Unsupervised Morphemic Segmentation

Daniel Zeman

zeman@ufal.mff.cuni.cz
Unsupervised Morphemic Segmentation

• Morpho Challenge (shared task) since 2005
  (http://www.cis.hut.fi/morphochallenge2010/)

• Linguistica (John A. Goldsmith)
  (http://humanities.uchicago.edu/faculty/goldsmith/Linguistica2000/)

• Morfessor (Mathias Creutz & Krista Lagus)
  (http://www.cis.hut.fi/projects/morpho/)

• ParaMor (Christian Monson)
  (http://www.cslu.ogi.edu/~monsonc/ParaMor.html)

• Affisix (Michal Hrušecký, MFF)

• Morseus (Dan Zeman, MFF)
  (http://ufal.mff.cuni.cz/~zeman/projekty/morseus/)

• And many others…
Some Terminology

• **Morpheme**
  – Smallest meaningful unit of text / utterance
  – Lexical meaning (e.g. “dog”)
  – Grammatical meaning (e.g. **plural**)

• **Morph**
  – Concrete realization of a morpheme on surface
  – **Allomorphs**: alternating realizations of the same morpheme. E.g. for plural in **en**: *s* or *es*

• For purposes of mere segmentation, the distinction between morpheme and morph does not matter too much
  – However, a smart system might want to figure out that *s* and *es* are morphs of the same morpheme.
Morfessor

- Minimum Description Length (MDL) principle (Rissanen 1989, information theory)
  - How to describe sequential data using a good set of codes?
  - Codebook (vocabulary of morphemes). Cost: how many bits are needed to store it?
  - Coded data (text corpus). Cost: how effective is the text represented using the morphemes from the codebook?
  - Extreme 1: Every word is a morph. Codebook is huge, just one code per token but each code is costly.
  - Extreme 2: Every character is a morph. Codebook is tiny, a code takes 5 bits on average but the number of tokens is unbearable.
  - A tradeoff is sought.
Codebook Cost

• How many bits are needed to store the codebook?
• \( k \) = number of bits needed for 1 character (5 bits needed for an alphabet of 32 lowercase letters)
• \( l(m_j) \) = length in characters of morph \( m_j \)

\[
\sum_{j \in m\text{types}} k \cdot l(m_j)
\]
Corpus Cost

- How effectively is the corpus represented by the codes?
- \( p(m_i) \) = probability of morph \( m_i \) estimated using maximum likelihood (count of occurrences of \( m_i \) / total occurrences of all morphs)
- Negative \( \log_2 \) probability should roughly reflect the number of bits needed to identify this morph in the codebook.

\[
\sum_{i \in \text{mtokens}} - \log p(m_i)
\]
Total Cost

\[ C = \sum_{i \in \text{mtokens}} -\log p(m_i) + \sum_{j \in \text{mtypes}} k \cdot l(m_j) \]
Finnish Motivation Example

*of the bus*  
*not even by the car driver*

*linja-auton*  
*autonkuljettajallakaan*

*linja-*  
*auton*  
*kuljettajallakaan*

*auto*  
*n*  
*kuljettajalla*  
*kaan*

*kuljettaja*  
*illa*
Input: Words with Frequencies (Corpus)

- 7174 said
- 3257 new
- 3078 company
- 2753 year
- 2648 market
- 2467 says
- 2423 pos
- 2317 tags
- 2002 stock
- 1867 also
- 1808 other
- 1798 share
- 1482 last
- 1444 shares
- 1431 president
- 1426 years
- 1415 trading
- 1331 sales
- 1195 fixing
- 1188 only
- 1171 business
- 1164 such
On-Line Training

• Read next token
• Try to split it into two morphs (new tokens)
  – Consider all possible split positions
  – Does the total cost (codebook + corpus) decrease?
• If split, recursively try to split each new morph
• “Dreaming” – at regular intervals, re-read previously segmented words in random order and re-segment them
Output: Segmented Words

- 254 are + a
- 243 with + in
- 132 a + .
- 69 no + .
- 54 s + .
- 48 million + s
- 47 he + at
- 41 billion + s
- 41 s + on
- 37 a + part
- 36 be + at
- 36 s + it
- 31 on + to
- 28 just + in
- 27 s + at
- 26 the + me
- 24 s + and
- 22 president + s
- 20 i + .
- 19 commercial + s
- 18 like + s
- 16 to + night
- 15 announcement + s
- 15 average + s
Baseline Morfessor Evaluation

- Number of morphemes per word is not limited
- It recognizes only *very frequent* morphs
- It does not want to split very frequent *words*
- Maximum one analysis per word
  - What about cs: *proud + it vs. pro + ud + it*
- It cannot detect phonological/spelling changes
  - *baby + es ⇒ babies*
  - But this is really difficult for unsupervised approaches
- It does not distinguish between prefixes and suffixes
  - *s + it … is that the plural “s”?*
  - Later extension can distinguish these
Logarithmic Frequency Scale

- \( n = \text{int}(\log(n)) + 1 \)
- \( n \) is now between 1 and 12. Example results: [show?]
  - a + s
  - it + s
  - say + s
  - share + s
  - year + s
  - s + o
  - synch + ing
  - accord + ing
  - third- + quarter

  - compar + ed
  - increase + d
  - business + es
  - current + ly
  - s + pending
  - transport + ation
  - institution + s

  - Even if both parts already exist in the codebook, splitting may not occur if it does not shorten the corpus encoding:
    - shareholder
    - over-the-counter
    - represent + ed
Frequency-less Scale

- $n = 1$
- $Sn$ is now always 1. Results:
  - th + at
  - the + y
  - be + cause
  - comp + an + ies
  - y + es + ter + day
  - bet + we + en
  - international *(no split?)*
  - depart + ment
  - spoke + s + man
  - administr + ation
  - re + cent + ly
  - lon + don
  - dec + lined
  - politic + al
  - la + test
  - francis + co
  - wash + ing + ton
  - pro + pose + d
  - euro + pe
  - out + standing
  - in + s + tea + d
  - perform + ance
  - com + petition
  - dis + closed
Morfessor Categories-ML

• Creutz and Lagus 2004
• Improved performance of the baseline Morfessor
• Words modeled by Hidden Markov Model
  – Cannot begin with suffix
  – Cannot end with prefix
  – Suffix cannot follow prefix without traversing a stem
• Very short morphs can be recognized as noise and joined with neighboring morphs
• Unlike baseline Morfessor (and unlike later Catmap), Categories-ML ignores word frequency
Morfessor-Catmap

- New algorithm by Creutz and Lagus (2005)
- Four categories of morphs:
  - Prefix (PRE)
  - Stem (STM)
  - Suffix (SUF)
  - Non-morpheme (NON)
- Hierarchical lexicon: morph consists of:
  - Either string of letters
  - Or two submorphs
- Word is modeled using an HMM (see above)
Search Algorithm

1. Initialization of segmentation
2. Splitting of morphs
3. Joining of morphs using a bottom-up strategy
4. Splitting of morphs
5. Resegmentation of corpus using Viterbi algorithm and re-estimation of probabilities until convergence
6. Repetition of steps 3–5 once
7. Expansion of the morph substructures to the finest resolution that does not contain non-morphemes
Initialization

• Morfessor baseline algorithm
• No morph categories are used
• Resulting morphs are categorized (tagged) as PRE / STM / SUF / NON
Splitting of Morphs

- Morphs are ordered by increasing length
- Most probable split into two submorphs (or no split) is chosen
- Different category taggings of the morphs are tested (HMM) in four contexts:
  - Word initial
  - Word final
  - Word initial and final
  - Word internal
- At times the morph splitting is interrupted
- Whole corpus is retagged using Viterbi algorithm
- Probabilities are re-estimated, then splitting resumes
Joining of Morphs Bottom-up

• Starting with most frequent morph bigrams, proceeding in order of decreasing frequency
• The most probable alternative is chosen:
  – Keep the two morphs separate
  – Concatenate them to an atomic morph
  – Add a higher-level morph internally structured to the two
• Different category taggings in different contexts are tested
• At times the joining of morphs is interrupted
• Whole corpus is retagged using the Viterbi algorithm
• Probabilities are re-estimated, then joining resumes
Morfessor-Catmap

- 38 address
- 11 address + ed
- 6 address + es
- 10 address + ing
- 18 adjust
- 27 adjust + able
- 67 adjust + ed
- 6 adjust + er
- 18 adjust + ers
- 9 adjust + ing
- 26 adjust + ment
- 27 adjust + ment + s
- 2 adjust + s
- 2 advanc + ed + - + technology
- 1 advanc + e + ment + s
- 1 al + am + ed + a
- 1 albert + ville
- 1 amsterdam + rot + ter + dam
- 1 a + sept + ically
- 3 back + fire + d
- 1 begin + n + ing + s
- 33 bio + technology
- 2 book + store + s
- 5 bou + nc + ing
- 1 bou + que + t
- 16 bourbon
Zellig Harris (1955)
Hervé Déjean (1998)

- Number of different letters that follow a given sequence of letters
- Increase of this number indicates a morpheme boundary. Corpus example:
  - After *direc* the only possibility is *t*
  - After *direct* the possible continuations are *i, l, o, e (direction, directly, director, directed)*
- False segmentations may be generated:
  - *start+ed, start+led, start+ling*
- Déjean’s improvement: three steps
  - Create list of most frequent morphs (*prefixes or suffixes*)
  - Extend the list by segmenting words with the help of already found morphs (50+% of continuations are known ⇒ others are morphs, too)
  - Segment all words using morphs learned in previous steps
Paradigm Acquisition

• Morphological paradigms, e.g. indicative verb

**cs:**
- děl + ám *I do*
- děl + áš *you do* (singular)
- děl + á *he / she / it does*
- děl + áme *we do*
- děl + áte *you do* (plural)
- děl + ají *they do*
Paradigm Acquisition

• Morphological paradigms, e.g. indicative verb
  **cs:**
  - řík + ám  \( I \text{ say} \)
  - řík + áš  \( you \text{ say} \) (singular)
  - řík + á  \( he / she / it \text{ says} \)
  - řík + áme  \( we \text{ say} \)
  - řík + áte  \( you \text{ say} \) (plural)
  - řík + ají  \( they \text{ say} \)
Paradigm Acquisition

- Morphological paradigms, e.g. indicative verb in Czech:
  - ber + u: I take
  - ber + eš: you take (singular)
  - ber + e: he / she / it takes
  - ber + eme: we take
  - ber + ete: you take (plural)
  - ber + ou: they take
The Idea

• Find frequently occurring suffixes
• Word-final string is not suffix if the remainder cannot occur alone or with other suffixes
  – Otherwise almost every letter could act as a frequent short suffix
  – Cyclic dependency: add a suffix ⇒ new strings become stems (occur with multiple suffixes) ⇒ new strings become suffixes (occur with the new stem) etc.
• A paradigm:
  – Set of suffixes occurring with the same stems
  – Set of stems occurring with these suffixes
• Prefixes can be found symmetrically
• What about compounds or complex affixes?
Some English “Paradigms”

- impersonat, incinerat
  - e, ed, es, ing, ion, ions, or, or’s, ors, ors’
- dwell, hijack
  - 0, ed, er, er’s, ers, ers’, ing, ing’s, ings, s
- demorali, visuali
  - sation, se, sed, sing, zation, ze, zed, zes, zing
- activat, cultivat, eliminat, emulat, exterminat, orchestrat, persecut, pontificat, terminat
  - e, ed, es, ing, ion, ions, or, ors
- abridg, acknowledg
  - e, ed, ement, ements, es, ing, ment, ments
- enthusiast, nomad, pessimist
  - ’s, 0, ic, ically, s, s’
Algorithm

- Assumption (wrong in general, OK for paradigms): maximum 1 split per word
- Consider all possible splits (including no-split) of all words
  - bank, ban+k, ba+nk, b+ank
  - Word frequencies are not used although they could help identify typos at least
- Identify sets of stems and suffixes occurring together: paradigm candidates
- Filter redundant paradigms
More Suffixes than Stems

• Both stems and suffixes can consist of just one letter
• How to rule out crazy paradigms such as
  – Single stem $s$
  – Thousands of “suffixes” for all words beginning in $s$
• Requiring that there be more stems than suffixes seems to be a reasonable heuristic
  – Real paradigms typically meet this requirement
Single Suffix Paradigms

- Not interesting
  - They merely state that a group of words end in the same sequence of letters

- Unreliable, especially if short
  - Suffix $n$ and thousands of “stems” for words ending in $n$

- They violate the linguistic principle of \textit{repeatability} of morphemes (stems in this case)

- Discard them
Subset Merging

- Many stems have not occurred with all applicable suffixes.

**CS:**
- A.suff = ou, á, é, ého, ém, ému, ý, ých, ým, ými
- B.suff = ou, á, é, ého, ém, ému, ý, ých, ým
- C.suff = ou, á, é, ého, ém, ý, ých, ým, ými
- D.suff = ou, á, é, ého, ém, ý, ých, ým

- Here, B, C and D are just incomplete instances of underlying A
  - New stem-suffix combinations help cover unseen words

- In general, merging of paradigms could introduce stem-suffix combinations that are not permitted

- More than one superset? Either create union or leave as is
Paradigm Filtering

- **Finnish Paradigm A**
  - Suff = a, in, ksi, lla, lle, n, na, ssa, sta
  - Stem = erikokoisi, funktionaalis, logistisi, mustavalkoisi, objektiivisi, rajallisi, subjektiivisi, tuotannollisi, uudenlaisi

- **Finnish Paradigm B**
  - Suff = ia, iin, iksi, illa, ille, in, ina, issa, ista
  - Stem = erikokois, funktionaalis, logistis, mustavalkois, objektiivis, rajallis, subjektiivis, tuotannollis, uudenlais

- **Finnish Paradigm C**
  - Suff = sia, siin, siksi, silla, sille, sin, sina, sissa, sista
  - Stem = erikokoi, funktionaali, logisti, mustavalkoi, objektiivi, rajalli, subjektiivi, tuotannoll, uudenlai

- **Finnish Paradigm D**
  - Suff = isia, isiin, isiksi, isilla, isille, isin, isina, isissa, isista
  - Stem = erikoko, funktionaal, logist, mustavalko, objektiiv, rajall, subjektiiv, tuotannoll, uudenla
Repeating Border Letter

- Two or more paradigm candidates
  - Does the border letter belong to the stem or to the suffix?
- Mapping between the candidates need not be reversible:
  \[ \text{CS:} \]
  - A.suff = l, la, li, lo, ly
  - A.stem = kouř, nosi, pádi
  - B.suff = il, ila, ili, ilo, ily, ů
  - B.stem = kouř, nos, pád

- Paradigm B can add suffixes but cannot add stems
  - Added stems would project to Paradigm A, too
Repeating Border Letter

• Two or more paradigm candidates
  – Does the border letter belong to the stem or to the suffix?

• Mapping between the candidates need not be reversible:
  
  **CS:**
  – A.suff = l, la, li, lo, ly
  – A.stem = kouři, nosi, pádi, sedě
  – B.suff = il, ila, ili, ilo, ily
  – B.stem = kouř, nos, pad

• Paradigm A can add stems but cannot add suffixes
  – Added suffixes would project to Paradigm B, too
Using Paradigms to Segment Words

- **Strict:** only stem-suffix combinations that occur in the same paradigm
  - Can cover unseen words because of subset merging
- **Weaker:** only known stems and suffixes (but they can be known from different paradigms)
  - Can help in cases where subset merging failed
- **Weakest:** allow known suffixes even with totally unknown stems
  - Reflects the fact that paradigms can be productively applied to new words
  - Unreliable: how do we know that this particular stem would belong to this paradigm?