Unsupervised Morpheme Discovery with Allomorfessor

Sami Virpioja and Oskar Kohonen
Adaptive Informatics Research Centre, Helsinki University of Technology
{sami.virpioja,oskar.kohonen}@tkk.fi

Abstract

We describe Allomorfessor, which extends the unsupervised morpheme segmentation method Morfessor to account for the linguistic phenomenon of allomorphy, where one morpheme has several different surface forms. The method discovers common base forms for allomorphs from an unannotated corpus by finding small modifications, called mutations, for them. Using Maximum a Posteriori estimation, the model is able to decide the amount and types of the mutations needed for the particular language. The method is evaluated in Morpho Challenge 2009.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: I.2.6 Learning; I.2.7 Natural Language Processing

General Terms

Algorithms, Experimentation, Languages

Keywords

Morphology, Morphological Analysis, Unsupervised Learning

1 Introduction

Morphological analysis is crucial to many modern natural language processing applications, especially when dealing with morphologically rich languages. The enormous number of inflected word forms may 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 (see, e.g., [7, 4, 2]). However, 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.

In Morpho Challenge 2008, we introduced an unsupervised model for morpheme segmentation and allomorphy learning [10]. In [11], some modifications to the model (now referred to as

Allomorfessor Alpha) were suggested. In this paper we describe and evaluate the modified Allomorfessor model (referred to as Allomorfessor Baseline). As indicated by the name, the model is an extension to the Morfessor Baseline model by Creutz and Lagus [3].

There are two main problems in literature on the unsupervised learning of allomorphy: finding morphologically related words (e.g. [13, 1]), and learning a morphological analyzer (e.g. [14, 5]). We try to solve the latter, which is more complex and general—as morphologically related words can be determined from the analyses. In contrast to the work by Yarowsky and Wicentowski [14], the framework based on Morfessor allows concatenative morphology, rather than only stem-suffix pairs. In the work by Dasgupta and Ng [5], concatenative morphology is allowed to some extent, but the approach is not as general and cannot find, e.g., suffixes between stems. Another difference is related to what information sources are used for finding the allomorphs. In addition to the orthographic similarity, word frequencies [14] and word contexts [13, 1] have been applied. We currently use only orthographic similarity.

This paper proceeds as follows: Section 2 presents the framework of the model and the learning task, both based on Morfessor. Section 3 describes how Allomorfessor models allomorphy by including new operations, mutations, to the model. Section 4 defines the model probabilities needed by the *Maximum a Posteriori* estimation. Section 5 describes the applied learning algorithm for the model, and Section 6 how the model can be used to analyze new words. Section 7 includes the initial results for the Morpho Challenge 2009. Finally, Section 8 concludes the paper.

2 Model Framework

To define our framework for learning morphology, we start with a probabilistic generative model $\mathcal{M}$ for a text corpus. With *Maximum a Posteriori* (MAP) estimation, we try to select 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., Chapter 3 in [6]). This MAP formulation can alternatively be formulated using a two-part coding approach of the Minimum Description Length (MDL) principle.

Modeling a corpus with a morphological model is not straightforward. For example, the occurrences of the 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., [8], we separate word-level model $\mathcal{M}_W$ and morpheme-level model $\mathcal{M}_M$, and estimate only the latter. Word-level model is assumed to be a constant given a word lexicon $\mathcal{L}_W$, which contains all the word forms in the corpus. 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 [4], but instead of modeling the original corpus, we are now modeling a lexicon of the words in the corpus.[1]

3 Modeling Allomorphy with Mutations

Our morpheme-level model is Morfessor Baseline extended with operations that can make minor modifications to the surface forms of the morphemes. These operations are called *mutations*.

[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.

In many cases, the mutations are *empty*, i.e., they do not affect the surface form. If all of the mutations are empty, the model is equivalent to Morfessor Baseline.

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. 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.

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., [12]), which can be modified to return also the edit operations needed. We want the optimal sequence of operations 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 can always find sequences of operations not containing insertions, if such sequences exist, by discarding candidates with too high costs. It is trivial to transform the edit operations into the Allomorfessor mutation format.

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)

To verify the suitability of the approach, we examined how well this kind of mutations are able to find the allomorphic variations in linguistic gold standard segmentations. The tests were performed on English, Finnish and Turkish, based on the gold standards used in Morpho Challenge.[2] Statistics were calculated separately for a word lexicon and for a corpus, where the common words had more weight. The results are in Table 2. The first column shows the number of morphs in the data. The second column shows how many of the morphs have allomorphs. The third column shows how many of the allomorphs can be constructed with mutations. We applied similar restrictions to those that were in our model; in practice, variations in affixes and other short morphemes were excluded from the search. Mutations provide reasonable good coverage for English and Finnish. E.g., for English, we can find at most 82% of the real allomorphs in the gold standard segmentation. The percentages for the corpora are lower, as affixes are more common than stems. For Turkish, where most of the allomorphy seems to be in affixes or other short morphemes, only 2% of the cases with allomorphic variants in a corpus can be found using mutations.

4 Model Probabilities

Next, we give a formal description of the probabilities in Equation 2 for the Allomorfessor Baseline model. Again, the formulation follows the work by Creutz and Lagus [4], especially the Morfessor Baseline model.

[2]See `http://www.cis.hut.fi/morphochallenge2009/datasets.shtml`.

Table 2: The portion of morphemes with allomorphs and how many of the allomorphic variations can be modeled with mutations for English, Finnish and Turkish lexicon and corpus.

	Morphemes	*Allomorphs*		*Mutation found*	
English lexicon	21 173	10 858	(51.3%)	8 912	(82.1%)
Finnish lexicon	68 743	56 653	(82.4%)	36 210	(63.9%)
Turkish lexicon	23 376	646	(2.8%)	102	(15.8%)
English corpus	76 968 382	42 282 837	(54.9%)	14 706 543	(34.8%)
Finnish corpus	73 512 023	61 583 251	(83.8%)	18 751 022	(30.5%)
Turkish corpus	23 288 821	11 978 142	(51.4%)	225 708	(1.9%)

First, every word form w_j in the word lexicon is represented by a sequence of morphs μ_{jk} and mutations δ_{jk}

$$P(\mathcal{L}_W|\mathcal{G}_M, \mathcal{L}_M) = \prod_{j=1}^{M_W} \prod_{k=1}^{n_j} P(\mu_{jk}) P(\delta_{jk}|\mu_{jk}), \tag{3}$$

where n_j is the number of morphs in word j. The probabilities of the morphs and the (conditional) probabilities of the mutations are estimated from the observed frequencies. I.e., if there is 10000 morph tokens in the word lexicon, and μ_{jk} occurs 200 times, its probablity will be $P(\mu_{jk}) = 200/10000 = 0.02$.

The probability of the morph lexicon $\mathcal{L}_M$ is based on the properties of the morphs:

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

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. Here we include only the frequency distribution of the morphs. For the probability of the distribution, we use a non-informative, implicit frequency prior

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

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. They are represented by a string of characters c_{ij}:

$$P(\text{form}(\mu_i)) = P(\text{len}(\mu_i)) \prod_{j=1}^{\text{len}(\mu_i)} P(c_{ij}), \tag{6}$$

where c_{ij} is the jth character of the morph. The lengths of the morphs are modeled explicitly using an appropriate probability distribution, such as an exponential (geometric) or a gamma distribution.

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

$$P(\mathcal{G}_M) = P(\text{size}(\Delta) = M_\delta) P(\text{properties}(\delta_1) \ldots \text{properties}(\delta_{M_\delta})) M_\delta!, \tag{7}$$

and properties can be divided into usage and form. Usage features include the frequencies of the mutations and their co-occurrences with the suffix morphs (needed in Equation 3). We apply a condition that each morph has to have at least one co-occurrence with an empty mutation ϵ. In consequence, the count of the empty mutation n_ϵ is at most N (number of morph tokens) and at least M (number of morph types). Applying the the uniform distribution,

$$P(\text{freq}(\epsilon)) = \frac{1}{N - M + 1}. \tag{8}$$

The other $M_\delta - 1$ mutation types have $N - n_\epsilon$ occurrences in total, as there are as many mutation tokens as there are morph tokens in the data. We apply the same non-informative prior as in Equation 5. Finally, we determine the probability of the co-occurrences of mutations and suffix morphs. For each non-empty mutation δ we divide its occurrences with the M possible morphs. There are $\binom{\text{freq}(\delta)+M-1}{M-1}$ possibilities, so a non-informative prior for the co-occurrences is

$$P(\text{co-freqs}(\Delta, \mathcal{L}_M)) = \prod_{\delta \in \Delta \setminus \epsilon} 1/\binom{\text{freq}(\delta) + M - 1}{M - 1}. \tag{9}$$

Note that after the others are determined, the co-occurrences with the empty mutation are:

$$\text{co-freq}(\mu, \epsilon) = \text{freq}(\mu) - \sum_{\delta \in \Delta \setminus \epsilon} \text{co-freq}(\mu, \delta). \tag{10}$$

The prior probability for the form of a mutation δ_i with $\text{len}(\delta_i)$ operations is given by:

$$P(\text{form}(\delta_i)) = P(\text{len}(\delta_i)) \prod_{j=1}^{\text{len}(\delta_i)} P(k_{ij}) P(\text{op}_{ij}) \tag{11}$$

$$P(\text{op}_{ij}) = \begin{cases} P(\text{del}) \frac{1}{\Sigma} & \text{if op}_{ij} \text{ is a deletion} \\ P(\text{sub}) \frac{1}{\Sigma^2} & \text{if op}_{ij} \text{ is a substitution} \end{cases} \tag{12}$$

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 $P(k_{ij})$ can be taken from any suitable prior distribution.

5 Algorithm for Model Learning

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. In the algorithm, 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, and (3) with all possible splits and a base form similar to the virtual prefix and the required mutation. The cases (1) and (2) are the same as in Morfessor Baseline and (3) 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^*. Thus, a naive algorithm would have time complexity $O(N^2)$, which is unfeasible for large datasets. 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 $O(NK \log(N))$.

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($w$,$n$)
  $A_w \leftarrow [w] + [(w_{1..i}, w_{(i+1)..n}) : i \in 1, ..., n-1] + \text{mutated\_analyses}(w, n)$
  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 $\textit{difflen} \leftarrow 4$ else $\textit{difflen} \leftarrow 3$
    $\textit{baseforms} \leftarrow \{v \in \mathcal{L}_W : v_{1..(n-\textit{difflen})} = w_{1..(n-\textit{difflen})}\}$
    Calculate mutations $\delta_j$ between each $\textit{baseforms}_j$ and $w_{(i+1)..n}$
    $A_w \leftarrow A_w + \big[(v_j, w_{(i+1)..n}, \delta_j) : v_j \in \textit{baseforms}\big]$
  end if
end for
return $A_w$ sorted by $i$ and descending $\text{len}(v_j)$
```

6 Algorithm for Analyzing New Data

After the model has been trained, it can be used to analyze words with a variant of the Viterbi algorithm, which is a dynamic programming algorithm that finds the most probable state sequences for Hidden Markov models [9]. In our case, the observation is the sequence of $|W|$ letters that form the word w, and the hidden states are the morphemes of the word. We need a grid s of length $|W|$ to fill with the best probability values $\alpha(s_i)$ and paths. Without mutations, the model is 0th order Markov model, and the grid is a one dimensional table. The grid position s_i indicates that the first i letters are observed. At each time step, we proceed with one letter and insert the probability $\alpha(s_i) = \max_j \alpha(s_j)P(\mu_{ji})$ and path indicator $\psi(s_i) = \arg\max_j \alpha(s_j)P(\mu_{ji})$ to the grid. We can come to s_i from any of the positions s_j between s_1 and s_{i-1}: the letters between j and i form the next morpheme μ_{ij}. The time complexity is of the algorithm is thus $O(|W|^2)$.

The mutations make things a bit more complicated. As they are conditioned on the suffixes, it is easier to run the algorithm from right to left. The grid has to be two dimensional: for each s_i there can be several states (morphemes) with their own costs and paths. The rule for updating the grid value for s_i is

$$\alpha(s_i, \hat{\mu}_{ij}) = \max_{j \in [i+1, |W|]} \Big\{ \max_{\mu \in s_j} \Big\{ \max_{\delta \in \Delta} \big\{ \alpha(s_j, \mu) P(\delta|\mu) P(\hat{\mu}_{ij}) \big\} \Big\} \Big\}, \tag{13}$$

where $\hat{\mu}_{ij}$ is a morpheme that produces the letters between i and j when modified by the mutation δ. Only those mutations that are observed before μ need to be tested, otherwise $P(\delta|\mu) = 0$. For the morphemes that are not observed before, we use an approximate cost of adding them into the lexicon. The worst case time complexity for the algorithm is $O(MM_\delta|W|^2)$. In practice, however, the number of morphemes and mutations tested in each position is quite limited.

7 Experiments and Evaluation

For Morpho Challenge 2009 Competitions 1 and 2 we trained the model with the Competition 1 data, where all words occurring only once were filtered out[3], with the exception of Arabic data sets, where the number of words was very low to start with. After training the model, we analyzed all the words in both data sets with the Viterbi algorithm (Section 6). For Competition 3, we used the Europarl data set for training, without any filtering. After training the model, the final analysis was calculated with the Viterbi algorithm. The following parameter settings were used: Morpheme length distribution in Equation 6 was geometric with parameter $p = M_W/(M_W + M_c)$, where M_W is the number of words and M_c the number of characters in the training corpus. The number of candidates considered for each virtual morph was $K = 20$. For the mutation lengths and k_{ij} in Equation 11, we used gamma distribution with scale and shape parameters equal to one, preferring short mutations.

In Table 3, the performance of the Allomorfessor Baseline (the current algorithm) is compared to Allomorfessor Alpha (the algorithm presented in Challenge 2008 [10]) and Morfessor Baseline

[3]The are plenty of "rubbish", such as misspelled words and foreign names, in the least frequent words.

Table 3: Results from the Morpho Challenge linguistic evaluation (Competition 1) for Arabic (nv = non-vowelized, vow = vowelized), English, Finnish, German and Turkish. Allomorfessor Baseline and Morfessor Baseline are trained with the same data sets. Allomorfessor Alpha has no implementation for Viterbi segmentation, so it is trained on the full data sets.

Language	*Measure*	*Allomorfessor Alpha*	*Allomorfessor Baseline*	*Morfessor Baseline*
Arabic (nv)	precision	-	91.62%	91.77%
	recall	-	6.59%	6.44%
	F-measure	-	**12.30%**	12.03%
Arabic (vow)	precision	-	88.28%	86.87%
	recall	-	4.37%	4.90%
	F-measure	-	8.33%	**9.28%**
English	precision	83.31%	68.98%	68.43%
	recall	15.84%	56.82%	56.19%
	F-measure	26.61%	**62.31%**	61.71%
Finnish	precision	92.64%	86.51%	86.07%
	recall	8.65%	19.96%	20.33%
	F-measure	15.83%	32.44%	**32.88%**
German	precision	87.82%	77.78%	76.47%
	recall	8.54%	28.83%	30.49%
	F-measure	15.57%	42.07%	**43.60%**
Turkish	precision	93.16%	85.89%	85.43%
	recall	9.56%	19.53%	20.03%
	F-measure	17.35%	31.82%	**32.45%**

[3] in the Competition 1 of Morpho Challenge. The improvement over the previous algorithm is remarkable. As indicated by the improved recall measures, the algorithm no longer undersegments. This can also be seen in Figure 1, where the average number of morphemes per word form is shown for the three algorithms.

Compared to Morfessor, the results are roughly at the same level. For English, Allomorfessor has both higher recall and higher precision. For all the other tasks, one is higher and the other is lower. Note that whenever the recall is higher, also the F-measure is higher, as improving the lower measure (in this case, recall) has more effect on the geometric mean of the measures. Figure 1 shows that on average, Morfessor always segments word forms to smaller parts. This usually leads to a higher recall. However, for English, Allomorfessor obtains higher recall while segmenting less than Morfessor, which implies that the majority of the common base forms extracted by Allomorfessor are correct. Also, Allomorfessor achieved the winning F-measure for English in Morpho Challenge 2009.

In Competition 2, the algorithms were applied in an information retrieval system for English, Finnish and German. The results for Allomorfessor and Morfessor Baseline, shown in Table 4, are roughly on the same level. Notably, Allomorfessor is better for Finnish and Morfessor for English in contrast to Competition 1; rigorous error analysis would be needed to find an explanation. Overall, Allomorfessor performed reasonably well in this task, being second in English and Finnish and third in German.

The number of non-empty mutations found by the algorithm (in the final analysis of all the word forms) is shown in Table 5. Generally, mutations are not used as much as linguistic analysis would prefer. One reason is that the model seems to favor storing frequent morphs directly, instead of deriving them using mutations. The method finds, e.g., the morph `pretti` instead of deriving it as `pretty (y|i)`. Therefore mutations are mostly used for morphs that occur only in few different word forms. When comparing languages, the most striking figures are in the Arabic sets: If the vowels are excluded (as usual in Arabic script), the model finds no useful mutations. However, when the vowels are in the text, the model finds 70 mutations, more than for any other tested

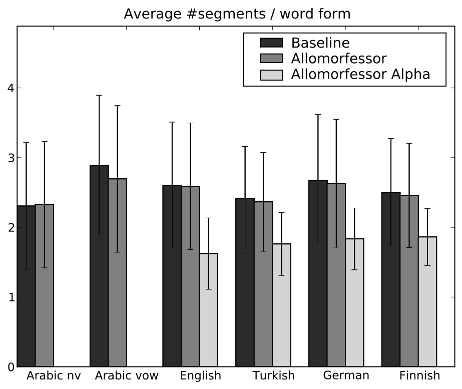

Figure 1: The average amount of morphemes per word indicated by the algorithms. The error bars show the standard deviations.

Table 4: Results from the Morpho Challenge information retrieval evaluation (Competition 2). Allomorfessor Baseline versus Morfessor Baseline, trained with the same data sets.

Language	*Average Precision (%)*	
	Allomorfessor	*Morfessor*
English	0.3852	**0.3873**
Finnish	**0.4601**	0.4475
German	0.4388	**0.4728**

language. This nicely demonstrates the method's ability to adapt to the particular languages and data sets. The fact that Arabic morphology is not concatenative, and thus does not fit well into the Morfessor framework, emphasizes the flexibility of the model.

In Table 6, the mutations found by the algorithm are shown for English and Finnish. As can be seen, a large part of the mutations correspond to linguistic analysis. The most common error, especially for Finnish, is having a derived form as the base form. This is because an unsupervised algorithm has trouble finding the correct base form. However, if the analysed morph is semantically related to the induced base form, such analyses can be useful in applications. Other errors include not finding the correct suffix, using a more complex mutation and suffix combination than necessary, and using a semantically unrelated base form. Mutations are also used commonly on misspelled word forms.

Table 5: The number of non-empty mutations found by Allomorfessor. Mutation usage is the number of non-empty mutation tokens divided by the number of morph tokens.

Language	Arabic (nv)	Arabic (vow)	English	Finnish	German	Turkish
Mutation types	0	69	15	66	26	55
Mutation usage	0.0%	4.61%	0.18%	0.44%	0.17%	0.12%

8 Conclusions

We have described the Allomorfessor Baseline method for unsupervised morphological analysis. It attempts to find the morphemes of the input data by segmenting the words into morphs and finding modifications that can restore allomorphic variations in stems back to their base forms. In the Morpho Challenge 2009 evaluations, significant improvements were obtained over the previous version of the method. The results are now close to those of the Morfessor Baseline method. In comparison to the methods by the other participants, Allomorfessor performed especially well in the linguistic evaluation for English (the best result in the task), and in the information retrieval evaluation for English (second), Finnish (second) and German (third).

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*, pages 873–880. Springer Berlin / Heidelberg, 2008.

[3] Mathias Creutz and Krista Lagus. Unsupervised discovery of morphemes. In *Proceedings of the Workshop on Morphological and Phonological Learning of ACL'02*, pages 21–30, Philadelphia, Pennsylvania, USA, 2002.

[4] Mathias Creutz and Krista Lagus. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing*, 4(1), January 2007.

[5] 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.

[6] Carl G. de Marcken. *Unsupervised Language Acquisition*. PhD thesis, MIT, 1996.

[7] John Goldsmith. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–189, 2001.

[8] 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.

[9] G. David Forney, Jr. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, March 1973.

[10] Oskar Kohonen, Sami Virpioja, and Mikaela Klami. Allomorfessor: Towards unsupervised morpheme analysis. In *Working notes for the CLEF 2008 Workshop*, Aarhus, Denmark, 2008.

[11] Oskar Kohonen, Sami Virpioja, and Mikaela Klami. Allomorfessor: Towards unsupervised morpheme analysis. In *Evaluating Systems for Multilingual and Multimodal Information Access – 9th Workshop of the CLEF*, Lecture Notes in Computer Science. Springer-Verlag, 2009. To appear.

[12] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, 2001.

[13] 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.

[14] 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.

Table 6: Mutation types with example usage for English and Finnish.

Mutation	*Count*	*Examples*	*Notes*
English			
(-e)	1182	adhering: adhere (-e) ing	
(-y)	300	vulnerabilities: vulnerability (-y) ies	
		temporarily: temporary (-y) ily	
(-t)	120	affluence: affluent (-t) ce	
		bankrupcy: bankrupt (-t) cy	word form misspelled
(-a)	66	encyclopedic: encyclopedia (-a) c	
		hemophilic: hemophilia (-a) c	
(-i)	41	publshed: publish (-i) ed	word form misspelled
(-s)	35	euripidean: euripides (-s) a () n	
		diocletian: diocles (-s) tian	
(-o)	27	aspirating: aspiration (-o) g	suffix ing not found
(-n)	27	proletariat: proletarian (-n) t	
		restauration: restaurant (-n) ion	wrong base form
(-c)	20	paraplegia: paraplegic (-c) a	
(t|c)	8	excellencies: excellent (t|c) ies	adjective chosen as base form
		inconveniencies: in () convenient (t|c) ies	adjective chosen as base form
(a|s)	2	ljubljanska: ljubljana (a|s) ka	different form of a proper name
(-g)	1	licensintorg: licensing (-g) torg	proper name oversegmented
(s|n)	1	sclerosing: sclerosis (s|n) g	suffix ing not found
(-h)	1	thorougbred: thorough (-h) bred	word form misspelled
(-a -y)	1	bulathkopitiya: bulathkopitya (-a -y) iya	different form of a proper name
Finnish			
(-n)	7771	ahdingolla: ahdingon (-n) lla	genitive chosen as base form
		aikojemme: aikojen (-n) mme	plural genitive chosen as base form
(-i)	4096	anakronismeille: anakronismi (-i) e () ille	ok, but (i|e) preferable
		desibelejä: desibeli (-i) ejä	ok, but (i|e) preferable
(-a)	2598	diakonissoja: diakonissa (-a) oja	ok, but (a|o) preferable
		eufemismi: eufemia (-a) smi	
(-t)	2507	fagotisti: fagotti (-t) sti	
		haltuunoton: haltuunotto (-t) n	
(-s)	1114	harvennuksen: harvennus (-s) ksen	
		yliherkkyydet: yliherkkyys (-s) det	
(-e)	939	vuosituhantista: vuosituhantiset (-e) a	plural chosen as base form
		viikattein: viikate (-e) tein	
(i|e)	675	videoprojektoreina: video () projektori (i|e) ina	
		transistoreita: transistori (i|e) ita	
(-ä)	532	tulennielijöitä: tulennielijä (-ä) öitä	
		tulokertymien: tulokertymä (-ä) ien	
(a|i)	430	kaavailemia: kaavailema (a|i) a	
		juurevia: juureva (a|i) a	
(n|s)	428	hankkeeseesi: hankkeeseen (n|s) i	base form undersegmented
		diabeteksesi: diabeteksen (n|s) i	base form undersegmented
(a|e)	322	emigranttien: emigranttia (a|e) n	partitive as base form
		hajuharhojen: haju () harhoja (a|e) n	plural partitive as base form
(-k)	311	agnostikoksi: agnostikko (-k) ksi	
		haaksirikossa: haaksirikko (-k) ssa	
(-a -t)	232	murhissa: murhista (-a -t) sa	elative as base form
		varainhankinnalla: varainhankinta (-a -t) na () lla	oversegmented, (t|n) preferable
(-n -i)	183	barrikadeja: barrikadin (-n -i) eja	genitive as base form
		kursseihen: kursseihin (-n -i) en	word form misspelled
(n|i -e)	143	aivotärähdyksiä: aivo () tärähdyksen (n|i -e) ä	genitive as base form
		hoplofoobisia: hoplofoobisen (n|i -e) a	genitive as base form
(-n n|s)	138	aivokurkiaisen: aivokurkiainen (-n n|s) n	
		mustapukuiset: mustapukuinen (-n n|s) t	
(t|d)	97	häädöt: häätö (t|d) t	
		kursivoidun: kursivoitu (t|d) n	
(a|s -t)	83	amppeleissa: amppeleita (a|s -t) sa	plural partitive as base form
		elintarvikkeissa: elintarvikkeita (a|s -t) sa	plural partitive as base form
(ä|t -l)	82	näöltään: näöllä (ä|t -l) ään	adessive as base form
(-e -s)	77	esoteerinen: esoteerisen (-e -s) en	genitive as base form, linguistic
		teksasilainen: teksasilaisen (-e -s) en	"inverse" of (-n n|s)
(t|n)	75	abstrahoinnin: abstrahointi (t|n) n	
(a|t -l)	75	matkapuhelimeltaan: matka () puhelimella (a|t -l) aan	adessive as base form