Allomorfessor: Towards Unsupervised Morpheme Analysis

Oskar Kohonen, Sami Virpioja, and Mikaela Klami

Adaptive Informatics Research Centre,
Helsinki University of Technology
{oskar.kohonen,sami.virpioja,mikaela.klami}@tkk.fi

Abstract. We extend the unsupervised morpheme segmentation method Morfessor Baseline to account for the linguistic phenomenon of allomorphy, where one morpheme has several different surface forms. Our method discovers common base forms for allomorphs from an unannotated corpus. We evaluate the method by participating in the Morpho Challenge 2008 competition 1, where inferred analyses are compared against a linguistic gold standard. While our competition entry achieves high precision, but low recall, and therefore low F-measure scores, we show that a small model change gives state-of-the-art results.

1 Introduction

Morphological analysis is crucial to many modern natural language processing applications, especially when dealing with morphologically rich languages where the enormous number of inflected word forms lead to severe problems with data sparsity and computational efficiency. There are several successful methods for unsupervised segmentation of word forms into smaller, morpheme-like units [2, 3]. The phenomenon of allomorphy limits the quality of morpheme analysis achievable by segmentation alone. Allomorphy is defined in linguistics as when an underlying morpheme-level unit has two or more morph-level surface realizations which only occur in a complementary distribution: only one of the different allomorphs of a given morpheme appear may appear in a certain morpho- and phonotactical context. For example, in Finnish, the singular genitive case is marked with a suffix `n`, e.g. `auto` (car) – `auton` (car's). Many Finnish nouns undergo a stem change when producing the genitive: `kenkä` (shoe) – `kengän` (shoe's), `pappi` (priest) – `papin` (priest's), `tapa` (habit) – `tavan` (habit's). A segmentation based approach models changed stems as distinct morphemes.

There are two main tasks in literature on learning allomorphy: finding morphologically related words (e.g. [9, 1]), and learning a morphological analyzer (e.g. [10, 4]). In our contribution to Morpho Challenge 2008 [7], we present an analyzer that is similar to Morfessor Baseline [3], but in addition finds common base forms for the inflected forms that derive from the same root word. We currently ignore allomorphic variation in suffixes. Information sources used in literature are orthographic similarity, word frequencies [10] and similar word contexts [9, 1]. We currently use only orthographic features. They are used in a similar manner in [10], but our model needs less supervision and allows concatenative morphology, rather than only stem-suffix pairs. Maybe the closest work to ours is presented in [4]. They study more general orthographic rewrite rules than we do, but the algorithm includes several phases and many heuristics. They also allow concatenative morphology, but the approach is not as general and cannot find, e.g., suffixes between stems. By embedding allomorphy learning into the Morfessor framework, we keep the algorithm flexible and conceptually simple.

2 Allomorfessor Model

In this section, we describe *Allomorfessor*, a morphological model that takes allomorphic variation into account. We start by defining a probabilistic generative model $\mathcal{M}$ for a text corpus. With *Maximum a Posteriori* (MAP) estimation, we try to find the model that is the most probable given the training corpus:

$$\mathcal{M}_{\text{MAP}} = \arg\max_{\mathcal{M}} P(\mathcal{M}|\text{corpus}) = \arg\max_{\mathcal{M}} P(\mathcal{M})P(\text{corpus}|\mathcal{M}) \tag{1}$$

$P(\mathcal{M})$ is the Bayesian prior probability for the model and $P(\text{corpus}|\mathcal{M})$ is the likelihood of the training corpus. Compared to Maximum Likelihood estimation, MAP provides a systematic way of balancing the model complexity and accuracy, and thus helps with the problem of overlearning (see, e.g., Ch. 3 in [5]).

Modeling a corpus with a morphological model is not straightforward. As occurrences of words in a corpus follow power law distributions (Zipf's law), any realistic model should abide by that phenomenon. Instead of using an explicit model for the corpus, as in, e.g., [6], we separate word-level and morpheme-level models, and concentrate only on the latter. We set the word-level model $\mathcal{M}_W$ to be a constant given a word lexicon $\mathcal{L}_W$, which contains all the word forms in the corpus, and try to find only the morpheme-level model $\mathcal{M}_M$. In addition, we divide $\mathcal{M}_M$ into two parts: morpheme lexicon $\mathcal{L}_M$ and morpheme grammar $\mathcal{G}_M$. The former models word-internal syntax and the latter provides the morphemes that from which the words are constructed. The optimization task is thus:

$$\mathcal{M}_{\text{MAP}} = \arg\max_{\mathcal{G}_M, \mathcal{L}_M} P(\mathcal{L}_W|\mathcal{G}_M, \mathcal{L}_M)P(\mathcal{G}_M)P(\mathcal{L}_M). \tag{2}$$

This is equivalent to the approach used in Morfessor [3], but instead of modeling the original corpus, we are now modeling a lexicon of the words in the corpus.[1]

Our morpheme-level model resembles Morfessor Baseline, but it has a hierarchical structure, whereas Morfessor Baseline models words as sequences of morphs. At its core, our model is a probabilistic context-free grammar. Terminals of the grammar are strings resembling linguistic morphemes, specifically root stems and affixes. Non-terminals μ are morphs or their combinations. There

[1] This has been recommended to be done also with Morfessor by setting all the word counts to one. Otherwise, frequent word forms are often undersegmented.

are only two kinds of rules: μ is either replaced by a terminal (string), or two non-terminals, a *prefix* and a *suffix morph*, with a *mutation* terminal. In the former case, μ is a single real morph (root stem or affix). In the latter case, it is a *virtual morph*, which has substructure. Prefix and suffix morphs of a virtual morph can be either real or virtual morphs. The mutation is a special kind of terminal which modifies the virtual prefix. The mutation may be *empty*, which corresponds to a regular inflection or compound word, where the previous morph does not undergo any changes. For an illustration, see Fig. 1.

When designing the mutation model for allomorphy we strive to: (1) Make wrong analyses costly by favoring mutations close to the suffix. E.g., the edit distance between `blue` and `glue` is only one, but they are not allomorphs of the same morpheme. (2) Use mutation types general enough to allow statistical analysis. I.e., similar variations in different words should be modeled with the same mutation. The mutation type used in Allomorfessor is a special case of the standard edit distance (see, e.g., [8]). We allow only substitution and deletion operations, and make the mutation position independent. The affected position is found by matching to k:th instance of a target letter, that is scanned for starting from the end of the virtual prefix (or previous operation). Examples are shown in Table 1.

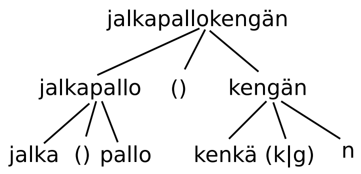

Fig. 1. An example analysis of the Finnish word `jalkapallokengän` (football shoe's). First, the word is split in two with an empty mutation denoted as `()`, then the virtual prefix `jalkapallo` is further split into the stems `jalka` and `pallo`. The virtual suffix `kengän` is split into the stem `kenkä`, the mutation `(k|g)` which transforms it into `kengä`, and the suffix `n`.

To calculate the smallest mutation of this kind between two arbitrary strings we apply the dynamic programming based algorithm for minimum edit distance (see, e.g., [8]), which can be modified to return also the edit operations needed. We want the optimal path not containing insertions, so we set the cost of insertions to be larger than what the other operations may yield for the given string lengths. In this way we always find alternative paths without insertions if possible, by discarding candidates with too high costs. It is trivial to transform the edit operations into the Allomorfessor mutation format.

2.1 Model Probabilities

Next we give a formal description of the probabilities of Equation 2 for the Allomorfessor model. The formulation follows the work by Creutz and Lagus [3], with a few changes. First, every word form in the word lexicon is represented by one real or virtual morph μ_j. Thus the likelihood of the word lexicon is simply

$$P(\mathcal{L}_W|\mathcal{G}_M, \mathcal{L}_M) = \prod_{j=1}^{M_W} P(\mu_j), \qquad (3)$$

Table 1. The allowed operations in mutations and some examples in Finnish.

Operation	*Notation*	*Description*
substitution	k`x|y`	Change k:th `x` to `y`
deletion	`-`k`x`	Remove k:th `x`

(k is omitted when $k = 1$)

Source	*Mutation*	*Target*
`kenkä` (shoe)	`(k|g)`	`kengä` (e.g. `kengä+ssä`, in shoe)
`tanko` (pole)	`(k|g)`	`tango` (e.g. `tango+t`, poles)
`ranta` (shore)	`(-a t|n)`	`rann` (e.g. `rann+oi+lla`, on shores)
`ihminen` (human)	`(2n|s)`	`ihmisen` (human's)

where M_W is the number of words in the lexicon. The probability of the morph μ is estimated from the number of references to it from the word lexicon and virtual morphs.

The morph lexicon $\mathcal{L}_M$ consists of the real and virtual morphs. The probability of the morph lexicon is based on the properties of the morphs:

$$P(\mathcal{L}_M) = P(\text{size}(\mathcal{L}_M) = M)P(\text{properties}(\mu_1)\ldots\text{properties}(\mu_M))M! \qquad (4)$$

If a non-informative prior is used for the probability of the lexicon size M, its effect is minimal and it can be neglected. The factor $M!$ is explained by the fact that there are $M!$ possible orderings of M items, and the lexicon is the same regardless of the order in which the morphs are discovered.

The properties of the morphs are divided into two parts, usage and form. The usage includes properties of the morph itself and the properties of its context. In this model, we use only morph frequencies. For the probability of the frequency distribution, we use a non-informative, implicit frequency prior

$$P(\text{usage}(\mu_1)\ldots\text{usage}(\mu_M)) = P(\text{freq}(\mu_1)\ldots\text{freq}(\mu_M)) = 1/\binom{N-1}{M-1}, \qquad (5)$$

where N is the sum of the counts of the morphs.

The form of a morph is its representation in the model. Forms of the morphs are assumed to be independent. As described before, μ_i is either a real morph represented by a string of letters, or a virtual morph consisting of prefix (μ_{pre}) and suffix (μ_{suf}) morphs and a (possibly empty) mutation δ_k. The probabilities are defined as:

$$P(\text{form}(\mu_i)) = \begin{cases} P(\text{sub})P(\mu_{\text{pre}})P(\delta_k)P(\mu_{\text{suf}}), & \text{if } \mu_i \text{ is virtual;} \\ \left[1 - P(\text{sub})\right]P(\text{len}(\mu_i))\prod_{j=1}^{\text{len}(\mu_i)} P(\hat{c}_{ij}), & \text{otherwise.} \end{cases} \qquad (6)$$

$P(\text{sub})$ is the probability that a morph has substructure, and for real morphs, $\hat{c}_{ij}$ is the jth character of the morph. The lengths of the real morphs are modeled explicitly using a gamma distribution with shape a and scale b:

$$P(\text{len}(\mu_i)) = \frac{1}{\Gamma(a)b^a}\text{len}(\mu_i)^{a-1}e^{-\text{len}(\mu_i)/b}. \qquad (7)$$

Grammar $\mathcal{G}_M$ of the model contains the set of mutations Δ. Similarly to the lexicons,

$$P(\mathcal{G}_M) = P(\text{size}(\Delta) = M_\delta) P(\text{properties}(\delta_1) \ldots \text{properties}(\delta_{M_\delta})) M_\delta!, \tag{8}$$

and properties can be divided into usage and form. Usage features include only the frequencies; the non-informative prior is applied (cf. Equation 5). The prior probability for the form of a mutation δ_i with $\text{len}(\delta_i)$ operations is given by:

$$P(\text{form}(\delta_i)) = P(\text{len}(\delta_i)) \prod_{j=1}^{\text{len}(\delta_i)} P(k_{ij}) P(\text{op}_{ij}) \tag{9}$$

$$P(\text{op}_{ij}) = \begin{cases} P(\text{del}) \frac{1}{\Sigma} & \text{if op}_{ij} \text{ is a deletion} \\ P(\text{sub}) \frac{1}{\Sigma^2} & \text{if op}_{ij} \text{ is a substitution} \end{cases} \tag{10}$$

For the weights we use $P(\text{del}) = P(\text{sub}) = 0.5$, Σ is the alphabet size, and k_{ij} tells which instance of the target letter of the operation op_{ij} is matched. $P(\text{len}(\delta_i))$ and $P(k_{ij})$ are taken from Gamma distributions.

2.2 Learning the Model

The model is learned by iteratively improving the model posterior $P(\mathcal{M}|\text{corpus})$, processing one word at a time and selecting the analysis of that word that maximizes the probability, as shown in Algorithm 1. Note that A_w is a list and we use $+$ to denote the append operation. The algorithm considers analyzing the word w (1) without splits (2) with all possible splits of w and an empty mutation (3) with all possible splits and a base form similar to the virtual prefix and the required mutation. The two former ones are the same as in Morfessor Baseline and the third is our extension, with details shown in Algorithm 2.

Since each word has $2^{(\text{len}(w)-1)}$ possible analyses without considering mutations, we search greedily for the best split at any time, reducing the search space to $O(\text{len}(w)^2)$. When considering mutations, any word w could potentially be the base form for any other word w^*. This would lead naturally to a $O(N^2)$ algorithm. This is unfeasible for large datasets, and therefore we constrain the candidates in heuristic ways, such as limiting the number of analyses to K per morph and iteration, as can be seen in Algorithm 2. Since finding the $baseforms$ can be done as a range search it requires $O(K \log(N))$ time, and thus the time complexity for the whole learning algorithm is $O(NK \log(N))$.

3 Experiments

The model was evaluated in Morpho Challenge 2008 competition 1 [7]. The following parameter settings are used: Morph lexicon length distribution in Equation 7: shape $a = 5$ and scale $b = 1$. The number of candidates considered for each virtual morph $K = 20$. For the mutation lengths and k_{ij} in Equation 9, we used parameters $a = 1$ and $b = 1$ of the gamma prior to prefer short mutations.

Algorithm 1 The learning algorithm

```
while $P(\mathcal{M} \mid \text{corpus})$ increases do
  for $w \in \mathcal{L}_W$ in random order do optimize($w$,len($w$))
end while
function optimize($w$,$n$)
  $A_w \leftarrow \big[w\big] + \big[(w_{1..i}, w_{(i+1)..n}) : i \in 1, ..., n-1\big] + \text{mutated\_analyses}(w, n)$
  Apply the analysis $a^*_w$ of the first $K$ elements of $A_w$ that maximizes $P(\mathcal{M} \mid \text{corpus})$
  if $a^*_w$ involved a split then optimize($w_{1..i}, i$); optimize($w_{(i+1)..n}, n-i$)
```

Algorithm 2 mutated_analyses(w, n)

```
for $i \in 1, ..., n-1$ do
  if $n >= 4 \wedge \text{len}(w_{(i+1)..n}) <= 5 \wedge w_{(i+1)..n} \in \mathcal{L}_M$ then
    if $n > 6$ then $difflen \leftarrow 4$ else $difflen \leftarrow 3$
    $baseforms \leftarrow \{v \in \mathcal{L}_W : v_{1..(n-difflen)} = w_{1..(n-difflen)}\}$
    Calculate mutations $\delta_j$ between each $baseforms_j$ and $w_{(i+1)..n}$
    $A_w \leftarrow A_w + \big[(v_j, w_{(i+1)..n}, \delta_j) : v_j \in baseforms\big]$
  end if
end for
return $A_w$ sorted by $i$ and descending len($v_j$)
```

The Morpho Challenge results are summarized in Table 2. The most striking figures are our very low recall numbers. Low recall means that the model undersegments heavily, i.e., the algorithm should find more morphemes per word (e.g. `kengän` and `papin` are both unsegmented). The precisions are quite good, especially for Turkish and Finnish, but are explained by the low recall.

Mutations were not used very frequently in the analyses. Where substructure was found in the word form, 98% of the mutations were empty for English and 96% for Finnish. The algorithm often favors using new base forms over using mutations, e.g. `prettier` is analysed as `pretti () er`, not `pretty -y er`. The five most common mutations for English and Finnish are shown in Table 3. Some of the example analyses shown are desired (e.g., `-e` in `abjure`, `-a` in `haljeta`), but in many cases the mutation is clearly unnecessary. E.g., a simpler analysis for `suspicions` would be `suspicion () s`. Mutations are also used commonly in misspelled words. E.g., both `contructed` and `contructive` exist in the English corpus, and mutation `-d-e` is used to get the missing base form `contruct`.

Table 2. Results from the Morpho Challenge evaluation. See [7] for details.

Language	*Precision*	*Recall*	*F-Measure*	*F/Winner*	*F/Morf.Baseline*
English	83.39%	13.43%	23.13%	56.26%	54.04%
German	87.92%	7.44%	13.71%	54.06%	31.01%
Turkish	93.25%	6.15%	11.53%	51.99%	20.08%
Finnish	92.55%	6.89%	12.82%	48.47%	21.16%

Table 3. The five most frequent mutations found by the algorithm for English (left side) and Finnish (right side).

Mutation	*Freq.*	*Example*	*Mutation*	*Freq.*	*Example*
`(-e)`	2033	abjure (-e) ed	`(-n)`	27510	antiikin (-n) lle
`(-s)`	537	actress (-s) s'	`(-n -e)`	15830	edustajien (-n-e) esi
`(-y)`	386	inequity (-y) able	`(-a)`	6241	haljeta (-a) essa
`(-n)`	243	suspicion (-n) ns	`(-i)`	4203	kliimaksi (-i) in
`(-d -e)`	183	contructed (-d-e) ive	`(-a -t)`	2792	alokkaita (-a-t) lle

4 Discussion

Our model gave poor results in Morpho Challenge even compared to Morfessor Baseline (see Table 2). Afterwards, we have found out the reasons for the undersegmentation and implemented a new version that solves the problems. The main reasons for the undersegmentation are hierarchical model structure and context independent mutations, which both result in increased cost of data. Compare, e.g., the following analyses of English word "mispronouncing":

$$\begin{aligned} P(\text{mispronouncing}|\mathcal{M}) &= P(\text{mis})P(\epsilon)P(\text{pronouncing}) \\ P(\text{pronouncing}|\mathcal{M}) &= P(\text{pronounce})P(\text{-e})P(\text{ing}) \\ &\text{vs.} \\ P(\text{mispronouncing}|\mathcal{M}) &= P(\text{mis})P(\text{pronounc})P(\text{ing}), \end{aligned}$$

where ϵ is an empty mutation. The former analysis may save one lexical item due to the use of the mutation `-e`, but the data cost will have six probabilities compared to three in the latter analysis. If "mispronouncing" were analyzed as a single morph, the data probabilities of the two model would be equal. Thus complex segmentations are penalized more in our model.

The problem of mutation costs can be solved by conditioning the mutations by the following morph (suffix). In the previous example, we can get:

$$\begin{aligned} P(\text{mispronouncing}|\mathcal{M}) = P(\text{mis})P(\epsilon|\text{mis}) \times P(\text{pronounce})P(\epsilon|\text{pronounce}) \times \\ P(\text{ing})P(\text{-e}|\text{ing}). \end{aligned}$$

Note that most of the morphs are stems, and occur only with the empty mutation. Then $P(\epsilon \,|\, \mu) = 1$, and the data cost does not increase. Using a flat structure and conditioning the mutation probabilities on suffixes do not require complicated changes to the Allomorfessor model. The most relevant change is that the word lexicon is represented by a sequence of morphs and mutations:

$$P(\mathcal{L}_W|\mathcal{G}_M, \mathcal{L}_M) = \prod_{j=1}^{M_W} \prod_{k=1}^{n_j} P(\mu_{jk})P(\delta_{jk}|\mu_{jk}), \tag{11}$$

where n_j is the number of morphs in word j. As morphs will not have any substructure, $P(\text{sub})$ is zero in Equation 6. Probabilities $P(\delta \,|\, \mu)$ are estimated from the observed frequencies. Non-informative prior probabilities of the co-occurrences of mutations and morphs are added to the usage properties of the morphs.

We have made preliminary test with the new version using the English task of competition 1. To speed up the computation and reduce misspellings, the word forms that occurred only once were excluded from training. With this setting, F-measure was 57.12% (precision 65.26%, recall 50.79%). This is significant improvement over the submitted version, and shows that the general framework is working. Part of the improvement in recall and F-measure was due to the pruned training data: with the same data, F-measure for Morfessor Baseline was 56.14% (precision 64.49%, recall 49.69%). However, the improvement over Morfessor Baseline was statistically significant for both precision and recall.

We conclude that our framework for learning allomorphy seems to be promising. In addition to more extensive testing and error analysis with the new version, future work will include using context and frequency information to both limit and weight the potential allomorphs.

References

1. Marco Baroni, Johannes Matiasek, and Harald Trost. Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. In *Proceedings of the ACL-02 workshop on Morphological and phonological learning*, pages 48–57, Morristown, NJ, USA, 2002. ACL.
2. Delphine Bernhard. Simple morpheme labelling in unsupervised morpheme analysis. In *Advances in Multilingual and Multimodal Information Retrieval, 8th Workshop of the CLEF*, volume 5152 of *Lecture Notes in Computer Science*, 2008.
3. Mathias Creutz and Krista Lagus. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing*, 4(1), January 2007.
4. Sajib Dasgupta and Vincent Ng. High-performance, language-independent morphological segmentation. In *In the annual conference of the North American Chapter of the ACL (NAACL-HLT)*, 2007.
5. Carl G. de Marcken. *Unsupervised Language Acquisition*. PhD thesis, MIT, 1996.
6. Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. Interpolating between types and tokens by estimating power-law generators. In *Advances in Neural Information Processing Systems (NIPS)*, page 18, 2006.
7. Mikko Kurimo, Ville Turunen, and Matti Varjokallio. Overview of Morpho Challenge 2008. In *Evaluating Systems for Multilingual and Multimodal Information Access – 9th Workshop of the CLEF*, Lecture Notes in Computer Science, Aarhus, Denmark, September 2008 (printed in 2009).
8. Gonzalo Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, 2001.
9. Patrick Schone and Daniel Jurafsky. Knowledge-free induction of morphology using latent semantic analysis. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning*, pages 67–72, Morristown, NJ, USA, 2000. ACL.
10. David Yarowsky and Richard Wicentowski. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of the 38th Meeting of the ACL*, pages 207–216, 2000.