Unsupervised Learning of Morphology with Graph Sampling

Maciej Sumalvico
University of Leipzig
NLP Group, Department of Computer Science
Augustusplatz 10, 04109 Leipzig
sumalvico@informatik.uni-leipzig.de

Abstract

We introduce a language-independent, graph-based probabilistic model of morphology, which uses transformation rules operating on whole words instead of the traditional morphological segmentation. The morphological analysis of a set of words is expressed through a graph having words as vertices and structural relationships between words as edges. We define a probability distribution over such graphs and develop a sampler based on the Metropolis-Hastings algorithm. The sampling is applied in order to determine the strength of morphological relationships between words, filter out accidental similarities and reduce the set of rules necessary to explain the data. The model is evaluated on the task of finding pairs of morphologically similar words, as well as generating new words. The results are compared to a state-of-the-art segmentation-based approach.

1 Introduction

The aim of unsupervised learning of morphology is to explain structural similarities between words of a language and discover mechanisms allowing one to predict or analyze unseen words using only a plain wordlist or an unannotated corpus as input. A vast majority of approaches to this problem concentrates on *morphological segmentation*, i.e. segmentation of words into minimal structural units, called *morphs* (Hammarström and Borin, 2011), usually employing statistical or machine learning methods.

While the segmentation approach has proven successful in some applications (especially compound splitting), it is known to suffer from serious limitations, like e.g. its intrinsic difficulty of handling non-concatenative morphology. Furthermore, phenomena like morphophonological rules (reflected by the orthography), allomorphs, or interactions between morphemes, are especially difficult to discover in an unsupervised setting, which is why most approaches limit themselves to rudimentary marking of postulated morpheme boundaries in the word's surface form. Because of the lack of morphotactical information, there is also no straightforward way to utilize such methods for generating new words: considering arbitrary morpheme sequences overgenerates massively and additional filtering approaches are needed, like the ones presented by (Rasooli et al., 2014). Finally, the notion of morpheme itself, although widely accepted in linguistics, has also been subject to criticism (e.g. Aronoff, 1976, 2007; Anderson, 1992).

An alternative linguistic theory, called Whole Word Morphology (Ford et al., 1997; Singh et al., 2003) (henceforth WWM), seems in our opinion particularly useful from the point of view of unsupervised learning. In WWM, the structural regularities between words are expressed as patterns operating simultaneously on multiple levels of lexical representation (phonological, syntactic, semantic). For example, the structural relationship between *cat* and *cats* would be captured by the following rule:[1]

$$\begin{bmatrix} \text{PHON:} & /X/ \\ \text{SYNT:} & \text{N, SG} \\ \text{SEM:} & \spadesuit \end{bmatrix} \leftrightarrow \begin{bmatrix} \text{PHON:} & /Xs/ \\ \text{SYNT:} & \text{N, PL} \\ \text{SEM:} & \text{many } \spadesuit \end{bmatrix} \tag{1}$$

The bidirectional arrow postulates, that if there is a word matching the pattern on one side of the rule, a counterpart matching the other side should also be a valid word. There is no notion of 'internal word structure' and neither of the words is

[1]For the sake of simplicity, we discard the allophony in the phonological representation and present the semantic layer in an extremely schematic way.

said to be morphologically 'more complex' than the other. The variable X stands for an arbitrary sequence of phonemes, which is retained on both sides of the rule. While the particular representations are further decomposable, for instance the phonological one in syllables and phonemes, the smallest unit of the language that combines all three layers is the word, and not the morpheme.

In this way, the morphological structure of a lexicon can be expressed as a graph, in which words constitute vertices and pairs of words following a regular structural pattern are connected with an edge, labeled with the corresponding rule. For the task of unsupervised learning, we see it as an advantage to learn a relation operating directly on observable objects (i.e. words), rather than some vaguely defined and theory-dependent underlying representation, like morpheme segmentation.

In the remainder of this paper, we briefly review previous unsupervised approaches to morphology (Sec. 2), present a generative probabilistic model for graphs of word derivations (Sec. 3), along with inference algorithms suited for unsupervised learning (Sec. 4). Our method is evaluated in Sec. 5.

2 Related Work

The approaches focusing on segmentation of words into smaller meaningful units have a long history, ranging from heuristic methods (Harris, 1955; Goldsmith, 2006) to the more recent approaches employing complete probabilistic models. A particularly successful example of the latter is Morfessor (Creutz and Lagus, 2005; Virpioja et al., 2013). Further probabilistic models include probabilistic hierarchical clustering (Can, 2011) and log-linear models (Poon et al., 2009). The yearly competition MorphoChallenge (Kurimo et al., 2010), which took place from 2005 until 2010, provides a good overview on the state of the art in morphological segmentation.

Another line of research concentrates on finding pairs of morphologically related words and using them to discover more general patterns. Various similarity measures are often used to identify such pairs, including orthographic and context similarity, or a combination of both (Yarowsky and Wicentowski, 2000; Baroni et al., 2002; Kirschenbaum, 2013). The first approach explicitly mentioning Whole Word Morphology as linguistic foundation has been presented by (Neuvel and Fulop, 2002), followed by a graph clustering method (Janicki, 2013) and a graph-based probabilistic model (Janicki, 2015). Recently, also word embeddings have been used to induce word-based transformation rules (Soricut and Och, 2015; Narasimhan et al., 2015). The latter method also produces graph-based representations of morphology, but is eventually evaluated on the segmentation task.

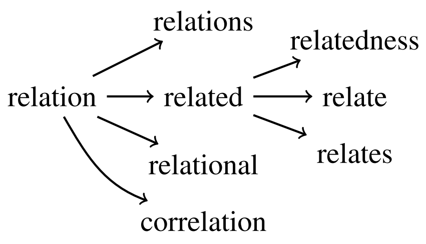


Figure 1: An example tree of word derivations. Edge labels are omitted for better readability.

While, to our best knowledge, sampling from a probability distribution over graph structures has not yet been applied in the task of learning morphology, it has been successfully applied in other NLP tasks, notably unsupervised dependency parsing (Mareček, 2012; Teichmann, 2014) and machine translation (Peng and Gildea, 2014).

3 The Probabilistic Model

The model described here is a slight modification of the one introduced by (Janicki, 2015). Our goal is a probability distribution $Pr(V, E|R)$ over graphs (consisting of a set of vertices V and a set of labeled edges E), given a set of transformation rules R. A single graph represents a hypothesis about the origin of words: the root nodes constitute the idiosyncratic part of the lexicon, i.e. they correspond to words, that are not derived from any other and must be memorized. Once some words are introduced, further ones can be derived using the transformation rules. Such derivations correspond to edges leading to new nodes. We restrict the allowed structure of the graphs to directed forests: they must not contain cycles and every node has at most one ingoing edge, i.e. every word has a unique analysis. Accordingly, we consider directed transformation rules.

An example of such graph is shown in Fig. 1. In this case, *relation* is the base for the whole word family: it is used to derive, among others, *related*, which in turn serves as a base for further words. Note that the tree shown in Fig. 1 does not constitute a linguistically correct analysis, nor does it have to. A unique 'correct' tree usually does not exist, e.g. the decision of whether the correct analysis of *correlation* should be *correlation* $\leftarrow$ *correlate* $\leftarrow$ *relate* or *correlation* $\leftarrow$ *relation* $\leftarrow$ *relate*, or yet another, is from our point of view completely arbitrary. This is the reason why we are not going to focus on single graphs, but rather on large samples, in which such multiple possibilities are accounted for.

Moving on to the probabilistic formulation of the model outlined above, let ρ be a probability distribution over strings, which should be nonzero at least for all well-formed words of the language. The root nodes of the graph are drawn from ρ. Then, we iteratively try to apply every rule on every word, resulting in a (possibly empty) set of candidate derivations. Each derivation is an independent[2] Bernoulli trial with probability θ_r: if it succeeds, the new word is introduced into the graph, along with the edge deriving it. We thus arrive at the following distribution:[3]

$$Pr(V,E|R,\Theta_R) \propto \prod_{v \in V_0} \rho(v) \times \prod_{v \in V} \prod_{r \in R} \prod_{v' \in r(v)} \begin{cases} \theta_r & \text{if } \langle v,v',r\rangle \in E, \\ 1-\theta_r & \text{if } \langle v,v',r\rangle \notin E \end{cases} \tag{2}$$

V_0 denotes the set of root nodes in the graph $\langle V, E\rangle$. Θ_R is the parameter vector, which contains θ_r values for each rule r. $r(v)$ is the set of words derived from v by applying r. Note that $r(v)$ might be empty if v does not match the left-hand side of r. As to the distribution ρ, we learn it from the observed vocabulary V using the ALERGIA algorithm (de la Higuera and Thollard, 2000; de la Higuera, 2010), which learns distributions over strings from samples. Once learnt, it is considered fixed and not changed by our inference procedure.

Furthermore, let n_r denote the number of edges in the graph $\langle V, E\rangle$ labeled with r and m_r denote the total number of words, that can be derived with r from words from V. More formally:

$$n_r := |\{\langle v,v',r'\rangle \in E : r' = r\}| \tag{3}$$

$$m_r := \sum_{v \in V} |r(v)| \tag{4}$$

Then (2) can be rewritten as:

$$Pr(V,E|R,\Theta_R) \propto \prod_{v \in V_0} \rho(v) \prod_{r \in R} \theta_r^{n_r}(1-\theta_r)^{m_r-n_r} \tag{5}$$

From (5), we can easily see, that the number of edges derived by a rule follows a Binomial distribution. Following the Bayesian approach, we use Beta distribution as conjugate prior for θ_r:

$$\theta_r \sim Be(\alpha,\beta) \tag{6}$$

In practice, α and β are always set to 1.1 (an almost uniform prior, but with exclusion of extreme values). The parameters can be integrated out:

$$\begin{aligned} Pr(V,E|R) &= \int_{(0,1)^{|R|}} Pr(V,E|R,\Theta_R) \\ &\qquad \times Pr(\Theta_R|R)d\Theta_R \\ &\propto \prod_{v \in V_0} \rho(v) \prod_{r \in R} \frac{B(n_r+\alpha, m_r-n_r+\beta)}{B(\alpha,\beta)} \end{aligned} \tag{7}$$

Finally, we introduce a probability distribution for the rule set. For this purpose, we represent the rules as sequences of edit operations. For example, the rule $/X\text{ation}/ \rightarrow /X\text{ate}/$ deriving *relate* from *relation* is represented as the following sequence:

$$\star \texttt{ a:a t:t i:e o:0 n:0 \#} \tag{8}$$

The item denoted by $\star$ means leaving an arbitrary sequence of characters unchanged and # is a special item terminating the sequence. The probability $\pi(r)$ of a rule is simply a product of the probabilities of individual sequence items (i.e. edit operations), which are treated as symbols of an alphabet. The item probabilities are learnt from the initial set of candidate rules and considered fixed, just like the ρ distribution for root words. The

[2]The well-formedness conditions of the graphs do not introduce any dependency between the edges. Note that each rule application derives a new word, so each edge leads to a new node. If multiple rules derived the same word, it should correspond to multiple graph nodes with the same label. However, our set of vertices V (which is considered given and fixed) contains one node per word, so such graphs are ruled out.

[3]The proportionality sign is due to the first term: the probability of a random set of independently chosen elements is proportional, but not equal, to the product of the probabilities of the elements. However, with the right choice of set size distribution (Poisson with mean 1), we can obtain a fixed and simple normalizing constant e^{-1}. The choice of Poisson distribution is motivated only by mathematical convenience. See (Janicki, 2015) for a more detailed explanation.

probability of the rule set is then proportional to the product of probabilities of individual rules:

$$Pr(R) \propto \prod_{r \in R} \pi(r) \tag{9}$$

The distribution $Pr(R)$ penalizes complex models: it attributes lower probabilities to large rule sets and specific, sophisticated rules.

4 Inference

While (Janicki, 2015) used an optimization approach to find the single best graph, our goal is to base the inference on large samples from the distribution over graphs using a Markov Chain Monte Carlo sampler. In particular, we are going to be interested in expected values of certain functions of the graph with respect to the edge structure, which is a latent variable. Thus, for a given function h (e.g. measuring the frequency of a rule or the likelihood of the graph), we want to be able to approximate the expectations of the following form:

$$\mathbb{E}_{E|V,R} h(V,E) = \sum_{E} h(V,E) Pr(E|V,R) \tag{10}$$

4.1 Finding Candidate Rules

We begin by extracting all pairs of words that might be morphologically related from the given vocabulary. We follow the approach of (Janicki, 2015): we apply a modified FastSS algorithm (Bocek et al., 2007) to find all pairs of words, that differ with at most 5 characters at the beginning and at the end and with at most 3 characters within a single slot inside the word. Those numbers are configurable parameters, but we have found the above values to be sufficient to capture most morphological rules in many languages.

Each pair of words is used to extract a number of transformation rules of the form (cf. (1)):

$$/a_1 X_1 a_2 X_2 a_3/ \rightarrow /b_1 X_1 b_2 X_2 b_3/ \tag{11}$$

where X_1, X_2 are variable elements, a_1, a_3, b_1, b_3 are constants of at most 5 characters, and a_2, b_2 are constants of at most 3 characters.[4] There are usually many rules that can be derived from a single pair of words, depending on whether we assign the common substrings of both words to the variable elements X_1, X_2, resulting in more general rules, or to the constants, resulting in more specific rules.

[4]In case a_2 and b_2 are both empty, X_1 and X_2 are merged into a single variable X.

For example, three of many rules that can be extracted from the pair ⟨*relate*, *correlate*⟩ are given below:

$$\begin{aligned} /X/ &\rightarrow /\text{cor}X/ \\ /\text{r}X/ &\rightarrow /\text{corr}X/ \\ /\text{r}X\text{ate}/ &\rightarrow /\text{corr}X\text{ate}/ \end{aligned} \tag{12}$$

In order to reduce the number of candidate rules and edges to tractable sizes, we apply the following filtering criteria:

max. number of rules: only N_{max} most frequent rules are kept;

max. number of edges per word pair: only k most general rules are extracted from each word pair;

min. rule frequency: only rules occurring in at least n_{min} word pairs are kept.

In all our experiments, we used fixed values of $N_{\text{max}} = 10000$, $k = 5$, $n_{\text{min}} = 3$, which led to good results for training data of various sizes. Those values can thus be safely used as default setting, without the need of parameter tuning.

4.2 The Sampler

Having a set of rules R and a vocabulary V, we are ready to draw samples of edge sets from the distribution $Pr(E|V,R)$, which is proportional to $Pr(V,E|R)$ up to a normalizing constant. In the following, we will define a sampler based on the Metropolis-Hastings algorithm, which belongs to a larger family of Markov Chain Monte Carlo methods.

The key idea of Markov Chain Monte Carlo methods is, given a probability distribution f, to construct a Markov chain, for which f is the limiting distribution (Robert and Casella, 2005). The Metropolis-Hastings algorithm (Metropolis et al., 1953; Hastings, 1970) provides a relatively simple recipe for such chain. It involves an instrumental distribution $q(y|x)$ used to propose the next item y of the sample given the current item x. Then, the next item of the sample is taken to be either y or x, depending on whether the proposal is accepted or rejected. The probability of acceptance is:

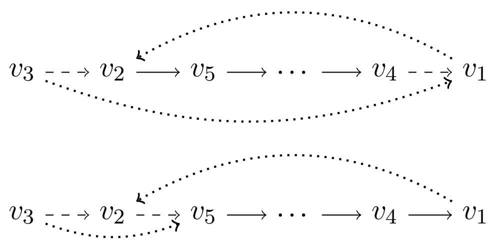
$$\alpha(x,y) = \min \left\{ \frac{f(y)}{f(x)} \frac{q(x|y)}{q(y|x)}, 1 \right\} \tag{13}$$

Note that the distributions f and q only have to be known up to a multiplicative constant, since the computation only involves quotients of their values. Also, q does not have to bear any relationship to f. It is sufficient that every point of the sample space, for which f is nonzero, can be reached from any other in a finite number of steps using q. Then (13) implies that the resulting Markov chain has f as its limiting distribution. Although the sample obtained from such chain is not independent, it can be used for approximating expectations like (10).

The preprocessing step described in the previous section provides us with a set of all candidate edges $\mathcal{E}$, which can be created by the rules from R. The idea for a proposal distribution $q(\cdot|E)$, given the current set of edges E, is the following: we choose an edge e uniformly from $\mathcal{E}$. If it is already present in E, we delete it, if not, we add it. In this way, we obtain a proposal for the next set of edges E'. Note that in this way we can reach any graph from any other in a finite number of steps by first deleting all the edges of the first graph and then adding all the edges of the second graph one by one.

There is however one problem with this approach: while deleting an edge is always possible, adding one could result in an ill-formed graph (e.g. create a cycle). If we simply discarded such proposals and proposed staying with the current graph instead, such procedure would still yield a valid instrumental distribution and a valid MCMC sampler. While the resulting chain would still be theoretically correct, its so-called *mixing properties* would be very bad, meaning that it would need an extremely large number of iterations to converge to the limiting distribution. For this reason, we are going to introduce additional moves which make adding of an edge possible in most cases.

Let's consider adding a new edge $v_1 \xrightarrow{r} v_2$. The first problematic case is when the new edge would create a cycle, i.e. v_1 is already a descendent of v_2. Let v_3 denote the parent of v_2, v_5 the immediate child of v_2 which leads to v_1 and v_4 the parent of v_1 (see Fig. 2). We have two possibilities to add an edge between v_1 and v_2 by 'cutting out' either v_1 or v_2 from its place in the tree. Both variants involve adding and deleting two edges. We refer to this kind of operation as 'flip'. The variant to be used is chosen randomly each time a 'flip' move is to be applied. Note that a 'flip' move might be impossible if the other edge to be added ($v_3 \rightarrow v_1$ or $v_3 \rightarrow v_5$, respectively) is not available in $\mathcal{E}$.



$$\begin{tikzcd} v_3 \arrow[r, dashed] \arrow[rrrrr, dotted, bend right] & v_2 \arrow[r] & v_5 \arrow[r] & \cdots \arrow[r] & v_4 \arrow[r, dashed] & v_1 \arrow[llll, dotted, bend right] \end{tikzcd}$$

$$\begin{tikzcd} v_3 \arrow[r, dashed] \arrow[rr, dotted, bend right] & v_2 \arrow[r, dashed] & v_5 \arrow[r] & \cdots \arrow[r] & v_4 \arrow[r] & v_1 \arrow[llll, dotted, bend right] \end{tikzcd}$$

Figure 2: The two variants of the 'flip' operation. The deleted edges are dashed and the newly added edges dotted. The goal of the operation is to make possible adding an edge from v_1 to v_2 without creating a cycle.

In this case, we propose staying with the current graph, which, as already mentioned, does not undermine the validity of the sampler. The absence of other involved edges (e.g. $v_3 \rightarrow v_2$ in case v_2 is a root) or the overlapping of some nodes (e.g. $v_5 = v_1$ and $v_4 = v_2$ if v_1 is a child of v_2 directly) can be simply ignored as the move is still valid.

Another problematic case arises when the newly added edge $v_1 \xrightarrow{r} v_2$ would not create a cycle, but v_2 already has an ingoing edge (possibly even from v_1, but with a different label). In this case, we simply exchange the previous ingoing edge of v_2 for the new one.

It is important to point out, that all moves are reversible and that the resulting proposal distribution is symmetric. The move is uniquely determined by the choice of the edge e to be added or deleted, except for the 'flip' case, where additionally one of the two variants of the move is randomly chosen. In case of adding or removing a single edge without side effects, selecting the same edge again reverses the operation. If the edge to be added is exchanged for another, selecting the previously deleted edge will reverse it. Finally, the first variant of 'flip' on $v_1 \rightarrow v_2$ can be reversed by the second variant on $v_4 \rightarrow v_1$ and the second variant on $v_1 \rightarrow v_2$ can be reversed by the first variant on $v_2 \rightarrow v_5$.

Note, that the complexity of a single sampling iteration is only $O(h)$, with h being the maximum height of a tree, since h operations are needed to determine whether v_1 is a descendant of v_2. In practice, h is usually bounded by a small constant. Also, we do not need to memorize the graphs seen in the sample, we only have to keep track of the current graph and the current values of the expectations we are interested in. The latter can be updated dynamically. A single sampling iteration is thus computationally very cheap, which enables us to draw large samples. In the experiments, we use sample sizes of 50 million graphs.

4.3 Model Selection

The sampler developed in the previous section has multiple applications. One of them is *model selection*, i.e. the task of selecting an optimal set of rules.[5] For the purpose of comparing models, we introduce the following expected log-likelihood function:

$$\begin{aligned}\ell(R) = \;& \mathbb{E}_{E|V,R} \ln Pr(V,E|R) + \ln Pr(R) \\ = \;& \mathbb{E}_{E|V,R}\Big[\sum_{v\in V_0} \ln \rho(v) \\ & + \sum_{r\in R} \ln B(n_r+\alpha, m_r-n_r+\beta) \\ & - |R| \ln B(\alpha,\beta)\Big] + \sum_{r\in R} \ln \pi(r) \quad (14)\end{aligned}$$

We use the graph sampler for computing the value of $\ell(R)$ for a single rule set. The model selection task is to maximize ℓ, i.e. to find:

$$\hat{R} = \arg\max_R \ell(R) \quad (15)$$

For this optimization task, we use the *simulated annealing* algorithm (Kirkpatrick et al., 1983), which is closely related to MCMC sampling. In this algorithm, we also use an instrumental distribution $q(y|x)$ to propose the next point y given the current point x. Let h denote the function to optimize. Then the acceptance probability of y while being at x is defined as:

$$\alpha(x,y) = \min\left\{\frac{q(x|y)}{q(y|x)} e^{\frac{h(y)-h(x)}{t}}, 1\right\} \quad (16)$$

The parameter t is called *temperature* and is meant to decrease with each iteration, thus decreasing the algorithm's willingness to move to points with lower h value. As (Besag, 2004) points out, if we consider a distribution $f(x) \propto \exp h(x)$, then simulated annealing corresponds to MCMC sampling from a distribution proportional to $f^{\frac{1}{t}}$. As t goes towards zero, the latter will converge to a uniform distribution over global maxima of h. In this way, the algorithm is able to find a global maximum and escape local maxima.

[5]Each set of rules R defines a *model*, parametrized by Θ_R.

We are now going to define the instrumental distribution $q(R'|R)$ used to propose the next rule set. All considered rule sets are going to be subsets of $\mathcal{R}$, which denotes the set of rules obtained after the preprocessing step described in Sec. 4.1. We begin by defining the *score* of a single rule, which is the expected contribution of this rule to the log-likelihood of the graph:

$$\begin{aligned}\zeta(r) =& \mathbb{E}_{E|V,R}\Big[-\sum_{\substack{\langle v,v',r'\rangle\in E\\ r'=r}} \ln \rho(v') \\ & + \ln B(n_r+\alpha, m_r-n_r+\beta) \\ & - \ln B(\alpha,\beta)\Big] + \ln \pi(r) \quad (17)\end{aligned}$$

The first term corresponds to the log-likelihood, that is gained by deriving words using r, rather than having them as roots. We subsequently apply the logistic function to turn the rule score into a probability:

$$g(r) = \frac{1}{1+\exp(-\gamma\cdot\zeta(r))} \quad (18)$$

where γ is a configurable parameter. $g(r)$ is the probability of r being selected to the next rule set. Note that $g(r)$ is also defined for rules not occurring in the current rule set: the first term and n_r are then simply always zero. Finally, we propose the next rule set R' by randomly determining for each rule from $\mathcal{R}$, whether it is going to be selected or not. Basing the selection probability on $\zeta(r)$ means that the chances of a rule to 'survive' in the next iteration depend on its usefulness in the current iteration. Once a rule is dropped, it can still be re-selected, but the probability of it happening is rather small. Finally, γ is set to be equal to inverse temperature of the annealing algorithm, which means that the importance of rule performance increases over time, allowing less and less changes in later iterations.

In practice, the cost of one iteration of the annealing algorithm is high, since it involves a whole run of the graph sampler. We are thus only going to perform a few annealing iterations (typically 20 or 30) which is obviously not enough to find the global maximum. Thus, we use this algorithm only as a local optimizer: to reduce the set of rules as much as possible within a reasonable runtime. However, it is still a plausible choice because of its flexibility, e.g. being able to move across a parameter space composed of discrete sets (unlike e.g.

	training	testing
English	45,391	15,608
German	76,813	221,190
Dutch	103,190	145,354

Table 1: The size (number of words) of datasets used for the analysis task.

the EM algorithm, which requires a space of real-valued vectors).

4.4 Model Fitting

Once the optimal model is selected, we can determine optimal values for the parameter vector Θ_R. For this purpose, we use the Monte Carlo EM algorithm (Wei and Tanner, 1990), which is a variant of EM using a sampler to approximate the expected values. In the maximization step, we set the θ_r values to their maximum a-posteriori likelihood estimates:

$$\theta_r^{(t+1)} := \frac{\mathbb{E}_{E|V,R,\Theta_R^{(t)}}[n_r] + \alpha - 1}{m_r + \alpha + \beta - 2} \tag{19}$$

5 Evaluation

We evaluate our approach on the tasks of morphological *analysis* and *generation*.

5.1 Analysis

As we intend to avoid postulating internal word structure, the task of analysis is defined as linking an unknown word to morphologically similar words from a known lexicon. We consider a pair of words to be morphologically similar if one word can be derived from the other using a single morphological operation (e.g. affix insertion, deletion or substitution). Pairs of morphologically similar words can also be extracted from segmentations by computing edit distance on morpheme sequences – words are considerered morphologically similar if such distance is equal to 1 and the difference does not include stems.

Dataset We use the CELEX lexical database to derive training and evaluation data. CELEX provides data for English, German and Dutch, including morphological segmentation, labeling of inflectional classes and corpus frequency. For training, we use words with nonzero corpus frequency, while the remaining words, i.e. those with zero frequency, constitute the testing dataset. The size of the datasets is shown in Table 1.

	–MS	+MS
English	9,434	3,943
German	8,432	4,888
Dutch	9,026	4,743

Table 2: The model size (number of rules) before and after model selection.

Language	Model	Precision	Recall	F-score
English	Morfessor	74.4 %	41.4 %	53.2 %
	–MS	38.5 %	73.2 %	50.5 %
	+MS	53.8 %	69.3 %	60.6 %
German	Morfessor	75.1 %	38.8 %	51.2 %
	–MS	56.4 %	67.8 %	61.6 %
	+MS	65.1 %	62.8 %	63.9 %
Dutch	Morfessor	73.1 %	39.2 %	51.0 %
	–MS	42.5 %	68.1 %	52.3 %
	+MS	54.1 %	63.1 %	58.3 %

Table 3: Results for the analysis task.

Experiment setup We report the results for two different setups: with model selection (+MS) and without it (–MS). We also compare our results to Morfessor Categories-MAP (Creutz and Lagus, 2005). In addition to morpheme segmentations, this tool labels each morpheme as prefix, affix or stem, which enables us to convert its output into morphologically similar pairs using the edit distance method.[6] The training dataset also constitutes the known lexicon, i.e. we evaluate pairs of similar words (w_1, w_2) with w_1 coming from the training and w_2 from the testing dataset.

Results The results are shown in Table 3. The setup with model selection achieves the best performance for all three languages, which demonstrates the usefulness of this step. The difference is especially clear for English, where the setup without model selection performs worse than Morfessor. Our method also manages to capture non-concatenative morphology: pairs like German (*findet*, *fändet*) or Dutch (*televisiekanalen*, *televisiekanaal*), are successfully detected. On the other hand, the method based on Morfessor segmentations suffers from low recall: even small errors in segmentation can result in a failure to discover many similar pairs.

Table 2 shows another benefit of model selection: reducing the number of rules approximately by half, which has a significant impact on the speeed of the analysis.

[6]Unfortunately, such conversion is not possible for most other approaches to unsupervised segmentation, which do not provide additional labeling for stems.

Language	Model	1k	5k	10k	50k	100k	200k	500k
English	–MS, –F	72.2 %	52.5 %	48.4 %	37.3 %	26.9 %	20.4 %	11.1 %
	–MS, +F	69.8 %	64.9 %	62.3 %	42.5 %	31.1 %	22.3 %	17.5 %
	+MS, +F	71.9 %	66.4 %	62.7 %	43.4 %	32.7 %	22.9 %	17.0 %
German	–MS, –F	85.6 %	80.0 %	74.3 %	56.0 %	42.3 %	32.2 %	17.5 %
	–MS, +F	86.0 %	82.0 %	83.7 %	68.0 %	51.2 %	35.7 %	21.4 %
	+MS, +F	86.4 %	83.0 %	83.9 %	67.8 %	51.5 %	35.9 %	21.1 %
Dutch	–MS, –F	50.0 %	53.3 %	53.3 %	31.6 %	24.6 %	18.0 %	10.3 %
	–MS, +F	73.9 %	60.1 %	60.5 %	38.7 %	29.4 %	20.6 %	11.2 %
	+MS, +F	73.9 %	62.2 %	62.2 %	38.8 %	29.8 %	21.0 %	11.4 %

Table 4: Results for the generation task. The scores show precision depending on the number of generated words. (‘k’ means ‘thousand’).

5.2 Generation

In generation task, we evaluate the capability of the model to derive new, valid words using the learnt morphological rules. As the recall measure for this task is hardly possible to compute (it would involve listing all and only words derivable from the training set), we evaluate the precision against the number of generated words.

Dataset For training, we use the same datasets as in the analysis task (CELEX words with nonzero corpus frequency). The testing wordlists, which are supposed to contain at least all words obtainable from the training words within a single morphological operation, are built by merging the complete CELEX vocabulary with lists of Wiktionary entries for the relevant language.

Experiment setup We conduct the experiment on three different settings, depending on whether the model selection (MS) and the fitting (F) step are carried out or omitted.[7] In case of omitting the fitting step, the rule probabilities (θ_r) are set according to (19), but using maximum rule frequency, i.e. the number of candidate edges in $\mathcal{E}$ labeled with this rule, as n_r. The generated words are ordered by their contribution to the log-likelihood of the data, which is equivalent to sorting according to the probability of the rule used to derived the word.

For this task, we do not include any comparison to a segmentation-based approach. Such approaches usually do not model morphotactics at all, or do it in a very simple way (like Morfessor Categories-MAP). Applying them to generate new words leads to a disastrous overgeneration (e.g. every common morpheme can be repeated many times), which renders a comparison pointless. Segmentation models are simply not designed for the generation task.

Results The results shown in Table 4 highlight the importance of a proper fitting step, which in most cases improves the results considerably. On the other hand, the impact of the model selection step is rather insignificant. This is understandable: the goal of model selection is to eliminate weak, useless rules, while in the word generation task, the strongest rules are applied first.

6 Conclusion

We have presented a method for unsupervised discovery of pairs of morphologically related words without performing morpheme segmentation. Our method is based on a probabilistic model describing graphs of word derivations. We developed a Markov Chain Monte Carlo sampler for such graphs, which allows us to approximate expectations over the latent edge structure. The evaluation results confirm our intuition, that the discovery of morphologically related words, as well as prediction of unseen words, is easier without the need of morpheme segmentation. While we limited ourselves to unannotated string forms of words, the formalism can be easily extended to include various other word features, like POS-tags or frequency. In the further work, we especially intend to follow the recent trend and include word embeddings as a feature. Furthermore, we are going to examine practical applications of a whole-word morphology model, like e.g. increasing the performance of statistical language models in dealing with unknown words. We believe that morpheme segmentation is neither sufficient nor necessary for learning morphology, especially in the unsupervised setting.

[7]The results for the +MS, –F case are omitted, because they are very similar to –MS, –F, and thus do not contribute anything noteworthy to the evaluation.

References

Stephen R. Anderson. 1992. *A-Morphous Morphology*.

Mark Aronoff. 1976. *Word Formation in Generative Grammar*. MIT Press.

Mark Aronoff. 2007. In the Beginning was the Word. *Language* 83(4):803–830.

Marco Baroni, Johannes Matiasek, and Harald Trost. 2002. Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. In *Proceedings of the 6th Workshop of the ACL Special Interest Group on Phonology*. volume 6, pages 48–57.

Julian Besag. 2004. An introduction to Markov chain Monte Carlo methods. In Mark Johnson, Sanjeev P. Khudanpur, Mari Ostendorf, and Roni Rosenfeld, editors, *Mathematical Foundations of Speech and Language Processing*, Springer-Verlag New York, Inc., pages 247–270.

Thomas Bocek, Ela Hunt, and Burkhard Stiller. 2007. Fast Similarity Search in Large Dictionaries. Technical report, University of Zurich.

Burcu Can. 2011. *Statistical Models for Unsupervised Learning of Morphology and POS Tagging*. Ph.D. thesis, University of York.

Mathias Creutz and Krista Lagus. 2005. 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)*.

Colin de la Higuera. 2010. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press, New York, NY, USA.

Colin de la Higuera and Franck Thollard. 2000. Identification in the Limit with Probability One of Stochastic Deterministic Finite Automata. In *ICGI '00*. pages 141–156.

Alan Ford, Rajendra Singh, and Gita Martohardjono. 1997. *Pace Pāṇini: Towards a word-based theory of morphology*. American University Studies. Series XIII, Linguistics, Vol. 34. Peter Lang Publishing, Incorporated.

John Goldsmith. 2006. An algorithm for the unsupervised learning of morphology. *Natural Language Engineering* 12(1):353.

Harald Hammarström and Lars Borin. 2011. Unsupervised Learning of Morphology. *Computational Linguistics* 37(2):309–350.

Zellig S Harris. 1955. From phoneme to morpheme. *Language* 31(2):190–222.

Wilfred K. Hastings. 1970. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika* 57(1):97–109.

Maciej Janicki. 2013. Unsupervised Learning of A-Morphous Inflection with Graph Clustering. In *Proceedings of the Student Research Workshop associated with RANLP 2013*. pages 93–99.

Maciej Janicki. 2015. A Multi-purpose Bayesian Model for Word-Based Morphology. In Cerstin Mahlow and Michael Piotrowski, editors, *Systems and Frameworks for Computational Morphology – Fourth International Workshop, SFCM 2015*. Springer.

S. Kirkpatrick, C.D. Gelatt, and M. P. Vecchi. 1983. Optimization by Simulated Annealing. *Science* 220(4598):671–680.

Amit Kirschenbaum. 2013. Unsupervised Segmentation for Different Types of Morphological Processes Using Multiple Sequence Alignment. In *1st International Conference on Statistical Language and Speech Processing, SLSP*. Tarragona, Spain, pages 152–163.

Mikko Kurimo, Sami Virpioja, Ville Turunen, and Krista Lagus. 2010. Morpho Challenge 2005-2010: Evaluations and results. In *Proceedings of the 11th Meeting of the ACL-SIGMORPHON, ACL 2010*. pages 87–95.

David Mareček. 2012. *Unsupervised Dependency Parsing*. Ph.D. thesis, Charles University in Prague.

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. 1953. Equation of state calculations by fast computing machines. *Journal of Chemical Physics* 21(6):1087–1092.

Karthik Narasimhan, Regina Barzilay, and Tommi S. Jaakkola. 2015. An unsupervised method for uncovering morphological chains. *TACL* 3:157–167.

Sylvain Neuvel and Sean A. Fulop. 2002. Unsupervised Learning of Morphology without Morphemes. In *Proceedings of the 6th Workshop of the ACL Special Interest Group in Computational Phonology (SIGPHON)*. pages 31–40.

Xiaochang Peng and Daniel Gildea. 2014. Type-based MCMC for Sampling Tree Fragments from Forests. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)* pages 1735–1745.

Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proceedings of Human Language Technologies The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics on NAACL 09*. page 209.

Mohammad Sadegh Rasooli, Thomas Lippincott, Nizar Habash, and Owen Rambow. 2014. Unsupervised Morphology-Based Vocabulary Expansion. In *ACL*. pages 1349–1359.

Christian P. Robert and George Casella. 2005. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Rajendra Singh, Stanley Starosta, and Sylvain Neuvel. 2003. *Explorations in Seamless Morphology*. SAGE Publications.

Radu Soricut and Franz Josef Och. 2015. Unsupervised Morphology Induction Using Word Embeddings. In *NAACL 2015*. pages 1626–1636.

Christoph Teichmann. 2014. *Markov Chain Monte Carlo Sampling for Dependency Trees*. Ph.D. thesis, University of Leipzig.

Sami Virpioja, Peter Smit, Stig-Arne Grönroos, and Mikko Kurimo. 2013. Morfessor 2.0: Python Implementation and Extensions for Morfessor Baseline. Technical report, Aalto University, Helsinki. http://urn.fi/URN:ISBN:978-952-60-5501-5.

Greg C. G. Wei and Martin A. Tanner. 1990. A Monte Carlo Implementation of the EM Algorithm and the Poor Man's Data Augmentation Algorithms. *Journal of the American Statistical Association* 85(411):699–704.

David Yarowsky and Richard Wicentowski. 2000. Minimally Supervised Morphological Analysis by Multimodal Alignment. In *ACL '00*. pages 207–216.