A Maximum Entropy Approach to Natural Language Processing

Adam L. Berger*
Stephen A. Della Pietra†
Vincent J. Della Pietra†

IBM T.J. Watson Research Center[1]
P.O. Box 704
Yorktown Heights, NY 10598

The concept of maximum entropy can be traced back along multiple threads to Biblical times. Only recently, however, have computers become powerful enough to permit the widescale application of this concept to real world problems in statistical estimation and pattern recognition. In this paper we describe a method for statistical modeling based on maximum entropy. We present a maximum-likelihood approach for automatically constructing maximum entropy models and describe how to implement this approach efficiently, using as examples several problems in natural language processing.

1. Introduction

Statistical modeling addresses the problem of constructing a stochastic model to predict the behavior of a random process. In constructing this model, we typically have at our disposal a sample of output from the process. Given this sample, representing an incomplete state of knowledge about the process, the modeling problem is to parlay this knowledge into a representation of the process. We can then use this representation to make predictions of the future behavior of the process.

Baseball managers (who rank among the better paid statistical modelers) employ batting averages, compiled from a history of at-bats, to gauge the likelihood that a player will succeed in his next appearance at the plate. Thus informed, they manipulate their lineups accordingly. Wall Street speculators (who rank among the *best* paid statistical modelers) build models based on past stock price movements to predict tomorrow's fluctuations and alter their portfolios to capitalize on the predicted future. At the other end of the pay scale reside natural language researchers, who design language and acoustic models for use in speech recognition systems and related applications.

The past few decades have witnessed significant progress toward increasing the predictive capacity of statistical models of natural language. In language modeling, for instance, (Bahl *et al* 1989) have used decision tree models and (Della Pietra *et al* 1994) have used automatically inferred link grammars to model long range correlations in language. In parsing, (Black *et al* 1992) has described how to extract grammatical rules from

* Now at Columbia University computer science department
† Now at Renaissance Technologies, Stony Brook, NY
1 Research supported in part by ARPA under grant ONR N00014-91-C-0135

© 1996 Association for Computational Linguistics

annotated text automatically and incorporate these rules into statistical models of grammar. In speech recognition, (Lucassen and Mercer 1984) have introduced a technique for automatically discovering relevant features for the translation of word spelling to word pronunciation.

These efforts, while varied in specifics, all confront two essential tasks of statistical modeling. The first task is to determine a set of statistics which capture the behavior of a random process. Given a set of statistics, the second task is to corral these facts into an accurate model of the process—a model capable of predicting the future output of the process. The first task is one of feature selection; the second is one of model selection. In the following pages we present a unified approach to these two tasks based on the maximum entropy philosophy.

Our discussion will proceed as follows. In Section 2 we give an overview of the maximum entropy philosophy and work through a motivating example. In Section 3 we describe the mathematical structure of maximum entropy models and give an efficient algorithm for estimating the parameters of such models. In Section 4 we discuss feature selection, and present an automatic method for discovering facts about a process from a sample of output from the process. We then present a series of refinements to the method to make it practical to implement. Finally, in Section 5 we describe the application of maximum entropy ideas to several tasks in stochastic language processing: bilingual sense disambiguation, word reordering, and sentence segmentation.

2. A Maximum Entropy Overview

We introduce the concept of maximum entropy through a simple example. Suppose we wish to model an expert translator's decisions concerning the proper French rendering of the English word *in*. Our model p of the expert's decisions assigns to each French word or phrase f an estimate, $p(f)$, of the probability that the expert would choose f as a translation of *in*. To guide us in developing p, we collect a large sample of instances of the expert's decisions. Our goal is to extract a set of facts about the decision-making process from the sample (the first task of modeling) that will aid us in constructing a model of this process (the second task).

One obvious clue we might glean from the sample is the list of allowed translations. For example, we might discover that the expert translator always chooses among the following five French phrases: {*dans*, *en*, *à*, *au cours de*, *pendant*}. With this information in hand, we can impose our first constraint on our model p:

$$p(dans) + p(en) + p(\grave{a}) + p(au\ cours\ de) + p(pendant) = 1$$

This equation represents our first statistic of the process; we can now proceed to search for a suitable model which obeys this equation. Of course, there are an infinite number of models p for which this identity holds. One model which satisfies the above equation is $p(dans) = 1$; in other words, the model always predicts *dans*. Another model which obeys this constraint predicts *pendant* with a probability of 1/2, and *à* with a probability of 1/2. But both of these models offend our sensibilities: knowing only that the expert always chose from among these five French phrases, how can we justify either of these probability distributions? Each seems to be making rather bold assumptions, with no empirical justification. Put another way, these two models assume more than we actually know about the expert's decision-making process. All we know is that the expert chose exclusively from among these five French phrases; given this, the most intuitively appealing model is the following:

$$\begin{aligned} p(dans) &= 1/5 \\ p(en) &= 1/5 \\ p(\grave{a}) &= 1/5 \\ p(au\ cours\ de) &= 1/5 \\ p(pendant) &= 1/5 \end{aligned}$$

This model, which allocates the total probability evenly among the five possible phrases, is the most uniform model subject to our knowledge. It is not, however, the most uniform overall; that model would grant an equal probability to every *possible* French phrase.

We might hope to glean more clues about the expert's decisions from our sample. Suppose we notice that the expert chose either *dans* or *en* 30% of the time. We could apply this knowledge to update our model of the translation process by requiring that p satisfy two constraints:

$$\begin{gathered} p(dans) + p(en) = 3/10 \\ p(dans) + p(en) + p(\grave{a}) + p(au\ cours\ de) + p(pendant) = 1 \end{gathered}$$

Once again there are many probability distributions consistent with these two constraints. In the absence of any other knowledge, a reasonable choice for p is again the most uniform—that is, the distribution which allocates its probability as evenly as possible, subject to the constraints:

$$\begin{aligned} p(dans) &= 3/20 \\ p(en) &= 3/20 \\ p(\grave{a}) &= 7/30 \\ p(au\ cours\ de) &= 7/30 \\ p(pendant) &= 7/30 \end{aligned}$$

Say we inspect the data once more, and this time notice another interesting fact: in half the cases, the expert chose either *dans* or *à*. We can incorporate this information into our model as a third constraint:

$$\begin{gathered} p(dans) + p(en) = 3/10 \\ p(dans) + p(en) + p(\grave{a}) + p(au\ cours\ de) + p(pendant) = 1 \\ p(dans) + p(\grave{a}) = 1/2 \end{gathered}$$

We can once again look for the most uniform p satisfying these constraints, but now the choice is not as obvious. As we have added complexity, we have encountered two difficulties at once. First, what exactly is meant by "uniform," and how can one measure the uniformity of a model? Second, having determined a suitable answer to these questions, how does one go about finding the most uniform model subject to a set of constraints like those we have described?

The maximum entropy method answers both these questions, as we will demonstrate in the next few pages. Intuitively, the principle is simple: model all that is known and assume nothing about that which is unknown. In other words, given a collection of facts, choose a model which is consistent with all the facts, but otherwise as uniform as possible. This is precisely the approach we took in selecting our model p at each step in the above example.

The maximum entropy concept has a long history. Adopting the least complex hypothesis possible is embodied in Occam's Razor ("Nunquam ponenda est pluralitas sine necesitate") and even appears earlier, in the Bible and the writings of Herotodus (Jaynes 1990). Laplace might justly be considered the father of maximum entropy, having enunciated the underlying theme 200 years ago in his "Principle of Insufficient Reason": when one has no information to distinguish between the probability of two events, the best strategy is to consider them equally likely (Guiasu and Shenitzer 1994). As E.T. Jaynes, a more recent pioneer of maximum entropy, put it (Jaynes 1990):

> *...the fact that a certain probability distribution maximizes entropy subject to certain constraints representing our incomplete information, is the fundamental property which justifies use of that distribution for inference; it agrees with everything that is known, but carefully avoids assuming anything that is not known. It is a transcription into mathematics of an ancient principle of wisdom...*

3. Maximum Entropy Modeling

We consider a random process which produces an output value y, a member of a finite set $\mathcal{Y}$. For the translation example just considered, the process generates a translation of the word *in*, and the output y can be any word in the set {*dans*, *en*, *à*, *au cours de*, *pendant*}. In generating y, the process may be influenced by some contextual information x, a member of a finite set $\mathcal{X}$. In the present example, this information could include the words in the English sentence surrounding *in*.

Our task is to construct a stochastic model that accurately represents the behavior of the random process. Such a model is a method of estimating the conditional probability that, given a context x, the process will output y. We will denote by $p(y|x)$ the probability that the model assigns to y in context x. With a slight abuse of notation, we will also use $p(y|x)$ to denote the entire conditional probability distribution provided by the model, with the interpretation that y and x are placeholders rather than specific instantiations. The proper interpretation should be clear from the context. We will denote by $\mathcal{P}$ the set of all conditional probability distributions. Thus a model $p(y|x)$ is, by definition, just an element of $\mathcal{P}$.

3.1 Training Data

To study the process, we observe the behavior of the random process for some time, collecting a large number of samples $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$. In the example we have been considering, each sample would consist of a phrase x containing the words surrounding *in*, together with the translation y of *in* which the process produced. For now we can imagine that these training samples have been generated by a human expert who was presented with a number of random phrases containing *in* and asked to choose a good translation for each. When we discuss real-world applications in Section 5, we will show how such samples can be automatically extracted from a bilingual corpus.

We can summarize the training sample in terms of its empirical probability distribution $\tilde{p}$, defined by

$$\tilde{p}(x,y) \equiv \frac{1}{N} \times \text{ number of times that } (x,y) \text{ occurs in the sample}$$

Typically, a particular pair (x,y) will either not occur at all in the sample, or will occur at most a few times.

3.2 Statistics, Features and Constraints

Our goal is to construct a statistical model of the process which generated the training sample $\tilde{p}(x,y)$. The building blocks of this model will be a set of statistics of the training sample. In the current example we have employed several such statistics: the frequency that *in* translated to either *dans* or *en* was 3/10; the frequency that it translated to either *dans* or *au cours de* was 1/2; and so on. These particular statistics were independent of the context, but we could also consider statistics which depend on the conditioning information x. For instance, we might notice that, in the training sample, if *April* is the word following *in*, then the translation of *in* is *en* with frequency 9/10.

To express the event that *in* translates as *en* when *April* is the following word, we can introduce the indicator function

$$f(x,y) = \begin{cases} 1 & \text{if } y = en \text{ and } April \text{ follows } in \\ 0 & \text{otherwise} \end{cases}$$

The expected value of f with respect to the empirical distribution $\tilde{p}(x,y)$ is exactly the statistic we are interested in. We denote this expected value by

$$\tilde{p}(f) \equiv \sum_{x,y} \tilde{p}(x,y) f(x,y) \tag{1}$$

We can express any statistic of the sample as the expected value of an appropriate binary-valued indicator function f. We call such function a *feature function* or *feature* for short. (As with probability distributions, we will sometimes abuse notation and use $f(x,y)$ to denote both the value of f at a particular pair (x,y) as well as the entire function f.)

When we discover a statistic that we feel is useful, we can acknowledge its importance by requiring that our model accord with it. We do this by constraining the expected value that the model assigns to the corresponding feature function f. The expected value of f with respect to the model $p(y|x)$ is

$$p(f) \equiv \sum_{x,y} \tilde{p}(x) p(y|x) f(x,y) \tag{2}$$

where $\tilde{p}(x)$ is the empirical distribution of x in the training sample. We constrain this expected value to be the same as the expected value of f in the training sample. That is, we require

$$p(f) = \tilde{p}(f) \tag{3}$$

Combining (1), (2) and (3) yields the more explicit equation

$$\sum_{x,y} \tilde{p}(x) p(y|x) f(x,y) = \sum_{x,y} \tilde{p}(x,y) f(x,y)$$

We call the requirement (3) a *constraint equation* or simply a *constraint*. By restricting attention to those models $p(y|x)$ for which (3) holds, we are eliminating from consideration those models which do not agree with the training sample on how often the output of the process should exhibit the feature f.

To sum up so far, we now have a means of representing statistical phenomena inherent in a sample of data (namely, $\tilde{p}(f)$), and also a means of requiring that our model of the process exhibit these phenomena (namely, $p(f) = \tilde{p}(f)$).

One final note about features and constraints bears repeating: though the words "feature" and "constraint" are often used interchangeably in discussions of maximum entropy, we will be vigilant to distinguish the two and urge the reader to do likewise: a feature is a binary-valued function of (x, y); a constraint is an equation between the expected value of the feature function in the model and its expected value in the training data.

3.3 The Maximum Entropy Principle

Suppose that we are given n feature functions f_i, which determine statistics we feel are important in modeling the process. We would like our model to accord with these statistics. That is, we would like p to lie in the subset $\mathcal{C}$ of $\mathcal{P}$ defined by

$$\mathcal{C} \equiv \Big\{ \; p \in \mathcal{P} \mid p(f_i) = \tilde{p}(f_i) \quad \text{for } i \;\in\; \{1, 2, \ldots, n\} \Big\} \tag{4}$$

Figure 1 provides a geometric interpretation of this setup. Here $\mathcal{P}$ is the space of all (unconditional) probability distributions on 3 points, sometimes called a *simplex*. If we impose no constraints (depicted in (a)), then all probability models are allowable. Imposing one linear constraint $\mathcal{C}_1$ restricts us to those $p \in \mathcal{P}$ which lie on the region defined by $\mathcal{C}_1$, as shown in (b). A second linear constraint could determine p exactly, if the two constraints are satisfiable; this is the case in (c), where the intersection of $\mathcal{C}_1$ and $\mathcal{C}_2$ is non-empty. Alternatively, a second linear constraint could be inconsistent with the first—for instance, the first might require that the probability of the first point is 1/3 and the second that the probability of the third point is 3/4—this is shown in (d). In the present setting, however, the linear constraints are extracted from the training sample and cannot, by construction, be inconsistent. Furthermore, the linear constraints in our applications will not even come close to determining $p \in \mathcal{P}$ uniquely as they do in (c); instead, the set $\mathcal{C} = \mathcal{C}_1 \cap \mathcal{C}_2 \cap \ldots \cap \mathcal{C}_n$ of allowable models will be infinite.

Among the models $p \in \mathcal{C}$, the maximum entropy philosophy dictates that we select the distribution which is most uniform. But now we face a question left open in Section 2: what does "uniform" mean?

A mathematical measure of the uniformity of a conditional distribution $p(y|x)$ is provided by the conditional entropy[2]

$$H(p) \equiv -\sum_{x,y} \tilde{p}(x) p(y|x) \log p(y|x) \tag{5}$$

The entropy is bounded from below by zero, the entropy of a model with no uncertainty at all, and from above by $\log|\mathcal{Y}|$, the entropy of the uniform distribution over all possible

2 A more common notation for the conditional entropy is $H(Y \mid X)$, where Y and X are random variables with joint distribution $\tilde{p}(x)p(y|x)$. To emphasize the dependence of the entropy on the probability distribution p, we have adopted the alternate notation $H(p)$.

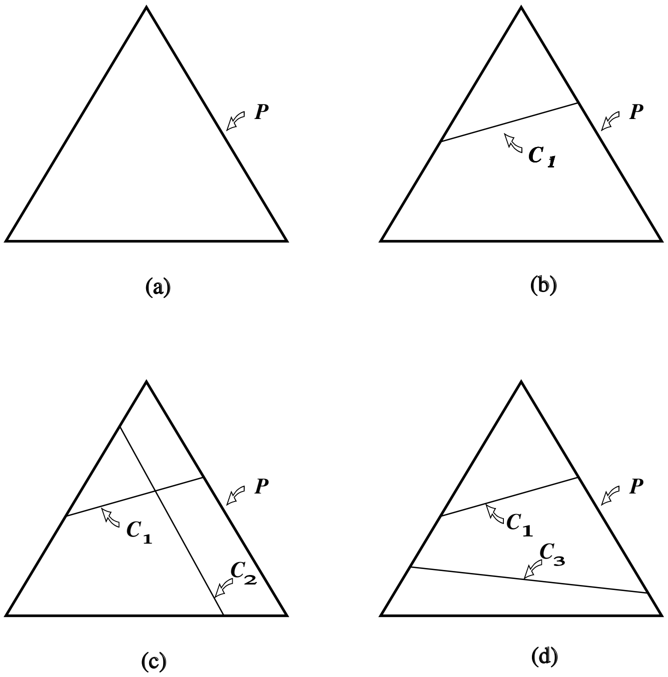

Figure 1
Four different scenarios in constrained optimization. $\mathcal{P}$ represents the space of all probability distributions. In (a), no constraints are applied, and all $p \in \mathcal{P}$ are allowable. In (b), the constraint $\mathcal{C}_1$ narrows the set of allowable models to those which lie on the line defined by the linear constraint. In (c), two consistent constraints $\mathcal{C}_1$ and $\mathcal{C}_2$ define a single model $p \in \mathcal{C}_1 \cap \mathcal{C}_2$. In (d), the two constraints are inconsistent (i.e. $\mathcal{C}_1 \cap \mathcal{C}_3 = \emptyset$); no $p \in \mathcal{P}$ can satisfy them both.

$|\mathcal{Y}|$ values of y. With this definition in hand, we are ready to present the principle of maximum entropy.

> To select a model from a set $\mathcal{C}$ of allowed probability distributions, choose the model $p_\star \in \mathcal{C}$ with maximum entropy $H(p)$:
>
> $$p_\star = \operatorname*{argmax}_{p \in \mathcal{C}} H(p) \tag{6}$$

It can be shown that $p_\star$ is always well-defined; that is, there is always a unique model $p_\star$ with maximum entropy in any constrained set $\mathcal{C}$.

3.4 Parametric Form

The maximum entropy principle presents us with a problem in constrained optimization: find the $p_\star \in \mathcal{C}$ which maximizes $H(p)$. In simple cases, we can find the solution to this problem analytically. This was true for the example presented in Section 2 when we imposed the first two constraints on p. Unfortunately, the solution of the general maximum entropy cannot be written explicitly, and we need a more indirect approach. (The reader is invited to try to calculate the solution for the same example when the third constraint is imposed.)

To address the general problem, we apply the method of Lagrange multipliers from the theory of constrained optimization. The relevant steps are outlined here; the reader is referred to (Della Pietra *et al* 1995) for a more thorough discussion of constrained optimization as applied to maximum entropy.

- We will refer to the original constrained optimization problem,

  $$\text{find } p_\star = \operatorname*{argmax}_{p \in \mathcal{C}} H(p)$$

  as the *primal* problem.

- For each feature f_i we introduce a parameter λ_i (a Lagrange multiplier). We define the Lagrangian $\Lambda(p, \lambda)$ by

  $$\Lambda(p, \lambda) \equiv H(p) + \sum_i \lambda_i \left(p(f_i) - \tilde{p}(f_i) \right) \tag{7}$$

- Holding λ fixed, we compute the unconstrained maximum of the Lagrangian $\Lambda(p, \lambda)$ over all $p \in \mathcal{P}$. We denote by p_λ the p where $\Lambda(p, \lambda)$ achieves its maximum and by $\Psi(\lambda)$ the value at this maximum:

  $$p_\lambda \equiv \operatorname*{argmax}_{p \in \mathcal{P}} \Lambda(p, \lambda) \tag{8}$$

  $$\Psi(\lambda) \equiv \Lambda(p_\lambda, \lambda) \tag{9}$$

  We call $\Psi(\lambda)$ the *dual* function. The functions p_λ and $\Psi(\lambda)$ may be calculated explicitly using simple calculus. We find

  $$p_\lambda(y|x) = \frac{1}{Z_\lambda(x)} \exp\left(\sum_i \lambda_i f_i(x, y) \right) \tag{10}$$

  $$\Psi(\lambda) = -\sum_x \tilde{p}(x) \log Z_\lambda(x) + \sum_i \lambda_i \tilde{p}(f_i) \tag{11}$$

where $Z_\lambda(x)$ is a normalizing constant determined by the requirement that $\sum_y p_\lambda(y|x) = 1$ for all x:

$$Z_\lambda(x) = \sum_y \exp\left(\sum_i \lambda_i f_i(x,y)\right) \tag{12}$$

- Finally, we pose the unconstrained *dual* optimization problem

$$\text{Find } \lambda^\star = \operatorname*{argmax}_\lambda \Psi(\lambda)$$

At first glance it is not clear what these machinations achieve. However, a fundamental principle in the theory of Lagrange multipliers, called generically the Kuhn-Tucker theorem, asserts that under suitable assumptions, the primal and dual problems are, in fact, closely related. This is the case in the present situation. Although a detailed account of this relationship is beyond the scope of this paper, it is easy to state the final result: Suppose that $\lambda^\star$ is the solution of the dual problem. Then $p_{\lambda^\star}$ is the solution of the primal problem; that is $p_{\lambda^\star} = p_\star$. In other words,

> The maximum entropy model subject to the constraints $\mathcal{C}$ has the parametric form[3] $p_{\lambda^\star}$ of (10), where the parameter values $\lambda^\star$ can be determined by maximizing the dual function $\Psi(\lambda)$.

The most important practical consequence of this result is that any algorithm for finding the maximum $\lambda^\star$ of $\Psi(\lambda)$ can be used to find the maximum $p_\star$ of $H(p)$ for $p \in \mathcal{C}$.

3.5 Relation to Maximum Likelihood

The log-likelihood $L_{\tilde{p}}(p)$ of the empirical distribution $\tilde{p}$ as predicted by a model p is defined by[4]

$$L_{\tilde{p}}(p) \equiv \log \prod_{x,y} p(y|x)^{\tilde{p}(x,y)} = \sum_{x,y} \tilde{p}(x,y) \log p(y|x) \tag{13}$$

It is easy to check that the dual function $\Psi(\lambda)$ of the previous section is, in fact, just the log-likelihood for the exponential model p_λ; that is

$$\Psi(\lambda) = L_{\tilde{p}}(p_\lambda) \tag{14}$$

With this interpretation, the result of the previous section can be rephrased as:

> The model $p_\star \in \mathcal{C}$ with maximum entropy is the model in the parametric family $p_\lambda(y|x)$ that maximizes the likelihood of the training sample $\tilde{p}$.

---

3 It might be that the dual function $\Psi(\lambda)$ does not achieve its maximum at any finite $\lambda^\star$. In this case, the maximum entropy model will not have the form p_λ for any λ. However, it will be the *limit* of models of this form, as indicated by the following result whose proof we omit:

> Suppose λ_n is any sequence such that $\Psi(\lambda_n)$ converges to the maximum of $\Psi(\lambda)$. Then p_{λ_n} converges to $p_\star$.

4 We will henceforth abbreviate $L_{\tilde{p}}(p)$ by $L(p)$ when the empirical distribution $\tilde{p}$ is clear from context.

	Primal	Dual
problem	$\text{argmax}_{p \in \mathcal{C}} H(p)$	$\text{argmax}_\lambda \Psi(\lambda)$
description	maximum entropy	maximum likelihood
type of search	constrained optimization	unconstrained optimization
search domain	$p \in \mathcal{C}$	real-valued vectors $\{\lambda_1, \lambda_2 \ldots\}$
solution	$p_\star$	$\lambda^\star$
Kuhn-Tucker theorem: $p_\star = p_{\lambda^\star}$		

Table 1
The duality of maximum entropy and maximum likelihood is an example of the more general phenomenon of duality in constrained optimization.

This result provides an added justification for the maximum entropy principle: if the notion of selecting a model $p_\star$ on the basis of maximum entropy isn't compelling enough, it so happens that this same $p_\star$ is also the model which, from among all models of the same parametric form (10), can best account for the training sample.

Table 1 summarizes the primal-dual framework we have established.

3.6 Computing the Parameters

For all but the most simple problems, the $\lambda^\star$ that maximize $\Psi(\lambda)$ cannot be found analytically. Instead, we must resort to numerical methods. From the perspective of numerical optimization, the function $\Psi(\lambda)$ is well behaved, since it is smooth and convex-$\cap$ in λ. Consequently, a variety of numerical methods can be used to calculate $\lambda^\star$. One simple method is coordinate-wise ascent, in which $\lambda^\star$ is computed by iteratively maximizing $\Psi(\lambda)$ one coordinate at a time. When applied to the maximum entropy problem, this technique yields the popular Brown algorithm (Brown 1959). Other general purpose methods that can be used to maximize $\Psi(\lambda)$ include gradient ascent and conjugate gradient.

An optimization method specifically tailored to the maximum entropy problem is the iterative scaling algorithm of Darroch and Ratcliff (Darroch and Ratliff 1972). We present here a version of this algorithm specifically designed for the problem at hand; a proof of the monotonicity and convergence of the algorithm is given in (Della Pietra *et al* 1995). The algorithm is applicable whenever the feature functions $f_i(x,y)$ are non-negative:

$$f_i(x,y) \geq 0 \qquad \text{for all } i,\ x, \text{ and } y \tag{15}$$

This is, of course, true for the binary-valued feature functions we are considering here. The algorithm generalizes the Darroch-Ratcliff procedure, which requires, in addition to the non-negativity, that the feature functions satisfy $\sum_i f_i(x,y) = 1$ for all x, y.

Algorithm 1 : Improved Iterative Scaling

`Input:` Feature functions $f_1, f_2, \ldots f_n$; empirical distribution $\tilde{p}(x,y)$
`Output:` Optimal parameter values $\lambda^\star{}_i$; optimal model $p_{\lambda^\star}$

1. Start with $\lambda_i = 0$ for all $i \in \{1, 2, \ldots, n\}$
2. Do for each $i \in \{1, 2, \ldots, n\}$:

   a. Let $\Delta\lambda_i$ be the solution to

$$\sum_{x,y} \tilde{p}(x) p(y|x) f_i(x,y) \exp\left(\Delta\lambda_i f^\#(x,y)\right) = \tilde{p}(f_i) \tag{16}$$

$$\text{where} \quad f^{\#}(x,y) \equiv \sum_{i=1}^{n} f_i(x,y) \tag{17}$$

b. Update the value of λ_i according to: $\lambda_i \leftarrow \lambda_i + \Delta\lambda_i$

3. Go to step 2 if not all the λ_i have converged

The key step in the algorithm is step (2a), the computation of the increments $\Delta\lambda_i$ that solve (16). If $f^{\#}(x,y)$ is constant ($f^{\#}(x,y) = M$ for all x, y, say) then $\Delta\lambda_i$ is given explicitly as

$$\Delta\lambda_i = \frac{1}{M} \log \frac{\tilde{p}(f_i)}{p_\lambda(f_i)}$$

If $f^{\#}(x,y)$ is not constant, then $\Delta\lambda_i$ must be computed numerically. A simple and effective way of doing this is by Newton's method. This method computes the solution $\alpha_\star$ of an equation $g(\alpha_\star) = 0$ iteratively by the recurrence

$$\alpha_{n+1} = \alpha_n - \frac{g(\alpha_n)}{g'(\alpha_n)} \tag{18}$$

with an appropriate choice for α_0 and suitable attention paid to the domain of g.

4. Feature Selection

Earlier we divided the statistical modeling problem into two steps: finding appropriate facts about the data; the second is to incorporate these facts into the model. Up to this point we have proceeded by assuming that the first task was somehow performed for us. Even in the simple example of Section 2, we did not explicitly state how we selected those particular constraints. That is, why is the fact that *dans* or *à* was chosen by the expert translator 50% of the time any more important than countless other facts contained in the data? In fact, the principle of maximum entropy does not directly concern itself with the issue of feature selection: it merely provides a recipe for combining constraints into a model. But the feature selection problem is critical, since the universe of possible constraints is typically in the thousands or even millions. In this section we introduce a method for automatically selecting the features to be included in a maximum entropy model, and then offer a series of refinements to ease the computational burden.

4.1 Motivation

We begin by specifying a large collection $\mathcal{F}$ of candidate features. We do not require *a priori* that these features are actually relevant or useful. Instead, we let the pool be as large as practically possible. Only a small subset of this collection of features will eventually be employed in our final model.

If we had a training sample of infinite size, we could determine the "true" expected value for a candidate feature $f \in \mathcal{F}$ simply by computing the fraction of events in the sample for which $f(x,y) = 1$. In real-life applications, however, we are provided with only a small sample of N events, which cannot be trusted to represent the process fully and accurately. Specifically, we cannot expect that for every feature $f \in \mathcal{F}$, the estimate of $\tilde{p}(f)$ we derive from this sample will be close to its value in the limit as n grows large. Employing a larger (or even just a different) sample of data from the same process might result in different estimates of $\tilde{p}(f)$ for many candidate features.

In short, we would like to include in the model only a subset $\mathcal{S}$ of the full set of candidate features $\mathcal{F}$. We will call $\mathcal{S}$ the set of *active* features. The choice of $\mathcal{S}$ must capture as much information about the random process as possible, yet only include features whose expected values can be reliably estimated.

To find $\mathcal{S}$, we adopt an incremental approach to feature selection, similar to the strategy used for growing decision trees (Bahl *et al* 1989). The idea is to build up $\mathcal{S}$ by successively adding features. The choice of feature to add at each step is determined by the training data. Let us denote the set of models determined by the feature set $\mathcal{S}$ as $\mathcal{C}(\mathcal{S})$. "Adding" a feature f is shorthand for requiring that the set of allowable models all satisfy the equality $\tilde{p}(f) = p(f)$. Only some members of $\mathcal{C}(\mathcal{S})$ will satisfy this equality; the ones that do we denote by $\mathcal{C}(\mathcal{S} \cup f)$.

Thus, each time a candidate feature is adjoined to $\mathcal{S}$, another linear constraint is imposed on the space $\mathcal{C}(\mathcal{S})$ of models allowed by the features in $\mathcal{S}$. As a result, $\mathcal{C}(\mathcal{S})$ shrinks; the model $p_\star$ in $\mathcal{C}$ with the greatest entropy reflects ever-increasing knowledge and thus, hopefully, becomes a more accurate representation of the process. This narrowing of the space of permissible models was represented in figure 1 by a series of intersecting lines (hyperplanes, in general) in a probability simplex. Perhaps more intuitively, we could represent it by a series of nested subsets of $\mathcal{P}$, as in figure 2.

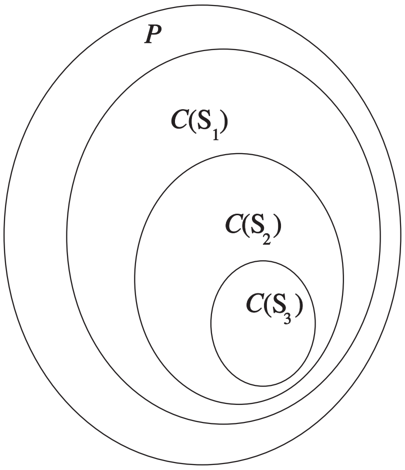

Figure 2
A nested sequence of subsets $\mathcal{C}(\mathcal{S}_1) \supset \mathcal{C}(\mathcal{S}_2) \supset \mathcal{C}(\mathcal{S}_3) \ldots$ of $\mathcal{P}$ corresponding to increasingly large sets of features $\mathcal{S}_1 \subset \mathcal{S}_2 \subset \mathcal{S}_3 \ldots$

4.2 Basic Feature Selection

The basic incremental growth procedure may be outlined as follows. Every stage of the process is characterized by a set of active features $\mathcal{S}$. These determine a space of models

$$\mathcal{C}(\mathcal{S}) \equiv \{p \in \mathcal{P} \mid p(f) = \tilde{p}(f) \quad \text{for all } f \in \mathcal{S}\} \tag{19}$$

The optimal model in this space, denoted by $p_{\mathcal{S}}$, is the model with the greatest entropy:

$$p_{\mathcal{S}} \equiv \operatorname*{argmax}_{p \in \mathcal{C}(\mathcal{S})} H(p) \tag{20}$$

By adding feature $\hat{f}$ to $\mathcal{S}$, we obtain a new set of active features $\mathcal{S} \cup \hat{f}$. Following (19), this set of features determines a set of models

$$\mathcal{C}(\mathcal{S} \cup \hat{f}) \equiv \{p \in \mathcal{P} \mid p(f) = \tilde{p}(f) \quad \text{for all } f \in \mathcal{S} \cup \hat{f}\} \tag{21}$$

The optimal model in this space of models is

$$p_{\mathcal{S}\cup\hat{f}} \equiv \operatorname*{argmax}_{p\in\mathcal{C}(\mathcal{S}\cup\hat{f})} H(p) \tag{22}$$

Adding the feature $\hat{f}$ allows the model $p_{\mathcal{S}\cup\hat{f}}$ to better account for the training sample; this results in a *gain* $\Delta L(\mathcal{S},\hat{f})$ in the log-likelihood of the training data

$$\Delta L(\mathcal{S},\hat{f}) \equiv L(p_{\mathcal{S}\cup\hat{f}}) - L(p_{\mathcal{S}}) \tag{23}$$

At each stage of the model-construction process, our goal is to select the candidate feature $\hat{f}\in\mathcal{F}$ which maximizes the gain $\Delta L(\mathcal{S},\hat{f})$; that is, we select the candidate feature which, when adjoined to the set of active features $\mathcal{S}$, produces the greatest increase in likelihood of the training sample. This strategy is implemented in

Algorithm 2: Basic Feature Selection

`Input:` Collection $\mathcal{F}$ of candidate features; empirical distribution $\tilde{p}(x,y)$
`Output:` Set $\mathcal{S}$ of active features; model $p_{\mathcal{S}}$ incorporating these features

1. Start with $\mathcal{S}=\emptyset$; thus $p_{\mathcal{S}}$ is uniform
2. Do for each candidate feature $f\in\mathcal{F}$:

   Compute the model $p_{\mathcal{S}\cup f}$ using Algorithm 1
   Compute the gain in the log-likelihood from adding this feature using (23)
3. Check the termination condition
4. Select the feature $\hat{f}$ with maximal gain $\Delta L(\mathcal{S},\hat{f})$
5. Adjoin $\hat{f}$ to $\mathcal{S}$
6. Compute $p_{\mathcal{S}}$ using Algorithm 1
7. Go to step 2

One issue left unaddressed by this algorithm is the termination condition. Obviously, we would like a condition which applies exactly when all the "useful" features have been selected. One reasonable stopping criterion is to subject each proposed feature to cross-validation on a held-out sample of data. If the feature does not lead to an increase in likelihood of the held-out sample of data, the feature is discarded. We will have more to say about the stopping criterion in Section 5.3.

4.3 Approximate Gains

Algorithm 2 is not a practical method for incremental feature selection. For each candidate feature $f\in\mathcal{F}$ considered in step 2, we must compute the maximum entropy model $p_{\mathcal{S}\cup f}$, a task that is computationally costly even with the efficient iterative scaling algorithm introduced earlier. We therefore introduce a modification to the algorithm, making it greedy but much more feasible. We replace the computation of the gain $\Delta L(\mathcal{S},f)$ of a feature f with an approximation, which we will denote by $\sim\!\Delta L(\mathcal{S},f)$.

Recall that a model $p_{\mathcal{S}}$ has a set of parameters λ, one for each feature in $\mathcal{S}$. The model $p_{\mathcal{S}\cup f}$ contains this set of parameters, plus a single new parameter α, corresponding to f.[5]

5 Another way to think of this is that the models $p_{\mathcal{S}\cup f}$ and $p_{\mathcal{S}}$ have the same number of parameters, but $\alpha=0$ for $p_{\mathcal{S}}$.

Given this structure, we might hope that the optimal values for λ do not change as the feature f is adjoined to $\mathcal{S}$. Were this the case, imposing an additional constraint would require only optimizing the single parameter α to maximize the likelihood. Unfortunately, when a new constraint is imposed, the optimal values of *all* parameters change.

However, to make the feature-ranking computation tractable, we make the approximation that the addition of a feature f affects only α, leaving the λ-values associated with other features unchanged. That is, when determining the gain of f over the model $p_{\mathcal{S}}$, we pretend that the best model containing features $\mathcal{S} \cup f$ has the form

$$p^{\alpha}_{\mathcal{S},f} = \frac{1}{Z_{\alpha}(x)} p_{\mathcal{S}}(y|x) e^{\alpha f(x,y)}, \quad \text{for some real valued } \alpha \tag{24}$$

$$\text{where } Z_{\alpha}(x) = \sum_y p_{\mathcal{S}}(y|x) e^{\alpha f(x,y)} \tag{25}$$

The only parameter which distinguishes models of the form (24) is α. Among these models, we are interested in the one which maximizes the *approximate gain*

$$\begin{aligned} G_{\mathcal{S},f}(\alpha) &\equiv L(p^{\alpha}_{\mathcal{S},f}) - L(p_{\mathcal{S}}) \\ &= -\sum_x \tilde{p}(x) \log Z_{\alpha}(x) + \alpha \tilde{p}(f) \end{aligned} \tag{26}$$

We will denote the gain of this model by

$$\sim\!\Delta L(\mathcal{S}, f) \equiv \max_{\alpha} G_{\mathcal{S},f}(\alpha) \tag{27}$$

and the optimal model by

$$\sim\! p_{\mathcal{S}\cup f} \equiv \operatorname*{argmax}_{p^{\alpha}_{\mathcal{S},f}} G_{\mathcal{S},f}(\alpha) \tag{28}$$

Despite the rather unwieldy notation, the idea is simple. Computing the approximate gain in likelihood from adding feature f to $p_{\mathcal{S}}$ has been reduced to a simple one-dimensional optimization problem over the single parameter α, which can be solved by any popular line-search technique such as Newton's method. This yields a great savings in computational complexity over computing the exact gain, an n-dimensional optimization problem requiring more sophisticated methods such as conjugate gradient. But the savings comes at a price: for any particular feature f, we are probably underestimating its gain, and there is a reasonable chance that we will select a feature f whose approximate gain $\sim\!\Delta L(\mathcal{S}, f)$ was highest and pass over the feature $\hat{f}$ with maximal gain $\Delta L(\mathcal{S}, \hat{f})$.

A graphical representation of this approximation is provided in figure 3. Here the log-likelihood is represented as an arbitrary convex function over two parameters: λ corresponds to the "old" parameter, and α to the "new" parameter. Holding λ fixed and adjusting α to maximize the log-likelihood involves a search over the darkened line, rather than a search over the entire space of (λ, α).

The actual algorithms, along with the appropriate mathematical framework, are presented in the appendix.

5. Case Studies

In the next few pages we discuss several applications of maximum entropy modeling within *Candide*, a fully automatic French-to-English machine translation system under

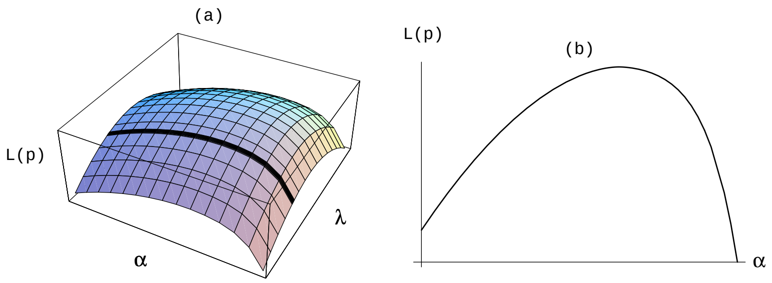

Figure 3
The likelihood $L(p)$ is a convex function of its parameters. If we start from a one-constraint model whose optimal parameter value is $\lambda = \lambda_0$ and consider the increase in $L_{\tilde{p}}(p)$ from adjoining a second constraint with the parameter α, the exact answer requires a search over (λ, α). We can simplify this task by holding $\lambda = \lambda_0$ constant and performing a line search over the possible values of the new parameter α. In (a), the darkened line represents the search space we restrict attention to. In (b) we show the reduced problem: a line search over α.

development at IBM. Over the past few years, we have used Candide as a test bed for exploring the efficacy of various techniques in modeling problems arising in machine translation.

We begin in Section 5.1 with a review of the general theory of statistical translation, describing in some detail the models employed in Candide. In Section 5.2 we describe how we have applied maximum entropy modeling to predict the French translation of an English word in context. In Section 5.3 we describe maximum entropy models that predict differences between French word order and English word order. In Section 5.4 we describe a maximum entropy model that predicts how to divide a French sentence into short segments that can be translated sequentially.

5.1 Review of Statistical Translation

When presented with a French sentence F, Candide's task is to find the English sentence $\hat{E}$ which is most likely given F:

$$\hat{E} = \operatorname*{argmax}_E p(E|F) \tag{29}$$

By Bayes' theorem, this is equivalent to finding $\hat{E}$ such that

$$\hat{E} = \operatorname*{argmax}_E p(F|E)p(E) \tag{30}$$

Candide estimates $p(E)$—the probability that a string E of English words is a well-formed English sentence—using a parametric model of the English language, commonly referred to as a *language model.* The system estimates $p(F|E)$—the probability that a French sentence F is a translation of E—using a parametric model of the process of English-to-French translation known as a *translation model.* These two models, plus a search strategy for finding the $\hat{E}$ which maximizes (30) for some F, comprise the engine of the translation system.

We now briefly describe the translation model for the probability $P(F|E)$; a more thorough account is provided in (Brown *et al* 1991). We imagine that an English sentence E generates a French sentence F in two steps. First each word in E independently generates zero or more French words. These words are then ordered to give a French sentence F. We denote the ith word of E by e_i and the jth word of F by y_j. (We employ y_j rather than the more intuitive f_j to avoid confusion with the feature function notation.) We denote the number of words in the sentence E by $|E|$ and the number of words in the sentence F by $|F|$. The generative process yields not only the French sentence F but also an association of the words of F with the words of E. We call this association an *alignment*, and denote it by A. An alignment A is parametrized by a sequence of $|F|$ numbers a_j, with $1 \leq a_i \leq |E|$. For every word position j in F, a_j is the word position in E of the English word that generates y_j. Figure 4 depicts a typical alignment.

The probability $p(F|E)$ that F is the translation of E is expressed as the sum over all possible alignments A between E and F of the probability of F and A given E:

$$p(F|E) = \sum_A p(F, A|E) \tag{31}$$

The sum in equation (31) is computationally unwieldy; it involves a sum over all $|E|^{|F|}$ possible alignments between the words in the two sentences. For this reason we sometimes

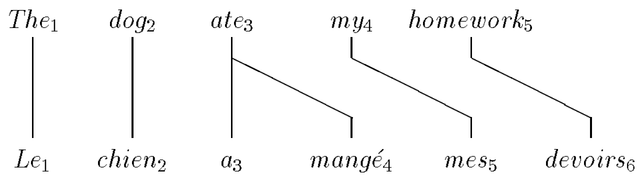

Figure 4
Alignment of a French–English sentence pair. The subscripts give the position of each word in its sentence. Here $a_1 = 1$, $a_2 = 2$, $a_3 = a_4 = 3$, $a_5 = 4$, and $a_6 = 5$.

make the simplifying assumption that there exists one extremely probable alignment $\hat{A}$, called the "Viterbi alignment," for which

$$p(F|E) \approx p(F, \hat{A}|E) \tag{32}$$

Given some alignment A (Viterbi or otherwise) between E and F, the probability $p(F, A|E)$ is given by

$$p(F, A|E) = \prod_{i=1}^{|E|} p(n(e_i)|e_i) \cdot \prod_{j=1}^{|F|} p(y_j|e_{a_j}) \cdot d(A|E, F) \tag{33}$$

where $n(e_i)$ denotes the number of French words aligned with e_i. In this expression

- $p(n|e)$ is the probability that the English word e generates n French words,
- $p(y|e)$ is the probability that the English word e generates the French word y; and
- $d(A|E, F)$ is the probability of the particular order of French words.

We call the model described by equations (31) and (33) the *basic translation model.*

We take the probabilities $p(n|e)$ and $p(y|e)$ as the fundamental parameters of the model, and parametrize the distortion probability in terms of simpler distributions. (Brown *et al* 1991) describe a method of estimating these parameters to maximize the likelihood of a large bilingual corpus of English and French sentences. Their method is based on the *Estimation-Maximization* (EM) algorithm, a well-known iterative technique for maximum likelihood training of a model involving hidden statistics. For the basic translation model, the hidden information is the alignment A between E and F.

We employed the EM algorithm to estimate the parameters of the basic translation model so as to maximize the likelihood of a bilingual corpus obtained from the proceedings of the Canadian parliament. For historical reasons, these proceedings are sometimes called "Hansards." Our Hansard corpus contains 3.6 million English-French sentence pairs for a total of a little under 100 million words in each language. Table 2 shows our parameter estimates for the translation probabilities $p(y|in)$. The basic translation model has worked admirably: given only the bilingual corpus, with no additional knowledge of the languages or any relation between them, it has uncovered some highly plausible translations.

Nevertheless, the basic translation model has one major shortcoming: it does not take the English context into account. That is, the model does not account for surrounding English words when predicting the appropriate French rendering of an English word. As we pointed out in Section 3, this is not how successful translation works. The best French translation of *in* is a function of the surrounding English words: if *a month's time* are

Translation	Probability
dans	0.3004
à	0.2275
de	0.1428
en	0.1361
pour	0.0349
(OTHER)	0.0290
au cours de	0.0233
,	0.0154
sur	0.0123
par	0.0101
pendant	0.0044

Table 2
Most frequent French translations of *in* as estimated using EM-training. (OTHER) represents a catch-all classifier for any French phrase not listed, none of which had a probability exceeding 0.0043.

Je dirais même que les chances sont supérieures à 50%.
$\downarrow$
I would even say that the odds are superior to 50%.

Il semble que Bank of Boston ait pratiquement achevé son réexamen de Shawmut.
$\downarrow$
He appears that Bank of Boston has almost completed its review of Shawmut.

Figure 5
Typical errors encountered in using EM-based model of Brown *et. al.* in a French-to-English translation system

the subsequent words, *pendant* might be more likely, but if *the fiscal year 1992* are what follows, then *dans* is more likely. The basic model is blind to context, always assigning a probability of 0.3004 to *dans* and 0.0044 to *pendant*.

This can yield errors when Candide is called upon to translate a French sentence. Examples of two such errors are shown in Figure 5. In the first example, the system has chosen an English sentence in which the French word *supérieures* has been rendered as *superior* when *greater* or *higher* is a preferable translation. With no knowledge of context, an expert translator is also quite likely to select *superior* as the English word which generates *supérieures*. But if the expert were privy to the fact that *50%* was among the next few words, he might be more inclined to select *greater* or *higher*. Similarly, in the second example, the incorrect rendering of *Il* as *He* might have been avoided had the translation model used the fact that the word following *it* is *appears*.

5.2 Context-Dependent Word Models

In the hope of rectifying these errors, we consider the problem of context-sensitive modeling of word translation. We envision, in practice, a separate maximum entropy model, $p_e(y|x)$, for each English word e, where $p_e(y|x)$ represents the probability that an expert translator would choose y as the French rendering of e, given the surrounding English context x. This is just a slightly recast version of a longstanding problem in computational linguistics, namely sense disambiguation—the determination of a word's sense from its context.

We begin with a training sample of English-French sentence pairs (E, F) randomly extracted from the Hansard corpus, such that E contains the English word *in*. For each sentence pair, we use the basic translation model to compute the Viterbi alignment $\hat{A}$ between E and F. Using this alignment, we then construct an (x, y) training event. The event consists of a context x containing the six words in E surrounding *in* and a future y equal to the French word which is (according to the Viterbi alignment $\hat{A}$) aligned with *in*. A few actual examples of such events for *in* are depicted in Table 3.

Next we define the set of candidate features. For this application, we employ features that are indicator functions of simply described sets. Specifically, we consider functions $f(x, y)$ which are one if y is some particular French word and the context x contains a given English word, and are zero otherwise. We employ the following notation to represent these features:

translation	e_{-3}	e_{-2}	e_{-1}		e_{+1}	e_{+2}	e_{+3}
dans	*the*	*committee*	*stated*	·	*a*	*letter*	*to*
à	*work*	*was*	*required*	·	*respect*	*of*	*the*
au cours de				·	*the*	*fiscal*	*year*
dans	*by*	*the*	*government*	·	*the*	*same*	*postal*
à	*of*	*diphtheria*	*reported*	·	*Canada*	*,*	*by*
de	*not*	*given*	*notice*	·	*the*	*ordinary*	*way*

Table 3
Several actual training events for the maximum entropy translation model for *in*, extracted from the transcribed proceedings of the Canadian parliament.

Template	Number of actual features	$f(x,y)=1$ if and only if ...
1	$\lvert\mathcal{V}_{\mathcal{F}}\rvert$	$y=\diamond$
2	$\lvert\mathcal{V}_{\mathcal{F}}\rvert\cdot\lvert\mathcal{V}_{\mathcal{E}}\rvert$	$y=\diamond$ and $\square \in \boxed{}\boxed{}\boxed{}\boxed{\bullet}\boxed{}\boxed{}$
3	$\lvert\mathcal{V}_{\mathcal{F}}\rvert\cdot\lvert\mathcal{V}_{\mathcal{E}}\rvert$	$y=\diamond$ and $\square \in \boxed{}\boxed{}\boxed{\bullet}\boxed{}\boxed{}\boxed{}$
4	$\lvert\mathcal{V}_{\mathcal{F}}\rvert\cdot\lvert\mathcal{V}_{\mathcal{E}}\rvert$	$y=\diamond$ and $\square \in \boxed{\bullet}\boxed{\bullet}\boxed{\bullet}\boxed{}\boxed{}\boxed{}$
5	$\lvert\mathcal{V}_{\mathcal{F}}\rvert\cdot\lvert\mathcal{V}_{\mathcal{E}}\rvert$	$y=\diamond$ and $\square \in \boxed{}\boxed{}\boxed{}\boxed{\bullet}\boxed{\bullet}\boxed{\bullet}$

Table 4
Feature templates for word-translation modeling. $|\mathcal{V}_{\mathcal{E}}|$ is the size of the English vocabulary; $|\mathcal{V}_{\mathcal{F}}|$ the size of the French vocabulary.

$$f_1(x,y) = \begin{cases} 1 & y = en \text{ and } April \in \boxed{}\boxed{}\boxed{}\boxed{\bullet}\boxed{}\boxed{} \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(x,y) = \begin{cases} 1 & y = pendant \text{ and } weeks \in \boxed{}\boxed{}\boxed{}\boxed{\bullet}\boxed{\bullet}\boxed{\bullet} \\ 0 & \text{otherwise} \end{cases}$$

Here $f_1 = 1$ when *April* follows *in* and *en* is the translation of *in*; $f_2 = 1$ when *weeks* is one of the three words following *in* and *pendant* is the translation.

The set of features under consideration is vast, but may be expressed in abbreviated form in Table 4. In the table, the symbol $\diamond$ is a placeholder for a possible French word and the symbol $\square$ is placeholder for a possible English word. The feature f_1 mentioned above is thus derived from template 2 with $\diamond = en$ and $\square = April$; the feature f_2 is derived from template 5 with $\diamond = pendant$ and $\square = weeks$. If there are $|\mathcal{V}_{\mathcal{E}}|$ total English words and $|\mathcal{V}_{\mathcal{F}}|$ total French words, there are $|\mathcal{V}_{\mathcal{F}}|$ template-1 features, and $|\mathcal{V}_{\mathcal{E}}| \cdot |\mathcal{V}_{\mathcal{F}}|$ features of templates 2,3,4 and 5.

Template 1 features give rise to constraints that enforce equality between the probability of any French translation y of *in* according to the model and the probability of that translation in the empirical distribution. Examples of such constraints are

$$\begin{aligned} p(y = dans) &= \tilde{p}(y = dans) \\ p(y = \grave{a}) &= \tilde{p}(y = \grave{a}) \\ p(y = de) &= \tilde{p}(y = de) \\ p(y = en) &= \tilde{p}(y = en) \\ &\vdots \end{aligned}$$

A maximum entropy model that uses only template 1 features predicts each French translation y with the probability $\tilde{p}(y)$ determined by the empirical data. This is exactly the distribution employed by the basic translation model.

Since template 1 features are independent of x, the maximum entropy model which employs only constraints derived from template 1 features takes no account of contextual information in assigning a probability to y. When we include constraints derived from template 2 features, we take our first step towards a context-dependent model. Rather than simply constraining the expected probability of a French word y to equal its empirical probability, these constraints require that the expected *joint* probability of the English word immediately following *in* and the French rendering of *in* be equal to its empirical probability. An example of a template 2 constraint is

$$p(y = pendant, \quad e_{+1} = several) = \tilde{p}(y = pendant, \quad e_{+1} = several)$$

A maximum entropy model that incorporates this constraint will predict the translations of *in* in a manner consistent with whether or not the following word is *several*. In particular, if in the empirical sample, the presence of *several* led to a greater probability for *pendant*, this will be reflected in a maximum entropy model incorporating this constraint. We have thus taken our first step toward context-sensitive translation modeling.

Templates 3, 4 and 5 consider, each in a different way, various parts of the context. For instance, template 5 constraints allow us to model how an expert translator is biased by the appearance of a word *somewhere* in the three words following the word he is translating. If *house* appears within the next three words (e.g. the phrases *in the house* and *in the red house*), then *dans* might be a more likely translation. On the other hand, if *year* appears within the same window of words (as in *in the year 1941* or *in that fateful year*), then *au cours de* might be more likely. Together, the five constraint templates allow the model to condition its assignment of probabilities on a window of six words around e_0, the word in question.

We constructed a maximum entropy model $p_{in}(y|x)$ by the iterative model-growing method described in Section 4. The automatic feature selection algorithm first selected a template 1 constraint for each of the translations of *in* seen in the sample (12 in all), thus constraining the model's expected probability of each of these translations to their empirical probabilities. The next few constraints selected by the algorithm are shown in Table 5. The first column gives the identity of the feature whose expected value is constrained; the second column gives $\sim\Delta L(\mathcal{S}, f)$, the approximate increase in the model's log-likelihood on the data as a result of imposing this constraint; the third column gives $L(p)$, the log-likelihood after adjoining the feature and recomputing the model.

Let us consider the fifth row in the table. This constraint requires that the model's expected probability of *dans*, if one of the three words to the right of *in* is the word *speech*, is equal to that in the empirical sample. Before imposing this constraint on the model during the iterative model-growing process, the log-likelihood of the current model on the empirical sample was -2.8703 bits. The feature selection algorithm described in Section 4 calculated that if this constraint were imposed on the model, the log-likelihood would rise by approximately 0.019059 bits; since this value was higher than for any other constraint considered, the constraint was selected. After applying iterative scaling to recompute the parameters of the new model, the likelihood of the empirical sample rose to -2.8525 bits, an increase of 0.0178 bits.

Table 6 lists the first few selected features for the model for translating the English word *run*. The "Hansard flavor"—the rather specific domain of parliamentary discourse related to Canadian affairs—is easy to detect in many of the features in this Table 5.

It is not hard to incorporate the maximum entropy word translation models into a translation model $p(F|E)$ for a French sentence given an English sentence. We merely

Feature $f(x, y)$							$\sim\Delta L(\mathcal{S}, f)$	$L(p)$
y=*à* and *Canada* $\in$				•			0.0415	−2.9674
y=*à* and *House* $\in$				•			0.0361	−2.9281
y=*en* and *the* $\in$				•			0.0221	−2.8944
y=*pour* and *order* $\in$				•			0.0224	−2.8703
y=*dans* and *speech* $\in$				•	•	•	0.0190	−2.8525
y=*dans* and *area* $\in$				•	•	•	0.0153	−2.8377
y=*de* and *increase* $\in$	•	•	•				0.0151	−2.8209
y=[*verb marker*] and *my* $\in$				•			0.0141	−2.8034
y=*dans* and *case* $\in$				•	•	•	0.0116	−2.7918
y=*au cours de* and *year* $\in$				•	•	•	0.0104	−2.7792

Table 5
Maximum entropy model to predict French translation of *in*. Features shown here were the first non template 1 features selected. [*verb marker*] denotes a morphological marker inserted to indicate the presence of a verb as the next word.

Feature $f(x,y)$							$\sim\Delta L(\mathcal{S},f)$	$L(p)$
y=*épuiser* and *out* $\in$				•	•	•	0.0252	-4.8499
y=*manquer* and *out* $\in$				•	•	•	0.0221	-4.8201
y=*écouler* and *time* $\in$				•	•	•	0.0157	-4.7969
y=*accumuler* and *up* $\in$				•			0.0149	-4.7771
y=*nous* and *we* $\in$			•				0.0140	-4.7582
y=*aller* and *counter* $\in$				•	•	•	0.0131	-4.7445
y=*candidat* and *for* $\in$				•	•	•	0.0124	-4.7295
y=*diriger* and *the* $\in$				•	•	•	0.0123	-4.7146

Table 6
Maximum entropy model to predict French translation of *to run*: top-ranked features not from template 1

Je dirais même que les chances sont supérieures à 50%.

↓

I would even say that the odds are greater than 50%.

Il semble que Bank of Boston ait pratiquement achevé son réexamen de Shawmut.

↓

It appears that Bank of Boston has almost completed its review of Shawmut .

Figure 6
Improved French-to-English translations resulting from maximum entropy-based system

replace the simple context-independent models $p(y|e)$ used in the basic translation model (33) with the more general context-dependent models $p_e(y|x)$:

$$p(F, A|E) = \prod_{i=1}^{|E|} p(n(e_i)|e_i) \cdot \prod_{j=1}^{|F|} p_{e_{a_j}}(y_j|x_{a_j}) \cdot d(A|E, F)$$

where x_{a_j} denotes the context of the English word e_{a_j}.

Figure 6 illustrates how using this improved translation model in the Candide system led to improved translations for the two sample sentences given earlier.

5.3 Segmentation

Though an ideal machine translation system could devour input sentences of unrestricted length, a typical stochastic system must cut the French sentence into polite lengths before digesting them. If the processing time is exponential in the length of the input passage (as is the case with the Candide system), then not splitting the French sentence into reasonably-sized segments would result in an exponential slowdown in translation.

Thus, a common task in machine translation is to find safe positions at which to split input sentences in order to speed the translation process. "Safe" is a vague term; one might, for instance, reasonably define a safe segmentation as one which results in coherent blocks of words. For our purposes, however, a safe segmentation is dependent on the Viterbi alignment $\hat{A}$ between the input French sentence F and its English translation E.

We define a *rift* as a position j in F such that for all $k < j$, $a_k \leq a_j$ and for all $k > j$, $a_k \geq a_j$. In other words, the words to the left of the French word y_j are generated by words to the left of the English word e_{a_j}, and the words to the right of y_j are generated by words to the right of e_{a_j}. In the alignment of figure 4, for example, there are rifts at positions $j = 1, 2, 4, 5$ in the French sentence. One visual method of determining whether a rift occurs after the French word j is to try to trace a line from the last letter of y_j up to the last letter of e_{a_j}; if the line can be drawn without intersecting any alignment lines, position f is a rift.

Using our definition of rifts, we can redefine a "safe" segmentation as one in which the segment boundaries are located only at rifts. Figure 7 illustrates an unsafe segmentation, in which a segment boundary (denoted by the || symbol) lies between *a* and *mangé*, where there is no rift. Figure 8, on the other hand, illustrates a safe segmentation.

The reader will notice that a safe segmentation does not necessarily result in semantically coherent segments: *mes* and *devoirs* are certainly part of one logical unit,

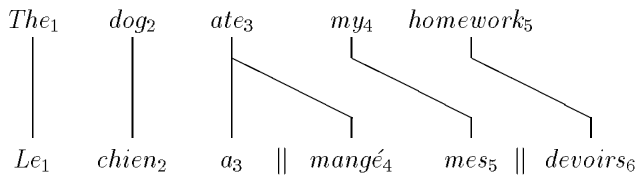

Figure 7
Example of an unsafe segmentation. A word in the translated sentence (e_3) is aligned to words (y_3 and y_4) in two different segments of the input sentence.

yet are separated in this safe segmentation. Once such a safe segmentation has been applied to the French sentence, we can make the assumption while searching for the appropriate English translation that no word in the translated English sentence will have to account for French words located in multiple segments. Disallowing intersegment alignments dramatically reduces the scale of the computation involved in generating a translation, particularly for large sentences. We can consider each segment sequentially while generating the translation, working from left to right in the French sentence.

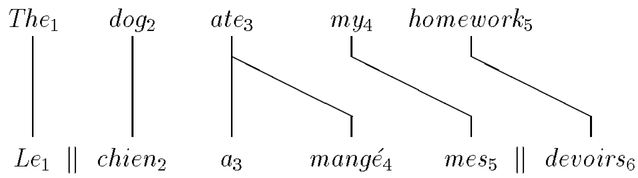

Figure 8
Example of a safe segmentation

We now describe a maximum entropy model which assigns to each location in a French sentence a score which is a measure of the safety in cutting the sentence at that location. We begin as in the word translation problem, with a training sample of English-French sentence pairs (E, F) randomly extracted from the Hansard corpus. For each sentence pair we use the basic translation model to compute the Viterbi alignment $\hat{A}$ between E and F. We also use a stochastic part of speech tagger as described in (Merialdo 1990) to label each word in F with its part of speech. For each position j in F we then construct a (x, y) training event. The value y is `rift` if a rift belongs at position j and is `no-rift` otherwise. The context information x is reminiscent of that employed in the word translation application described earlier. It includes a six-word window of French words: three to the left of y_j and three to the right of y_j. It also includes the part-of-speech tags for these words, and the classes of these words as derived from a mutual-information clustering scheme described in (Brown *et al* 1990). The complete (x, y) pair is illustrated in Figure 9.

In creating $p(\texttt{rift}|x)$, we are (at least in principle) modeling the decisions of an expert French segmenter. We have a sample of his work in the training sample $\tilde{p}(x, y)$,

`rift?`	e_{a_i-3}	...	e_{a_i+3}	tag(e_{a_i-3})	...	tag(e_{a_i+3})	class(e_{a_i-3})	...	class(e_{a_i+3})
y	x								

Figure 9
(x, y) for sentence segmentation

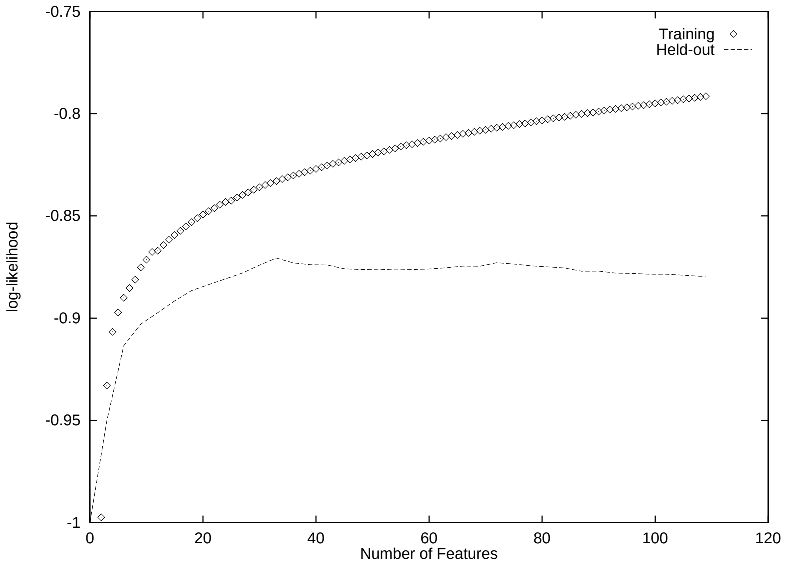

Figure 10
Change in log-likelihood during segmenting model-growing. (Overtraining begins to occur at about 40 features)

and we measure the worth of a model by the log-likelihood $L_{\tilde{p}}(p)$. During the iterative model-growing procedure, the algorithm selects constraints on the basis of how much they increase this objective function. As the algorithm proceeds, more and more constraints are imposed on the model p, bringing it into ever-stricter compliance with the empirical data $\tilde{p}(x,y)$. This is useful to a point; insofar as the empirical data embodies the expert knowledge of the French segmenter, we would like to incorporate this knowledge into a model. But the data contains only so much expert knowledge; the algorithm should terminate when it has extracted this knowledge. Otherwise, the model $p(y|x)$ will begin to fit itself to quirks in the empirical data.

A standard approach in statistical modeling to avoid the problem of overfitting the training data is employ cross-validation techniques. Separate the training data $\tilde{p}(x,y)$ into a training portion, $\tilde{p}_r$, and a heldout portion, $\tilde{p}_h$. Use only $\tilde{p}_r$ in the model-growing process; that is, select features based on how much they increase the likelihood $L_{\tilde{p}_r}(p)$. As the algorithm progresses, $L_{\tilde{p}_r}(p)$ thus increases monotonically. As long as each new constraint imposed allows p to better account for the random process which generated both $\tilde{p}_r$ and $\tilde{p}_h$, the quantity $L_{\tilde{p}_h}(p)$ also increases. At the point when overfitting begins, however, the new constraints no longer help p model the random process, but instead require p to model the noise in the sample $\tilde{p}_r$ itself. At this point, $L_{\tilde{p}_r}(p)$ continues to rise, but $L_{\tilde{p}_h}(p)$ no longer does. It is at this point that the algorithm should terminate.

Figure 10 illustrates the change in log-likelihood of training data $L_{\tilde{p}_r}(p)$ and held-out data $L_{\tilde{p}_h}(p)$. Had the algorithm terminated when the log-likelihood of the held-out data stopped increasing, the final model p would contain slightly less than 40 features.

We have employed this segmenting model as a component in a French-English machine translation system in the following manner. The model assigns to each position in the French sentence a score, $p(\texttt{rift} \mid x)$, which is a measure of how appropriate a split would be at that location. A dynamic programming algorithm then selects, given the "appropriateness" score at each position and the requirement that no segment may contain more than 10 words, an optimal (or, at least, reasonable) splitting of the sentence.

Monsieur l'Orateur

,

j'aimerais poser une question au
Ministre des Transports.

—–

A quelle date le
nouveau règlement devrait il entrer en vigeur?

—–

Quels furent les critères utilisés
pour l'évaluation
de ces biens.

—–

Nous
savons que si nous pouvions contrôler la folle avoine
dans l'ouest du Canada, en
un an nous
augmenterions notre rendement en
céréales de 1 milliard de dollars.

Figure 11
Maximum entropy segmenter behavior on four sentences selected at random from the Hansard data

Figure 11 shows the system's segmentation of four sentences selected at random from the Hansard data. We remind the reader to keep in mind when evaluating Figure 11 that the segmenter's task is not to produce logically coherent blocks of words, but to divide the sentence into blocks which can be translated sequentially from left to right.

5.4 Word Reordering

Translating a French sentence into English involves not only selecting appropriate English renderings of the words in the French sentence, but also selecting an ordering for the English words. This order is often very different from the French word order. One way Candide captures word-order differences in the two languages is to allow for alignments with crossing lines. In addition, Candide performs, during a pre-processing stage, a reordering step which shuffles the words in the input French sentence into an order more closely resembling English word order.

One component of this word reordering step deals with French phrases which have the NOUN *de* NOUN form. For some NOUN *de* NOUN phrases, the best English translation is nearly word for word: *conflit d'intérêt*, for example, is almost always rendered as *conflict of interest*. For other phrases, however, the best translation is obtained by interchanging the two nouns and dropping the *de*. The French phrase *taux d'intérêt*, for example, is best rendered as *interest rate*. Table 7 gives several examples of NOUN *de* NOUN phrases together with their most appropriate English translations.

In this section we describe a maximum entropy model which, given a French NOUN *de* NOUN phrase, estimates the probability that the best English translation involves an interchange of the two nouns. We begin with a sample of English-French sentence pairs (E, F) randomly extracted from the Hansard corpus, such that F contains a *de* phrase. For each sentence pair we use the basic translation model to compute the Viterbi alignment $\hat{A}$ between the words in E and F. Using $\hat{A}$ we construct an (x, y) training event as follows. We let the context x be the pair of French nouns $(\text{NOUN}_L, \text{NOUN}_R)$. We let y be

word-for-word phrases	
somme d'argent	*sum of money*
pays d'origin	*country of origin*
question de privilège	*question of privilege*
conflit d'intérêt	*conflict of interest*
interchanged phrases	
bureau de poste	*post office*
taux d'intérêt	*interest rate*
compagnie d'assurance	*insurance company*
gardien de prison	*prison guard*

Table 7
NOUN *de* NOUN phrases and their English equivalents

Template	Number of actual features	$f(x,y)=1$ if and only if ...		
1	$2\lvert\mathcal{V}_{\mathcal{F}}\rvert$	$y=\Diamond$	and	$\text{NOUN}_L = \Box$
2	$2\lvert\mathcal{V}_{\mathcal{F}}\rvert$	$y=\Diamond$	and	$\text{NOUN}_R = \Box$
3	$2\lvert\mathcal{V}_{\mathcal{F}}\rvert^2$	$y=\Diamond$	and	$\text{NOUN}_L = \Box_1$ and $\text{NOUN}_R = \Box_2$

Table 8
Template features for NOUN *de* NOUN model

`no-interchange` if the English translation is a word-for-word translation of the French phrase and $y =$ `interchange` if the order of the nouns in the English and French phrases are interchanged.

We define candidate features based upon the *template features* shown in Table 8. In this table, the symbol $\Diamond$ is a placeholder for either `interchange` or `no-interchange` and the symbols $\Box_1$ and $\Box_2$ are placeholders for possible French words. If there are $|\mathcal{V}_{\mathcal{F}}|$ total French words, there are $2|\mathcal{V}_{\mathcal{F}}|$ possible features of templates 1 and 2 and $2|\mathcal{V}_{\mathcal{F}}|^2$ features of template 3.

Template 1 features consider only the left noun. We expect these features to be relevant when the decision of whether to interchange the nouns is influenced by the identity of the left noun. For example, including the template 1 feature

$$f(x,y) = \begin{cases} 1 & y=\texttt{interchange} \text{ and } \text{NOUN}_L = \textit{système} \\ 0 & \text{otherwise} \end{cases}$$

gives the model sensitivity to the fact that the nouns in French NOUN *de* NOUN phrases which begin with *système* (such as *système de surveillance* and *système de quota*) are more likely to be interchanged in the English translation. Similarly, including the template 1 feature

$$f(x,y) = \begin{cases} 1 & y=\texttt{no-interchange} \text{ and } \text{NOUN}_L = \textit{mois} \\ 0 & \text{otherwise} \end{cases}$$

gives the model sensitivity to the fact that French NOUN *de* NOUN phrases which begin with *mois*, such as *mois de mai* (month of May) are more likely to be translated word for word.

Template 3 features are useful in dealing with translating NOUN *de* NOUN phrases in which the interchange decision is influenced by both nouns. For example, NOUN *de* NOUN phrases ending in *intérêt* are sometimes translated word for word, as in *conflit d'intérêt (conflict of interest)* and are sometimes interchanged, as in *taux d'intérêt (interest rate).*

We used the feature-selection algorithm of section 4 to construct a maximum entropy model from candidate features derived from templates 1,2 and 3. The model was grown on 10,000 training events randomly selected from the Hansard corpus. The final model contained 358 constraints.

To test the model, we constructed a NOUN *de* NOUN word-reordering module which interchanges the order of the nouns if $p(\texttt{interchange} \mid x) > 0.5$ and keeps the order the same otherwise. Table 9 compares performance on a suite of test data against a baseline NOUN *de* NOUN reordering module which never the swaps the word order.

Table 12 shows some randomly-chosen NOUN *de* NOUN phrases extracted from this test suite along with $p(\texttt{interchange}|x)$, the probability which the model assigned to inversion. On the right are phrases such as *saison d'hiver* for which the model strongly

Test data	Simple Model Accuracy	Maximum Entropy Model Accuracy
50,229 not interchanged	100%	93.5%
21,326 interchanged	0%	49.2%
71,555 total	70.2%	80.4%

Table 9
NOUN *de* NOUN model performance: simple approach vs. maximum entropy

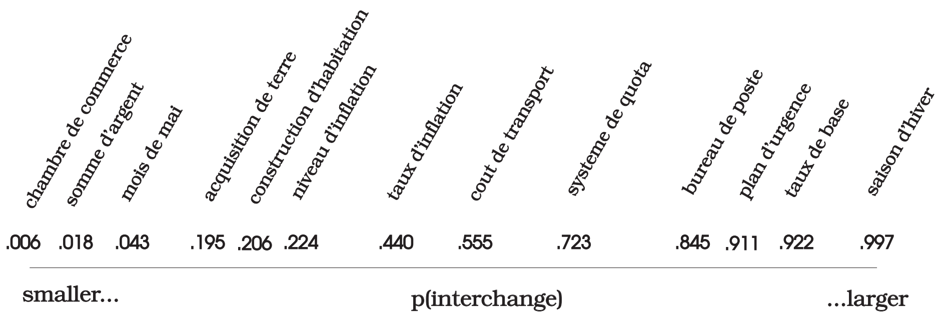

Figure 12
Predictions of the NOUN *de* NOUN interchange model on phrases selected from a corpus unseen during the training process

predicted an inversion. On the left are phrases which the model strongly prefers not to interchange, such as *somme d'argent*, *abus de privilège* and *chambre de commerce*. Perhaps most intriguing are those phrases which lie in the middle, such as *taux d'inflation*, which can translate either to *inflation rate* or *rate of inflation*.

6. Conclusion

We began by introducing the building blocks of maximum entropy modeling—real-valued features and constraints built from these features. We then discussed the maximum entropy principle. This principle instructs us to choose, among all the models consistent with the constraints, the model with the greatest entropy. We observed that this model was a member of an exponential family with one adjustable parameter for each constraint. The optimal values of these parameters are obtained by maximizing the likelihood of the training data. Thus two different philosophical approaches—maximum entropy and maximum likelihood—yield the same result: the model with the greatest entropy consistent with the constraints is the *same* as the exponential model which best predicts the sample of data.

We next discussed algorithms for constructing maximum entropy models, concentrating our attention on the two main problems facing would-be modelers: selecting a set of features to include in a model, and computing the parameters of a model which contains these features. The general feature-selection is too slow in practice, and we presented several techniques for making the algorithm feasible.

In the second part of this paper we described several applications of our algorithms, concerning modeling tasks arising in *Candide*, an automatic machine-translation system under development at IBM. These applications demonstrate the efficacy of maximum entropy techniques for performing context-sensitive modeling.

Acknowledgments

The authors wish to thank Harry Printz and John Lafferty for suggestions and comments on a preliminary draft of this paper, and Jerome Bellegarda for providing expert French knowledge.

References

Bahl, L., Brown, P., de Souza, P., Mercer, R. (1989) A tree-based statistical language model for natural language speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 7.

Berger, A., Brown, P., Della Pietra, S., Della Pietra, V., Gillett, J., Lafferty, J., Printz, H., Ureš, L. (1994) The Candide system for machine translation. *Proceedings of the ARPA Conference on Human Language Technology*, Plainsborough, New Jersey.

Black, E., Jelinek, F., Lafferty, J., Magerman, D., Mercer, R. and Roukos, S. (1992) Towards history-based grammars: using richer models for probabilistic parsing. *Proceedings of the DARPA Speech and Natural Language Workshop*, Arden House, New York.

Brown, D. (1959) A note on approximations to discrete probability distributions. *Information and Control*, vol. 2, 386–392.

Brown, P., Della Pietra, S., Della Pietra, V. Mercer, R. (1993) The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, vol. 19, no. 2, 263–311.

Brown, P., Cocke, J. Della Pietra, S., Della Pietra, V., Jelinek, F., Lafferty, J., Mercer, R., and Roossin, P. (1990) A statistical approach to machine translation. *Computational Linguistics*, vol. 16, 79–85.

Brown, P., Della Pietra, V, de Souza, P., Mercer, R. (1990) Class-based n-gram models of natural language. *Proceedings of the IBM Natural Language ITL*, 283–298.

Brown, P., Della Pietra, S., Della Pietra, V., Mercer, R. (1991) A statistical approach to sense disambiguation in machine translation. *DARPA Workshop on Speech and Natural Language*, 146–151.

Cover, T. and Thomas, J. (1991) *Elements of Information Theory.* John Wiley & Sons.

Csiszár, I. (1975) I-Divergence geometry of probability distributions and minimization problems, *The Annals of Probability*, vol. 3, No. 1, 146–158.

ibid. (1989) A geometric interpretation of Darroch and Ratcliff's generalized iterative scaling. *The Annals of Statistics*, vol. 17, No. 3, 1409–1413.

Csiszár, L. and Tusnády, G. (1984) Information geometry and alternating minimization procedures. *Statistics & Decisions, Supplemental Issue*, no. 1, 205–237.

Darroch, J.N. and Ratcliff, D. (1972) Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, no. 43, 1470–1480.

Della Pietra, S., Della Pietra, V., Gillett, J., Lafferty, J., Printz, H., Ureš, L. (1994) Inference and estimation of a long-range trigram model. *Second International Symposium on Grammatical Inference*, Alicante, Spain.

Della Pietra, S., Della Pietra, V., Lafferty, J. Inducing features of random fields, (1995) CMU Technical Report CMU-CS-95-144.

Dempster, A.P., Laird, N.M., and Rubin, D.B. (1977) Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, vol 39, no. B, 1–38.

Guiasu, S. and Shenitzer, A. (1985) The principle of maximum entropy. *The Mathematical Intelligencer*, vol. 7, no. 1.

Jaynes, E.T. (1990) Notes on present status and future prospects. In Grandy, W.T., and Schick, L.H. *Maximum Entropy and Bayesian Methods.* Kluwer. 1–13.

Jelinek, F. and Mercer, R. L. (1980) Interpolated estimation of Markov source parameters from sparse data. In *Proceedings, Workshop on Pattern Recognition in Practice*, Amsterdam, The Netherlands.

Lucassen, J. and Mercer, R. (1984) An information theoretic approach to automatic determination of phonemic baseforms. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, . San Diego, CA, 42.5.1–42.5.4.

Merialdo, B. (1990) Tagging text with a probabilistic model. *Proceedings of the IBM Natural Language ITL*, Paris, France, 161–172.

Nádas, A. Mercer, R., Bahl, L., Bakis, R., Cohen, P., Cole, A., Jelinek, F., and Lewis, B. (1981) Continuous speech recognition with automatically selected acoustic prototypes obtained by either bootstrapping or clustering. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Atlanta, GA, 1153–1155.

Sokolnikoff, I. S., Redheffer, R. M. (1966) *Mathematics of Physics and Modern Engineering*, Second Edition, McGraw-Hill Book Company.

Appendix: Efficient Algorithms for Feature Selection

Computing the Approximate Gain of One Feature

This section picks up where section 4 left off, describing in some detail a set of algorithms which implement the feature selection process efficiently.

We first describe an iterative algorithm for computing $\sim\Delta L(\mathcal{S}, f) \equiv \max_\alpha G_{\mathcal{S},f}(\alpha)$ for a candidate feature f. The algorithm is based on the fact that the maximum of $G_{\mathcal{S},f}(\alpha)$ occurs (except in rare cases) at the unique value $\alpha^\star$ at which the derivative $G'_{\mathcal{S},f}(\alpha^\star)$ is zero. To find this zero we apply Newton's iterative root-finding method. An important twist is that we do *not* use the updates obtained by applying Newton's method directly in the variable α. This is because there is no guarantee that $G_{\mathcal{S},f}(\alpha_n)$ increases monotonically for such updates. Instead, we use updates derived by applying Newton's method in the variables e^α or $e^{-\alpha}$. A convexity argument shows that using these updates the sequence of $G_{\mathcal{S},f}(\alpha_n)$ converges monotonically to the maximum approximate gain $\sim\Delta L(\mathcal{S}, f) \equiv G_{\mathcal{S},f}(\alpha^\star)$ and that α_n increases monotonically to $\alpha^\star$.

The value $\alpha^\star$ that maximizes $G_{\mathcal{S},f}(\alpha)$ can be found by solving the equation $G'_{\mathcal{S},f}(\alpha^\star) = 0$. Moreover, if α_n is any sequence for which $G'_{\mathcal{S},f}(\alpha_n)$ converges *monotonically* to 0, then

$G_{\mathcal{S},f}(\alpha_n)$ will increase monotonically. This is a consequence of the convexity of $G_{\mathcal{S},f}(\alpha)$ in α.

We can solve an equation $g(\alpha) = 0$ by Newton's method, which produces a sequence α_n by the recurrence given in (18), repeated here for convenience:

$$\alpha_{n+1} \quad = \quad \alpha_n - \frac{g(\alpha_n)}{g'(\alpha_n)} \tag{34}$$

If we start with α_0 sufficiently close to $\alpha_\star$, then the sequence α_n will converge to $\alpha_\star$ and $g(\alpha_n)$ will converge to zero. In general, though, the $g(\alpha_n)$ will not be monotonic. However, it can be shown that the sequence is monotonic in the following important cases: if $\alpha_0 \leq \alpha_\star$ and $g(\alpha)$ is either decreasing and convex-$\cup$ or increasing and convex-$\cap$.

The function $G'_{\mathcal{S},f}(\alpha)$ is neither convex-$\cap$ or convex-$\cup$ as a function of α. However, it can be shown (by taking derivatives) that $G'_{\mathcal{S},f}(\alpha)$ is decreasing and convex-$\cup$ in e^{α}, and is increasing and convex-$\cap$ in $e^{-\alpha}$. Thus, if $\alpha^\star > 0$ so that $e^0 < e^{\alpha^\star}$, we can apply Newton's method in e^{α} to obtain a sequence of α_n for which $G'_{\mathcal{S},f}(\alpha_n)$ increases monotonically to zero. Similarly, if $\alpha^\star < 0$ so that $e^0 < e^{-\alpha^\star}$, we can apply Newton's method in $e^{-\alpha}$ to obtain a sequence α_n for which $G'_{\mathcal{S},f}(\alpha_n)$ decreases monotonically to zero. In either case, $G_{\mathcal{S},f}(\alpha_n)$ increases monotonically to its maximum $G_{\mathcal{S},f}(\alpha^\star)$.

The updates resulting from Newton's method applied in the variable $e^{r\alpha}$, for $r = 1$ or $r = -1$ are easily computed:

$$\alpha_{n+1} = \alpha_n + \frac{1}{r} \log \left(1 - \frac{1}{r} \frac{G'_{\mathcal{S},f}(\alpha_n)}{G''_{\mathcal{S},f}(\alpha_n)} \right) \tag{35}$$

In order to solve the recurrence in (35), we need to compute $G'_{\mathcal{S},f}$ and $G''_{\mathcal{S},f}$. The zeroth, first and second derivatives of G are

$$G_{\mathcal{S},f}(\alpha) \quad = \quad -\sum_x \tilde{p}(x) \log Z_\alpha(x) + \alpha \tilde{p}(f) \tag{36}$$

$$G'_{\mathcal{S},f}(\alpha) \quad = \quad \tilde{p}(f) - \sum_x \tilde{p}(x) p^{\alpha}_{\mathcal{S},f}(f|x) \tag{37}$$

$$G''_{\mathcal{S},f}(\alpha) \quad = \quad -\sum_x \tilde{p}(x) p^{\alpha}_{\mathcal{S},f}((f - p^{\alpha}_{\mathcal{S},f}(f|x))^2|x) \tag{38}$$

$$\text{where } p^{\alpha}_{\mathcal{S},f}(h|x) \equiv \sum_y p^{\alpha}_{\mathcal{S},f}(y|x) h(x,y) \tag{39}$$

With these in place, we are ready to enumerate

Algorithm 3: Computing the Gain of a Single Feature

`Input:`	Empirical distribution $\tilde{p}(x,y)$; initial model $p_{\mathcal{S}}$; candidate feature f
`Output:`	Approximate gain $\sim\Delta L(\mathcal{S}, f)$ of feature f

1. Let

$$r = \begin{cases} 1 & \text{if } \tilde{p}(f) \leq p_{\mathcal{S}}(f) \\ -1 & \text{otherwise} \end{cases} \tag{40}$$

2. Set $\alpha_0 \leftarrow 0$

3. Repeat the following until $G_{\mathcal{S},f}(\alpha_n)$ has converged:

   Compute α_{n+1} from α_n using (35)
   Compute $G_{\mathcal{S},f}(\alpha_{n+1})$ using (26)

4. Set $\sim\Delta L(\mathcal{S}, f) \leftarrow G_{\mathcal{S},f}(\alpha_n)$

Computing Approximate Gains in Parallel

For the purpose of incremental model growing as outlined in Algorithm 2, we need to compute the maximum approximate gain $\sim\Delta L(\mathcal{S}, f)$ for *each* candidate feature $f \in \mathcal{F}$. One obvious approach is to cycle through all candidate features and apply Algorithm 3 for each one sequentially. Since Algorithm 3 requires one pass through every event in the training sample per iteration, this could entail millions of passes through the training sample. Because a significant cost often exists for reading the training data—if the data cannot be stored in memory but must be accessed from disk, for example—an algorithm which passes a minimal number of times through the data may be of some utility. We now give a parallel algorithm specifically tailored to this scenario.

Algorithm 4: Computing Approximate Gains for A Collection of Features

`Input:` Collection $\mathcal{F}$ of candidate features; empirical distribution $\tilde{p}(x, y)$; initial model $p_{\mathcal{S}}$
`Output:` Approximate gain $\sim\Delta L(\mathcal{S}, f)$ for each candidate feature $f \in \mathcal{F}$

1. For each $f \in \mathcal{F}$, calculate $\tilde{p}(f)$, the expected value of f in the training data

2. For each x, determine the set $\mathcal{F}(x) \subseteq \mathcal{F}$ of f that are *active* for x:

$$\mathcal{F}(x) \equiv \{f \in \mathcal{F} \mid f(x, y) p_{\mathcal{S}}(y|x)\tilde{p}(x) > 0 \text{ for some } y\} \tag{41}$$

3. For each f, let

$$r(f) = \begin{cases} 1 & \text{if } \tilde{p}(f) \leq p_{\mathcal{S}}(f) \\ -1 & \text{otherwise} \end{cases} \tag{42}$$

4. For each $f \in \mathcal{F}$, initialize $\alpha(f) \leftarrow 0$

5. Repeat the following until $\alpha(f)$ converges for each $f \in \mathcal{F}$:

   (a) For each $f \in \mathcal{F}$, set

$$\begin{aligned} G'(f) &\leftarrow \tilde{p}(f) \\ G''(f) &\leftarrow 0 \end{aligned}$$

   (b) For each x, do the following:

   For each $f \in \mathcal{F}(x)$, update $G'(f)$ and $G''(f)$ by

$$G'(f) \leftarrow G'(f) - \tilde{p}(x) p^{\alpha}_{\mathcal{S},f}(f|x) \tag{43}$$

$$G''(f) \leftarrow G''(f) - \tilde{p}(x) p^{\alpha}_{\mathcal{S},f}((f - p^{\alpha}_{\mathcal{S},f}(f|x))^2|x) \tag{44}$$

   where $p^{\alpha}_{\mathcal{S},f}(f|x) \equiv \sum_y p^{\alpha}_{\mathcal{S},f}(y|x) f(x, y)$

(c) For each $f \in \mathcal{F}$, update $\alpha(f)$ by

$$\alpha(f) \leftarrow \alpha(f) + \frac{1}{r(f)} \log \left(1 - \frac{1}{r(f)} \frac{G'(f)}{G''(f)}\right) \tag{45}$$

6. For each $f \in \mathcal{F}$, substitute $\alpha(f)$ into (26) to determine $\sim\Delta L(\mathcal{S}, f)$.

Convergence for this algorithm is guaranteed just as it was for algorithm 3 – after each iteration of step 5, the value of $\alpha(f)$ for each candidate feature f is closer to its optimal value $\alpha^\star(f)$ and, more importantly, the gain $G_{\mathcal{S},f}$ is closer to the maximal gain $\sim\Delta L(\mathcal{S}, f)$.