Finding Cognate Groups using Phylogenies

David Hall and **Dan Klein**
Computer Science Division
University of California, Berkeley
{dlwh,klein}@cs.berkeley.edu

Abstract

A central problem in historical linguistics is the identification of historically related *cognate* words. We present a generative phylogenetic model for automatically inducing cognate group structure from unaligned word lists. Our model represents the process of transformation and transmission from ancestor word to daughter word, as well as the alignment between the words lists of the observed languages. We also present a novel method for simplifying complex weighted automata created during inference to counteract the otherwise exponential growth of message sizes. On the task of identifying cognates in a dataset of Romance words, our model significantly outperforms a baseline approach, increasing accuracy by as much as 80%. Finally, we demonstrate that our automatically induced groups can be used to successfully reconstruct ancestral words.

1 Introduction

A crowning achievement of historical linguistics is the comparative method (Ohala, 1993), wherein linguists use word similarity to elucidate the hidden phonological and morphological processes which govern historical descent. The comparative method requires reasoning about three important hidden variables: the overall *phylogenetic guide tree* among languages, the *evolutionary parameters* of the ambient changes at each branch, and the *cognate group structure* that specifies which words share common ancestors.

All three of these variables interact and inform each other, and so historical linguists often consider them jointly. However, linguists are currently required to make qualitative judgments regarding the relative likelihood of certain sound changes, cognate groups, and so on. Several recent statistical methods have been introduced to provide increased quantitative backing to the comparative method (Oakes, 2000; Bouchard-Côté et al., 2007; Bouchard-Côté et al., 2009); others have modeled the spread of language changes and speciation (Ringe et al., 2002; Daumé III and Campbell, 2007; Daumé III, 2009; Nerbonne, 2010). These automated methods, while providing robustness and scale in the induction of ancestral word forms and evolutionary parameters, assume that cognate groups are already known. In this work, we address this limitation, presenting a model in which cognate groups can be discovered automatically.

Finding cognate groups is not an easy task, because underlying morphological and phonological changes can obscure relationships between words, especially for distant cognates, where simple string overlap is an inadequate measure of similarity. Indeed, a standard string similarity metric like Levenshtein distance can lead to false positives. Consider the often cited example of Greek /maːti/ and Malay /mata/, both meaning "eye" (Bloomfield, 1938). If we were to rely on Levenshtein distance, these words would seem to be a highly attractive match as cognates: they are nearly identical, essentially differing in only a single character. However, no linguist would posit that these two words are related. To correctly learn that they are not related, linguists typically rely on two kinds of evidence. First, because sound change is largely regular, we would need to commonly see /i/ in Greek wherever we see /a/ in Malay (Ross, 1950). Second, we should look at languages closely related to Greek and Malay, to see if similar patterns hold there, too.

Some authors have attempted to automatically detect cognate words (Mann and Yarowsky, 2001; Lowe and Mazaudon, 1994; Oakes, 2000; Kondrak, 2001; Mulloni, 2007), but these methods

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1030–1039,
Uppsala, Sweden, 11-16 July 2010.

typically work on language pairs rather than on larger language families. To fully automate the comparative method, it is necessary to consider multiple languages, and to do so in a model which couples cognate detection with similarity learning.

In this paper, we present a new generative model for the automatic induction of cognate groups given only (1) a known family tree of languages and (2) word lists from those languages. A prior on word survival generates a number of cognate groups and decides which groups are attested in each modern language. An evolutionary model captures how each word is generated from its parent word. Finally, an alignment model maps the flat word lists to cognate groups. Inference requires a combination of message-passing in the evolutionary model and iterative bipartite graph matching in the alignment model.

In the message-passing phase, our model encodes distributions over strings as weighted finite state automata (Mohri, 2009). Weighted automata have been successfully applied to speech processing (Mohri et al., 1996) and more recently to morphology (Dreyer and Eisner, 2009). Here, we present a new method for automatically compressing our message automata in a way that can take into account prior information about the expected outcome of inference.

In this paper, we focus on a transcribed word list of 583 cognate sets from three Romance languages (Portuguese, Italian and Spanish), as well as their common ancestor Latin (Bouchard-Côté et al., 2007). We consider both the case where we know that all cognate groups have a surface form in all languages, and where we do not know that. On the former, easier task we achieve identification accuracies of 90.6%. On the latter task, we achieve F1 scores of 73.6%. Both substantially beat baseline performance.

2 Model

In this section, we describe a new generative model for vocabulary lists in multiple related languages given the phylogenetic relationship between the languages (their family tree). The generative process factors into three subprocesses: survival, evolution, and alignment, as shown in Figure 1(a). Survival dictates, for each cognate group, which languages have words in that group. Evolution describes the process by which daughter words are transformed from their parent word. Finally, alignment describes the "scrambling" of the word lists into a flat order that hides their lineage. We present each subprocess in detail in the following subsections.

2.1 Survival

First, we choose a number G of ancestral cognate groups from a geometric distribution. For each cognate group g, our generative process walks down the tree. At each branch, the word may either survive or die. This process is modeled in a "death tree" with a Bernoulli random variable $S_{\ell g}$ for each language ℓ and cognate group g specifying whether or not the word died before reaching that language. Death at any node in the tree causes all of that node's descendants to also be dead. This process captures the intuition that cognate words are more likely to be found clustered in sibling languages than scattered across unrelated languages.

2.2 Evolution

Once we know which languages will have an attested word and which will not, we generate the actual word forms. The evolution component of the model generates words according to a branch-specific transformation from a node's immediate ancestor. Figure 1(a) graphically describes our generative model for three Romance languages: Italian, Portuguese, and Spanish.[1] In each cognate group, each word W_ℓ is generated from its parent according to a conditional distribution with parameter φ_ℓ, which is specific to that edge in the tree, but shared between all cognate groups.

In this paper, each φ_ℓ takes the form of a parameterized edit distance similar to the standard Levenshtein distance. Richer models – such as the ones in Bouchard-Côté et al. (2007) – could instead be used, although with an increased inferential cost. The edit transducers are represented schematically in Figure 1(b). Characters x and y are arbitrary phonemes, and $\sigma(x, y)$ represents the cost of substituting x with y. ε represents the empty phoneme and is used as shorthand for insertion and deletion, which have parameters η and δ, respectively.

As an example, see the illustration in Figure 1(c). Here, the Italian word /fwɔko/ ("fire") is generated from its parent form /fokus/ ("hearth")

[1] Though we have data for Latin, we treat it as unobserved to represent the more common case where the ancestral language is unattested; we also evaluate our system using the Latin data.

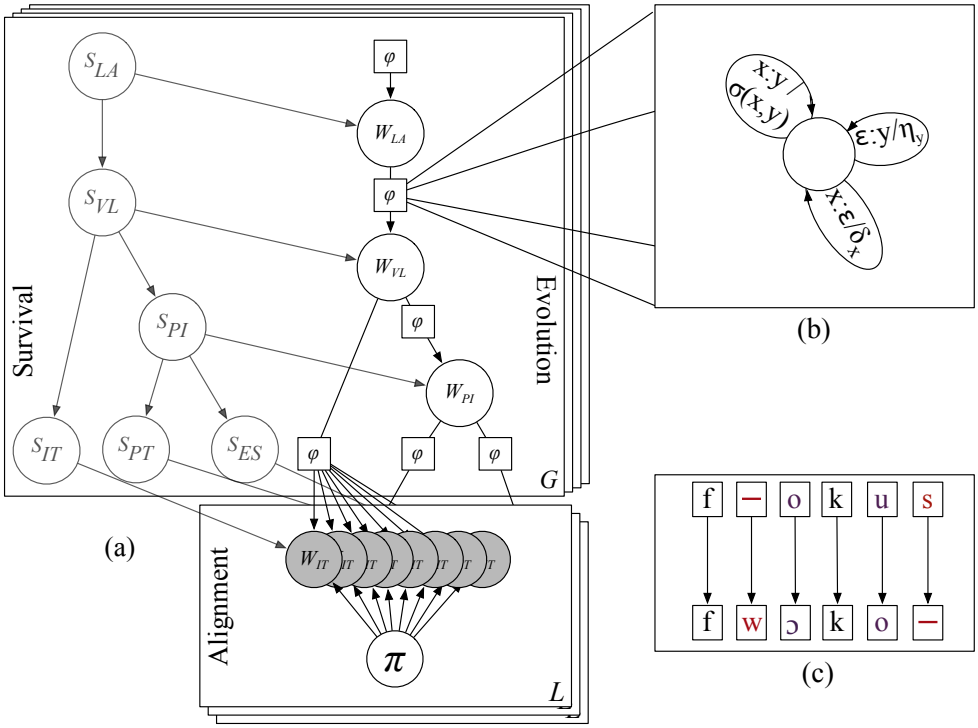

Figure 1: (a) The process by which cognate words are generated. Here, we show the derivation of Romance language words W_ℓ from their respective Latin ancestor, parameterized by transformations φ_ℓ and survival variables S_ℓ. Languages shown are Latin (LA), Vulgar Latin (VL), Proto-Iberian (PI), Italian (IT), Portuguese (PT), and Spanish (ES). Note that only modern language words are observed (shaded). (b) The class of parameterized edit distances used in this paper. Each pair of phonemes has a weight σ for deletion, and each phoneme has weights η and δ for insertion and deletion respectively. (c) A possible alignment produced by an edit distance between the Latin word *focus* ("hearth") and the Italian word *fuoco* ("fire").

by a series of edits: two matches, two substitutions (/u/→ /o/, and /o/→/ɔ/), one insertion (w) and one deletion (/s/). The probability of each individual edit is determined by φ. Note that the marginal probability of a specific Italian word conditioned on its Vulgar Latin parent is the sum over all possible derivations that generate it.

2.3 Alignment

Finally, at the leaves of the trees are the observed words. (We take non-leaf nodes to be unobserved.) Here, we make the simplifying assumption that in any language there is at most one word per language per cognate group. Because the assignments of words to cognates is unknown, we specify an unknown alignment parameter π_ℓ for each modern language which is an alignment of cognate groups to entries in the word list. In the case that every cognate group has a word in each language, each π_ℓ is a permutation. In the more general case that some cognate groups do not have words from all languages, this mapping is injective from words to cognate groups. From a generative perspective, π_ℓ generates observed positions of the words in some vocabulary list.

In this paper, our task is primarily to learn the alignment variables π_ℓ. All other hidden variables are auxiliary and are to be marginalized to the greatest extent possible.

3 Inference of Cognate Assignments

In this section, we discuss the inference method for determining cognate assignments under fixed parameters φ. We are given a set of languages and a list of words in each language, and our objective is to determine which words are cognate with each other. Because the parameters π_ℓ are either permutations or injections, the inference task is reduced to finding an alignment π of the respective word lists to maximize the log probability of the observed words.

$$\pi^* = \arg\max_{\pi} \sum_{g} \log p(w_{(\ell,\pi_\ell(g))} | \varphi, \pi, \mathbf{w}_{-\ell})$$

$w_{(\ell,\pi_\ell(g))}$ is the word in language ℓ that π_ℓ has assigned to cognate group g. Maximizing this quantity directly is intractable, and so instead we use a coordinate ascent algorithm to iteratively

maximize the alignment corresponding to a single language ℓ while holding the others fixed:

$$\pi_\ell^* = \arg\max_{\pi_\ell} \sum_g \log p(w_{(\ell,\pi_\ell(g))}|\varphi, \pi_{-\ell}, \pi_\ell, \mathbf{w}_{-\ell})$$

Each iteration is then actually an instance of bipartite graph matching, with the words in one language one set of nodes, and the current cognate groups in the other languages the other set of nodes. The edge affinities $\mathbf{aff}$ between these nodes are the conditional probabilities of each word w_ℓ belonging to each cognate group g:

$$\mathbf{aff}(w_\ell, g) = p(w_\ell|\mathbf{w}_{-\ell,\pi_{-\ell}(g)}, \varphi, \pi_{-\ell})$$

To compute these affinities, we perform inference in each tree to calculate the marginal distribution of the words from the language ℓ. For the marginals, we use an analog of the forward/backward algorithm. In the upward pass, we send messages from the leaves of the tree toward the root. For observed leaf nodes W_d, we have:

$$\mu_{d\to a}(w_a) = p(W_d = w_d|w_a, \varphi_d)$$

and for interior nodes W_i:

$$\mu_{i\to a}(w_a) = \sum_{w_i} p(w_i|w_a, \varphi_i) \prod_{d\in \text{child}(w_i)} \mu_{d\to i}(w_i) \quad (1)$$

In the downward pass (toward the language ℓ), we sum over ancestral words W_a:

$$\mu_{a\to d}(w_d) = \sum_{w_a} p(w_d|w_a, \varphi_d)\mu_{a'\to a}(w_a) \prod_{\substack{d'\in \text{child}(w_a)\\ d'\neq d}} \mu_{d'\to a}(w_a)$$

where a' is the ancestor of a. Computing these messages gives a posterior marginal distribution $\mu_\ell(w_\ell) = p(w_\ell|\mathbf{w}_{-\ell,\pi_{-\ell}(g)}, \varphi, \pi_{-\ell})$, which is precisely the affinity score we need for the bipartite matching. We then use the Hungarian algorithm (Kuhn, 1955) to find the optimal assignment for the bipartite matching problem.

One important final note is initialization. In our early experiments we found that choosing a random starting configuration unsurprisingly led to rather poor local optima. Instead, we started with empty trees, and added in one language per iteration until all languages were added, and then continued iterations on the full tree.

4 Learning

So far we have only addressed searching for Viterbi alignments π under fixed parameters. In practice, it is important to estimate better parametric edit distances φ_ℓ and survival variables S_ℓ. To motivate the need for good transducers, consider the example of English "day" */deɪ/* and Latin "diēs" */diːɛs/*, both with the same meaning. Surprisingly, these words are in no way related, with English "day" probably coming from a verb meaning "to burn" (OED, 1989). However, a naively constructed edit distance, which for example might penalize vowel substitutions lightly, would fail to learn that Latin words that are borrowed into English would not undergo the sound change */ɪ/→/eɪ/*. Therefore, our model must learn not only which sound changes are plausible (e.g. vowels turning into other vowels is more common than vowels turning into consonants), but which changes are appropriate for a given language.[2]

At a high level, our learning algorithm is much like Expectation Maximization with hard assignments: after we update the alignment variables π and thus form new potential cognate sets, we re-estimate our model's parameters to maximize the likelihood of those assignments.[3] The parameters can be learned through standard maximum likelihood estimation, which we detail in this section.

Because we enforce that a word in language d must be dead if its parent word in language a is dead, we just need to learn the conditional probabilities $p(S_d = \text{dead}|S_a = \text{alive})$. Given fixed assignments π, the maximum likelihood estimate can be found by counting the number of "deaths" that occurred between a child and a live parent, applying smoothing – we found adding 0.5 to be reasonable – and dividing by the total number of live parents.

For the transducers φ, we learn parameterized edit distances that model the probabilities of different sound changes. For each φ_ℓ we fit a non-uniform substitution, insertion, and deletion matrix $\sigma(x, y)$. These edit distances define a condi-

[2] We note two further difficulties: our model does not handle "borrowings," which would be necessary to capture a significant portion of English vocabulary; nor can it seamlessly handle words that are inherited later in the evolution of language than others. For instance, French borrowed words from its parent language Latin during the Renaissance and the Enlightenment that have not undergone the same changes as words that evolved "naturally" from Latin. See Bloomfield (1938). Handling these cases is a direction for future research.

[3] Strictly, we can cast this problem in a variational framework similar to mean field where we iteratively maximize parameters to minimize a KL-divergence. We omit details for clarity.

tional exponential family distribution when conditioned on an ancestral word. That is, for any fixed w_a:

$$\sum_{w_d} p(w_d|w_a,\sigma) = \sum_{w_d} \sum_{\substack{z\in \\ \text{align}(w_a,w_d)}} \text{score}(z;\sigma)$$
$$= \sum_{w_d} \sum_{\substack{z\in \\ \text{align}(w_a,w_d)}} \prod_{(x,y)\in z} \sigma(x,y) = 1$$

where $\text{align}(w_a, w_d)$ is the set of possible alignments between the phonemes in words w_a and w_d.

We are seeking the maximum likelihood estimate of each φ, given fixed alignments π:

$$\hat{\varphi}_\ell = \arg\max_{\varphi_\ell} p(\mathbf{w}|\varphi,\pi)$$

To find this maximizer for any given π_ℓ, we need to find a marginal distribution over the edges connecting any two languages a and d. With this distribution, we calculate the expected "alignment unigrams." That is, for each pair of phonemes x and y (or empty phoneme ε), we need to find the quantity:

$$E_{p(w_a,w_d)}[\#(x,y;z)] =$$
$$\sum_{w_a,w_d} \sum_{\substack{z\in \\ \text{align}(w_a,w_d)}} \#(x,y;z)p(z|w_a,w_d)p(w_a,w_d)$$

where we denote $\#(x,y;z)$ to be the number of times the pair of phonemes (x,y) are aligned in alignment z. The exact method for computing these counts is to use an expectation semiring (Eisner, 2001).

Given the expected counts, we now need to normalize them to ensure that the transducer represents a conditional probability distribution (Eisner, 2002; Oncina and Sebban, 2006). We have that, for each phoneme x in the ancestor language:

$$\eta_y = \frac{E[\#(\varepsilon,y;z)]}{E[\#(\cdot,\cdot;z)]}$$
$$\sigma(x,y) = (1-\sum_{y'}\eta_{y'})\frac{E[\#(x,y;z)]}{E[\#(x,\cdot;z)]}$$
$$\delta_x = (1-\sum_{y'}\eta_{y'})\frac{E[\#(x,\varepsilon;z)]}{E[\#(x,\cdot;z)]}$$

Here, we have $\#(\cdot,\cdot;z) = \sum_{x,y}\#(x,y;z)$ and $\#(x,\cdot;z) = \sum_y \#(x,y;z)$. The $(1-\sum_{y'}\eta_{y'})$ term ensure that for any ancestral phoneme x, $\sum_y \eta_y + \sum_y \sigma(x,y) + \delta_x = 1$. These equations ensure that the three transition types (insertion, substitution/match, deletion) are normalized for each ancestral phoneme.

5 Transducers and Automata

In our model, it is not just the edit distances that are finite state machines. Indeed, the words themselves are string-valued random variables that have, in principle, an infinite domain. To represent distributions and messages over these variables, we chose weighted finite state automata, which can compactly represent functions over strings. Unfortunately, while initially compact, these automata become unwieldy during inference, and so approximations must be used (Dreyer and Eisner, 2009). In this section, we summarize the standard algorithms and representations used for weighted finite state transducers. For more detailed treatment of the general transducer operations, we direct readers to Mohri (2009).

A weighted automaton (resp. transducer) encodes a function over strings (resp. pairs of strings) as weighted paths through a directed graph. Each edge in the graph has a real-valued weight[4] and a label, which is a single phoneme in some alphabet Σ or the empty phoneme ε (resp. pair of labels in some alphabet $\Sigma\times\Delta$). The weight of a string is then the sum of all paths through the graph that accept that string.

For our purposes, we are concerned with three fundamental operations on weighted transducers. The first is computing the sum of all paths through a transducer, which corresponds to computing the partition function of a distribution over strings. This operation can be performed in worst-case cubic time (using a generalization of the Floyd-Warshall algorithm). For acyclic or feed-forward transducers, this time can be improved dramatically by using a generalization of Djisktra's algorithm or other related algorithms (Mohri, 2009).

The second operation is the composition of two transducers. Intuitively, composition creates a new transducer that takes the output from the first transducer, processes it through the second transducer, and then returns the output of the second transducer. That is, consider two transducers T_1 and T_2. T_1 has input alphabet Σ and output alphabet Δ, while T_2 has input alphabet Δ and output alphabet Ω. The composition $T_1 \circ T_2$ returns a new transducer over Σ and Ω such that $(T_1 \circ T_2)(x,y) = \sum_u T_1(x,u)\cdot T_2(u,y)$. In this paper, we use composition for marginalization and factor products. Given a factor $f_1(x,u;T_1)$ and an-

[4]The weights can be anything that form a semiring, but for the sake of exposition we specialize to real-valued weights.

other factor $f_2(u, y; T_2)$, composition corresponds to the operation $\psi(x, y) = \sum_u f_1(x, u) f_2(u, y)$. For two messages $\mu_1(w)$ and $\mu_2(w)$, the same algorithm can be used to find the product $\mu(w) = \mu_1(w)\mu_2(w)$.

The third operation is transducer minimization. Transducer composition produces $O(nm)$ states, where n and m are the number of states in each transducer. Repeated compositions compound the problem: iterated composition of k transducers produces $O(n^k)$ states. Minimization alleviates this problem by collapsing indistinguishable states into a single state. Unfortunately, minimization does not always collapse enough states. In the next section we discuss approaches to "lossy" minimization that produce automata that are not exactly the same but are much smaller.

6 Message Approximation

Recall that in inference, when summing out interior nodes w_i we calculated the product over incoming messages $\mu_{d \to i}(w_i)$ (Equation 1), and that these products are calculated using transducer composition. Unfortunately, the maximal number of states in a message is exponential in the number of words in the cognate group. Minimization can only help so much: in order for two states to be collapsed, the distribution over transitions from those states must be indistinguishable. In practice, for the automata generated in our model, minimization removes at most half the states, which is not sufficient to counteract the exponential growth. Thus, we need to find a way to approximate a message $\mu(w)$ using a simpler automata $\tilde{\mu}(w; \theta)$ taken from a restricted class parameterized by θ.

In the context of transducers, previous authors have focused on a combination of n-best lists and unigram back-off models (Dreyer and Eisner, 2009), a schematic diagram of which is in Figure 2(d). For their problem, n-best lists are sensible: their nodes' local potentials already focus messages on a small number of hypotheses. In our setting, however, n-best lists are problematic; early experiments showed that a 10,000-best list for a typical message only accounts for 50% of message log perplexity. That is, the posterior marginals in our model are (at least initially) fairly flat.

An alternative approach might be to simply treat messages as unnormalized probability distributions, and to minimize the KL divergence between some approximating message $\tilde{\mu}(w)$ and the true message $\mu(w)$. However, messages are not always probability distributions and – because the number of possible strings is in principle infinite – they need not sum to a finite number.[5] Instead, we propose to minimize the KL divergence between the "expected" marginal distribution and the approximated "expected" marginal distribution:

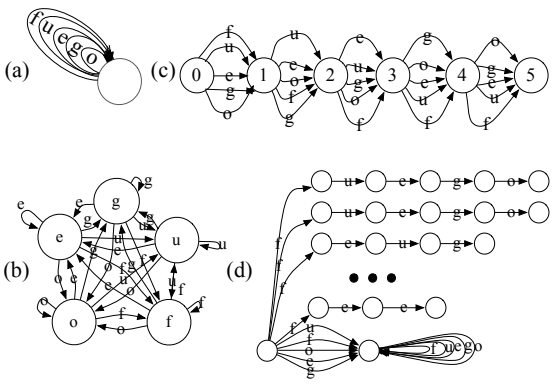

Figure 2: Various topologies for approximating topologies: (a) a unigram model, (b) a bigram model, (c) the anchored unigram model, and (d) the n-best plus backoff model used in Dreyer and Eisner (2009). In (c) and (d), the relative height of arcs is meant to convey approximate probabilities.

$$\begin{aligned}
\hat{\theta} &= \arg\min_{\theta} D_{KL}(\tau(w)\mu(w) || \tau(w)\tilde{\mu}(w; \theta)) \\
&= \arg\min_{\theta} \sum_w \tau(w)\mu(w) \log \frac{\tau(w)\mu(w)}{\tau(w)\tilde{\mu}(w; \theta)} \\
&= \arg\min_{\theta} \sum_w \tau(w)\mu(w) \log \frac{\mu(w)}{\tilde{\mu}(w; \theta)}
\end{aligned} \tag{2}$$

where τ is a term acting as a surrogate for the posterior distribution over w without the information from μ. That is, we seek to approximate μ not on its own, but as it functions in an environment representing its final context. For example, if $\mu(w)$ is a backward message, τ could be a stand-in for a forward probability.[6]

In this paper, $\mu(w)$ is a complex automaton with potentially many states, $\tilde{\mu}(w; \theta)$ is a simple parametric automaton with forms that we discuss below, and $\tau(w)$ is an arbitrary (but hopefully fairly simple) automaton. The actual method we use is

[5] As an extreme example, suppose we have observed that $W_d = w_d$ and that $p(W_d = w_d | w_a) = 1$ for all ancestral words w_a. Then, clearly $\sum_{w_d} \mu(w_d) = \sum_{w_d} \sum p(W_d = w_d | w_a) = \infty$ whenever there are an infinite number of possible ancestral strings w_a.

[6] This approach is reminiscent of Expectation Propagation (Minka, 2001).

as follows. Given a deterministic prior automaton τ, and a deterministic automaton topology $\tilde{\mu}^*$, we create the composed unweighted automaton $\tau \circ \tilde{\mu}^*$, and calculate arc transitions weights to minimize the KL divergence between that composed transducer and $\tau \circ \mu$. The procedure for calculating these statistics is described in Li and Eisner (2009), which amounts to using an expectation semiring (Eisner, 2001) to compute expected transitions in $\tau \circ \tilde{\mu}^*$ under the probability distribution $\tau \circ \mu$.

From there, we need to create the automaton $\tau^{-1} \circ \tau \circ \tilde{\mu}$. That is, we need to divide out the influence of $\tau(w)$. Since we know the topology and arc weights for τ ahead of time, this is often as simple as dividing arc weights in $\tau \circ \tilde{\mu}$ by the corresponding arc weight in $\tau(w)$. For example, if τ encodes a geometric distribution over word lengths and a uniform distribution over phonemes (that is, $\tau(w) \propto p^{|w|}$), then computing $\tilde{\mu}$ is as simple as dividing each arc in $\tau \circ \tilde{\mu}$ by p.[7]

There are a number of choices for τ. One is a hard maximum on the length of words. Another is to choose $\tau(w)$ to be a unigram language model over the language in question with a geometric probability over lengths. In our experiments, we find that $\tau(w)$ can be a geometric distribution over lengths with a uniform distribution over phonemes and still give reasonable results. This distribution captures the importance of shorter strings while still maintaining a relatively weak prior.

What remains is the selection of the topologies for the approximating message $\tilde{\mu}$. We consider three possible approximations, illustrated in Figure 2. The first is a plain unigram model, the second is a bigram model, and the third is an anchored unigram topology: a position-specific unigram model for each position up to some maximum length.

The first we consider is a standard unigram model, which is illustrated in Figure 2(a). It has $|\Sigma| + 2$ parameters: one weight σ_a for each phoneme $a \in \Sigma$, a starting weight λ, and a stopping probability ρ. $\tilde{\mu}$ then has the form:

$$\tilde{\mu}(w) = \lambda\rho \prod_{i \leq |w|} \sigma_{w_i}$$

Estimating this model involves only computing the expected count of each phoneme, along with the expected length of a word, $E[|w|]$. We then normalize the counts according to the maximum likelihood estimate, with arc weights set as:

$$\sigma_a \propto E[\#(a)]$$

Recall that these expectations can be computed using an expectation semiring.

Finally, λ can be computed by ensuring that the approximate and exact expected marginals have the same partition function. That is, with the other parameters fixed, solve:

$$\sum_w \tau(w)\tilde{\mu}(w) = \sum_w \tau(w)\mu(w)$$

which amounts to rescaling $\tilde{\mu}$ by some constant.

The second topology we consider is the bigram topology, illustrated in Figure 2(b). It is similar to the unigram topology except that, instead of a single state, we have a state for each phoneme in Σ, along with a special start state. Each state a has transitions with weights $\sigma_{b|a} = p(b|a) \propto E[\#(b|a)]$. Normalization is similar to the unigram case, except that we normalize the transitions from each state.

The final topology we consider is the positional unigram model in Figure 2(c). This topology takes positional information into account. Namely, for each position (up to some maximum position), we have a unigram model over phonemes emitted at that position, along with the probability of stopping at that position (i.e. a "sausage lattice"). Estimating the parameters of this model is similar, except that the expected counts for the phonemes in the alphabet are conditioned on their position in the string. With the expected counts for each position, we normalize each state's final and outgoing weights. In our experiments, we set the maximum length to seven more than the length of the longest observed string.

7 Experiments

We conduct three experiments. The first is a "complete data" experiment, in which we reconstitute the cognate groups from the Romance data set, where all cognate groups have words in all three languages. This task highlights the evolution and alignment models. The second is a much harder "partial data" experiment, in which we randomly prune 20% of the branches from the dataset according to the survival process described in Section 2.1. Here, only a fraction of words appear

[7] Also, we must be sure to divide each final weight in the transducer by $(1 - |\Sigma|p)$, which is the stopping probability for a geometric transducer.

in any cognate group, so this task crucially involves the survival model. The ultimate purpose of the induced cognate groups is to feed richer evolutionary models, such as full reconstruction models. Therefore, we also consider a proto-word reconstruction experiment. For this experiment, using the system of Bouchard-Côté et al. (2009), we compare the reconstructions produced from our automatic groups to those produced from gold cognate groups.

7.1 Baseline

As a novel but heuristic baseline for cognate group detection, we use an iterative bipartite matching algorithm where instead of conditional likelihoods for affinities we use Dice's coefficient, defined for sets X and Y as:

$$\text{Dice}(X, Y) = \frac{2|X \cap Y|}{|X| + |Y|} \quad (3)$$

Dice's coefficients are commonly used in bilingual detection of cognates (Kondrak, 2001; Kondrak et al., 2003). We follow prior work and use sets of bigrams within words. In our case, during bipartite matching the set X is the set of bigrams in the language being re-permuted, and Y is the union of bigrams in the other languages.

7.2 Experiment 1: Complete Data

In this experiment, we know precisely how many cognate groups there are and that every cognate group has a word in each language. While this scenario does not include all of the features of the real-world task, it represents a good test case of how well these models can perform without the non-parametric task of deciding how many clusters to use.

We scrambled the 583 cognate groups in the Romance dataset and ran each method to convergence. Besides the heuristic baseline, we tried our model-based approach using Unigrams, Bigrams and Anchored Unigrams, with and without learning the parametric edit distances. When we did not use learning, we set the parameters of the edit distance to (0, -3, -4) for matches, substitutions, and deletions/insertions, respectively. With learning enabled, transducers were initialized with those parameters.

For evaluation, we report two metrics. The first is pairwise accuracy for each pair of languages, averaged across pairs of words. The other is accuracy measured in terms of the number of correctly, completely reconstructed cognate groups.

		Pairwise Acc.	Exact Match
Heuristic			
Baseline		48.1	35.4
Model			
Transducers	Messages		
Levenshtein	Unigrams	37.2	26.2
Levenshtein	Bigrams	43.0	26.5
Levenshtein	Anch. Unigrams	68.6	56.8
Learned	Unigrams	0.1	0.0
Learned	Bigrams	38.7	11.3
Learned	Anch. Unigrams	**90.3**	**86.6**

Table 1: Accuracies for reconstructing cognate groups. Levenshtein refers to fixed parameter edit distance transducer. Learned refers to automatically learned edit distances. Pairwise Accuracy means averaged on each word pair; Exact Match refers to percentage of completely and accurately reconstructed groups. For a description of the baseline, see Section 7.1.

		Prec.	Recall	F1
Heuristic				
Baseline		49.0	43.5	46.1
Model				
Transducers	Messages			
Levenshtein	Anch. Unigrams	**86.5**	36.1	50.9
Learned	Anch. Unigrams	66.9	**82.0**	**73.6**

Table 2: Accuracies for reconstructing incomplete groups. Scores reported are precision, recall, and F1, averaged over all word pairs.

Table 1 shows the results under various configurations. As can be seen, the kind of approximation used matters immensely. In this application, positional information is important, more so than the context of the previous phoneme. Both Unigrams and Bigrams significantly under-perform the baseline, while Anchored Unigrams easily outperforms it both with and without learning.

An initially surprising result is that learning actually harms performance under the unanchored approximations. The explanation is that these topologies are not sensitive enough to context, and that the learning procedure ends up flattening the distributions. In the case of unigrams – which have the least context – learning degrades performance to chance. However, in the case of positional unigrams, learning reduces the error rate by more than two-thirds.

7.3 Experiment 2: Incomplete Data

As a more realistic scenario, we consider the case where we do not know that all cognate groups have words in all languages. To test our model, we ran-

domly pruned 20% of the branches according the survival process of our model.[8]

Because only Anchored Unigrams performed well in Experiment 1, we consider only it and the Dice's coefficient baseline. The baseline needs to be augmented to support the fact that some words may not appear in all cognate groups. To do this, we thresholded the bipartite matching process so that if the coefficient fell below some value, we started a new group for that word. We experimented on 10 values in the range (0,1) for the baseline's threshold and report on the one (0.2) that gives the best pairwise F1.

The results are in Table 2. Here again, we see that the positional unigrams perform much better than the baseline system. The learned transducers seem to sacrifice precision for the sake of increased recall. This makes sense because the default edit distance parameter settings strongly favor exact matches, while the learned transducers learn more realistic substitution and deletion matrices, at the expense of making more mistakes.

For example, the learned transducers enable our model to correctly infer that Portuguese /difemdu/, Spanish /defiendo/, and Italian /difɛndo/ are all derived from Latin /deːfendoː/ "defend." Using the simple Levenshtein transducers, on the other hand, our model keeps all three separated, because the transducers cannot know – among other things – that Portuguese /i/, Spanish /e/, and Italian /i/ are commonly substituted for one another. Unfortunately, because the transducers used cannot learn contextual rules, certain transformations can be over-applied. For instance, Spanish /nombrar/ "name" is grouped together with Portuguese /numirar/ "number" and Italian /numerare/ "number," largely because the rule Portuguese /u/ → Spanish /o/ is applied outside of its normal context. This sound change occurs primarily with final vowels, and does not usually occur word medially. Thus, more sophisticated transducers could learn better sound laws, which could translate into improved accuracy.

7.4 Experiment 3: Reconstructions

As a final trial, we wanted to see how each automatically found cognate group faired as compared to the "true groups" for actual reconstruction of proto-words. Our model is not optimized for faithful reconstruction, and so we used the Ancestry Resampling system of Bouchard-Côté et al. (2009). To evaluate, we matched each Latin word with the best possible cognate group for that word. The process for the matching was as follows. If two or three of the words in an constructed cognate group agreed, we assigned the Latin word associated with the true group to it. With the remainder, we executed a bipartite matching based on bigram overlap.

For evaluation, we examined the Levenshtein distance between the reconstructed word and the chosen Latin word. As a kind of "skyline," we compare to the edit distances reported in Bouchard-Côté et al. (2009), which was based on complete knowledge of the cognate groups. On this task, our reconstructed cognate groups had an average edit distance of 3.8 from the assigned Latin word. This compares favorably to the edit distances reported in Bouchard-Côté et al. (2009), who using oracle cognate assignments achieved an average Levenshtein distance of 3.0.[9]

8 Conclusion

We presented a new generative model of word lists that automatically finds cognate groups from scrambled vocabulary lists. This model jointly models the origin, propagation, and evolution of cognate groups from a common root word. We also introduced a novel technique for approximating automata. Using these approximations, our model can reduce the error rate by 80% over a baseline approach. Finally, we demonstrate that these automatically generated cognate groups can be used to automatically reconstruct proto-words faithfully, with a small increase in error.

Acknowledgments

Thanks to Alexandre Bouchard-Côté for the many insights. This project is funded in part by the NSF under grant 0915265 and an NSF graduate fellowship to the first author.

References

Leonard Bloomfield. 1938. *Language*. Holt, New York.

[8] This dataset will be made available at http://nlp.cs.berkeley.edu/Main.html#Historical

[9] Morphological noise and transcription errors contribute to the absolute error rate for this data set.

Alexandre Bouchard-Côté, Percy Liang, Thomas Griffiths, and Dan Klein. 2007. A probabilistic approach to diachronic phonology. In *EMNLP*.

Alexandre Bouchard-Côté, Thomas L. Griffiths, and Dan Klein. 2009. Improved reconstruction of protolanguage word forms. In *NAACL*, pages 65–73.

Hal Daumé III and Lyle Campbell. 2007. A Bayesian model for discovering typological implications. In *Conference of the Association for Computational Linguistics (ACL)*.

Hal Daumé III. 2009. Non-parametric Bayesian model areal linguistics. In *NAACL*.

Markus Dreyer and Jason Eisner. 2009. Graphical models over multiple strings. In *EMNLP*, Singapore, August.

Jason Eisner. 2001. Expectation semirings: Flexible EM for finite-state transducers. In Gertjan van Noord, editor, *FSMNLP*.

Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *ACL*.

Grzegorz Kondrak, Daniel Marcu, and Keven Knight. 2003. Cognates can improve statistical translation models. In *NAACL*.

Grzegorz Kondrak. 2001. Identifying cognates by phonetic and semantic similarity. In *NAACL*.

Harold W. Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97.

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

John B. Lowe and Martine Mazaudon. 1994. The reconstruction engine: a computer implementation of the comparative method. *Computational Linguistics*, 20(3):381–417.

Gideon S. Mann and David Yarowsky. 2001. Multipath translation lexicon induction via bridge languages. In *NAACL*, pages 1–8. Association for Computational Linguistics.

Thomas P. Minka. 2001. Expectation propagation for approximate bayesian inference. In *UAI*, pages 362–369.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 1996. Weighted automata in text and speech processing. In *ECAI-96 Workshop*. John Wiley and Sons.

Mehryar Mohri, 2009. *Handbook of Weighted Automata*, chapter Weighted Automata Algorithms. Springer.

Andrea Mulloni. 2007. Automatic prediction of cognate orthography using support vector machines. In *ACL*, pages 25–30.

John Nerbonne. 2010. Measuring the diffusion of linguistic change. *Philosophical Transactions of the Royal Society B: Biological Sciences*.

Michael P. Oakes. 2000. Computer estimation of vocabulary in a protolanguage from word lists in four daughter languages. *Quantitative Linguistics*, 7(3):233–243.

OED. 1989. "day, n.". In *The Oxford English Dictionary online*. Oxford University Press.

John Ohala, 1993. *Historical linguistics: Problems and perspectives*, chapter The phonetics of sound change, pages 237–238. Longman.

Jose Oncina and Marc Sebban. 2006. Learning stochastic edit distance: Application in handwritten character recognition. *Pattern Recognition*, 39(9).

Don Ringe, Tandy Warnow, and Ann Taylor. 2002. Indo-european and computational cladistics. *Transactions of the Philological Society*, 100(1):59–129.

Alan S.C. Ross. 1950. Philological probability problems. *Journal of the Royal Statistical Society Series B*.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2000. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *NAACL*.