Contents

1 Lost Language Decipherment **3**

1.1 Chapter Overview 4

1.2 Introduction 4

1.3 Related Work 6

1.4 Background on Ugaritic 7

1.4.1 Manual Decipherment of Ugaritic 8

1.4.2 Linguistic Features of Ugaritic 9

1.5 Problem Formulation 9

1.6 Model 10

1.6.1 Intuitions 10

1.6.2 Model Structure 11

1.7 Inference 22

1.7.1 Sampling Word Analyses 23

1.7.2 Sampling Cognate Indicators 33

1.7.3 Resampling Chinese Restaurant Tables 33

1.7.4 Sampling Sparsity Indicators 35

1.7.5 Prediction 36

1.7.6 Implementation Details 38

1.8 Experiments 40

1.8.1 Corpus and Annotations 41

1.9 Results 42

1.9.1 Alphabetic Mapping 43

1.9.2 Cognate Decipherment 45
1.9.3 Cognate Identification 47
1.9.4 Comparison to Model Variants 54
1.9.5 Combining Model Variants 57
1.9.6 Related Language Discovery 57
1.9.7 Error Analysis 61
1.10 Conclusion and Future Work 63

A Ugaritic-Hebrew letter mappings **65**

Chapter 1

Lost Language Decipherment

In the previous two chapters, we examined the classical NLP tasks of part-of-speech tagging and grammar induction. We showed that multilingual analysis leads to significant performance gains over monolingual models. However, in both cases we assumed the existence of multilingual parallel texts. For most tasks and languages, this is indeed a realistic assumption. In contrast, this chapter examines a problem for which parallel text is typically not available: lost language decipherment. Instead of using parallel text to induce cross-lingual regularities, we instead look for *language-wide* structural similarities between the lost language and a living relative.

We make several assumptions in our approach to this task. Crucially, we assume that a known related language has already been identified (or hypothesized). Another assumption is that the writing systems of both languages are more or less *alphabetic* in nature.[1] Because the languages are related, we can expect a stable mapping between their letters to exist, due to the presence of *cognates*. Cognates are pairs of words which descend from a common word in a shared ancestral language.

Our definition of the computational decipherment task closely follows the setup typically faced by human decipherers [100]. Our input consists of texts in a lost language and a corpus of non-parallel data in the known related language. The

[1] To be more precise, the two writing systems dealt with here are of the *abjad* type, as vowels are not fully represented. For syllabic and logographic systems, modifications to our approach may be required.

decipherment itself involves two related sub-tasks:

1. finding the mapping between alphabets of the known and lost languages, and

2. translating words in the lost language into corresponding cognates of the known language.

1.1 Chapter Overview

Section 1.2 gives a broad introduction to the chapter. We list some of the intuitions that have guided human decipherers. We argue that language-wide similarities between the known and lost language can be automatically discovered by encoding these intuitions in a probabilistic model. We give a summary of the model and of our experimental results. Section 1.3 describes previous work related to language decipherment. Section 1.4 gives some background information on the discovery and decipherment of the Ugaritic language. Section 1.5 considers some assumptions made in our formulation of the task. Section 1.6 fully describes our model, and section 1.7 details our inference algorithm. Section 1.8 gives our experiments and results, and section 1.10 closes the chapter with some discussion and directions for future work.

1.2 Introduction

Dozens of lost languages have been deciphered by humans in the last two centuries. In each case, the decipherment has been considered a major intellectual breakthrough, often the culmination of decades of scholarly efforts. Computers have played no role in the decipherment any of these languages. Andrew Robinson, a noted author on writing systems and lost languages, represents the skeptical scholarly view that computers do not possess the "logic and intuition" required to unravel the mysteries of ancient scripts.[2] In this chapter, we demonstrate that at least some of this logic

[2] *"Successful archaeological decipherment has turned out to require a synthesis of logic and intuition …that computers do not (and presumably cannot) possess."* [100]

and intuition can be successfully modeled, allowing computational tools to be used in the decipherment process.

While there is no single formula that human decipherers have employed, manual efforts have focused on several guiding principles. A common starting point is to compare letter and word frequencies between the lost and known languages. In the presence of cognates the correct mapping between the languages will reveal similarities in frequency, both at the character and lexical level. In addition, morphological analysis plays a crucial role here, as highly frequent morpheme correspondences can be particularly revealing. In fact, these three strands of analysis (character frequency, morphology, and lexical frequency) are intertwined throughout the human decipherment process. Partial knowledge of each drives discovery in the others.

We capture these intuitions in a generative Bayesian model. This model assumes that words in the lost language are composed of morphemes which were generated with latent counterparts in the known language. We model bilingual morpheme pairs as arising through a series of Dirichlet processes. This allows us to assign probabilities based both on character-level correspondences (using a character-edit base distribution) as well as higher-level morpheme correspondences. In addition, our model carries out an implicit morphological analysis of the lost language, utilizing the known morphological structure of the related language. This model structure allows us to capture the interplay between the character- and morpheme-level correspondences that humans have used in the manual decipherment process.

In addition, we introduce a novel technique for imposing structural sparsity constraints on character-level mappings. We assume that an accurate alphabetic mapping between related languages will be sparse in the following way: each letter will map to a very limited subset of letters in the other language. We capture this intuition by adapting the so-called "spike-and-slab" prior to the Dirichlet-multinomial setting. For each pair of characters in the two languages, we posit an indicator variable which controls the prior likelihood of those characters substituting for one another. We define a joint prior over these indicator variables which encourages sparse settings.

We applied our model to a corpus of Ugaritic, an ancient Semitic language discov-

ered in 1928. Ugaritic was manually deciphered in 1932, using knowledge of Hebrew, a related language. We compare our method against the only existing decipherment baseline, an HMM-based character substitution cipher [65, 66]. The baseline correctly maps the majority of letters — 23 out of 30 — to their correct Hebrew counterparts, but only correctly translates 29% of all cognates. In comparison, our method yields correct mappings for 28 of 30 letters, and correctly translates 63% of all cognates into their Hebrew counterparts.

1.3 Related Work

Our work on decipherment has connections to several lines of work in statistical NLP. First, our work relates to research on automatic cognate identification. Early work on this task assumed the existence of bilingual dictionaries and a complete table of sound correspondences [75, 46]. The goal was to predict whether a given pair of words, which are known to be a translation pair, had descended from a common ancestral word. Kondrak [69] extended this line of work by removing the assumption that cognates must have identical meanings. He instead measures semantic similarity using glosses from a dictionary. Bergsma and Kondrak [7] consider an extension beyond language pairs to include evidence from multiple languages. They employ an integer linear programming framework to globally constrain cognate decisions by a large set of languages. More recently, Hall and Klein [51] presented a Bayesian model for cognate induction from unaligned word lists. They model the latent phylogenetic structure of the languages in order to induce more accurate cognate predictions. They assume that the languages in question share a single writing system, and that all the words given have at least one cognate.

In contrast to this line of work, we do not assume access to any sort of dictionary for our lost language, nor do we know the phonetic values of the symbols. In fact, we deal with two entirely distinct writing systems and actual archeological texts rather than artificial word lists.

A second related line of work is lexicon induction from non-parallel corpora. While

this research has similar goals, it typically builds on information or resources unavailable for ancient texts, such as comparable corpora, a seed lexicon, and cognate information [37, 96, 68, 48]. Moreover, distributional methods that rely on co-occurrence analysis operate over large corpora, which are typically unavailable for a lost language.

Two recent papers complementary to ours work are Penn and Choma [93] and Bouchard-Cote et al. [12]. In the first paper, Penn and Choma propose a quantitative method for the automatic classification of writing systems. In fact, this chapter assumes throughout that the basic nature of the lost-language writing system is known (i.e. that it is more or less alphabetic in nature – technically an *abjad*). The results presented by Penn and Choma support the plausibility of this assumption. In the second paper, Bouchard-Cote et al present a probabilistic model of diachronic phonology. As input, they assume a list of cognates across several Romance languages, and predict latent ancestral forms. In essence, this work assumes as *input* what our model produces as *output*. Thus, after automatically deducing cognates between Ugaritic and Hebrew, we could theoretically use the model of Bouchard-Cote et al to induce the latent ancestral forms in Proto-Semitic which led to these cognates.

Finally, Knight and Yamada [65] and Knight et al. [66] describe a computational HMM-based method for deciphering an unknown script that represents a known spoken language. This method "makes the text speak" by gleaning character-to-sound mappings from non-parallel character and sound sequences. It does not relate words in different languages, thus it cannot encode deciphering constraints similar to the ones considered in this paper. More importantly, this method had not been applied to archaeological data. While lost languages are gaining increasing interest in the NLP community [67], there have been no successful attempts of their automatic decipherment.

1.4 Background on Ugaritic

In this section we give some background information on the Ugaritic language. We first describe the story of its decipherment, and then briefly list some of its linguistic

properties.

1.4.1 Manual Decipherment of Ugaritic

The Ugaritic tablets (dating from the 14th through 12 centuries BCE) were first discovered in Syria in 1928 [105, 120]. At the time of their discovery, the cuneiform writing on the tablets was of an unknown type. Charles Virolleaud, who lead the initial decipherment effort, recognized that the script was likely alphabetic, since the inscribed words consisted of only thirty distinct symbols. The location of the tablets discovery further suggested that Ugaritic was likely to have been a Semitic language from the Western branch, with properties similar to Hebrew and Aramaic. This realization was crucial for deciphering the Ugaritic script. In fact, German cryptographer and Semitic scholar Hans Bauer decoded the first two Ugaritic letters—*mem* and *lambda*—by mapping them to Hebrew letters with similar occurrence patterns in prefixes and suffixes. Bootstrapping from this finding, Bauer found words in the tablets that were likely to serve as cognates to Hebrew words—e.g., the Ugaritic word for *king* matches its Hebrew equivalent. Through this process a few more letters were decoded, but the Ugaritic texts were still unreadable. What made the final decipherment possible was a sheer stroke of luck—Bauer guessed that a word inscribed on an ax discovered in the Ras Shamra excavations was the Ugaritic word for *ax*. Bauer's guess was correct, though he selected the wrong phonetic sequence. Edouard Dhorme, another cryptographer and Semitic scholar, later corrected the reading, expanding a set of translated words. Discoveries of additional tablets allowed Bauer, Dhorme and Virolleaud to revise their hypothesis, completing the initial decipherment. Since these initial decipherment results, scholars have spent decades mapping the individual words of the Ugaritic vocabulary to cognates in other Semitic languages. The translation of the Ugaritic tablets remains a lively and controversial field of study.

1.4.2 Linguistic Features of Ugaritic

Ugaritic shares many features with other ancient Semitic languages, following the same word order, gender, number, and case structure [55]. It is a morphologically rich language, with triliteral roots and many prefixes and suffixes.

At the same time, it exhibits a number of features that distinguish it from Hebrew. Ugaritic has a bigger phonemic inventory than Hebrew, yielding a bigger alphabet – 30 letters vs. 23 in Hebrew. Another distinguishing feature of Ugaritic is that vowels are only indicated for diphthongs or when following the glottal stop (through the use of three distinct glottal stop characters) while in Hebrew many long vowels are written using homorganic consonants. Ugaritic also does not have articles, while Hebrew nouns and adjectives take definite articles which are realized as prefixes. These differences result in significant divergence between Hebrew and Ugaritic cognates, thereby complicating the decipherment process.

1.5 Problem Formulation

We are given a corpus in a lost language and a non-parallel corpus in a related language from the same language family. Our primary goal is to translate words in the unknown language by mapping them to cognates in the known language. As part of this process, we induce a lower-level mapping between the letters of the two alphabets, capturing the regular phonetic correspondences found in cognates.

We make several assumptions about the writing system of the lost language. First, we assume that the writing system is alphabetic in nature. In general, this assumption can be easily validated by counting the number of symbols found in the written record. Next, we assume that the corpus has been transcribed into electronic format, where the graphemes present in the physical text have been unambiguously identified. Finally, we assume that words are explicitly separated in the text, either by white space or a special symbol.

We also make a mild assumption about the morphology of the lost language. We posit that each word consists of a stem, prefix, and suffix, where the latter two may be

omitted. This assumption captures a wide range of human languages and a variety of morphological systems. While the correct morphological analysis of words in the lost language must be learned, we assume that the inventory and frequencies of prefixes and suffixes in the known language are given.

In summary, the observed input to the model consists of two elements: (i) a list of unanalyzed word types derived from a corpus in the lost language, and (ii) a morphologically analyzed lexicon in a known related language derived from a separate corpus, in our case non-parallel.

1.6 Model

In this section we describe our model for lost language decipherment. This model is designed to encode various intuitions that humans have used in lost language decipherment. We first describe some of these intuitions.

1.6.1 Intuitions

Our goal is to incorporate the logic and intuition used by human decipherers in an unsupervised statistical model. To make these intuitions concrete, consider the following toy example, consisting of a lost language much like English, but written using numerals:

- 15234 *(asked)*
- 1525 *(asks)*
- 4352 *(desk)*

Analyzing the undeciphered corpus, we might first notice a pair of endings, -34, and -5, which both occur after the initial sequence 152- (and may likewise occur at the end of a variety of words in the corpus). If we know this lost language to be closely related to English, we can surmise that these two endings correspond to the English verbal suffixes *-ed* and *-s*. Using this knowledge, we can hypothesize the following

character correspondences: $(3 = e)$, $(4 = d)$, $(5 = s)$. We now know that $(4352 = des2)$ and we can use our knowledge of the English lexicon to hypothesize that this word is *desk*, thereby learning the correspondence $(2 = k)$. Finally, we can use similar reasoning to reveal that the initial character sequence 152- corresponds to the English verb *ask*.

As this example illustrates, human decipherment efforts proceed by discovering both character-level and morpheme-level correspondences. This interplay implicitly relies on a morphological analysis of words in the lost language, while utilizing knowledge of the known language's lexicon and morphology.

One final intuition our model captures is the sparsity of the alphabetic correspondence between related languages. We know from comparative linguistics that the correct mapping will preserve regular phonetic relationships between the two languages (as exemplified by cognates). As a result, each character in one language will map to a small number of characters in the other language (typically one, but sometimes two or three). By incorporating this structural sparsity intuition, we can allow the model to focus on on a smaller set of linguistically valid hypotheses.

Below we give an overview of our model, which is designed to capture these linguistic intuitions.

1.6.2 Model Structure

We start with the assumption that some number of observed word-forms in the lost language are cognate to words in the known language. Our model posits that these lost language words are composed of a sequence of morphemes (prefix, stem, suffix) each of which was probabilistically generated jointly with a latent counterpart in the known language.

Our goal is to find the morphemic boundaries and known language counterparts that lead to consistent correspondences both at the character and morpheme level. The technical challenge is that each level of correspondence (character and morpheme) can completely describe the observed data. A probabilistic mechanism based simply on one leaves no room for the other to play a role. We resolve this tension by

employing a hierarchical non-parametric Bayesian model: the distributions over bilingual morpheme pairs assign probability based on recurrent patterns at the morpheme level. These distributions are themselves drawn from a prior probabilistic process which favors distributions with consistent character-level correspondences.

We now give a top-down formal description of the model. See figure 1-1 for an accompanying graphical overview. There are four basic layers in the generative process:

1. **Structural sparsity**: draw a set of indicator variables $\vec{\lambda}$, each corresponding to to a character substitution (u, h).

2. **String-edit distribution**: draw a *base distribution* G_0 parameterized by weights on character-level edit operations.

3. **Morpheme-pair distributions**: draw a set of distributions on bilingual morpheme pairs $G^{stm}, G_1^{pre}, G_1^{suf}, \ldots$ from the Dirichlet process $\mathrm{DP}(G_0, \alpha_0)$.

4. **Word generation**: draw cognate-pairs in the lost and known language, as well as words in the lost language with no cognates.

We now go through each step in more detail.

Structural Sparsity

The first step of the generative process provides a control on the structural sparsity of character-substitution probabilities. By "structural sparsity" in this context, we refer to the desire that each character in the lost and known languages map to a very limited number of characters in the other language.

For each pair of characters (u, h) (where u and h range over characters in the lost and known languages, respectively) we posit a 0-1 indicator variable $\lambda_{(u,h)}$. Intuitively, we would like $\lambda_{(u,h)} = 1$ to indicate that u and h are *reflexes* of one another. That is, that the phonemes these two characters represent descend from a common phoneme in an ancestor language and that therefore u and h are likely to substitute

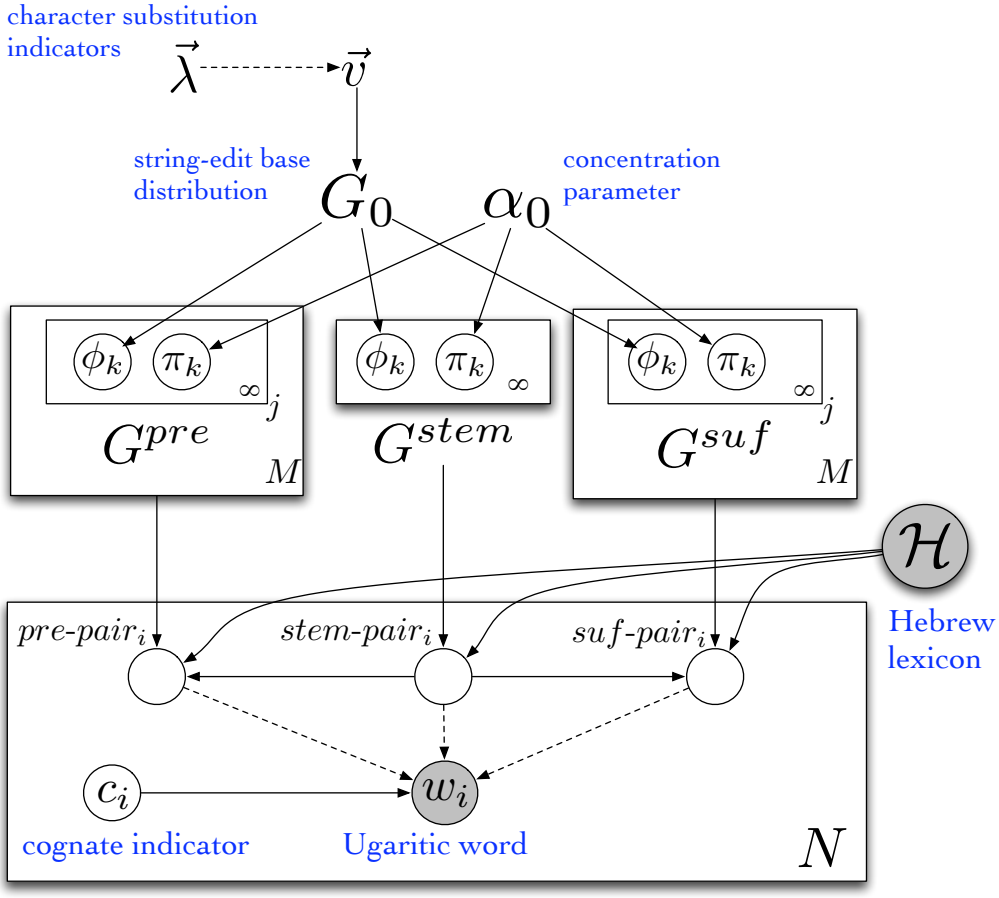

Figure 1-1: **Plate diagram of the decipherment model.** Observed variables are shaded in grey; full lines indicate probabilistic dependencies; dotted lines indicate deterministic dependencies; boxes indicate repeated variables, with the value in the bottom-right of each box indicating the number of repetitions. The structural sparsity indicator variables $\vec{\lambda}$ determine the values of the base distribution hyperparameters $\vec{v}$. The base distribution G_0 defines probabilities over string-pairs based on character-level edit operations. The stem-pair distribution G^{stm} is drawn from the Dirichlet process $\mathrm{DP}(G_0, \alpha_0)$ and is characterized by an infinite sequence of string-pairs (ϕ_k) and an accompanying sequence of weights (π_k). For each of M parts-of-speech, distributions over bilingual prefix-pairs and suffix pairs, G^{pre} and G^{suf} are likewise drawn. For each of N Ugaritic word-forms, an indicator variable c_i is drawn. If $c_i = 0$, the Ugaritic word w_i is drawn from an Ugaritic character-level language model. Otherwise, bilingual Ugaritic-Hebrew morpheme pairs are drawn, which deterministically yield the Ugaritic word w_i.

for one another in cognate-pairs. Thus $\{(u,h) \mid \lambda_{(u,h)} = 1\}$ represents the set of historically valid alphabetic mappings. We next define a joint prior over these variables $\vec{\lambda}$ which encourages sparse character mappings. With m lost-language characters and n known-language characters, we can view the set of possible values for $\vec{\lambda}$ as the set of all binary $m \times n$ matrices. Thus, the prior $P(\vec{\lambda})$ should define a distribution over such matrices which encourages both row and column sparsity:

$$\forall u : \sum_h \lambda_{(u,h)} \approx 1, \quad \forall h : \sum_u \lambda_{(u,h)} \approx 1.$$

Defining a normalized probability distribution over binary matrices that achieves this effect is difficult. Instead, we define our prior indirectly through a real-valued positive function g:

$$P(\vec{\lambda}) = \frac{g(\vec{\lambda})}{Z} \tag{1.1}$$

The value Z is a normalization term which depends only on the matrix dimensions m and n (i.e. the number of letters in the two alphabets):[3]

$$Z = \sum_{\vec{\lambda}} g(\vec{\lambda})$$

We define g in terms of two vector-valued feature functions $f(\vec{\lambda})$ and $f'(\vec{\lambda})$ along with a weight vector $\vec{w}$:

$$g(\vec{\lambda}) = \exp\left(f(\vec{\lambda}) \cdot \vec{w} + f'(\vec{\lambda}) \cdot \vec{w}\right) \tag{1.2}$$

Intuitively, f and f' count the number of sparsity violations for the lost and known languages, respectively, and $\vec{w}$ penalizes these violations. More precisely, we count the number of known-language characters to which each lost-language letter u maps: $c(u) = \sum_h \lambda_{(u,h)}$.[4] We then define a vector of features which count how many lost-

[3]Since computing Z is intractable, we develop an inference algorithm below (section 1.7.4) which only requires computation of the unnormalized function $g(\vec{\lambda})$.

[4]The functional dependence of $c(u)$ on $\vec{\lambda}$ is suppressed for notational clarity.

language characters u map to exactly i known-language characters beyond some allowed budget b_i:

$$f(\vec{\lambda})_i = \max\big(0,\ \big|\{u : c(u) = i\}\big| - b_i\big) \tag{1.3}$$

In similar fashion we define f', merely swapping the roles of u and h and defining corresponding budget values b'_i. Finally, we set:

$$\vec{w} = (w_0 = -\infty, w_1 = 0, w_2 = -50, w_{>2} = -\infty)$$
$$\vec{b} = (b_i = 0, \forall i)$$
$$\vec{b'} = (b'_0 = 0, b'_1 = 0, b'_2 = 7, b'_3 = 1, b'_{>3} = 0)$$

The asymmetry in our definition of $\vec{b}$ and $\vec{b'}$ results from our observation that the Ugaritic script contains seven characters more than Hebrew. Thus, we allow up to seven Hebrew letters to map to two Ugaritic letters without penalty, and we allow one Hebrew letter to map to three Ugaritic letters without penalty. In the reverse direction, no such allowances are made. Every Ugaritic letter which maps to more than one Hebrew letter is immediately penalized.

String-edit Distribution

The next step in the generative process is to draw our base measure G_0, which defines a distribution over all string pairs $(\mathbf{u}, \mathbf{h})$ (where $\mathbf{u}$ is composed of lost-language characters and $\mathbf{h}$ is composed of known-language characters). Distribution G_0 assigns probabilities based on character-level edit operations. These operations consist of substitutions (u, h), insertions (ϵ, h), deletions (u, ϵ), and a stop symbol (ϵ, ϵ).

Under G_0, each edit operation e is assigned a weight $0 \leq \rho_e \leq 1$. We partition edit operations into three categories,

$$SUB = \big\{(u, h) : \forall u\, \forall h\big\} \cup \big\{(\epsilon, \epsilon)\big\}$$
$$INS = \big\{(\epsilon, h) : \forall h\big\}$$
$$DEL = \big\{(u, \epsilon) : \forall u\big\},$$

and we require the weights corresponding to each set to sum to one:

$$\sum_{e \in SUB} \rho_e = 1, \quad \sum_{e \in INS} \rho_e = 1, \quad \sum_{e \in DEL} \rho_e = 1$$

In addition, G_0 provides a fixed distribution q over the *number* of insertions and deletions occurring in any single edit-sequence. Probabilities over edit-sequences are then defined according to G_0 by:

$$\frac{q\left(n_I, n_D\right)}{n_S^{[n_I+n_D]}} \cdot \prod_i \rho_{e_i}, \tag{1.4}$$

where n_I, n_D, n_S are the number of insertions, deletions, and substitutions in the edit-sequence, and the notation $x^{[n]}$ represents the rising factorial: $x(x+1)\cdots(x+n-1)$.

This function can be shown to be a probability mass function over all possible edit-sequences through the following generative process:

1. Draw substitutions according to $\{\rho_e : e \in SUB\}$ until the end symbol (ϵ, ϵ) is drawn.

2. Draw the number of insertions and deletions n_I, n_D according to q.

3. Draw n_I insertions according to $\{\rho_e : e \in INS\}$ and n_D deletions according to $\{\rho_e : e \in DEL\}$.

4. Place the insertions and deletions among the substitutions with uniform probability (i.e. with probability $\frac{1}{n_S^{[n_I+n_D]}}$)

The traditional probabilistic string-edit formulation (e.g. [98]) places all types of edit-operations under a single multinomial distribution. While this results in a simpler distribution over edit-sequences (i.e. simply $\prod_i \rho_i$), it has certain drawbacks which our formulation overcomes. In particular, we believe that it is important to provide an explicit distribution over the number of insertions and deletions and to prevent insertion and deletions from competing with substitutions for probability mass.

Note that each edit-sequence yields a string pair through projection and ϵ-removal:

$$\begin{aligned} y_1(\vec{e}) &\triangleq (e_1)_1 \cdots (e_k)_1 = \mathbf{u} \\ y_2(\vec{e}) &\triangleq (e_1)_2 \cdots (e_k)_2 = \mathbf{h} \\ y(\vec{e}) &\triangleq \big(y_1(\vec{e}),\ y_2(\vec{e})\big) = (\mathbf{u}, \mathbf{h}) \end{aligned} \tag{1.5}$$

If so desired, we can define an explicit distribution on string-pairs $(\mathbf{u}, \mathbf{h})$ by summing over all possible edit-sequences which yield $(\mathbf{u}, \mathbf{h})$:

$$P(\mathbf{u}, \mathbf{h}) = \sum_{\vec{e}\,:\, y(\vec{e})=(\mathbf{u},\mathbf{h})} P(\vec{e})$$

However, in the remainder of this thesis we will simply define G_0 as a distribution over edit-sequences $\vec{e}$.

In setting the value of q (the distribution over the number of insertions and deletions), we observe that the average Ugaritic word is over two letters longer than the average Hebrew word. Thus, occurrences of Hebrew character insertions are *a priori* likely, and Ugaritic character deletions are very unlikely. In our experiments, we simply set q to disallow Ugaritic deletions, and to allow up to one Hebrew insertion per morpheme (with probability 0.5).

As G_0 consists of three multinomial distributions (over SUB, INS, and DEL), we draw the three corresponding sets of weights from conjugate-prior Dirichlet distributions.

$$\big\{\rho_e : e \in INS\big\} \sim \text{Dirichlet}(\mathbf{1}) \tag{1.6}$$

$$\big\{\rho_e : e \in DEL\big\} \sim \text{Dirichlet}(\mathbf{1}) \tag{1.7}$$

$$\big\{\rho_e : e \in SUB\big\} \sim \text{Dirichlet}(\vec{v}) \tag{1.8}$$

For insertions and deletions we simply set all the Dirichlet hyperparameters to 1. In the case of substitutions, we employ the previously sampled sparsity indicator variables to deterministically set the hyperparameter vector $\vec{v}$. In particular, each

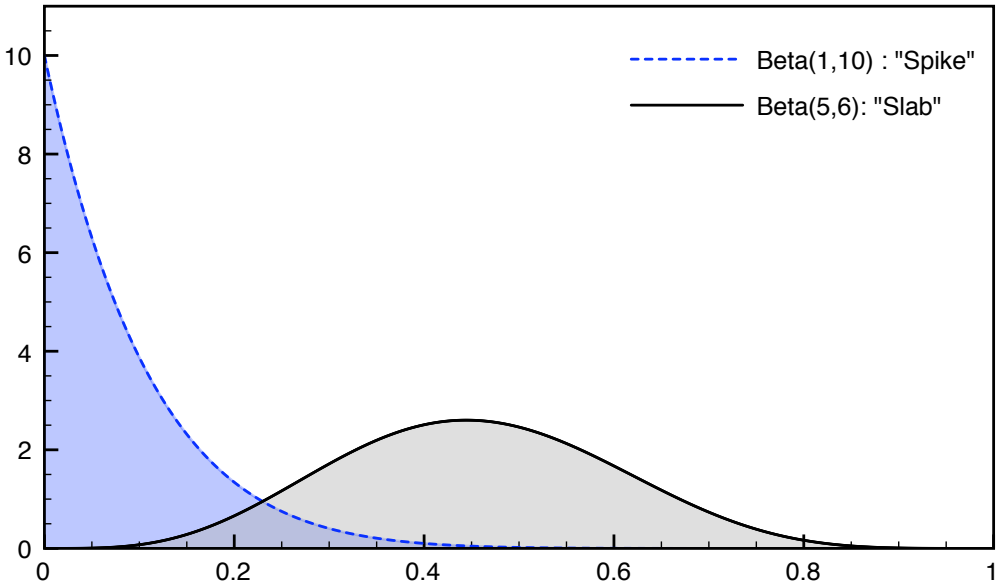

Figure 1-2: **"Spike-and-slab" effect** on character-edit probabilities $\vec{\rho}$. Assume three edit operations with indicator variables $\vec{\lambda} = (0, 1, 1)$ and resulting hyperparameters $\vec{v} = (1, 5, 5)$. Then the marginal priors are $\rho_{e_1} \sim \text{Beta}(1, 5 + 5)$ and $\rho_{e_2}, \rho_{e_3} \sim \text{Beta}(5, 1 + 5)$.

hyperparameter value v_e corresponds to an character edit-operation e and is set according to the indicator variable λ_e:

$$v_e = \begin{cases} 1 & \text{if } \lambda_e = 0, \\ K & \text{if } \lambda_e = 1. \end{cases} \tag{1.9}$$

where K is some constant value > 1 (set to 50 in our experiments). The resulting effect is that when $\lambda_e = 0$, the marginal prior density of the corresponding edit weight ρ_e spikes at 0. When $\lambda_e = 1$, the corresponding marginal prior density remains relatively flat and unconstrained. Figure 1-2 illustrates this effect graphically.

For similar applications of "spike and slab" priors, see [59] in the regression scenario and [119] in the context of topic models.

Morpheme-pair Distributions

Next we draw a series of distributions which *directly* assign probability to morpheme pairs (or more precisely to edit sequences which yield morpheme pairs). The previously drawn base measure G_0, along with a fixed concentration parameter α_0, define a *Dirichlet process* [2]: $\mathrm{DP}(G_0, \alpha_0)$, which provides a probability distribution over all possible morpheme-pair distributions. Distributions drawn from this Dirichlet process assign large probability mass to a small number of morpheme pairs, while remaining sensitive to the character-level substitution probabilities of the base distribution.

Our model distinguishes between three types of morphemes: prefixes, stems, and suffixes. Since each part-of-speech in a language carries with it unique prefix and suffix frequencies, we generate distinct prefix and suffix distributions for each of M parts-of-speech, and a single distribution over all stem-pairs:

$$\begin{aligned} & G^{stm} && \sim \quad \mathrm{DP}(G_0, \alpha_0) \\ \forall j \in 1 \ldots M: \quad & G_j^{pre} && \sim \quad \mathrm{DP}(G_0, \alpha_0) \\ \forall j \in 1 \ldots M: \quad & G_j^{suf} && \sim \quad \mathrm{DP}(G_0, \alpha_0) \end{aligned}$$

While we avoid dealing directly with these distributions in our inference procedure, they can be viewed as arising from a stick-breaking process [103]:

1. Draw an infinite sequence of i.i.d. edit-sequences from the base distribution:
   $\phi_1, \phi_2, \ldots \sim G_0$
2. Draw an infinite sequence of i.i.d. weights:
   $\pi'_1, \pi'_2, \ldots \sim \mathrm{Beta}(1, \alpha_0)$
3. Normalize the weights:
   $\pi_k = \pi'_k \cdot \sum_{i<k} (1 - \pi'_i)$
4. Define a probability density function over edit-sequences using the dirac delta function[5]:

[5] $\delta_b = (1 \text{ if } b, 0 \text{ if } \neg b)$

$$p(\vec{e}) = \sum_{k=1}^{\infty} \pi_k \cdot \delta_{\phi_k = \vec{e}} \tag{1.10}$$

Word Generation

Once the morpheme-pair distributions have been drawn, actual words may now be generated. To generate word $\mathbf{u}_i$ of the N word-forms observed in the lost-language texts, we first draw a cognate indicator variable c_i. This variable determines whether $\mathbf{u}_i$ is to be generated along with a known-language cognate, or alone as a non-cognate. We model c_i as a simple Bernoulli random variable, with fixed parameter $P(c_i = 1)$. As discussed below in section 1.9.3, this cognate prior may be varied to induce different prediction thresholds. In our main experiments, we simply set $P(c_i = 1) = 0.5$. If $c_i = 1$, then a cognate word pair $(\mathbf{u}_i, \mathbf{h}_i)$ is generated according to our model as follows:

$\vec{e}_{stm}$	$\sim$	G^{stm}	// draw stem edit-sequence
pos_i	$\leftarrow$	$j = \text{pos}(\vec{e}_{stm})$	// determine part-of-speech j (see below)
$\vec{e}_{pre}$	$\sim$	G_j^{pre}	// draw prefix edit-sequence
$\vec{e}_{suf}$	$\sim$	G_j^{suf}	// draw suffix edit-sequence
$\left(\mathbf{u}^{pre}, \mathbf{h}^{pre}\right)$	$\leftarrow$	$y(\vec{e}_{pre})$	// yield morpheme pairs (equation 1.5)
$\left(\mathbf{u}^{stm}, \mathbf{h}^{stm}\right)$	$\leftarrow$	$y(\vec{e}_{stm})$	
$\left(\mathbf{u}^{suf}, \mathbf{h}^{suf}\right)$	$\leftarrow$	$y(\vec{e}_{suf})$	
$\mathbf{u}_i$	$\leftarrow$	$\mathbf{u}^{pre}\,\mathbf{u}^{stm}\,\mathbf{u}^{suf}$	// concatenate morphemes
$\mathbf{h}_i$	$\leftarrow$	$\mathbf{h}^{pre}\,\mathbf{h}^{stm}\,\mathbf{h}^{suf}$	

Besides observing the resulting lost-language word $\mathbf{u}_i$, we also assume the existence of a known-language lexicon $\mathcal{H}$. This lexicon provides us with knowledge of all possible stems, prefixes, and suffixes, for each part-of-speech j in the known language: $\mathcal{H}_j^{stm}, \mathcal{H}_j^{pre}, \mathcal{H}_j^{suf}$. We treat this knowledge as a hard constraint on the set of possible values for the latent cognate word $\mathbf{h}_i$. In particular, we treat as an observation that

for some part-of-speech j:

$$\mathbf{h}^{pre} \in \mathcal{H}_j^{pre} \quad \wedge \quad \mathbf{h}^{stm} \in \mathcal{H}_j^{stm} \quad \wedge \quad \mathbf{h}^{suf} \in \mathcal{H}_j^{suf}.$$

Probabilistically, we view the part-of-speech assignment j as being determined by the generated stem edit-sequence: [6]

$$\text{pos}(\vec{e}_{stm}) \triangleq j \text{ such that } y_2(\vec{e}_{stm}) \in G_j^{stm}$$

The prefix and suffix morpheme-pairs are then drawn from the appropriate distributions G_j^{pre} and G_j^{suf}. In this way, the prefix and suffix both probabilistically depend on the stem (by way of its part-of-speech). We will see in section 1.7.1 what role all these observations play in our inference algorithm.

If $c_i = 0$, then $\mathbf{u}_i$ was generated without a known-language cognate. We assume that the lone lost-language word was generated according to a unigram character-level language model

$$P(\mathbf{u}_i | c_i = 0) = P(\#) \cdot \prod_j P\left(\mathbf{u}_i[j]\right), \tag{1.11}$$

where # is a special end-word character and $\mathbf{u}_i[j]$ denotes the jth character in word $\mathbf{u}_i$. We note that this differs somewhat from the model presented in our previous publication [109], where a uniform distribution was assumed over letters of the lost language for the purpose of noncognate generation. See section 1.9.3 below for a more detailed discussion of this issue.

In summary, this model structure captures both character and lexical level correspondences, while utilizing morphological and part-of-speech knowledge of the known language. An additional feature of this multi-layered model structure is that each

[6] If more than one part-of-speech meet this criterion, then we assume a uniform distribution over all such values.

Algorithm 1: Gibbs sampler for lost language decipherment.

Input: Lost-language word forms $\mathbf{u}_1, \ldots, \mathbf{u}_N$ and known-language lexicon $\mathcal{H}$
Output: 1000 samples of latent variables

Initialize latent variables;
for $r \leftarrow 1$ **to** 1000 **do**
  for $i \leftarrow 1$ **to** N **do**
    Sample word analysis $\left[\vec{e}_{pre}, \vec{e}_{stm}, \vec{e}_{suf}\right]_i^{(r)}$ // Section 1.7.1
    Sample cognate indicator $c_i^{(r)}$ // Section 1.7.2
  Resample Chinese restaurant tables; // Section 1.7.3
  foreach *lost-language character* u **do**
    Sample sparsity indicators $\left\{\lambda_{(u,h)} : \forall h\right\}^{(r)}$ // Section 1.7.4
  foreach *known-language character* h **do**
    Sample sparsity indicators $\left\{\lambda_{(u,h)} : \forall u\right\}^{(r)}$ // Section 1.7.4

distribution over morpheme pairs is derived from the single character-level base distribution G_0. As a result, any character-level mappings learned from one type of morphological correspondence will be propagated to all other morpheme distributions. Finally, the character-level mappings discovered by the model are encouraged to obey linguistically motivated structural sparsity constraints. In the next section we describe our inference procedure at length.

1.7 Inference

For each word-form $\mathbf{u}_i$ in our undeciphered language we wish to first predict a cognate indicator variable c_i and, if $c_i = 1$, then the corresponding morphemes in the known language $\left(\mathbf{h}^{pre}\, \mathbf{h}^{stm}\, \mathbf{h}^{suf}\right)_i$. Ideally we would predict each word analysis with highest posterior marginal probability under our model given the observed undeciphered corpus $\mathbf{u}_1, \ldots, \mathbf{u}_N$ and known-language lexicon $\mathcal{H}$. In order to do so, we need to integrate out all other latent variables in our model:

- Structural sparsity indicator variables $\vec{\lambda}$
- String-edit base distribution G_0

- Morpheme-pair distributions $G^{stm}, G_1^{pre}, G_1^{suf}, \ldots$
- Morphological segmentations
- Latent cognates of all other words $\mathbf{u}_{k \neq i}$

As these integrals are intractable to compute exactly, we resort to the standard Monte Carlo approximation. We collect samples of the variables over which we wish to marginalize but for which we cannot compute closed-form integrals. We then approximate the marginal probabilities for each undeciphered word $\mathbf{u}_i$ by summing over all the samples, and predicting the analysis with highest posterior probability.

In our sampling algorithm, we avoid sampling the base distribution G_0 and the morpheme-pair distributions (G^{stm} etc.), instead marginalizing them out using analytical closed forms. We explicitly sample the sparsity indicator variables $\vec{\lambda}$, the cognate indicator variables c_i, and latent word analyses (segmentations and cognate counterparts). To sample these variables tractably, we use Gibbs sampling to sample each latent variable conditioned on our current sample of the others. Although the samples are no longer independent, they form a Markov chain whose stationary distribution is the true joint distribution defined by the model [40].

See algorithm 1 for a high-level overview of our Gibbs sampler. In the following sections we provide details for each sampling step.

1.7.1 Sampling Word Analyses

For each lost-language word $\mathbf{u}_i$ with corresponding cognate indicator $c_i = 1$, we sample a segmentation $\mathbf{u}^{pre}\,\mathbf{u}^{stm}\,\mathbf{u}^{suf}$ with corresponding cognate morphemes $\mathbf{h}^{pre}\,\mathbf{h}^{stm}\,\mathbf{h}^{suf}$.

More precisely, we sample three edit-sequences $\vec{e}_{pre}, \vec{e}_{stm}, \vec{e}_{suf}$ which yield:

$$\begin{aligned} y(\vec{e}_{pre}) &= \left(\mathbf{u}^{pre}, \mathbf{h}^{pre}\right) \\ y(\vec{e}_{stm}) &= \left(\mathbf{u}^{stm}, \mathbf{h}^{stm}\right) \\ y(\vec{e}_{suf}) &= \left(\mathbf{u}^{suf}, \mathbf{h}^{suf}\right), \end{aligned}$$

with the hard constraint that the resulting analysis must be consistent (i) with the observed lost-language word-form $\mathbf{u}_i$:

$$\mathbf{u}_i \quad = \quad \mathbf{u}^{pre}\,\mathbf{u}^{stm}\,\mathbf{u}^{suf},$$

and (ii) with the observed known-language lexicon $\mathcal{H}$:

$$\mathbf{h}^{pre} \in \mathcal{H}_j^{pre} \ \wedge \ \mathbf{h}^{stm} \in \mathcal{H}_j^{stm} \ \wedge \ \mathbf{h}^{suf} \in \mathcal{H}_j^{suf} \quad \text{(for some part-of-speech } j\text{)}.$$

We break this task down into two steps, (i) first sampling a segmentation and part-of-speech, and (ii) then sampling the actual edit-sequences (yielding the corresponding cognate morphemes).

Sampling Segmentations and Parts-of-speech

We sample a segmentation and part-of-speech by simply enumerating all possibilities, calculating their posterior probabilities, and sampling from the resulting discrete distribution.

We now show how to calculate the posterior for a each segmentation $\mathbf{u}_i = \mathbf{u}^{pre}\,\mathbf{u}^{stm}, \mathbf{u}^{suf}$

and part-of-speech j:[7]

$$
\begin{aligned}
& P(\mathbf{u}^{pre}, \mathbf{u}^{stm}, \mathbf{u}^{suf}, \mathrm{pos}_i = j \mid \mathbf{u}_i, \mathcal{H}) \\
&= \sum_{\vec{e}_{pre}, \vec{e}_{stm}, \vec{e}_{suf}} P(\vec{e}_{pre}, \vec{e}_{stm}, \vec{e}_{suf}, \mathbf{u}^{pre}, \mathbf{u}^{stm}, \mathbf{u}^{suf}, j \mid \mathbf{u}_i, \mathcal{H}) \\
&\propto \sum_{\vec{e}_{pre}, \vec{e}_{stm}, \vec{e}_{suf}} P(\vec{e}_{pre}, \vec{e}_{stm}, \vec{e}_{suf}, \mathbf{u}^{pre}, \mathbf{u}^{stm}, \mathbf{u}^{suf}, j) \;\; P(\mathcal{H} \mid \vec{e}_{pre}, \vec{e}_{stm}, \vec{e}_{suf}, j) \\
&= \sum_{\vec{e}_{pre}, \vec{e}_{stm}, \vec{e}_{suf} \in C_1 \times C_2 \times C_3} P(\vec{e}_{pre}, \vec{e}_{suf} \mid j) \;\; P(\vec{e}_{stm}) \; P(\mathcal{H}) \\
&\propto \sum_{\vec{e}_{pre} \in C_1} P(\vec{e}_{pre} \mid j) \sum_{\vec{e}_{stm} \in C_2} P(\vec{e}_{stm}) \sum_{\vec{e}_{suf} \in C_3} P(\vec{e}_{suf} \mid j)
\end{aligned}
\tag{1.12}
$$

where C_1, C_2, C_3 are the sets of all edit-sequences (i) yielding the respective morphemes $\mathbf{u}^{pre}, \mathbf{u}^{stm}, \mathbf{u}^{suf}$ and (ii) consistent with the known-language lexicon for part-of-speech j. More precisely:

$$
\begin{aligned}
C_1 &= \left\{ \vec{e} \mid y_1(\vec{e}) = \mathbf{u}^{pre} \;\wedge\; y_2(\vec{e}) \in \mathcal{H}_j^{pre} \right\} \\
C_2 &= \left\{ \vec{e} \mid y_1(\vec{e}) = \mathbf{u}^{stm} \;\wedge\; y_2(\vec{e}) \in \mathcal{H}_j^{stm} \right\} \\
C_3 &= \left\{ \vec{e} \mid y_1(\vec{e}) = \mathbf{u}^{suf} \;\wedge\; y_2(\vec{e}) \in \mathcal{H}_j^{suf} \right\}
\end{aligned}
$$

Computing Edit-sequence Probabilities

We defer for now the question of how to efficiently sum over these sets, and instead start by deriving the individual probability terms in equation 1.12. Let us consider the posterior probability of a stem edit-sequence $\vec{e}$. We first note that were we to Gibbs-sample the parameters of the stem morpheme-pair distribution G^{stm}, $P(\vec{e})$ would be directly given by equation 1.10. Instead, we marginalize out these parameters using

[7] For notational clarity, we leave the conditioning on the other sampled variables implicit throughout this section.

the standard Chinese Restaurant Process closed form [34]:

$$P(\vec{e}) = \frac{N_{stm}(\vec{e}) + \alpha_0 \cdot F(\vec{e})}{N_{cog} + \alpha_0}, \tag{1.13}$$

where $N_{stm}(\vec{e})$ gives the number of other cognated words for which $\vec{e}$ appears as the stem: $\big|\{k \neq i \mid (\vec{e}_{stm})_k = \vec{e} \ \wedge \ c_k = 1\}\big|$, and N_{cog} gives the total number of other cognated words: $\big|\{k \neq i \mid c_k = 1\}\big|$.[8] The function $F(\vec{e})$ gives the probability assigned to $\vec{e}$ by the string-edit base distribution. If the multinomial parameters of the base distribution were Gibbs-sampled, then F would be given by equation 1.4. Instead, we employ the standard marginalized posterior distribution for multinomials with Dirichlet hyperparameters to obtain:

$$F(\vec{e}) = \frac{q(n_I, n_D)}{n_S^{[n_I+n_D]}} \cdot \prod_{e \in \vec{e}} f(e), \tag{1.14}$$

with:

$$f(e) = \begin{cases} \frac{N(e)+1}{N_I + \sum_{\text{ins } e'} 1} & \text{if } e = (\epsilon, h), \\ \frac{N(e)+1}{N_D + \sum_{\text{del } e'} 1} & \text{if } e = (u, \epsilon), \\ \frac{N(e)+v_e}{N_S + \sum_{\text{sub } e'} v_{e'}} & \text{if } e = (u, h) \text{ or } e = (\epsilon, \epsilon) \end{cases}$$

where $N(e)$ denotes the number of times the edit-operation e has occurred in the *unique* edit-sequences obtained from each morpheme-pair distribution $(G^{stm}, G_1^{pre}, \ldots)$; or more precisely, the number of times e appears among the sets:

$$\begin{aligned} E^{stm} &= \big\{(\vec{e}_{stm})_{k \neq i}\big\} \\ E_j^{pre} &= \big\{(\vec{e}_{pre})_{k \neq i} \mid \text{pos}_k = j\big\}, \ \forall j \in 1 \ldots M \\ E_j^{suf} &= \big\{(\vec{e}_{suf})_{k \neq i} \mid \text{pos}_k = j\big\}, \ \forall j \in 1 \ldots M. \end{aligned}$$

[8] Both according to the most recently sampled latent values for words $\mathbf{u}_{k \neq i}$.

And N_I, N_D, N_S give the *total* number of respective insertions, deletions, and substitutions among those sets. We emphasize that these sets only distinguish between *unique* edit-sequences (e.g. the edit-sequence $(\vec{e}_{stm})_m = (\vec{e}_{stm})_n$ for $m \neq n$ would only occur once in E^{stm}). Only a single instance of each edit-sequence type is drawn from the base distribution itself – the first person at each Chinese restaurant table, as it were. Finally, for an explanation of the first factor in equation 1.14 we refer the reader back to equation 1.4.

Similar reasoning applies in deriving the probability terms for prefix and suffix edit-sequences, with the exception that counts are now restricted to a individual parts-of-speech (since we have different G_j^{pre} and G_j^{suf} for each part-of-speech j). E.g. for prefixes:

$$P(\vec{e} \mid j) = \frac{N_{pre,j}(\vec{e}) + \alpha_0 \cdot F(\vec{e})}{N_{cog,j} + \alpha_0}$$

where $N_{pre,j}(\vec{e})$ gives the number of other cognated words with part-of-speech j for which sequence $\vec{e}$ appears as the prefix: $\big|\{k \neq i \mid (\vec{e}_{pre})_k = \vec{e} \ \wedge \ c_k = 1 \ \wedge \ \text{pos}_k = j\}\big|$, and N_{cog} gives the total number of other cognated words with part-of-speech j: $\big|\{k \neq i \mid c_k = 1 \ \wedge \ \text{pos}_k = j\}\big|$. Note that all edit-sequence posteriors depend on the function $F(\vec{e})$ since all morpheme-pair distributions derive from the same base measure G_0.

Summing Probabilities with Finite-state Machines

The individual probability terms in equation 1.12 can thus be computed by simply caching counts from each word's sampled analysis. However, to compute segmentation probabilities we still need to efficiently sum these terms over the sets C_1, C_2, and C_3. For example, plugging equation 1.13 into the middle sum of equation 1.12 yields the following computation (after removing the constant denominator):

$$\sum_{\vec{e} \in C_2} N_{stm}(\vec{e}) \quad + \quad \alpha_0 \sum_{\vec{e} \in C_2} F(\vec{e}) \tag{1.15}$$

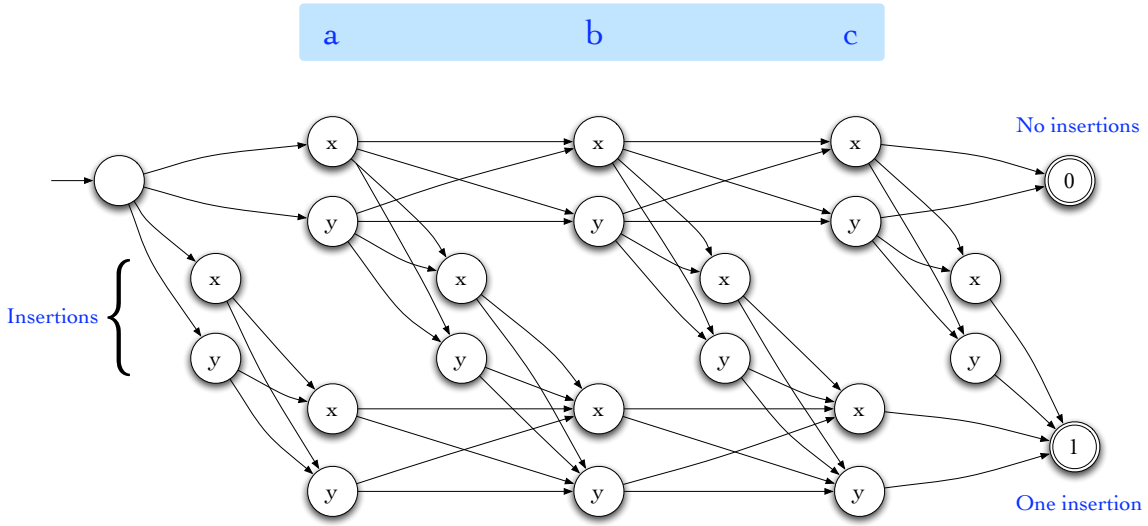

Figure 1-3: **WFSA** $A(abb)$**.** States correspond to following sets of edits: $\{(a,x),(a,y),(b,x),(b,y),(\epsilon,x),(\epsilon,y),(\epsilon,\epsilon)\}$. The top two rows correspond to substitution states for which no insertion has yet occurred. The middle two rows correspond to insertion states. The bottom two rows correspond to substitution states for which an insertion has already occurred.

Recall that set C_2 contains all edit-sequences $\vec{e}$ yielding a particular lost-language morpheme $\mathbf{u}^{stm}$ along with any $\mathbf{h}^{stm} \in \mathcal{H}_j^{stm}$. The left-hand sum is easy to compute, as we can ignore any edit-sequence absent in our current sample. (We simply enumerate the edit-sequences which have already been observed for identical stems: $\left\{(\vec{e}_{stm})_{k\neq i} \mid \mathbf{u}_k^{stm} = \mathbf{u}_i^{stm}\right\}$, and count how many times each appears.)

The right-hand sum is more difficult. Its computation requires summing over all edit-sequences in C_2, even those never seen before. A brute force computation would require: (i) the enumeration of all possible edit-sequences which yield $\mathbf{u}^{stm}$ (exponential in the length of $\mathbf{u}^{stm}$), and (ii) the removal of any such edit-sequence which yields a value $\mathbf{h}^{stm} \notin \mathcal{H}_j^{stm}$.

Fortunately, the function $F(\vec{e})$ (defined in equation 1.14) nearly factors over the individual edits $e \in \vec{e}$. This fact will allow us to compactly represent the set C_2 as a weighted finite-state acceptor (WFSA). We start by constructing a WFSA $A(\mathbf{u})$ which accepts *any string* $\mathbf{h}$ which can be jointly generated with $\mathbf{u}$ through a sequence of edits $\vec{e}$. In other words, the *language* of $A(\mathbf{u})$ is the set of strings $\left\{\mathbf{h} \mid \exists \vec{e} \ : \ y(\vec{e}) = (\mathbf{u}, \mathbf{h})\right\}$. Each state s of $A(\mathbf{u})$ will correspond to a single edit-operation e and incoming arcs

will be weighted by the appropriate factors of equation 1.14:

- If $e = (\epsilon, h)$, incoming arcs accept the symbol h and are weighted by $f(e)$.
- If $e = (u, \epsilon)$, incoming arcs accept the symbol ϵ and are weighted by $f(e)$.
- If $e = (u, h)$, incoming arcs accept the symbol h and are weighted by $f(e)$.
- If $e = (\epsilon, \epsilon)$, s is an end-state, incoming arcs accept ϵ, and are weighted by $f(e) \cdot \frac{q(n_I, n_D)}{n_S^{[n_I + n_D]}}$.

The final case requires some clarification, as the values n_I, n_D, and n_S count the total number of insertions, deletions, and substitutions throughout the entire edit-sequence. This is the only item in equation 1.14 which doesn't factor over individual edits. As a result, the states in the WFSA need to keep track of the number of previously performed insertions and deletions. Since our distribution $q(n_I, n_D)$ only allows a single insertion (and no deletions) per morpheme, we can simply double our set of substitution states to track whether the allowed insertion has occurred yet or not and provide a unique end-state for each possibility. See figure 1-3 for an example.

Thus, every path of arcs $\vec{a}$ through $A(\mathbf{u})$ corresponds to an edit-sequence $\vec{e}$, with path arc-weights which satisfy: $\prod_{a \in \vec{a}} w(a) = F(\vec{e})$. However, $A(\mathbf{u})$ is not yet restrictive enough for our purposes. It accepts some strings $\mathbf{h} \notin \mathcal{H}_j^{stm}$. Set C_2, on the other hand, is restricted to edit-sequences which yield actual known-language morphemes $\mathbf{h}^{stm} \in \mathcal{H}_j^{stm}$. To add this restriction, we construct a *lexicon* acceptor A_j^{stm}. This WFSA accepts all and only those strings in the lexicon $\mathcal{H}_j^{stm}$ (with all arc weights set to 1).

We construct this lexicon WFSA by enumerating each morpheme $\mathbf{h} \in \mathcal{H}_j^{stm}$ and adding a separate path through A_j^{stm} which sequentially accepts the letters of $\mathbf{h}$. Initially, A_j^{stm} will be quite large due to many redundant states (e.g. the paths accepting strings *yyy* and *yyyx* would be entirely disjoint). However, after its initial construction, we apply the Hopcroft minimization algorithm [57] which yields an equivalent, but optimally compact, WFSA (e.g. the paths accepting strings *yyyy* and *yyyx* would now share an initial prefix).

Next, we intersect the optimized A_j^{stm} with $A(\mathbf{u})$ to produce a new WFSA $A_j^{stm}(\mathbf{u})$. This operation essentially *prunes* our original $A(\mathbf{u})$ and restricts its paths to those which correspond to edit-sequences which yield a string in the lexicon $\mathcal{H}_j^{stm}$. (The original arc weights of $A(\mathbf{u})$ remain unaffected.) Finally, we are ready to compute the right-hand sum in equation 1.15 as the sum of path-weights:

$$\sum_{\vec{e} \in C_2} F(\vec{e}) = \sum_{\vec{a}} \prod_{a \in \vec{a}} w(a)$$

where each $\vec{a}$ is a unique sequence of arcs from the start-state of $A_j^{stm}(\mathbf{u})$ to an end-state. Since this WFSA contains no cycles, its path-weight sum can easily be computed with a dynamic program. To each state s_k, we associate a value β_k with the following recursive definition:

$$\beta_k = \begin{cases} 1 & \text{if } s_k \text{ is an end state,} \\ \sum_{a \,:\, s_k \to s_\ell} w(a) \cdot \beta_\ell & \text{otherwise.} \end{cases} \tag{1.16}$$

Intuitively, β_k gives the total path-weight of paths starting at state s_k and ending at an end-state. Values can easily be computed by starting with the end-states and stepping backwards along arcs. The resulting value β_0, corresponding to the start-state s_0, gives the desired sum. This construction and computation can be similarly carried out for the other two sums in equation 1.12.

Sampling Edit-sequences

Once we've sampled a segmentation $\mathbf{u}_i = \mathbf{u}^{pre}\, \mathbf{u}^{stm}, \mathbf{u}^{suf}$ and part-of-speech $\text{pos}_i = j$, we turn to the next step: sampling the actual edit-sequences $\vec{e}_{pre}, \vec{e}_{stm}, \vec{e}_{suf}$. We start

by examining their posterior probabilities:

$$
\begin{aligned}
& P(\vec{e}_{pre}, \vec{e}_{stm}, \vec{e}_{suf} \mid \mathbf{u}^{pre}, \mathbf{u}^{stm}, \mathbf{u}^{suf}, j, \mathcal{H}) \\
& \quad \propto \quad P(\mathbf{u}^{pre}, \mathbf{u}^{stm}, \mathbf{u}^{suf}, \mathcal{H} \mid \vec{e}_{pre}, \vec{e}_{stm}, \vec{e}_{suf}, j)\ P(\vec{e}_{pre}, \vec{e}_{stm}, \vec{e}_{suf} \mid j) \\
& \quad \propto \quad \begin{cases} P(\vec{e}_{pre}|j) P(\vec{e}_{stm}) P(\vec{e}_{suf}|j) & \text{if } \vec{e}_{pre}, \vec{e}_{stm}, \vec{e}_{suf} \in C_1 \times C_2 \times C_3, \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}
$$

The sets C_1, C_2, C_3 are defined as in equation 1.12 to include all edit-sequences which yield the fixed lost-language morphemes $\mathbf{u}^{pre}, \mathbf{u}^{stm}, \mathbf{u}^{suf}$ along with known-language morphemes in the vocabulary $\mathcal{H}$. Now, instead of summing over these sets, we instead need to sample from them. We focus here on the sampling procedure for $\vec{e}_{stm}$, but similar reasoning applies to $\vec{e}_{pre}$ and $\vec{e}_{suf}$.

The individual probability term $P(\vec{e})$ has already been derived in equation 1.13, but we repeat it here for convenience:

$$P(\vec{e}) \ \propto \ N_{stm}(\vec{e}) + \alpha_0 \cdot F(\vec{e})$$

We break our sampling procedure into two steps. First we decide whether to draw our sample $\vec{e}$ according to the left-hand term (from an existing table in the "Chinese restaurant"), or according to the right-hand term (the base distribution). To make this decision, we calculate $p = \sum_{\vec{e} \in C_2} N_{stm}(\vec{e})$ and $Q = p + \alpha_0 \cdot \sum_{\vec{e} \in C_2} F(\vec{e})$. To calculate p, we need merely consider values which have already been observed for identical stems: $\left\{(\vec{e}_{stm})_{k \neq i} \mid \mathbf{u}_k^{stm} = \mathbf{u}_i^{stm}\right\}$. To calculate Q, we sum all the path-weight of WFSA $A_j^{stm}(\mathbf{u}^{stm})$ using the dynamic program derived in the previous section.

We then sample a Bernoulli random variable with parameter p/Q. If we draw a heads, we proceed to sample from the already observed edit-sequences $\left\{(\vec{e}_{stm})_{k \neq i} \mid \mathbf{u}_k^{stm} = \mathbf{u}_i^{stm}\right\}$ in proportion to the number of times they have each been observed: $N_{stm}(\vec{e})$. If the Bernoulli comes up tails, we proceed to sample from all edit-sequences

$\vec{e} \in C_2$ according to $F(\vec{e})$. In other words, we wish to draw $\vec{e}$ with probability:

$$p(\vec{e}) = \begin{cases} \frac{F(\vec{e})}{Z} & \text{if } \vec{e} \in C_2, \\ 0 & \text{otherwise.} \end{cases} \tag{1.17}$$

As before, we utilize our WFSA $A_j^{stm}(\mathbf{u}^{stm})$. Recall that each unique path through this WFSA corresponds to an edit-sequence $\vec{e} \in C_2$ with path-weight $F(\vec{e})$. As before, we employ the values β_k defined recursively in equation 1.16. This time, we sample a path arc-by-arc from the start-state s_0 until an end-state is reached. When we are in state s_k, we sample the next arc $a : s_k \rightarrow s_\ell$ according to:

$$P(a : s_k \rightarrow s_\ell) \;\propto\; w(a) \cdot \beta_\ell$$

This procedure results in a draw of $\vec{e} \in C_2$ with the desired probability. To see this, we can rewrite $p(\vec{e})$ using the chain rule as $\prod_i p(e_i \mid e_1, \ldots, e_{i-1})$. We can further rewrite each conditional probability by marginalizing over all possible edit-sequence completions, and then plugging in the definitions of $p(\vec{e})$ (equation 1.17) and $F(\vec{e})$ (equation 1.14):

$$\begin{aligned} p(e_i \mid e_1, \ldots, e_{i-1}) \quad &= \sum_{e_{i+1} \ldots e_m} p(e_i, \ldots, e_m \mid e_1, \ldots, e_{i-1}) \\ &\propto \sum_{e_{i+1} \ldots e_m \,:\, e_1 \ldots e_m \in C_2} \left[\frac{q(n_I, n_D)}{n_S^{[n_I + n_D]}} \cdot \prod_{j=i}^{m} f(e_j) \right] \end{aligned}$$

Switching to the equivalent arc-view in the WFSA, we get:

$$\begin{aligned} p(a_i : s_k \rightarrow s_\ell \mid a_1, \ldots, a_{i-1}) \;&\propto \sum_{a_{i+1} \ldots a_m \,:\, a_1 \ldots a_m \in A_j^{stm}(\mathbf{u}^{stm})} \left[\prod_{j=i}^{m} w(a_j) \right] \\ &= \; w(a_i) \cdot \beta_\ell, \end{aligned}$$

giving us the proposed sampling formula.

1.7.2 Sampling Cognate Indicators

For each word $\mathbf{u}_i$, we sample a corresponding cognate indicator variable c_i. Recall that $c_i = 1$ indicates that $\mathbf{u}_i$ was generated along with a latent known-language cognate $\mathbf{h}_i$ (via edit-sequences $\vec{e}_{pre}, \vec{e}_{stm}, \vec{e}_{suf}$). Value $c_i = 0$ indicates that lost-language word $\mathbf{u}_i$ was generated alone according to a lost-language language model. The posterior for c_i is given by:

$$P(c_i \mid \mathbf{u}_i, \mathcal{H}) \;\propto\; P(\mathbf{u}_i \mid c_i, \mathcal{H}) \cdot P(c_i)$$

Thus we wish to to simple a Bernoulli random variable in proportion to the two values:

$$P(\mathbf{u}_i \mid c_i = 0, \mathcal{H}) \cdot P(c_i = 0)$$
$$P(\mathbf{u}_i \mid c_i = 1, \mathcal{H}) \cdot P(c_i = 1)$$

In our model we treat $P(c_i = 1)$ as a fixed parameter (set to 0.5 in our experiments). The value $P(\mathbf{u}_i \mid c_i = 0, \mathcal{H}) = P(\mathbf{u}_i \mid c_i = 0)$ is given by the lost-language character language model of equation 1.11. The language model parameters are fixed using the observed character frequencies in the lost-language corpus. Finally, we calculate $P(\mathbf{u}_i \mid c_i = 1, \mathcal{H})$ by marginalizing over all segmentations and parts-of-speech of $\mathbf{u}_i$:

$$P(\mathbf{u}_i \mid c_i = 1, \mathcal{H}) = \sum_{\mathbf{u}^{pre}, \mathbf{u}^{stm}, \mathbf{u}^{suf}, j} P(\mathbf{u}^{pre}, \mathbf{u}^{stm}, \mathbf{u}^{suf}, j \mid \mathbf{u}_i, \mathcal{H})$$

The terms in this sum are identical to equation 1.12 and are calculated in the same manner. The sum itself is calculated through explicit enumeration of all segmentations $\mathbf{u}^{pre}, \mathbf{u}^{stm}, \mathbf{u}^{suf} = \mathbf{u}_i$ and parts-of-speech $j \in 1 \ldots M$.

1.7.3 Resampling Chinese Restaurant Tables

Our sampling of individual word analyses can be viewed as inducing a three-part *clustering* of words, based on prefix, stem, and suffix edit-sequences. Consider the set of currently sampled stem edit-sequences: $E = \left\{\vec{e} \mid \exists i : (\vec{e}_{stm})_i = \vec{e}\right\}$. For each

such edit-sequence $\vec{e} \in E$, we define its *cluster* as $c(\vec{e}) = \left\{ i \mid (\vec{e}_{stm})_i = \vec{e} \right\}$. Thus $c[E]$ (the image of set E under function c) defines a partition over instances based on their sampled stem edit-sequences. In the Chinese restaurant metaphor, $c[E]$ gives us the set of sampled tables.

In the standard Gibbs sampling scenario, it can be difficult for these clusters to mix properly. To see this, consider a stem edit-sequence $\vec{e}$ and its cluster of instances $c(\vec{e})$. Single instances $i \in c(\vec{e})$ may shift in and out of the cluster either due to sampling a different edit-sequence $\vec{e'}$ for $\mathbf{u}_i^{stm}$ or due to segmenting $\mathbf{u}_i$ in a new way. However, changing the edit-sequence of the entire cluster would require individually sampling a new value *separately* for each instance $i \in c(\vec{e})$. In essence, this would require the temporary fragmentation of the cluster and its eventual rebuilding. So even if some other value $\vec{e'}$ is a much more likely candidate for $c(\vec{e})$, reaching it may involve passing through a very low probability sample path.

In order to avoid this problem, we introduce an additional procedure in which we explicitly resample each cluster's edit-sequence from the base distribution, conditioned on the clustering itself. Thus, for each cluster $c \in c[E]$ with a stem morpheme $\mathbf{u}^{stm}$ and part-of-speech j, we resample according to:

$$P(\vec{e}_{stm} | \mathbf{u}^{stm}, \mathcal{H}) \propto \begin{cases} F(\vec{e}) & \text{if for } y(\vec{e}) = (\mathbf{u}, \mathbf{h}) : \ \mathbf{u} = \mathbf{u}^{stm} \ \wedge \ \mathbf{h} \in \mathcal{H}_j^{stm}, \\ 0 & \text{otherwise,} \end{cases}$$

where $F(\vec{e})$ is the marginalized posterior base distribution previously given in equation 1.14.[9] As in section 1.7.1, we sample a path through the WFSA $A_j^{stm}(\mathbf{u}^{stm})$ using the techniques discussed after equation 1.17. Once a new edit-sequence $\vec{e'}$ has been sampled, it is assigned to all the instances in cluster c. Similar operations are performed for the clusters induced by prefix and suffix assignments.

[9] With the caveat that the cached counts now exclude all instances in the cluster under consideration.

1.7.4 Sampling Sparsity Indicators

Recall from section 1.6.2 that for each pair of characters (u, h) in the lost and know languages, we posit a sparsity indicator variable $\lambda_{(u,h)}$. Intuitively, $\lambda_{(u,h)} = 1$ indicates that u and h represent historically related phonemes which often substitute for one another in cognate pairs. Formally, $\lambda_{(u,h)}$ determines the value of $v_{(u,h)}$, the Dirichlet hyperparameter corresponding to base distribution probability $\rho_{(u,h)}$ (see equation 1.9 and figure 1-2 for the resulting effect on $\rho_{(u,h)}$). We start by deriving the posterior probability for a given joint setting of $\vec{\lambda}$. We indicate the other sampled variables (including all word analyses and cognate indicators) with an ellipsis "...":

$$\begin{aligned} P(\vec{\lambda} \mid \ldots) \;&\propto\; P(\vec{\lambda}) \cdot P(\ldots \mid \vec{\lambda}) \\ &\propto\; g(\vec{\lambda}) \cdot \frac{\prod_{\text{sub } e} v_e^{[N(e)]}}{\left(\sum_{\text{sub } e} v_e\right)^{[N_S]}} \end{aligned}$$

The first factor $g(\vec{\lambda})$ is the unnormalized structural sparsity prior given in equation 1.1. The second factor is the predictive probability of the base-distribution (with Dirichlet prior hyperparameters v_e) when the multinomial parameters (ρ_e) have been marginalized out. As before in equation 1.14, $N(e)$ denotes the number of times edit-operation e has occurred in the *unique* edit-sequences obtained from each morpheme-pair distribution $(G^{stm}, G_1^{pre}, \ldots)$; or more precisely, the number of times e appears among the sets:

$$\begin{aligned} E^{stm} &= \left\{(\vec{e}_{stm})_{k \neq i}\right\} \\ E_j^{pre} &= \left\{(\vec{e}_{pre})_{k \neq i} \mid \text{pos}_k = j\right\},\ \forall j \in 1 \ldots M \\ E_j^{suf} &= \left\{(\vec{e}_{suf})_{k \neq i} \mid \text{pos}_k = j\right\},\ \forall j \in 1 \ldots M \end{aligned}$$

and N_S give the *total* number of substitutions among those sets. The notation $x^{[n]}$ represents the rising factorial: $x(x+1)\cdots(x+n-1)$.

In order to speed mixing of our sampler, we jointly sample blocks of sparsity indicator variables. In particular, for each lost-language letter u, we jointly sample

all variables involving u: $\left\{\lambda_{(u,h)} \mid \forall h\right\}$ (in the binary matrix view, a *row* of $\vec{\lambda}$), and for each known-language letter h, we jointly sample: $\left\{\lambda_{(u,h)} \mid \forall u\right\}$ (a *column*). To do so, we enumerate all possible values $\vec{\lambda}, \vec{\lambda'}, \vec{\lambda''}, \ldots$ (keeping fixed the values for λ_e's not being sampled), compute their posteriors, and sample from the resulting discrete distribution. Note that we can avoid enumerating settings of $\vec{\lambda}$ which are assigned probability zero by the structural sparsity prior (e.g. where $\sum_h \lambda_{(u,h)} > 3$). As a result, we very rarely need consider more than $\binom{30}{2} = 435$ possible values, and often many fewer suffice.

1.7.5 Prediction

The output of our sampler (algorithm 1) is a set of latent variable values sampled from the posterior distribution of our model. In this section we describe how we make our final model predictions on the basis of these samples. As a reminder, sampled variables include:

- Word analyses for each instance i:

$$\left\{\left[\vec{e}_{pre}, \vec{e}_{stm}, \vec{e}_{suf}, \mathbf{u}^{pre}, \mathbf{u}^{stm}, \mathbf{u}^{suf}, \mathbf{h}^{pre}, \mathbf{h}^{stm}, \mathbf{h}^{suf}, \text{pos}\right]_i^{(r)}\right\}_{r=1}^{1000}$$

- Cognate indicator variables for each instance i:

$$\left\{c_i^{(r)}\right\}_{r=1}^{1000}$$

- Sparsity indicator variables:

$$\left\{\left\{\lambda_{(u,h)} \mid \forall (u,h)\right\}^{(r)}\right\}_{r=1}^{1000}$$

We use the following procedure to induce our model's final predictions:

1. Predict the set of sparsity indicator variables $\left\{\lambda_{(u,h)} \mid \forall (u,h)\right\}^{(*)}$ which occurs most frequently among sampled values.

2. Collect sets of *unique* edit-sequences over all sampling rounds:

$$
\begin{aligned}
E^{stm} &= \left\{\vec{e} \mid \exists(i,r) : (\vec{e}_{stm})_i^{(r)} = \vec{e}\right\} \\
E_j^{pre} &= \left\{\vec{e} \mid \exists(i,r) : (\vec{e}_{pre})_i^{(r)} = \vec{e} \ \wedge \ \text{pos}_i = j\right\}, \ \forall j \in 1 \ldots M \\
E_j^{suf} &= \left\{\vec{e} \mid \exists(i,r) : (\vec{e}_{suf})_i^{(r)} = \vec{e} \ \wedge \ \text{pos}_i = j\right\}, \ \forall j \in 1 \ldots M
\end{aligned}
$$

3. Define a base distribution $F^*(\vec{e})$ as in equation 1.14 except with counts now based on the preceding collection of sets (i.e. over all sampling rounds).

4. For each instance i:

   (a) Predict cognate indicator value $c^{(*)}$ which occurs most frequently among sampled values for instance i.

   (b) Predict segmentation $\left[\mathbf{u}^{pre}, \mathbf{u}^{stm}, \mathbf{u}^{suf}\right]^{(*)}$ which occurs most frequently among sampled values for instance i.

   (c) Define constraint sets based on the predicted segmentation and the known-language lexicon:

$$
\begin{aligned}
C_j^{pre} &= \left\{\vec{e} \mid y_1(\vec{e}) = [\mathbf{u}^{pre}]^{(*)} \ \wedge \ y_2(\vec{e}) \in \mathcal{H}_j^{pre}\right\}, \ \forall j \in 1 \ldots M \\
C_j^{stm} &= \left\{\vec{e} \mid y_1(\vec{e}) = [\mathbf{u}^{stm}]^{(*)} \ \wedge \ y_2(\vec{e}) \in \mathcal{H}_j^{stm}\right\}, \ \forall j \in 1 \ldots M \\
C_j^{suf} &= \left\{\vec{e} \mid y_1(\vec{e}) = [\mathbf{u}^{suf}]^{(*)} \ \wedge \ y_2(\vec{e}) \in \mathcal{H}_j^{suf}\right\}, \ \forall j \in 1 \ldots M
\end{aligned}
$$

   (d) Predict edit-sequences $[\vec{e}_{pre}, \vec{e}_{stm}, \vec{e}_{suf}]^{(*)} =$

$$
\begin{aligned}
&\underset{\vec{e}_{pre},\vec{e}_{stm},\vec{e}_{suf}}{\text{argmax}} \quad F^*(\vec{e}_{pre}) \cdot F^*(\vec{e}_{stm}) \cdot F^*(\vec{e}_{suf}) \\
&\quad \text{s.t. for some } j\text{:} \\
&\qquad \vec{e}_{pre}, \vec{e}_{stm}, \vec{e}_{suf} \in C_j^{pre} \times C_j^{stm} \times C_j^{suf}
\end{aligned}
\tag{1.18}
$$

   (e) Predict cognate:

$$
\left[\mathbf{h}^{pre}\, \mathbf{h}^{stm}\, \mathbf{h}^{suf}\right]^{(*)} = y_2\big([\vec{e}_{pre}]^{(*)}\big)\ y_2\big([\vec{e}_{stm}]^{(*)}\big)\ y_2\big([\vec{e}_{suf}]^{(*)}\big)
$$

We compute the constrained maximum in equation 1.18 using the same finite-state machines used during sampling. Recall that we construct $A_j^{stm}(\mathbf{u})$ to be a WFSA for which every path $\vec{a}$ corresponds to an edit-sequence $\vec{e}$ with $y_1(\vec{e}) = \mathbf{u}$ and $y_2(\vec{e}) \in \mathcal{H}_j^{stm}$. We now weight the arcs so that $\prod_{a \in \vec{a}} w(a) = F^*(\vec{e})$. Thus computing the constrained maximum for sets $C_j^{pre}, C_j^{stm}, C_j^{suf}$ requires constructing corresponding finite-state acceptors A_1, A_2, A_3 and computing:

$$\max_{\vec{a} \in A_1} \prod_{a \in \vec{e}} w(a) \;\cdot\; \max_{\vec{a} \in A_2} \prod_{a \in \vec{e}} w(a) \;\cdot\; \max_{\vec{a} \in A_3} \prod_{a \in \vec{e}} w(a)$$

To find the maximum weight path through each WFSA, we use the same dynamic program given in equation 1.16, merely replacing the summation with a maximum:

$$\beta_k = \begin{cases} 1 & \text{if } s_k \text{ is an end state,} \\ \max_{a \,:\, s_k \to s_\ell} w(a) \cdot \beta_\ell & \text{otherwise.} \end{cases} \tag{1.19}$$

Intuitively, β_k gives the maximum path-weight of paths starting at state s_k and ending at an end-state. Values can again be computed by starting with the end-states and progressing backwards along arcs. The resulting value β_0, corresponding to the start-state s_0, gives the desired maximum. To retrieve the actual maximizing path, backpointers are stored during each step of the dynamic program.

1.7.6 Implementation Details

This section describes implementation details that are necessary to reproduce our experiments.

Computational Details

Most steps in our sampling algorithm simply require us to compute counts over our currently drawn sample of variables. Instead of recomputing these values on the fly, we keep a persistent cache of counts which is incrementally updated after each sampling step. Thus, most sampling steps simply require a constant-time lookup in

a hashtable and a quick computation of probabilities.

When sampling lost-language word segmentations, we do explicitly enumerate all possible segmentations and parts-of-speech. However, we use an extremely coarse notion of part-of-speech (only 4, see 1.8.1 for details) and we cap the length of prefixes and suffixes to three characters (in line with what we observe for the known-language prefix and suffix length). Thus, even for very long words, we need only consider $4^3 = 64$ possibilities.

Many steps during sampling require the construction of weighted finite-state automata and the computation of dynamic programs over these automata. Each WFSA is of polynomial-size in the length of the corresponding lost-language morpheme $\mathbf{u}$. The required dynamic programs are all linear-time in the size of the corresponding WFSA. The construction of each WFSA, however, can be expensive, as it requires intersecting a lost-language morpheme WFSA $A(\mathbf{u})$ with a much larger WFSA representing the known-language lexicon. In order to avoid unnecessary computation, we cache the resulting WFSA and store it for future use (simply reweighting the arcs as necessary).

For example, the first time we compute probabilities for lost-language prefix $\mathbf{u}^{pre}$ with part-of-speech j, we intersect $A(\mathbf{u}^{pre})$ with A_j^{pre}. We use the resulting WFSA and store it in our cache. The next time we encounter prefix $\mathbf{u}^{pre}$ with part-of-speech j, we retrieve the WFSA and reweight its arcs according to our current cache of counts.

Initialization and Pruning

We initialize our latent variables with results from the HMM baseline (see 1.9). In particular, the baseline provides us with letter substitution probabilities $P(h|u)$. First, we prune our search space by ruling out all substitutions (u, h) which are given probability < 0.05. We then initialize insertion probabilities $P(u)$ based on the frequencies of known-language letters. As another pruning step, we rule out insertions of all but the two most frequent letters. The result is a string-edit distribution $S(\vec{e})$. We then use this distribution to initialize our word analyses. In particular, for each

word $\mathbf{u}_i$, we consider all segmentations and parts-of-speech, and for each compute the constrained maximum given in equation 1.18 (replacing $F^*(\vec{e})$ with our initializing distribution $S(\vec{e})$). We initialize $\mathbf{u}_i$ with the resulting values. As a final pruning step, the character substitutions for each letter u are restricted to a single letter h, if after the initialization round (u, h) is found to occur more than five times more than any (u, h').

We initialize all cognate indicator variables c_i to 1, and we initialize the sparsity indicator variables $\lambda_{(u,h)}$ to be the character-mapping predictions made by the baseline (in particular, Baseline 1 in section 1.9).

Hyperparameter Values

Finally, we list all the values used for fixed hyperparameters:

- $\vec{b} = (b_i = 0, \forall i)$ // Structural sparsity parameters (equation 1.2)
- $\vec{b'} = (b'_0 = 0, b'_1 = 0, b'_2 = 7, b'_3 = 1, b'_{>3} = 0)$
- $\vec{w} = (w_0 = -\infty, w_1 = 0, w_2 = -50, w_{>2} = -\infty)$
- $K = 50$ // Spike-and-slab parameter (equation 1.9)
- $q(N_I, N_D) = \begin{cases} 0.5 & \text{if } N_I, N_D = (0, 0) \\ 0.5 & \text{if } N_I, N_D = (1, 0) \\ 0 & \text{otherwise} \end{cases}$ // Base distribution (equation 1.4)
- $\alpha_0 = 1000$ // Concentration parameter (section 1.6.2)
- $P(c_i = 1) = 0.5$ // Cognate prior (section 1.6.2)

1.8 Experiments

In this section we describe experiments applying our model to the ancient Ugaritic language (see section 1.4 for background) with biblical Hebrew as the observed known

language. In section 1.8.1 we describe the Ugaritic corpus and gold-standard annotations; in section 1.9 we describe our evaluation tasks and baseline, and in the remaining sections we describe our various experiments and results.

1.8.1 Corpus and Annotations

Our undeciphered corpus consists of an electronic transcription of the Ugaritic tablets [28]. This corpus contains 7,386 unique word-forms. As our known language corpus, we use the Hebrew Bible, which is both geographically and temporally close to Ugaritic. To extract a Hebrew morphological lexicon we assume the existence of manual morphological and part-of-speech annotations [45]. We divide Hebrew stems into four main part-of-speech categories each with a distinct affix profile: Noun, Verb, Pronoun, and Particle. For each part-of-speech category, we automatically determine the set of allowable affixes using the annotated Bible corpus.

To evaluate the output of our model, we annotated the words in the Ugaritic lexicon with the corresponding Hebrew cognates found in the standard reference dictionary [30]. In addition, manual morphological segmentation was carried out with the guidance of a standard Ugaritic grammar [101]. Although Ugaritic is an inflectional rather than agglutinative language, in its written form (which lacks vowels) words can easily be segmented (e.g. *wyplṯn* becomes *wy-plṯ-n*). Note that our analysis allows only a single prefix or suffix string, and as a result multiple prefixes or suffixes are combined into a single string.

Overall, we identified Hebrew cognates for 2,214 word forms, covering almost one-third of the Ugaritic vocabulary. We are confident that a majority of Ugaritic words with known Hebrew cognates were thus identified. The remaining Ugaritic words include many personal and geographic names, words with cognates in other Semitic languages, and words whose etymology is uncertain.

Since our annotation was performed at the vocabulary-level rather than the text-level, we faced the problem of word ambiguity. A single Ugaritic word-form can often be identified with several potential Hebrew cognates, depending on actual context. For example, the Ugaritic word-form *bth* could be identified with at least four He-

	Number of Hebrew Cognates						
	0	1	2	3	4	5	6
Number of Ugaritic Word-forms	5172	1677	359	160	14	1	3

Table 1.1: Number of Ugaritic word-forms with various numbers of identified Hebrew cognates.

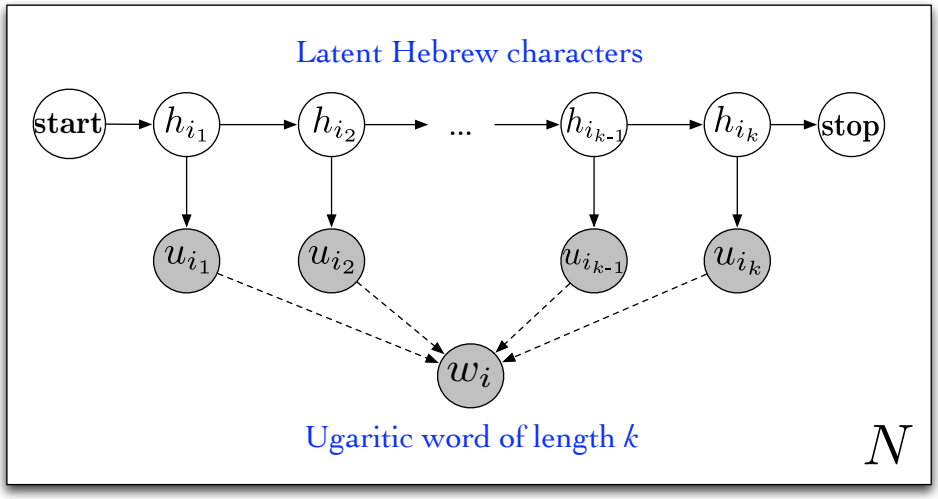

Figure 1-4: **Plate diagram of the baseline HMM model** (shown here as a first-order rather than second-order HMM for simplicity). Each of N observed Ugaritic word-forms w_i is determined by its observed character sequence. Each such character is generated by a latent Hebrew letter. Hebrew character transition distributions are estimated directly from transition counts in the Hebrew Bible. Emission distributions and the latent Hebrew characters are estimated using EM.

brew cognates: *bth* (her daughter), *btw* (his daughter), *byth* (her house), or *bytw* (his house). In such cases, we annotated the Ugaritic word form with all Hebrew cognate possibilities. Table 1.1 gives the number of Ugaritic word-forms with various numbers of identified Hebrew cognates.

1.9 Results

In the following section we evaluate our model on three separate decipherment tasks: (i) Learning alphabetic mappings, (ii) Cognate decipherment, and (iii) Cognate identification.

As a baseline for these tasks, we use the HMM-based method of [66] for learning letter substitution ciphers. In its original setting, this model was used to automatically map the written form of a language to its spoken form, under the assumption that each written character was emitted from a hidden phonemic state. In our adaptation, we assume instead that each Ugaritic character was generated by a hidden Hebrew letter, and that each hidden Hebrew letter was generated by the previous two Hebrew letters (a second-order character-level HMM). See figure 1-4 for a graphical depiction of this baseline. Hebrew character trigram transition probabilities are estimated using counts from the Hebrew Bible, and Hebrew to Ugaritic character emission probabilities are learned by applying EM to the Ugaritic vocabulary (see [66] for details). Finally, the highest probability sequence of latent Hebrew letters is predicted for each Ugaritic word-form, using the Viterbi algorithm.

1.9.1 Alphabetic Mapping

The first essential step towards successful decipherment is recovering the mapping between the symbols of the lost language and the alphabet of a known language. Although the exact phonetic values of letters in ancient scripts can never be known with complete certainty, it is possible to recover the historical relationships between phonemes (and thus alphabets) of related languages using the comparative method [19]. As a gold standard for our comparison, we use the well-established historical relationship between the sounds of the Ugaritic and Hebrew alphabets [55]. In particular, we wish to automatically recover pairs of *reflexes* in two languages – that is, pairs of letters whose corresponding phonemes descend from a common phoneme in an ancestral language (in this case Proto-Semitic).

This mapping is not one-to-one but is generally quite sparse. Of the 30 Ugaritic symbols, 27 map almost exclusively to a single Hebrew letter, and the remaining three map to two Hebrew letters. The Hebrew alphabet contains 23 letters, of which three map to three Ugaritic letters, four map to two Hebrew letters, and the remainder map to a single Ugaritic letter. See table A.2 in appendix A for the gold standard alphabetic mapping used in evaluation.

We recover our model's predicted alphabetic mappings by simply examining the predicted values of the binary indicator variables $\lambda_{u,h}$ for each Ugaritic-Hebrew letter pair (u, h). Due to our structural sparsity prior $P(\vec{\lambda})$, the predicted mappings are quite sparse: all Ugaritic letters maps to a single Hebrew letter, 17 Hebrew letters map to a single Ugaritic letter, five Hebrew letters map to two Ugaritic letters, and one Hebrew letter maps to three Ugaritic letters. See table A.3 for the predicted alphabetic mappings.

To recover alphabetic mappings from the HMM substitution cipher baseline, we consider several possibilities.

Baseline 1: For each Hebrew letter h, we can simply choose the single Ugaritic letter u with highest emission probability: $u = \text{argmax}_{u'} P(u'|h)$. Table A.5 gives the predictions under this baseline. Notice that this procedure results in many Ugaritic letters that are not mapped to any Hebrew letter at all.

Baseline 2: For each Ugaritic letter u, we can simply choose the single Hebrew letter h such that: $h = \text{argmax}_{h'} P(h'|u) \propto P(u|h')P(h')$. Table A.4 gives the predictions under this baseline. This procedure guarantees that all Ugaritic letters are mapped to a single Hebrew letter. However, three Hebrew letters remain unmapped.

Baseline 3: This procedure simply combines the procedures of Baselines 1 and 2. A mapping between (u, h) is predicted if (u, h) are mapped under either (or both) of the first two baselines. This procedure results in a many-to-many mapping where every letter in each alphabet is guaranteed to map to at least one letter in the other. Table A.6 gives the predictions under this baseline.

To evaluate these predicted mappings, we consider several metrics. Our first, somewhat crude, measure is to simply count the number of Ugaritic letters that are correctly mapped to at least one of their Hebrew reflexes. Under this metric our model recovers correct mappings for 28 of 30 Ugaritic letters (yielding 93.3% accuracy), while the best baseline predictions (Baselines 2 and 3) yield correct mappings for 23 of 30 Ugaritic letters (76.7% accuracy).

	ACCURACY	PRECISION	RECALL	F1-MEASURE
Baseline 1	.57	.74	.52	.61
Baseline 2	.77	.77	.70	.73
Baseline 3	.77	.69	.73	.71
Our Model	.93	.93	.85	.89

Table 1.2: **Evaluation of alphabetic mappings** predicted by HMM baseline variations and our model. Column one (ACCURACY) simply counts the number of Ugaritic letters that are correctly mapped to at least one of their Hebrew reflexes. Columns 2-4 treats each possible character pair as an example in a binary classification problem.

Note that this first evaluation metric ignores the fact that the gold-standard mappings are many-to-many (though quite sparse). We can evaluate performance with greater sensitivity by instead treating each mapped character pair (u, h) as a positive prediction in a binary classification problem. Under this scenario, the gold-standard contains 33 positive examples out of 690 possible letter pairings. Results under both this and the first metric are given in table 1.2. Our model yields performance superior to the baselines on all measures, achieving F1-measure of .89, compared to .73 for the best baseline variant.

1.9.2 Cognate Decipherment

One of the primary goals of lost language decipherment is to accurately translate and understand ancient texts. An important step in this process is the recovery of cognate pairs between the lost language and a known related language. Cognate are words in sister languages that descend from a common word in a shared ancestral language. As such, cognates are often accurate translations of one another, or at least share common semantic features. As detailed in section 1.8.1, we manually identified gold-standard Hebrew cognates for about one-third of the Ugaritic word-forms. When multiple gold-standard cognates exist for an Ugaritic word-form (see table 1.1) we evaluate predictions using the Hebrew cognate which yields the best performance.

Cognate predictions for our model are produced by examining the predicted values of the latent Hebrew morphemes associated with each Ugaritic word-form. Even for words where we predict a cognate indicator variable $c_i = 0$ (indicating that the word

	WORDS				MORPHEMES			
	ACCURACY		EDIT-DISTANCE		ACCURACY		EDIT-DISTANCE	
	TYPE	TOKEN	TYPE	TOKEN	TYPE	TOKEN	TYPE	TOKEN
Baseline	.288	.460	1.261	.878	n/a		n/a	
Our Model	.630	.697	.501	.400	.763	.838	.337	.239
Prev Version	.604	.683	.552	.450	.740	.813	.369	.270
Only Sub	.599	.683	.529	.414	.745	.828	.323	.216
Only Mapped	.567	.648	.555	.436	.731	.799	.366	.276
No Cogs	.447	.560	.802	.660	.677	.760	.448	.341
No Spike	.471	.577	.722	.587	.710	.787	.400	.302
No Morph	.363	.554	1.175	.779	n/a		n/a	
Know Cogs	.710	.787	.422	.314	.834	.892	.231	.152

Table 1.3: **Evaluation of cognate decipherments** for the HMM baseline as well as our model. Two evaluation measures are used: 0-1 accuracy (higher is better) and Levenshtein edit-distance (lower is better). The unit of prediction can be either complete words or each of their three morphemic parts (prefix, stem, and suffix). Evaluation is carried out both at the word-form (TYPE) level, ignoring word frequency, as well as at the token-level, taking frequency into account. The first two rows show results for the baseline and our full model. The final seven rows show results for variants of our model (see section 1.9.4 for details).

does not have a Hebrew cognate), we still predict the most likely latent Hebrew word $\mathbf{h}_i^*$ by conditioning on $c_i = 1$. For the HMM baseline, we simply predict the most likely latent sequence of Hebrew characters for each Ugaritic word-form using the Viterbi algorithm.

We evaluate these cognate predictions using several measures. The simplest measure simply counts how many predicted Hebrew cognates exactly match one of the gold-standard cognates. As seen in table 1.3 (WORDS → ACCURACY → TYPE), our model achieves 63% accuracy on this measure, while the baseline achieves accuracy of 28.8%.

To gain a finer sense of how close our predictions are to the true Hebrew cognates, we also measure the Levenshtein edit-distance [72] between predicted and gold-standard cognates. This distance metric gives the minimal number of character edit-operations (substitutions, insertions, and deletions) needed to make an input string identical to a reference string. As seen in table 1.3 (WORDS → EDIT-DISTANCE → TYPE), our model's predictions are on average .5 edit-operations away

from a gold-standard cognate, whereas the baseline predictions require, on average, 1.26 edit-operations to match a true cognate.

Since our model also predicts prefix-stem-suffix morpheme boundaries, we can evaluate its decipherment performance on a per-morpheme basis as well. This metric can be both stricter and more lenient than per-word accuracy. For example, if *b-bt-w* were predicted in place of the correct *b-byt-w*, two out of three morpheme predictions would be judged as correct, whereas the per-word measure would judge this prediction as fully incorrect. On the other hand, we also now require that the morphemic segmentation boundaries be correct. For example, a prediction of *bbytw* (no prefix or suffix) would be judged as correctly predicting *none* of the morphemes of the correct *b-byt-w*, whereas the per-word evaluation would count this as a fully correct prediction.

As table 1.3 shows, per-morpheme performance (both in terms of simple accuracy and edit-distance) is consistently better than per-word performance. In fact, our model correctly deciphers over 3/4 of the morphemes on all Ugaritic word-forms with Hebrew cognates (MORPHEMES → EDIT-DISTANCE → TYPE). This metric is not available for the baseline, as it does not predict morpheme boundaries.

Besides carrying out these evaluations at the word-form-level (TYPE in table 1.3), we also investigated whether predictions were are more or less accurate for frequent words by evaluating at the token-level as well (TOKEN) in table 1.3). As shown, performance improves across the board at the token-level, indicating that more frequent words are easier to decipher than their infrequent counterparts.

1.9.3 Cognate Identification

The previous section evaluated our model's ability to decipher Ugaritic words which are, in fact, cognate to one or more Hebrew words. In this section we consider the problem of identifying which Ugaritic words have such cognates.

As before, we use our annotated Ugaritic corpus as a gold-standard (see section 1.8.1 and in particular table 1.1). About one-third of Ugaritic word-forms were identified as having known Hebrew cognates. We note that this is a conservative

gold-standard. Our knowledge of ancient Hebrew comes almost exclusively via the texts of the Hebrew Bible (about 300,000 tokens) and there are certainly many ancient Hebrew words which have been lost to history. In addition, it is possible that some Hebrew cognates were missed during the the annotation process (though we are confident that the great majority were included).

We evaluate our model's ability to identify cognates using the predicted values of the indicator variables c_i. Note also that for Ugaritic word-form $\mathbf{u}_i$: $P(c_i|\mathbf{u}_i) \propto P(\mathbf{u}_i|c_i) \cdot P(c_i)$. Since $P(c_i)$ (the cognate prior) is a fixed Bernoulli parameter of our model, we can vary its value to achieve different cognate prediction thresholds, allowing a trade-off between precision and recall.

As before, we compare our performance against the HMM substitution cipher baseline. To produce baseline cognate identification predictions, we calculate the probability (using the learned emission and transition parameters) of each Ugaritic word $\mathbf{u}_i$ given the latent Hebrew letter sequence predicted by the HMM. This probability can be regarded as measuring the likelihood that the given Ugaritic word-form $\mathbf{u}_i$ was generated by a latent Hebrew cognate: $P(\mathbf{u}_i|c_i = 1)$.[10] The probability that $\mathbf{u}_i$ was generated as a lone Ugaritic word, $P(\mathbf{u}_i|c_i = 0)$, is simply given by $(\frac{1}{31})^{length(\mathbf{u}_i)+1}$ (assuming a uniform distribution over the 30 Ugaritic letters and a special end symbol). Finally, as in our model, we assume a fixed cognate prior $P(c_i)$ and predict that $\mathbf{u}_i$ has a Hebrew cognate if:

$$\frac{P(\mathbf{u}_i|c_i = 1)}{P(\mathbf{u}_i|c_i = 0)} > \frac{P(c_i = 0)}{P(c_i = 1)}$$

As for our model, when the prior $P(c_i = 1)$ is set higher, we will detect more true cognates, but the number of false positives increases as well.

Finally, we compare our model's performance to that of our previously published version [109]. The primary difference between the model presented in this thesis and the previously published version is how we model the generation of non-cognate

[10] Or more precisely, this quantity should be regarded as a Viterbi approximation to $P(\mathbf{u}_i|c_i = 1)$ as it only accounts for the highest probability latent Hebrew letter sequence.

Ugaritic words: $P(\mathbf{u}_i|c_i = 0)$. In the previous publication we used a simple uniform distribution Ugaritic language model:

$$P(\mathbf{u}_i|c_i = 0) = \left(\frac{1}{31}\right)^{length(\mathbf{u}_i)+1} \tag{1.20}$$

Since our cognate generation sub-model $P(\mathbf{u}_i|c_i = 1)$ (detailed in section 1.6.2) leads to an exponential distribution over the length of Ugaritic words with latent Hebrew cognates, it is important that $P(\mathbf{u}_i|c_i = 0)$ display the same exponential decay on word length. Otherwise, the dominating factor in predicting c_i would be the length of $\mathbf{u}_i$ rather than the intrinsic plausibility of its joint generation with a Hebrew counterpart.

While definition 1.20 helps us avoid a length bias, it ignores the frequency of the Ugaritic characters in word $\mathbf{u}_i$. In contrast, the posterior of the cognate generation sub-model $P(\mathbf{u}_i|c_i = 1)$ is quite sensitive to the observed frequencies of the characters composing $\mathbf{u}_i$, generally assigning far lower probability to Ugaritic words with infrequent characters.[11] As a result, the previous version of our model displayed an unfortunate bias towards predicting that Ugaritic words with infrequent characters had no Hebrew cognates. To correct this bias, the model described in this thesis defines the distribution over non-cognate Ugaritic words using a unigram (rather than uniform) Ugaritic language model:

$$P(\mathbf{u}_i|c_i = 0) = P(END) \cdot \prod_{0<j\leq length(\mathbf{u}_i)} P\left(u_i[j]\right) \tag{1.21}$$

This definition allows a trade-off between $P(\mathbf{u}_i|c_i = 0)$ and $P(\mathbf{u}_i|c_i = 1)$ which focuses on the inherent posterior plausibility of $\mathbf{u}_i$ having been generated along with a Hebrew word $\mathbf{h}_i$ (i.e. in terms of consistent of character substitutions and morpheme matchings) rather than the length or character frequency of $\mathbf{u}_i$.

As cognate identification is a binary classification problem, we evaluate performance by plotting Receiver Operator Characteristic (ROC) and Precision-Recall

[11] This effect stems from the string-edit base distribution, which can be thought of as a character-level language model over bilingual character pairs.

curves. An ROC curve shows the achievable trade-offs between the False Positive Rate, defined as the fraction of all positive predictions which are actually negative

$$FPR = \frac{FP}{TP + FP},$$

and the True Positive Rate, defined as the fraction of all positive instances which are correctly predicted as positive:

$$TPR = \frac{TP}{TP + FN}.$$

A Precision-Recall curve shows the achievable trade-offs between Precision, defined as the fraction of all positive predictions which are in fact positive

$$Precision = \frac{TP}{TP + FP},$$

and Recall, which is identical to the True Positive Rate.

Figure 1-5 shows the ROC curve for our model, the ACL 2010 version of our model, and the HMM baseline. All three models achieve better than random performance for all possible operating points. The ROC curve of our previous model dominates the the curve of the HMM baseline, and likewise our current model dominates the performance of our previous model.

Figure 1-6 shows the Precision-Recall curve for the same three models. All three models show fluctuations in Precision when Recall is set very low. This may be due to the fact that the number of positive predictions in this setting is very small so the addition of a single false positive or true positive can have a great effect on Precision. For values of Recall above 0.1, the trend we observed in the ROC setting reemerges: Our previous model dominates the HMM baseline, and our current model dominates our previous model.

To gain some further insight we can graph F1-Measure as a function of Recall. F1-Measure is defined as the harmonic mean of Precision and Recall and is often used

as a unified measure of binary classification performance:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Figure 1-7 shows the various achievable F1 scores for the three models under consideration. The respective maximum achievable F1 scores are .473, .522, and .563 for the HMM baseline, our previous model, and our current model.

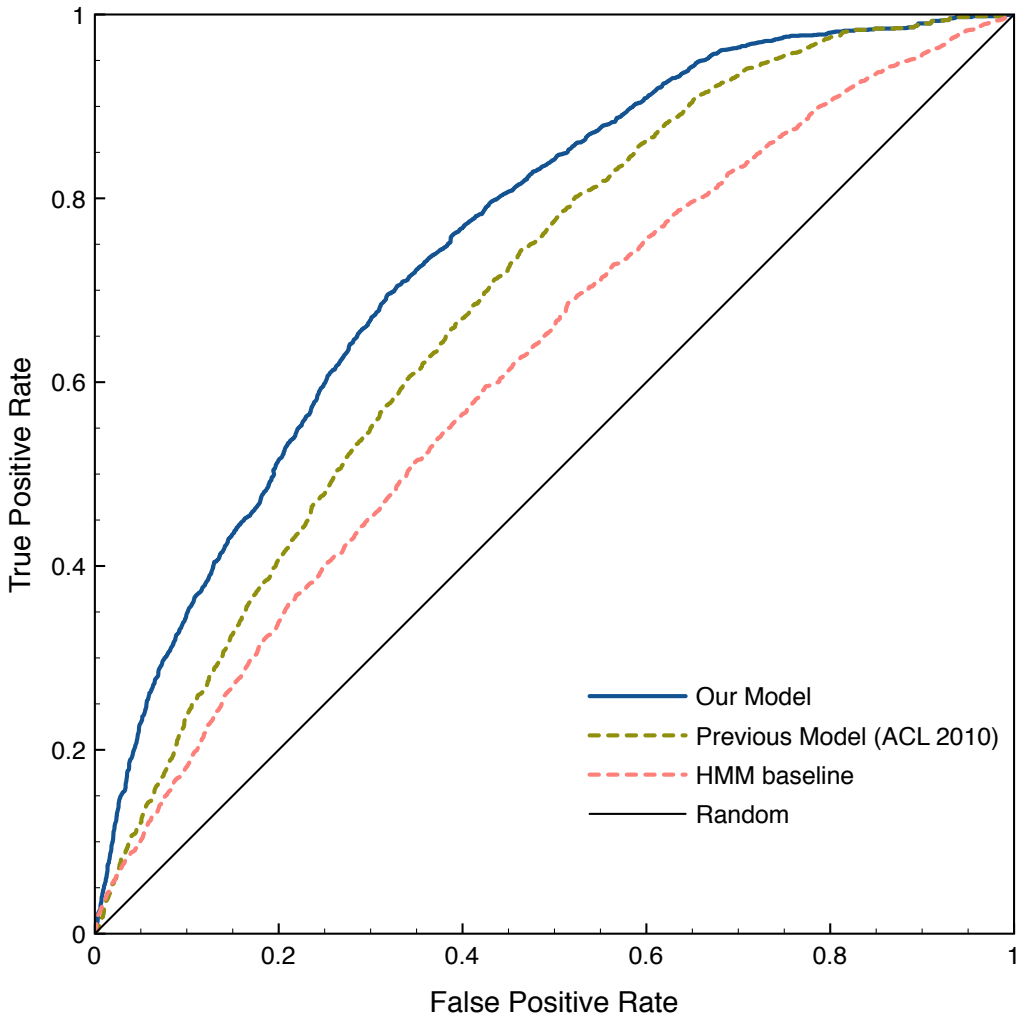

Figure 1-5: **Cognate identification ROC curves** for the model presented in this thesis (Our Model), the version of our model presented at ACL 2010 (using a uniform character model for non-cognate probabilities), and the HMM baseline.

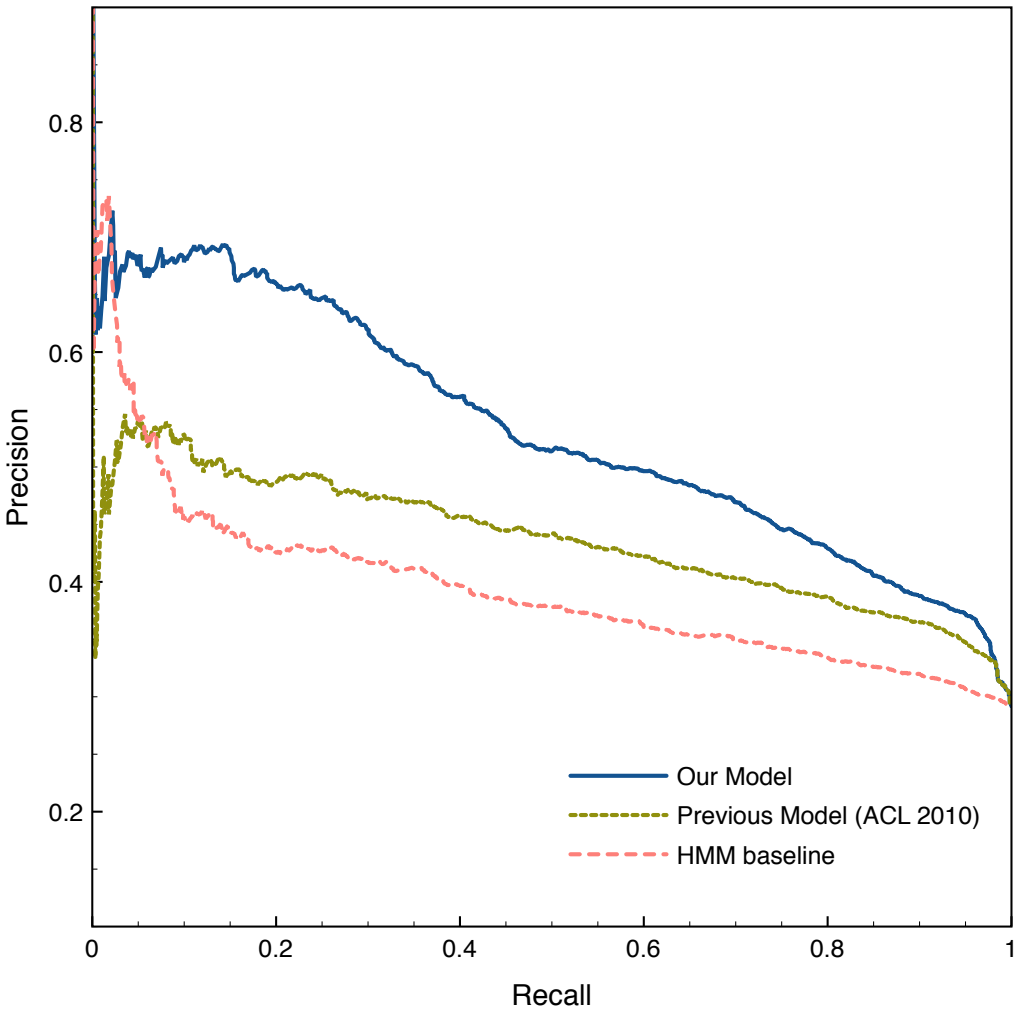

Figure 1-6: **Cognate identification Precision-Recall curves** for the model presented in this thesis (Our Model), the version of our model presented at ACL 2010 (using a uniform character model for non-cognate probabilities), and the HMM baseline.

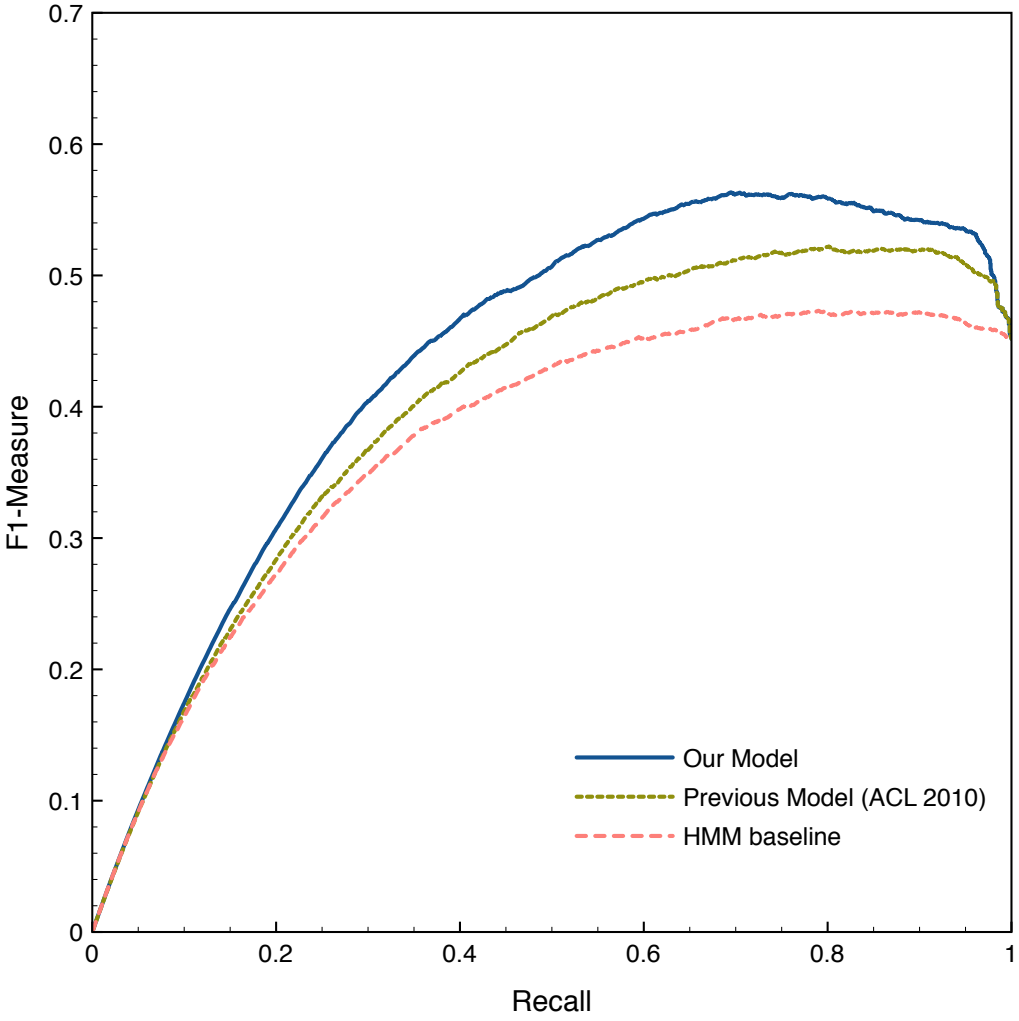

Figure 1-7: **Cognate identification F1 curves** for the model presented in this thesis (Our Model), the version of our model presented at ACL 2010 (using a uniform character model for non-cognate probabilities), and the HMM baseline.

1.9.4 Comparison to Model Variants

In this section we describe experiments comparing the performance of our full model to different variants. We focus on the performance of these variants on the task of cognate decipherment (section 1.9.2 above). Below we list each model variant and discuss its performance. Results are shown in the final rows of table 1.3.

Previous Version: This is the variant of our model originally published [109]. The key difference is that this older version models non-cognate word generation, $P(\mathbf{u}_i|c_i = 0)$, using a uniform distribution over Ugaritic letters. The model presented in this thesis, however, uses a unigram character distribution, taking into account letter frequency. As discussed in section 1.9.3, this leads to more accurate predictions of the cognate indicators c_i. In fact, as table 1.3 shows ("Prev Version"), this difference also leads to better performance on the task of cognate decipherment. While the previous version achieved 60.4% accuracy (WORDS → ACCURACY → TYPE), the current model yields 63% accuracy.

Only Substitutions: In this variant of our model, the only string-edit operation allowed is character substitution. All insertions and deletions are given zero probability. This allows a more precise comparison to the HMM baseline, which also only allows a one-to-one mapping between characters.

The results of this variant are reported in table 1.3 (**Only Sub**). Perhaps surprisingly, strictly requiring one-to-one character substitutions has little effect on performance. Word type prediction accuracy falls from 63% (when insertions are permitted) down to to 59.5%. Even smaller differences are seen across the other performance metrics. This small difference is partially due to the fact that substitutions are by far the dominant edit operation. Of equal importance, though, is the fact that our full model simply doesn't model insertions very effectively. As we discuss below, our model's biggest source of error involves missing a character insertion for a very common suffix.

Only Mapped: Recall that our model predicts alphabetic mappings via the structural sparsity indicator variables $\{\lambda_{(u,h)}\}$. These variables lead to higher prior probability being assigned to substitution (u, h) when the corresponding $\lambda_{(u,h)} = 1$. When predicting latent cognates, however, other substitutions do sometimes still occur, even when the corresponding $\lambda_{(u,h)} = 0$.

In this model variant, we assess the impact of allowing these substitutions. We run our sampling algorithm as usual, but at prediction time (section 1.7.5), we *only* allow substitution (u, h) to occur if $\lambda_{(u,h)} = 1$. As table 1.3 shows (**Only Mapped**), this restriction leads to lower performance on all our evaluation metrics. This result indicates that although our model is effective at predicting alphabetic mappings, it still benefits from allowing "unauthorized" substitutions to occur occasionally. In fact, an analysis shows that the most common of these substitutions is for the letter pair (š, š) which is actually a false-negative letter mapping prediction.

In the next three experiments, we explore the relative contribution of various components of our model. In each case, we remove one model component and report the resulting performance.

No Cognate Indicators: In this experiment, we test the performance of our model when *all* lost-language words are assumed to have cognates. In other words, the model is not allowed to "explain away" difficult word-forms by setting the corresponding cognate indicator to 0. This is achieved experimentally by simply setting the cognate prior $P(c_i = 1)$ to 1. As table 1.3 indicates (**No Cogs**), this leads to a serious degradation of performance across all measures. Thus, it seems crucial to allow the model to ignore word-forms which don't allow consistent mappings, even though the cognate identifications themselves are imperfect.

No Spike-and-slab: We test the performance of our model when the structural sparsity prior (section 1.6.2) is removed. Recall that the purpose of this prior was to ensure that predicted alphabetic mappings obey the following intuition: that each letter map to a very limited number of letters in the other language. In this experiment

we test the importance of this intuition. In particular, the value of K in equation 1.9 is set to 1 in this experiment. The result is that the structural sparsity indicator variables $\vec{\lambda}$ are essentially ignored. As table 1.3 shows (**No Spike**), performance in the absence of these variables degrades quite seriously. This finding confirms that incorporating the intuition of alphabetic-mapping sparsity is quite crucial for achieving high performance.

No Morphology: We test the performance of our model when no morphological segmentation is performed. Instead of segmenting Ugaritic words and matching the resulting morphemes to latent Hebrew morphemes, we instead match entire Ugaritic word-forms to entire Hebrew word-forms. This is achieved experimentally by setting the Hebrew prefix and suffix lexicons to the empty set, and setting the stem lexicon to the set of *entire* Hebrew words. As table 1.3 indicates (**No Morph**), this variant achieves only 36% accuracy on word types, far below that of all the other variants. This finding confirms that, just as for humans, morphological awareness is one of the key ingredients of success for computational decipherment.

Knowing Hebrew Cognates: Finally, we test the performance of our model when the Hebrew vocabulary is restricted to those morphemes which *actually* occur as cognates with Ugaritic words. The identity of these morphemes would not be known in a realistic decipherment scenario. Nevertheless, one could imagine a separate model which first predicts which Hebrew morphemes and words are likely to have Ugaritic cognates. This model could exploit the general tendencies of languages to preserve certain words and could also examine the cross-linguistic evidence given by cognates among known Semitic languages (Aramaic, Arabic, Akkadian, etc). The experiment presented here is intended to show our model's performance in the most ideal of situations, where the set of cognate morphemes is known *exactly*. The results given in table 1.3 (**Know Cogs**) indicate, perhaps unsurprisingly, that our model yields significantly improved performance in this scenario. Accuracy on word types reaches 71%.

1.9.5 Combining Model Variants

To further tease out the contributions of each component of our model, we consider various combinations of the above variants. First we consider a model very similar to the HMM baseline: All Ugaritic words are assumed to be cognates, no spike-and-slab prior is used, and only character substitutions are allowed (i.e. **Only Substitutions + No cognate Indicators + No Spike-and-slab**). The chief difference between this variant and the HMM baseline itself is that the latter attempts to match Ugaritic and Hebrew and the character trigram level, whereas our model matches the languages at the morphemic level. The results for this model are given in table 1.4 (**Combo 1**). The cognate decipherment accuracy at the word-type level drops to 35%, from 63% for the full model. However, we note that performance is still above that of the HMM baseline (29%). This result indicates that matching the lost and known languages at the morphemic level can indeed be more powerful than character-level matching alone.

Next, we drop the use of morphology as well (i.e. **Only Substitutions + No Cognate Indicators + No Spike-and-slab**). Now our model simply tries to find entire Hebrew words which match entire Ugaritic words, with a consistent character-level mapping. The results here are drastically worse. As table 1.4 indicates (**Combo 2**), the word-type decipherment accuracy for this model is only 21%, far below that of the baseline. This finding confirms once again the importance of morphological-level analysis for decipherment. Finally, **Combo 3** and **Combo 4** mirror the first two combinations, except that character insertions are now allowed. In both cases, allowing such insertions leads to about 2 percentage points of improved accuracy.

1.9.6 Related Language Discovery

One of the key assumptions we have made throughout this chapter is that the lost language is related to a known language, and that the known language has been identified. In the case of Ugaritic at least, human decipherers immediately surmised that Ugaritic was likely to be a Semitic language, due to the dating and geographical

	WORDS				MORPHEMES			
	ACCURACY		EDIT-DISTANCE		ACCURACY		EDIT-DISTANCE	
	TYPE	TOKEN	TYPE	TOKEN	TYPE	TOKEN	TYPE	TOKEN
Combo 1	.345	.491	.976	.744	.620	.737	.480	.333
Combo 2	.210	.270	1.61	1.20	.418	.595	1.01	.632
Combo 3	.361	.483	.984	.785	.635	.727	.468	.348
Combo 4	.230	.283	1.55	1.20	.414	.592	1.04	.657
Baseline	.288	.460	1.261	.878	n/a		n/a	
Our Model	.630	.697	.501	.400	.763	.838	.337	.239

Table 1.4: Combinations of the variants given in table 1.3. **Combo 1** only allows substitutions, assumes all words have cognates, and does not employ the structural sparsity prior (**Only Substitutions + No Cognate Indicators + No Spike-and-slab**). **Combo 2**, in addition, removes morphological analysis from the model (**Only Substitutions + No Cognate Indicators + No Spike-and-slab + No Morphology**). **Combo 3** allows all string edits, but assumes that all words have cognates, and does not use the structural sparsity prior (**No Cognate Indicators + No Spike-and-slab**). **Combo 4**, in addition, removes morphological analysis (**No Cognate Indicators + No Spike-and-slab + No Morphology**). The baseline and full model performance are repeated here for easy comparison.

location of the discovered clay tablets. However, in many other cases, the identity of a known, related language is far from certain. The Linear B script, for example, was discovered to encode an early form of Greek only after 50 years of decipherment efforts. The currently undeciphered Indus Valley symbol system remains even more mysterious. Although some scholars believe that it represents an early Dravidian language, others have argued that it is not likely to encode any spoken language.

For future statistical decipherment efforts we clearly need to move beyond the assumption that a known, related language has been clearly identified. This line of thought is largely beyond the scope of this thesis. However, we did run one final experiment to test whether our model can at least *distinguish* between a related and an unrelated known language.

In particular, we have applied our model to the decipherment of Ugaritic using *English* as the known language. In fact, of course, no known cognate pairs exist between Ugaritic (a Semitic language) and English (a Germanic language with significant Romance influence). The question we pose is the following: Can we automatically distinguish between our system output when using an actual related language

	Average entropy of...		
	$P_u(h)$	$P_h(u)$	$P_u(h)\&P_h(u)$
Ugaritic-Hebrew	0.43	0.56	0.48
Ugaritic-English	1.63	1.51	1.58

Table 1.5: **Cross-character entropy** when the known language is Hebrew versus English. Column one gives the average entropy of known-language letters for all Ugaritic letters. Column two gives the average entropy of Ugaritic letters for all known-language letters. Column three averages the entropies over letters from both alphabets.

(Hebrew) and our system output when using a non-related language (English).

To make our English lexicon as comparable as possible to the Hebrew lexicon, we base it on an English translation of the Hebrew Bible [94]. We use the Stanford Tagger [117], version 1.6[12] to part-of-speech tag the English corpus. We map the predicted parts-of-speech to five inflectional categories: adjective, adverb, noun, verb, and particle (non-inflectional). We use the Porter2 stemming algorithm[13] to induce an inventory of stems and suffixes for each of these categories.

We tested two criteria for automatically distinguishing between Hebrew and English output. In both cases, we consider predictions for *all* Ugaritic word-forms (not just those that actually have Hebrew cognates).

Cross-character Entropy

For both English and Hebrew the predicted indicator variables $\{\lambda_{(u,h)}\}$ will themselves be sparse (simply due to the structural sparsity prior). However, as noted in section 1.9.4, even when $\lambda_{(u,h)} = 0$, the corresponding substitution (u, h) will sometimes be used in cognate predictions. We hypothesize that if a large number of cognates truly exist between the lost and known languages, then character substitutions will display much greater regularity than would otherwise be possible. In other words, the actual substitutions used in cognate predictions will be far more sparse if the languages are truly related.

[12] `http://nlp.stanford.edu/software/tagger.shtml`
[13] `http://snowball.tartarus.org/algorithms/english/stemmer.html`

We measure this sparsity by computing the *cross-character entropy* of the predicted cognates. For each lost-language character u, we compute an empirical distribution over known-language characters h:

$$P_u(h) = \frac{N(u,h)}{\sum_{h'} N(u,h')},$$

where $N(u,h)$ denotes the number of times the substitution (u,h) appears in the final cognate decipherment predictions. Likewise, for each known-language character h we compute:

$$P_h(u) = \frac{N(u,h)}{\sum_{u'} N(u',h)}$$

We then compute the Shannon entropy (log base 2) for each distribution. As table 1.5 shows, the average cross-character entropy is indeed over three times higher when English is used as the known language, clearly distinguishing it from Hebrew, an actual related language.

Decipherment Count

Another method for distinguishing between related and unrelated languages uses the predicted alphabetic mapping $\{\lambda_{(u,h)}\}$. At prediction time we can force our model to *only* use character substitutions with $\lambda_{(u,h)} = 1$ (as in the "Only Mapped" experiment in section 1.9.4). One consequence of this constraint will be that we simply *won't be able* to find candidate cognates for some number of lost-language words. However, if the languages are truly related and our alphabetic mapping is mostly correct, then we should still be able to find candidate cognates for a good portion of the lost language vocabulary.

We thus hypothesize that, under this constraint, we will find a much larger number of impossible-to-map words when an unrelated language is used. Our experiments bear out this hypothesis in the cases of English and Hebrew. When using Hebrew, we are still able to find candidate mappings for over 67% of all Ugaritic words (4,797 / 7,386). However, when using English as the known language, we can only propose cognates for 7% of Ugaritic words (551 / 7,386). As before, this difference would

allow us to easily distinguish a related language (Hebrew) from an unrelated language (English) even before the decipherment predictions themselves have even been viewed by a human, let alone authenticated.

1.9.7 Error Analysis

In this section we analyze some of the errors made by our model in the task of cognate decipherment. We separately examine prefix errors, stem errors, and suffix errors. Table 1.6 gives the top errors in each category (see appendix A for a mapping from original graphemes to the characters in our transcription). We divide the errors into three major categories.

Prefix errors				*Stem errors*				*Suffix errors*			
Ug.	Predict	Hb.		Ug.	Predict	Hb.		Ug.	Predict	Hb.	
m-	m-		68	šlm	ślm	šlm	12	-m	-m	-ym	190
y-	y-		26	mit	mat	ma-h	8	-n	-n	-nw	43
a-	a-		19	ʕn	ʕwn	ʕyn	8			-h	29
		b-	18	šbʕ	śbʕ	šbʕ	8	-t	-t		28
b-	b-		14	a$\underline{t}$t	ašm	aš-h	7	-m	-m		28
		l-	13	ʕd	ʕd	bʕd	7	-y	-y		21
w-	w-	wy-	11	ib	ab	awyb	7	-n	-n		20
		k-	10	špt	śpt	śp-h	6	-n	-nw		16
		y-	8	dd	dd	dwd	6	-h	-h	-th	11
		t-	8	a$\underline{h}$t	aḥt	aḥ-wt	6	-k	-k	-tk	10
t-	m-	t-	7	šrš	śry	šrš	5	-k	-k		10
n-	n-		7	śś	nwn	śwś	5			-ym	9
wy-	wy-	w-	7	šm	śm	šm	5	-k	-k	-yk	7
h-	h-		6	qn	qn	qn-h	5			-t	7
k-	k-		6	šir	śar	šar	5	-w	-w		6
i-	a-		5	ḫrš	ḫrś	ḫrš	4	-m	-m	-wt	6

Table 1.6: Top cognate decipherment errors for prefixes, stems, and suffixes. For each morpheme category, the first column gives the Ugaritic morpheme (as segmented by our model), the second column gives the predicted Hebrew morpheme, the third column gives the correct Hebrew morpheme, and the fourth column gives the number of times this particular error was made.

Segmentation Errors: The most obvious category of errors consists of segmentation mistakes. All but one of the top prefix errors falls into this category. For example, the most common prefix error is predicting a prefix *m-* when in reality no prefix occurs at all (for either the Ugaritic word or its Hebrew counterpart). Sometimes a prefix occurs but is not predicted. For example, the fourth most common prefix error consists of predicting the empty prefix instead of the actual prefix *b-*. In addition, segmentation errors explain some decipherment errors of stems as well. For example, our model deciphers the Ugaritic word *a̱tt* (woman) as *ašm* (guilt), rather than correctly segmenting the feminine suffix and predicting the Hebrew word *aš-h*.

Substitution Errors: Another category of errors consists of incorrect letter substitutions. For example, the Ugaritic letter šmaps to both šand śin Hebrew. However, our model only predicts the mapping (š,ś). In many instances, replacing Ugaritic š with either š *or* ś results in valid Hebrew words (with entirely different meanings). Six of the top stem prediction errors involve ambiguities of this sort. For instance, the Ugaritic word *šbʕ*, meaning "seven" is incorrectly mapped to the Hebrew word *śbʕ*, meaning "satisfied" rather than the correct Hebrew word for seven, *šbʕ*. Even if our model were to correctly predict both character mappings ((š,š) and (š,ś)), it is not obvious if it would automatically pick out the correct Hebrew word.

Insertion Errors: Finally, another category of errors consists of missing Hebrew character *insertions*. As discussed in the previous section, our model does almost equally well if we disallow insertions altogether. The most common error, by far, is a suffix error involving a missing insertion. The Ugaritic masculine plural suffix, *-m*, corresponds to the Hebrew masculine plural suffix *-ym*. However, *-m* is *also* a suffix in Hebrew, indicating the third person masculine plural possessive. As before, it is not obvious how such errors can be corrected. In the next and final section of the chapter we discuss some possible directions for enriching the model to account for these shortcomings.

1.10 Conclusion and Future Work

In this chapter we proposed a method for the automatic decipherment of lost languages. The key strength of our model lies in its ability to incorporate a range of linguistic intuitions in a statistical framework.

First among these intuitions is that both character and lexical correspondences across related languages should be consistent. In addition, morphological analysis played a crucial role in our model, as the correspondences between highly frequent prefixes and suffixes can be particularly revealing (and easy to find). Finally, we developed a novel prior that encodes a crucial intuition: that the mapping between alphabets should be *structurally sparse*. Each character in the lost language should map to a very limited number of characters in the related language, and vice versa.

We applied our decipherment model to a corpus of Ugaritic, an ancient Semitic language discovered in 1928 and manually deciphered four years later, using knowledge of Hebrew, a related language. As input to our model, we use the corpus of Ugaritic texts along with a Hebrew lexicon extracted from the Hebrew Bible.

Our main experiments show that by modeling the interplay between morphology, character correspondences, and lexical correspondences, our model was able to predict a largely correct decipherment of Ugaritic. 28 of 30 letters were correctly mapped to their Hebrew counterparts, and over 63% of words with Hebrew cognates were correctly deciphered. Further experiments indicated that several factors were crucial to this success. In the absence of morphological modeling and the prior constraint on character fertility, prediction accuracy degrades significantly.

Finally, we examined the issue of related language identification. For many currently undeciphered lost languages, the key challenge lies in finding a related living language (if one exists). While our model is not designed to *find* related languages, our experiments show that it can at least distinguish between related and unrelated pairs.

We hope to address several issues in future work. One deficiency of our model is that it fails to take into account the known *frequency* of Hebrew words and mor-

phemes. The existence of a word or morpheme in the Hebrew lexicon is simply treated as a hard constraint. If the word exists, it may be matched to an Ugaritic counterpart, and otherwise it may not. What we see, in fact, is that the most common error of our model can be attributed to this feature: Our model incorrectly deciphers the Ugaritic masculine plural suffix (-*m*) as the Hebrew third person plural possessive suffix (-*m*), rather than the correct and much more common plural suffix (-*ym*). One way to achieve this frequency matching would be to simultaneously model the vocabularies of Ugaritic *and* Hebrew. Our current model treats the Hebrew lexicon as a wholly observed conditioning variable. Instead, we could assume that, just as Ugaritic words have latent Hebrew counterparts, so too do Hebrew words have latent Ugaritic counterparts. In this way, frequently occurring Hebrew morphemes will have to be accounted for by frequently occurring Ugaritic morphemes.

Another direction for future work is to add *contextual cues* to our model. Currently, our model operates purely at the vocabulary level. As we saw in the previous section, many of the errors our model makes are due to ambiguity. A single Ugaritic word could be legitimately translated into several Hebrew counterparts based solely on the historical character mappings. Scholars, of course, use the *literary context* of words to help uncover their meanings. While the Hebrew words for "seven" and "satisfied" both fit the characters of the Ugaritic word *šbʕ*, it is unlikely that both words would fit the context in which this word appears.

Appendix A

Ugaritic-Hebrew letter mappings

א	a
ב	b
ג	g
ד	d
ה	h
ו	w
ז	z
ח	ḥ
ט	ṭ
י	y
כ	k
ל	l
מ	m
נ	n
ס	s
ע	ʿ
פ	p
צ	ṣ
ק	q
ר	r
שׁ	š
שׂ	ś
ת	t

𐎀	a
𐎁	b
𐎂	g
𐎃	ḫ
𐎄	d
𐎅	h
𐎆	w
𐎇	z
𐎈	ḥ
𐎉	ṭ
𐎊	y
𐎋	k
𐎌	š
𐎍	l
𐎎	m
𐎏	ḏ
𐎐	n
𐎑	ẓ
𐎒	s
𐎓	ʿ
𐎔	p
𐎕	ṣ
𐎖	q
𐎗	r
𐎘	ṯ
𐎙	ġ
𐎚	ś
𐎛	t
𐎜	i
𐎝	u

Table A.1: Hebrew letters (left) and Ugaritic letters (right) with phonetic transcription.

Ugaritic \ Hebrew	a	b	g	d	h	w	z	ḥ	ṭ	y	k	l	m	n	s	ʕ	p	ṣ	q	r	š	ś	t
a	X																						
b		X																					
g			X																				
ḫ								X															
d				X			X																
h					X																		
w						X																	
z							X																
ḥ								X															
ṭ									X														
y										X													
k											X												
š																					X	X	
l												X											
m													X										
ḏ							X																
n														X									
ẓ																		X					
s															X								
ʕ																X							
p																	X						
ṣ																		X					
q																			X				
r																				X			
ṯ																					X		
ġ																X		X					
ś															X								
t																							X
i	X																						
u	X																						

Table A.2: **Gold Standard:** Mappings between Ugaritic and Hebrew letters, reflecting the historical relationship between the corresponding phonemes.

Ugaritic \ Hebrew	a	b	g	d	h	w	z	ḥ	ṭ	y	k	l	m	n	s	ʕ	p	ṣ	q	r	š	ś	t
a	X																						
b		X																					
g			X																				
ḫ								X															
d				X			*																
h					X																		
w						X																	
z		X*					*																
ḥ								X															
ṭ									X														
y										X													
k											X												
š																					*	X	
l												X											
m													X										
ḏ							X																
n														X									
ẓ																		X					
s															X								
ʕ																X							
p																	X						
ṣ																		X					
q																			X				
r																				X			
ṯ																					X		
ġ																X		*					
ś														X*	*								
t																							X
i	X																						
u	X																						

Table A.3: **Model Predictions:** Mappings between Ugaritic and Hebrew letters, as predicted by the matrix of indicator variables $\{\lambda_{(u,h)}\}$ in our model. Entries where predictions differ from the gold-standard mapping are indicated with an asterisk (*).

Hebrew

Ugaritic	a	b	g	d	h	w	z	ḥ	ṭ	y	k	l	m	n	s	ʕ	p	ṣ	q	r	š	ś	t
a	*																						
b		X																					
g			X																				
ḫ								*															
d				X			*																
h					*																		
w						*																	
z							X																
ḥ								X															
ṭ									X														
y						X*				*													
k											X												
š																					*	X	
l												X											X*
m										X*			X										
ḏ							*																
n														X									
ẓ																		*					
s															X								
ʕ	X*															X							
p																	X						
ṣ																		X					
q																			X				
r																				X	X*		
ṯ																					*		
ġ															*		*						
ś															*								
t					X*																		*
i	*																						
u	*																						

Table A.4: **Baseline Predictions 1:** Mappings between Ugaritic and Hebrew letters, as predicted by the HMM baseline (where (u, h) is predicted iff $u = \text{argmax}_{u'} P(u'|h)$). Entries where predictions differ from the gold-standard mapping are indicated with an asterisk (*).

Ugaritic \ Hebrew	a	b	g	d	h	w	z	ḥ	ṭ	y	k	l	m	n	s	ʕ	p	ṣ	q	r	š	ś	t
a	X																						
b		X																					
g			X																				
ḫ								X															
d				X			*																
h					*	X*																	
w						*				X*													
z							X																
ḥ								X															
ṭ									X														
y						X*				*													
k											X												
š																					X	*	
l												X											
m													X										
ḏ							X																
n														X									
ẓ			X*															*					
s															X								
ʕ																X							
p																	X						
ṣ																		X					
q																			X				
r																				X			
ṯ																					X		
ġ																*		*	X*				
ś															*					X*			
t					X*																		*
i	X																						
u	X																						

Table A.5: **Baseline Predictions 2:** Mappings between Ugaritic and Hebrew letters, as predicted by the HMM baseline (where (u, h) is predicted iff $h = \text{argmax}_{h'} P(h'|u) \propto P(u|h')P(h')$). Entries where predictions differ from the gold-standard mapping are indicated with an asterisk (*).

Hebrew

Ugaritic	a	b	g	d	h	w	z	ḥ	ṭ	y	k	l	m	n	s	ʕ	p	ṣ	q	r	š	ś	t
a	X																						
b		X																					
g			X																				
ḫ								X															
d				X			*																
h					*	X*																	
w						*				X*													
z							X																
ḥ								X															
ṭ									X														
y						X*				*													
k											X												
š																					X	X	
l												X											X*
m										X*			X										
ḏ							X																
n														X									
ẓ			X*															*					
s															X								
ʕ	X*															X							
p																	X						
ṣ																		X					
q																			X				
r																				X	X*		
ṯ																					X		
ġ															*			*	X*				
ś															*					X*			
t					X*																		*
i	X																						
u	X																						

Table A.6: **Baseline Predictions 3:** Mappings between Ugaritic and Hebrew letters, as predicted by the HMM baseline (where (u, h) is predicted iff **either** $u = \text{argmax}_{u'} P(u'|h)$ **or** $h = \text{argmax}_{h'} P(h'|u) \propto P(u|h')P(h'))$. Entries where predictions differ from the gold-standard mapping are indicated with an asterisk (*).

Bibliography

[1] Steven Abney and Steven Bird. The human language project: Building a universal corpus of the world's languages. In *Proceedings of the ACL*, pages 88–97, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[2] C. E. Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, 2:1152–1174, November 1974.

[3] James Baker. Trainable grammars for speech recognition. In *Proceedings of the Acoustical Society of America*, 1979.

[4] Michele Banko and Robert C. Moore. Part-of-speech tagging in context. In *Proceedings of the COLING*, pages 556–561, 2004.

[5] Emily M. Bender. Linguistically naïve != language independent: why NLP needs linguistic typology. In *Proceedings of the EACL 2009 Workshop on the Interaction between Linguistics and Computational Linguistics*, pages 26–32, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

[6] Taylor Berg-Kirkpatrick and Dan Klein. Phylogenetic grammar induction. In *Proceedings of the ACL*, pages 1288–1297, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[7] Shane Bergsma and Greg Kondrak. Multilingual cognate identification using integer linear programming. In *RANLP Workshop on Acquisition and Management of Multilingual Lexicons*, Borovets, Bulgaria, September 2007.

[8] Nicola Bertoldi, Madalina Barbaiani, Marcello Federico, and Roldano Cattoni. Phrase-based statistical machine translation with pivot languages. In *International Workshop on Spoken Language Translation Evaluation Campaign on Spoken Language Translation (IWSLT)*, pages 143–149, 2008.

[9] Indrajit Bhattacharya, Lise Getoor, and Yoshua Bengio. Unsupervised sense disambiguation using bilingual probabilistic models. In *Proceedings of ACL*, page 287, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

[10] Ann Bies, Martha Palmer, Justin Mott, and Colin Warner. *English Chinese translation treebank v 1.0*. LDC2007T02, 2007.

[11] Phil Blunsom, Trevor Cohn, and Miles Osborne. Bayesian synchronous grammar induction. In *Proceedings of NIPS*, 2008.

[12] Alexandre Bouchard, Percy Liang, Thomas Griffiths, and Dan Klein. A probabilistic approach to diachronic phonology. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 887–896, 2007.

[13] Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565, 1995.

[14] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. Word-sense disambiguation using statistical methods. In *Proceedings of the ACL*, pages 264–270, 1991.

[15] Sabine Buchholz and Erwin Marsi. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City, June 2006. Association for Computational Linguistics.

[16] David Burkett, John Blitzer, and Dan Klein. Joint parsing and alignment with weakly synchronized grammars. In *Proceedings of NAACL*, 2010.

[17] David Burkett and Dan Klein. Two languages are better than one (for syntactic parsing). In *Proceedings of EMNLP*, pages 877–886, 2008.

[18] David Burkett, Slav Petrov, John Blitzer, and Dan Klein. Learning better monolingual models with unannotated bilingual text. In *Proceedings of CoNLL*, 2010.

[19] Lyle Campbell. *Historical Linguistics: An Introduction*. Cambridge: MIT Press, 2004.

[20] Eugene Charniak and Glen Carroll. Two experiments on learning probabilistic dependency grammars from corpora. In *Proceedings of the AAAI Workshop on Statistically-Based NLP Techniques*, pages 1–13, 1992.

[21] Yu Chen, Andreas Eisele, and Martin Kay. Improving statistical machine translation efficiency by triangulation. In *Proceedings of LREC*, 2008.

[22] David Chiang. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the ACL*, pages 263–270, 2005.

[23] Shay B. Cohen and Noah A. Smith. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of the NAACL/HLT*, 2009.

[24] Trevor Cohn and Mirella Lapata. Machine translation by triangulation: Making effective use of multi-parallel corpora. In *Proceedings of ACL*, 2007.

[25] M. Collins. Head-driven statistical models for natural language parsing. *Computational linguistics*, 29(4):589–637, 2003.

[26] Bernard Comrie. *Language universals and linguistic typology: Syntax and morphology*. Oxford: Blackwell, 1989.

[27] CTIA-The Wireless Association. CTIA's wireless industry indices: 1985 - 2009.

[28] Jesus-Luis Cunchillos, Juan-Pablo Vita, and Jose-Ángel Zamora. Ugaritic data bank. CD-ROM, 2002.

[29] Ido Dagan, Alon Itai, and Ulrike Schwall. Two languages are more informative than one. In *Proceedings of the ACL*, pages 130–137, 1991.

[30] Gregoria del Olo Lete and Joaquín Sanmartín. *A Dictionary of the Ugaritic Language in the Alphabetic Tradition*. Number 67 in Handbook of Oriental Studies. Section 1 The Near and Middle East. Brill, 2004.

[31] Mona Diab and Philip Resnik. An unsupervised method for word sense tagging using parallel corpora. In *Proceedings of the ACL*, pages 255–262, 2002.

[32] Jason Eisner. Learning non-isomorphic tree mappings for machine translation. In *The Companion Volume to the Proceedings of the ACL*, pages 205–208, 2003.

[33] T. Erjavec. MULTEXT-East version 3: Multilingual morphosyntactic specifications, lexicons and corpora. In *Fourth International Conference on Language Resources and Evaluation, LREC*, volume 4, pages 1535–1538, 2004.

[34] M.D. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the american statistical association*, 90(230):577–588, 1995.

[35] Anna Feldman, Jirka Hana, and Chris Brew. A cross-language approach to rapid creation of new morpho-syntactically annotated resources. In *Proceedings of LREC*, pages 549–554, 2006.

[36] T.S. Ferguson. A Bayesian analysis of some nonparametric problems. *The annals of statistics*, 1:209–230, 1973.

[37] Pascale Fung and Kathleen McKeown. Finding terminology translations from non-parallel corpora. In *Proceedings of the Annual Workshop on Very Large Corpora*, pages 192–202, 1997.

[38] Kuzman Ganchev, Joao Graca, and Ben Taskar. Better alignments = better translations? In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics*. Association for Computational Linguistics, 2008.

[39] Andrew Gelman, John B. Carlin, Hal .S. Stern, and Donald .B. Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 2004.

[40] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[41] Dmitriy Genzel. Inducing a multilingual dictionary from a parallel multitext in related languages. In *Proceedings of HLT/EMNLP*, pages 875–882, 2005.

[42] W.R. Gilks, S. Richardson, and DJ Spiegelhalter. *Markov chain Monte Carlo in practice*. Chapman & Hall/CRC, 1996.

[43] Sharon Goldwater and Thomas L. Griffiths. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the ACL*, pages 744–751, 2007.

[44] Joshua Goodman. *Parsing inside-out*. PhD thesis, Harvard University, 1998.

[45] Alan Groves and Kirk Lowery, editors. *The Westminster Hebrew Bible Morphology Database*. Westminster Hebrew Institute, Philadelphia, PA, USA, 2006.

[46] Jacques B. M. Guy. An algorithm for identifying cognates in bilingual wordlists and its applicability to machine translation. *Journal of Quantitative Linguistics*, 1(1):35–42, 1994.

[47] Aria Haghighi and Dan Klein. Prototype-driven learning for sequence models. In *Proceedings of HLT-NAACL*, pages 320–327, 2006.

[48] Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. Learning bilingual lexicons from monolingual corpora. In *Proceedings of the ACL/HLT*, pages 771–779, 2008.

[49] Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[50] Kenneth Hale, Michael Krauss, Lucille J. Watahomigie, Akira Y. Yamamoto, Colette Craig, L. M. Jeanne, and N. C. England. Endangered languages. *Language*, 68(1):1–42, 1992.

[51] David Hall and Dan Klein. Finding cognates using phylogenies. In *Proceedings of ACL*, 2010.

[52] C. Han, N.R. Han, E.S. Ko, H. Yi, and M. Palmer. Penn Korean Treebank: Development and evaluation. In *Proc. Pacific Asian Conf. Language and Comp*, 2002.

[53] Jiri Hana, Anna Feldman, and Chris Brew. A resource-light approach to russian morphology: Tagging russian using czech resources. In *Proceedings of EMNLP*, pages 222–229, 2004.

[54] W. K. Hastings. Monte carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109, 1970.

[55] Robert Hetzron, editor. *The Semitic Languages*. Routledge, 1997.

[56] Geoffrey E. Hinton. Products of experts. In *Proceedings of the Ninth International Conference on Artificial Neural Networks*, volume 1, pages 1–6, 1999.

[57] John E. Hopcroft. An n log n algorithm for minimizing states in a finite automaton. Technical report, Stanford, CA, USA, 1971.

[58] R. Hwa, P. Resnik, A. Weinberg, C. Cabezas, and O. Kolak. Bootstrapping parsers via syntactic projection across parallel texts. *Journal of Natural Language Engineering*, 11(3):311–325, 2005.

[59] H. Ishwaran and J.S. Rao. Spike and slab variable selection: frequentist and Bayesian strategies. *The Annals of Statistics*, 33(2):730–773, 2005.

[60] T. Jiang, L. Wang, and K. Zhang. Alignment of trees – an alternative to tree edit. *Theoretical Computer Science*, 143(1):137–148, 1995.

[61] Mark Johnson. Why doesn't EM find good HMM POS-taggers? In *Proceedings of EMNLP/CoNLL*, pages 296–305, 2007.

[62] Mark Johnson, Thomas Griffiths, and Sharon Goldwater. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 139–146, Rochester, New York, April 2007. Association for Computational Linguistics.

[63] D. Klein. *The Unsupervised Learning of Natural Language Structure*. PhD thesis, Stanford University, 2005.

[64] Dan Klein and Christopher D. Manning. A generative constituent-context model for improved grammar induction. In *Proceedings of the ACL*, pages 128–135, 2002.

[65] K. Knight and K. Yamada. A computational approach to deciphering unknown scripts. In *ACL Workshop on Unsupervised Learning in Natural Language Processing*, 1999.

[66] Kevin Knight, Anish Nair, Nishit Rathod, and Kenji Yamada. Unsupervised analysis for decipherment problems. In *Proceedings of the COLING/ACL*, pages 499–506, 2006.

[67] Kevin Knight and Richard Sproat. Writing systems, transliteration and decipherment. NAACL Tutorial, 2009.

[68] Philipp Koehn and Kevin Knight. Learning a translation lexicon from monolingual corpora. In *Proceedings of the ACL-02 workshop on Unsupervised lexical acquisition*, pages 9–16, 2002.

[69] Grzegorz Kondrak. Identifying cognates by phonetic and semantic similarity. In *Proceeding of NAACL*, pages 1–8, 2001.

[70] Jonas Kuhn. Experiments in parallel-text based grammar induction. In *Proceedings of the ACL*, page 470, 2004.

[71] K. Lari and S.J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56, 1990.

[72] Vladimir Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966. Doklady Akademii Nauk SSSR, V163 No4 845-848 1965.

[73] Cong Li and Hang Li. Word translation disambiguation using bilingual bootstrapping. In *Proceedings of the ACL*, pages 343–351, 2002.

[74] Jun S. Liu. The collapsed Gibbs sampler in Bayesian computations with applications to a gene regulation problem. *Journal of the American Statistical Association*, 89(427):958–966, 1994.

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

[76] Gideon S. Mann and David Yarowsky. Multipath translation lexicon induction via bridge languages. In *Proceedings of NAACL*, pages 1–8, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[77] M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330, 1994.

[78] I. Dan Melamed. Multitext grammars and synchronous parsers. In *Proceedings of the NAACL/HLT*, pages 79–86, 2003.

[79] Bernard Merialdo. Tagging english text with a probabilistic model. *Computational Linguistics*, 20(2):155–171, 1994.

[80] N. Metropolis and S. Ulam. The Monte Carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949.

[81] Rada Mihalcea. *Current Issues in Linguistic Theory: Recent Advances in Natural Language Processing*, chapter Unsupervised Natural Language Disambiguation Using Non-Ambiguous Words. John Benjamins Publisher, 2004.

[82] Iain Murray, Zoubin Ghahramani, and David MacKay. MCMC for doubly-intractable distributions. In *UAI*, 2006.

[83] Jerome L. Myers and Arnold D. Well. *Research Design and Statistical Analysis*. Lawrence Erlbaum, 2nd edition, 2002.

[84] Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of EMNLP*, 2010.

[85] Tahira Naseem, Benjamin Snyder, Jacob Eisenstein, and Regina Barzilay. Multilingual part-of-speech tagging: two unsupervised approaches. *Journal of Artificial Intelligence Research*, 36(1):341–385, 2009.

[86] Hwee Tou Ng, Bin Wang, and Yee Seng Chan. Exploiting parallel texts for word sense disambiguation: an empirical study. In *Proceedings of the ACL*, pages 455–462, 2003.

[87] Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[88] Joakim Nivre and Ryan McDonald. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL*, pages 950–958, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[89] Franz Josef Och and Hermann Ney. Statistical multi-source translation. In *MT Summit 2001*, pages 253–258, 2001.

[90] Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

[91] Sebastian Padó and Mirella Lapata. Cross-linguistic projection of role-semantic information. In *Proceedings of the HLT*, pages 859–866, 2005.

[92] Sebastian Padó and Mirella Lapata. Optimal constituent alignment with edge covers for semantic projection. In *Proceedings of ACL*, pages 1161 – 1168, 2006.

[93] Gerald Penn and Travis Choma. Quantitative methods for classifying writing systems. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 117–120, New York City, USA, June 2006. Association for Computational Linguistics.

[94] Oxford University Press. *The Revised Standard Version Bible*. Oxford University Press, 2002.

[95] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[96] Reinhard Rapp. Automatic identification of word translations from unrelated english and german corpora. In *Proceedings of the ACL*, pages 519–526, 1999.

[97] Philip Resnik and David Yarowsky. A perspective on word sense disambiguation methods and their evaluation. In *Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?*, pages 79–86, 1997.

[98] Eric Sven Ristad and Peter N. Yianilos. Learning string-edit distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(5):522–532, 1998.

[99] C.P. Robert. *The Bayesian choice: from decision-theoretic foundations to computational implementation*. Springer Verlag, 2007.

[100] Andrew Robinson. *Lost Languages: The Enigma of the World's Undeciphered Scripts*. Thames & Hudson, 2009.

[101] William M. Schniedewind and Joel H. Hunt. *A Primer on Ugaritic: Language, Culture and Literature*. Cambridge University Press, 2007.

[102] Yoav Seginer. Fast unsupervised incremental parsing. In *Proceedings of the ACL*, pages 384–391, 2007.

[103] J. Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4(2):639–650, 1994.

[104] David A. Smith and Noah A. Smith. Bilingual parsing with factored estimation: Using English to parse Korean. In *Proceeding of EMNLP*, pages 49–56, 2004.

[105] Mark S. Smith, editor. *Untold Stories: The Bible and Ugaritic Studies in the Twentieth Century*. Hendrickson Publishers, 1955.

[106] Noah A. Smith and Jason Eisner. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the ACL*, pages 354–362, 2005.

[107] Benjamin Snyder and Regina Barzilay. Cross-lingual propagation for morphological analysis. In *Proceedings of the AAAI*, pages 848–854, 2008.

[108] Benjamin Snyder and Regina Barzilay. Unsupervised multilingual learning for morphological segmentation. In *Proceedings of the ACL/HLT*, pages 737–745, 2008.

[109] Benjamin Snyder, Regina Barzilay, and Kevin Knight. A statistical model for lost language decipherment. In *Proceedings of the ACL*, pages 1048–1057, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[110] Benjamin Snyder, Tahira Naseem, and Regina Barzilay. Unsupervised multilingual grammar induction. In *Proceedings of the ACL*, pages 73–81, 2009.

[111] Benjamin Snyder, Tahira Naseem, Jacob Eisenstein, and Regina Barzilay. Unsupervised multilingual learning for POS tagging. In *Proceedings of EMNLP*, pages 1041–1050, 2008.

[112] Benjamin Snyder, Tahira Naseem, Jacob Eisenstein, and Regina Barzilay. Adding more languages improves unsupervised multilingual part-of-speech tagging: a Bayesian non-parametric approach. In *Proceedings of the NAACL*, pages 83–91, 2009.

[113] Andreas Stolcke and Stephen M. Omohundro. Inducing probabilistic grammars by Bayesian model merging. In *Proceedings of ICGI*, pages 106–118, 1994.

[114] Kuo-Chung Tai. The tree-to-tree correction problem. *J. ACM*, 26(3):422–433, 1979.

[115] Inc. The Radicati Group. Email statistics report, 2009-2013.

[116] Kristina Toutanova and Mark Johnson. A Bayesian LDA-based model for semi-supervised part-of-speech tagging. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *NIPS*, pages 1521–1528, Cambridge, MA, 2008. MIT Press.

[117] Kristina Toutanova and Christopher D. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of*

EMNLP, pages 63–70, Morristown, NJ, USA, 2000. Association for Computational Linguistics.

[118] Masao Utiyama and Hitoshi Isahara. A comparison of pivot methods for phrase-based statistical machine translation. In *Proceedings of NAACL/HLT*, pages 484–491, 2006.

[119] Chong Wang and David Blei. Decoupling sparsity and smoothness in the discrete hierarchical dirichlet process. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *NIPS*, pages 1982–1989. MIT Press, 2009.

[120] Wilfred Watson and Nicolas Wyatt, editors. *Handbook of Ugaritic Studies*. Brill, 1999.

[121] Dekai Wu. Stochastic inversion transduction grammars, with application to segmentation, bracketing, and alignment of parallel corpora. In *IJCAI*, pages 1328–1337, 1995.

[122] Dekai Wu. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403, 1997.

[123] Dekai Wu and Hongsing Wong. Machine translation with a stochastic grammatical channel. In *Proceedings of the ACL/COLING*, pages 1408–1415, 1998.

[124] Chenhai Xi and Rebecca Hwa. A backoff model for bootstrapping resources for non-english languages. In *Proceedings of HLT/EMNLP*, pages 851 – 858, 2005.

[125] JL Xu. Multilingual search on the world wide web. In *Proceedings of the Hawaii International Conference on System Sciences HICSS*, volume 33, 2000.

[126] David Yarowsky and Grace Ngai. Inducing multilingual pos taggers and np bracketers via robust projection across aligned corpora. In *Proceedings of the NAACL*, pages 1–8, 2001.

[127] David Yarowsky, Grace Ngai, and Richard Wicentowski. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of HLT*, pages 161–168, 2000.

[128] David Yarowsky and Richard Wicentowski. Minimally supervised morphological analysis by multimodal alignment. In *ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 207–216, Morristown, NJ, USA, 2000. Association for Computational Linguistics.

[129] Hao Zhang and Daniel Gildea. Stochastic lexicalized inversion transduction grammar for alignment. In *Proceedings of the ACL*, pages 475–482, 2005.

[130] Kaizhong Zhang, Rick Statman, and Dennis Shasha. On the editing distance between unordered labeled trees. *Inf. Process. Lett.*, 42(3):133–139, 1992.