Name Phylogeny: A Generative Model of String Variation

Nicholas Andrews and **Jason Eisner** and **Mark Dredze**
Department of Computer Science and Human Language Technology Center of Excellence
Johns Hopkins University
3400 N. Charles St., Baltimore, MD 21218 USA
{noa,eisner,mdredze}@jhu.edu

Abstract

Many linguistic and textual processes involve transduction of strings. We show how to learn a stochastic transducer from an unorganized collection of strings (rather than string pairs). The role of the transducer is to organize the collection. Our generative model explains similarities among the strings by supposing that some strings in the collection were not generated *ab initio*, but were instead derived by transduction from other, "similar" strings in the collection. Our variational EM learning algorithm alternately reestimates this phylogeny and the transducer parameters. The final learned transducer can quickly link any test name into the final phylogeny, thereby locating variants of the test name. We find that our method can effectively find name variants in a corpus of web strings used to refer to persons in Wikipedia, improving over standard untrained distances such as Jaro-Winkler and Levenshtein distance.

1 Introduction

Systematic relationships between pairs of strings are at the core of problems such as transliteration (Knight and Graehl, 1998), morphology (Dreyer and Eisner, 2011), cross-document coreference resolution (Bagga and Baldwin, 1998), canonicalization (Culotta et al., 2007), and paraphrasing (Barzilay and Lee, 2003). Stochastic transducers such as probabilistic finite-state transducers are often used to capture such relationships. They model a conditional distribution $p(y \mid x)$, and are ordinarily trained on input-output *pairs* of strings (Dreyer et al., 2008).

In this paper, we are interested in learning from an *unorganized* collection of strings, some of which might have been derived from others by *transformative linguistic processes* such as abbreviation, morphological derivation, historical sound or spelling change, loanword formation, translation, transliteration, editing, or transcription error. We assume that each string was derived from at most one parent, but may give rise to any number of children.

The difficulty is that most or all of these parent-child relationships are unobserved. We must reconstruct this evolutionary phylogeny. At the same time, we must fit the parameters of a model of the relevant linguistic process $p(y \mid x)$, which says what sort of children y might plausibly be derived from parent x. Learning this model of $p(y \mid x)$ helps us organize the training collection by reconstructing its phylogeny, and also permits us to generalize to new forms.

We will focus on the problem of name variation. We observe a collection of person names—full names, nicknames, abbreviated or misspelled names, etc. Some of these names can refer to the same person; we hope to detect this. It would be an unlikely coincidence if two mentions of John Jacob Jingleheimer Schmidt referred to different people, since this is a long and unusual name. Similarly, John Jacob Jingelhimer Smith and Dr. J. J. Jingleheimer may also be related names for this person. That is, these names may be derived from one another, via unseen relationships, although we cannot be sure.

Readers may be reminded of unsupervised clustering, in which "suspiciously similar" points can be explained as having been generated by the same cluster. Since each name is linked to at most one parent, our setting resembles single-link clustering—with a learned, asymmetric distance measure $p(y \mid x)$.

We will propose a generative process that makes explicit assumptions about how strings are copied with mutation. It is assumed to have generated all the names in the collection, in an unknown order. Given learned parameters, we can ask the model whether a name Dr. J. J. Jingelheimer in the collection is more likely to have been generated from scratch, or derived from some previous name.

1.1 Related Work

Several previous papers have also considered learning transducers or other models of word pairs when

Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 344–355, Jeju Island, Korea, 12–14 July 2012.

the pairing between inputs and outputs is not given. Most commonly, one observes parallel or comparable corpora in two languages, and must reconstruct a matching from one language's words to the other's before training on the resulting pairs (Schafer, 2006b; Klementiev and Roth, 2006; Haghighi et al., 2008; Snyder et al., 2010; Sajjad et al., 2011).

Hall and Klein (2010) extend this setting to more than two languages, where the phylogenetic tree is known. A given lexeme (abstract word) can be realized in each language by at most one word (string type), derived from the parent language's realization of the same lexeme. The system must match words that share an underlying lexeme (i.e., cognates), creating a matching of each language's vocabulary to its parent language's vocabulary. A further challenge is that the parent words are *unobserved* ancestral forms.

Similarly, Dreyer and Eisner (2011) organize words into morphological paradigms of a given structure. Again words with the same underlying lexeme (i.e., morphemes) must be identified. A lexeme can be realized in each grammatical inflection (such as "first person plural present") by exactly one word type, related to other inflected forms of the same lexeme, which as above may be unobserved. Their inference setting is closer to ours because the input is an unorganized collection of words—input words are not tagged with their grammatical inflections. This contrasts with the usual multilingual setting where each word is tagged with its true language.

In one way, our problem differs significantly from the above problems. We are interested in random variation that may occur *within* a language as well as across languages. A person name may have *unboundedly* many different variants. This is unlike the above problems, in which a lexeme has at most K realizations, where K is the (small) number of languages or inflections.[1] We cannot assign the observed strings to positions in an existing structure that is shared across all lexemes, such as a given phylogenetic tree whose K nodes represent languages, or a given inflectional grid whose K cells represent grammatical inflections. Rather, we must organize them into a idiosyncratic phylogenetic tree whose nodes are the string types or tokens themselves.

Names and words are not the only non-biological objects that are copied with mutation. Documents, database records, bibliographic entries, code, and images can evolve in the same way. Reconstructing these relationships has been considered by a number of papers on authorship attribution, near-duplicate detection, deduplication, record linkage, and plagiarism detection. A few such papers reconstruct a phylogeny, as in the case of chain letters (Bennett et al., 2003), malware (Karim et al., 2005), or images (Dias et al., 2012). In fact, the last of these uses the same minimum spanning tree method that we apply in §5.3. However, these papers do not *train* a similarity measure as we do. To our knowledge, these two techniques have not been combined outside biology.

In molecular evolutionary analysis, phylogenetic techniques *have* often been combined with estimation of some parametric model of mutation (Tamura et al., 2011). However, names mutate differently from biological sequences, and our mutation model for names (§4, §8) reflects that. We also posit a specific process (§3) that generates the name phylogeny.

2 An Example

A fragment of a phylogeny for person names is shown in Figure 1. Our procedure learned this automatically from a collection of name tokens, without observing any input/output pairs. The nodes of the phylogeny are the observed name types,[2] each one associated with a count of observed tokens.

Each arrow corresponds to a hypothesized mutation. These mutations reflect linguistic processes such as misspelling, initialism, nicknaming, transliteration, etc. As an exception, however, each arrow from the distinguished root node $\diamondsuit$ generates an initial name for a new entity. The descendants of this initial name are other names that subsequently evolved for that entity. Thus, the child subtrees of $\diamondsuit$ give a partition of the name types into entities.

Thanks to the phylogeny, the seemingly disparate names Ghareeb Nawaz and Muinuddin Chishti are seen to refer to the same entity. They may be traced back to their common ancestor Khawaja Gharib-

[1] In the above problems, one learns a set of $O(K)$ or $O(K^2)$ specialized transducers that relate Latin to Italian, singular to plural, etc. We instead use one global mutation model that applies to all names—but see footnote 14 on incorporating specialized transductions (Latin to Italian) within our mutation model.

[2] We cannot currently hypothesize unobserved intermediate forms, e.g., common ancestors of similar strings. See §6.2.

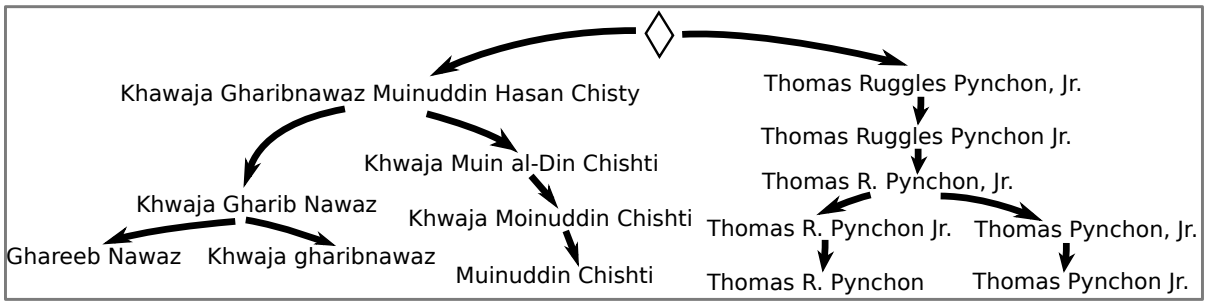


Figure 1: A portion of a spanning tree found by our model.

nawaz Muinuddin Hasan Chisty, from which both were derived via successive mutations.

Not shown in Figure 1 is our learned family p of conditional probability distributions, which models the likely mutations in this corpus. Our EM learning procedure found p jointly with the phylogeny. Specifically, it alternated between improving p and improving the distribution over phylogenies. At the end, we extracted the single best phylogeny.

Together, the learned p and the phylogeny in Figure 1 form an *explanation* of the observed collection of names. What makes it more probable than other explanations? Informally, two properties:

- Each node in the tree is plausibly derived from its parent. More precisely, the product of the edge probabilities under p is comparatively high. A different p would have reduced the probability of the events in this phylogeny. A different phylogeny would have involved a more improbable collection of events, such as replacing Chishti with Pynchon, or generating many unrelated copies of Pynchon directly from $\diamondsuit$.

- In the phylogeny, the parent names tend to be used often enough that it is plausible for variants of these names to have emerged. Our model says that new tokens are derived from previously generated tokens. Thus—other things equal—Barack Obama is more plausibly a variant of Barack Obama, Jr. than of Barack Obama, Sr. (which has fewer tokens).

3 A Generative Model of Tokens

Our model should reflect the *reasons* that name variation exists. A named entity has the form $y = (e, w)$ where w is a string being used to refer to entity e. A single entity e may be referred to on different occasions by different name strings w. We suppose that this is the result of *copying* the entity with occasional *mutation* of its name (as in asexual reproduction).

Thus, we assume the following simple generative process that produces an *ordered sequence* of tokens $y_1, y_2, \ldots$, where $y_i = (e_i, w_i)$.

• After the first k tokens $y_1, \ldots y_k$ have been generated, the author responsible for generating y_{k+1} must choose whom to talk about next. She is likely to think of someone she has heard about often in the past. So to make this choice, she selects one of the previous tokens y_i uniformly at random, each having probability $1/(k + \alpha)$; or else she selects $\diamondsuit$, with probability $\alpha/(k + \alpha)$.

• If the author selected a previous token y_i, then with probability $1 - \mu$ she copies it faithfully, so $y_{k+1} = y_i$. But with probability μ, she instead draws a mutated token $y_{k+1} = (e_{k+1}, w_{k+1})$ from the mutation model $p(\cdot \mid y_i)$. This preserves the entity ($e_{k+1} = e_i$ with probability 1), but the new name w_{k+1} is a stochastic transduction of w_i drawn from $p(\cdot \mid w_i)$.[3] For example, in referring to e_i, the author may shorten and respell $w_i =$ Khwaja Gharib Nawaz into $w_{k+1} =$ Ghareeb Nawaz (Figure 1).

• If the author selected $\diamondsuit$, she must choose a fresh entity $y_{k+1} = (e_{k+1}, w_{k+1})$ to talk about. So she sets e_{k+1} to a newly created entity, sampling its name w_{k+1} from the distribution $p(\cdot \mid \diamondsuit)$. For example, $w_{k+1} =$ Thomas Ruggles Pynchon, Jr. (Figure 1). Nothing prevents w_{k+1} from being a name that is already in use for another entity (i.e., w_{k+1} may equal w_j for some $j \leq k$).

[3] Straightforward extensions are to allow a variable mutation rate $\mu(y_i)$ that depends on properties of y_i, and to allow w_{k+1} to depend on known properties of e_i. See footnote 14 for further discussion of enriched tokens.

3.1 Relationship to other models

If we ignore the name strings, we can see that the sequence of entities $e_1, e_2, \ldots e_N$ is being generated from a Chinese restaurant process (CRP) with concentration parameter α. To the extent that α is low (so that $\diamond$ is rarely used), a few randomly chosen entities will dominate the corpus.

The CRP is equivalent to sampling $e_1, e_2, \ldots$ IID from an unknown distribution that was itself drawn from a Dirichlet process with concentration α. This is indeed a standard model of a distribution over entities. For example, Hall et al. (2008) use it to model venues in bibliographic entries.

From this characterization of the CRP, one can see that any permutation of this entity sequence would have the same probability. That is, our distribution over sequences of *entities* e is **exchangeable**.

However, our distribution over sequences of *named entities* $y = (e, w)$ is **non-exchangeable**. It assigns different probabilities to different orderings of the same tokens. This is because our model posits that later authors are influenced by earlier authors, copying entity names from them with mutation. So ordering is important. The mutation process is not symmetric—for example, Figure 1 reflects a tendency to shorten rather than lengthen names.

Non-exchangeability is one way that our present model differs from (parametric) transformation models (Eisner, 2002) and (non-parametric) transformation processes (Andrews and Eisner, 2011). These too are defined using mutation of strings or other types. From a transformation process, one can draw a distribution over types, from which the tokens are then sampled IID. This results in an *exchangeable* sequence of tokens, just as in the Dirichlet process.

We avoid transformation models here for three reasons. (1) Inference is more expensive. (2) A transformation process seems less realistic as a model of authorship. It constructs a distribution over derivational *paths*, similar to the paths in Figure 1. It effectively says that each token is generated by recapitulating some previously used path from $\diamondsuit$, but with some chance of deviating *at each step*. For an author to generate a name token this way, she would have to know the whole derivational history of the previous name she was adapting. Our present model instead allows an author simply to select a name she previously saw and copy or mutate its surface form. (3) One should presumably prefer to explain a novel name y as a mutation of a *frequent* name x, other things equal (§2). But surprisingly, inference under the transformation process does not prefer this.[4]

Another view of our present model comes from the literature on random graphs (e.g., for modeling social networks or the link structure of the web). In a *preferential attachment* model, a graph's vertices are added one by one, and each vertex selects some previous vertices as its neighbors. Our phylogeny is a *preferential attachment tree*, a random *directed* graph in which each vertex selects a *single* previous vertex as its parent. Specifically, it is a *random recursive tree* (Smythe and Mahmoud, 1995) whose vertices are the tokens.[5] To this simple random topology we have added a random labeling process with mutation. The first α vertices are labeled with $\diamondsuit$.

4 A Mutation Model for Strings

Our model in §3 samples the next token y, when it is not simply a faithful copy, from $p(y \mid x)$ or $p(y \mid \diamondsuit)$. The key step there is to sample the name string w_y from $p(w_y \mid w_x)$ or $p(w_y \mid \diamondsuit)$.

Our model of these distributions could easily incorporate detailed linguistic knowledge of the mutation process (see §8). Here we describe the specific model that we use in our experiments. Like many such models, it can be regarded as a stochastic finite-state string-to-string transducer parameterized by θ.

There is much prior work on stochastic models of edit distance (Ristad and Yianilos, 1998; Bilenko and Mooney, 2003; Oncina and Sebban, 2006; Schafer, 2006a; Bouchard-Côté et al., 2008; Dreyer et al., 2008, among others). For the present experiments, we designed a moderately simple one that employs (1) conditioning on one character of right context, (2) latent "edit" and "no-edit" regions to capture the fact that groups of edits are often made in close proximity, and (3) some simple special handling for the distribution conditioned on the root $p(w_y \mid \diamondsuit)$.

We assume a stochastic mutation process which, when given an input string w_x, edits it from left to

[4] The very fact that x has been frequently observed demonstrates that it has often chosen to stop mutating. This implies that it is likely to choose stop again rather than mutate into y.

[5] This is not the tree shown in Figure 1, whose vertices are types rather than tokens.

right into an output string w_y. Then $p(w_y \mid w_x)$ is the total probability of all operation sequences on w_x that would produce w_y. This total can be computed in time $O(|w_x| \cdot |w_y|)$ by dynamic programming.

Our process has four character-level edit operations: copy, substitute, insert, delete. It also has a distinguished no-edit operation that behaves exactly like copy. At each step, the process first randomly chooses whether to edit or no-edit, conditioned only on whether the previous operation was an edit. If it chooses to edit, it chooses a random edit type with some probability conditioned on the next input character. In the case of insert or substitute, it then randomly chooses an output character, conditioned on the type of edit *and* the next input character.

It is common to mutate a name by editing contiguous substrings (e.g., words). Contiguous regions of copying versus editing can be modeled by a low probability of transitioning between no-edit and edit regions.[6] Note that an edit region may include some copy edits (or substitute edits that replace a character with itself) without leaving the edit region. This is why we distinguish copy from no-edit.

Input and output strings are augmented with a trailing EOS ("end-of-string") symbol that is seen by the single-character lookahead. If the next character is EOS, the only available edit is insert. Alternatively, if the process selects no-edit, then EOS is copied to the output string and the process terminates.

In the case of $p(w_y \mid \diamondsuit)$, the input string is empty, and both input and output are augmented with a trailing EOS′ character that behaves like EOS. Then w_y is generated by a sequence of insertions followed by a copy. These are conditioned as usual on the next character, here EOS′, so the model can learn to insert more or different characters when the input is $\diamondsuit$.

The parameters θ determining the conditional probabilities of the different operations and characters are estimated with backoff smoothing.

5 Inference

The input to inference is a collection of named entity tokens y. Most are *untagged tokens* of the form $y = (?, w)$. In a semi-supervised setting, however, some of the tokens may be *tagged tokens* of the form $y = (e, w)$, whose true entity is known. The entity tags place a constraint on the phylogeny, since each child subtree of $\diamondsuit$ must correspond to exactly one entity.

5.1 An unrealistically supervised setting

Suppose we were lucky enough to fully observe the *sequence* of named entity tokens $y_i = (e_i, w_i)$ produced by our generative model. That is, suppose all tokens were tagged *and we knew their ordering*.

Yet there would still be something to infer: which tokens were derived from which previous tokens. This phylogeny is described by a spanning tree over the tokens. Let us see how to infer it.

For each potential edge $x \rightarrow y$ between named entity tokens, define $\delta(y \mid x)$ to be the probability of choosing x and copying it (possibly with mutation) to obtain y. So

$$\delta(y_j \mid \diamondsuit) = \alpha\, p(y_j \mid \diamondsuit) \quad (1)$$

$$\delta(y_j \mid y_i) = \mu\, p(y_j \mid y_i) + (1-\mu)\mathbb{1}(y_j = y_i) \quad (2)$$

except that if $i \geq j$ or if $e_i \neq e_j$, then $\delta(y_j \mid y_i) = 0$ (since y_j can only be derived from an *earlier* token y_i with the *same entity*).

Now the prior probability of generating $y_1, \ldots y_N$ with a given phylogenetic tree is easily seen to be a product over all tree edges, $\prod_j \delta(y_j \mid \text{pa}(y_j))$ where $\text{pa}(y_j)$ is the parent of y_j. As a result, it is known that the following are *efficient to compute* from the $(N+1) \times (N+1)$ matrix of δ values (see §5.3):

(a) the max-probability spanning tree

(b) the total probability of all spanning trees

(c) the marginal probability of each edge, under the posterior distribution on spanning trees

(a) is our single best guess of the phylogeny. We use this during evaluation. (b) gives the model likelihood, i.e., the total probability of the observed data $y_1, \ldots y_N$. To locally maximize the model likelihood, (c) can serve as the E step of our EM algorithm (§6) for tuning our mutation model. The M step then retrains the mutation model's parameters θ on input-output pairs $w_i \rightarrow w_j$, weighting each pair by its edge's posterior marginal probability (c), since that is the expected count of a $w_i \rightarrow w_j$ mutation. This computation is iterated.

[6] This somewhat resembles the traditional affine gap penalty in computational biology (Gusfield, 1997), which makes deletions or insertions cheaper if they are consecutive. We instead make consecutive edits cheaper regardless of the edit type.

5.2 The unsupervised setting

Now we turn to a real setting—fully unsupervised data. Two issues will force us to use an approximate inference algorithm. First, we have an *untagged* corpus: a token's entity tag e is never observed. Second, the *order* of the tokens is not observed, so we do not know which other tokens are candidate parents.

Our first approximation is to consider only phylogenies over *types* rather than tokens.[7] The type phylogeny in Figure 1 represents a set of possible token phylogenies. Each node of Figure 1 represents an untagged name type $y = (?, w)$. By grouping all n_y tokens of this type into a single node, we mean that the *first* token of y was derived by mutation from the parent node, while each *later* token of y was derived by copying an (unspecified) earlier token of y.

A token phylogeny *cannot* be represented in this way if two or more tokens of y were created by mutations. In that case, their name strings are equal only *by coincidence*. They may have different parents (perhaps of different entities), whereas the y node in a type phylogeny can have only one parent.

We argue, however, that these unrepresentable token phylogenies are comparatively unlikely *a posteriori* and can be reasonably ignored during inference. The first token of y is necessarily a mutation, but later tokens are much more likely to be copies. The probability of generating a later token y by copying some previous token is *at least*

$$(1 - \mu)/(N + \alpha),$$

while the probability of generating it in some other way is *at most*

$$\max(\alpha\, p(y \mid \diamondsuit),\ \ \mu \max_{x \in \mathcal{Y}} p(y \mid x))$$

where $\mathcal{Y}$ is the set of observed types. The second probability is typically much smaller: an author is unlikely to invent exactly the observed string y, certainly from $\diamondsuit$ but even by mutating a similar string x (especially when the mutation rate μ is small).

How do we evaluate a type phylogeny? Consider the probability of generating untagged tokens $y_1, \ldots y_N$ in that order and respecting the phylogeny:

$$\left(\prod_{k=1}^{N} \frac{1}{k+\alpha}\right) \prod_{y \in \mathcal{Y}} g(y \mid \text{pa}(y)) \left(\prod_{i=1}^{n_y - 1} i\,(1-\mu)\right) \tag{3}$$

where $g(y \mid \text{pa}(y))$ is a factor for generating the *first* token of y from its parent $\text{pa}(y)$, defined by

$$g(y \mid \diamondsuit) = \alpha \cdot p(y \mid \diamondsuit) \tag{4}$$

$$g(y \mid x) = \mu \cdot (\#\text{ tokens of } x \text{ preceding first token of } y) \cdot p(y \mid x) \tag{5}$$

But we do not actually know the token order: by assumption, our input corpus is only an *unordered* bag of tokens. So we must treat the hidden ordering like any other hidden variable and maximize the *marginal* likelihood, which sums (3) over all *possible* orderings (permutations). This sum can be regarded as the number of permutations $N!$ (which is fixed given the corpus) times the expectation of (3) for a permutation chosen uniformly at random.

This leads to our second approximation. We approximate this expectation of the product (3) with a product of expectations of its individual factors.[8] To find the expectation of (5), observe that the *expected* number of tokens of x that precede the first token of y is $n_x/(n_y+1)$, since each of the n_x tokens of x has a $1/(n_y + 1)$ chance of falling before all n_y tokens of y. It follows that the approximated probability of generating all tokens in *some* order, with our given type parentage, is *proportional to*

$$\prod_{y \in \mathcal{Y}} \delta(y \mid \text{pa}(y)) \tag{6}$$

where

$$\delta(y \mid \diamondsuit) = \alpha \cdot p(y \mid \diamondsuit) \tag{7}$$

$$\delta(y \mid x) = \mu \cdot p(y \mid x) \cdot n_x/(n_y + 1) \tag{8}$$

and the constant of proportionality depends on the corpus.

The above equations are analogous to those in §5.1. Again, the approximate posterior probability of a given type parentage tree is *edge-factored*—it is the product of individual edge weights defined by δ. Thus, we are again eligible to use the spanning tree algorithms in §5.3 below.

[7] Working over types improves the quality of our second approximation, and also speeds up the spanning tree algorithms. §6 explains how to regard this approximation as variational EM.

[8] In general this is an overestimate for each phylogeny.

Notice that the ratio α/μ controls the preference for an entity to descend from $\diamondsuit$ versus an existing entity. Thus, by tuning this ratio, we can control the number of entities inferred by our method, where each entity corresponds to one of the child subtrees of $\diamondsuit$.

Also note that n_x in the numerator of (8) means that y's parent is more likely to be frequent. Also, $n_y + 1$ in the denominator means that a frequent y is not as likely to have any parent $x \neq \diamondsuit$, because its first token probably falls early in the sequence where there are fewer available parents $x \neq \diamondsuit$.

5.3 Spanning tree algorithms

Define a complete directed graph G over the vertices $\mathcal{Y} \cup \{\diamondsuit\}$. The weight of an edge $x \to y$ is defined by $\delta(y \mid x)$. The (approximate) posterior probability of a given phylogeny given our evidence, is proportional to the product of the δ values of its edges.

Formally, let $\mathcal{T}_\diamondsuit(G)$ denote the set of spanning trees of G rooted at $\diamondsuit$, and define the weight of a particular spanning tree $T \in \mathcal{T}_\diamondsuit(G)$ to be the product of the weights of its edges:

$$w(T) = \prod_{(x \to y) \in T} \delta(y \mid x) \tag{9}$$

Then the posterior probability of spanning tree T is

$$p_\theta(T) = \frac{w(T)}{Z(G)} \tag{10}$$

where $Z(G) = \sum_{T \in \mathcal{T}_\diamondsuit(G)} w(T)$ is the partition function, i.e. the total probability of generating the data G via any spanning tree of the form we consider. This distribution is determined by the parameters θ of the transducer p_θ, along with the ratio α/μ.

There exist several algorithms to find the single maximum-probability spanning tree, notably Tarjan's implementation of the Chu-Liu-Edmonds algorithm, which runs in $O(m \log n)$ for a sparse graph or $O(n^2)$ for a dense graph (Tarjan, 1977). Figure 1 shows a spanning tree found by our model using Tarjan's algorithm. Here n is the number of vertices (in our case, types and $\diamond$), while m is the number of edges (which we can keep small by pruning, §6.1).

6 Training the Transducer with EM

Our inference algorithm assumes that we know the transducer parameters θ. We now explain how to optimize θ to maximize the marginal likelihood of the training data. This marginal likelihood sums over all the other latent variables in the model—the spanning tree, the alignments between strings, and the hidden token ordering.

The EM procedure repeats the following until convergence:

E-step: Given θ, compute the posterior marginal probabilities c_{xy} of all possible phylogeny edges.

M-step Given all c_{xy}, retrain θ to assign a high conditional probability to the mutations on the probable edges.

We actually use a variational EM algorithm: our E step approximates the true distribution q over all phylogenies with the closest distribution p that assigns positive probability only to type-based phylogenies. This distribution is given by (10) and minimizes $\mathrm{KL}(p \mid\mid q)$. We argued in section §5.2 that it should be a good approximation. The posterior marginal probability of a directed edge from vertex x to vertex y, according to (10), is

$$c_{xy} = \sum_{T \in \mathcal{T}_\diamondsuit(G) : (x \to y) \in T} p_\theta(T) \tag{11}$$

The probability c_{xy} is a "pseudocount" for the expected number of mutations from x to y. This is at most 1 under our assumptions.

Calculating c_{xy} requires summing over all spanning trees of G, of which there are n^{n-2} for a fully connected graph with n vertices. Fortunately, Tutte (1984) shows how to compute this sum by the following method, which extends Kirchhoff's classical matrix-tree theorem to weighted directed graphs. This result has previously been employed in non-projective dependency parsing (Koo et al., 2007; Smith and Smith, 2007).

Let $\mathbf{L} \in \mathbb{R}^{n \times n}$ denote the Laplacian of G, namely

$$\mathbf{L} = \begin{cases} \sum_{x'} \delta(y \mid x') & \text{if } x = y \\ -\delta(y \mid x) & \text{if } x \neq y \end{cases} \tag{12}$$

Tutte's theorem relates the determinant of the Laplacian to the spanning trees in graph G. In particular, the cofactor $\mathbf{L}^{0,0}$ is equal to the sum of the weights

of all directed spanning trees rooted at 0, which (supposing $\diamondsuit$ is indexed at 0) yields the partition function $Z(G)$.

The edge marginals of interest are related to the log partition function by

$$c_{xy} = \frac{\partial Z(G)}{\partial \delta(y \mid x)} \tag{13}$$

which has the closed-form solution

$$c_{xy} = \begin{cases} \delta(y \mid \diamondsuit)\mathbf{L}^{-1}_{yy} & \text{if } x = y \\ \delta(y \mid x)(\mathbf{L}^{-1}_{xx} - \mathbf{L}^{-1}_{xy}) & \text{if } x \neq y \end{cases} \tag{14}$$

Thus, the problem of computing edge marginals reduces to that of computing a matrix inverse, which may be done in $O(n^3)$ time.

At the M step, we retrain the mutation model parameters θ to maximize $\sum_{xy} c_{xy} \log p(w_y \mid w_x)$. This is tantamount to maximum conditional likelihood training on a supervised collection of (w_x, w_y) pairs that are respectively weighted by c_{xy}.

The M step is nontrivial because the term $p(w_y \mid w_x)$ sums over a hidden alignment between two strings. It may be performed by an inner loop of EM, where the E step uses dynamic programming to efficiently consider all possible alignments, as in (Ristad and Yianilos, 1996). In practice, we have found it effective to take only a single step of this inner loop. Such a Generalized EM procedure enjoys the same convergence properties as EM, but may reach a local optimum faster (Dempster et al., 1977).

6.1 Pruning the graph

For large graphs, it is essential to prune the number of edges to avoid considering all $n(n-1)$ input-output pairs. To prune the graph, we eliminate all edges between strings that do not share any common trigrams (case- and diacritic-insensitive), by setting their matrix entries to 0. As a result, the graph Laplacian is a sparse matrix, which often allows faster matrix inversion using preconditioned iterative algorithms. Furthermore, pruned edges do not appear in any spanning tree, so the E step will find that their posterior marginal probabilities are 0. This means that the input-output pairs corresponding to these edges can be ignored when re-estimating the transducer parameters in the M step. We found that pruning significantly improves training time with no appreciable loss in performance.[9]

6.2 Training with unobserved tokens?

A deficiency of our method is that it assumes that authors of our corpus have *only* been exposed to previous tokens in our corpus. In principle, one could also train with U additional tokens (e, w) where we observe neither e nor w, for very large U. This is the "universe of discourse" in which our authors operate.[10] In this case, we would need (expensive) new algorithms to reconstruct the strings w. However, this model could infer a more realistic phylogeny by positing unobserved ancestral or intermediate forms that relate the observed tokens, as in transformation models (Eisner, 2002; Andrews and Eisner, 2011).

7 Experimental Evaluation

7.1 Data preparation

Scraping Wikipedia. Wikipedia documents many variant names for entities. As a result, it has frequently been used as a source for mining name variations, both within and across languages (Parton et al., 2008; Cucerzan, 2007). We used Wikipedia to create a list of name aliases for different entities. Specifically, we mined English Wikipedia[11] for all redirects: page names that lead directly to another page. Redirects are created by Wikipedia users for resolving common name variants to the correct page. For example, the pages titled Barack Obama Junior and Barack Hussein Obama automatically redirect to the page titled Barack Obama. This redirection implies that the first two are name variants of the third. Collecting all such links within English Wikipedia yields a large number of aliases for each page. However, many redirects are for topics other than individual people, and these would be poor examples of name variation. In addition, some phrases

[9] For instance, on a dataset of approximately 6000 distinct names, pruning reduced the number of outgoing edges at each vertex to fewer than 100 per vertex.

[10] Notice that the N observed tokens would be approximately exchangeable in this setting: they are unlikely to depend on one another when $N \ll U$, and hence their order no longer matters much. In effect, generating the U hidden tokens constructs a rich distribution (analogous to a sample from the Dirichlet process) from which the N observed tokens are then sampled IID.

[11] Using a Wikipedia dump from February 2, 2011.

Ho Chi Minh, Ho chi mihn, Ho-Chi Minh, Ho Chih-minh
Guy Fawkes, Guy fawkes, Guy faux, Guy Falks, Guy Faukes, Guy Fawks, Guy foxe, Guy Falkes
Nicholas II of Russia, Nikolai Aleksandrovich Romanov, Nicholas Alexandrovich of Russia, Nicolas II
Bill Gates, Lord Billy, Bill Gates, BillGates, Billy Gates, William Gates III, William H. Gates
William Shakespeare, William shekspere, William shakspeare, Bill Shakespear
Bill Clinton, Billll Clinton, William Jefferson Blythe IV, Bill J. Clinton, William J Clinton

Figure 2: Sample alias lists scraped from Wikipedia. Note that only partial alias lists are shown for space reasons.

that redirect to an entity are descriptions rather than names. For example, 44th President of the United States also links to Barack Obama, but it is not a name variant.

Freebase filtering. To improve data quality we used Freebase, a structured knowledge base that incorporates information from Wikipedia. Among its structured information are entity types, including the type "person." We filtered the Wikipedia redirect collection to remove pairs where the target page was not listed as a person in Freebase. Additionally, to remove redirects that were not proper names (44th President of the United States), we applied a series of rule based filters to remove bad aliases: removing numerical names, parentheticals after names, quotation marks, and names longer than 5 tokens, since we found that these long names were rarely person names (e.g. United States Ambassador to the European Union, Success Through a Positive Mental Attitude which links to the author Napoleon Hill.) While not perfect, these modifications dramatically improved quality. The result was a list of 78,079 different person entities, each with one or more known names or aliases. Some typical names are shown in Figure 2.

Estimating empirical type counts. Our method is really intended to be run on a corpus of string tokens. However, for experimental purposes, we instead use the above dataset of string *types* because this allows us to use the "ground truth" given by the Wikipedia redirects. To synthesize token counts, empirical token frequencies for each type were estimated from the LDC Gigaword corpus,[12] which is a corpus of newswire text spanning several years. Wikipedia name types that did not appear in Gigaword were assigned a "backoff count" of one. Note that by virtue of the domain, many misspellings will not appear; however, edges "popular" names (which may be canonical names) will be assigned higher weight.

7.2 Experiments

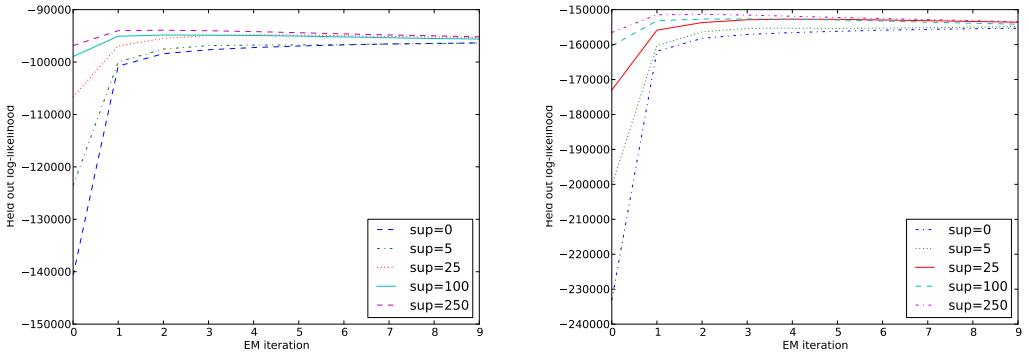
We begin by evaluating the generalization ability of a transducer trained using a transformation model. To do so, we measure log-likelihood on held-out entity title and alias pairs. We then verify that the generalization ability according to log-likelihood translates into gains for a name matching task. For the experiments in this section, we use $\alpha = 0.9$ and $\mu = 0.1$.[13]

Held-out log-likelihood. We construct pairs of entity title (input) and alias (output) names from the Wikipedia data. For different amounts of supervised data, we trained the transformation model on the training set, and plotted the log-likelihood of held-out test data for the transducer parameters at each iteration of EM. The held-out test set is constructed from a disjoint set of Wikipedia entities, the same number of entities as in the training set. We used different corpora of 1000 and 1500 entities for train and test.

Name matching. For each alias a in a test set (not seen at training time), we produce a ranking of test entity titles t according to transducer probabilities $p_\theta(a \mid t)$. A good transducer should assign high probability to transformations from the correct title for the alias. Mean reciprocal rank (MRR) is a commonly used metric to estimate the quality of a ranking, which we report in Figure 4. The reported mean is over all aliases in the test data. In addition to evaluating the ranking for different initializations of our transducer, we compare to two baselines: Levenshtein distance and Jaro-Winkler similarity. Jaro-Winkler is a measure on strings that was specifically designed for record linkage (Winkler, 1999). The

[12] LDC Catalog No. LDC2003T05.

[13] We did not find these parameters to be sensitive.



(a) 1000 entities.

(b) 1500 entities.

Figure 3: Learning curves for different initializations of the transducer parameters. Above, "sup=100" (for instance) means that 100 entities were used as training data to initialize the transducer parameters (constructing pairs between all title-alias pairs for those Wikpedia entities).

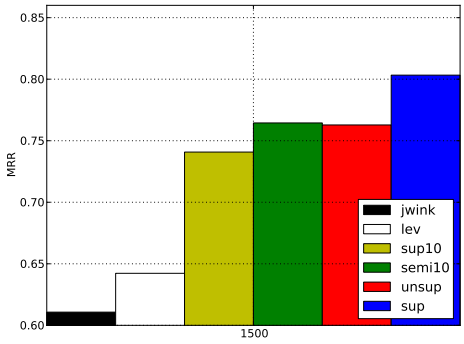


Figure 4: Mean reciprocal rank (MRR) results for different training conditions: "sup10" means that 10 entities (roughly 40 name pairs) were used as training data for the transducer; "semi10" means that the "sup10" model was used as initialization before re-estimating the parameters using our model; "unsup" is the transducer trained using our model without any initial supervision; "sup" is trained on all 1500 entities in the training set (an upper bound on performance); "jwink" and "lev" correspond to Jaro-Winkler and Levenshtein distance baselines.

matching experiments were performed on a corpus of 1500 entities (with separate corpora of the same size for training and test).

8 Conclusions and Future Work

We have presented a new unsupervised method for learning string-to-string transducers. It learns from a collection of related strings whose relationships are unknown. The key idea is that some strings are mutations of common strings that occurred earlier. We compute a distribution over the unknown phylogenetic tree that relates these strings, and use it to reestimate the transducer parameters via EM.

One direction for future work would be more sophisticated transduction models than the one we developed in §4. For names, this could include learning common nicknames (nonparametrically); explicitly modeling abbreviation processes such as initials; conditioning on name components such as title and middle name; and transliterating across languages.[14] In other domains, one could model bibliographic entry propagation, derivational morphology, or historical sound change (again using language tags).

Another future direction would be to incorporate the context of tokens in order to help reconstruct which tokens are coreferent. For example, we might extend the generative story to generate a context for token (e, w) conditioned on e. Combining contextual similarity with string similarity has previously proved very useful for identifying cognates (Schafer and Yarowsky, 2002; Schafer, 2006b; Bergsma and Van Durme, 2011). In our setting it would help to distinguish people with identical names, as well as determining whether two people with similar names are really the same.

[14] These last two points suggest that the mutation model should operate not on simple (entity, string) pairs, but on richer representations in which the name has been parsed into its components (Eisenstein et al., 2011), labeled with a language ID, and perhaps labeled with a phonological pronunciation. These additional properties of a named entity may be either observed or latent in training data. For example, if w_y and ℓ_y denote the string and language of name y, then define $p(y \mid x) = p(\ell_y \mid \ell_x) \cdot p(w_y \mid \ell_y, \ell_x, w_x)$. The second factor captures transliteration from language ℓ_x to language ℓ_y, e.g., by using §4's model with an (ℓ_x, ℓ_y)-specific parameter setting.

References

Nicholas Andrews and Jason Eisner. 2011. Transformation process priors. In *NIPS 2011 Workshop on Bayesian Nonparametrics: Hope or Hype?*, Sierra Nevada, Spain, December. Extended abstract (3 pages).

A. Bagga and B. Baldwin. 1998. Algorithms for scoring coreference chains. In *LREC*.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. In *Proc. of NAACL-HLT*, pages 16–23, Stroudsburg, PA, USA.

C. H. Bennett, M. Li, , and B. Ma. 2003. Chain letters and evolutionary histories. *Scientific American*, 288(3):76–81, June. More mathematical version available at `http://www.cs.uwaterloo.ca/~mli/chain.html`.

Shane Bergsma and Benjamin Van Durme. 2011. Learning bilingual lexicons using the visual similarity of labeled web images. In *Proc. of IJCAI*, pages 1764–1769, Barcelona, Spain.

Mikhail Bilenko and Raymond J. Mooney. 2003. Adaptive duplicate detection using learnable string similarity measures. In *Proc. of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 39–48, New York, NY, USA. ACM.

Alexandre Bouchard-Côté, Percy Liang, Thomas Griffiths, and Dan Klein. 2008. A probabilistic approach to language change. In *Proc. of NIPS*, pages 169–176.

S. Cucerzan. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *Proc. of EMNLP*.

Aron Culotta, Michael Wick, Robert Hall, Matthew Marzilli, and Andrew McCallum. 2007. Canonicalization of database records using adaptive similarity measures. In *Proc. of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 201–209.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Z. Dias, A. Rocha, and S. Goldenstein. 2012. Image phylogeny by minimal spanning trees. *IEEE Trans. on Information Forensics and Security*, 7(2):774–788, April.

Markus Dreyer and Jason Eisner. 2011. Discovering morphological paradigms from plain text using a Dirichlet process mixture model. In *Proc. of EMNLP*, pages 616–627. Supplementary material (9 pages) also available.

Markus Dreyer, Jason Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *Proc. of EMNLP*, pages 1080–1089, Honolulu, Hawaii, October. Association for Computational Linguistics.

Jacob Eisenstein, Tae Yano, William Cohen, Noah Smith, and Eric Xing. 2011. Structured databases of named entities from Bayesian nonparametrics. In *Proc. of the First workshop on Unsupervised Learning in NLP*, pages 2–12, Edinburgh, Scotland, July. Association for Computational Linguistics.

Jason Eisner. 2002. Transformational priors over grammars. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Philadelphia, July.

Dan Gusfield. 1997. *Algorithms on Strings, Trees, and Sequences—Computer Science and Computational Biology*. Cambridge University Press.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proc. of ACL-08: HLT*, pages 771–779.

David Hall and Dan Klein. 2010. Finding cognates using phylogenies. In *Association for Computational Linguistics (ACL)*.

Rob Hall, Charles Sutton, and Andrew McCallum. 2008. Unsupervised deduplication using cross-field dependencies. In *Proc. of the ACM SIGKDD International Conference On Knowledge Discovery and Data Mining*, KDD '08, pages 310–317.

Md. Enamul. Karim, Andrew Walenstein, Arun Lakhotia, and Laxmi Parida. 2005. Malware phylogeny generation using permutations of code. *Journal in Computer Virology*, 1(1–2):13–23.

Alexandre Klementiev and Dan Roth. 2006. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Proc. of COLING-ACL*, pages 817–824.

K. Knight and J. Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24:599–612.

Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. 2007. Structured prediction models via the matrix-tree theorem. In *Proc. of EMNLP-CoNLL*, pages 141–150.

Jose Oncina and Marc Sebban. 2006. Using learned conditional distributions as edit distance. In *Proc. of the 2006 Joint IAPR international Conference on Structural, Syntactic, and Statistical Pattern Recognition*, SSPR'06/SPR'06, pages 403–411.

Kristen Parton, Kathleen R. McKeown, James Allan, and Enrique Henestroza. 2008. Simultaneous multilingual search for translingual information retrieval. In *Proceeding of the ACM conference on Information and Knowledge Management*, CIKM '08, pages 719–728.

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

Eric Sven Ristad and Peter N. Yianilos. 1998. Learning string edit distance. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 20(5):522–532, May.

Hassan Sajjad, Alexander Fraser, and Helmut Schmid. 2011. An algorithm for unsupervised transliteration mining with an application to word alignment. In *Proc. of ACL*, pages 430–439.

Charles Schafer and David Yarowsky. 2002. Inducing translation lexicons via diverse similarity measures and bridge languages. In *Proc. of CONLL*, pages 146–152.

Charles Schafer. 2006a. Novel probabilistic finite-state transducers for cognate and transliteration modeling. In *7th Biennial Conference of the Association for Machine Translation in the Americas (AMTA)*.

Charles Schafer. 2006b. *Translation Discovery Using Diverse Smilarity Measures*. Ph.D. thesis, Johns Hopkins University.

David A. Smith and Noah A. Smith. 2007. Probabilistic models of nonprojective dependency trees. In *Proc. of EMNLP-CoNLL*, pages 132–140.

R. T. Smythe and H. M. Mahmoud. 1995. A survey of recursive trees. *Theory of Probability and Mathematical Statistics*, 51(1–27).

Benjamin Snyder, Regina Barzilay, and Kevin Knight. 2010. A statistical model for lost language decipherment. In *Proc. of ACL*, pages 1048–1057.

Koichiro Tamura, Daniel Peterson, Nicholas Peterson, Glen Stecher, Masatoshi Nei, and Sudhir Kumar. 2011. Mega5: Molecular evolutionary genetics analysis using maximum likelihood, evolutionary distance, and maximum parsimony methods. *Molecular Biology and Evolution*, 28(10):2731–2739.

R E Tarjan. 1977. Finding optimum branchings. *Networks*, 7(1):25–35.

W. Tutte. 1984. *Graph Theory*. Addison-Wesley.

William E. Winkler. 1999. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Census Bureau.