Allomorfessor: Towards Unsupervised Morpheme Analysis

Oskar Kohonen   Sami Virpioja   Mikaela Klami

Department of Information and Computer Science
Helsinki University of Technology

17, September 2008

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Information and Computer Science

Outline

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Information and Computer Science

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`

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Information and Computer Science

Finding allomorphs

Allomorphs often very similar to a hypothetical regular form:

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

Use string similarity to find potential allomorphs.

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Information and Computer Science

Finding allomorphs

General metrics of string similarity inappropriate for allomorphy identification.

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

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Information and Computer Science

Finding allomorphs

General metrics of string similarity inappropriate for allomorphy identification.

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

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

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Information and Computer Science

Finding allomorphs

General metrics of string similarity inappropriate for allomorphy identification.

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

Observation: Most allomorphic variation close to boundary between stem and affix.
Solution: Use *string mutations* to encode prior information into model.

Mutations

Simplest thing that could possibly work:

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

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Information and Computer Science

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

Mutations

Simplest thing that could possibly work:

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

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

- `-kä+gä` ≠ `-ko+go`

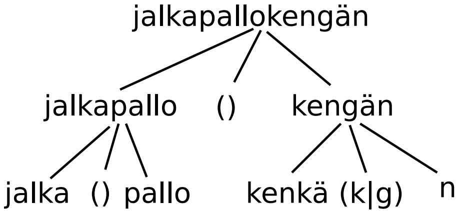
Solution:

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

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Information and Computer Science

Model description

Morfessor-Baseline inspired segmentation model with added mutations.



HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Information and Computer Science

Model description

MAP Formulation:

$$\begin{aligned}\mathcal{M}_{\text{MAP}} &= \arg\max_{\mathcal{M}} P(\mathcal{M}|\text{corpus}) \\ &\propto \arg\max_{\mathcal{M}} P(\text{corpus}|\mathcal{M})P(\mathcal{M}) \\ &= \arg\max_{\mathcal{G}_M,\mathcal{L}_M} P(\mathcal{L}_W|\mathcal{G}_M,\mathcal{L}_M)P(\mathcal{G}_M)P(\mathcal{L}_M).\end{aligned}$$

Segmentation principle

A morph is either generated by:

- String of letters $P(\mathrm{length}(\mu_i)) \prod_{j=1}^{\mathrm{length}(\mu_i)} P(\hat{c}_{ij})$
- Two virtual submorphs $P(\mu_j)P(\delta_k)P(\mu_k)$

Segmentation principle

A morph is either generated by:

- String of letters $P(\text{length}(\mu_i)) \prod_{j=1}^{\text{length}(\mu_i)} P(\hat{c}_{ij})$
- Two virtual submorphs $P(\mu_j)P(\delta_k)P(\mu_k)$

Example:

- String of letters $P(\text{prettier}) = P(\text{length}(\text{prettier}))P(\text{p})P(\text{r})P(\text{e})P(\text{t})P(\text{t})P(\text{i})P(\text{e})P(\text{r})$
- Two virtual submorphs $P(\text{prettier}) = P(\text{pretty})P(\text{y}|\text{i})P(\text{er})$

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Information and Computer Science

Model Learning

Go through words in random order. For each word w, evaluate model probability for different analyses and choose the most probable:

1. No-split, w generated from letters
2. Split word at position i:
   - $w_{..i}$ as virtual prefix, $w_{(i+1)..}$ as virtual suffix, mutation empty
   - If virtual suffix $w_{(i+1)..}$ exists in lexicon, consider mutated forms. Find (at most K) forms that share a prefix with $w_{..i}$, calculate the required mutation.

If the best analysis involved a split, then recursively analyze both parts.

Continue with next word. Repeat for all words, start over, until convergence.

Results

Language	*Precision*	*Recall*	*F-Measure*	*Winner F*
English	83.39%	13.43%	23.13%	56.26%
German	87.92%	7.44%	13.71%	54.06%
Turkish	93.25%	6.15%	11.53%	51.99%
Finnish	92.55%	6.89%	12.82%	48.47%

Model undersegments, but found segments are accurate.

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Information and Computer Science

Example analyses

Mutation	Freq.	Example
-e	2033	abjure (-e) ed
-s	537	actress (-s) s'
-y	386	inequity (-y) able
-n	243	suspicion (-n) ns
-d-e	183	contructed (-d-e) ive

Mutation	Freq.	Example
(-n)	27510	antiikin (-n) lle
(-n-e)	15830	edustajien (-n-e) esi
(-a)	6241	haljeta (-a) essa
(-i)	4203	kliimaksi (-i) in
(-a-t)	2792	alokkaita (-a-t) lle

Conclusions

- Allomorphy is an important phenomenon in morphology
- A statistical model can learn allomorphic variation to a degree
- Further work needed to get results to state-of-art level

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Information and Computer Science