Finite Growth Models and the Learning of Edit Distance Costs

Eric Sven Ristad* Peter N. Yianilos†

July, 1996

Abstract

We introduce finite growth models (FGMs) and apply them to the problem of learning optimal costs for string edit distance. We present the first algorithm that learns the optimal insertion, deletion and substitution costs from a training corpus of similar strings. Since edit distance has found widespread application, our discovery of a simple way to learn costs may lead to improved performance in many problem areas. Our approach also yields a more powerful class of learnable string edit distances.

Finite growth models are acyclic stochastic automata which allow observations to be processed in many orderings and groupings – not just one-by-one in sequential order. These capabilities, not present in earlier approaches, are essential to our stochastic formulation of string edit distance, and separate our approach from the related field of hidden Markov modeling.

Keywords: *String Edit Distance, String Similarity, Expectation Maximization (EM), Hidden Markov Models (HMM), Baum-Welch Reestimation, Maximum Likelihood, Stochastic Models, Metric Learning.*

*Department of Computer Science, Princeton University. Email: ristad@cs.princeton.edu.

†NEC Research Institute; and Department of Computer Science, Princeton University. Email: pny@research.nj.nec.com.

1 Introduction

Many complex objects have a special recursive structure that arises from the process by which they were created. For example, an arrangement of dominos is the result of placing a single domino on the boundary of an existing domino arrangement. A mechanical assembly arises from the process of attaching a single component to an incomplete assembly. And a string is the outcome of process that repeatedly concatenates symbols onto an incomplete string. This manner of stochastic object formation process may be modeled with a finite growth model.

A *finite growth model* (FGM) is an acyclic stochastic automaton that assigns probability to objects in an associated domain. An FGM may also be thought of as a stochastic interpretation of a direct acyclic graph. We believe that the graph-based development of the FGM formalism provides a natural way for Computer Scientists to learn and reason about stochastic models including hidden Markov models (HMMs).

In this paper, we introduce finite growth models and apply them to the problem of learning a similarity measure on strings. Our development leads to a natural reformulation of string edit distance in stochastic terms, first described by Hall and Dowling in [6]. The central advantage of stochastic reformulation is that we may adapt the classic Baum-Welch reestimation algorithm to efficiently learn the optimal insertion, deletion and substitution costs from a training corpus of similar strings. This set may be thought of as positive examples of the *are similar* concept. Our algorithm then in effect learns a computational notion for similarity by adjusting the costs so as to best explain the training set. This is an instance of what the authors have called the *metric learning paradigm*.

A related benefit of our particular reformulation is that it paves the way to a more powerful class of string edit distances, where the cost of an edit operation depends on the target of the operation as well as the context in which the operation is applied. For example, such context-sensitive string edit distances would allow the cost of inserting a symbol to depend on the symbol left-adjacent to the insertion site.

The remainder of this paper consists of five sections. We begin in section 2 by developing the finite growth model formalism as a stochastic interpretation of directed acyclic graphs (DAGs). Our development of FGMs is self-contained, but we do include some discussion of their relationship to HMMs for those readers familiar with the latter. Next, section 3 applies FGMs to the string edit distance problem and section 4 shows that this application is a special case of a higher-order product operation for FGMs. These product FGMs define a stochastic notion of similarity between objects more complex then symbol strings. In section 5 we describe how stronger notions of edit distance may be expressed within the FGM framework, in which the cost of an edit operation depends on the context in which the operation is applied. Section 6 concludes by contrasting FGMs to hidden Markov models (HMMs).

2 Finite Growth Models

Let x be a vector observation with components $x_1, \ldots, x_d$, and $\mathcal{X}$ denote the space of all such observations. We will not specify the type of each component since for much of our development it matters only that a probability distribution exists on the space of values that a component may assume. As such, components may represent discrete values such as letters of the alphabet, continuous values such as real numbers, or other more complex structures. Within the observation vector, component types need not be the same. In simple *time series* settings, components do have the same

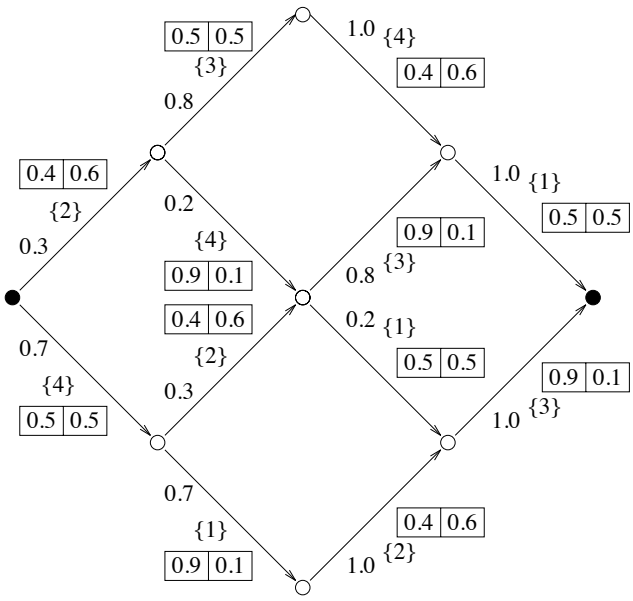

Figure 1: A simple finite growth model which induces a probability measure on Boolean strings of length 4.

type, and position within the vector corresponds to the passage of time.

A *Finite Growth Model* (FGM) consists of a DAG having a single source and sink, along with other information identified on its edges. We first introduce these models informally and give a concrete example. Later a more formal and abstract definition is presented. The source is the starting state, and the graph represents a stochastic automaton which is free of cycles. The sink is the final state. State advances (stochastically) in discrete time steps.

Each edge contains three pieces of information. The first is a *transition probability*. If an edge connects vertices $i \to j$, this value gives the probability of the machine being in state j in the next time step, conditioned on its being in state i at the present time. The sum of the transition probabilities over edges leaving any vertex must be unity. We imagine running this machine many times, always starting from the source. During each trial it randomly selects edges to follow according to the transition probabilities, and ultimately traces out a single path from source to sink. Each such single path generates a single complete observation vector as described below. In figure 1 the first value appearing along each directed edge is the transition probability.

While following an edge the machine generates one or more components of the observation vector. The second piece of information associated with each edge identifies which components are generated. In figure 1 a single observation is generated along each edge, and is shown enclosed in braces. In other settings, including the string edit distance problem we discuss later, more than one observation is generated along some edges. Along a source-sink path the observations need not be generated in any particular order. However an FGM DAG must satisfy the property that each source-sink path generates each component once, and may be thought of as corresponding to a permutation of the indices $1, \ldots, d$.

The third piece of information along an edge specifies the probability with which the designated observation components are generated. The domain of the probability function must agree in dimen-

sion and type with the designated observation components.

In figure 1 we assume that the observation vector has four components, and that each is a Boolean value. Accordingly, the probability distribution associated with each edge is represented as a 1×2 table giving the probability of 0 and 1 respectively.

Proposition 1 *A Finite Growth Model induces a probability measure on its corresponding space of observations.*

proof: Follows immediately from their construction. □

Later we give a formal definition of an FGM, that includes the concept of *parameter tying* introduced below. The "finite growth model" framework is so named because we imagine that complete observations are grown, starting from the empty set, in many possible orderings, as specified by the model's underlying graph.

2.1 Evaluating Probabilities

The probability of a particular observation is of course just the frequency of its appearance as the machine generates random vectors. Each source-sink path t is followed with some probability $\Pr(t)$, which is the product of the transition probabilities along it. Given that we follow t, observation vector x is generated with probability $\Pr(x|t)$, which is the product the component probability distributions on the path. It is then apparent that any FGM has a canonical *inflated* form in which every source-sink path in the original FGM, corresponds to a disjoint path in its canonical form. By *disjoint* we mean that any two source-sink paths, have only the source and sink vertices in common. An FGM is mathematically equivalent to its canonical form, but may represent a much more succinct representation in terms of vertices and edges. Since the number of paths through a DAG is frequently exponential in the number of vertices and edges, the issue of succinctness is an important one. The probability of observation x is then:

$$\Pr(x) = \sum_t \Pr(x,t) = \sum_t \Pr(x|t)\Pr(t)$$

Direct evaluation of this expression corresponds to visiting every vertex and edge of an FGM's canonical form. Fortunately $\Pr(x)$ may be computed using the original FGM graph using dynamic programming. To express this precisely requires some notation. Let $v_1, \ldots, v_n$ be a topologically sorted listing of the original FGM's vertices. Following the convention of the HMM literature, we define real variables $\alpha_1, \ldots, \alpha_n$ corresponding to each vertex. The set of edges *into* and *out from* a vertex v are denoted by $\mathcal{I}(v)$ and $\mathcal{O}(v)$ respectively. The index of the source vertex of edge e is denoted $s(e)$, and the index of the destination vertex is denoted $d(e)$. The transition probability associated with an edge e is denoted τ_e, and the set of generated observation components is written g_e. We denote by $\pi_{g_e}(x)$, the *projection* of observation vector x corresponding to selection of the components in g_e. For some edge e, suppose $g_e = \{2, 4\}$. Then $\pi_{g_e}(x)$ is a vector with only two components which are equal in value to the second and fourth components of x. Finally, the probability distribution associated with an edge e is written $p_e(\cdot)$.

Using the notation above, the contribution of an edge e to the probability of a path containing it, is $\tau_e \cdot p_e(\pi_{g_e}(x))$. The product of these contributions is the probability of the path. The complete dynamic program is then given by the following simple algorithm:

Algorithm 1

procedure *forward*

$\alpha_1 = 1$

for $i = 2, \ldots, n$

$\alpha_i = \sum_{e \in \mathcal{I}(v_i)} \alpha_{s(e)} \cdot \tau_e \cdot p_e(\pi_{g_e}(x))$

This is the forward step of the Baum-Welch *forward-backward* algorithm[2, 3, 1]. [1] After it has run, α_n equals $\Pr(x)$. However the other α values have meaning as well. In general α_i is the probability of arriving at v_i *and* observing that part of x corresponding to the paths between v_i and the source. Note that because every source-sink path corresponds to a permutation of the observation vector, every path from the source to v_i must generate the same observation components — possibly however in different orders. Dually, α_i is the probability that random operation of the machine encounters v_i, while at the same time generating those observation components accounted for by the source to v_i paths.

By construction, the edges leaving every vertex are stochastic, that is, their sum is one. This is seldom true in the reverse direction. Nevertheless, we still compute $\Pr(x)$ if the direction of our sweep through the DAG is reversed. We introduce a new set of real variables $\beta_1, \ldots, \beta_n$, and algorithm 1 becomes:

Algorithm 2

procedure *backward*

$\beta_n = 1$

for $i = n - 1, \ldots, 1$

$\beta_i = \sum_{e \in \mathcal{O}(v_i)} \beta_{d(e)} \cdot \tau_e \cdot p_e(\pi_{g_e}(x))$

This is the backward step of the Baum-Welch algorithm. After it has run, β_1 equals $\Pr(x)$, and therefore α_n. In general β_i is the probability of arriving at v_i *and* observing that part of x corresponding to the paths between v_i and the sink. Dually, β_i is the probability that random operation of the machine encounters v_i, while at the same time generating those observation components accounted for by the v_i to sink paths. The correctness of the *backward* algorithm follows from induction on DAG depth. One removes the source, creating a shallower sub-DAG for every edge leaving it. This argument is necessary since as remarked earlier, simply reversing edge direction does not leave a well-formed FGM, since in-bound edges in the original DAG need not sum to one. Note however that it is a simple exercise to show that after reversal, one may adjust transition probabilities in a straightforward way[2] to result in a valid FGM which is equivalent to the original.

The probability $\Pr(e, x)$ of following an edge e and observing x is then $\alpha_{s(e)} \tau_e p_e(\pi_{g_e}(x)) \beta_{d(e)}$. Note that when we write $\Pr(e, x)$, 'e' refers to the event of a transition over edge e. Then $\Pr(e|x) =$

[1] Several years following the development of HMMs, essentially the same mathematical ideas expressed in the Baum-Welch algorithm were rediscovered as the expectation maximization (EM) algorithm for mixture densities [5]. This work focused on parameter estimation for mixture densities, and seems to have had considerable influence on a somewhat independent community. A nice survey is provided by [10]. Our later mathematical development will draw from the EM perspective, so in what follows we will use the phrase *Baum-Welch/EM* to refer to our adaptation of these ideas to the FGM setting.

[2] First compute α values without using the observation models. Then working backwards through the DAG's vertices, for each $e \in \mathcal{I}(v)$, replace τ_e with $(\alpha_{s(e)} \tau_e) / \sum_{h \in \mathcal{I}(v)} (\alpha_{s(h)} \tau_h)$.

$\Pr(e,x)/\Pr(x)$, and is denoted γ_e following the naming convention of the HMM literature. That is, γ_e gives the *a posteriori* probability of following edge e, conditioned on the fact that we are observing (or generating) observation x. Having set the α and β variables, each γ may now be set. We show this for clarity as a separate pass through the DAG, although in practice it may be combined with the backward step.

Algorithm 3
procedure *posterior*
for $i = n-1,\ldots,1$
for $e \in \mathcal{O}(v_i)$
$\gamma_e = (\alpha_{s(e)} \cdot \tau_e \cdot p_e(\pi_{g_e}(x)) \cdot \beta_{d(e)})/\alpha_n$

The algorithms above are conceptually straightforward, but an important numerical issue must be faced in their implementation. As one sweeps through the DAG, the α and β variables can easily become too small to represent as conventional floating point values. Even the final value $\Pr(x)$ can become too small to represent since the probability of generating any particular observation, in general declines exponentially with dimension. Logarithmic representation is one solution, but we instead prefer a floating point representation with extended exponent range as reported in [11]. Because the γ variables are normalized by $\Pr(x)$, they may be adequately represented using standard floating point. For any x, the γ_e values must satisfy the following constraint which can in practice be used to check the algorithms above.

Proposition 2 *Given a finite growth model, an observation, and γ values computed by algorithms 1, 2, and 3, then:*

1. *If $|g_e| = 1, \forall e$, then $\sum_e \gamma_e = d$.*
2. *In general, $\sum_e \gamma_e \cdot |g_e| = d$.*

proof: Assume $|g_e| = 1, \forall e$. Because every source sink path path generates each observation component exactly once, the set of edges E_j which generate a particular component j, corresponds to a set of mutually exclusive and exhaustive events. Hence $\sum_{e\in E_j} \Pr(e|x) = \sum_{e\in E_j} \gamma_e = 1$. There are d observation components, and these partition E into disjoint sets $\{E_i\}$. So $\sum_{e\in E} \gamma_e = d$, establishing the first part of the proposition. In general, the $\{E_i\}$ are not disjoint, and the number of sets to which an edge e belongs is $|g_e|$. Multiplying each γ_e by $|g_e|$ in effect gives each set its own copy, so that the sum remains d. □

2.2 Parameter Tying and a Formal Definition

The concept of *parameter tying* from the HMM literature is also part of the FGM framework. Referring to figure 1, the transition probabilities on the edges leaving a vertex can be thought of as defining a choice between several possibilities. When designing a model, this choice often has a semantic interpretation. If the same choice must be made at several places within the DAG, then the model designer may choose to *tie* the associated transition probabilities, so that they have the same value everywhere the choice is made. Without tying, the learning process, to which we will turn

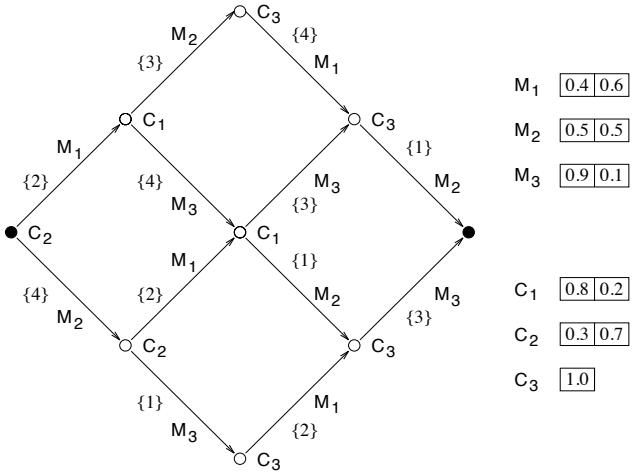

Figure 2: The finite growth model of figure 1 with with transitions and observation models tied.

next, might revise initially identical choices, so that they no longer match. In the FGM of figure 1 notice that two of the out-degree 2 vertices choose the upper vs. lower path with probability $0.3, 0.7$ respectively. The other two choose according to $0.8, 0.2$. All the out-degree 1 vertices must of course choose their only available path with probability 1. Figure 2 shows the same FGM in tied form. Note that there are no longer transition probabilities attached to an edge. Instead there is a label on each vertex (except the sink) identifying a probability distribution (referred to as a choice model). The learning process modifies these models ensuring that all their instances will remain identical in the updated model. We remark that one need not tie choices just because they are identical. In our example we might instead have created a fourth choice model C_4 associated only with the source. This essentially unties the two $0.3, 0.7$ vertices, so that they may diverge during training. Observation models may be tied as well, and this too is is depicted in figure 2. Tying clearly reduces the number of free parameters in the model, and this provides another motivation for the technique. The notion of parameter tying completes our introduction to FGMs, and we now define them in a more formal, albeit less transparent way.

Definition 1 *A "finite growth model" F for a d-dimensional observation space $\mathcal{X}$, is a 7-tuple (V, E, M, C, m, c, g), such that:*

1. *(V, E) is a finite DAG having a single source v_1 and sink v_n.*
2. *M is a finite collection $\{M_i\}$ of parameterized probability functions, each having domain corresponding to some projection of $\mathcal{X}$ (i.e. some subset of its dimensions), and parameters denoted Φ_{M_i}. We write $p_{M_i}(x|\Phi_{M_i})$ to denote evaluation of the ith function.*
3. *C is a finite collection $\{C_i\}$ of discrete probability functions, where N_i denotes the number of choices in the ith model. That is, a collection of values $\{C_{i_j}\}$ such that:*

$$\sum_{j=1}^{N_i} C_{i_j} = 1, \forall i$$

4. $m : E \to M$ *is a function associating an element of* M *with each edge.*

5. $c : E \to \{C_{i_j}\}$ *is a function such that if* $c(e) = C_{i_j}$*, then the out-degree of* $s(e)$ *equals* N_i*, and given distinct edges* e_1, e_2 *with* $s(e_1) = s(e_2)$ *and* $c(e_1) = C_{i_j}, c(e_2) = C_{k_\ell}$*, then* $i = k$ *and* $j \neq \ell$*.*

6. $g : E \to d^*$ *where by* d^* *we denote the set of all nonempty subsets of the* d *dimensions of* $\mathcal{X}$*, such that every source sink path corresponds under* g *to a partition of these dimensions, and for every edge* e*, the projection corresponding to* $g(e)$ *is compatible with observation model* $m(e)$*.*

The "parameters" of F*, by convention denoted* Φ*, consist of the choice model probabilities* $\{C_{i_j}\}$ *together with the observation model parameters* $\{\Phi_i\}$*.*

2.3 Baum-Welch Learning

We have seen that given an observation x, we may easily compute its probability given the model. Our learning objective is to find a set of model parameter values which maximizes this probability. This corresponds to *maximum-likelihood* parameter estimation in statistics. Learning from a single example might seem rather uninteresting, but as figure 3 illustrates, a method for doing so will readily generalize to the multiple observation case. In the figure, a training series $x^1, x^2, \ldots, x^T$ of independent observations is available. Notice that each has a different number of components: $4, 7, 3$. Associated with the observations are FGMs denoted $F^1, F^2, \ldots, F^T$. Simply placing their DAGs end to end results in a single combined FGM which now generates a combined observation vector of length 14. This single vector represents the whole training set, and its probability under the combined FGM is clearly just the product of the probabilities assigned by the individual FGMs.

Proposition 3 *A finite cascade* F *of FGMs* $F^1, \ldots, F^T$ *is again an FGM over the direct product of the individual observation spaces. Also, given a corresponding series of observations* $x^1, \ldots, x^T$*, and their vector concatenation* x*:*

$$\Pr(x|F) = \prod_{i=1}^{T} \Pr(x^i|F^i)$$

proof: That the cascade is again an FGM follows immediately from the definition. Notice that in algorithm 1, the initial value α_1 is set to 1. Operating on the cascade, the algorithm will reach the sink of F^1 and set its α value to $\Pr(x^1|F^1)$. But this is the source of F^2, and the algorithm will continue as though its first α value is $\Pr(x^1|F^1)$, not 1. So when the sink of F^2 is reached, its α is set to $\Pr(x^1|F^1) \cdot \Pr(x^2|F^2)$, and so on. □

Now the series of FGMs need have no relationship to one other, but typically are either identical, or are instances of a single design – varied in order to accommodate the structure of the observation vector. In the case of HMMs, and the approach we describe later for the string edit distance problem, the FGMs share a common set of choice models and observation probability models. So a set of parameter values which maximizes the probability of this single observation given the combined model, leaves us with a set of shared parameter values which assign maximal probability to the training set.

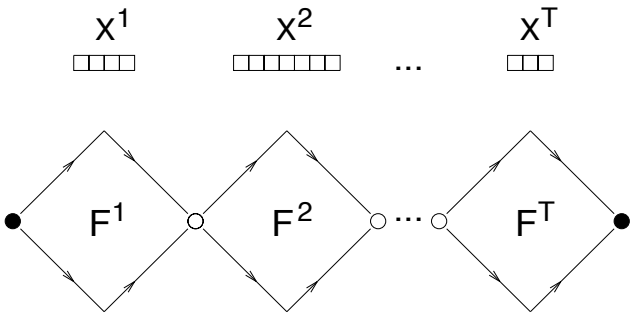

Figure 3: Cascading finite growth models results in a single model over combined observation vectors.

The Baum-Welch/EM prescription for maximizing the probability of x is to climb towards a local maximum in a series of steps. Given a set Φ of parameter values, what is needed is a set $\overline{\Phi}$ such that $\Pr(x|\overline{\Phi}) > \Pr(x|\Phi)$, unless Φ is already a local maximum. In our setting, the parameters are of course the values within the various choice models (transition probabilities) and observation models associated with the FGM. The beauty of Baum-Welch/EM is that a simple closed form expression frequently exists for $\overline{\Phi}$ in terms of Φ and x. We first focus on discrete observation spaces, but our proof of theorem 1 will apply to continuous settings as well.

Notice that in figure 2, both the choice models and observation models consist of a stochastic vector. Every edge in the FGM corresponds to a particular position within a single choice vector. Given that we have observed x, every edge also corresponds to a position within a single observation vector. One Baum-Welch/EM iteration for discrete observation spaces, then consists of the following simple steps:

1. Make a copy of all parameter vectors in the model (both choice and observation), and initialize them to zero. In step 2 they serve as accumulators, then after step 3 hold the improved model parameters.

2. Compute γ values for each edge, and add each to its two corresponding accumulators.

3. Normalize each accumulator vector so that it sums to one.

Remarkably, the resulting model is always an improvement. In addition to being simple, this method is also quite intuitive. The improved transition probabilities are merely set to be proportional to the frequency with which we expect to see each transition given x. Likewise for the observation probabilities. We next define this method in general terms, where the observation models need not be discrete – and in principle could themselves consist of entire FGMs.

Definition 2 *Let $F = (V, E, M, C, m, c, g)$ be an FGM with parameters Φ, and x an observation. Then the "Baum-Welch/EM reestimate" $\overline{\Phi}$ of the parameters of F is formed by solving two sets of independent optimization problems:*

1. *For each i, let $\gamma_{C_{i_j}} \triangleq \sum_{e \in c^{-1}(C_{i_j})} \gamma_e$, where $\gamma_e = \Pr(e|x)$. Then the improved choice model $\overline{C}_i$ is given by:*

$$\overline{C}_i \in \text{argmax}_{\{C'_{ij}\}} \sum_{j=1}^{N_i} \gamma_{C_{ij}} \log C'_{ij}, \quad \textit{subject to} \quad \sum_{j=1}^{N_i} C'_{ij} = 1$$

which has the simple and unique well-known solution:

$$\overline{C}_{ij} = \frac{\gamma_{ij}}{\sum_{j=1}^{N_i} \gamma_{ij}}$$

2. For each i, the improved parameters $\overline{\Phi}_{M_i}$ of observation model M_i are given by:

$$\overline{\Phi}_{M_i} \in \text{argmax}_{\Phi'_{M_i}} \sum_{e \in m^{-1}(M_i)} \gamma_e \log p_{M_i}(\pi_{g_e}(x) | \Phi'_{M_i})$$

Each summation term in the definition's expression for reestimation of observation models, corresponds to some portion $\pi_{g_e}(x)$ of the entire observation x. Each term is then weighted by γ_e. Exponentiating the summation, yields a product – from which it is apparent that this step amounts to maximum-likelihood parameter estimation, where the γ_e terms appear as exponents, and are thought of as multiplicities. For example, if $\gamma_e = 2$, it is as though the corresponding portion $\pi_{g_e}(x)$ of the observation vector, occurs twice in the sample. We view Baum-Welch/EM as separating the problem of improving an FGM, into independent weighted subproblems. If some M_i is again an FGM, then its improvement may again be broken up into smaller independent subproblems – demonstrating the recursive nature of Baum-Welch/EM applied to FGMs.

If each $\pi_{g_e}(x)$ is a symbol σ from some finite alphabet Σ, then M_i is a discrete probability function, and is easily maximized. Let Γ_σ denote the sum of all γ_e such that $\pi_{g_e}(x) = \sigma$. Then as for the choice models:

$$p_{\overline{M}_i}(\sigma) = \frac{\Gamma_\sigma}{\sum_{\nu \in \Sigma} \Gamma_\nu}$$

and the correspondence of definition 2 with our earlier narrative description of Baum-Welch/EM is established.

If M_i is a multivariate normal density, then the $\pi_{g_e}(x)$ are real vectors, and here too maximization is easily accomplished. The weighted sample mean, and weighted sample covariance define, the optimal normal density. The weights are γ_e/Γ where Γ denotes the sum of all γ_e associated with M_i. In general, any density for which a maximum-likelihood estimate is readily available, may be used as an observation model. Through the FGM graph structure, normal mixtures may be expressed, and through parameter tying the *semi-continuous* models of [8] are readily implemented. Both are common in speech recognition systems. See [7] for a treatment of HMMs including a compact discussion of the discrete and continuous maximum-likelihood optimization problems discussed above.

Theorem 1 *Let F be an FGM with parameter set Φ, and x denote an observation. Then Baum-Welch/EM reestimation, restated for FGMs, yields a revised parameter set $\overline{\Phi}$ such that* $\Pr(x|\overline{\Phi}) \geq \Pr(x|\Phi)$*, with the inequality strict so long as progress is made in at least one of the defining* argmax *operations.*

Before proving this theorem we establish an important lemma which is the mathematical heart of Baum-Welch/EM, and applies equally well to HMMs, FGMs, and mixture density estimation. It is the central result of the EM literature, from which our proof is drawn. As noted earlier, this basic mathematical idea was first discovered some years earlier in the development of HMMs. Because FGMs are discrete state models, our lemma deals with finite mixtures, and is stated so as to fit into our development. But the lemma and its brief proof carry over easily to the continuous mixture case. See [10] pages 214–217 for general and compact development. We state and prove it here so that our treatment of FGMs is self-contained. Finite mixtures are relevant to HMMs operating on a finite observation sequence, and to FGMs, since as remarked earlier, both may be viewed as immense mixtures having one component for every path through the machine's states. HMMs are a succinct representation of many such mixtures, and FGMs succinctly represent all of these and more.

Lemma 1 *(Dempster, Laird, Rubin) Let* $\Pr(x|\Phi) = \sum_{i=1}^{k} g_i(x|\omega_i, \Psi)h(\omega_i|\Upsilon)$ *be a finite parameterized mixture density, where* ω_i *denotes the discrete random variable selecting a component of the mixture, and* $\Phi \triangleq \{\Psi, \Upsilon\}$*. Then:*

$$\overline{\Phi} \in \text{argmax}_{\Psi',\Upsilon'} \left(\sum_{i=1}^{k} \Pr(\omega_i|x,\Phi) \log g_i(x|\omega_i,\Psi') + \sum_{i=1}^{k} \Pr(\omega_i|x,\Phi) \log h(\omega_i|\Upsilon') \right)$$

is an improved parameter set. That is, $\Pr(x|\overline{\Phi}) \geq \Pr(x|\Phi)$*, with the inequality strict unless* Φ *is already a member of* $\text{argmax}_{\Psi',\Upsilon'}(\cdot)$*. It is understood that* Ψ', Υ' *vary over their defined domain.*

proof: We begin by establishing the identity:

$$\begin{aligned} E_{\omega|x,\Phi}(\log \Pr(x|\Phi')) &= \sum_{i=1}^{k} \Pr(\omega_i|x,\Phi) \log \Pr(x|\Phi') \\ &= \log \Pr(x|\Phi') \sum_{i=1}^{k} \Pr(\omega_i|x,\Phi) \\ &= \log \Pr(x|\Phi') \end{aligned}$$

Next $\log \Pr(x|\Phi') = \log \Pr(x,\omega|\Phi') - \log \Pr(\omega|x,\Phi')$ since $\Pr(x,\omega|\Phi') = \Pr(\omega|x,\Phi') \Pr(x|\Phi')$. So:

$$\begin{aligned} \log \Pr(x|\Phi') &= E_{\omega|x,\Phi} \log \Pr(x|\Phi') \\ &= E_{\omega|x,\Phi} \log \Pr(x,\omega|\Phi') - E_{\omega|x,\Phi} \log \Pr(\omega|x,\Phi') \\ &= \sum_{i=1}^{k} \Pr(\omega_i|x,\Phi) \log \Pr(x,\omega_i|\Phi') - \sum_{i=1}^{k} \Pr(\omega_i|x,\Phi) \log \Pr(\omega_i|x,\Phi') \end{aligned}$$

It follows from Jensen's inequality that the second summation is minimized by $\Phi = \Phi'$. This can also be seen by recognizing it as $-[D(\Phi\|\Phi') + H(\Phi)]$, where $D(\cdot\|\cdot)$ is the Kullbak-Leibler distance, and $H(\cdot)$ denotes entropy (see [4]). Thus maximizing the first term, written $Q(\Phi, \Phi')$ in the literature, will surely increase $\log \Pr(x|\Phi')$ and hence $\Pr(x|\Phi')$. But:

$$\sum_{i=1}^{k} \Pr(\omega_i|x,\Phi) \log \Pr(x,\omega_i|\Phi') = \sum_{i=1}^{k} \Pr(\omega_i|x,\Phi) \log g_i(x|\omega_i,\Psi') + \sum_{i=1}^{k} \Pr(\omega_i|x,\Phi) \log h(\omega_i|\Upsilon')$$

and we are done. □

Lemma 1 and theorem 1 define the reestimated parameters $\overline{\Phi}$ as the combined solutions of a set of independent maximization problems. But from the lemma's proof it is clear that *any* improvement in these related problems will translate to an improved parameter set. This is important when exact solutions are not available. For discrete observation models, and many continuous ones (notably normal densities), each independent maximization problem has a unique globally optimal solution, and satisfies certain continuity and differentiability requirements. In this case the lemma can be strengthened to say that process of repeated reestimation will converge to a local maximum of the likelihood function. These comments apply to theorem 1 as well. See See [10] for a survey of the convergence properties of the EM algorithm.

If one of the component densities is itself a mixture, then one may in principle expand the original mixture to absorb the submixture. But equivalently a weighted Baum-Welch/EM step may be taken for the submixture in isolation. This illustrates the intrinsically recursive nature of the approach. The same is true also at the FGM level where an observation model may consist of another FGM.

proof: (of theorem 1) We denote by $\omega_1, \ldots, \omega_N$ the source-sink paths through the FGM, and by E_{ω_i} the set of edges along a particular path. Then:

$$\Pr(x|\Phi) = \sum_{i=1}^{N} \underbrace{\left(\prod_{e \in E_{\omega_i}} p_{m(e)}(\pi_{g_e}(x)|\Phi_{m(e)}) \right)}_{g_i(x|\omega_i,\Psi)} \cdot \underbrace{\left(\prod_{e \in E_{\omega_i}} c(e) \right)}_{h(\omega_i|\Upsilon)}$$

which illustrates the correspondence between our FGM and lemma 1. The parameters Ψ of the lemma correspond to $\{\Phi_{M_i}\}$, and Υ to $\{C_{i_j}\}$. Then denoting by γ_{ω_i} the expression $\Pr(\omega_i|x,\Phi)$ we have by the lemma:

$$\overline{\Phi} \in \text{argmax}_{\Phi'_{M_j}, C'_{j_k}} \left[\sum_{i=1}^{N} \gamma_{\omega_i} \log \left(\prod_{e \in E_{\omega_i}} p_{m(e)}(\pi_{g_e}(x)|\Phi'_{m(e)}) \right) + \sum_{i=1}^{N} \gamma_{\omega_i} \log \left(\prod_{e \in E_{\omega_i}} c'(e) \right) \right]$$

where $c'(e)$ denotes the corresponding choice parameter C'_{j_k}. After converting the product terms to sums we have:

$$\overline{\Phi} \in \text{argmax}_{\Phi'_{M_j}, C'_{j_k}} \left[\sum_{i=1}^{N} \sum_{e \in E_{\omega_i}} \gamma_{\omega_i} \log p_{m(e)}(\pi_{g_e}(x)|\Phi'_{m(e)}) + \sum_{i=1}^{N} \sum_{e \in E_{\omega_i}} \gamma_{\omega_i} \log c'(e) \right]$$

The double summations in the expression above, enumerate every edge, along every path though the FGM. To complete the proof we have only to enumerate them in a different order. Given edge e, let $\Omega_e = \{\omega | e \in E_\omega\}$. Then:

$$\overline{\Phi} \in \text{argmax}_{\Phi'_{M_j}, C'_{j_k}} \left[\sum_{i=1}^{|M|} \sum_{e \in m^{-1}(M_i)} \sum_{\omega \in \Omega_e} \gamma_\omega \log p_{M_i}(\pi_{g_e}(x) | \Phi'_{M_i}) + \sum_{i=1}^{|C|} \sum_{j=1}^{N_i} \sum_{e \in c^{-1}(C_{i_j})} \sum_{\omega \in \Omega_e} \gamma_\omega \log C'_{i_j} \right]$$

Now $\gamma_e = \sum_{\omega \in \Omega_e} \gamma_\omega$, and using the notation $\gamma_{C_{i_j}}$ from definition 2 we have:

$$\overline{\Phi} \in \text{argmax}_{\Phi'_{M_j}, C'_{j_k}} \left[\sum_{i=1}^{|M|} \left(\sum_{e \in m^{-1}(M_i)} \gamma_e \log p_{M_i}(\pi_{g_e}(x) | \Phi'_{M_i}) \right) + \sum_{i=1}^{|C|} \left(\sum_{j=1}^{N_i} \gamma_{C_{i_j}} \log C'_{i_j} \right) \right]$$

and the parenthesized terms exactly correspond to the independent subproblems of definition 2. □

We have established that observations may be generated in an arbitrary order, and in various groupings, while retaining the ability to use Baum-Welch/EM reestimation. It is interesting that in the proof above, the only real requirement is that along a single source-sink path, the product of observation models yields a probability on the entire observation vector. This implies that the observation models might, for example, have conditional form – so long as their product yields a probability. In this case the theorem's reestimation prescription remains mathematically correct, but the γ_e values it depends on may in practice be hard to compute since, given conditional forms, either the forward or the backward algorithm becomes problematic.

For completeness, we conclude the section with several observations – all of them known to HMM practitioners. i) Our definition of FGMs requires that for every edge e, $g(e) \neq \emptyset$. This restriction is not necessary, but we find it convenient, and point out that even in its presence, one may easily synthesize so called *nonemitting* edges. To do so, the observation vector is augmented with dummy components as necessary, each capable of assuming only a single value. ii) Next we observe that zero valued choice parameters remain zero upon reestimation. The same is true of discrete observation models. iii) Finally we remark that it is clear from the proof of theorem 1 ,that one may choose to not reestimate one or more portions of the model, and still improve it.

3 String Edit Distance

The most common notion for edit distance between two finite strings s, t over finite alphabet Σ, finds the least costly way to transform s into t via single-symbol insertions, deletions, and substitutions. The non-negative costs of these primitive edit operations are parameters to the algorithm. In practice the costs depend on the problem domain, and may be represented as a single table $c_{i,j}$ of size $|\Sigma|+1 \times |\Sigma|+1$, where row and column 0 correspond to an additional alphabet member ϵ representing the null character. The entry in table position i, j gives the cost of substituting symbol j of s for symbol i of t. The first row gives insertion costs, the first column gives deletion costs, and the remaining entries specify substitution costs. The table need not be symmetric. Recall that a simple dynamic program finds the edit distance in $O(|s| \cdot |t|)$ time. See [12] for a thorough treatment of string edit distance, or [6] for more compact discussion.

3.1 A Stochastic Formulation

Rather than imagining an editing process that transforms s into t, we equivalently imagine that the pair (s,t) is *grown* in a series of steps of three kinds: joint steps, left steps, and right steps. This outlook was first introduced in section 3.2.2 of [6]. Our contribution is the further development of this viewpoint to include the learning of costs, and the construction of more sophisticated distance functions as described in section 5. The initial state of the process is $(\emptyset,\emptyset)$, where $\emptyset$ denotes the null string. A joint step concatenates symbols to both elements of the pair. A right step adds a symbol to only the right element of the pair, while a left step adds a symbol to the left element. In the language of string edit distance, a joint step corresponds to a substitution operation. When a joint step adds the same symbol to both strings, one is substituting a symbol for itself. A right step corresponds to an insertion operation, and a left step corresponds to a deletion operation. This generative view of edit distance begins to make clear its relationship to FGMs.

The task of learning an optimal cost table is clearly under-constrained, because the string edit distance for all strings may be trivially minimized by setting all edit costs to zero. Our outlook is stochastic and we therefore interpret each cost as $-\log_2()$ of the corresponding event probability. For example, an entry in the top row expresses the probability of choosing to insert a particular symbol into the left string. The natural constraint then requires that the probabilities that underly the cost table must sum to one. Expressed in terms of costs $c_{i,j}$, this translates to:

$$\sum_{i,j} 2^{-c_{i,j}} = 1$$

For technical reasons which will be clarified later, we must introduce the '\$' symbol as a second extension to the alphabet. This adds an additional row and column to the cost table. Figure 4 depicts the table after conversion to probabilities. The zero entries correspond to infinite costs. We denote by $\chi_{i,j}$ the cost table in probabalistic form. The stochastic choice between left, right, and joint generation is implicit in this table. That is, χ combines transition and observation probabilities. For example, the sum of the top row may be interpreted as the probability of choosing to perform a left insertion.

Definition 3 *The "stochastic edit distance" between strings s and t, is $-\log_2(\Pr((s,t)|\chi)$. Here the probability refers to the infinite generative stochastic process in which symbols are inserted at the end of the right string, or at the end of the left string, or jointly at the end of both. Specifically, it is the probability that pair (s,t) will occur as an intermediate stage of generation, during a random trial. The process is parameterized by table χ giving the probability of these events, as described in the text.*

The $-\log_2()$ operation converts the probability to a non-negative "distance", which we will later see corresponds closely to the standard notion of edit distance. Also notice that in the definition, we refer to the probability of the ordered pair (s,t). This is an important distinction since χ need not be symmetric, and it might be that $\Pr((s,t)|\chi,|s|,|t|) \neq \Pr((t,s)|\chi,|s|,|t|)$. It is also possible to interpret $\Pr((s,t)|\chi)$ with respect to a finite event space – namely the set of all string pairs with the same combined length as (s,t). That is, the sum of the probabilities of all such strings is one. This is seen by observing that the events $\{G_{a,b}\}$ consisting of the generation of an intermediate result having lengths a,b where $a+b=|s|+|t|$, are mutually exclusive and exhaustive.

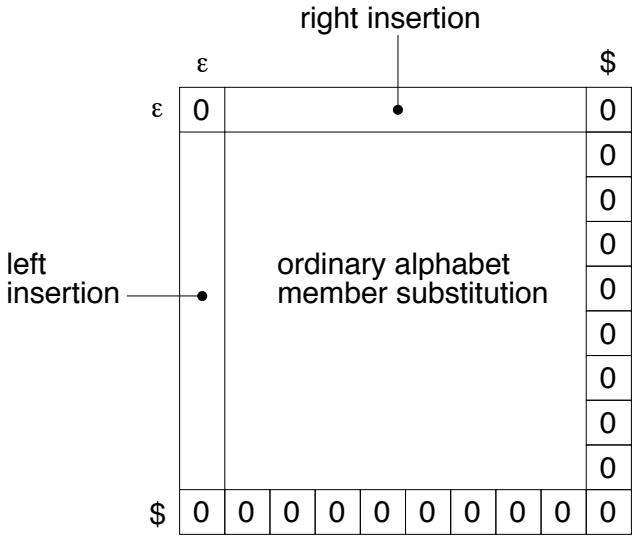

Figure 4: Augmented table of edit distance probabilities. This table sums to one, and the corresponding edit costs are $-\log_2()$ of its values.

Figure 5 depicts the FGM corresponding to stochastic edit distance between strings of length 4 and 5. Its 3-way branching pattern follows from the structure of the imagined growth process, and matches exactly the dependency structure of the dynamic program for conventional edit distance. The FGM for strings of arbitrary fixed length, is an obvious generalization.

The role of '$' is to effectively *truncate* the stochastic process, so that we can compute $\Pr((s,t)|\chi)$ as efficiently as possible. Before evaluating the probability of (s,t) using the FGM, we add '$' to the end of both, resulting in the pair (s^+,t^+). The FGM is then designed so that the probability of this modified pair as computed by the FGM, is exactly $\Pr((s,t)|\chi)$, from definition 3. Notice that all but the bottom and rightmost vertices have ternary outdegree, and that the edges leaving them are drawn with solid lines. These are all tied, and both their choice and observation models are specified by χ, corresponding to normal operation of the generative stochastic process. Now focus on the vertices and edges along the right column and bottom row of Figure 5 – excluding the sink. To reach one of them from the FGM's interior, whether by single or joint generation, a '$' must be observed. But as shown in figure 4, the row and column corresponding to '$' is entirely zero. These edge vertices must then have zero α values, and therefore make no contribution to the sink's α. This is true for any observation model that might be attached to them. Next consider the sink's diagonal predecessor, and observe that the edges leaving it are drawn with a dashed line. The choice model for this vertex selects the diagonal outgoing edge with probability one. That edge's observation model selects joint generation of a pair of '$' symbols, with probability one. The sink's α value is then identical to that of its diagonal predecessor, which is of course $\Pr((s,t)|\chi)$. In this way truncation of the stochastic process is effected.

The stochastic edit distance $-\log_2 \Pr((s,t)|\chi)$ is highly related, but not identical to conventional edit distance. If stochastic table χ is converted by $-\log_2()$ to a table of non-negative costs, then conventional edit distance corresponds to the log-probability of the *single best path* through the FGM – and is sometimes called the *Vitterbi decode* in the HMM literature. Stochastic edit distance is the

sum over *all paths*. Basing decisions on the Vitterbi decode is analogous to focusing on the mode of a distribution while stochastic edit distance corresponds to the mean. In classification systems we expect that both forms will lead to nearly identical decisions, but predict that stochastic edit distance may be slightly superior. The improved costs we learn in the next section will provably reduce stochastic edit distances, but we also expect that they will in practice improve results for the conventional form as well.

3.2 Algorithms

We formalize the application of FGMs to stochastic edit distance with the following:

Proposition 4 *Let χ be a table of stochastic edit distance probabilities, and $(s_1, t_1), (s_2, t_2), \ldots, (s_T, t_T)$ denote a set of string pairs over finite alphabet Σ, where ℓ bounds the length of any single string. Next let $E_\chi(s,t)$ denote the stochastic edit distance $-\log_2 \Pr((s,t)|\chi)$. Then:*

1. *There exists an $O(|s| \cdot |t|)$ time, $O(\min(|s|, |t|))$ space algorithm to compute $E_\chi(s,t)$ – matching the complexity of the conventional dynamic program.*

2. *There exists an $O(\ell^2 T)$ time and space algorithm for producing a revised cost table χ' such that:*

$$\sum_{i=1}^{T} E_{\chi'}(s_i, t_i) \leq \sum_{i=1}^{T} E_\chi(s_i, t_i)$$

where the inequality is strict unless χ is already a local maximum, and the sequence of rees-timates converges to a cost table which locally optimizes $E_\chi(s,t)$.

proof:

1. From the cost table χ we may construct a choice model and observation models, completing the FGM. The algorithm is then the α computation of algorithm 1, and was first reported as a simple recurrence in [6]. We assume without loss of generality that $|s| \leq |t|$. With reference to figure 5, we choose the topological vertex ordering which starts at the source, scans columns from top to bottom, and after each proceeds to the top of the adjacent column. Under this ordering, a vertex's α value depends only on values in its column, and the previous column. So there is no need to maintain a full $|s| \times |t|$ array of α values, and the space requirement is $O(\min(|s|, |t|))$. The time complexity is $O(|s| \cdot |t|))$ since as many vertices are visited, and constant work is performed for each.

2. Baum-Welch/EM in FGM form is applied – from which the proposition follows. Convergence follows from the fact that we have a finite discrete observation space.

□

The FGM formalism is useful to arrive at a correct solution, but because of the extensive parameter tying employed in our construction, the associated computations may be greatly simplified, and there is no need to build actual DAGs during the computation. Also as remarked earlier, the

choice model is implicit in χ, so that creation of separate choice and observation models is unnecessary when computing $E_\chi(s,t)$. The probability of passing over an edge is just the corresponding entry in χ. Finally, the role of '\$' is entirely formal, and the extra DAG row and column it adds has no effect on the algorithm. This is easy to see since all paths through this fringe are of probability zero given (s^+, t^+).

The result is algorithm 4. It requires an $(|s|+1) \times 2$ array α of extended exponent range floating point values (see section 2.1 and [11]). We write $\alpha_{i,j}$ to denote an element, where $1 \leq i \leq |s|+1$ and j is 1 or 2. This array corresponds to the vertices within two adjancent columns of the FGM as depicted in figure 5. The logarithm E_χ of the joint probability of s and t is returned rather than the probability itself, so that the caller need not deal with extended range floating point values. As observed by [6], the algorithm's structure resembles that of the standard dynamic program for edit distance.

Algorithm 4

```
procedure Evaluate(s, t)
    α_{1,1} = 1
    for j = 1, ..., |t| + 1
        k = 2 − j mod 2
        ℓ = 1 + j mod 2
        for i = 1, ..., |s| + 1
            α_{i,k} = 0
            if i > 1
                α_{i,k} += α_{(i−1),k} · χ_{s_{(i−1)},0}
            if j > 1
                α_{i,k} += α_{i,ℓ} · χ_{0,t_{(j−1)}}
            if i > 1 ∧ j > 1
                α_{i,k} += α_{(i−1),ℓ} · χ_{s_{(i−1)},t_{(j−1)}}
    return log_2 α_{(|s|+1),k}
```

As in the case of evaluation, there is no need to create seperate choice and observation models when implementing reestimation. A brief argument establishes this fact. Assume we do have seperate models. Then each edge e is of type *left*, *right*, or *joint*, and during Baum-Welch/EM the quantity γ_e will be added to a choice accumulator, and also to an accumulator corresponding to the observed alphabet symbol or symbols. So the total γ contribution to each of the three choice accumulators will exactly match the contribution to its corresponding observation model. Hence, if we instead add the γ values directly into an initially zero array $\overline{\chi}$, the total contributions to the left column, top row, and interior, will exactly correspond to the choice accumulator contributions above. So after normalization, $\overline{\chi}$ will represent the same model as that resulting from reestimation based on seperate models.

Algorithm 5 is the central routine in the stochastic edit distance learning process. It is called repeatedly with pairs of strings. The current cost table $\chi_{i,j}$ must be initialized before the first call, and the improved cost table $\overline{\chi}$ must be set to zeros. As each pair is processed, non-negative values are added to locations within $\overline{\chi}$. After all the pairs have been presented, the $\overline{\chi}$ array is simply normalized so that the sum of its entries is one, and then represents the set of improved costs. This

entire process may be repeated to further improve the model. That is, $\overline{\chi}$ is copied to χ and $\overline{\chi}$ is again set to zeros. The sequence of training pairs is then presented again, and so on.

The algorithm uses two work arrays, $\alpha_{i,j}$ and $\beta_{i,j}$, corresponding to the algorithms 1 and 2 respectively. It is assumed that these arrays are large enough to accomodate the training pairs. They need not be initialized by the caller. As in algorithm 4 it is possible to reduce the size either α or β, but not both. Doing so results in a constant 2:1 space savings, but for clarity we present the algorithm using complete arrays. Finally we remark that our formulation of and algorithms for stochastic string edit distance, are readily generalized to continuous observations.

Algorithm 5

procedure *LearnCosts(s, t)*

$\alpha_{1,1} = 1$
for $i = 1, \ldots, |s| + 1$
  for $j = 1, \ldots, |t| + 1$
    $\alpha_{i,j} = 0$
    if $i > 1$
      $\alpha_{i,j} \mathrel{+}= \alpha_{(i-1),j} \cdot \chi_{s_{(i-1)},0}$
    if $j > 1$
      $\alpha_{i,j} \mathrel{+}= \alpha_{i,(j-1)} \cdot \chi_{0,t_{(j-1)}}$
    if $i > 1 \wedge j > 1$
      $\alpha_{i,j} \mathrel{+}= \alpha_{(i-1),(j-1)} \cdot \chi_{s_{(i-1)},t_{(j-1)}}$
$p = \alpha_{(|s|+1),(|t|+1)}$
$\beta_{(|s|+1),(|t|+1|)} = 1$
for $i = |s| + 1, \ldots, 1$
  for $j = |t| + 1, \ldots, 1$
    $\beta_{i,j} = 0$
    if $i \leq |s|$
      $\beta_{i,j} \mathrel{+}= \beta_{(i+1),j} \cdot \chi_{s_{(i+1)},0}$
      $\overline{\chi}_{s_{(i+1)},0} \mathrel{+}= (\alpha_{i,j} \cdot \chi_{s_{(i+1)},0} \cdot \beta_{(i+1),j})/p$
    if $j \leq |t|$
      $\beta_{i,j} \mathrel{+}= \beta_{i,(j+1)} \cdot \chi_{0,t_{(j+1)}}$
      $\overline{\chi}_{0,t_{(j+1)}} \mathrel{+}= (\alpha_{i,j} \cdot \chi_{0,t_{(j+1)}} \cdot \beta_{i,(j+1)})/p$
    if $i \leq |s| \wedge j \leq |t|$
      $\beta_{i,j} \mathrel{+}= \beta_{(i+1),(j+1)} \cdot \chi_{s_{(i+1)},t_{(j+1)}}$
      $\overline{\chi}_{s_{(i+1)},t_{(j+1)}} \mathrel{+}= (\alpha_{i,j} \cdot \chi_{s_{(i+1)},t_{(j+1)}} \cdot \beta_{(i+1),(j+1)}) /p$

4 An FGM Product Operation

In our edit distance FGM, each vertex faces three choices. Two choices correspond to single symbol generation, and a third choice selects joint symbol generation. One may view the resulting graph as a product of two linear FGMs, each of which grows a single string from left to right. We now generalize this notion. Given two FGMs F^A, F^B we define the generation graph of the product FGM $F^A \otimes F^B$ as follows:

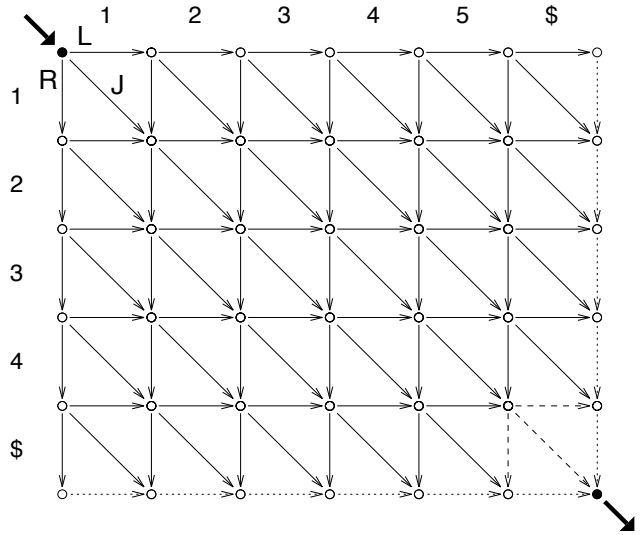

Figure 5: Finite growth model DAG corresponding to computation of the edit distance between strings of length 4 and 5.

1. The vertices of the product are pairs (v, w) where v, w are vertices F^A and F^B respectively.
2. Edge $(v, w) \to (r, s)$ exists in the product graph iff:
   (a) $v \to r \in F^A$ and $w = s$. Edges of this type *run* the left FGM only while keeping the right *idle*. For edit distance this corresponds to insertion into the left string, or;
   (b) $w \to s \in F^B$ and $v = r$ corresponding in edit distance to right string insertion, or;
   (c) $v \to r \in F^A$ and $w \to s \in F^B$. Here both machines run and a pair of symbols are generated corresponding to edit distance substitution.

It is readily verified that the resulting product FGM has a single source and sink, and that each path corresponds to a permutation of the joint observation vector. However the product FGM is incomplete until the probabilities of the new transitions and the joint observations, are specified.

Given a pair of observations, the product model then computes their joint probability. Each generation path may combine individual and joint generation. The joint probability is the sum over all such generation paths. Exactly as in the case of edit distance, a corpus of observation pairs may be used to learn an improved model. In this way product FGMs may be used to learn a similarity function for objects more complex then strings, with generation patterns beyond simple left to right growth. Finally we remark that algorithms 4 and 5 are easily generalized to FGM products – including the idea of reducing space complexity by finding a bound on backward α dependencies and forward β dependencies.

5 Context Sensitive Edit Distance

The costs employed in the edit distance computation depend only on the targets of the edit operations. They are independent of the context in which the edit operation occurs. Yet in most naturally

occurring edit processes, the likelihood of an edit operation depends strongly on the context in which the operation occurs. Within the FGM framework one may strengthen our model for edit distance so that the cost of an edit operation depends on the context in which that operation is performed. Insertion and deletion costs can depend on one or more positions of trailing context in a single string, while substitution costs can depend on a joint context constructed from both strings. This modeling technique should lead to more powerful similarity functions.

Building context models within the FGM framework is possible because given discrete observations, one may construct an equivalent FGM in which only trivial observation models are used. By this we mean models which assign probability 1 to a single alphabet member. To see this, imagine replacing a discrete observation model with a simple FGM which is a mixture over the alphabet of these trivial models. This transformation in effect moves all of the work to the choice models, and during reestimation it is easy to see that trivial models are stationary. The significance of the transformation is that after passing over an edge, one has certain knowledge of the corresponding observation. States in the machine can then correspond to specific trailing contexts in the string. To specify simple stochastic edit distance, we must learn only a single χ table. With contexts, many tables are necessary – adding free parameters to the model. In principle very large context models could be built in this way, but in practice size will be limited by the amount of training data available.

6 Related Work

Finite growth models are highly related to the powerful class of hidden Markov models (HMMs). Like an FGM, a hidden Markov model (HMM) is a doubly stochastic finite state automaton. Since their introduction in the 1960's [2, 3, 1], HMMs have been widely adopted in speech recognition, handwriting recognition, and computational biology. They are typically applied in time-series settings when it is reasonable to assume that the observed data are generated or approximated well by the output of an automaton which changes state and generates output values stochastically. See [9, 7] for HMM overviews. In this section, we compare these two model classes and highlight three essential differences between them.

1. The first essential difference is that FGMs are acyclic, and therefore a given FGM can only generate objects containing a fixed number of components. In contrast, HMMs are intrinsically cyclic and therefore are capable of generating objects with arbitrary numbers of components. However, for finite observations, our discussion below reveals that any HMM can be represented as an FGM.

2. The second difference is that an FGM step can generate one object component, or jointly generate several object components. Only in discrete domains can HMMs emulate this behavior, so that in this respect FGMs are more powerful.

3. The third difference is that FGMs decouple the order in which components are generated from the order in which they occur in the observation vector. Thus, the underlying order in which an observation sequence was generated by an FGM is hidden from the observer. In contrast, HMMs are limited to generating observations in a fixed left-to-right order. Given a problem for which no single natural component ordering exists, such as string edit distance, this is an important characteristic. Provided that the observation alphabet is finite, it is possible to construct an equivalent HMM, i.e., one which imposes the same extrinsic probability

distribution, but the canonical such machine has exponentially more states than the FGM representation. Precisely characterizing the complexity of this transformation represents an interesting area for future work.

The new capabilities of items 2 and 3 above may be viewed as restrictions imposed by HMMs, which FGMs eliminate. In fact, for any given input size, it is possible to convert an HMM into an equivalent FGM for inputs of that size. (The converse is only true for finite alphabets.) This is accomplished by *unfolding* the HMM in time, resulting in an FGM DAG. The basic operations of evaluating the probability of a sequence and of performing a single step of the forward-backward algorithm have the same time and space complexity in the equivalent FGM as they do in the original HMM. In some sense this FGM is a simpler representation of the HMM because the temporal order of events has been made explicit.

With FGMs one may model new problems not easily dealt with in the HMM framework. In this paper we demonstrated that the well-known notion of string edit distance can be naturally cast into stochastic terms (following [6]) and expressed in the FGM framework. The string edit distance application has no natural expression in the HMM framework because observations must be generated in multiple orderings – and sometimes in pairs. The central benefit of our reformulation is that the symbol insertion, deletion, and substitution costs may be optimized, and we give the first algorithms for doing so. The final optimized costs are often quite different from the initial ones. It is in this sense that we *learn* them. In previous work, the costs have been stipulated after study of the problem domain. Even worse, maximally uninformative unit costs (i.e., Levenshtein distance) are sometimes stipulated when the problem domain is nontrivial. Using the methods of this paper, it is now possible to improve any educated guess, or start virtually from scratch from unit costs, to arrive at a set of locally optimal costs. The FGM framework also gives rise to an improved context sensitive notion for edit distance. At a more general level we introduced an FGM product operation, of which string edit distance is a special case. We plan to publish software which implements the algorithms of this paper for learning string edit distance costs from a collection of similar strings.

References

[1] L. E. Baum, *An inequality and associated maximization technique in statistical estimatation of probabilistic functions of Markov processes*, Inequalities, 3 (1972), pp. 1–8.

[2] L. E. Baum and J. E. Eagon, *An inequality with application to statistical estimation for probabalistic functions of a Markov process and to models for ecology*, Bull. AMS, 73 (1967), pp. 360–363.

[3] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, *A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains*, Ann. Math Stat., 41 (1970), pp. 164–171.

[4] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley, 1991.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin, *Maximum-likelihood from incomplete data via the EM algorithm*, J. Royal Statistical Society Ser. B (methodological), 39 (1977), pp. 1–38.

[6] P. A. V. Hall and G. R. Dowling, *Approximate string matching*, Computing Surveys, 12 (1980), pp. 381–402.

[7] X. D. Huang, Y. Ariki, and M. A. Jack, *Hidden Markov Models for Speech Recognition*, Edinburgh University Press, 1990.

[8] X. D. Huang and M. A. Jack, *Unified modeling of vector quantization and hidden markov models using semi-continuous hidden markov models*, in Proc. ICASSP, 1989, pp. 639–642.

[9] A. B. Poritz, *Hidden Markov models: a guided tour*, in Proc. ICASSP-88, 1988, pp. 7–13.

[10] R. A. Redner and H. F. Walker, *Mixture densities, maximum likelihood, and the EM algorithm*, SIAM Review, 26 (1984), pp. 195–239.

[11] E. S. Ristad and P. N. Yianilos, *Probability value library*, tech. rep., Princeton University, Department of Computer Science, 1994.

[12] D. Sankoff and J. B. Kruskal, *Macromolecules: The Theory and Practice of Sequence Comparison*, Addison-Wesley, 1983, 1983.