Graphical Models over Multiple Strings*

Markus Dreyer and **Jason Eisner**
Department of Computer Science / Johns Hopkins University
Baltimore, MD 21218, USA
{markus,jason}@cs.jhu.edu

Abstract

We study graphical modeling in the case of *string-valued* random variables. Whereas a weighted finite-state transducer can model the probabilistic relationship between *two* strings, we are interested in building up joint models of *three or more strings*. This is needed for inflectional paradigms in morphology, cognate modeling or language reconstruction, and multiple-string alignment. We propose a Markov Random Field in which each factor (potential function) is a weighted finite-state machine, typically a transducer that evaluates the relationship between just two of the strings. The full joint distribution is then a product of these factors. Though decoding is actually undecidable in general, we can still do efficient joint inference using approximate belief propagation; the necessary computations and messages are all finite-state. We demonstrate the methods by jointly predicting morphological forms.

1 Overview

This paper considers what happens if a graphical model's variables can range over *strings* of unbounded length, rather than over the typical *finite* domains such as booleans, words, or tags. Variables that are connected in the graphical model are related by some weighted finite-state transduction.

Graphical models have become popular in machine learning as a principled way to work with collections of interrelated random variables. Most often they are used as follows:

1. **Build:** Manually specify the n variables of interest; their domains; and the possible direct interactions among them.
2. **Train:** Train this model's parameters θ to obtain a specific joint probability distribution $p(V_1, \ldots, V_n)$ over the n variables.
3. **Infer:** Use this joint distribution to predict the values of various unobserved variables from observed ones.

*Supported by the Human Language Technology Center of Excellence at Johns Hopkins University, and by National Science Foundation grant No. 0347822 to the second author.

Note that 1. requires intuitions about the domain; 2. requires some choice of training procedure; and 3. requires a choice of exact or approximate inference algorithm.

Our graphical models over strings are natural objects to investigate. We motivate them with some natural applications in computational linguistics (section 2). We then give our formalism: a Markov Random Field whose potential functions are rational weighted languages and relations (section 3). Next, we point out that inference is in general undecidable, and explain how to do approximate inference using message-passing algorithms such as belief propagation (section 4). The messages are represented as weighted finite-state machines.

Finally, we report on some initial experiments using these methods (section 7). We use incomplete data to train a joint model of morphological paradigms, then use the trained model to complete the data by predicting unseen forms.

2 Motivation

The problem of mapping between different forms and representations of strings is ubiquitous in natural language processing and computational linguistics. This is typically done between string *pairs*, where a pronunciation is mapped to its spelling, an inflected form to its lemma, a spelling variant to its canonical spelling, or a name is transliterated from one alphabet into another.
However, many problems involve more than just two strings:

- in *morphology*, the inflected forms of a (possibly irregular) verb are naturally considered together as a whole morphological paradigm in which different forms reinforce one another;
- mapping an English word to its foreign *transliteration* may be easier when one considers the orthographic *and* phonological forms of both words;
- similar *cognates* in multiple languages are naturally described together, in orthographic or phonological representations, or both;

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 101–110,
Singapore, 6-7 August 2009.

- modern and ancestral word forms form a phylogenetic tree in *historical linguistics*;
- in *bioinformatics* and in *system combination*, multiple sequences need to be aligned in order to identify regions of similarity.

We propose a unified model for multiple strings that is suitable for all the problems mentioned above. It is robust and configurable and can make use of task-specific overlapping features. It learns from observed and unobserved, or latent, information, making it useful in supervised, semi-supervised, and unsupervised settings.

3 Formal Modeling Approach

3.1 Variables

A **Markov Random Field** (MRF) is a joint model of a set of random variables, $\mathcal{V} = \{V_1, \ldots, V_n\}$. We assume that all variables are string-valued, i.e. the value of V_i may be any string $\in \Sigma_i^*$, where Σ_i is some finite alphabet.

We may use meaningful names for the integers i, such as $V_{2\mathrm{SA}}$ for the ***2nd singular past*** form of a verb.

The assumption that *all* variables are string-valued is not crucial; it merely simplifies our presentation. It is, however, sufficient for many practical purposes, since most other discrete objects can be easily encoded as strings. For example, if V_1 is a part of speech tag, it may be encoded as a length-1 string over the finite alphabet $\Sigma_1 \stackrel{\text{def}}{=} \{\mathsf{Noun}, \mathsf{Verb}, \ldots\}$.

3.2 Factors

A Markov Random Field defines a probability for each assignment $\mathcal{A}$ of values to the variables in $\mathcal{V}$:

$$p(\mathcal{A}) \stackrel{\text{def}}{=} \frac{1}{Z} \prod_{j=1}^{m} F_j(\mathcal{A}) \qquad (1)$$

This distribution over assignments is specified by the collection of **factors** $F_j : \mathcal{A} \mapsto \mathbb{R}_{\geq 0}$. Each factor (or **potential function**) is a function that depends on only a *subset* of $\mathcal{A}$.

Fig. 1 displays an undirected **factor graph**, in which each factor is connected to the variables that it depends on. F_1, F_3, F_5 in this example are **unary** factors because each one scores the value of a single variable, while F_2, F_4, F_6 are **binary** factors.

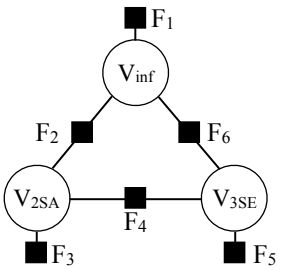

Figure 1: Example of a factor graph. Black boxes represent factors, circles represent variables (infinitive, 2nd past, and 3rd present-tense forms of the same verb; different samples from the MRF correspond to different verbs). Binary factors evaluate how well one string can be transduced into another, summing over all transducer paths (i.e., alignments, which are not observed in training).

In our setting, we will assume that each unary factor is specified by a **weighted finite-state automaton** (WFSA) whose weights fall in the semiring $(\mathbb{R}_{\geq 0}, +, \times)$. Thus the score $F_3(\ldots, V_{2SA} = x, \ldots)$ is the total weight of all paths in the F_3's WFSA that accept the string $x \in \Sigma_{2SA}^*$. Each path's weight is the product of its component arcs' weights, which are non-negative.

Similarly, we assume that each binary factor is specified by a **weighted finite-state transducer** (WFST). Such a model is essentially a generalization of stochastic edit distance (Ristad and Yianilos, 1996) in which the edit probabilities can be made sensitive to a finite summary of context.

Formally, a WFST is an automaton that resembles a weighted FSA, but it nondeterministically reads *two* strings x, y in parallel from left to right. The score of (x, y) is given by the total weight of all accepting paths in the WFST that map x to y. For example, different paths may consider various monotonic alignments of x with y, and we sum over these mutually exclusive possibilities.[1]

A factor might depend on $k > 2$ variables. This requires a k-tape **weighted finite-state machine** (WFSM), an obvious generalization where each path reads k strings in some alignment.[2]

To ensure that Z is finite in equation (1), we can require each factor to be a **"proper" WFSM**, i.e., its accepting paths have finite total weight (even if the WFSM is cyclic, with infinitely many paths).

[1] Each string is said to be on a different "tape," which has its own "read head," allowing the WFSM to maintain a separate position in each string. Thus, a path in a WFST may consume any number of characters from x before consuming the next character from y.

[2] Weighted acceptors and transducers are the cases $k = 1$ and $k = 2$, which are said to define **rational languages** and **rational relations**.

3.3 Parameters

Our probability model has trainable parameters: a vector of **feature weights** $\theta \in \mathbb{R}$. Each arc in each WFSM has a real-valued weight that depends on θ. Thus, tuning θ during training will change the arc weights, hence the path weights, the factor functions, and the whole probability distribution $p(\mathcal{A})$.

Designing the probability model includes specifying the topology and weights of each WFSM. Eisner (2002) explains how to specify and train such **parameterized WFSMs**. Typically, the weight of an arc is a simple sum like $\theta_{12} + \theta_{55} + \theta_{72}$, where θ_{12} is included on all arcs that share feature 12. However, more interesting parameterizations arise if the WFSM is constructed by operations such as transducer composition, or from a weighted regular expression.

3.4 Power of the formalism

Factored finite-state string models (1) were originally suggested by the second author, in Kempe et al. (2004). That paper showed that even in the unweighted case, such models could be used to encode relations that could not be recognized by any k-tape FSM. We offer a more linguistic example as a small puzzle. We invite the reader to specify a factored model (consisting of three FSTs as in Fig. 1) that assigns positive probability to just those triples of character strings (x, y, z) that have the form (*red_ball*, *ball_red*, *red*), (*white_house*, *house_white*, *white*), etc. This uses the auxiliary variable Z to help encode a relation between X and Y that swaps words of unbounded length. By contrast, no FSM can accomplish such unbounded swapping, even with 3 or more tapes.

Such extra power might be linguistically useful. Troublingly, however, Kempe et al. (2004) also observed that the framework is powerful enough to express computationally *undecidable* problems.[3] This implies that to work with arbitrary models, we will need approximate methods.[4] Fortunately, the graphical models community has already developed many such methods, to deal with the computational intractability (if not undecidability) of exact inference.

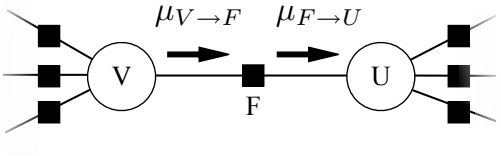

Figure 2: Illustration of messages being passed from variable to factor and factor to variable. Each message is represented by a finite-state acceptor.

4 Approximate Inference

In this paper, we focus on how **belief propagation** (BP)—a simple well-known method for approximate inference in MRFs (Bishop, 2006)—can be used in our setting. BP in its general form has not yet been widely used in the NLP community.[5] However, it is just a generalization to *arbitrary* factor graphs of the familiar forward-backward algorithm (which operates only on chain-structured factor graphs). The algorithm becomes approximate (and may not even converge) when the factor graphs have cycles. (In that case it is more properly called "loopy belief propagation.")

4.1 Belief propagation

We first sketch how BP works in general. Each variable V in the graphical model maintains a **belief** about its value, in the form of a marginal distribution $\tilde{p}_V$ over the possible values of V. The final beliefs are the output of the algorithm.

Beliefs arise from **messages** that are sent between the variables and factors along the edges of the factor graph. Variable V sends factor F a message $\mu_{V \to F}$, which is an (unnormalized) probability distribution over V's values v, computed by

$$\mu_{V \to F}(v) := \prod_{F' \in \mathcal{N}(V), F' \neq F} \mu_{F' \to V}(v) \quad (2)$$

where $\mathcal{N}$ is the set of neighbors of V in the graphical model. This message represents a consensus of V's *other* neighboring factors concerning V's value. It is how V tells F what its belief $\tilde{p}_V$ would be if F were absent. Informally, it communicates to F: *Here is what my value would be if it were up to my other neighboring factors F' to determine.*

[3] Consider a simple model with two variables and two binary factors: $p(V_1, V_2) \stackrel{\text{def}}{=} \frac{1}{Z} \cdot F_1(V_1, V_2) \cdot F_2(V_1, V_2)$. Suppose F_1 is 1 or 0 according to whether its arguments are equal. Under this model, $p(\epsilon) < 1$ iff there exists a string $x \neq \epsilon$ that can be transduced to itself by the unweighted transducer F_2. This question can be used to encode any instance of Post's Correspondence Problem, so is undecidable.

[4] Notice that the simplest approximation to cure undecidability would be to impose an arbitrary maximum on string length, so that the random variables have a finite domain, just as in most discrete graphical models.

[5] Notable exceptions are Sutton et al. (2004) for chunking and tagging, Sutton and McCallum (2004) for information extraction, Smith and Eisner (2008) for dependency parsing, and Cromierès and Kurohashi (2009) for alignment.

The factor F can then collect such incoming messages from neighboring variables and send its own message on to another neighbor U. Such a message $\mu_{F\to U}$ suggests good values for U, in the form of an (unnormalized) distribution over U's values u, computed by

$$\mu_{F\to U}(u) := \sum_{\mathcal{A}\ s.t. \mathcal{A}[U]=u} F(\mathcal{A}) \prod_{U'\in\mathcal{N}(F), U'\neq U} \mu_{U'\to F}(\mathcal{A}[U']) \tag{3}$$

where $\mathcal{A}$ is an assignment to all variables, and $\mathcal{A}[U]$ is the value of variable U in that assignment. This message represents F's prediction of U's value based on its *other* neighboring variables U'. Informally, via this message, F tells U: *Here is what I would like your value to be, based on the messages that my other neighboring variables have sent me about their values, and how I would prefer you to relate to them.*

Thus, each edge of the factor graph maintains two messages $\mu_{V\to F}, \mu_{F\to V}$. All messages are updated repeatedly, in some order, using the two equations above, until some stopping criterion is reached.[6] The *beliefs* are then computed:

$$\tilde{p}_V(v) \stackrel{\text{def}}{=} \prod_{F\in\mathcal{N}(V)} \mu_{F\to V}(v) \tag{4}$$

If variable V is *observed*, then the right-hand sides of equations (2) and (4) are modified to tell V that it *must* have the observed value v. This is done by multiplying in an extra message $\mu_{\text{obs}\to V}$ that puts probability 1 on v[7] and 0 on other values. That affects *other* messages and beliefs. The final belief at each variable estimates its *posterior* marginal under the MRF (1), *given all observations*.

4.2 Finite-state messages in BP

Both $\mu_{V\to F}$ and $\mu_{F\to V}$ are unnormalized distributions over the possible values of V—in our case, strings. A distribution over strings is naturally represented by a WFSA. Thus, belief propagation translates to our setting as follows:

- Each message is a WFSA.
- Messages are typically initialized to a one-state WFSA that accepts all strings in Σ^*, each with weight 1.[8]
- Taking a pointwise product of messages to V in equation (2) corresponds to WFSA intersection.
- If F in equation (3) is binary,[9] then there is only one U'. Then the outgoing message $\mu_{F\to U}$, a WFSA, is computed as $\text{domain}(F \circ \mu_{U'\to F})$.

  Here $\circ$ composes the factor WFST with the incoming message WFSA, yielding a WFST that gives a joint distribution over (U, U'). The domain operator projects this WFST onto the U side to obtain a WFSA, which corresponds to marginalizing to obtain a distribution over U.
- In general, F is a k-tape WFSM. Equation (3) "composes" $k-1$ of its tapes with $k-1$ incoming messages $\mu_{U'\to F}$, to construct a joint distribution over the k variables in $\mathcal{N}(F)$, then projects onto the k^{th} tape to marginalize over the $k-1$ U' variables and get a distribution over U. All this can be accomplished by the WFSM generalized composition operator $\boxtimes$ (Kempe et al., 2004).

After projecting, it is desirable to determinize the WFSA. Otherwise, the summation in (3) is only implicit—the summands remain as distinct paths in the WFSA[10]—and thus the WFSAs would get larger and larger as BP proceeds. Unfortunately, determinizing a WFSA still does not guarantee a small result. In fact it can lead to exponential blowup, or even infinite blowup.[11] Thus, in practice we recommend against determinizing the messages, which may be inherently complex. To shrink a message, it is safer to *approximate* it with a small deterministic WFSA, as discussed in the next section.

4.3 Approximation of messages

In our domain, it is possible for the finite-state messages to *grow unboundedly in size* as they flow around a cycle. After all, our messages are not just multinomial distributions over a fixed finite

[6] Preferably when the beliefs converge to some fixed point (a local minimum of the Bethe free energy). However, convergence is not guaranteed.

[7] More generally, on *all* possible observed variables.

[8] This is an (improper) uniform distribution over Σ^*. Although is *not* a proper WFSA (see section 3.2), there is an upper bound on the weights it assigns to strings. That guarantees that all the messages and beliefs computed by (2)–(4) will be proper FSMs, provided that all the factors are proper WFSMs.

[9] If it is unary, (3) trivially reduces to $\mu_{F\to U} = F$.

[10] The usual implementation of projection does not change the topology of the WFST, but only deletes the U' part of its arc labels. Thus, multiple paths that accept the same value of U remain distinct according to the distinct values of U' that they were paired with before projection.

[11] If there is no deterministic equivalent (Mohri, 1997).

set. They are distributions over the infinite set Σ^*. A WFSA represents this in finite space, but more complex distributions require bigger WFSAs, with more distinct states and arc weights.

Facing the same problem for distributions over the infinite set $\mathbb{R}$, Sudderth et al. (2002) simplified each message $\mu_{V \to F}$, approximating a complex Gaussian mixture by using fewer components.

We could act similarly, variationally approximating a large WFSA P with a smaller one Q. Choose a family of message approximations (such as bigram models) by specifying the topology for a (small) deterministic WFSA Q. Then choose Q's edge weights to minimize the KL divergence $\mathrm{KL}(P \,\|\, Q)$. This can be done in closed form.[12]

Another possible procedure—used in the experiments of this paper—approximates $\mu_{V \to F}$ by pruning it back to a finite set of most plausible strings.[13] Equation (2) requests an intersection of several WFSAs, e.g., $\mu_{F_1 \to V} \cap \mu_{F_2 \to V} \cap \cdots$. List all strings that appear on any of the 1000-best paths in any of these WFSAs, removing duplicates. Let $\bar{Q}$ be a uniform distribution over this combined list of plausible strings, represented as a determinized, minimized, acyclic WFSA. Now approximate the intersection of equation (2) as $((\bar{Q} \cap \mu_{F_1 \to V}) \cap \mu_{F_2 \to V}) \cap \cdots$. This is efficient to compute and has the same topology as $\bar{Q}$.

5 Training the Model Parameters

Any standard training method for MRFs will transfer naturally to our setting. In all cases we draw on Eisner (2002), who showed how to train the parameters θ of a *single* WFST, F, to (locally) maximize the joint or conditional probability of fully or partially observed training data. This involves computing the gradient of that likelihood function with respect to θ.[14]

We must generalize this to train a *product* of WFSMs. Typically, training data for an MRF (1) consists of some fully or partially observed IID samples of the joint distribution $p(V_1, \ldots V_n)$. It is well-known how to tune an MRF's parameters θ by stochastic gradient descent to locally maximize the probability of this training set, even though both the probability and its gradient are in general intractable to compute in an MRF. The gradient is a sum of quantities, one for each factor F_j. While the summand for F_j cannot be computed exactly, it can be estimated using the BP messages to F_j. Roughly speaking, the gradient for F_j is computed much as in supervised training (see above), but treating any message $\mu_{V_i \to F_j}$ as an uncertain observation of V_i—a form of noisy supervision.[15]

Our concerns about training are the same as for any MRF. First of all, BP is approximate. Kulesza and Pereira (2008) warn that its estimates of the gradient can be misleading. Second, semi-supervised training (which we will attempt below) is always difficult and prone to local optima. As in EM, a small number of supervised examples for some variable may be drowned out by many noisily reconstructed examples.

Faster and potentially more stable approaches include the **piecewise training** methods of Sutton and McCallum (2008), which train the factors independently or in small groups. In the semi-supervised case, each factor can be trained on only the supervised forms available for it. It might be useful to reweight the trained factors (cf. Smith et al. (2005)), or train the factors consecutively (cf. Fahlman and Lebiere (1990)), in a way that minimizes the loss of BP decoding on held-out data.

6 Comparison With Other Approaches

6.1 Multi-tape WFSMs

In principle, one could use a 100-tape WFSM to jointly model the 100 distinct forms of a typical Polish verb. In other words, the WFSM would describe the distribution of a random variable $\vec{V} = \langle V_1, \ldots, V_{100} \rangle$, where each V_i is a string. One would train the parameters of the WFSM on a sample of $\vec{V}$, each sample being a fully or partially observed paradigm for some Polish verb. The resulting distribution could be used to infer missing forms for these or other verbs.

[12] See Li et al. (2009, footnote 9) for a sketch of the construction, which finds locally normalized edge weights. Or if Q is large but parameterized by some compact parameter vector ϕ, so we are only allowed to control its edge weights via ϕ, then Li and Eisner (2009, section 6) explain how to minimize $\mathrm{KL}(P \,\|\, Q)$ by gradient descent. In both cases Q must be deterministic.

We remark that if a factor F were specified by a synchronous grammar rather than a WFSM, then its outgoing messages would be weighted context-free languages. Exact intersection of these is undecidable, but they too can be approximated variationally by WFSAs, with the same methods.

[13] We are also considering other ways of adaptively choosing the topology of WFSA approximations at runtime, particularly in conjunction with expectation propagation.

[14] The likelihood is usually non-convex; even when the two strings are observed (supervised training), their accepting path through the WFST may be ambiguous and unobserved.

[15] See Bishop (2006), or consult Smith and Eisner (2008) for notation close to that of this paper.

As a simple example, either a morphological generator or a morphological analyzer might need the probability that *krzyczałoby* is the neuter third-person singular conditional imperfective of *krzyczeć*, despite never having observed it in training. The model determines this probability based on other observed and hypothesized forms of *krzyczeć*, using its knowledge of how neuter third-person singular conditional imperfectives are related to these other forms in other verbs.

Unfortunately, such a 100-tape WFSM would be huge, with an astronomical number of arcs (each representing a possible 100-way edit operation). Our approach is to factor the problem into a number of (e.g.) pairwise relationships among the verb forms. Using a factored distribution has several benefits over the k-tape WFSM: (1) a smaller representation in memory, (2) a small number of parameters to learn, (3) efficient approximate computation that takes advantage of the factored structure, (4) the ability to reuse WFSAs and WFSTs previously developed for smaller problems, (5) additional modeling power.

6.2 Simpler graphical models on strings

Some previous researchers have used factored joint models of several strings. To our knowledge, they have all chosen *acyclic*, *directed* graphical models. The acyclicity meant that exact inference was at least possible for them, if not necessarily efficient. The factors in these past models have been WFSTs (though typically simpler than the ones we will use).

Many papers have used cascades of probabilistic finite-state transducers. Such a cascade may be regarded as a directed graphical model with a linear-chain structure. Pereira and Riley (1997) built a speech recognizer in this way, relating acoustic to phonetic to lexical strings. Similarly, Knight and Graehl (1997) presented a generative cascade using 4 variables and 5 factors: $p(w, e, j, k, o) \stackrel{\text{def}}{=} p(w) \cdot p(e \mid w) \cdot p(j \mid e) \cdot p(k \mid j) \cdot p(o \mid k)$ where e is an English word sequence, w its pronunciation, j a Japanese version of the pronunciation, k a katakana rendering of the Japanese pronunciation, and o an OCR-corrupted version of the katakana. Knight and Graehl used finite-state operations to perform inference at test time, observing o and recovering the most likely w, while marginalizing out e, j, and k.

Bouchard-Côté et al. (2009) reconstructed ancient word forms given modern equivalents. They used a directed graphical model, whose tree structure reflected the evolutionary development of the modern languages, and which included latent variables for historical intermediate forms that were never observed in training data. They used Gibbs sampling rather than an exact solution (possible on trees) or a variational approximation (like our BP).

Our work seeks to be general in terms of the graphical model structures used, as well as efficient through the use of BP with approximate messages. We also seek to avoid local normalization, using a globally normalized model.[16]

6.3 Unbounded objects in graphical models

We distinguish our work from "dynamic" graphical models such as Dynamic Bayesian Networks and Conditional Random Fields, where the string *brechen* would be represented by creating 7 letter-valued variables. Those methods can represent strings (or paths) of any length—but the length for each training or test string must be specified in advance, not inferred. Furthermore, it is awkward and costly to model unknown alignments, since the variables are position-specific, and any position in *brechen* could in principle align with any position in *brichst*. WFSTs are a much more natural and flexible model of string pairs.

We also distinguish our work from current nonparametric Bayesian models, which sometimes generate unbounded strings, trees, or grammars. If they generate two unbounded objects, they model their relationship by a single synchronous generation process (akin to Section 6.1), rather than by a globally normalized product of overlapping factors.

7 Experiments

To study our approach, we conducted initial experiments that reconstruct missing word forms in morphological paradigms. In *inflectional morphology*, each uninflected verb form (**lemma**) is associated with a vector of forms that are inflected for tense, person, number, etc. Some inflected forms may be observed frequently in natural text, others rarely. Two variables that are usually predictable from each other may or may not keep this relationship in the case of an irregular verb.

[16] Although we do normalize locally during piecewise training (see section 7.3).

(a) # paradigms	9,393
(b) # finite forms per paradigm	9
(c) # hidden finite forms per paradigm (avg.)	8.3
(d) # paradigms with some finite form(s) observed	2,176
(e) In (d), # of finite forms observed (avg.)	3.4

Table 1: Statistics of our training data.

Our task is to reconstruct (*generate*) specific unobserved morphological forms in a paradigm by learning from observed ones. This is a particularly interesting semisupervised scenario, because different subsets of the variables are observed on different examples.

7.1 Experimental data

We used orthographic rather than phonological forms. We extracted morphological paradigms for all 9393 German verbs in the CELEX morphological database. Each paradigm lists 5 present-tense and 4 past-tense indicative forms, as well as the verb's lemma, for a total of 10 string-valued variables.[17] In each paradigm, we removed, or hid, verb forms that occur only rarely in natural text, i.e, verb forms with a small frequency figure provided by CELEX.[18] All paradigms other than *sein* ('to be') were now incompletely observed. Table 1 gives some statistics.

7.2 Model factors and parameters

Our current MRF uses only binary factors. Each factor is a WFST that is trained to relate 2 of the 10 variables (morphological forms). Each WFST can score an aligned pair using a log-linear model that counts features in a sliding 3-character window. To score an unaligned pair, it sums over all possible alignments. Specifically, our WFST topology and parameterization follow the state-of-the-art approach to *supervised* morphology in Dreyer et al. (2008), although we dropped some of their features to speed up these early experiments.[19] We implemented and manipulated all WFSMs using the OpenFST library (Allauzen et al., 2007).

7.3 Training in the experiments

We trained θ on the incompletely observed paradigms. As suggested in section 5, we used a variant of piecewise pseudolikelihood training (Sutton and McCallum, 2008). Suppose there is a binary factor F attached to forms U and V. For any value of θ, we can define $p_{UV}(U \mid V)$ from the tiny MRF consisting only of U, V, and F. We can therefore compute the goodness $L_{UV} \stackrel{\text{def}}{=} \log p_{UV}(u_i \mid v_i) + \log_{VU}(v_i \mid u_i)$,[20] summed over all *observed* (U, V) pairs in training data. We attempted to tune θ to maximize the total L_{UV} over all U, V pairs,[21] regularized by subtracting $||\theta||^2$. Note that different factors thus enjoyed different amounts of observed training data, but training was fully supervised (except for the unobserved alignments between u_i and v_i).

7.4 Inference in the experiments

At test time, we are given each lemma (e.g. *brechen*) and all its observed (frequent) inflected forms (e.g., *brachen*, *bricht*,...), and are asked to predict the remaining (rarer) forms (e.g., *breche, brichst,* ...).

We run approximate joint inference using belief propagation.[22] We extract our output from the final beliefs: for each unseen variable V, we pre-

[17] Some pairs of forms are always identical in German, hence are treated as a single form by CELEX. We likewise use a single variable—these are the "1,3" variables in Fig. 3.

Occasionally a form is listed as UNKNOWN. We neither train nor evaluate on such forms, although the model will still predict them.

[18] The frequency figure for each word form is based on counts in the Mannheim News corpus. We hide forms with frequency < 10.

[19] We dropped their latent classes and regions as well as features that detected which characters were orthographic vowels. Also, we retained their "target language model features" only in the baseline "U" model, since elsewhere they seemed to hurt in our current training setup.

We followed Dreyer et al. (2008) in slightly pruning the space of possible alignments. We compensated by replacing their WFST, F, with the union $F \cup 10^{-12}(0.999\Sigma \times \Sigma)^*$. This ensured that the factor could still map any string to any other string (though perhaps with very low weight), guaranteeing that the intersection at the end of section 4.3 would be non-empty.

[20] The second term is omitted if V is the lemma. We do not train the model to predict the lemma since it is always observed in test data.

[21] Unfortunately, just before press time we discovered that this was not quite what we had done. A shortcut in our implementation trained $p_{UV}(U \mid V)$ and $p_{VU}(V \mid U)$ separately. This let them make different use of the (unobserved) alignments—so that even if each individually liked the pair (u, v), they might not have been able to agree on the same accepting path for it at test time. This could have slightly harmed our joint inference results, though not our baselines.

[22] To derive the update order for message passing, we take an arbitrary spanning tree over the factor graph, and let O be a list of all factors and variables that is topologically sorted according to the spanning tree, with the leaves of the tree coming first. We then discard the spanning tree. A single iteration visits all factors and variables in order of O, updating each one's messages to later variables and factors, and then visits all factors and variables in reverse order, updating each one's messages to earlier variables and factors.

dict its value to be $\text{argmax}_v \tilde{p}_V(v)$. This prediction considers the values of all other unseen variables but sums over their possibilities. This is the Bayes-optimal decoder for our scoring function, since that function reports the fraction of *individual forms* that were predicted *perfectly*.[23]

7.5 Model selection of MRF topology

It is hard to know *a priori* what the causal relationships might be in a morphological paradigm. In principle, one would like to *automatically* choose which factors to have in the MRF. Or one could start with many factors, but use methods such as those suggested in section 5 to learn that certain less useful factors should be left weak to avoid confusing loopy BP.

For our present experiments, we simply compared several fixed model topologies (Fig. 3). These were variously unconnected (U), chain graphs (C1,. . . , C4), trees (T1, T2), or loopy graphs (L1,. . . , L4). We used several factor graphs that differ only by one or two added factors and compared the results. The graphs were designed by hand; they connect some forms with similar morphological properties more or less densely.

We trained different models using the observed forms in the 9393 paradigms as training data. The first 100 paradigms were then used as development data for *model selection*:[24] we were given the answers to their hidden forms, enabling us to compare the models. The best model was then evaluated on the 9293 remaining paradigms.

7.6 Development data results

The models are compared on development data in Table 2. Among the factor graphs we evaluated, we find that L4 (see Fig. 3) performs best overall (whole-word accuracy 82.1). Note that the unconnected graph U does not perform very well (69.0), but using factor graphs with more connecting factors generally helps overall accuracy (see C1–C3). Note, however, that in some cases the additional structure hurts: The chain model C4 and the loopy model L1 perform relatively badly. The

[23] If we instead wished to maximize the fraction of entire paradigms that were predicted perfectly, then we would have approximated full MAP decoding over the paradigm (Viterbi decoding) by using max-product BP. Other loss functions (e.g., edit distance) would motivate other decoding methods.

[24] Using these paradigms was simply a quick way to avoid model selection by cross-validation. If data were really as sparse as our training setup pretends (see Table 2), then 100 complete paradigms would be too valuable to squander as mere development data.

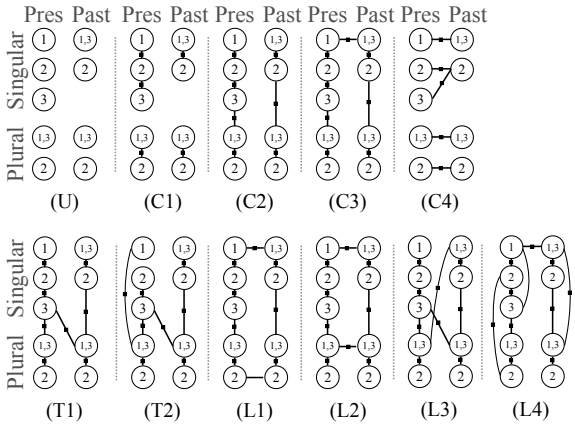

Figure 3: The graphs that we evaluate on development data. The nodes represent morphological forms, e.g. the first node in the left of each graph represents the first person singular present. Each variable is also connected to the lemma (not shown). See results in Table 2.

reason for such a performance degradation is that undertrained factors were used: The factors relating second-person to second-person forms, for example, are trained from only 8 available examples.

Non-loopy models always converge (exactly) in one iteration (see footnote 22). But even our loopy models appeared to converge in accuracy within two iterations. Only L3 and L4 required the second iteration, which made tiny improvements.

7.7 Test data results

Based on the development results, we selected model L4 and tested on the remaining 9293 paradigms.

We regard the unconnected model U as a baseline to improve upon. We also tried a rather different baseline as in (Dreyer et al., 2008). We trained the machine translation toolkit Moses (Koehn et al., 2007) to translate groups of letters rather than groups of words ("phrases"). For each form f to be predicted, we trained a Moses model on all supervised form pairs (l, f) available in the data, to learn a prediction for the form given the lemma l. The M,3 condition restricted Moses use "phrases" no longer than 3 letters, comparable to our own trigram-based factors (see section 7.2). M,15 could use up to 15 letters.

Again, our novel L4 model far outperformed the others overall. Breaking the results down by form, we find that this advantage mainly comes from the 3 forms with the fewest observed training examples (Table 3, first 3 rows). The M and U models are barely able to predict these forms at all from the lemma, but L4 can predict them bet-

Unconn.	Chains				Trees		Loops			
U	C1	C2	C3	C4	T1	T2	L1	L2	L3	L4
69.0	72.9	73.4	74.8	65.2	78.1	78.7	62.3	79.6	78.9	**82.1**

Table 2: Whole-word accuracies of the different models in reconstructing the missing forms in morphological paradigms, here on 100 verbs (development data). The names refer to the graphs in Fig. 3. We selected L4 as final model (Table 3).

Form	# obs.	M,3	M,15	U	L4
2.Sg.Pa.	4	0.0	0.2	0.8	**69.7**
2.Pl.Pa.	9	0.9	1.1	1.4	**45.6**
2.Sg.Pr.	166	49.4	62.6	74.7	**90.5**
1.Sg.Pr.	285	**99.6**	98.8	99.3	97.2
1,3.Pl.Pa.	673	46.5	**78.3**	75.0	75.6
1,3.Sg.Pa.	1124	65.0	**88.8**	84.0	74.8
2.Pl.Pr.	1274	98.3	**99.2**	99.0	96.4
3.Sg.Pr.	1410	91.0	**95.9**	95.2	88.2
1,3.Pl.Pr.	1688	**99.8**	98.9	**99.8**	98.0
All	6633	59.2	67.3	68.0	**81.2**

Table 3: Whole-word accuracies on the missing forms from 9293 test paradigms. The Moses baselines and our unconnected model (U) predict each form separately from the lemma, which is always observed. L4 uses all observations jointly, running belief propagation for decoding. Moses,15 memorizes phrases of length up to 15, all other models use max length 3. The table is sorted by the column "# obs.", which reports the numbers of observations for a given form.

ter by exploiting other observed or latent forms. By contrast, well-trained forms were already easy enough for the M and U models that L4 had little new to offer and in fact suffered from its approximate training and/or inference.

Leaving aside the comparisons, it was useful to confirm that loopy BP could be used in this setting at all. 8014 of the 9293 test paradigms had ≤ 2 observed forms (in addition to the lemma) but ≥ 7 missing forms. One might have expected that loopy BP would have failed to converge, or converged to the wrong thing. Nonetheless, it achieved quite respectable success at exactly predicting various inflected forms.

For the curious, Table 4 shows accuracies grouped by different categories of paradigms, where the category is determined by the number of missing forms to predict. Most paradigms fall in the category where 7 to 9 forms are missing, so the accuracies in that line are similar to the overall accuracies in Table 3.

# missing	# paradig.	M,3	M,15	U	L4
1–3	205	20.3	20.8	26.8	**74.4**
4–6	1037	44.2	50.5	52.7	**82.8**
7–9	8014	60.6	68.8	69.4	**81.1**

Table 4: Accuracy on test data, reported separately for paradigms in which 1–3, 4–6, or 7–9 forms are missing. Missing words have CELEX frequency count < 10; these are the ones to predict. (The numbers in col. 2 add up to 9256, not 9293, since some paradigms are incomplete in CELEX to begin with, with no forms to be removed or evaluated.)

8 Conclusions

We have proposed that one can jointly model several multiple strings by using Markov Random Fields. We described this formally as an undirected graphical model with string-valued variables and whose factors (potential functions) are defined by weighted finite-state transducers. Each factor evaluates some subset of the strings.

Approximate inference can be done by loopy belief propagation. The messages take the form of weighted finite-state acceptors, and are constructed by standard operations. We explained why the messages might become large, and gave methods for approximating them with smaller messages. We also discussed training methods.

We presented some pilot experiments on the task of jointly predicting multiple missing verb forms in morphological paradigms. The factors were simplified versions of statistical finite-state models for supervised morphology. Our MRF for this task might be used not only to conjugate verbs (e.g., in MT), but to guide further learning of morphology—either active learning from a human or semi-supervised learning from the distributional properties of a raw text corpus.

Our modeling approach is potentially applicable to a wide range of other tasks, including transliteration, phonology, cognate modeling, multiple-sequence alignment and system combination.

Our work ties into a broader vision of using algorithms like belief propagation to coordinate the work of several NLP models and algorithms. Each individual factor considers some portion of a joint problem, using classical statistical NLP methods (weighted grammars, transducers, dynamic programming). The factors coordinate their work by passing marginal probabilities. Smith and Eisner (2008) reported complementary work in this vein.

References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *Proc. of CIAA*, volume 4783 of *Lecture Notes in Computer Science*, pages 11–23.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

Alexandre Bouchard-Côté, Thomas L. Griffiths, and Dan Klein. 2009. Improved reconstruction of protolanguage word forms. In *Proc. of HLT-NAACL*, pages 65–73, Boulder, Colorado, June. Association for Computational Linguistics.

Fabien Cromierès and Sadao Kurohashi. 2009. An alignment algorithm using belief propagation and a structure-based distortion model. In *Proc. of EACL*, pages 166–174, Athens, Greece, March. Association for Computational Linguistics.

Markus Dreyer, Jason Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *Proc. of EMNLP*, Honolulu, Hawaii, October.

Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proc. of ACL*, pages 1–8, Philadelphia, July.

Scott E. Fahlman and Christian Lebiere. 1990. The cascade-correlation learning architecture. Technical Report CMU-CS-90-100, School of Computer Science, Carnegie Mellon University.

André Kempe, Jean-Marc Champarnaud, and Jason Eisner. 2004. A note on join and auto-intersection of n-ary rational relations. In Loek Cleophas and Bruce Watson, editors, *Proceedings of the Eindhoven FASTAR Days (Computer Science Technical Report 04-40)*. Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Netherlands.

Kevin Knight and Jonathan Graehl. 1997. Machine transliteration. In *Proc. of ACL*, pages 128–135.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL, Companion Volume*, pages 177–180, Prague, Czech Republic, June. Association for Computational Linguistics.

Alex Kulesza and Fernando Pereira. 2008. Structured learning with approximate inference. In *Proc. of NIPS*.

Zhifei Li and Jason Eisner. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proc. of EMNLP*.

Zhifei Li, Jason Eisner, and Sanjeev Khudanpur. 2009. Variational decoding for statistical machine translation. In *Proc. of ACL*.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2).

Fernando C. N. Pereira and Michael Riley. 1997. Speech recognition by composition of weighted finite automata. In Emmanuel Roche and Yves Schabes, editors, *Finite-State Language Processing*. MIT Press, Cambridge, MA.

Eric Sven Ristad and Peter N. Yianilos. 1996. Learning string edit distance. Technical Report CS-TR-532-96, Princeton University, Department of Computer Science, October.

David Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proc. of EMNLP*.

Andrew Smith, Trevor Cohn, and Miles Osborne. 2005. Logarithmic opinion pools for conditional random fields. In *Proc. of ACL*, pages 18–25, June.

Erik B. Sudderth, Alexander T. Ihler, Er T. Ihler, William T. Freeman, and Alan S. Willsky. 2002. Nonparametric belief propagation. In *Proc. of CVPR*, pages 605–612.

Charles Sutton and Andrew McCallum. 2004. Collective segmentation and labeling of distant entities in information extraction. In *ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields*.

Charles Sutton and Andrew McCallum. 2008. Piecewise training for structured prediction. *Machine Learning*. In submission.

Charles Sutton, Khashayar Rohanimanesh, and Andrew McCallum. 2004. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *Proc. of ICML*.