Learning Morphology of Natural Language as a Finite-State Grammar

Javad Nouri and Roman Yangarber(✉)

Department of Computer Science, University of Helsinki, Helsinki, Finland
{javad.nouri,roman.yangarber}@cs.helsinki.fi

Abstract. We present algorithms that learn to segment words in morphologically rich languages, in an unsupervised fashion. Morphology of many languages can be modeled by finite state machines (FSMs). We start with a baseline MDL-based learning algorithm. We then formulate well-motivated and general linguistic principles about morphology, and incorporate them into the algorithm as heuristics, to constrain the search space. We evaluate the algorithm on two highly-inflecting languages. Evaluation of segmentation shows gains in performance compared to the state of the art. We conclude with a discussion about how the learned model relates to a morphological FSM, which is the ultimate goal.

Keywords: Unsupervised morphology induction · Minimum description length principle · MDL · Finite-state automata

1 Introduction

We present work on unsupervised *segmentation* for languages with complex morphology. Our ultimate research question is to explore to what extent morphological structure can be induced without supervision, from a large body of unannotated text (a corpus). This has implications for the question whether the morphological system is somehow *"inherently encoded"* in language, as represented by the corpus.

The paper is organized as follows: we state the morphology induction problem (Sect. 2), review prior work (Sect. 3), present our models (Sects. 4 and 5), present the evaluation and experiments (Sect. 6), and finally discuss future work.

2 Morphological Description

We focus on highly-inflecting languages. In the experiments, we use Finnish and Turkish, which belong to different language families (Uralic and Turkic) and exhibit different morphological phenomena.

Finnish is considered to be agglutinative, although it has some fusional phenomena—morpho-phonological alternation. It has complex derivation and inflection, and productive compounding. Derivation and inflection are achieved via suffixation; prefixes do not exist. Multiple stems in the compound may be inflected, e.g., *kunnossapidon* = "of keeping (smth.) in usable condition":

kunno	+	*ssa*	#	*pido*	+	*n*
condition	+	Iness.	#	keeping	+	Gen.

where # is a compound boundary, + is a morpheme boundary, *Iness.* marks *inessive* case (presence in a location, or being in a state), and *Gen.* marks the *genitive*. The morph *kunno* is a "weak" allomorph of the stem *kunto*; the weakening is conditioned by the closed syllable environment, i.e., the following double consonant *-ss-*.

Turkish is similar to Finnish: agglutination, no prefixation, minimal compounding.

The ultimate goal in the future is to model aspects of morphology, including classification of morphemes into morphological categories, and capturing *allomorphy*—morpho-phonological alternation.[1] However, at present, as in most prior work, we address the problem of *segmentation* only, to try to establish a solid baseline. Once we have a good way of segmenting words into morphs, we plan to move to modeling the more complex morphological phenomena.[2]

3 Prior Work

Interest in unsupervised morphology induction has surged since 2000. Detailed surveys are found, e.g., in [15,19], and in proceedings from a series of "Morpho-Challenge" workshops between 2005 and 2010 [17,18,20]. Our approach is founded on the Minimum Description Length principle (MDL) as a measure of model quality, see e.g., [14]. Linguistica, [11], also uses MDL, combined with the idea of a *signature*—set of affixes that belong to a morphological paradigm; e.g., suffixes for a certain class of nouns form one signature, etc. Our models also group morphs into different morphological *classes*, though using a different approach (Sect. 4).

Finite state machines (FSMs) are used in [12], which are similar to our FSMs; however, the approach there is less general, since it uses heuristics unsuitable for languages with very rich morphology.

The MDL-based Morfessor and its variants, e.g., [3,5], are closely related to our work. Unlike [4], we do not posit morphological classes *a priori*, but allow the model to learn the classes and distributions of morphs automatically. Word embedding, modeling semantic relations between words, have been used with orthographic features of words to learn the morphology of languages as "morphological chains" in [21].

[1] In the example above, the morph *kunno-* is an allomorph of *kunto*; which allomorph appears in a given word is determined by its phonological environment.

[2] Note, we do not claim that the problem must be sub-divided this way. As others before us, we begin with segmentation because it is more tractable.

Evaluation of morphological learning is a complex challenge in itself [23,26]; prior work on this topic is described in detail in Sect. 6.

The Morpho-Challenges have seen attempts to model aspects of morphology which we do not address: using analogical reasoning, handling *ablaut*-type morphology (as in German, and Semitic languages), etc. Beyond segmentation, modeling allomorphy has been attempted, e.g., by [24], but the performance of the proposed algorithms on segmentation so far falls short of those that do not model allomorphy. Research on induction of morphological structure is driven in part by the observation that children learn it at a very early age,[3] which makes acquisition by machine a fascinating challenge.

4 The Learning Algorithm: SMorph

Morphological systems for many languages are modeled by a finite-state machine (FSM), where each state corresponds to a morphological *class*. The seminal approach of *Two-Level Morphology*, [16], represents morphological grammar by a FSM. We can associate a state with, e.g., a set of verb stems that belong to a certain inflectional paradigm; or the set of suffixes in a certain noun paradigm, etc. The edges in the FSM define the *morphotactics* of the language—the permissible ordering among the states.

The data $\mathcal{D}$ (the *corpus*) is a large list of words in the given language. For every word $w \in \mathcal{D}$, SMorph tries to learn the most probable sequence of states that generates w: it treats the problem as finding a model that produces the most probable **segmentation** for each word w into "morphs"—fragments of w—and the **classification**—assignment of the morphs to classes (classes are identified with the states).[4]

The learning algorithm searches for the best model in a certain model class. Thus, the full description of the algorithm must specify a. the *model class* (Sect. 4.1), b. the *objective function*: a way of assigning a score—the cost—to each model (Sect. 4.3), and c. a *search strategy* for optimizing the objective across the model class (Sect. 4.2).

4.1 Morphology Models

We begin with a hidden Markov model (HMM), with a set of hidden *states/classes* $\{C_i\}$. States emit morphs with certain *emission* probabilities, and *transition* probabilities between states. To code each word w, the model starts at the initial state C_0. From state C_i, it can transition to another state C_j with a certain probability $P_{tr}(C_j|C_i)$ and emit a morph—a segment of w from C_j. The probability of emitting a morph μ from state C_j is denoted by $P_{em}(\mu|C_j)$. Once the entire w is emitted, the model transitions to the final state C_F and emits a special end-of-word symbol $\#$.

[3] This relates to the *poverty of stimulus* claim, [2], about human ability to learn complex systems from very limited data.

[4] Note that we evaluate only the *segmentation*, (Sect. 6).

Ideally, the states will correspond to “true” morphological classes; for example, nouns may fall into different declension paradigms and each paradigm would be assigned its own class/state in the model.

Probabilities P_{tr} and P_{em} are determined by counting how words are segmented into morphs, and which states emit which morphs. Many segmentations are possible for the given corpus. We need a way to choose the best segmentation for $\mathcal{D}$—the best parameters for the model. To approach this model selection problem via MDL we define a *code-length* for the data $\mathcal{D}$, which is the number of bits required to encode it.

4.2 Search

Algorithm 1. Baseline search algorithm, using Expectation-Maximization [10]

```
  Input: Data: a large list $\mathcal{D}$ of words in the language
1 Initialize: create a random segmentation and classification—split all words in $\mathcal{D}$ into
  morphs randomly, and assign morphs to classes randomly;
2 repeat
3     Compute Parameters: (E-step) based on the current segmentation and
      classification, compute all emission and transition costs;
4     Re-segment: (M-step) given the newly computed parameter values find the best
      segmentation for all words in the corpus (using Dynamic Programming, Sect. 4.6)
5 until convergence in cost;
```

Convergence is determined by the MDL code-length of the complete model (*cost*), defined in Sect. 4.3. We fix the number of classes K,[5] and begin with a random segmentation and classification (assignment of morphs to classes). We then greedily re-segment each word $w \in \mathcal{D}$, minimizing the MDL code-length of the model plus the data.

4.3 The Objective: Two-Part MDL Cost

Finding the best segmentation and classification can be viewed as the problem of *compressing* $\mathcal{D}$. In MDL 2-part coding, we try to minimize the cost of the *complete* data: the cost of the model plus the cost of the data given the model. In our case, this means summing the costs of coding the Lexicon, the transitions and the emissions:

$$L(\mathcal{D}) = L(M) + L(\mathcal{D}|M) = L(Lex) + L(Tr) + L(Em)$$

The cost of the **Lexicon:** $L(Lex)$, is the number of bits needed to encode each class, morph by morph, *irrespective of the order* of the morphs in the class:

$$L(Lex) = \sum_{i=1}^{K} \left[\sum_{\mu \in C_i} L(\mu) - \log |C_i|! \right] \tag{1}$$

[5] Ideally, K should be large, to give the model sufficient expressiveness.

where K is the number of classes, μ ranges over all morphs in class C_i, $L(\mu)$ is the code length of a morph μ (Eq. 2), and $|C_i|$ is the number of morphs in class C_i. The term $-\log|C_i|!$ accounts for the fact C_i is a set, and we do not need to code the morphs in C_i in any particular order.

The code length $L(\mu)$ of morph μ is computed similarly to [5], as the number of bits needed to encode μ:

$$L(\mu) = (|\mu| + 1) \cdot \log(|\Sigma| + 1) \tag{2}$$

where $|\mu|$ is the number of symbols in μ, and $|\Sigma|$ is the size of the alphabet; one is added to $|\Sigma|$ to account for one special *morph-boundary* symbol.[6]

Transitions: Given the lexicon, we code the paths of class transitions from C_0 to C_F, from word start to word finish, using Bayesian Marginal Likelihood (ML), as introduced in [9]. In ML, we treat each transition (C_iC_j) in the data as an "event" to be coded. If in a set of events $\mathcal{E} = \{E_j\}$, each E_j has a corresponding "prior" count α_j and count of observed occurrences O_j, the cost of coding $\mathcal{E}$ is:

$$L(\mathcal{E}) = -\sum_j \log\Gamma(O_j + \alpha_j) + \sum_j \log\Gamma(\alpha_j) + \log\Gamma\sum_j(O_j + \alpha_j) - \log\Gamma\sum_j \alpha_j \tag{3}$$

where the summations range over the set $\mathcal{E}$.[7] We use uniform priors, $\alpha_j = 1, \forall j$. Thus $\log\Gamma(\alpha_j) = 0$, and the second term is always zero in this equation.

To compute the cost of all transitions $L(Tr)$, we apply Eq. 3 to each class C_i, as i ranges from 0 to K; the set of "events" $\mathcal{E}$ is the set of all classes C_j, which are the targets of transitions from C_i:

$$\sum_{i=0}^{K}\left[-\sum_{j=1}^{K+1}\log\Gamma\Big(f(C_iC_j)+1\Big) + \log\Gamma\left(\sum_{j=1}^{K+1}\Big[f(C_iC_j)+1\Big]\right) - \log\Gamma\left(K\right)\right]$$

Emissions: We code the emissions $L(Em)$ analogously:

$$\sum_{i=1}^{K}\left[-\sum_{\mu\in C_i}\log\Gamma\Big(f(\mu,C_i)+1\Big) + \log\Gamma\left(\sum_{\mu\in C_i}\Big[f(\mu,C_i)+1\Big]\right) - \log\Gamma\left(|C_i|\right)\right]$$

where $f(C_i, C_j)$ is the count of transitions from C_i to C_j, and $f(\mu, C)$ is the number of times morph μ was emitted from class C. The cost $L(Em)$ is computed by applying Eq. 3, where the set of "events" $\mathcal{E}$ is now the set of emissions (μ, C_i) of all morphs μ from C_i.

[6] Another way to code $L(\mu)$ is to account for the *frequencies* of the symbols of the alphabet.

[7] For additional explanation about this coding scheme please refer to the original paper.

4.4 Input

For each language, we pre-process a large text by collecting *distinct* words from the text. We do not model the *text*—i.e., the distribution of words in the text—but the *language*, i.e., the observed distinct words, as is done in recent prior work. In this paper, *corpus* refers to the list of *distinct* words.[8]

4.5 Initialization

The initial segmentation and classification is obtained by randomly placing morph boundaries in each input word w, independently, according to a Bernoulli distribution,[9] and by randomly assigning the morphs to one of the classes in the Lexicon.

4.6 Dynamic Programming Re-segmentation

We compute the most probable segmentation of every word w in the corpus into morphs, at iteration t, given a set of transitions and emissions from iteration $t-1$.

We apply a Viterbi-like dynamic programming (DP) search for every word w, to compute the most probable path through the HMM, given w, *without* using the segmentation of w at iteration $t-1$. Standard Viterbi would only give us the best class assignment *given* a segmentation. Here, the search algorithm fills in the DP matrix starting from the leftmost column toward the right.

Notation: σ_a^b is a *substring* of w, from position a to position b, inclusive. We number positions starting from 1. The shorthand $\sigma^b \equiv \sigma_1^b$ is a prefix of w up to b, and $\sigma_a \equiv \sigma_a^n$ is a suffix, when $|w| = n$. A single **morph** μ_a^b lies between positions a and b in w. Note that σ_a^b is just a sub-string, and may contain several morphs, or cut across morph boundaries. In the cell (i, j), marked X, in the DP matrix, we compute the cost of the HMM being in state C_i and having emitted the prefix up to the j-th symbol of w, $L(C_i|\sigma^j)$. This cost is computed as the minimum over the following expressions, using values computed previously and already available in columns to the left of σ^j:

$$L(C_i|\sigma^j) = \min_{q,b} \Big(L(C_q|\sigma^b) + L(C_i|C_q) + L(\mu_{b+1}^j|C_i) \Big) \tag{4}$$

This says that the best way of getting to state C_i and emitting w up to σ^j is to come from some state C_q having emitted w up to σ^b, then jump from C_q to C_i,

[8] This is different from some of the earlier work, e.g., [3,5], and agrees with [25], who observe "that training on word types...give similar scores, while...training on word tokens, is significantly worse." We do not claim that there is no useful information in the distribution of words for learning morphology; however, the current models do not utilize it.

[9] With parameter ρ. In the current experiments we used $\rho = 0.20$ and 0.25. This is similar to the approach in [6], where they used a Poisson distribution with a fixed parameter.

and emit μ_{b+1}^{j} as a *single morph* from C_i. $L(C_i|C_q)$ is the cost of transition from state C_q to C_i. This can be calculated using the Bayesian Marginal Likelihood code-length formula as:

$$L(C_i|C_q) = \Delta L = L(Tr \cup t) - L(Tr) = -\log \frac{f(C_q C_i) + 1}{\sum_{k=0}^{K-1} \sum_{j=1}^{K} \big(f(C_k C_j) + 1\big)}$$

where t is the transition $C_q \rightarrow C_i$. The cost of emitting morph μ from class C_i is:

$$L(\mu|C_i) = \begin{cases} -\log \frac{f(\mu, C_i)+1}{f(C_i)+|C_i|} & \text{if } \mu \in C_i \\ -\log \frac{|C_i|}{\big(f(C_i)+|C_i|\big)\big(f(C_i)+|C_i|+1\big)} + L(\mu) & \text{if } \mu \notin C_i \end{cases}$$

The second case is for when μ is not emitted from C_i yet and does not exist in its lexicon and $L(\mu)$ is the cost of adding μ to the lexicon.

In Eq. 4, the minimum is taken over all states $q = 0, 1, \ldots, K$, including the initial state C_0, and over all columns b that precede column j: $b = j-1, \ldots, 0$. Here $L(\mu_{b+1}^{j}|C_i)$ is the cost of emitting the μ_{b+1}^{j} from C_i, for some $b < j$. For the empty string, $\sigma^0 \equiv \epsilon$, we set $L(C_0|\sigma^0) \equiv 0$ for the initial state, and $L(C_q|\sigma^0) \equiv \infty$ for $q \neq 0$.

The transition to the final state C_F is computed in the rightmost column of the matrix, marked #, using the transition from the last morph-emitting state—in column σ^n—to C_F. (State C_F always emits the word boundary #). Thus, the cost of the best path to generate w is:

$$L(w) = \min_{q=1,\ldots,K} L(C_q|\sigma^n) + L(C_F|C_q) + L(\#|C_F)$$

where the last factor $L(\#|C_F)$ is always 0. In addition to storing $L(C_i|\sigma^j)$ in cell (i, j) of the matrix, we store also the "best" (least expensive) state q and the column b from which we arrived at this cell. These values, the previous row and column, allow us to backtrack through the matrix at the end, to reconstruct the lowest-cost—most probable—path through the HMM.

5 Enhancements to Baseline Model

We next present enhancements to the baseline algorithm described in Sect. 4, which yield improvements in performance.

Simulated Annealing: The greedy search for the best segmentation for all words in the corpus quickly converges to local—far from global—optima. To avoid local optima, we use simulated annealing, with temperature T varying between fixed starting and ending values, T_0 and T_F, and a geometric cooling schedule, α. In Eq. 4, rather than using min to determine the *best* cell (q, b) from which to jump to X in the DP matrix, we select a candidate cell at random, depending on its cost, from a gradually narrowing window.[10]

[10] We use a standard approach to simulated annealing, for details please see [1].

This ensures that the model does not always greedily choose the best solution, and enables it to initially make random jumps to avoid local optima.

Next, as mentioned in the abstract, we introduce heuristics that constrain the search, based on simple yet general linguistic principles.

1. Directionality of the FSM: The FSM must be *directional*: the morphotactics of any language specify exactly the order in which morphological classes may follow one another. In many Indo-European languages, e.g., a word can have some prefixes, then a stem, then suffixes—always in a fixed order. Further, different kinds of suffixes have strict ordering among them—e.g., derivation precedes inflection.[11]
To enforce directionality, in Sect. 4.6, we constrain the DP matrix so that the preceding state q in Eq. 4 ranges only from 0 up to $i - 1$, rather than up to K. Since the states are ordered, this blocks transitions from a later state to an earlier one.
2. Natural Classes of Morphs: As a general principle, morph classes fall into two principal kinds: stems vs. affixes. We arbitrarily fix some range of states in the beginning to be *prefix* states, followed by a range of *stem* states, followed by *suffix* states.[12] A simple heuristic based on this is that the HMM must pass through at least one stem state during the DP search.
3. Bulk Re-segmentation: An important linguistic principle is that stems and affixes have very *different properties*. First, stem classes are usually *open*—i.e., potentially very large, whereas all affix classes are necessarily *closed*—very limited. This is reflected, e.g., in borrowing: one language may borrow any number of stems from another freely, but it is extremely unlikely to borrow a suffix.

Second, in general a randomly chosen affix is typically expected to occur *much* more frequently in the corpus than a random stem.[13]

Based on this principle, we introduce another heuristic to guide the search: after the normal re-segmentation step, we check all classes for "bad" morphs that violate this principle: very frequent morphs in stem classes, and very rare morphs in affix classes.[14] With a certain probability $\pi(T)$ which depends only on the simulated annealing temperature,[15] all words that contain a bad morph

[11] A problem for the directionality heuristic is compounding, where, e.g., in Finnish, the FSM can jump back to a stem class, even after some suffix classes, as seen in the examples in Sect. 2. We will model compounding in the future via a special "restart" state in the grammar. Despite this, even for Finnish, with its heavy compounding, the directional models still perform better than non-directional ones.

[12] Here we divide 15 available states as: 2 classes for prefixes, 6 for stems, and 7 for suffixes.

[13] This is true in general (though a language may have some exceptionally rarely used affix, which might happen to be less frequent than a very frequent stem).

[14] We model this via two hyper-parameters: s_{max} for maximum tolerated count of a stem, and a_{min} for minimum frequency of an affix. In the experiments, we set both to 100.

[15] For example, $\pi(T)$ can be e^{-T}.

are removed from the model *in bulk* (from the lexicon, and their transition and emission counts), and re-segmented afresh. When T is high, $\pi(T)$ is small; as T cools, $\pi(T) \to 1$.

6 Evaluation

Evaluation of morphology discovery is challenging, specifically since morphological analysis is limited to *segmentation*—here, as well as in most prior work—because in general it is not possible to posit definitively "correct" segmentation boundaries, which by definition ignore allomorphy.

An evaluation based on probabilistic sampling is suggested in [23]. Another scheme is suggested in the papers about the HUTMEGS "Gold-standard" evaluation corpus for Finnish, [6,7]. However, these approaches to evaluation are problematic, in that they ignore the issue of **consistency** of the segmentation.

[7] observe correctly that positing a single "proper" morphological analysis for a word w is not possible, in general. A motivating example is English $w = tries$: it can be analyzed as *tri+es* or *trie+s*. In actuality, w has two morphemes, which can have more than one allomorph—a stem {*try-/tri-*} or {*try-/trie-*}, and 3rd person suffix {*-s/-es*} or {*-s*}. Restricting morphological analysis to segmentation makes the problem ill-defined: it is not possible to posit a "proper" way to place the morpheme boundary and then to expect an automatic system to discover that particular way. HUTMEGS proposes "fuzzy" morpheme boundaries, to allow the system free choice within the bounds of the fuzzy boundary—as long as the system splits the word somewhere inside the boundary, it is not penalized.

Recent work has called into question the validity of evaluation schemes that disregard the *consistency* of segmentations across the data, considering such schemes as too permissive: the system should commit to one way of segmenting similar words—its chosen "theory"—and then *consistently* segment according to its theory—and should be penalized for violating its own theory by placing the boundaries *differently* in similar words. We follow the evaluation scheme in [22], which provides for gold-standard segmentations and an evaluation algorithm so as to enforce consistency while giving the learning algorithm maximal benefit of the doubt.

Consider again the example of *tries*. In the suggested gold standard, one annotates the segmentation neither as *tri-*$\underline{es}$ nor as *trie-*$\underline{s}$, but as $tri \overset{X}{\cdot} e \overset{X}{\cdot} s$—the special markers (dots) indicate that this segmentation is "ambiguous"—can be handled in more than one way. The label X identifies this particular kind of ambiguity. (The gold-standard defines a separate set of labels for each language.) In this case, the definition of X states that it accepts two "*theories*": $\{10, 01\}$ as valid—i.e., a morpheme boundary in the first position (10), or in the second (01), but not both (not 11; also not 00). Similar words: *cries*, *dries*, *flies* are then annotated similarly in the gold-standard.

Suppose the words *tries*, *flies*, and *applies* are in the gold standard, annotated with X as above, and the model segments them as: *trie-*$\underline{s}$, *flie-*$\underline{s}$, but *appli-*$\underline{es}$. We conclude that A. its preferred theory for handling X is to put the boundary in the second position (2 out of 3), and B. when it segmented *appli-es* it placed the boundary incorrectly: it violated its own preferred theory (defined by the majority of its choices). Thus its accuracy will be 2/3. The label X is used to co-index possible "ambiguous" boundaries in the gold standard, for a particular type of ambiguity. Annotators use these labels consistently across the evaluation corpus.

6.1 Experiments

We test on data from Finnish and Turkish. For each language, the corpus consists of about 100,000 distinct word forms, from texts of novels. Approximately 1000 words extracted at random from the data were annotated for the gold-standard by at least two annotators with knowledge of morphology and native proficiency.[16]

Ablation Studies: Shown in Table 1, confirm the gains in performance yielded by the heuristics. The enhancements to the basic algorithm are simulated annealing (SA), directionality (Dir), natural classes (NC) and bulk re-segmentation (Bu). Without SA, performance drops substantially. With SA, the table shows the gain obtained from adding all possible combinations of the heuristics. It might seem that adding directionality reduces the scores of the model, however, adding heuristics built upon directionality help the model achieve better results compared to the non-directional model.[17]

Comparison Studies: Are shown in Fig. 1, for Finnish and Turkish. Each point in the plots represents a single run of an algorithm. The coordinates of each point are its recall and precision; the accuracy of each run is in its label.

For comparison, we ran Morfessor CatMAP [8], on the same data, since it currently obtains the best performance over all Morfessor variants, as explained in [13]. FlatCat performs better than CatMAP with *semi-supervised* learning, but falls short of CatMAP performance in the unsupervised setting. CatMAP has a perplexity threshold parameter, b, "which indicates the point where a

Table 1. Ablation studies—Finnish

SA	Dir	NC	Bu	*Recall*	*Precis.*	*F-1*	*Accuracy*
−	−	−	−	30.83	70.79	42.96	75.94
+	−	−	−	34.30	79.48	47.92	78.06
+	+	−	−	33.93	77.80	47.25	77.70
+	+	+	−	34.77	73.32	47.17	77.65
+	+	+	+	**36.37**	**83.83**	**50.73**	**79.80**

[16] Disagreements between annotators were resolved. All annotated gold-standard data, for Finnish and Turkish, will be made publicly available with this paper on-line.

[17] To save space, we show results for Finnish only; other languages follow similar patterns (included in final paper).

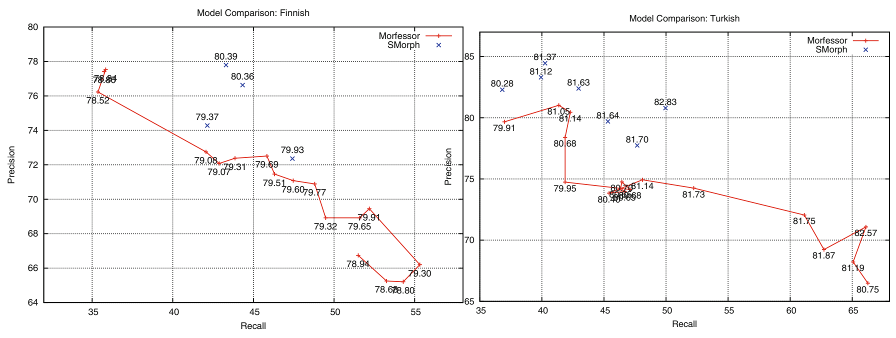

Fig. 1. Precision vs. recall: A—Finnish data, B—Turkish data (Color figure online)

morph is as likely to be a prefix as a non-prefix" [8]. This parameter trades off recall for precision; the more data there is, the higher b should be. As b grows, words are split less, giving higher precision but lower recall. Running Morfessor with varying $5 \leq b \leq 800$, yields the red line in the plots. SMorph also has hyper-parameters, which we test in the experiments. Probability ρ of a morph boundary between two adjacent symbols during the initial random segmentation, 0.20–0.25; the number of classes K, 15; assignment of classes to prefix, stem and suffix kinds, s_{max} and a_{min}.

The blue points in the plots correspond to runs of SMorph, with different settings of the hyper-parameters,[18] which can be optimized further, e.g., on a development corpus.

The runs of SMorph show an improvement in terms of recall and precision over Morfessor CatMAP: the blue points lie above the red curve. For example, *at a given level of recall*, SMorph reaches higher precision. For Finnish, the gain in precision is 2–8%; for Turkish, 2–7%. Conversely, *at a given level of precision*, SMorph reaches higher recall; for very large b, Morfessor reaches higher recall, but generally at a loss in precision. SMorph and Morfessor obtain similar accuracy values, though at a fixed level of recall SMorph has higher accuracy. More fine-grained effects of the hyper-parameters on performance are to be explored and investigated in future work.

Qualitative Evaluation of Classification: An important feature of SMorph is that the morph *classes* it learns are of a high quality. Manual inspection confirms that the classes group together morphs of a similar nature: noun-stem classes separate from verb stems, classes of affixes of similar kinds, etc.; hence the high precision.

As is natural in MDL, if some affixes appear frequently together, they will eventually be learned as a *single* affix; this explains lower recall. However,

[18] The parameters have not been tuned jointly; we started with values for the parameters as above, and checked the effect of varying them independently. Choice of the parameter values is driven by the observed total MDL cost (i.e., with no reference to the gold-standard evaluations).

this problem may be addressable as a post-processing step, after learning is complete.[19] (To be explored in future work.) Of course, evaluating the classes quantitatively is difficult, hence we evaluate quantitatively only the segmentations.

6.2 Error Analysis

A *non-directional* model trained on Finnish is shown in Fig. 2—with only 5 classes for clearer visualization. Each node shows the 8 most frequent morphs emitted from it, as well as the number of distinct morphs ($|Lex|$) and emission frequency ($freq$). Probabilities of transition between states are shown on the edges. (Edges with probability <0.02 are omitted for clarity.) The model has learned to often emit stems from states S_1 and S_5, and suffixes from S_2, S_3 and S_4. As expected, stem states are much larger than suffix states. They exhibit different properties: S_1 and S_5 have much flatter distribution, while S_3 and S_4 have spiked distributions: a few morphs with very high frequencies, and many with very low frequencies.

Further, S_1 has mostly verbal stems, whereas S_5 mostly nominal ones; S_4 is heavy on nominal suffixes, while S_2 has mostly verbal ones.[20]

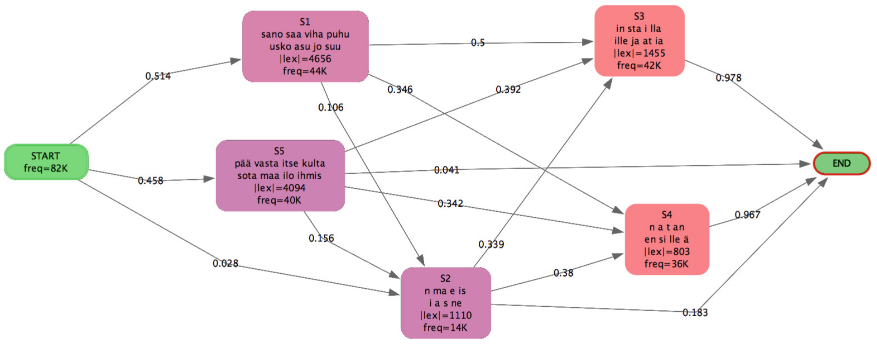

Fig. 2. Visualization of a model trained on Finnish data (with only 5 states)

7 Conclusions and Current Work

We have presented an algorithm for segmentation of a large corpus of words, which improves upon the state of the art. There are several important differences between SMorph and Morfessor models. SMorph tries to approach the

[19] Hence it is easier to recover from recall errors (false negatives) than from precision errors (false positives), and thus they are not equally important in this setting. Note that we do not consider F-score in the evaluation, but rather follow both recall and precision. F-score favors points where recall and precision are as near as possible. For example, whereas Morfessor trades off precision for recall to achieve a higher F-score, we do not consider it a benefit.

[20] This shows that the model indeed begins to resemble a FSM that we hope to achieve.

problem in a systematic way by grouping the discovered morphs into classes that respect general linguistic principles: directionality of morphotactics and natural differences between stems vs. affixes. It starts from a random initial model with no prior assumptions about the language, and learns to segment the data by optimizing a two-part cost function and Bayesian Marginal Likelihood, different from coding schemes used in prior work. The model is evaluated using a scheme, which avoids some of the problems in earlier evaluations.

To assure replicability, all gold-standard segmentations and code are made publicly available. Future improvements include those mentioned above, learning the optimal number of classes automatically, which should be reflected in the code-length, modeling compounding and allomorphy.

Acknowledgments. This research was supported in part by the FinUgRevita Project, No. 267097, of the Academy of Finland. We thank Hannes Wettig for his contributions to this work.

References

1. Černý, V.: Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. J. Optim. Theory Appl. **45**(1), 41–51 (1985)
2. Chomsky, N.: Rules and Representations. Basil Blackwell, Oxford (1980)
3. Creutz, M.: Unsupervised segmentation of words using prior distributions of morph length and frequency. In: Proceedings of 41st Meeting of ACL, Sapporo, Japan (2003)
4. Creutz, M.: Induction of a simple morphology for highly-inflecting languages. In: Proceedings of ACL SIGPHON, Barcelona, Spain (2004)
5. Creutz, M., Lagus, K.: Unsupervised discovery of morphemes. In: Proceedings of Workshop on Morphological and Phonological Learning, Philadelphia, PA, USA (2002)
6. Creutz, M., Lagus, K., Lindén, K., Virpioja, S.: Morfessor and Hutmegs: unsupervised morpheme segmentation for highly-inflecting and compounding languages. In: Proceedings of 2nd Baltic Conference on Human Language Technologies, Tallinn, Estonia (2005)
7. Creutz, M., Lindén, K.: Morpheme segmentation gold standards for Finnish and English. Technical report A77, HUT (2004)
8. Creutz, M., Lagus, K.: Inducing the morphological lexicon of a natural language from unannotated text. In: Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR-05), Espoo, Finland (2005)
9. Dawid, A.: Statistical theory: the prequential approach. J. Roy. Stat. Soc. A **147**(2), 278–292 (1984)
10. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. J. Roy. Stat. Soc. **39**(B), 1–38 (1977)
11. Goldsmith, J.: Unsupervised learning of the morphology of a natural language. ACL **27**(2), 153–198 (2001)
12. Goldsmith, J., Hu, Y.: From signatures to finite state automata. In: Midwest Computational Linguistics Colloquium, Bloomington, IN (2004)

13. Grönroos, S.A., Virpioja, S., Smit, P., Kurimo, M.: Morfessor FlatCat: an HMM-based method for unsupervised and semi-supervised learning of morphology. In: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics, Dublin, Ireland (2014)
14. Grünwald, P.: The Minimum Description Length Principle. MIT Press, Cambridge (2007)
15. Hammarström, H., Borin, L.: Unsupervised learning of morphology. Comput. Linguist. **37**(2), 309–350 (2011)
16. Koskenniemi, K.: Two-level morphology: a general computational model for word-form recognition and production. Ph.D. thesis, University of Helsinki, Finland (1983)
17. Kurimo, M., Creutz, M., Lagus, K. (eds.): Proceedings of the PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes. PASCAL European Network of Excellence, Venice (2006)
18. Kurimo, M., Turunen, V., Varjokallio, M.: Overview of morpho challenge 2008. In: Peters, C., et al. (eds.) CLEF 2008. LNCS, vol. 5706, pp. 951–966. Springer, Heidelberg (2009). doi:10.1007/978-3-642-04447-2_127
19. Kurimo, M., Virpioja, S., Turunen, V., Lagus, K.: Morpho-challenge 2005–2010: evaluations and results. In: Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology. Association for Computational Linguistics, Uppsala (2010)
20. Kurimo, M., Virpioja, S., Turunen, V.T.: Proceedings of the Morpho Challenge 2010 Workshop. Technical report, TKK-ICS-R37, Aalto University, School of Science and Technology, Department of Information and Computer Science, Espoo, Finland (2010)
21. Narasimhan, K., Barzilay, R., Jaakkola, T.: An unsupervised method for uncovering morphological chains. Trans. Assoc. Comput. Linguist. **3**, 157–167 (2015)
22. Nouri, J., Yangarber, R.: A novel evaluation method for morphological segmentation. In: Proceedings of LREC 2016: The Tenth International Conference on Language Resources and Evaluation, Portorož, Slovenia (2016)
23. Spiegler, S., Monson, C.: EMMA: a novel evaluation metric for morphological analysis. In: Proceedings of the 23rd International Conference on Computational Linguistics. Association for Computational Linguistics (2010)
24. Virpioja, S., Kohonen, O., Lagus, K.: Unsupervised morpheme analysis with allomorfessor. In: Peters, C., Di Nunzio, G.M., Kurimo, M., Mandl, T., Mostefa, D., Peñas, A., Roda, G. (eds.) CLEF 2009. LNCS, vol. 6241, pp. 609–616. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15754-7_73
25. Virpioja, S., Kohonen, O., Lagus, K.: Evaluating the effect of word frequencies in a probabilistic generative model of morphology. In: Proceedings of the NODALIDA Conference (2011)
26. Virpioja, S., Turunen, V.T., Spiegler, S., Kohonen, O., Kurimo, M.: Empirical comparison of evaluation methods for unsupervised learning of morphology. TAL **52**(2), 45–90 (2011). http://www.atala.org/IMG/pdf/2-Virpioja-TAL52-2-2011.pdf