Unsupervised Morphemic Segmentation

Daniel Zeman

November 1, 2023
Unsupervised Morphemic Segmentation

- Morpho Challenge (shared task) since 2005
- 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 (Daniel Zeman, MFF)
  (http://ufal.mff.cuni.cz/~zeman/projekty/morseus/)
- And many others…

- Byte Pair Encoding (Rico Sennrich, 2016)
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 English: *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
"of the bus"  
linja-auton  

"not even by the car driver"  
autonkuljettajallakaan  

drive  
kuljettajallakaan  

even  
kaan  

"car"  
auto  

"of"  
n  

"driver"  
kuljettaja  

"at/by"  
lla
• 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 efficiently 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 \(m_j\)

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

• How efficiently 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.

$$\sum_{i \in m\text{-tokens}} - \log_2 p(m_i)$$
Total Cost

\[ C = \sum_{j \in m\text{-types}} k \times l(m_j) + \sum_{i \in m\text{-tokens}} -\log_2 p(m_i) \]
• *hello world*
• DICT = 25 + 25 = 50
• CORP = $-2 \times \log(1/2) = 2$
• TOTAL = 52
Example

- *hello worlds*
- DICT = 25 + 30 = 55
- CORP = $-2 \times \log(1/2) = 2$
- TOTAL = 57
Example

- *hello world and hellos other worlds*
- \( \text{DICT} = 25 + 25 + 15 + 30 + 25 + 30 = 150 \)
- \( \text{CORP} = -6 \times \log(1/6) = 15.5 \)
- \( \text{TOTAL} = 165.5 \)
Example

• *hello world and hello other worlds*

• $\text{DICT} = 15 + 10 + 15 + 10 + 15 + 15 + 25 + 15 = 120$

• $\text{CORP} = -4 \times \log(1/5) - 6 \times \log(1/10) = 29.2$

• $\text{TOTAL} = 149.2$
Example

• *hello world and hello s other world s*
• \( \text{DICT} = 25 + 25 + 15 + 5 + 25 = 95 \)
• \( \text{CORP} = -6 \times \log(1/4) - 2 \times \log(1/8) = 18 \)
• \( \text{TOTAL} = 113 \)
Example

• *hello world and hello other worlds*
• DICT = 11 × 5 = 55
• CORP =
  \[-6 \times \log(1/5) - 5 \times \log(1/6) - 12 \times \log(1/10) - 4 \times \log(1/15) - 3 \times \log(1/30) = 97.1\]
• TOTAL = 152.1
## Input: Words with Frequencies (Corpus)

<table>
<thead>
<tr>
<th>Word</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>said</td>
<td>7141</td>
</tr>
<tr>
<td>new</td>
<td>3257</td>
</tr>
<tr>
<td>company</td>
<td>3078</td>
</tr>
<tr>
<td>year</td>
<td>2753</td>
</tr>
<tr>
<td>market</td>
<td>2648</td>
</tr>
<tr>
<td>says</td>
<td>2467</td>
</tr>
<tr>
<td>stock</td>
<td>2002</td>
</tr>
<tr>
<td>also</td>
<td>1867</td>
</tr>
<tr>
<td>other</td>
<td>1808</td>
</tr>
<tr>
<td>share</td>
<td>1798</td>
</tr>
<tr>
<td>last</td>
<td>1482</td>
</tr>
<tr>
<td>shares</td>
<td>1444</td>
</tr>
<tr>
<td>president</td>
<td>1431</td>
</tr>
<tr>
<td>years</td>
<td>1426</td>
</tr>
<tr>
<td>trading</td>
<td>1415</td>
</tr>
<tr>
<td>sales</td>
<td>1331</td>
</tr>
<tr>
<td>only</td>
<td>1188</td>
</tr>
<tr>
<td>business</td>
<td>1171</td>
</tr>
<tr>
<td>such</td>
<td>1164</td>
</tr>
<tr>
<td>york</td>
<td>1129</td>
</tr>
<tr>
<td>group</td>
<td>1102</td>
</tr>
<tr>
<td>time</td>
<td>1032</td>
</tr>
</tbody>
</table>
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
<table>
<thead>
<tr>
<th>Word</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>are+a</td>
<td>254</td>
</tr>
<tr>
<td>with+in</td>
<td>243</td>
</tr>
<tr>
<td>a+</td>
<td>132</td>
</tr>
<tr>
<td>no+</td>
<td>69</td>
</tr>
<tr>
<td>s+</td>
<td>54</td>
</tr>
<tr>
<td>million+s</td>
<td>48</td>
</tr>
<tr>
<td>he+at</td>
<td>47</td>
</tr>
<tr>
<td>billion+s</td>
<td>41</td>
</tr>
<tr>
<td>s+on</td>
<td>41</td>
</tr>
<tr>
<td>a+part</td>
<td>37</td>
</tr>
<tr>
<td>be+at</td>
<td>36</td>
</tr>
<tr>
<td>s+it</td>
<td>36</td>
</tr>
<tr>
<td>on+to</td>
<td>31</td>
</tr>
<tr>
<td>just+in</td>
<td>28</td>
</tr>
<tr>
<td>s+at</td>
<td>27</td>
</tr>
<tr>
<td>the+me</td>
<td>26</td>
</tr>
<tr>
<td>s+and</td>
<td>24</td>
</tr>
<tr>
<td>president+s</td>
<td>22</td>
</tr>
<tr>
<td>i+</td>
<td>20</td>
</tr>
<tr>
<td>commercial+s</td>
<td>19</td>
</tr>
<tr>
<td>like+s</td>
<td>18</td>
</tr>
<tr>
<td>to+night</td>
<td>16</td>
</tr>
<tr>
<td>announcement+s</td>
<td>15</td>
</tr>
<tr>
<td>average+s</td>
<td>15</td>
</tr>
</tbody>
</table>
Baseline Morfessor Evaluation

- Number of morphemes per word is not limited
- It recognizes only *very frequent* morphemes
- 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
  - en: *baby + es* ⇒ *babies*
  - But it is really difficult for unsupervised approaches
- It does not distinguish between prefixes and suffixes
  - en: *s + it* ... is that the plural “s”?
  - Later extension can distinguish these
$n = \text{int}(\log(n)) + 1$

$n$ is now between 1 and 12. Example results:

- 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
$n = 1$

$n$ is now always 1. Example results:

- that
- they
- because
- companies
- yes
ter
day
- between
- international
- (no split?)
- department
- spokesman
- administration
- recently
- london
- decided
- political
- latest
- francisco
- washington
- proposed
- europe
- outstanding
- in
tea
d
- performed
- competition
- dismissed
• 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
• Creutz and Lagus 2005, new algorithm
• 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 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
• 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 Viterbi algorithm
• Probabilities are re-estimated, then joining resumes
<table>
<thead>
<tr>
<th>word</th>
<th>count</th>
<th>word</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>address</td>
<td>38</td>
<td>advanc+ed+-+technology</td>
<td>2</td>
</tr>
<tr>
<td>address+ed</td>
<td>11</td>
<td>advanc+e+ment+s</td>
<td>1</td>
</tr>
<tr>
<td>address+es</td>
<td>6</td>
<td>al+am+ed+a</td>
<td>1</td>
</tr>
<tr>
<td>address+ing</td>
<td>10</td>
<td>albert+ville</td>
<td>1</td>
</tr>
<tr>
<td>adjust</td>
<td>18</td>
<td>amsterdam+-+rot+ter+dam</td>
<td>1</td>
</tr>
<tr>
<td>adjust+able</td>
<td>27</td>
<td>a+sept+ically</td>
<td>1</td>
</tr>
<tr>
<td>adjust+ed</td>
<td>67</td>
<td>back+fire+d</td>
<td>3</td>
</tr>
<tr>
<td>adjust+er</td>
<td>6</td>
<td>begin+n+ing+s</td>
<td>1</td>
</tr>
<tr>
<td>adjust+ers</td>
<td>18</td>
<td>bio+technology</td>
<td>33</td>
</tr>
<tr>
<td>adjust+ing</td>
<td>9</td>
<td>book+store+s</td>
<td>2</td>
</tr>
<tr>
<td>adjust+ment</td>
<td>26</td>
<td>bou+nc+ing</td>
<td>5</td>
</tr>
<tr>
<td>adjust+ment+s</td>
<td>27</td>
<td>bou+que+t</td>
<td>1</td>
</tr>
<tr>
<td>adjust+s</td>
<td>2</td>
<td>bourbon</td>
<td>16</td>
</tr>
</tbody>
</table>
• Number of different letters that follow a given sequence of letters
• Increase of this number indicates a morpheme boundary. Corpus example (en):
  • 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 in 🇨🇿 Czech:

  - **děl + ám** “I do”
  - **děl + ěš** “you do” (singular)
  - **děl + á** “he / she / it does”
  - **děl + áme** “we do”
  - **děl + áte** “you do” (plural)
  - **děl + ají** “they do”
• Morphological paradigms, e.g. indicative verb in 🇨🇿 Czech:

\[ \begin{align*}
\text{řík} + \text{ám} & \quad \text{“I say”} \\
\text{řík} + \text{áš} & \quad \text{“you say” (singular)} \\
\text{řík} + \text{á} & \quad \text{“he / she / it says”} \\
\text{řík} + \text{áme} & \quad \text{“we say”} \\
\text{řík} + \text{áte} & \quad \text{“you say” (plural)} \\
\text{řík} + \text{ají} & \quad \text{“they say”}
\end{align*} \]
Paradigm Acquisition

- Morphological paradigms, e.g. indicative verb in 

\[
\begin{align*}
ber + u & \quad \text{"I take"} \\
ber + eš & \quad \text{"you take" (singular)} \\
ber + e & \quad \text{"he / she / it takes"} \\
ber + eme & \quad \text{"we take"} \\
ber + ete & \quad \text{"you take" (plural)} \\
ber + ou & \quad \text{"they take"}
\end{align*}
\]
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 *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, funktionaalisi, logistisi, mustavalkoisi, objektiivisi, rajallisi, subjektiivisi, tuotannollisi, uudenlaisi
Paradigm Filtering

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

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

- Finnish paradigm A
  - Suff = a, in, ksi, lla, lle, n, na, ssa, sta
  - Stem = erikokoisi, funktionaalis, logistis, mustavalkois, objektiivisi, rajallis, subjektiivisi, tuotannollisi, uudenlaisi

- Finnish paradigm B
  - Suff = ia, iin, iksi, illa, ille, in, ina, isa, ista
  - Stem = erikokois, funktionaalis, logistis, mustavalkois, objektiivis, rajallis, subjektiivis, tuotannollis, uudenlaisi

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

• **Finnish paradigm A**
  - Suff = a, in, ksi, lla, lle, n, na, ssa, sta
  - Stem = erikokoisi, funktionaalisi, 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, uudenlaiss

• **Finnish paradigm C**
  - Suff = sia, siin, siksi, sillä, sille, sin, sina, sissa, sista
  - Stem = erikokoi, funktionaali, logisti, mustavalkoi, objektiivi, rajalli, subjektiivi, tuotannolli, uudenlai

• **Finnish paradigm D**
  - Suff = isia, isiin, isikki, isilla, isille, isn, 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

   \[\begin{array}{l}
   \text{cs}: \\
   \text{A.suff} = l, la, li, lo, ly \\
   \text{A.stem} = kouři, nosi, pádi \\
   \text{B.suff} = il, ila, ili, ilo, ily, ů \\
   \text{B.stem} = kouř, nos, pād \\
   \end{array} \]

• Paradigm B can add suffixes but cannot add stems
  • Added stems would project to paradigm A, too
• 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, pád

• Paradigm A can add stems but cannot add suffixes
  • Added suffixes would project to paradigm B, 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 ⇒ ignore/split/merge?

**cs:**
- $A_1.\text{suff} = l, \text{la, li, lo, ly}$
- $A_1.\text{stem} = \text{kouři, nosi, pádi, sedě}$
- $A_2.\text{suff} = ů$
- $A_2.\text{stem} = \text{kouř, nos, pád, (sed)}$
- $B_1.\text{suff} = il, \text{ila, ili, ilo, ily, ů}$
- $B_1.\text{stem} = \text{kouř, nos, pád}$
- $B_2.\text{suff} = ěl, ěla, ěli, ělo, ěly, (ů)$
- $B_2.\text{stem} = \text{sed}$
Using Paradigms to Segment Words

- **Strict:** only stem-suffix combinations that occur in the same paradigm
  - Can cover unseen words because of subset merging
  - Highest precision

- **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
  - Reflect 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?
  - Highest recall
Byte Pair Encoding (BPE)

  - Input: sequence of bytes
  - Vocabulary of bytes with frequencies, some bytes are unused
  - Unused bytes can be used for compression: they will represent *pairs of bytes*

1. Find most frequent pair of neighboring bytes
2. Replace their occurrences with the first unused byte, remember the mapping (e.g. \( z = ab \))
3. Repeat while there are unused bytes and at least one byte pair that occurs more than once

Adapted for tokenization in NLP (Rico Sennrich, Barry Haddow, Alexandra Birch: Neural Machine Translation of Rare Words with Subword Units, ACL, 2016)

- Subword vocabulary of desired size
- Initially: characters are subwords
  - Find most frequent pair of subwords, add it as a new subword
  - Unknown words can be represented using subwords
Byte Pair Encoding (BPE)

  - Input: sequence of bytes
  - Vocabulary of bytes with frequencies, some bytes are unused
  - Unused bytes can be used for compression: they will represent pairs of bytes
  
  1. Find most frequent pair of neighboring bytes
  2. Replace their occurrences with the first unused byte, remember the mapping (e.g. $z = ab$)
  3. Repeat while there are unused bytes and at least one byte pair that occurs more than once

- Adapted for tokenization in NLP (Rico Sennrich, Barry Haddow, Alexandra Birch: Neural Machine Translation of Rare Words with Subword Units, ACL, 2016)
  - Subword vocabulary of desired size
  - Initially: characters are subwords
  - Find most frequent pair of subwords, add it as a new subword
  - Unknown words can be represented using subwords