# Unsupervised Morpheme Analysis with Allomorfessor

Sami Virpioja     Oskar Kohonen

Department of Information and Computer Science
Helsinki University of Technology

30, September 2009

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Information and Computer Science

# Outline

Modeling allomorphy

Allomorfessor

Results and discussion

# Allomorphy

**Definition:** One morpheme-level unit may have two or more morph-level surface realizations which only occur in a complementary distribution

Examples

- `/prettI/ pretty pretti-er`
- `/kenkä/ (shoe) kenkä kengä-n`

# Finding allomorphs

Allomorphs often very similar to a hypothetical regular form:

- prettier vs *prettyer
- kengän vs *kenkän

# Finding allomorphs

Allomorphs often very similar to a hypothetical regular form:

- prettier vs *prettyer
- kengän vs *kenkän

General string similarity does not work well:

- beat vs peat (Levenshtein distance = 1)

# Finding allomorphs

Allomorphs often very similar to a hypothetical regular form:

- `prettier` vs `*prettyer`
- `kengän` vs `*kenkän`

General string similarity does not work well:

- `beat` vs `peat` (Levenshtein distance = 1)

**Observation:** Most allomorphic variation close to boundary between stem and affix.

# Finding allomorphs

Allomorphs often very similar to a hypothetical regular form:

- prettier vs *prettyer
- kengän vs *kenkän

General string similarity does not work well:

- beat vs peat (Levenshtein distance $= 1$)

**Observation:** Most allomorphic variation close to boundary between stem and affix.

**Solution:** Use a class of *string mutations* to encode the prior information.

## String mutations

Simplest thing that could possibly work:

- `pretti` = `pretty -y +i`

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Information and Computer Science

# String mutations

Simplest thing that could possibly work:

- `pretti` = `pretty -y +i`

Problem: `kenkä` – `kengän` and `tanko` – `tangon`

- `-kä+gä` ≠ `-ko+go`

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Information and Computer Science

# String mutations

Simplest thing that could possibly work:

- `pretti` = `pretty -y +i`

Problem: `kenkä` – `kengän` and `tanko` – `tangon`

- `-kä+gä` ≠ `-ko+go`

Solution:

- Allow only deletions and substitutions
- Begin from the end of the morph. Find the first matching letter, apply operation to that one ⇒ `k|g`
- Efficiently computed via Levenshtein algorithm

# Feasibility check for mutations

How much of the allomorphic variations can be found with the selected class of mutations?

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Information and Computer Science

# Feasibility check for mutations

How much of the allomorphic variations can be found with the selected class of mutations?

- Allomorphic variations in linguistic gold standards:

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

# Allomorfessor Baseline

In short: Morfessor Baseline + mutations.

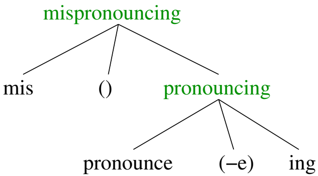
# Allomorfessor Baseline

In short: Morfessor Baseline + mutations. What's new?

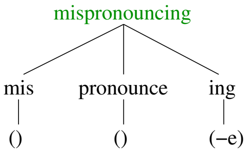
# Allomorfessor Baseline

In short: Morfessor Baseline + mutations. What's new?

- The previous version had a hierarchical model that resulted in undersegmentation.



Allomorfessor Alpha
(Morpho Challenge 2008)

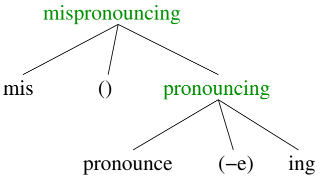


Allomorfessor Baseline
(Morpho Challenge 2009)

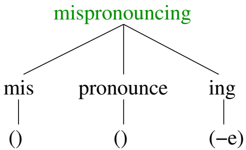
# Allomorfessor Baseline

In short: Morfessor Baseline + mutations. What's new?

- The previous version had a hierarchical model that resulted in undersegmentation.



Allomorfessor Alpha
(Morpho Challenge 2008)



Allomorfessor Baseline
(Morpho Challenge 2009)

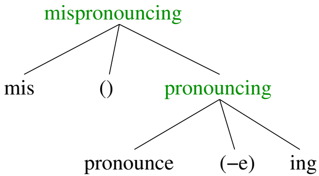
- Now it's really like Morfessor Baseline.

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Information and Computer Science

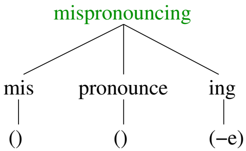
# Allomorfessor Baseline

In short: Morfessor Baseline + mutations. What's new?

- The previous version had a hierarchical model that resulted in undersegmentation.



Allomorfessor Alpha
(Morpho Challenge 2008)



Allomorfessor Baseline
(Morpho Challenge 2009)

- Now it's really like Morfessor Baseline.
- Viterbi-like algorithm for analyzing new words.

# Model definition

Task: Find the model which maximizes the posterior probability:

$$\mathcal{M}_{\mathrm{MAP}} = \operatorname*{arg\,max}_{\mathcal{L}_M, \mathcal{G}_M} P(\mathcal{L}_W | \mathcal{L}_M, \mathcal{G}_M) P(\mathcal{L}_M, \mathcal{G}_M)$$

# Model definition

Task: Find the model which maximizes the posterior probability:

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

- Word lexicon $\mathcal{L}_W$: Input data

# Model definition

Task: Find the model which maximizes the posterior probability:

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

- Word lexicon $\mathcal{L}_W$: Input data
- Morpheme lexicon $\mathcal{L}_M$:
  - A set of morphs with *form* (string of letters) and *usage* (frequency in data) properties

# Model definition

Task: Find the model which maximizes the posterior probability:

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

- Word lexicon $\mathcal{L}_W$: Input data
- Morpheme lexicon $\mathcal{L}_M$:
  - A set of morphs with *form* (string of letters) and *usage* (frequency in data) properties
- Morpheme grammar $\mathcal{G}_M$:
  - A set of mutations with form and usage properties.

# Model definition

Task: Find the model which maximizes the posterior probability:

$$\mathcal{M}_{\mathrm{MAP}} = \underset{\mathcal{L}_M, \mathcal{G}_M}{\arg\max}\, P(\mathcal{L}_W | \mathcal{L}_M, \mathcal{G}_M) P(\mathcal{L}_M, \mathcal{G}_M)$$

- Word lexicon $\mathcal{L}_W$: Input data
- Morpheme lexicon $\mathcal{L}_M$:
  - A set of morphs with *form* (string of letters) and *usage* (frequency in data) properties
- Morpheme grammar $\mathcal{G}_M$:
  - A set of mutations with form and usage properties.
  - Probability conditioned on the subsequent morph.

# Model definition

Task: Find the model which maximizes the posterior probability:

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

- Word lexicon $\mathcal{L}_W$: Input data
- Morpheme lexicon $\mathcal{L}_M$:
  - A set of morphs with *form* (string of letters) and *usage* (frequency in data) properties
- Morpheme grammar $\mathcal{G}_M$:
  - A set of mutations with form and usage properties.
  - Probability conditioned on the subsequent morph.

  → Most morphs (e.g., stems) give non-zero probability only for the empty mutation.

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Information and Computer Science

# Search algorithm

A greedy and local recursive search algorithm similar to Morfessor.

# Search algorithm

A greedy and local recursive search algorithm similar to Morfessor.

- Splitting a morph into prefix and suffix part includes a mutation (usually the empty one).

# Search algorithm

A greedy and local recursive search algorithm similar to Morfessor.

- Splitting a morph into prefix and suffix part includes a mutation (usually the empty one).
- Computational challenge: large amount of possible analyses.

# Search algorithm

A greedy and local recursive search algorithm similar to Morfessor.

- Splitting a morph into prefix and suffix part includes a mutation (usually the empty one).
- Computational challenge: large amount of possible analyses.
- Implemented restrictions:
    - If non-empty mutation, preceding morph has to occur as a word.
    - If non-empty mutation, suffix has to be already in the lexicon.
    - Consider only maximum $K = 20$ analyses per morph.

# Evaluations

Results were good for most tasks and languages:

- C1: winner for English, moderate results for the rest (low recall)
- C2: 2nd and 3rd positions
- C3: winner for both tasks

# Evaluations

Results were good for most tasks and languages:

- C1: winner for English, moderate results for the rest (low recall)
- C2: 2nd and 3rd positions
- C3: winner for both tasks

However, Allomorfessor not better than Morfessor Baseline.

- Sometimes even worse.
- But the differences were not statistically significant.

# Mutations

Almost all mutations included for English and Finnish were correct or at least useful. E.g.,

- `adhering = adhere (-e) ing`
- `publshed = publish (-i) ed`
- `excellencies = excellent (t|c) ies`

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Information and Computer Science

# Mutations

Almost all mutations included for English and Finnish were correct or at least useful. E.g.,

- `adhering = adhere (-e) ing`
- `publshed = publish (-i) ed`
- `excellencies = excellent (t|c) ies`

Mutations not applied as much as linguistic analyses suggest.

# Mutations

Almost all mutations included for English and Finnish were correct or at least useful. E.g.,

- `adhering = adhere (-e) ing`
- `publshed = publish (-i) ed`
- `excellencies = excellent (t|c) ies`

Mutations not applied as much as linguistic analyses suggest.

- Mostly due to a property of Morfessor Baseline: Morphs that occur in many word forms are often undersegmented.

# Conclusions

Allomorfessor Baseline: Morfessor Baseline plus modeling of allomorphy in stems with string mutations.

# Conclusions

Allomorfessor Baseline: Morfessor Baseline plus modeling of allomorphy in stems with string mutations.

- The applied type of mutations can handle significant part of the allomorphic variations in English and Finnish.

# Conclusions

Allomorfessor Baseline: Morfessor Baseline plus modeling of allomorphy in stems with string mutations.

- The applied type of mutations can handle significant part of the allomorphic variations in English and Finnish.

Some unsolved issues:

# Conclusions

Allomorfessor Baseline: Morfessor Baseline plus modeling of allomorphy in stems with string mutations.

- The applied type of mutations can handle significant part of the allomorphic variations in English and Finnish.

Some unsolved issues:

- Improving the efficiency of the search algorithm

# Conclusions

Allomorfessor Baseline: Morfessor Baseline plus modeling of allomorphy in stems with string mutations.

- The applied type of mutations can handle significant part of the allomorphic variations in English and Finnish.

Some unsolved issues:

- Improving the efficiency of the search algorithm
- Obtaining analyses “deep” enough for common word forms

# Conclusions

Allomorfessor Baseline: Morfessor Baseline plus modeling of allomorphy in stems with string mutations.

- The applied type of mutations can handle significant part of the allomorphic variations in English and Finnish.

Some unsolved issues:

- Improving the efficiency of the search algorithm
- Obtaining analyses "deep" enough for common word forms
- Allomorphy in suffixes