Unsupervised Morphemic Segmentation

Daniel Zeman

October 23, 2020

EUROPEAN UNION
European Structural and Investment Fund
Operational Programme Research, Development and Education

Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

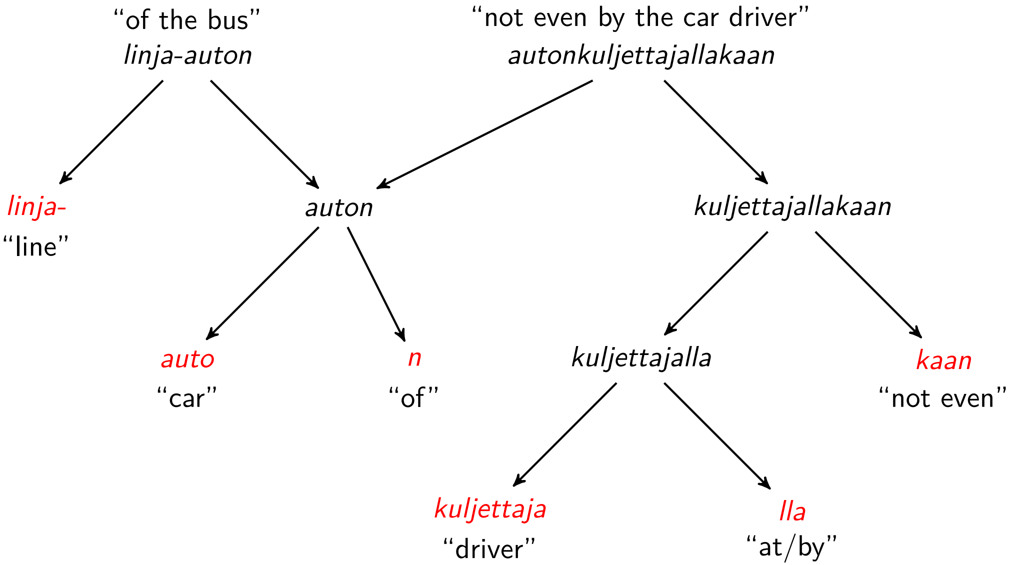
Unsupervised Morphemic Segmentation

- 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...

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

Finnish Motivation Example



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

$$\sum_{j \in \text{m-types}} k \times l(m_j)$$

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.

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

Total Cost

$$C = \sum_{j \in \text{m-types}} k \times l(m_j) + \sum_{i \in \text{m-tokens}} -\log_2 p(m_i)$$

Example

- *hello world*
- $\text{DICT} = 25 + 25 = 50$
- $\text{CORP} = -2 \times \log(1/2) = 2$
- $\text{TOTAL} = 52$

Example

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

Example

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

Example

- *hel lo wor ld and hel los other wor lds*
- DICT $= 15 + 10 + 15 + 10 + 15 + 15 + 25 + 15 = 120$
- CORP $= -2 \times \log(1/5) - 6 \times \log(1/10) = 24.6$
- TOTAL $= 144.6$

Example

- *hello world and hello s other world s*
- DICT $= 25 + 25 + 15 + 5 + 25 = 95$
- CORP $= -3 \times \log(1/4) - 2 \times \log(1/8) = 12$
- TOTAL $= 107$

Example

- $h\ e\ l\ l\ o\ w\ o\ r\ l\ d\ a\ n\ d\ h\ e\ l\ l\ o\ s\ o\ t\ h\ e\ r\ w\ o\ r\ l\ d\ s$
- DICT $= 11 \times 5 = 55$
- CORP $= -4 \times \log(1/10) - \log(1/5) - \log(1/6) - 2 \times \log(1/15) - 3 \times \log(1/30) = 40.7$
- TOTAL $=$ 95.7

- It should be better with a larger corpus!

Input: Words with Frequencies (Corpus)

said	7141	shares	1444
new	3257	president	1431
company	3078	years	1426
year	2753	trading	1415
market	2648	sales	1331
says	2467	only	1188
stock	2002	business	1171
also	1867	such	1164
other	1808	york	1129
share	1798	group	1102
last	1482	time	1032

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

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

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

Logarithmic Frequency Scale

```
$n = int(log($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

Frequency-less Scale

```
$n = 1
```

$n is now always 1. Example 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
- 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

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

- 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

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 Viterbi algorithm
- Probabilities are re-estimated, then joining resumes

Morfessor-Catmap

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

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 (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 + *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 in Czech:

řík + ám	"I say"
řík + áš	"you say" (singular)
řík + á	"he / she / it says"
řík + áme	"we say"
řík + áte	"you say" (plural)
řík + ají	"they 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 $\Rightarrow$ new strings become stems (occur with multiple suffixes) $\Rightarrow$ 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, 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, uudenlais*

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, uudenlais*
- 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, uudenlais*
- Finnish paradigm C
  - Suff = *sia, siin, siksi, silla, sille, sin, sina, sissa, sista*
  - Stem = *erikokoi, funktionaali, logisti, mustavalkoi, objektiivi, rajalli, subjektiivi, tuotannolli, 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
  cs:
  - A.suff = *l, la, li, lo, ly*
  - A.stem = *kouři, 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, pád*
- 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
  - 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