A Markov Model for the Acquisition of Morphological Structure

Leonid Kontorovich[1] Dana Ron[2] Yoram Singer[3]

June 3, 2003
CMU-CS-03-147

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We describe a new formalism for word morphology. Our model views word generation as a random walk on a trellis of units where each unit is a set of (short) strings. The model naturally incorporates segmentation of words into morphemes. We capture the statistics of unit generation using a probabilistic suffix tree (PST) which is a variant of variable length Markov models. We present an efficient algorithm that learns a PST over the units whose output is a compact stochastic representation of morphological structure. We demonstrate the applicability of our approach by using the model in an allomorphy decision problem.

[1]Carnegie Mellon University, lkontor@cs.cmu.edu
[2]Tel-Aviv University, danar@eng.tau.ac.il
[3]Hebrew University, singer@cs.huji.ac.il

Part of this research was done while L.K. was at the Hebrew University of Jerusalem. This research was supported by the F.I.R.S.T program of the Israeli Science Foundation. The work of L.K. at CMU was supported by NSF grant EIA-0205456.

Keywords: morphology, Markov, probabilistic suffix tree

1 Introduction

There has been much recent work on modeling and learning morphology. It seems reasonable to consider the Workshop on Morphological and Phonological Learning[1] as representative of the state of the art in the field. Reading the presented publications, one gets the sense that morphology does not present as daunting a problem to the scientific community as, say, speech recognition — if one is willing to invest enough human-labor, one will get a morphological analyzer for a given language.

Indeed, a voluminous amount of formal work on morphology has been devoted to finding a formalism using finite state automata for analyzing morphological structure. The most popular approach is based on the notion of two-level morphology which is modeled by a two-tier automaton. See for instance [9, 10] and the references therein. Much of the work on finite state automata for morphological structure has focused on studying the formal properties of the automata and has not involved any learning. While the two-level morphology is rich enough to capture the morphological structure of natural languages, the induction of an automaton for the morphological analysis of a given language often proceeds by compiling rules that were listed by humans.

The amount of human labor that is required to build finite state automata for morphology has spurred research on *automatic* and *unsupervised* methods for morphological structure induction using machine learning techniques. We now discuss a few of the approaches that have been taken. The summary below is by no means comprehensive and our goal is merely to underscore what is common to the various unsupervised methods for morphological analysis which motivated this work.

Morphological analysis is often viewed as the process of discovering the affixes of a language, ignoring to a large extent the nonconcatenative phenomena that are certainly non-negligible even in the Indo-European languages (see for instance [5, 6, 8, 16, 17]). Various unsupervised and lightly-supervised methods for automatic acquisition of morphological structure and analysis have been proposed [2, 8, 11, 17], often with impressive results.

The vast majority of the work translates the problem of learning morphological structure into a problem of clustering sequences or of splitting whole words into sub-sequences. For instance, Baroni et al. [2] employ a similarity measure to cluster and discover morphologically related words. In a widely cited work, Goldsmith [8] achieves unsupervised learning of the suffixes of a language using minimum description length (MDL) scoring. Neuvel and Fulop [11] propose a method for discovering "word-formation strategies" of a language without explicitly relying on the notion of a morpheme. Their method is based on a notion of similarity between morphologically related strings. Snover et al. [17] propose a probabilistic language generation model based on an inverse-square distribution for the number of affixes and the stem and affix length of words.

There have been also more direct attempts to learn a *deterministic* finite state automaton for phonological and morphological rules [7, 18]. These approaches are based on the assumption that a deterministic finite state machine can be inferred from almost *unstructured* data and rely on automaton induction techniques that are guaranteed to work *in the limit* of an *infinite* number of examples (see for instance the analysis of OSTIA in [12]). However, passively learning automata from a *finite sample* is known to be hard in an information theoretic sense [1]. Furthermore, Ristad [14] provides an elaborate formal morphophonemic model and proves that learning his model is also NP-hard.

The various difficulties outlined above have motivated the approach we describe in this paper. Our method of addressing these problems is based on a new probabilistic model for morphological structure. Informally, we view word generation as biased random walks on trellises, where the trellises capture morphological structure. Such trellises are translated and generated by a probabilistic suffix tree (PST) [15, 19]

[1] 11 July 2002, `www.ldc.upenn.edu/maxwell/MorphologyLearning.html`

which we describe in the sequel. The learning problem then reduces, to a large extent, to the problem of inferring a hidden *alphabet* over which the PST operates. To render the learning task feasible, we provide the algorithm with more explicit supervision in the form of alignments of words belonging to the same root. By limiting the structure of the internal alphabet of the PST and providing the learning algorithm with partially parsed and aligned words we are able to devise an algorithm that seems to sidestep the intractability issues of the more general morphological structure.

The paper is organized as follows. In Sec. 2 we present our formalism for bounded morphological structures and describe our generative model for capturing the statistics of word generation from morphological structures. In Sec. 3 we describe our learning algorithm for the generative model. In Sec. 4 we describe and analyze the results of applying our model and learning algorithm to the task of structure acquisition in Latin.

2 Formal setting

In this section we present the definitions for our morphological structure. We then proceed to formally describe the generative model we employ.

2.1 Morphological structure

Units and morphemes: A *unit* U over an alphabet Σ is a finite set of $u_i \in \Sigma^*$, $U = \{u_1, u_2, \ldots, u_n\}$, satisfying the following restrictions:

- If U consists of a single element, that element must be a literal in Σ or ε:

$$U = \{u\} \Rightarrow u \in \Sigma \cup \{\varepsilon\}.$$

- No element of U may be longer than M: $\forall u \in U$, $|u| \leq M$ for some fixed constant $M > 0$ (for all our computations, we set $M = 1$).

We denote by $\mathcal{U}$ the (finite) set of all possible units over Σ and call it the *unit space*.

A unit U generalizes a literal $a \in \Sigma$ in two ways. First, a unit may contain strings longer than a single literal long. Second, a unit may contain several literals. Most generally, a unit contains strings of varying lengths.

The intuition is that the elements of a unit are somehow “related” or “interchangeable”. For example, after seeing the English words illegal, impractical, ineligible, irreplaceable, (and knowing that they all contain the “same” prefix), it is natural to define the unit $U = \{\mathsf{l}, \mathsf{m}, \mathsf{n}, \mathsf{r}\}$ for the different ways in which the prefix may end. Incidentally, in this case U corresponds to a known phonetic class — the so-called *liquids*. That is not a coincidence: the intended interpretation for a unit is “a collection of strings related phonetically or grammatically”. Another example: in Russian, we have iskat′ (“to search”) and ixu (“I search”)[2]; the numerous examples of this kind necessitate the unit $U = \{\mathsf{sk}, \mathsf{x}\}$.

We define a *morpheme* μ to be a sequence of units: $\mu = \langle U_1 U_2 \ldots U_T \rangle$, where $U_t \in \mathcal{U}$. We use triangular brackets to emphasize that μ is an ordered sequence whose elements may repeat, and omit the commas between the terms to evoke a concatenative association. This is a formalization of the less formal definition of “morpheme” from linguistics – “the smallest unit of meaning”, “the smallest part into which

[2]Throughout this work, when providing examples we use a quasi-morphophonemic representation that seems best to illustrate a point; it should not be taken literally and may contain minor inconsistencies.

a word can be meaningfully broken down", etc. If we have the units $U_1 = \{\mathsf{i}\}$ and $U_2 = \{\mathsf{l}, \mathsf{m}, \mathsf{n}, \mathsf{r}\}$, then the morpheme $\mu = \langle U_1 U_2 \rangle = \langle \{\mathsf{i}\}\{\mathsf{l}, \mathsf{m}, \mathsf{n}, \mathsf{r}\} \rangle$ captures the common prefix in the words illegal, impractical, ineligible, irreplaceable. Another example: after seeing theater and theatrical, and assuming we have the units $\{\mathsf{t}\}$, $\{\mathsf{h}\}$, $\{\mathsf{e}\}$, $\{\mathsf{a}\}$, $\{\mathsf{r}\}$, $\{\mathsf{e}, \varepsilon\}$, we build the morpheme $\mu = \langle \{\mathsf{t}\}\{\mathsf{h}\}\{\mathsf{e}\}\{\mathsf{a}\}\{\mathsf{t}\}\{\mathsf{e}, \varepsilon\}\{\mathsf{r}\} \rangle$ to capture the *theater/theatr-* root. We can also understand the definition of a morpheme as follows. A morpheme has

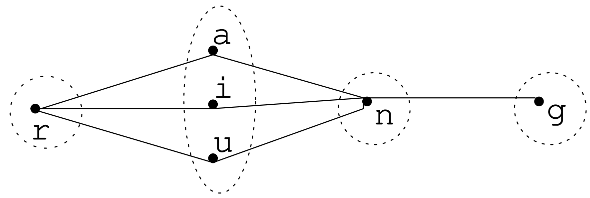

Figure 1: The morpheme $\langle \{\mathsf{r}\}\{\mathsf{i}, \mathsf{a}, \mathsf{u}\}\{\mathsf{n}\}\{\mathsf{g}\} \rangle$ visualized as a trellis with 3 paths, corresponding to ring, rang, and rung.

the structure of a trellis where units are the basic building blocks that make up these morpheme-trellises in a given language. A particular word is derived from a given morpheme by traversing the morpheme's trellis and outputting a string form Σ^* at each traversed node. An illustration of this alternative view is given in Fig. 1.

Span and allomorph: We next define the *span* of a sequence of units $\omega = U_1 U_2 \dots U_T$ by $\mathrm{span}(\omega) = \{u \in \Sigma^* \mid u = u_1 u_2 \dots u_T,\ u_t \in U_t\}$. (The unit sequence ω may be a single morpheme or a string of morphemes.) An *allomorph* a of a morpheme μ is an element of its span: $a \in \mathrm{span}(\mu)$. We shall use "the span of μ" and "the allomorphs of μ" interchangeably. Identifying morphemes with trellises as above, the span of a morpheme is seen to be the set of all string-paths through the morpheme-trellis. For example,

$$\mathrm{span}(\langle \{\mathsf{i}\}\{\mathsf{l}, \mathsf{m}, \mathsf{n}, \mathsf{r}\} \rangle) = \{\mathsf{il}, \mathsf{im}, \mathsf{in}, \mathsf{ir}\}$$

and

$$\mathrm{span}(\langle \{\mathsf{t}\}\{\mathsf{h}\}\{\mathsf{e}\}\{\mathsf{a}\}\{\mathsf{t}\}\{\mathsf{e}, \varepsilon\}\{\mathsf{r}\} \rangle) = \{\mathsf{theater}, \mathsf{theatr}\}.$$

Observe that our formalism embodies an abstraction from the specifics of a particular encoding, whether it be phonetic, morphophonemic, orthographic, or other. For example, the Latin word for "night" is nox and when inflected, its stem becomes noct-. In the orthographic transcription, we would have the morpheme $\langle \{\mathsf{n}\}\{\mathsf{o}\}\{\mathsf{x}, \mathsf{ct}\} \rangle$. If the transcription were phonetic, however, we might get a different morpheme: $\langle \{\mathsf{n}\}\{\mathsf{o}\}\{\mathsf{k}\}\{\mathsf{t}, \varepsilon\} \rangle$ (the latter being "truer" to the underlying morphophonemic process).

A morpheme μ defines a language over Σ^*, namely, its span. It is easy to see that, for a fixed set of units $\mathcal{W}$, the class of languages defined by the morphemes is strictly weaker than bounded expressions. On one hand, any finite collection of strings is captured by some bounded expression. On the other hand, the strings a and b cannot be allomorphs of the same morpheme unless the units $U_1 = \{\mathsf{a}, \varepsilon\}$ and $U_2 = \{\mathsf{b}, \varepsilon\}$, or $U_3 = \{\mathsf{a}, \mathsf{b}\}$ are in $\mathcal{W}$. The formalism, as presented so far, is clearly far from being complete. For example, it does not allow precise definitions of morphological synthesis or analysis, and does not handle inflections. We do have a more comprehensive morphological formalism, which we intend to include in a long version of this work. While the extended formalism is capable of capturing more complex morphological structures, its complexity makes the learning problem much more difficult. In order to facilitate efficient learning of morphological structure, we therefore confine ourselves to the rather restrictive formal structure that was given above.

2.2 Generative model: Variable-Context Probabilistic Suffix Trees

Human language morphology is inherently ambiguous. For example, there is more than one way to synthesize the past tense of `dream` (`dreamed` and `dreamt`) and more than one way to analyze the word `resent` ('to dislike' or 'sent again'). Thus the structural formalism ought to have an accompanying probabilistic generative model $\Pr(\mu, u) \in [0, 1]$, which assigns a probability to the event of generating the sequence of units ω and its word-realization u. Given a generative model we can use it for both *synthesis* and *analysis*. The synthesis map Φ takes strings of units (which may consist of several morphemes) to their "most probable" word-realizations:

$$\Phi(\mu) = \operatorname*{argmax}_{u:u\in\mathrm{span}(\mu)} \Pr(u|\mu, \texttt{context}). \tag{1}$$

The analysis map Ψ is a sort of inverse of synthesis – it takes words in a language to their "most probable" parent unit sequences:

$$\Psi(u) = \operatorname*{argmax}_{\mu:u\in\mathrm{span}(\mu)} \Pr(u|\mu, \texttt{context}). \tag{2}$$

(Without committing to a specific definition of *context*, what we have in mind is sufficient neighboring units to remove the types of ambiguities mentioned above.[3])

In what follows we suggest a particular generative model which is a certain type of Markov model. It is important to emphasize that the descriptive formalism we introduced in the previous subsection does *not* force a unique generative model. What we describe here is simply one of many possibilities. We start with a high level description of the model and then turn to a more formal definition.

A High Level Description. Our model generates sequences of pairs, [unit,substring] where the substring in the pair always belongs to the unit in the pair. The idea is that the projection of such a sequence on the first element of each pair (the unit), results in a morpheme (or a concatenation of morphemes), while the projection on the second element results in an allomorph (or concatenation of allomorphs). Note that the allomorph always belongs to the span of the corresponding morpheme. In some cases we may view the units in the sequence as being "hidden", similarly to the way states in a Hidden Markov Model are hidden. In other cases, such as during the learning phase, the units are partially hidden (see Subsection 3.2 for further elaboration). However, it will be convenient at this stage to view both parts of the pairs in the sequence as observed.

These sequences are generated iteratively, one pair at a time. The probability distribution on the next pair that is output, depends on (some of) the pairs previously generated (the "history" or "context"). Here we actually assume that the next-pair distribution depends only on the units previously generated, and not on the substrings. This assumption is elaborated subsequently. An important aspect of our model is that the *length* of the context (the number of previously output units) that determines the distribution on the next pair may vary.

The intuitive explanation behind the variable contexts that are used for generating the next pair is that different contexts have different degrees of "predictive power". For example, suppose that we have a unit for each letter in the English alphabet, and that the unit is a singleton set that contains only the given letter. In this special case, there is a one-to-one mapping between units and the single symbol they contain. We hence view the output sequence as a string over the English alphabet, which we denote by Σ. Suppose that given a sequence of English text, $u_1 \ldots u_T \in \Sigma^T$, we are trying to predict the next letter, u_{T+1}. If $u_T = \mathtt{q}$ then this context is pretty much sufficient — we can say with a high confidence that $u_{T+1} = \mathtt{u}$. Other letters are

[3] The `dreamed`/`dreamt` ambiguity is unlikely to be resolved by any local neighbors; the choice is more a function of the overall discourse style. For the purpose of the formalism, we include this with the "context".

less informative and a longer context is required. This suggest to have a variable context length. In addition, since our model is constructed based on a limited-size data set, longer contexts appear less often and are hence are less reliable statistically. We will take this into account in the learning phase. We now proceed to formalize the above description.

A Formal Definition of the Model. Let $\mathcal{W} \subset \mathcal{U}$ denote a fixed set of units, and let $\Sigma_{\mathsf{tree}} = \mathcal{W} \times \Sigma^*$. We refer to Σ_{tree} as our *symbol alphabet*, where each symbol consists of a pair $[U, u]$, $U \in \mathcal{W}$, $u \in U \subset \Sigma^*$. Our model generates sequences $[U_1, u_1], [U_2, u_2] \ldots \in \Sigma^*_{\mathsf{tree}}$ in the following probabilistic manner. Suppose that the sequence generated up to time T is $[U_1, u_1] \ldots [U_T, u_T]$. Then the distribution over the next symbol (pair) $[U_{T+1}, u_{T+1}]$ is determined by the last d units $U_{T-d+1} \ldots U_T$, where $d \leq D$, and D is the *maximum order* of the model.

Specifically, our model is a slight variant of a *Probabilistic Suffix Tree* (*PST*), of maximum order D. We later explain in what sense our model differs, and for the sake of brevity, we refer to our variant as a PST.

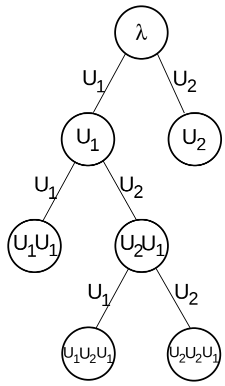

Figure 2: Variable context suffix tree.

A PST is represented by a tree of depth at most D, and degree $|\mathcal{W}|$. Every edge of the tree is labeled by a unit in $\mathcal{W}$, where the $|\mathcal{W}|$ edges between a node and its children are labeled by the different units in $\mathcal{W}$. The nodes of the tree are labeled by sequences over $\mathcal{W}$: the root of the tree is labeled by the empty sequence λ, and the label of every other node is the sequence of units on the path from the node to the root. Every node represents a *context*, where the length of the context is the depth of the node in the tree. For an illustration, see Fig. 2.

With each node in the tree we associate a *next-symbol* probability distribution. Namely,

- $\gamma : \mathcal{W}^{\leq D} \times \Sigma_{\mathsf{tree}} \to [0, 1]$ is the (conditional) next-symbol probability distribution, where for each node $q = U_1 \ldots U_d \in \mathcal{W}^d$ in the tree, we have

$$\sum_{U \in \mathcal{W}, u \in U} \gamma(q, [U, u]) = 1 \ .$$

A sequence is generated as follows. Let $U_1 \ldots U_T$ be the sequence of units generated up to time T. That is, $U_1 \ldots U_T$ is the projection of the output sequence $[U_1, u_1] \ldots [U_T, u_T]$ onto the first coordinate of each pair. Then, the next symbol $[U, u] \in \Sigma_{\mathsf{tree}}$ is selected probabilistically according to $\gamma(q, \cdot)$, where $q = U_{T-d+1} \ldots U_T$ is the deepest node in the tree that corresponds to a suffix of $U_1 \ldots U_T$.

The difference between a PST as define above and PST's as defined in other works (cf. [15, 19]), is that we assume that the next symbol probability distribution at time $T + 1$ depends only on the sequence of

$d \leq D$ *units* $U_{T-d+1} \dots U_T$ that were previously generated, and not on the $d \leq D$ *symbols* $u_{T-d+1} \dots u_T$ generated, as is more standard. In other words, we take a projection of the sequence of symbols onto the first part of each symbol, and condition the next symbol only on the projection.

For some problems concerning human language morphology, such as for the purpose of synthesizing words, this assumption is too strong. Indeed, the actual substrings output, $u_{T-D+1} \dots u_T$, and not only the sequence of units, may matter. A full model would condition the event $[U_{T+1}, u_{T+1}]$ on the entire $d \leq D$-history $[U_{T-d+1}, u_{T-d+1}] \dots [U_T, u_T]$. Since in this investigation we only deal with the analysis mapping (see Subsection 4.1), this approximation seems appropriate.

3 Learning Algorithm

3.1 Overview

Since our learning process has several stages, we first present a high-level overview of the learning algorithm, and then give more details on each stage. Our initial data consists of subsets of words, where each subset contains several allomorphs that belong to the span of a single morpheme (or possibly each word is a concatenation of several allomorphs from the spans of a fixed list of morphemes). We start by aligning the words in each subset and obtaining in this way sequences of [unit,substring] pairs that are suited for training. We then compute empirical next-symbol probability distributions $\hat{\gamma}(\cdot, \cdot)$ based on raw counts. In the next stage we perform a certain *smoothing* process on these empirical probabilities. Using these smoothed counts we decide what units should be merged. Given the new units, we reparse the subsets of input strings and thus obtain modified sequences of [unit,substring] pairs. We continue smoothing, merging and reparsing, until no improvement is obtained by merging.

1. input aligned, unit-segmented strings
2. compute $\hat{\gamma}(\cdot, \cdot)$ using raw counts
3. smooth the raw probabilities
4. merge units
5. (reparse the input strings)
6. if nothing left to merge, stop; else goto 1

Figure 3: Learning algorithm overview.

3.2 Training data

An ideal training set would consist of segmented and aligned morpheme/allomorph sequences. For example, the Latin verb `ago` means 'I do/act/drive' and the prefix `com-` has the general meaning of 'with' and can act as an intensifier. When these are concatenated, several things occur at the morphological level[4] (Fig. 4). In the perfect tense, there is a different stem change: we have `ago` $\rightarrow$ `egi` ('I acted'), where a short `a` becomes a

[4]There is also a semantic change: `cogo` means 'I compel'. However, we currently make no mention of semantics.

$$\begin{array}{llll} \mathsf{com} + \mathsf{ago} & \rightarrow & *\mathsf{comago} & \text{(direct concatenation)} \\ & \rightarrow & *\mathsf{coago} & \text{(intervocalic m disappears)} \\ & \rightarrow & \mathsf{cogo} & \text{(a is absorbed into o)} \end{array}$$

Figure 4: (*) indicates an unattested or hypothetical intermediate form.

long e. Likewise, in the perfect passive participle we get actus ('the thing having been acted').[5] Information of this sort would be presented roughly as shown in Fig. 5 in a standard Latin dictionary. We transcribe all of this morphological information as in Fig. 6. From there, this information is naturally expressed in terms of sequences of [unit,substring] pairs (Fig. 7).

ago, egi, actus	to do/act/drive
cogo, coegi, coactus	to compel
exigo, exegi, exactus	to exact

Figure 5: Typical Latin dictionary entries.

```
1. /-aei/cg/-t/              [ag-]
2. /-aei/cg/-t/              [eg-]
3. /-aei/cg/-t/              [act]
4. /c/o/-lmnr/#/-aei/cg/-t/  [co-#-g-]
5. /c/o/-lmnr/#/-aei/cg/-t/  [co-#eg-]
6. /c/o/-lmnr/#/-aei/cg/-t/  [co-#act]
7. /e/-cfx/#/-aei/cg/-t/     [ex#ig-]
8. /e/-cfx/#/-aei/cg/-t/     [ex#eg-]
9. /e/-cfx/#/-aei/cg/-t/     [ex#ag-]
```

Figure 6: Examples of aligned morpheme (left) and allomorph (right) transcription. Note that the data structure used here forces $M = 1$.

Note that we have introduced a morpheme break unit $U_{\#} = \{\varepsilon\}$, whose function is to indicate morpheme boundaries for the PST. The unit $U_{\#}$ serves two purposes. During synthesis, the morpheme boundaries it demarcates are essential for capturing morphotactic phenomena. During analysis, inferring the presence of $U_{\#}$ amounts to segmenting words into morphemes – something our model yields quite naturally with no added effort.

The training setup we've described above is perhaps overly idealistic and we might consider various relaxations of it. The former might be called the "intact morphemes with placeholders" scenario, in the sense that, for example, the morpheme $\langle\{\mathsf{c}\}\{\mathsf{o}\}\{\mathsf{l},\mathsf{m},\mathsf{n},\mathsf{r},\varepsilon\}\rangle$ remains intact even though we don't observe any literal of the $\{\mathsf{l},\mathsf{m},\mathsf{n},\mathsf{r},\varepsilon\}$ unit in the cogo/coegi/coactus entry in Fig. 6. Since we *have* observed the other allomorphs of this morpheme in the other entries, we have left a placeholder in the alignment for this unit.

One might imagine less propitious learning scenarios – for example, we might be forced to rely only on local alignment information, as in Fig. 8.

As a further relaxation, one might consider removing the morpheme break unit $U_{\#}$ from the training da-

[5] Note that we do not concern ourselves with the inflectional endings -o, -i, -us, etc.; only stem changes such as ag $\rightarrow$ eg $\rightarrow$ act interest us for now. In future work, we will address the issue of inflectional affixes.

1. $[\{\mathsf{a},\mathsf{e},\mathsf{i},\varepsilon\},\mathsf{a}],[\{\mathsf{c},\mathsf{g}\},\mathsf{g}],[\{\mathsf{t},\varepsilon\},\varepsilon]$
2. $[\{\mathsf{a},\mathsf{e},\mathsf{i},\varepsilon\},\mathsf{e}],[\{\mathsf{c},\mathsf{g}\},\mathsf{g}],[\{\mathsf{t},\varepsilon\},\varepsilon]$
3. $[\{\mathsf{a},\mathsf{e},\mathsf{i},\varepsilon\},\mathsf{a}],[\{\mathsf{c},\mathsf{g}\},\mathsf{c}],[\{\mathsf{t},\varepsilon\},\mathsf{t}]$
4. $[\{\mathsf{c}\},\mathsf{c}],[\{\mathsf{o}\},\mathsf{o}],[\{\mathsf{l},\mathsf{m},\mathsf{n},\mathsf{r},\varepsilon\},\varepsilon],[U_{\#},\varepsilon],[\{\mathsf{a},\mathsf{e},\mathsf{i},\varepsilon\},\varepsilon],[\{\mathsf{c},\mathsf{g}\},\mathsf{g}],[\{\mathsf{t},\varepsilon\},\varepsilon]$
5. $[\{\mathsf{c}\},\mathsf{c}],[\{\mathsf{o}\},\mathsf{o}],[\{\mathsf{l},\mathsf{m},\mathsf{n},\mathsf{r},\varepsilon\},\varepsilon],[U_{\#},\varepsilon],[\{\mathsf{a},\mathsf{e},\mathsf{i},\varepsilon\},\mathsf{e}],[\{\mathsf{c},\mathsf{g}\},\mathsf{g}],[\{\mathsf{t},\varepsilon\},\varepsilon]$
6. $[\{\mathsf{c}\},\mathsf{c}],[\{\mathsf{o}\},\mathsf{o}],[\{\mathsf{l},\mathsf{m},\mathsf{n},\mathsf{r},\varepsilon\},\varepsilon],[U_{\#},\varepsilon],[\{\mathsf{a},\mathsf{e},\mathsf{i},\varepsilon\},\mathsf{a}],[\{\mathsf{c},\mathsf{g}\},\mathsf{c}],[\{\mathsf{t},\varepsilon\},\mathsf{t}]$

Figure 7: Obtaining $[U, u]$ sequences from alignments.

1.1	`[ag-]`	`/ae/cg/-t/`
1.2	`[eg-]`	`/ae/cg/-t/`
1.3	`[act]`	`/ae/cg/-t/`
2.1	`[co#-g-]`	`/c/o/#/-ae/cg/-t/`
2.2	`[co#eg-]`	`/c/o/#/-ae/cg/-t/`
2.3	`[co#act]`	`/c/o/#/-ae/cg/-t/`
3.1	`[ex#ig-]`	`/e/x/#/aei/cg/-t/`
3.1	`[ex#eg-]`	`/e/x/#/aei/cg/-t/`
3.1	`[ex#act]`	`/e/x/#/aei/cg/-t/`

Figure 8: The right column is superfluous: we form the units out of the columns of each local alignment.

ta, and perhaps even all of the alignment information, feeding the learner just the groups $\{\mathsf{cog},\mathsf{coeg},\mathsf{coact}\}$, etc. Although we have ideas for learning in various settings, in this paper we shall stick to the simplest one, with a minor twist of difficulty described below.[6]

Using the notation presented in the previous section, the training data that is given to the learning algorithm consists of sequences of pairs $[\tilde{U}_1, u_1] \ldots [\tilde{U}_T, u_T]$, where $u_i \in \tilde{U}_i$, and each $\tilde{U}_i$ is a subset of a real unit $U \in \mathcal{W}$. We refer to $\tilde{U}_i$ as a *pre-unit*, and assume that there are no two identical pre-units that are subsets of two different units. The set of pre-units is denoted $\tilde{\mathcal{W}}$. Hence, we do not assume that we get the actual sequences of symbols generated by the target model, but rather we have partial information concerning these sequences. Note that if the units were completely "hidden" (that is, if the pre-unit $\tilde{U}_i$ to which u_i belongs is simply $\{u_i\}$), then the problem of learning PST's, even of order 1, would be as hard as learning an HMM.

3.3 Empirical probabilities

Given the input data as described above, we define the empirical probabilities induced by the data, in a natural manner. For sake of notational simplicity, we use U_i and not $\tilde{U}_i$ with the understanding that U_i is a pre-unit and not necessarily a unit. Similarly, we use $\mathcal{W}$ and not $\tilde{\mathcal{W}}$. Let S be a data sample consisting of sequences of the form $[U_1, u_1] \ldots [U_T, u_T] \in \Sigma^*_{\mathsf{tree}}$.

- For any subsequence $q = U_1 \ldots U_d \in \mathcal{W}^d$, $d \le D$, let N_q^S be the number of occurrences of q in S. That is, the number of occurrences of the form $[*,*] \ldots [U_1,*] \ldots [U_d,*] \ldots [*,*]$ in S.

  For any $q = U_1 \ldots U_d$ and pair $[U, u] \in \Sigma_{\mathsf{tree}}$, let $N^S_{q,[U,u]}$ be the number of occurrences of q that are followed by $[U, u]$.

[6]An excellent question is *how* one obtains such data. We created our training set starting with essentially a Latin dictionary as in Fig. 5 and running standard alignment algorithms with various Latin-specific heuristics. In the future we plan to address the issue of automating this process further.

- Given the above let

$$\hat{\gamma}(q,[U,u]) \stackrel{\text{def}}{=} \frac{N^S_{q,[U,u]}}{N^S_q}$$

denote the next-symbol empirical probability distribution.

3.4 Smoothing empirical probabilities

Given the empirical probabilities $\hat{\gamma}$ as defined above, we can easily construct a PST whose nodes correspond to subsequences of (pre-)units that occur in the data, and whose next next symbol probability distributions are $\hat{\gamma}$. However, even if the pre-units we have were actually the correct units, such a PST would most probably be a bad hypothesis of the target PST due to the unreliability of some of the empirical counts. In order to address this difficulty we would like to assign *weights* to the nodes of this "empirical PST", and to output as our hypothesis a weighted version of this tree. Intuitively speaking, the weight of each node measures how "reliable" this node is as compared to its parent (and other ancestors in the tree).

More precisely, an empirical PST with empirical probabilities $\hat{\gamma}$ and weights $\alpha(q)$ assigned to its nodes $q = U_1 \dots U_d$, $d \geq 1$, induces the following next symbol probability distributions. For each node $q = U_1 \dots U_d$, and pair $[U,u]$,

$$\tilde{\gamma}(q,[U,u]) = \begin{cases} \hat{\gamma}(q,[U,u]) & : q = \lambda \\ \alpha(q) \cdot \hat{\gamma}(q,[U,u]) + (1-\alpha(q)) \cdot \tilde{\gamma}(q_{[2:d]},[U,u]) & : q \neq \lambda \end{cases} \tag{3}$$

where $q_{[2:d]} = U_2 \dots U_d$ is the parent of q in the tree, and in general, for $1 \leq i \leq j$, we let $q_{[i:j]} = U_i \dots U_j$. If we want to unravel the recursion then we get that for $d > 0$,

$$\begin{aligned} &\tilde{\gamma}(q,[U,u]) \\ &\quad = \alpha(q) \cdot \hat{\gamma}(q,[U,u]) \\ &\qquad + \sum_{l=1}^{d} \left(\prod_{i=1}^{l} \left(1 - \alpha(q_{[i:d]})\right) \right) \cdot \alpha(q_{[l+1:d]},[U,u]) \cdot \hat{\gamma}(q_{[l+1:d]},[U,u]) \end{aligned} \tag{4}$$

where $\alpha(\lambda) = 1$.

3.5 The smoothing procedure

We now discuss a procedure for obtaining the smoothing weights (that is, the α's). This procedure is performed for a fixed set of (pre-)units. We later discuss how pre-units are merged so as to obtain units. Our smoothing method builds upon previous work and combines the online Bayesian averaging procedure for context tree weighting [13, 19] with EM-based techniques for smoothing word probabilities in n-gram models (cf. [3]).

It is assumed that the corpus has been split into two parts: a training sample ($\sim 90\%$) denoted S, and a validation sample ($\sim 10\%$) denoted V. We use the validation sample to set the weights $\alpha(q)$, using the counts $N^V_{q,[U,u]}$.

The α's are computed iteratively, using a straightforward to derive EM update rule. We initialize $\alpha^{(0)}(q) = 0.5$ for every q. For every $t \geq 0$ we let $\tilde{\gamma}^{(t)}(\lambda,[U,u])$ be as in Equation (3), with $\alpha^{(t)}(q)$ in place of $\alpha(q)$.

Given this we iterate:

$$
\begin{aligned}
Y^{(t)}_{q,[U,u]} &= \frac{\alpha^{(t)}(q) \cdot \hat{\gamma}(q,[U,u])}{\tilde{\gamma}^{(t)}(q,[U,u])} \\
\alpha^{(t+1)}(q) &= \frac{1}{N^V_q} \sum_{[U,u]} N^V_{q,[U,u]} Y^{(t)}_{q,[U,u]}
\end{aligned}
\tag{5}
$$

The update rule in (5) is derived as follows. Fix a sequence $q = U_1 \ldots U_D$, put $q_d = q_{[D-d+1:D]}$ for $d = 0, 1, \ldots, D$. To compute $\alpha(q_d)$, assume $\alpha(q_{d-1})$ and $\tilde{\gamma}(q_{d-1})$ were already computed. Observe that $\alpha(q_d)$ is the probability that, given the history q_d, the probability of the next symbol is conditioned on all of q_d, while $(1 - \alpha(q_d))$ is the probability that this event is conditioned on q_{d-1}. Define the indicator variable $I_{[U,u]}$ to be 1 if the generation of $[U,u]$ was conditioned on the full q_d history and 0 if it was conditioned on q_{d-1}. Initialize $\alpha^{(0)}(q_d) = 0$ and assume t iterations have been executed. The (additive) contribution of q_d to the log-likelihood of the held-out data is

$$
\mathcal{L}_{q_d} = \sum_{U \in \mathcal{W}, u \in U} N^V_{q,[U,u]} \left[I_{[U,u]} \log(\alpha^{(t)} \tilde{\gamma}^{(t)}) + (1 - I_{[U,u]}) \log[(1 - \alpha^{(t)}) \tilde{\gamma}^{(t)}] \right] \tag{6}
$$

where $\tilde{\gamma}$ is computed as in Equ. (3) and several q_d and $[U,u]$ arguments/subscripts have been omitted for readability. The E-step is to set the "hidden" variable $I_{[U,u]}$ equal to its current expected value; we define

$$
\begin{aligned}
Y^{(t)}_{q_d,[U,u]} &= \mathbb{E}[I_{[U,u]} | \alpha^{(t)}(q_d)] && (7) \\
&= \frac{\alpha^{(t)}(q_d) \cdot \hat{\gamma}(q_d,[U,u])}{\tilde{\gamma}^{(t)}_d(q_d,[U,u])}. && (8)
\end{aligned}
$$

The M-step is to maximize $\alpha^{(t+1)}(q_d)$ at the next iteration; this is achieved by replacing $\alpha^{(t)}$ with $\alpha^{(t+1)}$ in (6), differentiating $\mathcal{L}_{q_d}$ w.r.t. $\alpha^{(t+1)}(q_d)$, and setting the derivative to 0.

3.6 Merging units

As noted previously, the problem with the units observed in the training data is that they may be fragmentary and over-specific. For example, we may have the following morphemes in the training data:

$$
\mu_{\text{drag}} = \langle \{\mathsf{t}\}\{\mathsf{r}\}\{\mathsf{a}\}\{\mathsf{c},\mathsf{h},\mathsf{x}\}\{\mathsf{t},\varepsilon\} \rangle
$$

and

$$
\mu_{\text{join}} = \langle \{\mathsf{i}\}\{\mathsf{u}\}\{\mathsf{n}\}\{\mathsf{c},\mathsf{g},\mathsf{x}\}\{\mathsf{t},\varepsilon\} \rangle
$$

raising the question of whether we have observed two distinct units $U_1 = \{\mathsf{c},\mathsf{h},\mathsf{x}\}$ and $U_2 = \{\mathsf{c},\mathsf{g},\mathsf{x}\}$ or two instances of a single unit $U_3 = U_1 \cup U_2 = \{\mathsf{c},\mathsf{g},\mathsf{h},\mathsf{x}\}$. We cast this question in terms of hypothesis testing and use the likelihood-ratio test, described in the sequel.

Define a *merge operator*, whose argument is a set of units, $\{U_1, U_2, \ldots, U_n\}$ and whose action is to replace each occurrence of U_i, $1 \le i \le n$ in the training data with $U' = \bigcup_i^n U_i$. Of course, after such a merger, the likelihood of the observed data will also change.

For any subsequence $q = U_1 \ldots U_D$ and pair $[U,u]$, define

$$
\ell^S_{q,[U,u]} = N^S_{q,[U,u]} \log\left(\tilde{\gamma}(q,[U,u])\right), \tag{9}
$$

prefix	root	form	result	comment
-	iac	1 pers. sng. pres. ind. act.	iacio	"I throw"
in	iac	1 pers. sng. pres. ind. act.	inicio	a reduces to ε with prefix
in	iac	1 pers. sng. perf. ind. act.	inieci	a lengthens to to e in perfect
-	ag	1 pers. sng. pres. ind. act.	ago	"I act"
com	ag	1 pers. sng. pres. ind. act.	cogo	nasal m disappears, vowel a reduces to ε
ex	ag	1 pers. sng. pres. ind. act.	exigo	a reduces to i
-	ag	1 pers. sng. perf. ind. act.	egi	stem vowel lengthens in perfect
com	ag	1 pers. sng. perf. ind. act.	coegi	stem vowel lengthens in perfect
ex	ag	1 pers. sng. perf. ind. act.	exegi	stem vowel lengthens in perfect
-	ag	perf. pass. part. masc. sng.	actus	g alternates with ct in perfect part.
-	tang	1 pers. sng. pres. ind. act.	tango	"I touch"
ad	tang	1 pers. sng. pres. ind. act.	attingo	d assimilates to t, a reduces to i
-	tang	1 pers. sng. perf. ind. act.	tetigi	reduplication and nasal infix reduction in perf.
-	tang	perf. pass. part. masc. sng.	tactus	g alternates with ct in perfect part.

Figure 9: Examples of morphological structures in Latin.

where $N^S_{q,[U,u]}$ and $\tilde{\gamma}(q,[U,u])$ are as in Subsections 3.3 and 3.4, and $0\log 0 \equiv 0$. Using simple algebraic manipulations, we can write now the log-likelihood of the data in terms of $\ell^S_{q,[U,u]}$ as follows,

$$\mathcal{L}^S = \sum_{q\in\mathcal{W}^D,U,u} \ell^S_{q,[U,u]}. \tag{10}$$

Let $\mathcal{L}^S_0$ be the log-likelihood of the data before the merger of a set of units and $\mathcal{L}^S_1$ its log-likelihood after the merger. We are thus faced with a classical hypothesis testing problem: whether the two units should be kept intact (the null hypothesis) or be merged. The likelihood-ratio is the optimal procedure for a *single* hypothesis testing problem [4]. The log-likelihood ratio in our case amounts to computing $\Delta\mathcal{L}^S = \mathcal{L}^S_1 - \mathcal{L}^S_0$ for all legal mergers. An ideal algorithm for discovering the best set of units would find all the sets of units whose merger would maximize $\Delta\mathcal{L}^S = \mathcal{L}^S_1 - \mathcal{L}^S_0$, and merge them. Unfortunately, such an algorithm would be rather inefficient. Therefore, instead of considering all subsets of units for merging, we examine them two at a time. The actual computation is made more efficient by the fact that when considering the units U_1 and U_2 for merger, we need not compute the whole sum in Equation (10) — we only sum over those terms in which either of U_1 or U_2 appears. Thus we greedily merge pairs of units until there is no longer a pair whose merger increases the likelihood of the data. The appendix shows the calculations involved in such a merger.

Remark Note the following potential problem with our merging procedure. Suppose there is one real unit, $U_1 = \{\mathsf{a},\mathsf{u},\mathsf{i}\}$ and another real unit, $U_2 = \{\mathsf{a},\mathsf{u},\mathsf{e}\}$. If we were to observe the pre-unit $\tilde{U} = \{\mathsf{a},\mathsf{u}\}$, sometimes as a realization of U_1 and sometimes of U_2, then our procedure would merge $\tilde{U}$ with at most one of (U_1, U_2), but not with both. Let us call this the problem of *merging without replacement*. To address this issue we make the simplifying assumption that every pre-unit we observe belongs to exactly one real unit. We do not know if there is an efficient way to merge with replacement (the naive method would require an exponential calculation).

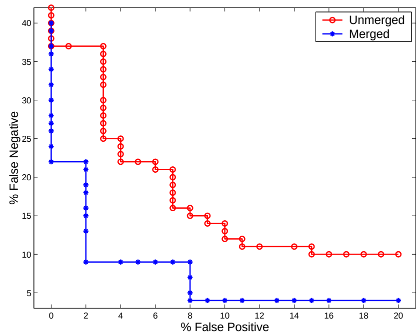

Figure 10: False-negative vs. false-positive curves for the learned (merged) and initial (unmerged) PST.

4 Evaluation

Our training data consisted of 1684 groups of segmented and aligned Latin verbs (one such group is displayed in lines 4-6 of Fig. 6), resulting in a total of 5145 sequences of $[U, u]$ pairs. A separate test set consisted of 100 groups totalling 297 $[U, u]$ sequences. We chose Latin as a test case because of its rich yet regular morphology. Latin being an Indo-European language, its morphology is concatenative, yet displays numerous nonlinear phenomena, such as prefix- and tense-dependent vowel reduction, assimilation, reduplication, and various consonant alternations. We provide a few illustrations of these phenomena in Fig. 3.6.

We note that the training data started out with 39 units (candidates), and after iterated greedy pairwise merging we ended up with 23 units. Browsing the results that the merging procedure produces reveals a reasonable set of units. Of course, the real test of this routine is how well the resulting morphological engine can synthesize and analyze the forms. In this extended abstract we restrict our attention to analysis, and provide experimental evidence to our model's capabilities.

4.1 Word Analysis

Analysis is the process of taking a set of strings $\{s_i\}_{i=1}^K \subset \Sigma^*$ and producing a sequence of units $\omega = U_1 \dots U_T$ such that $\{s_i\} \subset \mathrm{span}(\omega)$. We may view analysis as an application of the Ψ operator defined in (2) to each s_i, with the constraint that $\Psi(s_i) = \Psi(s_j)$ for $1 \le i, j \le K$. Thus analysis subsumes both alignment and segmentation: we align the strings s_i by forcing them to be generated from the same sequence of units and segment them by identifying indices t such that $U_t = U_\#$ (the morpheme break) at various locations in ω.

The generative model we described above, which assigns probabilities to single sequences of $[U, u]$, is readily adapted to this task. For $q \in \mathcal{W}^D$, $U \in \mathcal{W}$ and $u \in U$, define

$$\delta(q, U) = \sum_{u \in U} \tilde{\gamma}(q, [U, u])$$

and

$$\beta(q, U, u) = \frac{\tilde{\gamma}(q, [U, u])}{\delta(q, U)}.$$

Define A to be an *alignment* of $\{s_i\}_{i=1}^K$ if A is a $K \times T$ matrix over $\Sigma \cup \{\square\}$ ($\square$ denotes here the empty symbol), and the ith row of A, with the $\square$'s removed, equals s_i. For $\omega = U_1 \dots U_T$, we say that ω is *compatible* with alignment A if the tth column of A, $A_{[*,t]}$, is a subset of U_t (notation is slightly abused).

We define the following generation process for alignments: A sequence of T units is generated Markovially with $\delta(q, U_t)$ as the probability of the next unit being U_t conditioned on the context $q \in \mathcal{W}^D$, and, given the choice of U_t, generating K independent emissions $u_i \in U_t$, according to $\beta(\cdot, U_t, u_i)$. Let $\sharp_t u$ denote the number of times character u occurs in the tth column of A. Then the joint probability of (ω, A) is computed to be:

$$\Pr(\omega, A) = \prod_t \delta(\omega_{[t-D+1:t]}, U_{t+1}) \cdot K! \prod_{u \in A_{[*,t]}} \frac{(\beta(\omega_{[t-D+1:t]}, U_{t+1}, u))^{\sharp_t u}}{(\sharp_t u)!},$$

where the factorials come from the multinomial coefficient. Observe that for $K = 1$ this reduces to $\prod_t \tilde{\gamma}(\cdot, [U_t, u_t])$. Our algorithm for computing the most probable (ω, A) for a given $\{s_i\}$ is based on the Viterbi algorithm with a modification to handle ε-emissions

4.2 Classification Based on Word Analysis

In order to evaluate the analysis capabilities of the model we have learned, we consider the problem of determining, given two words s_1 and s_2 in a language, whether or not they are allomorphs of the same morpheme. Our model provides a natural way of doing this: parse the words $\{s_1, s_2\}$ together to obtain $\Pr(\omega^{(12)}, A^{(12)})$ and then each one separately to obtain $\Pr(\omega^{(1)}, A^{(1)})$ and $\Pr(\omega^{(2)}, A^{(2)})$. If $\log(\Pr(\omega^{(12)}, A^{(12)})) - \log(\Pr(\omega^{(1)}, A^{(1)}) \cdot \Pr(\omega^{(2)}, A^{(2)})) > \tau$, for some fixed threshold τ, output yes; otherwise output no. The choice of the threshold τ naturally affects the results in terms of the number of false positives versus false negatives. The intuition behind parsing strings "together" vs. "separately" is as follows. Let us view the negative log probability of a transition or emission as a part of the cost for a parse. When we parse strings together, we only charge their unit (δ) transitions once; when we parse strings separately, we charge the sum of the $-\log \delta$'s for each string. Thus the δ contribution to the parse cost will always be lower (more favorable) when we parse strings together. However, if two strings with wildly different characters are forced to be generated by the same unit sequence, the emissions β are going to have a low probability, resulting in a high parse cost. This captures the idea that allomorphs are realizations of some underlying sequence of units.

The setup is as follows: we took the 100 test word groups, (297 actual words), and computed the edit distance between each pair (with a cost of 1 for swap or delete). Then we took the 50 string pairs that came from the *same* morpheme and had the *highest* edit distance, as well as the 50 string pairs that came from *different* morphemes and had the *lowest* edit distance, and these 100 string pairs are the evaluation set. Note that this deliberate choice of strings to compare makes it as hard as possible on our algorithm. In Fig. 10 we demonstrate the performance of our algorithm by ploting the percentage of false-negatives vs. the false-positives. The curves in the figure illustrate how the performance of the merged units-PST is consistently better than the initial PST that was merely smoothed. This behavior indicates that the Markovian trellis model combined with the learning algorithm captured the stochasic properties of morphological structure. The overall misclassification error of the learned PST is 11% while for the unmerged PST the error is 22%. While this is a good indication that learning of the trellis statistics indeed took place, the error of the learned PST is fairly high. We believe that this can largely be attributed to the relatively small number of examples. In future research we plan to enrich our training set by using the learned PST as a boostrap mechanism to segment and align more words.

5 Conclusion

Our fundamental assumption here is that it is much easier to obtain training data of the type that we use than to manually list the morphological rules for a language. Indeed, the former task was accomplished by one person with the aid of a computer in the timespan of about a week, while the latter has been known to be far more labor-intensive. Another aspect that seems to set our work apart from many others is that while heuristics often enter NLP algorithms, in our case their presence is limited to constructing the training data – the learning algorithm itself is a faithful implementation of the simple computational model described.

Our work raises further questions of how to extract groups of morphologically related words from an unlabeled corpus, and how to turn these into aligned strings to feed into our algorithm – we plan to address these in future work.

Appendix

Here we give the calculations for the merging operation described in Sec. 3.6.

Let $N_{q,[U,u]}$ be the as in Sec. 3.3, with the superscript omitted for clarity. Here, all contexts q have the maximum length D. Suppose we are merging units U_{i_1} and U_{i_2}. We define an equivalence relation, $\equiv_{\mathrm{m}}$, on contexts by setting

$$q^{(1)} \equiv_{\mathrm{m}} q^{(2)} \quad \text{iff whenever } q^{(1)}[j] \neq q^{(2)}[j], \text{ we have } \{q^{(1)}[j], q^{(2)}[j]\} = \{U_{i_1}, U_{i_2}\} \tag{11}$$

(in other words, two contexts are $\equiv_{\mathrm{m}}$ equivalent if whenever they differ by a unit, that unit is either U_{i_1} or U_{i_2}).

Define a new unit $U' = U_{i_1} \cup U_{i_2}$ and let $\mathcal{W}' = \mathcal{W} \cup \{U'\}$ [7]. Observe that the merging operation φ (which replaces each occurrence of U_{i_1} and U_{i_2} in the training data with U') collapses each equivalence class of unmerged-unit contexts into a single merged-unit context:

$$\varphi : \mathcal{W}^D \to (\mathcal{W}')^D.$$

Thus if q is a merged-unit context, $\varphi^{-1}(q)$ is the equivalence class of the unmerged-unit contexts that get mapped to q.

Let $Q(U_i) \subset (\mathcal{W}')^D$ be the set of all length-D strings of units that contain at least one occurrence of U_i; let $Q(U_{i_1}, U_{i_2}) = Q(U_{i_1}) \cup Q(U_{i_2})$. The unmerged log-likelihood was computed in Eq. (10):

$$\mathcal{L}^{\mathrm{u}} = \sum_{q \in \mathcal{W}^D, U, u} N_{q,[U,u]} \log(\tilde{\gamma}(q, [U,u])). \tag{12}$$

We can now define the merged counts $N'_{q,[U,u]}$:

$$N'_{q,[U,u]} = \begin{cases} 0 & , (q \in Q(U_{i_1}, U_{i_2})) \vee (U \in \{U_{i_1}, U_{i_2}\}) \\ N_{q,[U,u]} & , (q \notin Q(U')) \wedge (U \neq U') \\ N_{q,[U_{i_1},u]} + N_{q,[U_{i_2},u]} & , (q \notin Q(U')) \wedge (U = U') \\ \sum_{q' \in \varphi^{-1}(q)} N_{q',[U,u]} & , (q \in Q(U')) \wedge (U \neq U') \\ \sum_{q' \in \varphi^{-1}(q)} (N_{q',[U_{i_1},u]} + N_{q',[U_{i_2},u]}) & , (q \in Q(U')) \wedge (U = U') \end{cases} . \tag{13}$$

[7] Logically it might be more appropriate to define $\mathcal{W}' = \mathcal{W} \setminus \{U_{i_1}, U_{i_2}\} \cup \{U'\}$, that is, to remove U_{i_1} and U_{i_2} from the set of units. We only keep them for notational convenience.

Then, we get an expression for the merged log-likelihood:

$$\mathcal{L}^{\mathrm{m}} = \sum_{q \in (\mathcal{W}')^D, U, u} N'_{q,[U,u]} \log(\tilde{\gamma}'(q, [U, u])), \tag{14}$$

where $\tilde{\gamma}'(q, [U, u])$ is the context-smoothed next-symbol probability, computed from the renormalized merged counts. The likelihood change, $\Delta\mathcal{L} = \mathcal{L}^{\mathrm{m}} - \mathcal{L}^{\mathrm{u}}$ is simply the difference of the two sums in (14) and (12). Note that the actual summation need only be carried out over those terms which have changed after the merger, namely

$$\begin{aligned}
\Delta\mathcal{L} \quad = \quad & - \sum_{q,U,u:(q \in Q(U_{i_1}, U_{i_2})) \vee (U \in \{U_{i_1}, U_{i_2}\})} N_{q,[U,u]} \log(\tilde{\gamma}(q, [U, u])) \\
& + \sum_{q,U,u:(q \notin Q(U')) \wedge (U = U')} N'_{q,[U,u]} \log(\tilde{\gamma}'(q, [U, u])) \\
& + \sum_{q,U,u:(q \in Q(U')) \wedge (U \neq U')} N'_{q,[U,u]} \log(\tilde{\gamma}'(q, [U, u])) \\
& + \sum_{q,U,u:(q \in Q(U')) \wedge (U \in U')} N'_{q,[U,u]} \log(\tilde{\gamma}'(q, [U, u])).
\end{aligned} \tag{15}$$

References

[1] D. Angluin. A note on the number of queries needed to identify regular languages. *Information and Control*, 51:76–87, 1981.

[2] M. Baroni, J. Matiasek, and H. Trost. Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. *ACL Workshop on Morphological and Phonological Learning*, 2002.

[3] P. Brown, V. Della Pietra, P. de Souza, J. Lai, and R. Mercer. Class based n-gram models for natural language. *Computational Linguistics*, 18(4):467–479, 1992.

[4] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., 1991.

[5] M. Creutz and K. Lagus. Unsupervised discovery of morphemes. *ACL Wksp. on Morp. and Phon. Learning*, 2002.

[6] E. Gaussier. Unsupervised learning of derivational morphology from inflectional lexicons. *ACL Workshop Unsupervised Learning in Natural Language Processing*, 1999.

[7] D. Gildea and D. Jurafsky. Automatic induction of finite state transducers for simple phonological rules. In *Proc. of the 33rd Annual Conf. of the ACL*, pages 9–15, 1995.

[8] J. Goldsmith. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198, 2001.

[9] L. Karttunen. Finite-state constraints. In *Intl. Conf. on Current Issues in Computational Linguistics*, 1991.

[10] K. Koskenniemi. Two-level model for morphological analysis. In *Proc. of the Eighth Intl. Joint Conference on Artificial Intelligence*, pages 683–685, 1983.

[11] S. Neuvel and S. Fulop. Unsupervised learning of morphology without morphemes. *ACL Workshop Unsupervised Learning in Natural Language Processing*, 2002.

[12] J. Oncina, P. Garcia, and E. Vidal. Learning subsequential transducers for pattern recognition and interpretation tasks. *IEEE Trans. on PAMI*, 15(5):448–458, 1993.

[13] F.C. Pereira and Y. Singer. An efficient extension to mixture techniques for prediction and decision trees. *Machine Learning*, 36(3):183–199, 1999.

[14] E. Ristad. Complexity of morpheme acquisition. In *Language Computations*. AMS, 1994.

[15] D. Ron, Y. Singer, and N. Tishby. The power of amnesia: learning probabilistic automata with variable memory length. *Machine Learning*, 25(2):117–150, 1996.

[16] P. Schone and D. Jurafsky. Knowledge-free induction of morphology using latent semantic analysis. In *Proc. of the 4th Conf. on Computational Nat. Lang. Learning*, 2000.

[17] M. Snover, G. Jarosz, and M. Brent. Unuspervised learning of morphology using a novel directed search algorithm: Taking the first step. *ACL Workshop on Morphological and Phonological Learning*, 2002.

[18] P. Theron and I. Cloete. Automatic acquisition of two-level morphological rules. In *Fifth Conference on Applied Natural Language Processing*, pages 103–110, 1997.

[19] F.M.J. Willems, Y.M. Shtarkov, and T.J. Tjalkens. The context tree weighting method: basic properties. *IEEE Trans. on Information Theory*, 41(3):653–664, 1995.