}}$ term is unnormalized—that is, the model is deficient under this framework for the sake of mathematical convenience.

Model 2 clearly falls under the multinomial framework; consequently, this new model falls under the parametrized multinomial framework described in the appendix.

Therefore, the addition we made to the mathematical formulation in the last section results in a single, easy change to the IBM Model 2 algorithm: after each EM iteration, the old D values are replaced by their maximum-likelihood Gaussian counterparts. That is to say, we transform the D's obtained from Model 2 as follows:

$$\begin{aligned}
\mu_{j,\ell,m} &= \frac{1}{m}\sum_{i=1}^{m} i \cdot D(i \mid j,\ell,m) \\
\sigma_{j,\ell,m} &= \sqrt{\frac{1}{m}\sum_{i=1}^{m} i^2 \cdot D(i \mid j,\ell,m) - \mu_{j,\ell,m}^2}
\end{aligned}$$

Adding the constraint that the alignment variables are samples of univariate Gaussians corresponds to an altogether simple change to the algorithm; the full algorithm is shown in Figure 2-4. (We let the 0th word, e_0, of every English sentence be the NULL word to simplify the notation.)

Our argument for this Gaussian approximation is only based on the idea that the alignment variables look like one-dimensional Gaussian densities and that Gaussians are easy to deal with; this is clearly not the only approximation that satisfies these properties. In fact, there are two other obvious choices based on the Gaussian: "truncated" Gaussians and "integrated" Gaussians. Here they are in math beneath our original formulation:

$$\begin{aligned}
\text{vanilla:} \quad D(a_j \mid j,\ell,m) &= f_{\mathcal{N}}(a_j \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m}), \\
\text{truncated:} \quad D(a_j \mid j,\ell,m) &= \frac{f_{\mathcal{N}}(a_j \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m})}{\sum_i f_{\mathcal{N}}(i \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m})}, \\
\text{and integrated:} \quad D(a_j \mid j,\ell,m) &= \frac{\int_{a_j-1}^{a_j} f_{\mathcal{N}}(i \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m})\, di}{\int_0^{\ell} f_{\mathcal{N}}(i \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m})\, di}.
\end{aligned}$$

In both the vanilla and truncated models, the alignment variable a_j can take on the values $1, \ldots, \ell$; the difference is that, for mathematical simplicity, the vanilla version is deficient, assigning probabilities to values outside this range (i.e.,that

Initialize $t(f \mid e)$ and $D(i \mid j, \ell, m)$

do:

- zero $t'(f \mid e)$ and $D'(i \mid j, \ell, m)$
- **for** $(\mathbf{e}, \mathbf{f})$ **in** corpus:
  - $m = |\mathbf{f}|$, $\ell = |\mathbf{e}|$
  - **for** j=1...m:
    - **for** i=0...ℓ:
      - $a_i = t(\mathbf{f}_j \mid \mathbf{e}_i) \cdot D(i \mid j, \ell, m)$
    - $a_i = a_i / (\sum_{i'} a_{i'})$
    - **for** i=0...ℓ:
      - $t'(\mathbf{f}_j \mid \mathbf{e}_i) = t'(\mathbf{f}_j \mid \mathbf{e}_i) + a_i$
      - $D'(i \mid j, \ell, m) = D'(i \mid j, \ell, m) + a_i$
- $t'(f \mid e) = t'(f \mid e) / (\sum_{f'} t'(f' \mid e))$
- $D'(i \mid j, \ell, m) = D'(i \mid j, \ell, m) / (\sum_{i'} D'(i' \mid j, \ell, m))$
- $t = t', D = D'$
- **for** ℓ, m:
  - **for** j=1...m:
    - $\mu = \frac{1}{\ell} \left(\sum_{i=1}^{\ell} D(i \mid j, \ell, m) \cdot i \right) \Big/ (1 - D(0 \mid j, \ell, m))$
    - $\sigma = \frac{1}{\ell} \left(\sum_{i=1}^{\ell} D(i \mid j, \ell, m) \cdot (i - \mu)^2 \right) \Big/ (1 - D(0 \mid j, \ell, m))$
    - **for** i=1...ℓ:
      - $D(i \mid j, \ell, m) = \exp(-(i - \mu)^2 / 2\sigma)$
    - $D(i \mid j, \ell, m) = D(i \mid j, \ell, m) \cdot (1 - D(0 \mid j, \ell, m)) / (\sum_{i'=1}^{\ell} D(i' \mid j, \ell, m))$

until convergence

Figure 2-4: The Vanilla Gaussian Algorithm.

$\sum_{a_j} D(a_j \mid j, \ell, m) \lneq 1$). Although we obviously cannot align words outside of the sentence, we allow the algorithm to assign non-zero probability mass to those alignments; we recover by normalizing over legal alignments afterward. We can instead make this restriction in the algorithm itself. Unfortunately, the M-step of the EM algorithm is no longer a beautiful closed form, but instead requires numerical optimization. Although this is still tractable, it is undesirable and the results are not sufficiently improved to warrant this computational burden (our experiments showed virtually identical results to the vanilla model).

Likewise, the integrated version is very attractive intuitively, but the shape of the density is so close to that of the vanilla edition that any improvements are minute and are outweighed by the added computational complexity, as the Q-function in EM must once again be optimized by a numerical optimization technique. For completeness, we explain how one would perform such optimization; in this case, just as in the vanilla algorithm, the change only amounts to "fitting" (in the maximum-likelihood sense) the appropriate density to the intermediate D-values after each EM step. We can compute the gradient of the likelihood function in both of these cases, so we optimize it using a gradient optimization technique. In our experience, the likelihood functions tend to have long, narrow valleys, so we find that optimizing using conjugate gradient descent is faster than just using steepest descent. (Note again that we do not include the NULL parameter $D(0 \mid \cdot, \cdot, \cdot)$ in the fit.)

We have to fit a curve to each set of D-values, so fix j, ℓ, and m. Define $\overline{D} = D/(1 - D_0)$ to reflect the fact that we are not including the NULL word position in our model and to thus further ease the notational burden. We wish to maximize the log-likelihood, so let's begin by writing it down:

$$\begin{aligned}
\mathcal{L}_{\text{truncated}} &= \sum_{i=1}^{\ell} \overline{D}(i \mid j, \ell, m) \log \frac{f_{\mathcal{N}}(a_j \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m})}{\sum_i f_{\mathcal{N}}(i \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m})} \\
&= \sum_{i=1}^{\ell} \overline{D}(i \mid j, \ell, m) \cdot \frac{-(i-\mu)^2}{2\sigma^2} - \log \left(\sum_{i=1}^{\ell} f_{\mathcal{N}}(i \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m}) \right) \\
\mathcal{L}_{\text{integrated}} &= \sum_{i=1}^{\ell} \overline{D}(i \mid j, \ell, m) \log \frac{\int_{a_j-1}^{a_j} f_{\mathcal{N}}(i \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m})\, di}{\int_0^{\ell} f_{\mathcal{N}}(i \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m})\, di}
\end{aligned}$$

$$= \sum_{i=1}^{\ell} \overline{D}(i \mid j, \ell, m) \cdot \log(F_{\mathcal{N}}(i \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m}) - F_{\mathcal{N}}(i-1 \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m}))$$
$$- \log(F_{\mathcal{N}}(\ell \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m}) - F_{\mathcal{N}}(0 \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m})),$$

where $F_{\mathcal{N}}$ denotes the cumulative distribution function. We evaluate the gradient:

$$\frac{\partial \mathcal{L}_{\text{truncated}}}{\partial \mu} = \sum_{i=1}^{\ell} \overline{D}(i \mid j, \ell, m) \cdot \frac{i-\mu}{\sigma^2} - \frac{\sum_{i=1}^{\ell} f_{\mathcal{N}}(i \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m}) \cdot \frac{i-\mu}{\sigma^2}}{\sum_{i=1}^{\ell} f_{\mathcal{N}}(i \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m})}$$

$$\frac{\partial \mathcal{L}_{\text{truncated}}}{\partial \sigma^2} = \sum_{i=1}^{\ell} \overline{D}(i \mid j, \ell, m) \cdot \frac{-(i-\mu)^2}{2\sigma^4} - \frac{\sum_{i=1}^{\ell} f_{\mathcal{N}}(i \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m}) \cdot \frac{-(i-\mu)^2}{2\sigma^4}}{\sum_{i=1}^{\ell} f_{\mathcal{N}}(i \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m})}$$

$$\frac{\partial \mathcal{L}_{\text{integrated}}}{\partial \mu} = \sum_{i=1}^{\ell} \overline{D}(i \mid j, \ell, m) \cdot \sqrt{2\sigma^2} \cdot \frac{f_{\mathcal{N}}(i-1 \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m}) - f_{\mathcal{N}}(i \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m})}{F_{\mathcal{N}}(i \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m}) - F_{\mathcal{N}}(i-1 \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m})}$$
$$-\sqrt{2\sigma^2} \cdot \frac{f_{\mathcal{N}}(0 \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m}) - f_{\mathcal{N}}(\ell \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m})}{F_{\mathcal{N}}(\ell \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m}) - F_{\mathcal{N}}(0 \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m})}$$

$$\frac{\partial \mathcal{L}_{\text{integrated}}}{\partial \sigma^2} = \sum_{i=1}^{\ell} \overline{D}(i \mid j, \ell, m) \cdot \frac{1}{4\sigma^2} \cdot \frac{f_{\mathcal{N}}(i-1 \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m}) - f_{\mathcal{N}}(i \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m})}{F_{\mathcal{N}}(i \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m}) - F_{\mathcal{N}}(i-1 \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m})}$$
$$-\frac{1}{4\sigma^2} \cdot \frac{f_{\mathcal{N}}(0 \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m}) - f_{\mathcal{N}}(\ell \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m})}{F_{\mathcal{N}}(\ell \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m}) - F_{\mathcal{N}}(0 \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m})}$$

We apply the conjugate gradient descent algorithm to the D values [10]:

Given $D \in \mathbb{R}^m$
- Select $\vec{x}_0 = (\mu, \sigma^2) \in \mathbb{R}^2$ at random
- $i = 0$, $\vec{g}_0 = \nabla \mathcal{L}(\vec{x}_0)$, $\vec{h}_0 = -\vec{g}_0$
- **do**:
  - $\lambda_i = \arg\min_{\lambda \geq 0} \mathcal{L}(\vec{x}_i + \lambda_i \vec{h}_i)$
  - $\vec{x}_{i+1} = \vec{x}_i + \lambda_i \vec{h}_i$
  - $\vec{g}_{i+1} = \nabla \mathcal{L}(\vec{x}_{i+1})$
  - $\gamma_i = (\vec{g}_{i+1} - \vec{g}_i) \cdot \vec{g}_{i+1} / \|\vec{g}_i\|^2$
  - $\vec{h}_{i+1} = -\vec{g}_{i+1} + \gamma_i \vec{h}_i$
  - $i = i + 1$
- **until** convergence

return $\vec{x}_i$

2.3 Implementation and Evaluation

The dataset used to evaluate our system was the standard EUROPARL set of Köhn *et al.* We used German-English as a representative language pair, as translation is neither especially easy nor especially difficult [5]. The data was aligned at the sentence-level using the standard tools and sentences of vastly differing lengths were removed. Finally, we trained the system on the data and produced Viterbi alignments for each sentence pair. These alignments were output to the Köhn phrase-based system, which itself produced a phrase dictionary. This dictionary was applied to the Pharoah decoder (with a language model trained on the entire available training set). Finally, we applied the system to the standard test set chosen by Köhn in [5]. The resulting translations were compared to the human-translated reference using the BLEU metric of [9].

The BLEU metric is a standard method for evaluating machine translation system performance by comparing translations to one or many human translations. The translations are compared by precision and recall on n-grams of successively greater length; the BLEU score typically refers to a smoothed 4-gram comparison; mathematically, it can be described by the following formula:

$$\text{BLEU} = e^{I_{c \leq r} \cdot (1 - r/c)} \cdot \sqrt[4]{p_1 p_2 p_3 p_4},$$

where r is the length of the reference corpus, c is the total length of the candidate translation produced by the system being evaluated, r is sum of the lengths of the reference sentences that most closely match the lengths of the candidate sentences, $I_{c \leq r}$ is 1 if $c \leq r$ and 0 otherwise, and p_j refers to the j-gram precision of the test set.

We evaluated our technique using the EUROPARL corpus [5] and the applied a BLEU scorer to our model's output on the standard section 24 test set. Our results (when training is done on the full data set) are shown in Table 1. Our results are clearly very competitive with IBM Model 2.

Considering the small number of parameters in the one-dimensional gaussian model, intuition suggests that it should converge to its limiting BLEU score with

less data. If this is true, it is an insignificant effect, as Figure 2-5 shows; we believe that this is due to the still overwhelming number of translation parameters in the model. The performance of the one-dimensonal gaussian model is, in fact, indistinguishable from that of Model 2.

An added advantage of our technique is that it provides a metric for the difficulty of translation from one language to the other: the variance of the model. Intuitively, increasing variance implies a larger search-space for the placement of each word and it is, consequently, more likely that our search will not find the optimal translation. We can normalize this over sentence lengths and weight it by frequency using the following formula:

$$\frac{\sum_{m,\ell,j} \frac{\sigma^2_{j,\ell,m}}{\ell^2} \cdot \frac{n(m)}{m}}{\sum_m \frac{n(m)}{m}},$$

where $n(m)$ denotes the number of French sentences of length m in the corpus. We call this final score the Normalized Language Cross-Variance and it is compared to the phrase-based model BLEU scores of [5] for twenty language pairs in Table 2.1. We feel that this new metric will be useful for researchers working along the lines suggested in [2].

The expected use is in a scenario where one whishes to improve translation accuracy using clause restructuring; to measure roughly whether a transformation will increase the translation quality, one may simply compare the cross-variance with and without the change. This approach could also be used to suggest transformations; one could measure the variance of members of a particular type of clause or part-of-speech to pinpoint sources of variance.

As expected, data reordered using the rules suggested in [2] exhibited a reduction in NLCV for both English to German (from 0.02152 to 0.02066) and from German to English (from 0.01839 to 0.01721). This reduction in variance agrees with the 6% BLEU score improvement they report, along with the high ratings achieved in their subjective evaluations.

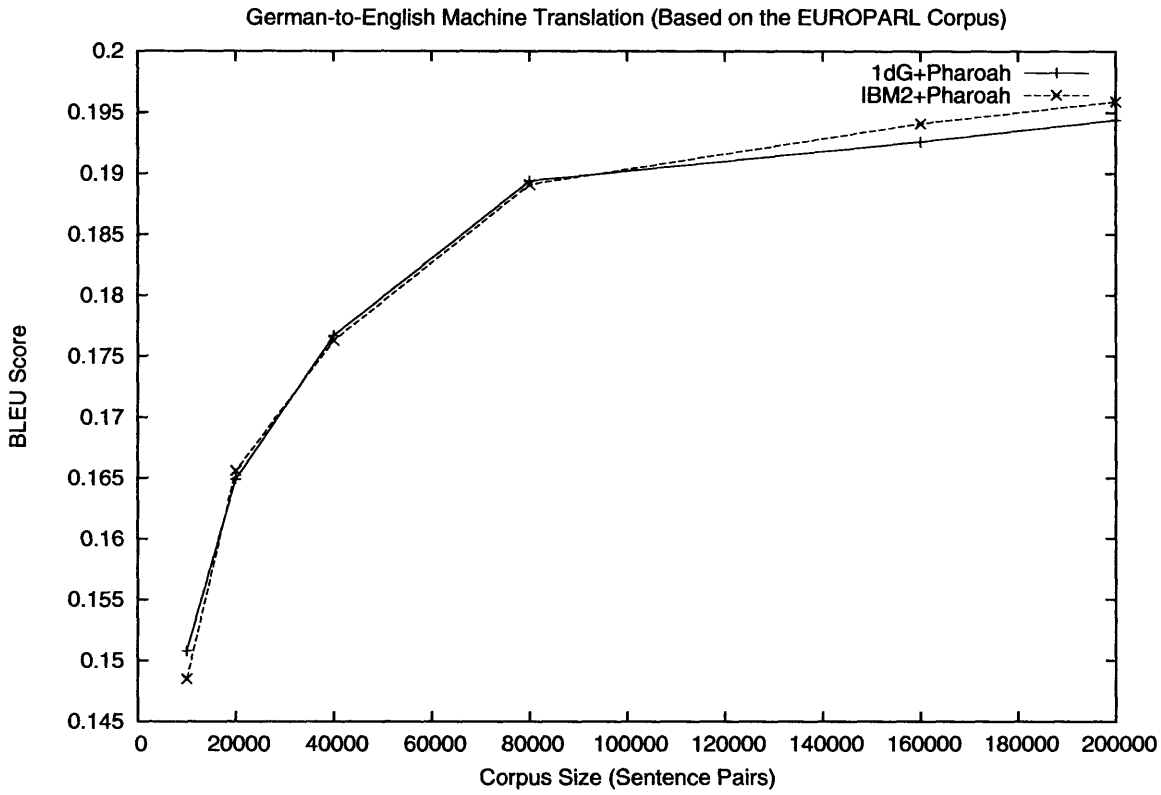
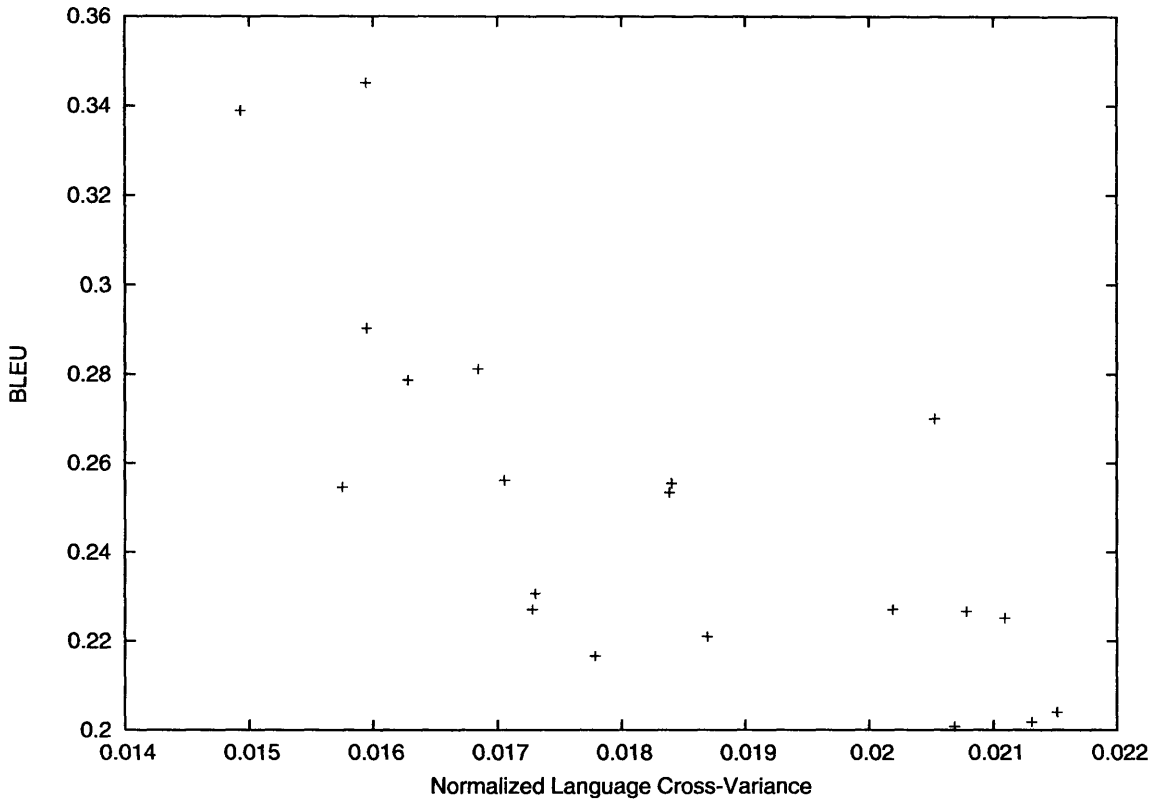


Figure 2-5: The performance of the alignments induced by the one-dimensional gaussian model is similar to those induced by the IBM-2 model. This graph shows the BLEU metric of the two models when applied to the EUROPARL training data and standard test set for the German-English language pair.



to / from	Danish	German	English	Spanish	French
Danish	-	1575 / 2546	1706 / 2561	1779 / 2166	1730 / 2307
German	1728 / 2271	-	2152 / 2040	2068 / 2008	2131 / 2018
English	1595 / 2903	1839 / 2534	-	1684 / 2812	1628 / 2787
Spanish	2019 / 2271	2109 / 2252	2053 / 2700	-	1594 / 3452
French	1869 / 2210	2078 / 2267	1841 / 2555	1493 / 3389	-

Table 2.1: BLEU score versus Normalized Language Cross-Variance (NLCV). We expect that translation quality will decrease as cross-variance increases; furthermore, we expect that BLEU scores increase with increasing translation quality. Thus, we expect that BLEU scores decrease with increasing cross-variance. This graph suggests this expected trend. The BLEU scores here were taken from [5]; our metric was measured on 10,000 sentence pairs in each case taken from the EUROPARL corpus, the same corpus used to train and test the phrase-based model in [5]. The values in the table are NLCV$\cdot 10^5$/BLEU$\cdot 10^4$.

Chapter 3

Multivariate Gaussian Alignments

3.1 Motivation

Now that we have exchanged a large fraction of alignment parameters for an almost neglibile reduction in accuracy, we are free to select more meaningful alignment parameters that will hopefully significantly increase accuracy.

We choose to make a natural extension to the one-dimensional gaussian model by way of addressing one of the disturbing assumptions of the early IBM Models: we will introduce alignment covariation into the model, allowing words to be influenced by the movement of both neighboring and distant words. Our technique for modelling covariance is to replace the m one-dimensional Gaussians of the last model by a single m-dimensional Gaussian for each encountered sentence-length pair (ℓ, m). We thereby replace the $2m$ parameters of the last model by $m(m+1)$ parameters in the new model, comparable to the ℓm parameters we started out with in Model 2.

We wind up with the following, somewhat familiar, expression for the model's joint probability:

$$P(\mathbf{f}, \vec{a} \,|\, \mathbf{e}; T, D) = D(\vec{a}, |\mathbf{e}|, |\mathbf{f}|) \prod_{j=1}^{m} T(\mathbf{f}_j \,|\, \mathbf{e}_{a_j})$$

where

$$D(\vec{a}, \ell, m) = \frac{f_{\mathcal{N}}(\vec{a} \mid \vec{\mu}_{\ell,m}, \Sigma_{\ell,m})}{\sum_{\vec{a}' \in \{1,\ldots,m\}^{\ell}} f_{\mathcal{N}}(\vec{a}' \mid \vec{\mu}_{\ell,m}, \Sigma_{\ell,m})},$$

and

$$f_{\mathcal{N}}(\vec{x} \mid \vec{\mu}, \Sigma) = \frac{1}{\sqrt{|2\pi\Sigma|}} \exp\left(-\frac{1}{2}(\vec{x} - \vec{\mu})^T \Sigma^{-1} (\vec{x} - \vec{\mu})\right).$$

3.2 Algorithm

Fortunately, despite this update to the probability model, it still falls within the parametrized multinomial framework. Unfortunately for us, the updates are no longer tractable. Since we are now free to violate the independence assumptions that enabled us to factor the probabilities, we have to evaluate the old model probability for each alignment to compute the exact updates; this corresponds to ℓ^m computations for each sentence pair consisting of a source sentence of m words and target sentence of ℓ words. Considering that sentences of forty or more words occur in reasonable-sized corpora, we are faced with 40^{40} computations—definitely outside the realm of reason.

On the other hand, intuition suggests that the vast majority of possible alignments are very unlikely. Thus, we expect that we are computing sums of a great many infinitesimal numbers and a relatively small number of relatively large numbers. It follows that we should be able to make good approximations by ignoring the most unlikely alignments. Given that it is easy to sample from multivariate Gaussians, we can use Monte Carlo techniques to make these approximations. There is a vast literature on the use of Monte Carlo techniques to approximate the E-step of the EM algorithm beginning with [12], which contains the first formal description of the technique, continuing to [11], which justifies this approximation mathematically and establishes convergence conditions for the modified algorithm. We will elide these details here, but will digress briefly into a discussion of sampling techniques to motivate the final algorithm.

3.2.1 Sampling

Consider a general situation when one wishes to evaluate an expectation $E_X[g(X)]$ where X is random variable with density $f_X(x)$. This corresponds to the integral:

$$E_X[g(X)] = \int_X g(x) f_X(x)\, dx.$$

Now suppose that this integral is impossible to compute because we cannot simplify the integral and the state space is too large for us to visit every point; we are forced to approximate the integral.

Uniform Sampling

An immediate idea is to draw samples of X from the distribution f_X, evaluate g at these points, and to average the results. That is, we suggest an algorithm like this:

---

sum = 0 ; count = 0

for a bunch of times:

- draw x with probability $f_X(x)$
- sum += $g(x)$
- count += 1

return sum/count

This will certainly return the correct answer in expectation and in the limit of infinitely many samples; furthermore, the variance will decrease as the number of samples increases. Unfortunately, the variance may still be forbiddingly large for realistic numbers of samples. In some cases, this may not even be an issue, as we may not even be able to easily draw samples from f_X.

Importance Sampling

Since drawing samples from f_X itself is difficult, let us consider using uniform sampling from a different density, ϕ, making use of the following "transformation":

$$E_{X\sim f_X}[g(X)] = E_{X\sim\phi}[g(X)\cdot(f_X(X)/\phi(X))],$$

which is valid as long as ϕ is nonzero everywhere f_X is nonzero. Although this transform does not change the value of the expectation, it can change the variance (though this can be difficult to calculate empirically, particularly if sampling from f_X was intractable). Conventional wisdom holds that desirable ϕ peak where f_X peaks and has "heavy tails." My own opinion, of which I expect I will be disabused shortly, is that one should have a distribution that is peaked at the right places but that one should not be overly concerned about correctly modelling the tails.

Gibbs Sampling

Unfortunately, it is often difficult to find a distribution that satisfies even these properties. For instance, sampling from random variables with several components is usually intractable due to the curse of dimensionality. It is often true that sampling from the conditional densities (i.e., from the distribution of one component conditioned on the other components) is easy, however, whence we can appeal to more advanced mathematics to achieve a decent approximation: Markov chains.

We know that for Markov chains that are sufficiently nice, the influence of some state in its history on the the current state diminshes as time passes. In more concrete terms, a Markov chain that is regular[1] has a stationary distribution that is independent of the starting state.

Our strategy for sampling from f_X is to construct a Markov chain whose stationary distribution is f_X. Starting from an arbitrary starting point, after sampling from the Markov chain very many times, we will be drawing samples from the stationary

[1]It is frustrating how often the term "regular" is used. A regular Markov chain is one for which, for any initial state, after some number of steps independent of the initial state, the probability of being in any state is nonzero.

distribution, regardless of how our initial state was chosen.

Gibbs sampling is a technique for creating such a Markov chain. In this case, we vary one variable at a time, fixing the others, by sampling from the conditional distribution. After a suitable "burn-in" period to allow the distribution to approach the stationary distribution, we can extract a single sample to use for Monte Carlo integration, skipping over several consecutive samples as they are obviously not independent. Instead of starting over and incurring another full burn-in, however, we can have a mini-burn-in to recover independence. That is, after extracting a sample, we take many steps in the Markov chain until we are visiting points that arguably have little to do with the sample we chose, then extract another sample. Alternatively, we can mix-and-match, drawing several samples from a single chain and then completely resetting it, performing the burn-in and collecting a set of samples to improve the independence guarantees.

Here is the Gibbs sampling algorithm in pseudo-code (N is the number of samples desired, M is the number of samples drawn from a single chain before doing a full burn in, and d is the number of components of the state vector).

---

for sample $k = 0 \ldots N$:
- **if** $k \,\%\, M == 0$:
  - $\vec{S}$ = random state
  - burn-in size = full
- **else**:
  - burn-in size = mini
- **for** size burn-in:
  - **for** $j = 1 \ldots d$:
    - draw a sample $\vec{S}_j$ given $\vec{S}_i$ for $i \neq j$
- append $\vec{S}$ to the list of samples

return list of samples

3.2.2 Sampling from Weighted Multivariate Gaussians

It is time to face the expressions that we must approximate:

$$\begin{aligned}
\vec{\mu}_{\ell,m} &= \sum_{\substack{s \\ |\mathbf{e}^{(s)}|=\ell \\ |\mathbf{f}^{(s)}|=m}} \sum_{\vec{a}} \left[\prod_{j=1}^{m} T(\mathbf{f}_j^{(s)} \mid \mathbf{e}_{a_j}^{(s)}) \right] D(\vec{a} \mid \ell, m)\ \vec{a} \\
\Sigma_{\ell,m} + \vec{\mu}_{\ell,m}\vec{\mu}_{\ell,m}^T &= \sum_{\substack{s \\ |\mathbf{e}^{(s)}|=\ell \\ |\mathbf{f}^{(s)}|=m}} \sum_{\vec{a}} \left[\prod_{j=1}^{m} T(\mathbf{f}_j^{(s)} \mid \mathbf{e}_{a_j}^{(s)}) \right] D(\vec{a} \mid \ell, m)\ \vec{a}\vec{a}^T \\
T(f \mid e) &= \sum_{\substack{s \\ e \in \mathbf{e}^{(s)} \\ f \in \mathbf{f}^{(s)}}} \sum_{\vec{a}} \left[\prod_{j=1}^{m} T(\mathbf{f}_j^{(s)} \mid \mathbf{e}_{a_j}^{(s)}) \right] D(\vec{a} \mid \ell, m) \left(\sum_{j=1}^{m} \delta(\mathbf{f}_j^{(s)}, f)\delta(\mathbf{e}_j^{(s)}, e) \right)
\end{aligned}$$

It is the inner sum, over $\vec{a}$, that we cannot compute and therefore must approximate. We have several choices: one obvious idea is to draw samples of $\vec{a}$ from D, which is easy since D is a multivariate Gaussian (given several uniform variates drawn from the interval $(0, 1]$, use the Box-Muller transform to obtain m standard normal variates that make up the coordinates of a sample $\vec{n}$ of an m-dimensional Gaussian with mean 0 and covariance matrix $I_{m \times m}$, which we can transform using $\sqrt{\Sigma}\vec{n} + \vec{\mu}$). Another obvious choice is Gibbs sampling since almost any expression in which only one of the alignments varies is easy to compute. Let us write pseudo-code for both algorithms, then give the full pseudo-code for the multivariate technique. First, importance sampling:

Given a sentence pair $\mathbf{e}, \mathbf{f}$

list of samples = empty list

for however many samples one wishes to generate:

- draw a sample $\vec{a}$ from the multivariate gaussian distribution D
- append $(\vec{a}, \prod_{j=1}^{m} T(\mathbf{f}_j, \mathbf{e}_{a_j}))$ to the list of samples

return list of samples

Next, we give the algorithm for Gibbs sampling with twenty samples generated from each chain (here % denotes the modulus operator and we assume that the language

is sensibly zero-indexed).

Given a sentence pair $\mathbf{e}, \mathbf{f}$
for sample k of however many samples one wishes to generate:
· **if** $k \,\%\, 20 == 0$:
· · $\vec{a}$ = random alignment
· · burn-in size = full
· **else**:
· · burn-in size = mini
· **for** size burn-in:
· · **for** $j = 1 \ldots m$:
· · · draw a sample $\vec{a}_j$ given $\vec{a}_i$ for $i \neq j$
· append $\vec{a}$ to the list of samples
return list of samples

Again, our idea is to draw a number of alignments from the old parameters in each EM step and use these alignments to estimate the updated parameters. The conditional distribution of $\vec{a}_j$ given the remaining coordinates of $\vec{a}$ is given by:

$$\Pr\left(\vec{a}_j = i \,|\, \vec{a}_{j'} = \vec{a}'_{j'} \text{ for } j' \neq j\right) \;=\; \frac{\Pr\left(\vec{a} = \vec{a}'_{j=i}\right)}{\sum_{i'=1}^{\ell} \Pr\left(\vec{a} = \vec{a}'_{j=i'}\right)},$$

where we use $\vec{a}'_{j=i}$ to denote the vector $\vec{a}'$ where the jth coordinate $\vec{a}'_j$ is replaced by i. So, to sample from the distribution of $\vec{a}_j$ given $\vec{a}_i$ for $i \neq j$, simply compute the full probability $P(\vec{a})$ for each of the ℓ possible values of $\vec{a}_j$ and normalize the results. Let us write this expression in terms of D and T:

$$\begin{aligned}
\Pr\left(\vec{a} = \vec{a}'_{j=i}\right) &= D(\vec{a}'_{j=i}, \ell, m)\; T(\mathbf{f}_j \,|\, \mathbf{e}_{a_i}) \prod_{j' \neq j} T(\mathbf{f}_{j'} \,|\, \mathbf{e}_{\vec{a}'_{j'}}) \\
&= \frac{f_{\mathcal{N}}(\vec{a}'_{j=i} \,|\, \vec{\mu}_{\ell,m}, \Sigma_{\ell,m})}{\sum_{\vec{a}'' \in \{1,\ldots,m\}^{\ell}} f_{\mathcal{N}}(\vec{a}'' \,|\, \vec{\mu}_{\ell,m}, \Sigma_{\ell,m})} T(\mathbf{f}_j \,|\, \mathbf{e}_{a_i}) \prod_{j' \neq j} T(\mathbf{f}_{j'} \,|\, \mathbf{e}_{\vec{a}'_{j'}})
\end{aligned}$$

$$\Rightarrow \qquad \Pr\left(\vec{a}_j = i \,|\, \vec{a}_{j'} = \vec{a}'_{j'} \text{ for } j' \neq j\right) \;=\; \frac{f_{\mathcal{N}}(\vec{a}'_{j=i} \,|\, \vec{\mu}_{\ell,m}, \Sigma_{\ell,m}) T(\mathbf{f}_j \,|\, \mathbf{e}_i)}{\sum_{i'=1}^{\ell} f_{\mathcal{N}}(\vec{a}'_{j=i'} \,|\, \vec{\mu}_{\ell,m}, \Sigma_{\ell,m}) T(\mathbf{f}_j \,|\, \mathbf{e}_{i'})}.$$

Now, since computing $f_{\mathcal{N}}$ involves the matrix multiplication $(\vec{a}-\vec{\mu})^T\Sigma^{-1}(\vec{a}-\vec{\mu})$, it takes $O(m^2)$ time. However, if we keep track of the former value of $\vec{a}_j$, we can avoid repeating much of the work:

$$\begin{aligned}(\vec{a}-\vec{\mu})^T\Sigma^{-1}(\vec{a}-\vec{\mu}) &= \sum_{r=1}^{m}\sum_{s=1}^{m}(\vec{a}_r-\vec{\mu}_r)\left[\Sigma^{-1}\right]_{rs}(\vec{a}_s-\vec{\mu}_s) \\ &= \sum_{r\neq j}\sum_{s\neq j}(\vec{a}_r-\vec{\mu}_r)\left[\Sigma^{-1}\right]_{rs}(\vec{a}_s-\vec{\mu}_s) \\ &\quad +2\sum_{r\neq j}(\vec{a}_r-\vec{\mu}_r)\left[\Sigma^{-1}\right]_{js}(\vec{a}_j-\vec{\mu}_j)+\left[\Sigma^{-1}\right]_{jj}(\vec{a}_j-\vec{\mu}_j)^2.\end{aligned}$$

Thus, to calculate the change in $(\vec{a}-\vec{\mu})^T\Sigma^{-1}(\vec{a}-\vec{\mu})$ brought about by changing $\vec{a}_j$, we simply have to calculate the difference in the second two terms, which takes $O(m)$ time.

The algorithm describing how we would make use of the sampling is below.

---

for **e**, **f** in corpus:
- **for** each iteration:
  - draw $\vec{a}$ from $P(\vec{a}\,|\,\mathbf{f},\mathbf{e},T',D')$ with probability-multiplier p
  - $p' = p\cdot\prod_{j=1}^{m}T(\mathbf{f}_j,\mathbf{e}_{a_j})$
  - $n_{\ell,m,\text{pre}}$ += p'
  - $\vec{\mu}_{\ell,m,\text{pre}}$ += $\vec{a}\cdot p'$
  - $\Sigma_{\ell,m,\text{pre}}$ += $\vec{a}\vec{a}^T\cdot p'$
  - **for** j in $1\ldots m$:
    - $T(\mathbf{f}_j\,|\,\mathbf{e}_{a_j})$ += p'

for sentence pair sizes ℓ, m:
- $\vec{\mu}_{\ell,m} = \vec{\mu}_{\ell,m,\text{pre}}/n_{\ell,m,\text{pre}}$
- $\Sigma_{\ell,m} = \Sigma_{\ell,m,\text{pre}}/n_{\ell,m,\text{pre}} - \vec{\mu}\vec{\mu}^T$

for e in English Vocabulary:
- sum = 0
- **for** f in French Vocabulary:
  - sum += $T(f\,|\,e)$
- **for** f in French Vocabulary:
  - $T(f\,|\,e)$ /= sum

1dG	24
Equal	50
MdG	27

Table 3.1: Results of the double-blind trial on alignments generated by 101 sentence pairs using 1dG (the one-dimensional Gaussian technique of the last chapter) and MdG (the multi-variate Gaussian technique of this chapter) models trained on 100,000 sentence pairs of the EUROPARL corpus.

3.3 Implementation and Evaluation

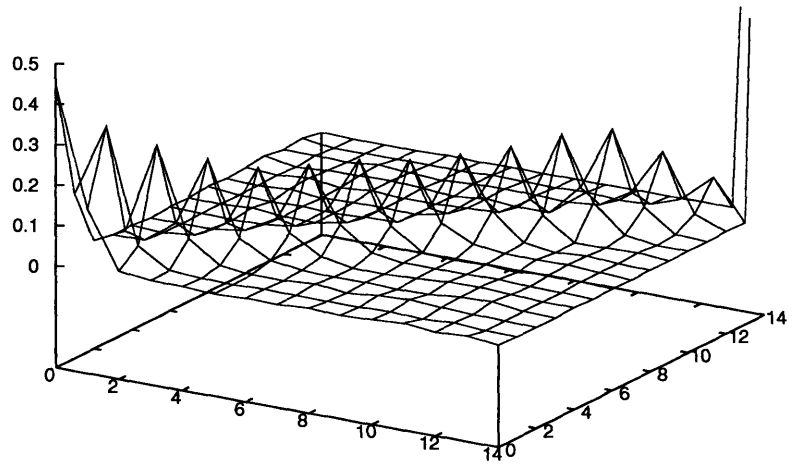
To evaluate the value of the expensive, full covariance matrix, we trained the one-dimensional Gaussian model described in the last chapter on 100,000 sentences from the EUROPARL corpus and converted the model parameters to the multi-variate Gaussian by combining the m one-dimensional Gaussians for each sentence length pair m, ℓ to a single m-dimensional Gaussian in the natural fashion: $\vec{\mu}_i = \mu_{i,m,\ell}$, $\Sigma_{ii} = \sigma_{i,m,\ell}$, $\Sigma_{ij} = 0$ for $i \neq j$. Then, we perform EM iterations on $\vec{\mu}$ and Σ until convergence. For evaluation purposes, we iterate only on sentence pairs with $m = \ell = 15$ words. The covariance matrix is depicted in Figure 3-1. The matrix is effectively tridiagonal[2].

Therefore, we repeat the experiment, constraining the matrix to be tridiagonal (simply by setting the other matrix elements to zero after each EM iteration) and outputting maximum-likelihood alignments for each word. The alignments were compared to alignments generated by the one-dimensional technique and evaluated using double-blind trials in a subjective evaluation for 101 sentence pairs. The results are in Table 3.1.

Quantitatively, the improvement is insignificant. Out of 3434 sentences, 1105 had identical alignments (excluding words aligned to the NULL word, since the multi-variate model has no mechanism for aligning words to NULL) and out of the 51510 German words, only 7169 (about 14%) were aligned to different English words by the two different models. The alignments are considered equally correct by human evaluators half the time. The rest of the time, the new model makes the alignments

[2]A tridiagonal matrix A is one for which $A_{ij} = 0$ for $|i - j| > 1$.

(a)



(b)

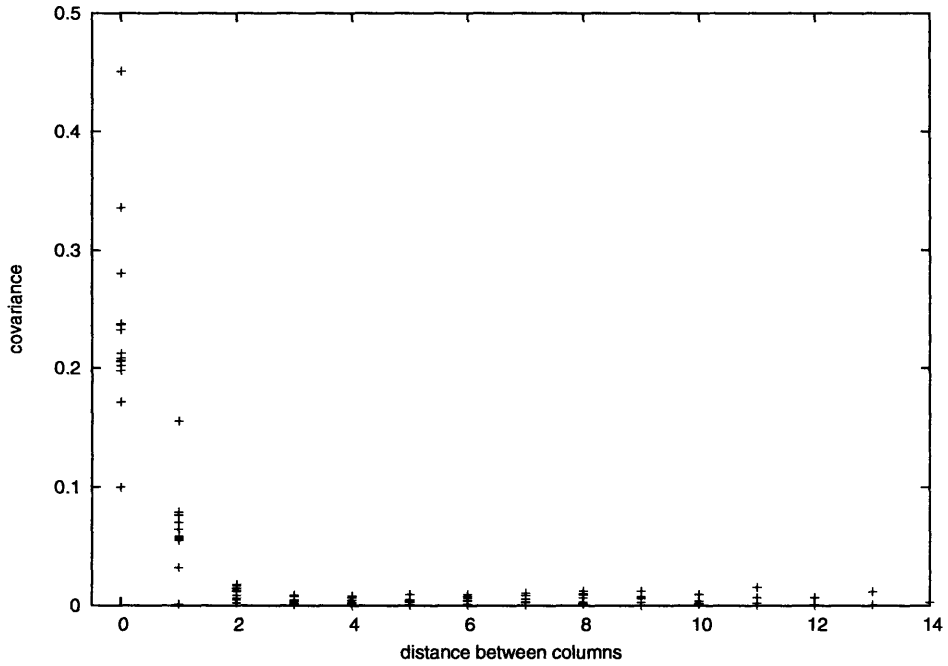


Figure 3-1: The covariance matrix. Graph (a) shows the actual values of the cells. Graph (b) shows the values versus the distance between the cell indices. Clearly values off the tridiagonal are nearly zero.

worse almost as often as it makes them better.

Even so, several concrete improvements and problem areas were identified in post-analysis: The multivariate model aligns pronouns and multiple articles throughout a sentence more accurately than the plain one-dimensional Gaussian model; we suggest that this is simply due to the covariance factor, associating the movement of proximate words. However, the multivariate model tends to align common verbs in dependent clauses and infinitives with their associated pronouns instead of with the corresponding verb in English. Furthemore, the multivariate model (with less frequency, but worthy of note) assigns German articles to the corresponding noun in English. This is easy to understand; German simply tends to have more determiners than English and the model became "used" to explaining determiners with the nouns they modify. These improvements and issues are illustrated in Figure 3-2.

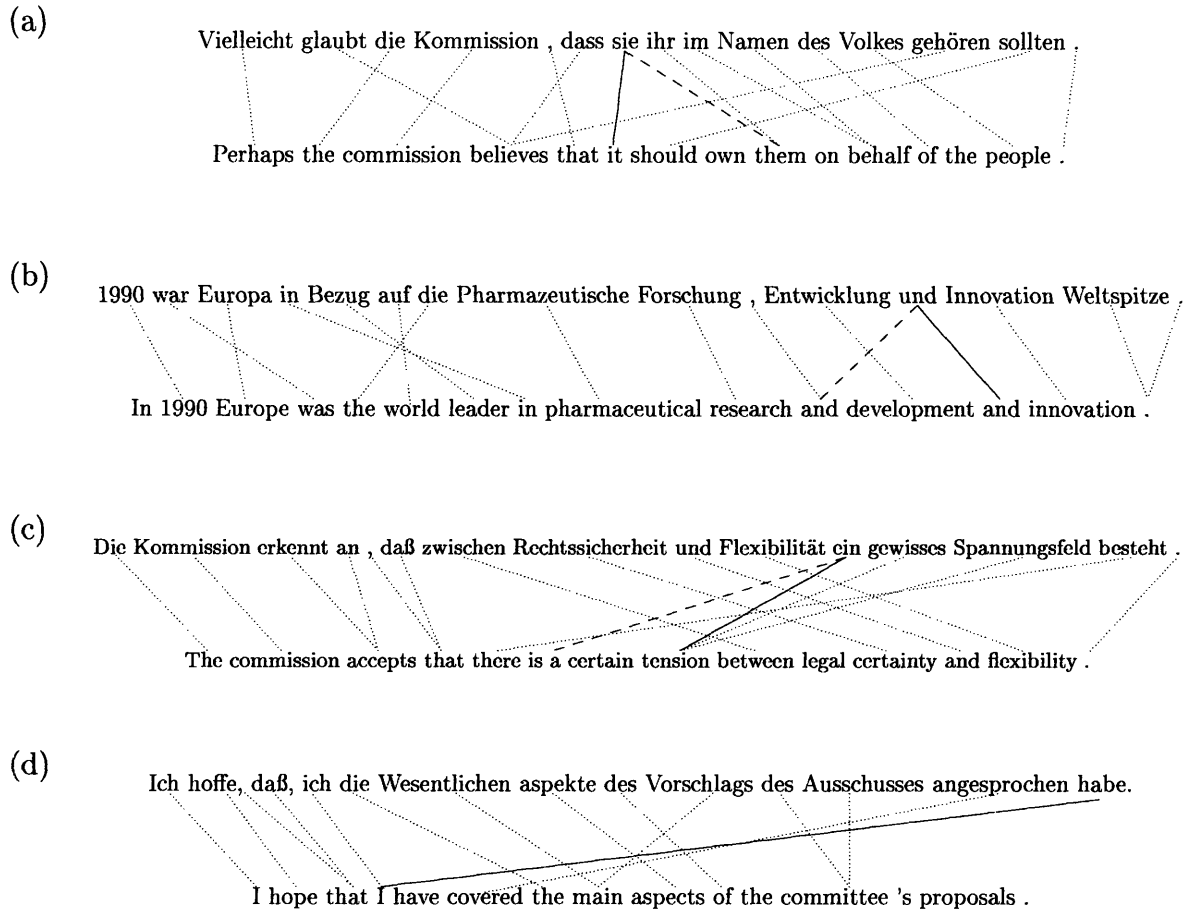


Figure 3-2: Alignments of four German-English sentence pairs. The dotted lines indicate alignments common to the output of both the univariate and multivariate models; the dashed line is only in the univariate model and the solid line is only in the multivariate model. (a) Improved pronoun alignment. Due to the covariance of neighboring words, the multivariate model aligns the German pronoun "sie" correctly. (b) Improved multiple particle alignment. Multiple instances of a particle (or similar particles) are aligned correctly in the multi-variate model despite the presence of multiple high-scoring candidate words in the English sentence, which confuses the univariate Gaussian model. (c) Article misalignment. Due to the frequency with which German articles appear without English analogs, German articles sometimes align to the English word that corresponds to the German word that it modifies even if there is also a corresponding English article. (d) Infinitive misalignment. It is not clear why this happens, though we imagine that frequently occurring verbs (such as "habe") occur so frequently that apparently pronouns are likely candidates for translation in the translation model. Since the last word in a German sentence often aligns to a word far from the second-to-last word, "habe" tried to align to a word away from the word that "angesprochen" aligned to.

Chapter 4

Conclusions and Future Work

We have greatly reduced the number of parameters required to achieve current state-of-the-art translation quality. We have also shown one choice of further parameters that improves alignment quality slightly. One obvious path for future work is the exploration of other choices for how to make use of these parameters.

Syntactic models are particularly attractive. Introducing an alignment variable for each part-of-speech and each source-sentence position, for instance, would be a promising next step. A natural extension of this idea would be to add the position within a parse tree as another feature.

Introducing covariances to neighboring words was an improvement, to the point that we began to wonder if phrases could be induced from the covariances themselves (i.e., if words are coaligned "due" to the presence of covariance, we group them into phrases).

Unfortunately, low-quality translation tables became an issue in our experiments. The "garbage collection" property of the IBM models (the tendency to align many words in a target sentence to a new word in the source sentence) is an area where we would like to see improvement.

Finally, we hate to echo this unending complaint, but we believe that serious improvement could be achieved simply by having cleaner data sources. Parallel corpora are riddled with noise: very often, corresponding sentences are syntactically unrelated; fairly frequently, entire clauses in one sentence are absent in its analog; oc-

cassionally, complete misalignment occurs and two unrelated sentences are paired. We realize that noise will never be eliminated, but we hope that someday soon corpora will be generated specifically for this purpose: pre-aligned, syntactically-motivated translations.

Appendix A

EM Algorithm Reference

A.1 Definitions

$$
\begin{aligned}
\mathcal{L}(\Theta) &= \sum_X \log P(X;\Theta) && \text{Likelihood} \\
Q(\Theta',\Theta) &= \sum_X E_Y\left(\log P(X,Y;\Theta') \mid X;\Theta\right) \\
H(\Theta',\Theta) &= -\sum_X E_Y\left(\log P(Y \mid X;\Theta') \mid X;\Theta\right) && \text{Cross-Entropy} \\
KL(\Theta',\Theta) &= H(\Theta',\Theta) - H(\Theta,\Theta) && \text{KL-Divergence} \\
\Theta^{(i)} &= \arg\max_{\Theta}\ Q\left(\Theta,\Theta^{(i-1)}\right)
\end{aligned}
$$

A.2 Lemmas

Jensen's Inequality:

$$E(\log(X)) \leq \log(E(X))$$

Ali's Identity:

$$E_X\left(\frac{f(X)}{P(X)}\right) = \int_X f(X)\, dX$$

Lemma:

$$H(\Theta,\Theta) \leq H(\Theta',\Theta) \qquad \forall\ \Theta,\Theta'$$

Equivalently:

$$KL(\Theta', \Theta) \geq 0 \qquad \forall\, \Theta, \Theta'$$

Proof:

$$\begin{aligned}
\forall\, X \qquad 0 &= \log(1) = \log\left(\int_Y P\left(Y \mid X; \Theta'\right)\, dY\right) \\
&= \log\left(\int_Y P\left(Y \mid X; \Theta\right) \cdot \frac{P\left(Y \mid X; \Theta'\right)}{P\left(Y \mid X; \Theta\right)}\, dY\right) \\
&= \log E_Y\left(\frac{P\left(Y \mid X; \Theta'\right)}{P\left(Y \mid X; \Theta\right)} \,\middle|\, X; \Theta\right) \\
&\geq E_Y\left(\log\left[\frac{P\left(Y \mid X; \Theta'\right)}{P\left(Y \mid X; \Theta\right)}\right] \,\middle|\, X; \Theta\right) \\
&= E_Y\left(\log P\left(Y \mid X; \Theta'\right) \mid X; \Theta\right) - E_Y\left(\log P\left(Y \mid X; \Theta\right) \mid X; \Theta\right) \\
E_X(0) = 0 \;&\geq E_X\left(E_Y\left(\log P\left(Y \mid X; \Theta'\right) \mid X; \Theta\right)\right) - E_X\left(E_Y\left(\log P\left(Y \mid X; \Theta\right) \mid X; \Theta\right)\right) \\
&= H(\Theta, \Theta) - H(\Theta', \Theta).
\end{aligned}$$

$$\Rightarrow \qquad H(\Theta, \Theta) \leq H(\Theta', \Theta)$$

A.3 EM is Nondecreasing

Theorem:

$$\boxed{\mathcal{L}\left(\Theta^{(i)}\right) \leq \mathcal{L}\left(\Theta^{(i+1)}\right)}$$

Proof:

$$\begin{aligned}
\mathcal{L}\left(\Theta\right) &= \sum_X \log P\left(X; \Theta\right) \\
&= \sum_X E_Y\left(\log P\left(X; \Theta\right) \mid X; \Theta^{(i)}\right) \\
&= \sum_X E_Y\left(\log P\left(X; \Theta\right) \mid X; \Theta^{(i)}\right) - Q\left(\Theta, \Theta^{(i)}\right) + Q\left(\Theta, \Theta^{(i)}\right) \\
&= \sum_X \left(E_Y\left(\log P\left(X; \Theta\right) \mid X; \Theta^{(i)}\right) - E_Y\left(\log P\left(X, Y; \Theta\right) \mid X; \Theta^{(i)}\right)\right) \\
&\qquad + Q\left(\Theta, \Theta^{(i)}\right) \\
&= -\sum_X E_Y\left(\log\left(\frac{P\left(X, Y; \Theta\right)}{P\left(X; \Theta\right)}\right) \,\middle|\, X; \Theta^{(i)}\right) + Q\left(\Theta, \Theta^{(i)}\right)
\end{aligned}$$

$$
\begin{aligned}
&= -\sum_X E_Y\left(\log P\left(Y \mid X;\Theta\right) \mid X;\Theta^{(i)}\right) + Q\left(\Theta,\Theta^{(i)}\right) \\
&= H\left(\Theta,\Theta^{(i)}\right) + Q\left(\Theta,\Theta^{(i)}\right). \\
\mathcal{L}\left(\Theta^{(i+1)}\right) - \mathcal{L}\left(\Theta^{(i)}\right) &= \left[H\left(\Theta^{(i+1)},\Theta^{(i)}\right) - H\left(\Theta^{(i)},\Theta^{(i)}\right)\right] \\
&\qquad + \left[Q\left(\Theta^{(i+1)},\Theta^{(i)}\right) - Q\left(\Theta^{(i+1)},\Theta^{(i)}\right)\right] \\
&\geq 0.
\end{aligned}
$$

Thus, the likelihood of successive EM parameter vectors is non-decreasing. (This is a long way from convergence proof...)

Incidentally, we have also shown that, $\forall\ \Theta,\ \Theta'$:

$$\boxed{\mathcal{L}\left(\Theta\right) = H\left(\Theta,\Theta'\right) + Q\left(\Theta,\Theta'\right)}$$

A.4 EM on Multinomials

EM is easy in the special case when $P(X,Y \mid \Theta)$ is a multinomial distribution; that is, it can be written in the form:

$$P(X,Y;\Theta) = \prod_{r=1}^{N} \Theta_r^{\text{Count}_r(X,Y)}.$$

This is a form that occurs very often in language processing tasks.

Let's go!

$$
\begin{aligned}
Q(\Theta',\Theta) &= \sum_X E_Y(\log P(X,Y;\Theta') \mid X;\Theta) \\
&= \sum_X \sum_{r=1}^{N} E_Y\left(\text{Count}_r(X,Y) \mid X;\Theta\right) \log \Theta'_r \\
&= \sum_{r=1}^{N} \left(\sum_X E_Y\left(\text{Count}_r(X,Y) \mid X;\Theta\right)\right) \cdot \log \Theta'_r
\end{aligned}
$$

It's easy to write an algorithm to maximize Q; we just have to set each Θ'_r to its coefficient and normalize (in ways corresponding to inherent constraints of the pa-

rameters):

$$\Theta'_r \propto \sum_X E_Y \left(\text{Count}_r(X,Y) \mid X;\Theta\right).$$

Let's formalize the normalization; the indices $r \in \{1,2,\ldots,N\}$ are partitioned into disjoint subsets R_1, R_2,..., R_m such that $\sum_{r\in R_i} \Theta_r = 1$. Now we set:

$$\Theta'_r = \frac{\sum_X E_Y \left(\text{Count}_r(X,Y) \mid X;\Theta\right)}{\sum_{r'\in R_i} \sum_X E_Y \left(\text{Count}_{r'}(X,Y) \mid X;\Theta\right)} \qquad \forall\, r \in R_i$$

A.5 Parametrizing Multinomials

Suppose we wish to further parametrize the parameters Θ by another set of parameters α; that is, we define a set of events E_r and set $\Theta_r = P(E_r;\alpha)$ whilst preserving the normalization conditions on Θ_r (i.e., that $\sum_{r\in R_i} P(E_r;\alpha) = 1$, $\forall\, R_i$); thus,

$$P(X,Y;\alpha) = \prod_{r=1}^{N} P(E_r;\alpha)^{\text{Count}_r(X,Y)}.$$

We assume that we can easily find maximum-likelihood α given E_r data (that is, the number of occurences of each event E_r—not necessarily integral). We proceed:

$$\begin{aligned} Q(\alpha',\alpha) &= \sum_X E_Y(\log P(X,Y;\alpha') \mid X;\alpha) \\ &= \sum_{r=1}^{N} \left(\sum_X E_Y(\text{Count}_r(X,Y) \mid X;\alpha) \right) \log P(E_r;\alpha') \end{aligned}$$

Then, the EM update of α is simply the maximum-likelihood α' where event E_r has occurred this many times:

$$\sum_X E_Y(\text{Count}_r(X,Y) \mid X,\alpha).$$

Note that these counts are pre-normalization! When the α_i are "grouped" the same

way as the Θ_r, the normalization does not enter into the picture (that is, when

$$P(E_r; \alpha') = P(E_r; \alpha'_i) \qquad \forall r \in R_i,$$

where the R_i are defined as above to be sets of Θ parameters that must be normalized); consequently, the Q-maximizing Θ_r can themselves be used as counts to maximize the likelihood of α. Again, if, for any i, we were to multiply the coefficients of Θ_r for $r \in R_i$ by a constant, the maximum likelihood values do not change; thus, no normalization is necessary.

[It is easy to see that normalization can be harmful; consider, for instance, the following experiment: we repeatedly select one of two biased coins to flip and record which coin we flipped and the outcome. Then, the maximum-likelihood probability that the first coin will flip heads, for instance, is the number of heads we got from the first coin divided by the number of times we flipped the first coin. Suppose, however, that we add the constraint that the coins are identically biased. Then the number of times we flipped each coin is important; we cannot correctly estimate the probability of heads with the unconstrained maximum-likelihood probability of heads for each coin alone.]

A.6 EM on IBM2+1dG

Here we give the full derivation of the EM updates for the one-dimensional gaussian framework described in chapter 2 for the sake of the mathematically skeptical.

Let's begin by defining the model:

- Training Data

  The training data consists of several triplets $(\mathbf{e}, \mathbf{f}, \vec{a})$.

  1. $\mathbf{e}$ and $\mathbf{f}$ are English and French sentences, respectively, that are translations of each other (observed in training).
  2. $\vec{a} \in \{0, \ldots, \ell\}^m$ represents an alignment between the sentences, where $\ell =$

$|\mathbf{e}|$ and $m = |\mathbf{f}|$ and $a_j = i$ implies that the ith English word corresponds to the jth French word (hidden in training).

- Parameters

  1. $T(f \mid e)$ for all French words f and English words e (the NULL word is added to the English vocabulary).
  2. $\mu_{j,\ell,m}$, $\sigma_{j,\ell,m}$ for all French sentence lengths m, English sentence lengths ℓ, and French word indices $j \in \{1, \ldots, m\}$, respectively the mean and standard deviation of the index of the corresponding English word, given that it is not the NULL word.
  3. $N(j, \ell, m)$ for all French sentence lengths m, English sentence lengths ℓ, and French word indices $j \in \{1, \ldots, m\}$, the probability that that French word is aligned to the English NULL word.

- Model

$$P(\mathbf{f}, \vec{a} \mid \mathbf{e}; T, \mu, \sigma) = \prod_{j=1}^{m} T(\mathbf{f}_j \mid \mathbf{e}_{a_j}) D(a_j \mid j, \ell, m)$$

where

$$D(a_j \mid j, \ell, m) = \begin{cases} (1 - N(j, \ell, m)) \cdot f_{\mathcal{N}}(a_j \mid \mu_{j,\ell,m}, \sigma_{j,\ell,m}) & j \neq 0 \\ N(j, \ell, m) & j = 0 \end{cases}$$

Note that this model is deficient; that is, we are not enforcing the proper normalization constraints on D.

- Normalization conditions

$$\sum_{f} T(f \mid e) = 1 \qquad \forall\, e$$

- Parametrized Multinomial

$$P(\mathbf{f}, \vec{a} \mid \mathbf{e}; T, \mu, \sigma) = \prod_{f,e} T(f \mid e)^{\text{Count}_{f,e}(\mathbf{e},\mathbf{f},\vec{a})} \times \prod_{i,j,\ell,m} D(i \mid j, \ell, m)^{\text{Count}_{i,j,\ell,m}(\mathbf{e},\mathbf{f},\vec{a})}$$

where

$$\begin{aligned}\text{Count}_{f,e}(\mathbf{e},\mathbf{f},\vec{a}) &= \sum_{j=1}^{m}\delta(f,\mathbf{f}_j)\delta(e,\mathbf{e}_{a_j})\\ \text{Count}_{i,j,\ell,m}(\mathbf{e},\mathbf{f},\vec{a}) &= \delta(\ell,|\mathbf{e}|)\delta(m,|\mathbf{f}|)\delta(a_j,i),\end{aligned}$$

where $\delta(\cdot,\cdot)$ denotes the Kronecker delta function.

We parametrize T and D by T, μ, σ, and N. Thus, T is trivially parametrized by itself, whereas D is parametrized as described above by μ, σ, and N.

We wish to maximize the function $Q((T,\mu,\sigma,N),(T^i,\mu^i,\sigma^i,N^i))$ given T^i, μ^i, σ^i, and N^i. Since the model can neatly be factored into a term that depends on T alone and a term that depends on D alone, we can optimize these parameters independently. Clearly, the EM procedure for finding the optimal T is unchanged from Model 2; thus, we need only focus on finding the optimal values for μ, σ, and N. We compute:

$$\begin{aligned}&Q((\mu,\sigma,N),(T',\mu',\sigma',N'))\\ &= \sum_{\mathbf{e},\mathbf{f}}\sum_{\vec{a}} P(\vec{a}\,|\,\mathbf{e},\mathbf{f},T',N',\mu',\sigma')\log P(\mathbf{f},\vec{a}\,|\,\mathbf{e};N,\mu,\sigma)\\ &= \sum_{i,j,\ell,m}\Bigg[\underbrace{\Big(\sum_{\mathbf{e},\mathbf{f}} E_{\vec{a}}(\text{Count}_{i,j,\ell,m}(\mathbf{e},\mathbf{f},\vec{a})\,|\,\mathbf{e},\mathbf{f};T',\mu',\sigma',N')\Big)}_{C(i,j,\ell,m)}\log D(i\,|\,j,\ell,m)\Bigg]\\ &= \sum_{j,\ell,m} C(0,j,\ell,m)\log N(j,\ell,m)\\ &\qquad + \sum_{j,\ell,m}\sum_{i=1}^{\ell} C(i,j,\ell,m)\log[(1-N(j,\ell,m))f_{\mathcal{N}}(i\,|\,\mu_{i,j,\ell,m},\sigma_{i,j,\ell,m})]\\ &= \sum_{j,\ell,m} C(0,j,\ell,m)\log N(j,\ell,m) + \sum_{j,\ell,m}\sum_{i=1}^{\ell} C(i,j,\ell,m)\log(1-N(j,\ell,m))\\ &\qquad + \sum_{j,\ell,m}\sum_{i=1}^{\ell} C(i,j,\ell,m)\log f_{\mathcal{N}}(i\,|\,\mu_{i,j,\ell,m},\sigma_{i,j,\ell,m}).\end{aligned}$$

Thus, the optimal value of N is given by:

$$N(j,\ell,m) = \frac{C(0,j,\ell,m)}{\sum_{i=0}^{\ell} C(i,j,\ell,m)}$$

and the μ and σ are optimized by their usual maximum-likelihood estimators:

$$\begin{aligned}
\mu_{j,\ell,m} &= \sum_{i=1}^{\ell} i\, C(i,j,\ell,m) \Big/ \sum_{i=1}^{\ell} C(i,j,\ell,m) \\
\sigma^2_{j,\ell,m} &= \sum_{i=1}^{\ell} (i-\mu_{j,\ell,m})^2\, C(i,j,\ell,m) \Big/ \sum_{i=1}^{\ell} C(i,j,\ell,m)
\end{aligned}$$

Bibliography

[1] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Association for Computational Linguistics*, 19(2):263–311, 1993.

[2] Michael Collins, Philipp Koehn, and Ivona Kucerova. Clause restructuring for statistical machine translation. pages 531–540. Association for Computational Linguistics, July 2005.

[3] F. Jelinek, R. L. Mercer, L. Bahl, and J. Baker. Interpolated estimation of Markov source parameters from sparse data. Pattern Recognition in Practice, May 1980.

[4] Frederick Jelinek. *Statistical methods for speech recognition.* MIT Press, Cambridge, MA, USA, 1997.

[5] Philipp Koehn. Europarl: A multilingual corpus for evaluation of machine translation. *Unpublished*, December 2002.

[6] Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. pages 48–54. Association for Computational Linguistics, May–June 2003.

[7] Daniel Marcu and William Wong. A phrase-based, joint probability model for statistical machine translation. pages 133–139. Association for Computational Linguistics, July 2002.

[8] Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

[9] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. pages 311–318. Association for Computational Linguistics, July 2002.

[10] E. Polak. *Optimization: Algorithms and Consistent Approximations.* Springer, New York, 1997.

[11] R. P. Sherman, Y. Y. K. Ho, and S. R. Dalal. Conditions for convergence of monte carlo em sequences with an application to product diffusion modeling. *The Econometrics Journal*, 1999.

[12] Greg C. G. Wei and Matrin A. Tanner. Conditions for convergence of monte carlo em sequences with an application to product diffusion modeling. *Journal of the American Statistical Association*, 85(411):699–704, September 1990.

[13] Eric W. Weisstein. Central limit theorem. *MathWorld.*