Word Segmentation and Morphology

11-711 Advanced NLP

October 2021

(Some slides adapted from Lori Levin, David Mortenson, and J&M)

Two major classes of approaches

- Linguistic approaches:
  - Segmenting into words that make sense with grammars/meanings
  - Segmenting into subword units that make sense with grammars/meanings
- Technological approaches:
  - Segmenting into words to make processing efficient/better
  - Segmenting into subwords to make processing efficient/better

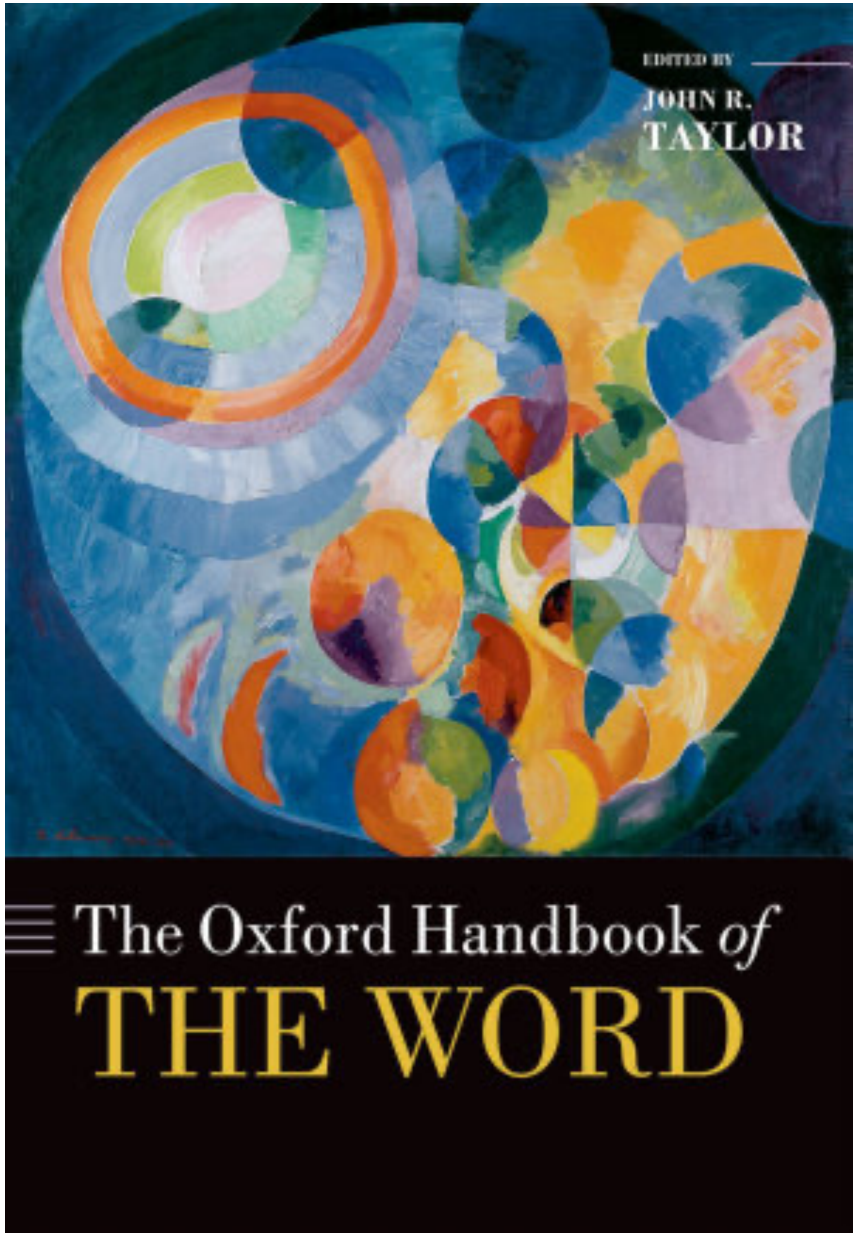
We will look at both; linguistic approaches matter more if later stages involve parsing and/or semantics

But first: What is a word?

- The things that are in the dictionary?
  - But how did the lexicographers decide what to put in the dictionary?
- The things between spaces and punctuation?
- The smallest unit that can be uttered in isolation?
  - You could say this word in isolation: *Unimpressively*
  - This one too: *impress*
  - But you probably wouldn't say these in isolation, unless you were talking about morphology:
    - *un*
    - *ive*
    - *ly*

So what is a word?

- Can get pretty tricky:
  - didn’t
  - would’ve
  - gonna
  - shoulda woulda coulda
  - Ima
  - blackboard (*vs*. school board)
  - baseball (*vs*. golf ball)
  - the person who left**’s** hat; Jim and Gregg**’s** apartment
  - acct.
  - LTI



About 1000 pages. $139.99

You don't have to read it.

The point is that it takes 1000 pages just to survey the issues related to what words are.

So what is a word?

- It is up to you or the software you use for processing words.
- Take linguistics classes.
- Make good decisions in software design and engineering.

Tokenization

Tokenization

- Some Asian languages have obvious issues:

  利比亚“全国过渡委员会”执行委员会主席凯卜22日在首都的黎波里公布“过渡政府”内阁名单，宣告过渡政府正式成立。

Tokenization

- Some Asian languages have obvious issues:

  利比亚“全国过渡委员会”执行委员会主席凯卜22日在首都的黎波里公布“过渡政府”内阁名单，宣告过渡政府正式成立。

- But German too: Noun-noun compounds:

  Gesundheitsversicherungsgesellschaften

Tokenization

- Some Asian languages have obvious issues:

  利比亚“全国过渡委员会”执行委员会主席凯卜22日在首都的黎波里公布“过渡政府”内阁名单，宣告过渡政府正式成立。

- But German too: Noun-noun compounds:

Gesundheits-versicherungs-gesellschaften (health insurance companies)

Tokenization

- Some Asian languages have obvious issues:

  利比亚“全国过渡委员会”执行委员会主席凯卜22日在首都的黎波里公布“过渡政府”内阁名单，宣告过渡政府正式成立。

- But German too: Noun-noun compounds:

  Gesundheitsversicherungsgesellschaften

- Spanish clitics: *Darmelo*

Tokenization

- Some Asian languages have obvious issues:

  利比亚“全国过渡委员会”执行委员会主席凯卜22日在首都的黎波里公布“过渡政府”内阁名单，宣告过渡政府正式成立。

- But German too: Noun-noun compounds:

  Gesundheitsversicherungsgesellschaften

- Spanish clitics: *Dar-me-lo (To give me it)*

Tokenization

- Some Asian languages have obvious issues:

  利比亚“全国过渡委员会”执行委员会主席凯卜22日在首都的黎波里公布“过渡政府”内阁名单，宣告过渡政府正式成立。

- But German too: Noun-noun compounds:

  Gesundheitsversicherungsgesellschaften

- Spanish clitics: *Darmelo*

- Even English has issues, to a smaller degree: *Gregg and Bob’s house*

Tokenization

Input: raw text

```
Dr. Smith said tokenization of English is “harder than you’ve thought.”
When in New York, he paid $12.00 a day for lunch and wondered what it would
be like to work for AT&T or Google, Inc.
```

Output from Stanford Parser: http://nlp.stanford.edu:8080/parser/index.jsp
with part-of-speech tags:

```
Dr./NNP Smith/NNP said/VBD tokenization/NN of/IN English/NNP
is/VBZ ``/`` harder/JJR than/IN you/PRP 've/VBP thought/VBN ./.
''/''

When/WRB in/IN New/NNP York/NNP ,/, he/PRP paid/VBD $/$ 12.00/CD
a/DT day/NN for/IN lunch/NN and/CC wondered/VBD what/WP it/PRP
would/MD be/VB like/JJ to/TO work/VB for/IN AT&T/NNP or/CC
Google/NNP ,/, Inc./NNP ./.
```

Tokenization approaches

- Traditional:
  - For languages with word spaces: spaces, punctuation, plus rules
  - For Chinese etc: Large dictionaries, punctuation, plus rules

- SentencePiece:
  - For Chinese etc: Use subword segmentation to find the words *without* pre-tokenization

    (We'll see subword segmentation later today.)

Sentence Segmentation

- !, ? mostly unambiguous but **period** "." is very ambiguous:
  - Sentence boundary
  - Abbreviations like Inc. or Dr.
  - Numbers like .02% or 4.3
- Common algorithm: Tokenize first: use rules or ML to classify a period as either (a) part of the word or (b) a sentence-boundary.
  - An abbreviation dictionary can help
- Sentence segmentation can then often be done by rules based on this tokenization.

Morphological Phenomena

What is Linguistic Morphology?

- Morphology is the study of the internal structure of words.
  - **Derivational morphology.** How new words are created from existing words.
    - *[grace]*
    - *[[grace]ful]*
    - *[un[grace]ful]]*
  - **Inflectional morphology.** How features relevant to the syntactic context of a word are marked on that word.
    - This example illustrates number (singular and plural) and tense (present and past).
    - Green indicates irregular. Blue indicates zero marking of inflection. Red indicates regular inflection.
    - This student walks.
    - These students walk.
    - These students walked.
  - ***Compounding.*** Creating new words by combining existing words
    - With or without spaces: surfboard, golf ball, blackboard

Morphemes

- **Morphemes.** Minimal pairings of form and meaning.
  - **Roots.** The “core” of a word that carries its basic meaning.
    - *apple* : ‘apple’
    - *walk* : ‘walk’
  - **Affixes** (**prefixes**, **suffixes**, **infixes**, and **circumfixes**). Morphemes that are added to a base (a root or stem) to perform either derivational or inflectional functions.
    - *un-* : ‘NEG’
    - *-s* : ‘PLURAL’

Language Typology

Types of Languages:

- In order of morphological complexity:
  - Isolating (or Analytic)
  - Fusional (or Inflecting)
  - Agglutinative
  - Polysynthetic
  - Others

Isolating Languages: Chinese

Little morphology other than compounding

- **Chinese** inflection
  - few affixes (prefixes and suffixes):
    - 们： 我们， 你们， 他们， 。 。 。 同志们
      mén: wǒmén, nǐmén, tāmén, tóngzhìmén
      plural: we, you (pl.), they comrades, LGBT people
    - “suffixes” that mark aspect: 着 *-zhě* ‘continuous aspect’
- Chinese derivation
  - 艺术家 ***yìshù****jiā* **‘artist’**
- Chinese is a champion in the realm of compounding—up to 80% of Chinese words are actually compounds.

毒	+	贩	→	毒贩
dú		*fàn*		*dúfàn*
‘poison, drug’		‘vendor’		‘drug trafficker’

Agglutinative Languages: Swahili

Verbs in Swahili have an average of 4-5 morphemes, http://wals.info/valuesets/22A-swa

Swahili	English
mtu alilala	'The person slept'
mtu atalala	'The person will sleep'
watu walilala	'The people slept'
watu watalala	'The people will sleep'

- Orange prefixes mark noun class (like gender, except **Swahili** has nine instead of two or three).
  - Verbs agree with nouns in noun class.
  - Adjectives also agree with nouns.
  - Very helpful in parsing.
- Black prefixes indicate tense.

Turkish

Example of extreme agglutination

But most Turkish words have around three morphemes

uygarlaştıramadıklarımızdanmışsınızcasına

"(behaving) as if you are among those whom we were not able to civilize"

uygar	"civilized"
+laş	"become"
+tır	"cause to"
+ama	"not able"
+dık	past participle
+lar	plural
+ımız	first person plural possessive ("our")
+dan	ablative case ("from/among")
+mış	past
+sınız	second person plural ("y' all")
+casına	finite verb → adverb ("as if")

Operationalization

- operate (opus/opera + ate)
- ion
- al
- ize
- ate
- ion

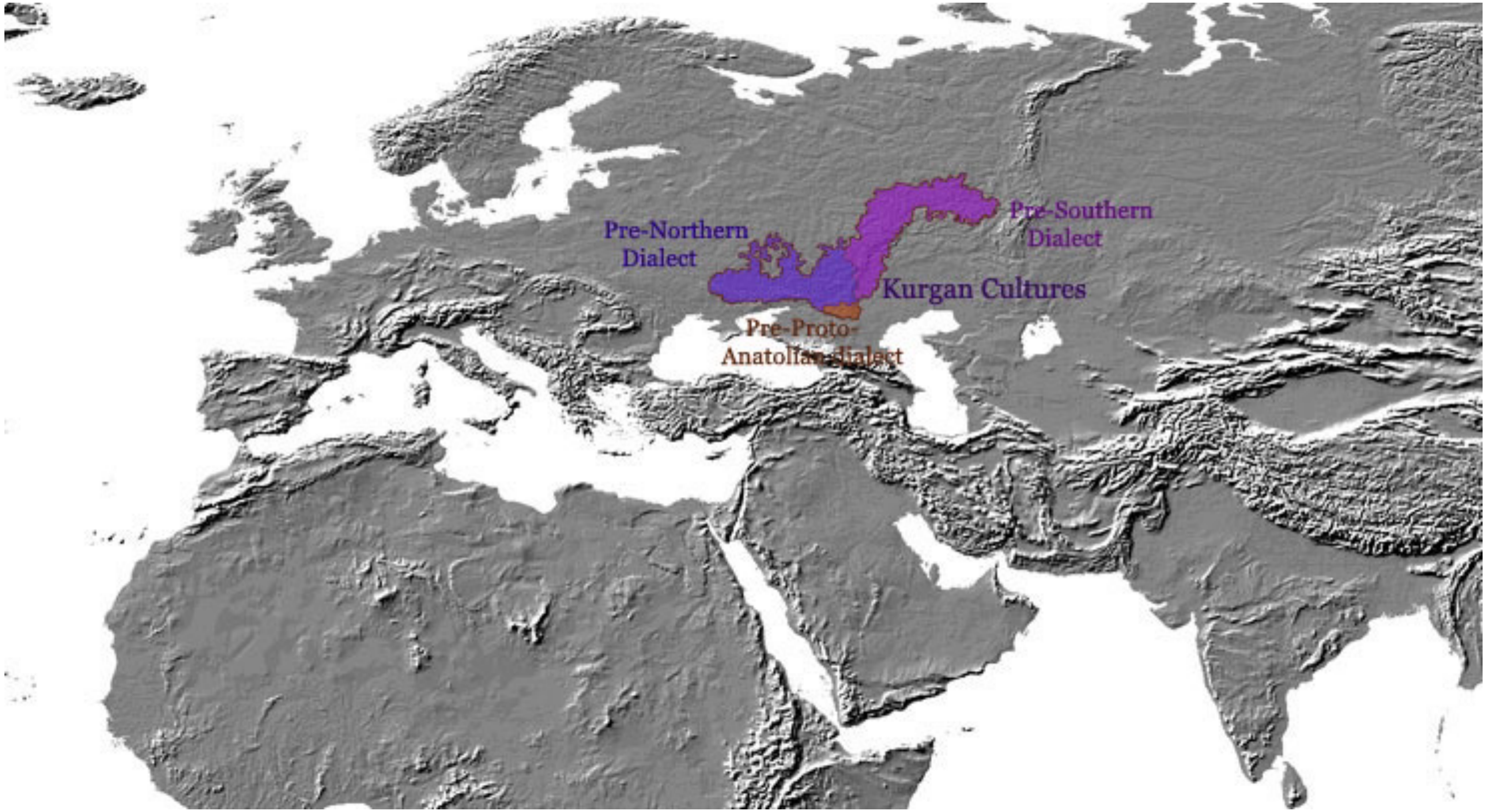
Polysynthetic Languages: Yupik

- Polysynthetic morphologies allow the creation of full "sentences" by morphological means.
- They often allow the incorporation of nouns into verbs.
- They may also have affixes that attach to verbs and take the place of nouns.
- Yupik Eskimo
  untu-ssur-qatar-ni-ksaite-ngqiggte-uq
  reindeer-hunt-FUT-say-NEG-again-3SG.INDIC
  'He had not yet said again that he was going to hunt reindeer.'

Fusional Languages: Spanish

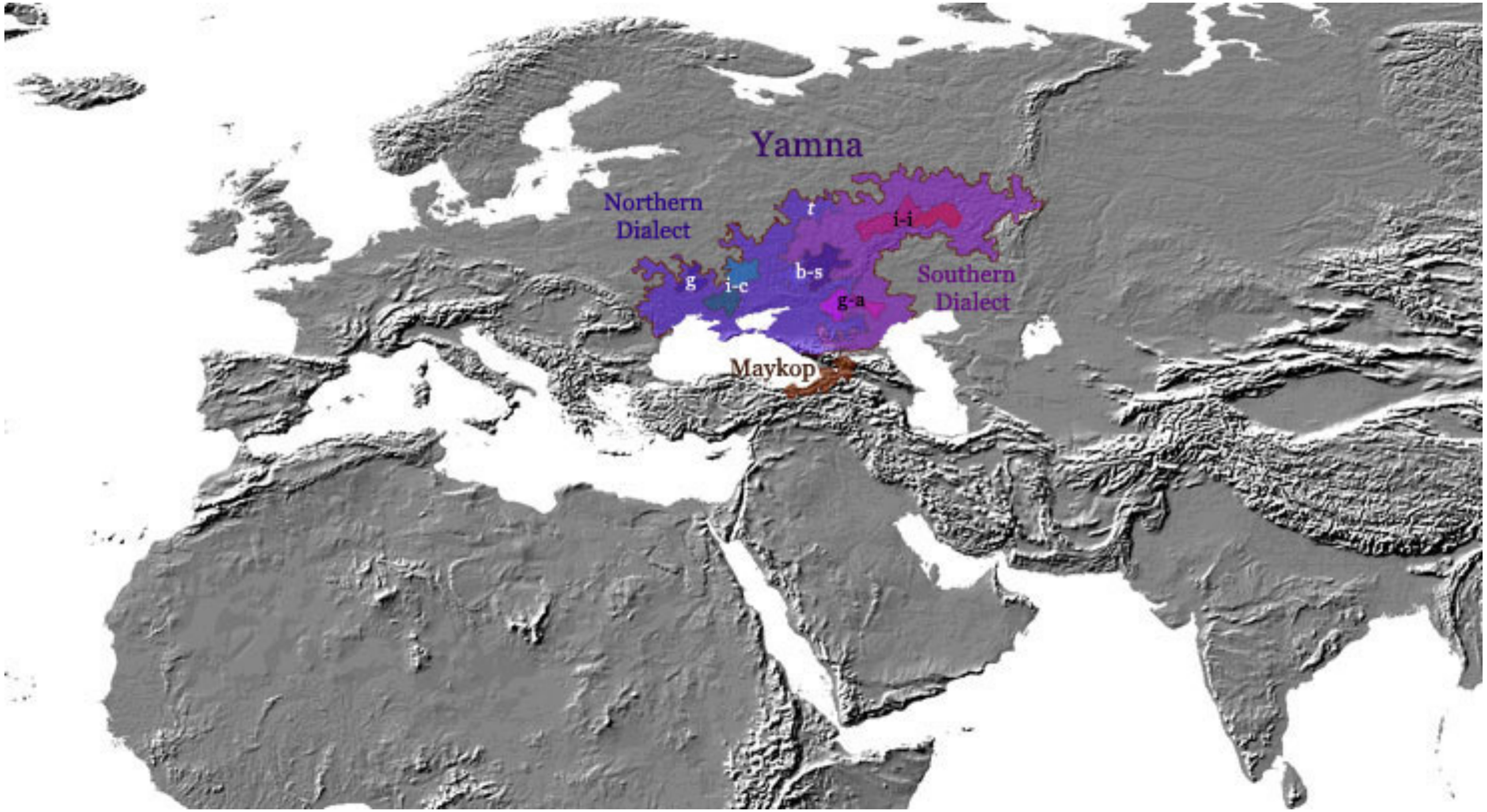
	Singular			Plural		
	1st	2nd	3rd formal 2nd	1st	2nd	3rd
Present	*am-o*	*am-as*	am-a	*am-a-mos*	*am-áis*	*am-an*
Imperfect	*am-ab-a*	*am-ab-as*	*am-ab-a*	*am-áb-a-mos*	*am-ab-ais*	*am-ab-an*
Preterit	*am-é*	*am-aste*	*am-ó*	*am-a-mos*	*am-asteis*	*am-aron*
Future	*am-aré*	*am-arás*	*am-ará*	*am-are-mos*	*am-aréis*	*am-arán*
Conditional	*am-aría*	*am-arías*	*am-aría*	*am-aría-mos*	*am-aríais*	*am-arían*

Indo-European: 4000BC

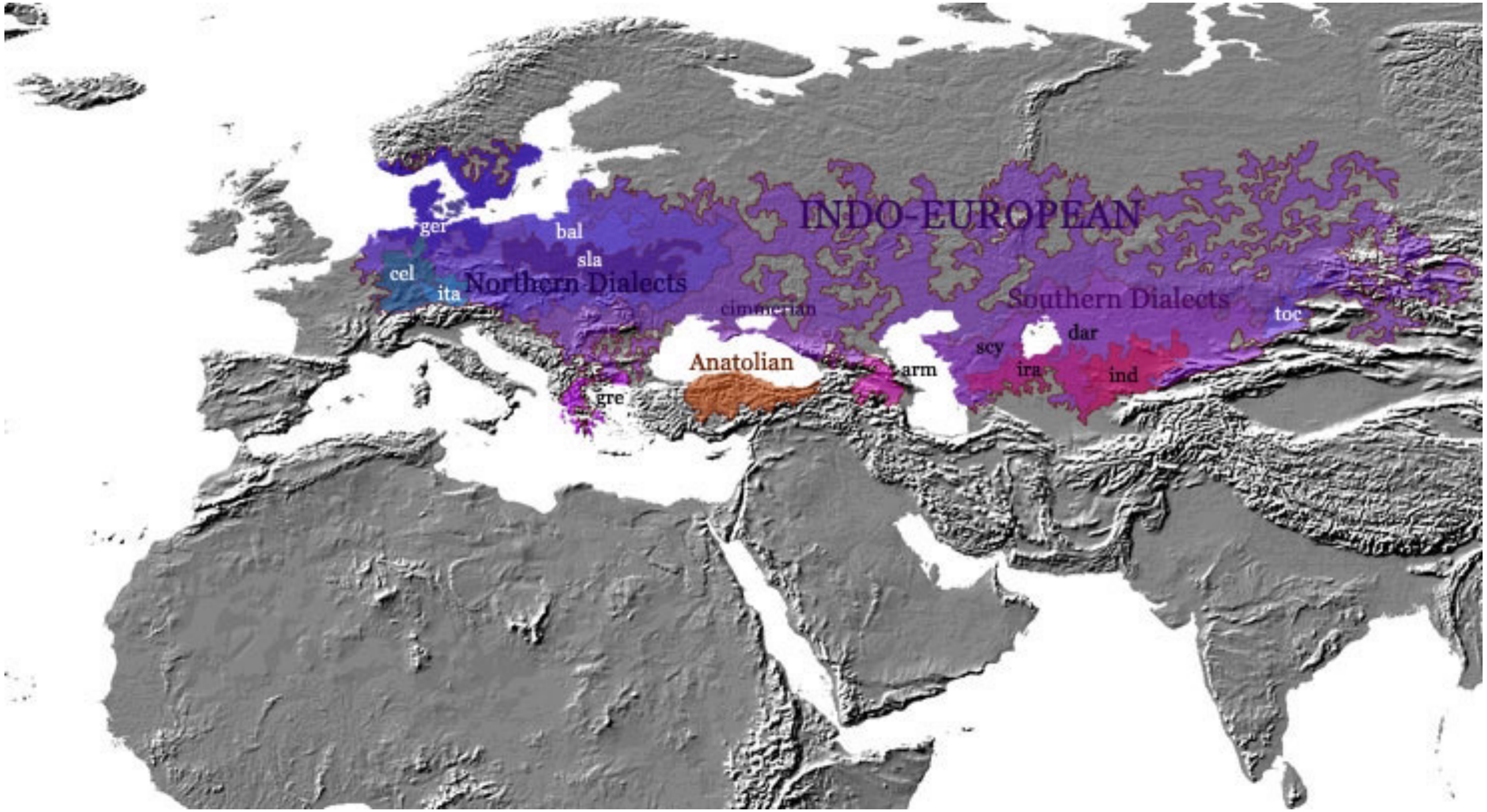


From Wikipedia

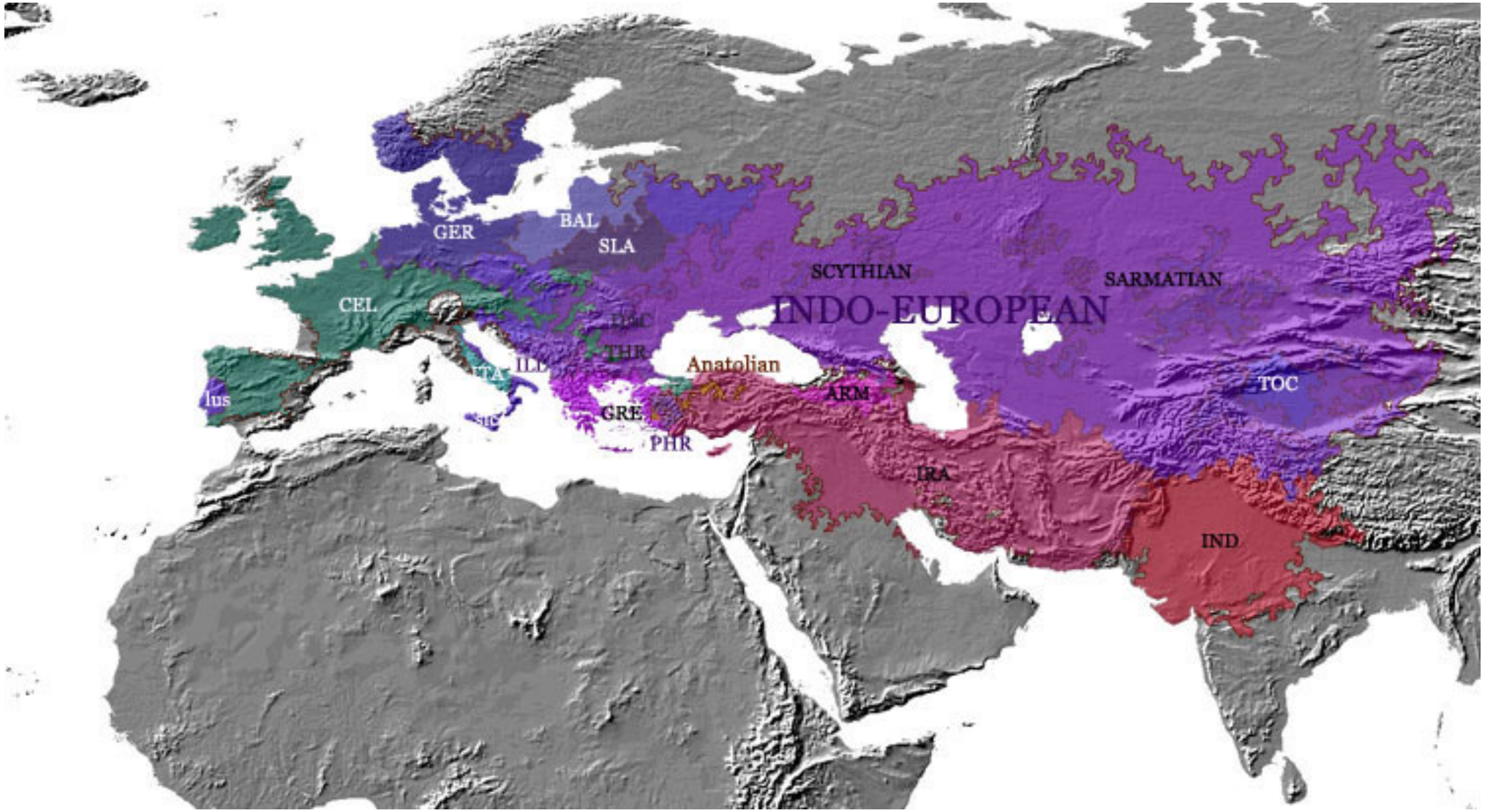
Indo-European: 3000BC



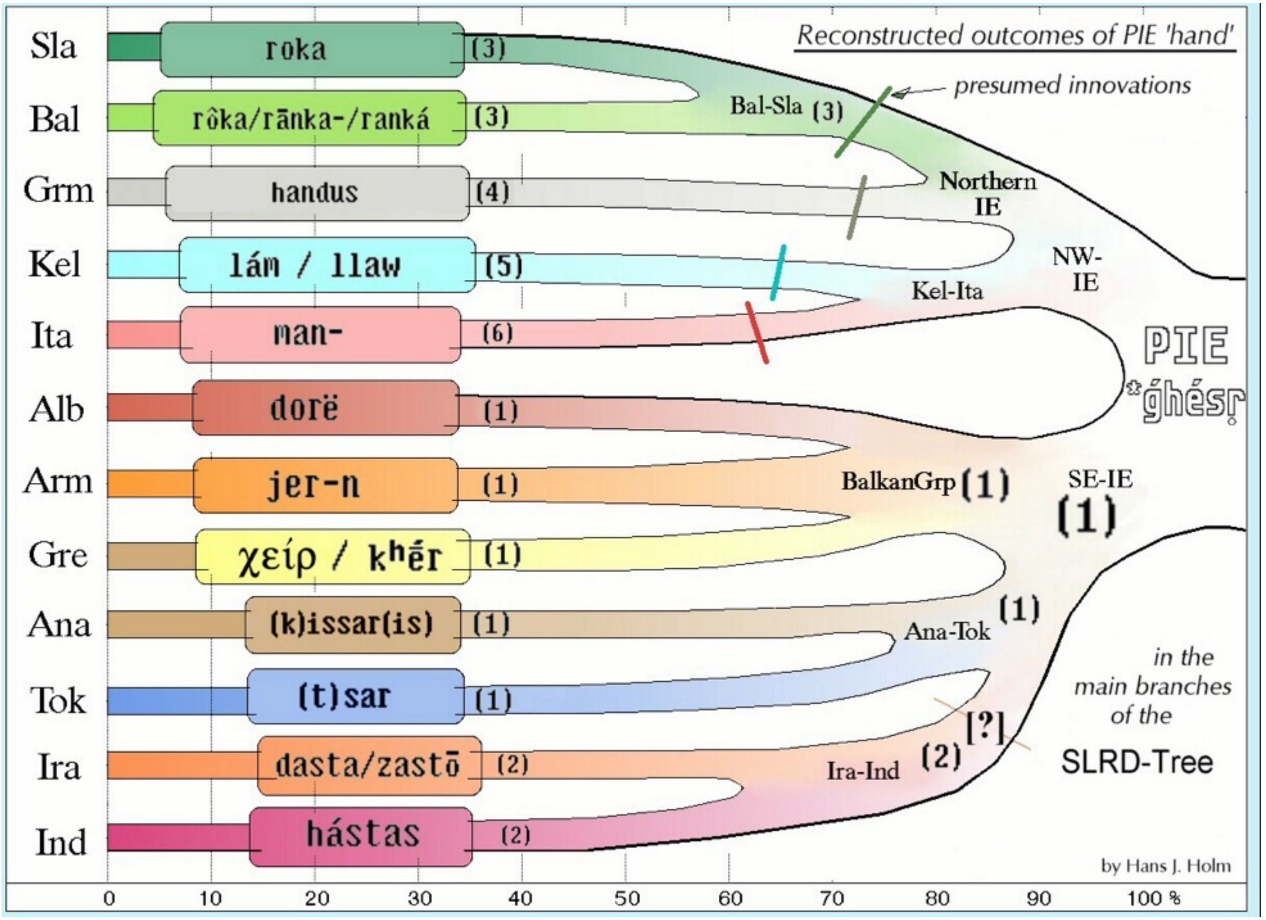
Indo-European: 2000BC



Indo-European: 500BC



Indo-European: “hand”



A Brief History of English

- 900,000 BC?	Humans invade British Isles
- 800 BC?	Celts invade *(Gaelic) [first Indo-Europeans there]*
- 40 AD	Romans invade *(Latin)*
- 410 AD	Anglo-Saxons invade *(West German)*
- 790 AD	Vikings invade *(North German)*
- 1066 AD	Normans invade *(Norman French/Latin)*
- The English spend a few hundred years invading rest of British Isles
- A little later, British start invading *everyone* else
  - North America, India, China, …

Root-and-Pattern Morphology: Arabic

- **Root-and-pattern**. A special kind of fusional morphology found in Arabic, Hebrew, and their cousins.
- Root usually consists of a sequence of consonants.
- Words are derived and, to some extent, inflected by patterns of vowels intercalated among the root consonants.
  - **kitaab** ‘book’
  - **kaatib** ‘writer; writing’
  - **maktab** ‘office; desk’
  - **maktaba** ‘library’

Other Non-Concatenative Morphological Processes

Non-concatenative morphology involves operations other than the concatenation of affixes with bases.

- Infixation. A morpheme is inserted inside another morpheme instead of before or after it.
- Reduplication. Can be prefixing, suffixing, and even infixing.
    - **Tagalog**:
        - sulat (write, imperative)
        - susulat (reduplication) (write, future)
        - sumulat (infixing) (write, past)
        - sumusulat (infixing and reduplication) (write, present)
- **Apophony**, including the umlaut in English *tooth* → *teeth*; **subtractive morphology**, including the truncation in English nickname formation (*David* → *Dave*); and so on.
- Tone change; stress shift. And more...

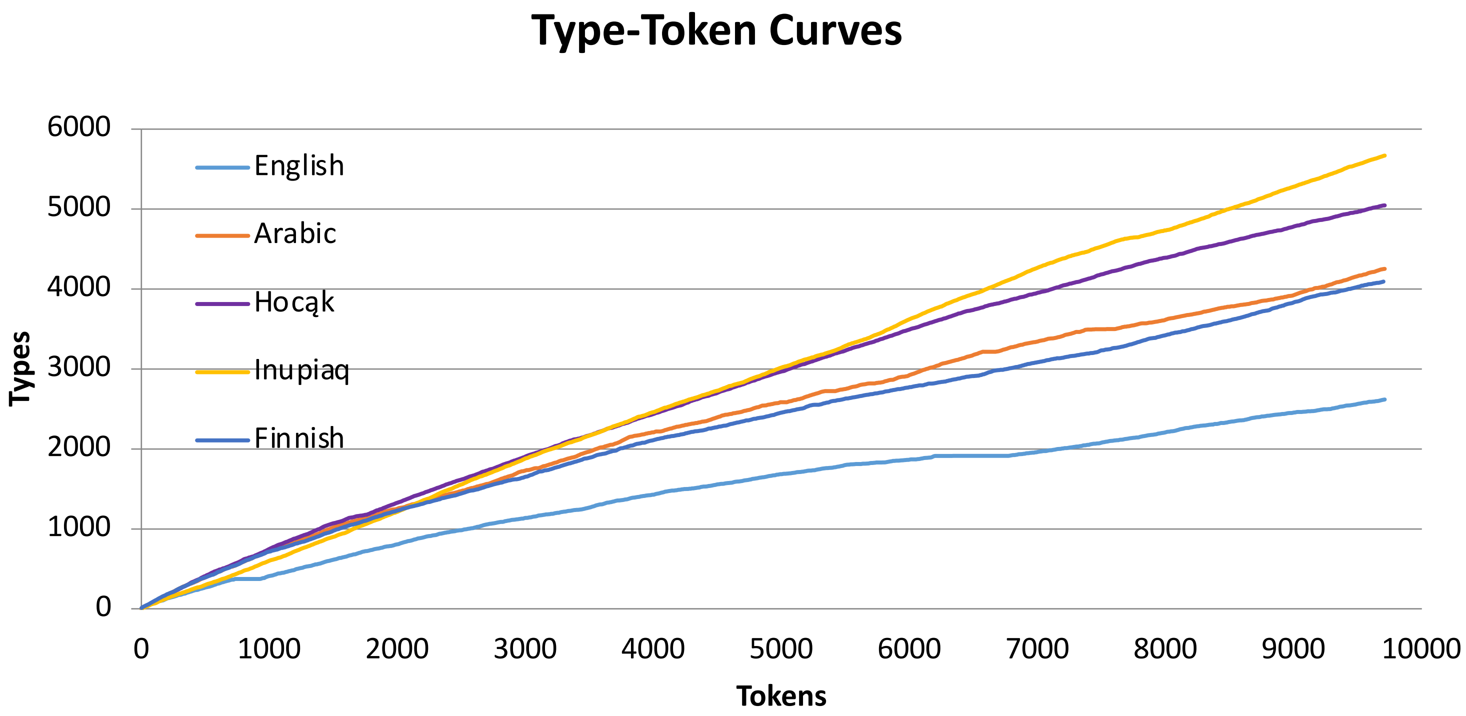
Type-Token Curves

Finnish is agglutinative
Iñupiaq is polysynthetic

Types and Tokens:
"I like to walk.   I am walking now. I took a long walk earlier too."

The type *walk* occurs twice. So there are two tokens of the type *walk*.

Walking is a different type that occurs once.



Morphological Processing

Recognizing the words of a language

- Input: a string (from some alphabet)
- Output: is it a legal word? (yes or no)

Morphology information sources

- Lexicon plus list of affixes
- Morphotactics: rules for how morphemes combine
- Spelling/pronunciation rules

fox+Pl	goose+Pl	*stem+feature*
fox –s (or fox^s)	geese	*morpheme sequence*
foxes	geese	*surface form*

FSA for English Noun inflections

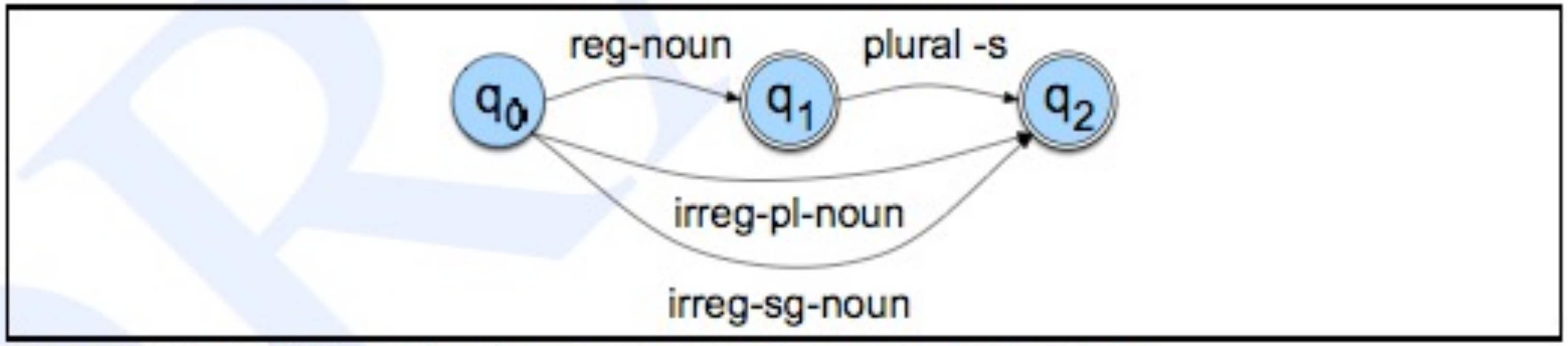


Figure 3.3 A finite-state automaton for English nominal inflection.

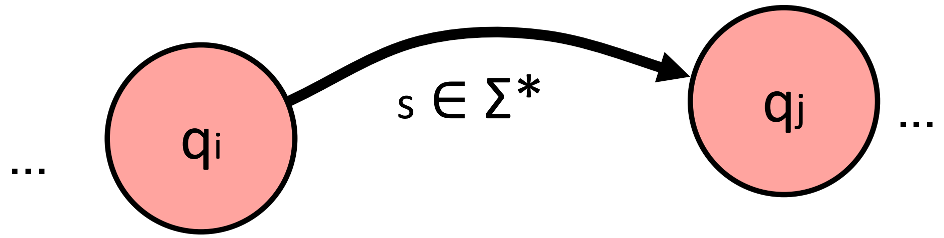
Lexicon:

reg-noun	irreg-pl-noun	irreg-sg-noun	plural
fox	geese	goose	-s
cat	sheep	sheep	
aardvark	mice	mouse	

Note: “fox” becomes plural by adding “es” not “s”. We will get to that later.

Finite-State Automaton

- Q: a finite set of states
- $q_0 \in Q$: a special start state
- $F \subseteq Q$: a set of final states
- Σ: a finite alphabet
- Transitions:



- Encodes a **set** of strings that can be recognized by following paths from q_0 to some state in F.

FSA for English Adjective derivations

Big, bigger, biggest
Happy, happier, happiest, happily
Unhappy, unhappier, unhappiest, unhappily
Clear, clearer, clearest, clearly
Unclear, unclearly

Cool, cooler, coolest, coolly
Red, redder, reddest
Real, unreal, really

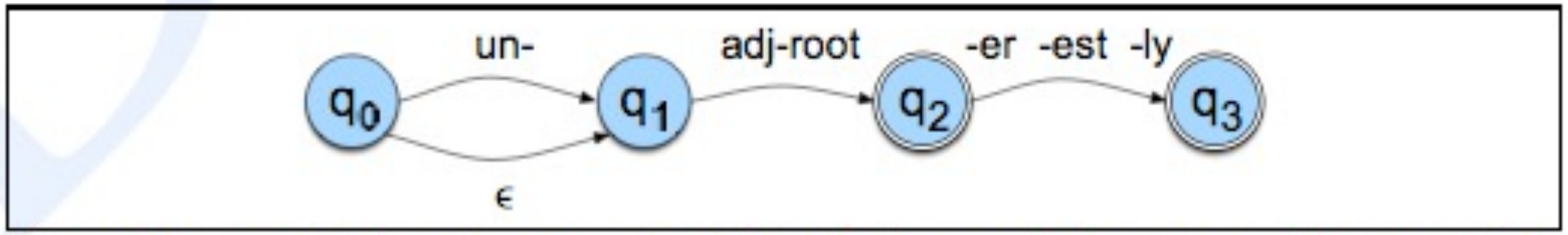


Figure 3.5 An FSA for a fragment of English adjective morphology: Antworth's Proposal #1.

But note that this accepts words like "unbig".

FSA for English Derivational Morphology

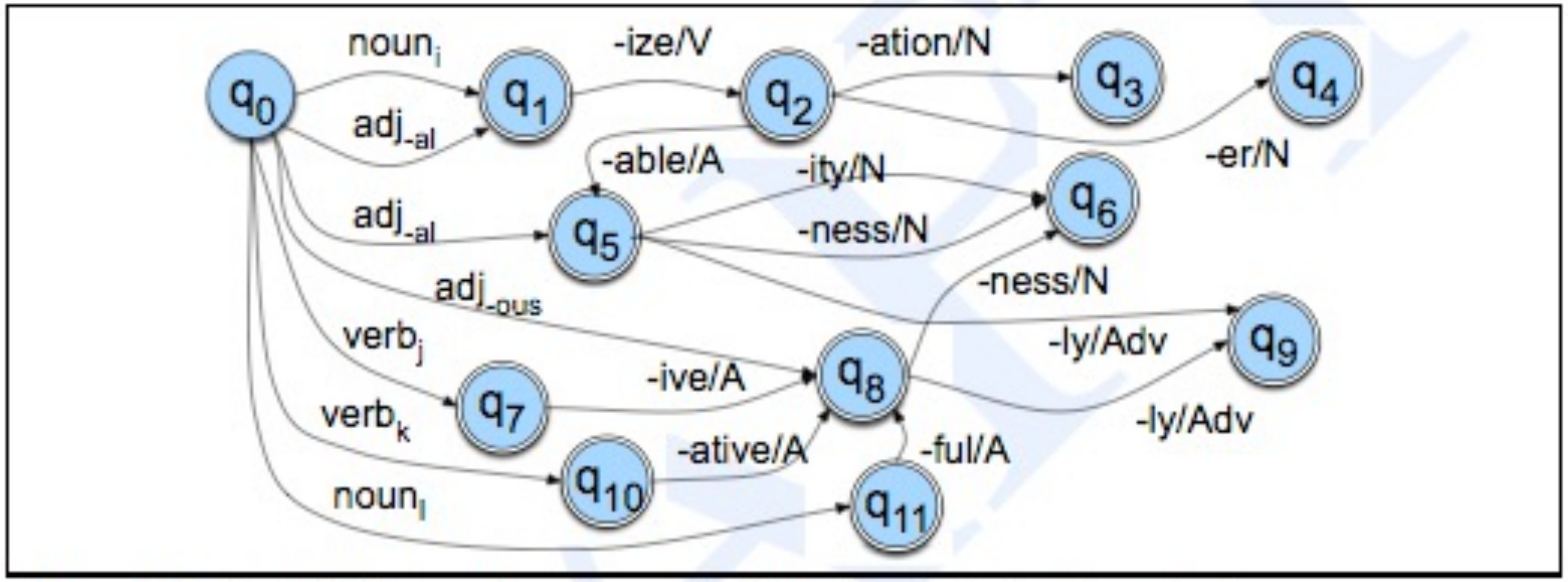


Figure 3.6 An FSA for another fragment of English derivational morphology.

How big do these automata get? Reasonable coverage of a language takes an expert about two to four months.

What does it take to be an expert? Study linguistics to get used to all the common and not-so-common things that happen, and then practice.

Morphological *Parsing*

Input: a word

Output: the word's **stem**(s) and **features** expressed by other morphemes.

Example: geese → goose +N +Pl

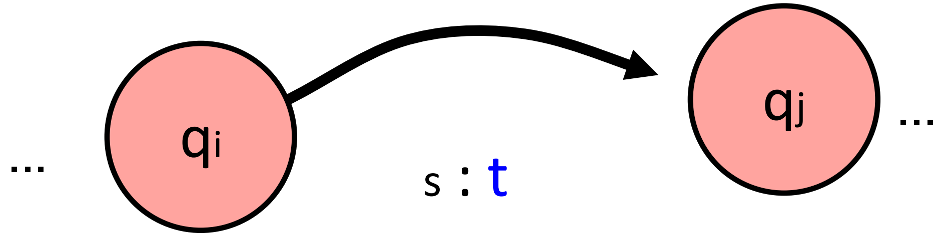
gooses → goose +V +3P +Sg

dog → {dog +N +Sg, dog +V}

leaves → {leaf +N +Pl, leave +V +3P +Sg}

Finite State Transducers

- Q: a finite set of states
- $q_0 \in Q$: a special start state
- $F \subseteq Q$: a set of final states
- Σ and Δ: two finite alphabets
- Transitions:

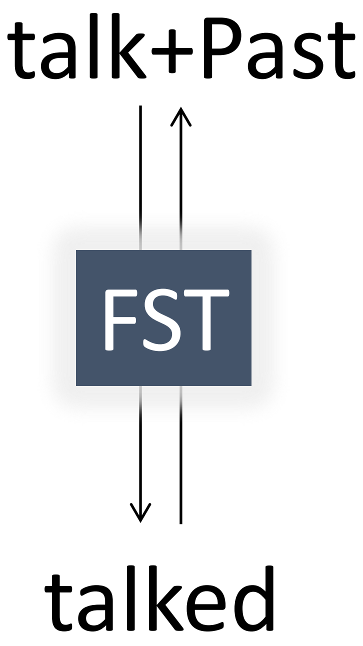


$s \in \Sigma^*$ and $t \in \Delta^*$

Two-level Morphology

upper side or underlying form

lower side or surface form



Morphological Parsing with FSTs

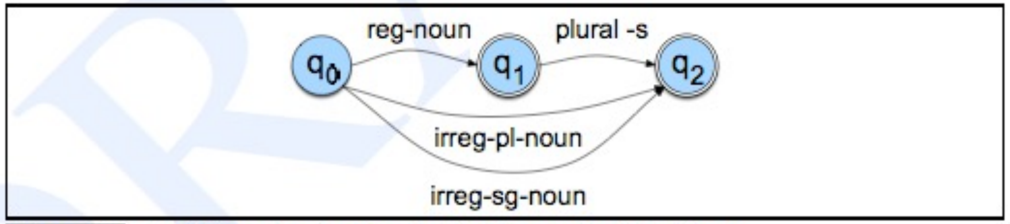


Figure 3.3 A finite-state automaton for English nominal inflection.

reg-noun	**irreg-pl-noun**	**irreg-sg-noun**	**plural**
fox	geese	goose	-s
cat	sheep	sheep	
aardvark	mice	mouse	

reg-noun	**irreg-pl-noun**	**irreg-sg-noun**
fox	g o:e o:e s e	goose
cat	sheep	sheep
aardvark	m o:i u:ϵ s:c e	mouse

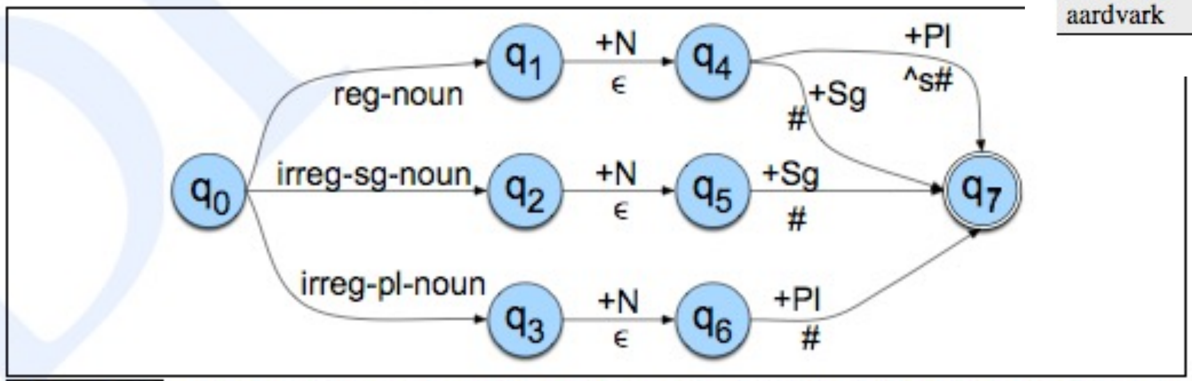


Figure 3.13 A schematic transducer for English nominal number inflection T_{num}. The symbols above each arc represent elements of the morphological parse in the lexical tape; the symbols below each arc represent the surface tape (or the intermediate tape, to be described later), using the morpheme-boundary symbol ^ and word-boundary marker #. The labels on the arcs leaving q_0 are schematic, and need to be expanded by individual words in the lexicon.

Note "same symbol" shorthand.

^ denotes a morpheme boundary.

# denotes a word boundary.

English Spelling

Getting back to fox+s = foxes

Name	Description of Rule	Example
Consonant doubling	1-letter consonant doubled before *-ing*/*-ed*	beg/begging
E deletion	Silent e dropped before *-ing* and *-ed*	make/making
E insertion	e added after *-s*,*-z*,*-x*,*-ch*, *-sh* before *-s*	watch/watches
Y replacement	*-y* changes to *-ie* before *-s*, *-i* before *-ed*	try/tries
K insertion	verbs ending with *vowel* + *-c* add *-k*	panic/panicked

The E Insertion Rule as a FST

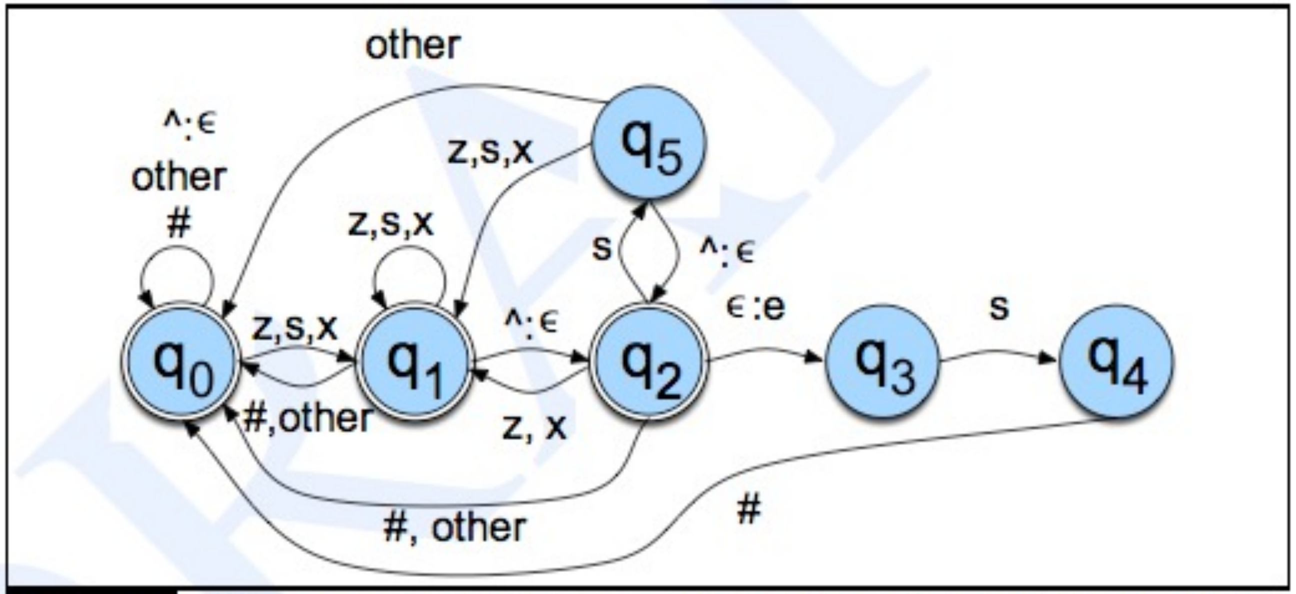


Figure 3.17 The transducer for the E-insertion rule of (3.4), extended from a similar transducer in Antworth (1990). We additionally need to delete the # symbol from the surface string; this can be done either by interpreting the symbol # as the pair #:ϵ, or by postprocessing the output to remove word boundaries.

Generate a normally spelled word from an abstract representation of the morphemes:

Input: fox^s# (fox^εs#)
Output: foxes# (foxεes#)

$$\epsilon \rightarrow e / \left\{ \begin{array}{c} s \\ x \\ z \end{array} \right\} \wedge \_s\#$$

The E Insertion Rule as a FST

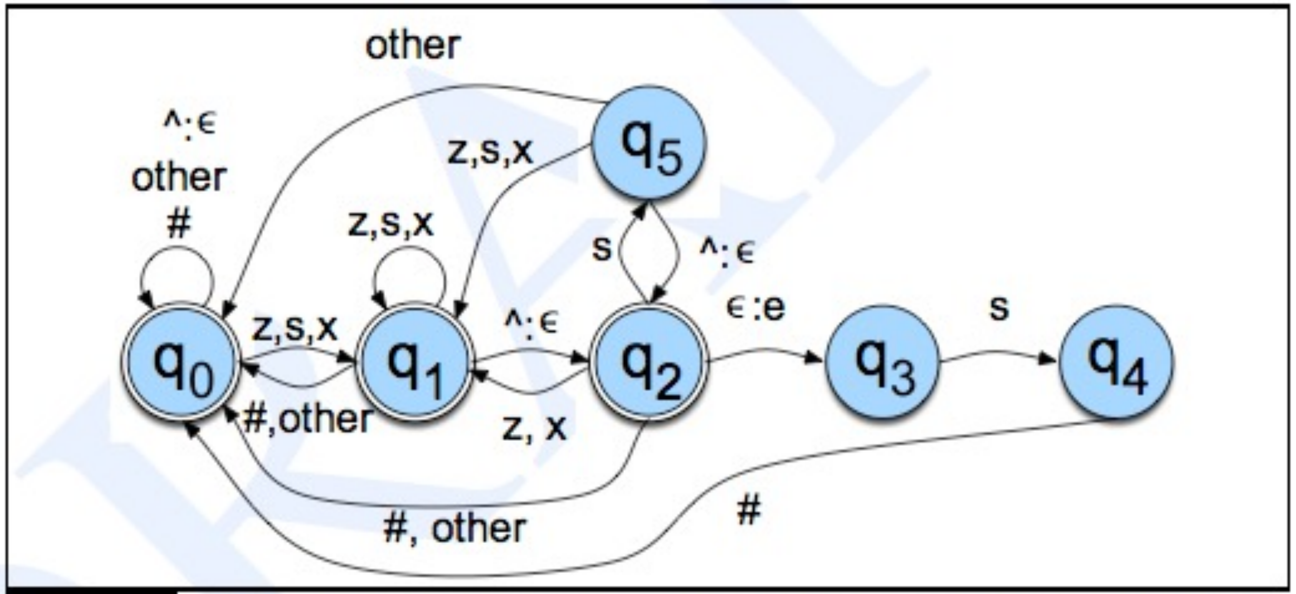


Figure 3.17 The transducer for the E-insertion rule of (3.4), extended from a similar transducer in Antworth (1990). We additionally need to delete the # symbol from the surface string; this can be done either by interpreting the symbol # as the pair #:ϵ, or by postprocessing the output to remove word boundaries.

Parse a normally spelled word into an abstract representation of the morphemes:

Input: foxes# (foxεes#)
Output: fox^s# (fox^εs#)

$$\epsilon \rightarrow e / \left\{ \begin{array}{c} s \\ x \\ z \end{array} \right\} \wedge \_s\#$$

Combining FSTs

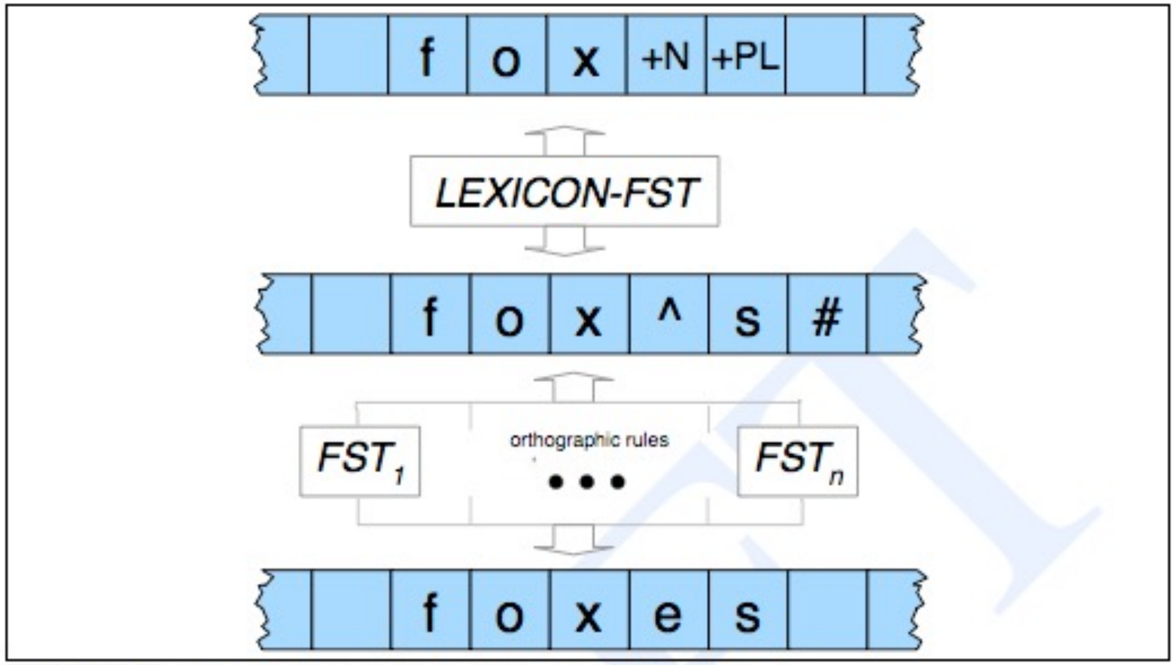


Figure 3.19 Generating or parsing with FST lexicon and rules

parse

generate

FST Operations

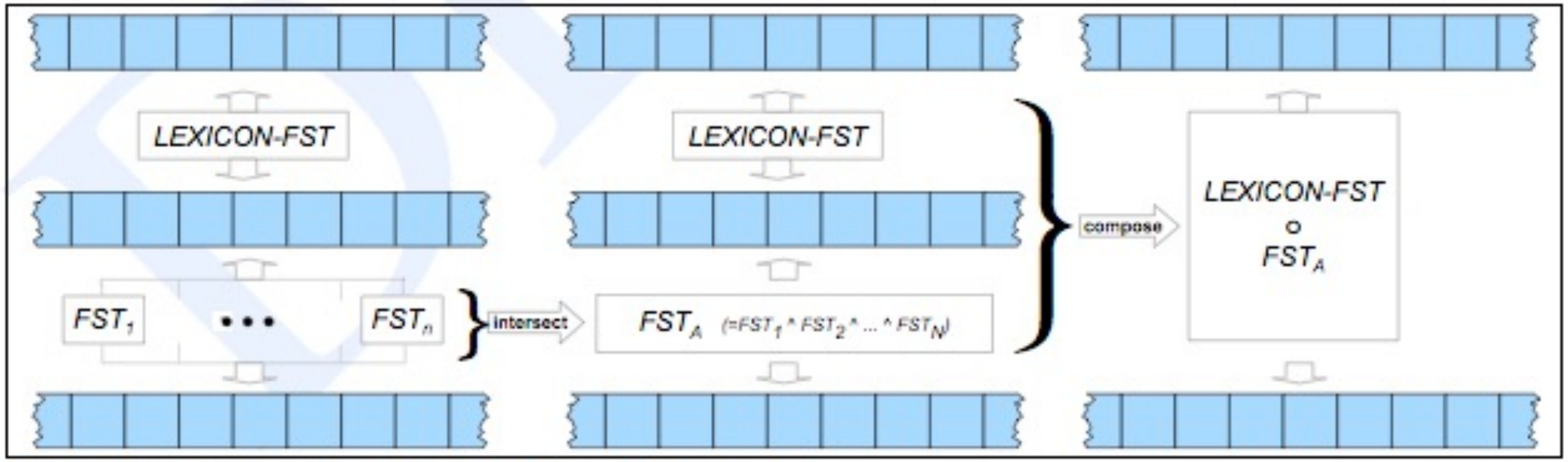


Figure 3.21 Intersection and composition of transducers.

Input: fox +N +pl
Output: foxes#

Language Type Comparison wrt FSTs

- Morphologies of all types can be analyzed using finite state methods.
- Some present more challenges than others:
  - **Analytic languages**. Trivial, since there is little or no morphology (other than compounding).
  - **Agglutinating languages**. Straightforward—finite state morphology was “made” for languages like this.
  - **Polysynthetic languages**. Similar to agglutinating languages, but with blurred lines between morphology and syntax.
  - **Fusional languages.** Easy enough to analyze using finite state method as long as one allows “morphemes” to have lots of simultaneous meanings and one is willing to employ some additional tricks.
  - **Root-and-pattern languages.** Require some very clever tricks.

The Good News

- More than almost any other problem in computational linguistics, morphology is a solved problem (as long as you can afford to write rules by hand).
- Finite state methods provide a simple and powerful means of generating and analyzing words (as well as the phonological alternations that accompany word formation/inflection).
- Finite state morphology is one of the great successes of natural language processing.
- One brilliant aspect of using FSTs for morphology: the same code can handle both analysis and generation.

Stemming ("Poor Man's Morphology")

Input: a word

Output: the word's stem (approximately)

Examples from the Porter stemmer:

- -sses → -ss
- -ies → i
- -ss → s

no	no
noah	noah
nob	nob
nobility	nobil
nobis	nobi
noble	nobl
nobleman	nobleman
noblemen	noblemen
nobleness	nobl
nobler	nobler
nobles	nobl
noblesse	nobless
noblest	noblest
nobly	nobli
nobody	nobodi
noces	noce
nod	nod
nodded	nod
nodding	nod
noddle	noddl
noddles	noddl
noddy	noddi
nods	nod

Two major classes of approaches

- Linguistic approaches:
  - Segmenting into words that make sense with grammars/meanings
  - Segmenting into subword units that make sense with grammars/meanings

- **Technological approaches:**
  - Segmenting into words to make processing efficient/better
  - Segmenting into subwords to make processing efficient/better

We will look at both; linguistic approaches matter more if later stages involve parsing and/or semantics

Subword segmentation: motivation:

- Neural systems typically use a fixed vocabulary
  - Preferably a relatively small fixed vocabulary
- Real world contains very many words
  - New words all the time: *doomscrolling, quarenteen*, …
  - For morphologically rich languages, even more so
  - But most of them are rare (Zipf's Law)
- Note that infrequent words do not have good corpus statistics

- So, fix the size of vocabulary, start with single characters, and learn most frequent words, and useful subword segments for the rest

Unsupervised subword segmentation

Instead of

- white-space segmentation
- single-character segmentation

Use the data to tell us how to tokenize.

Subword segmentation/tokenization (because tokens can be parts of words as well as whole words)

Subword tokenization

- Three common algorithms:
  - **Byte-Pair Encoding (BPE)** (Sennrich et al., 2016)
  - **Unigram language modeling tokenization** (Kudo, 2018)
  - **WordPiece** (Schuster and Nakajima, 2012)

- All have 2 parts:
  - A token **learner** that takes a raw training corpus and induces a vocabulary (a set of tokens).
  - A token **segmenter** that takes a raw test sentence and tokenizes it according to that vocabulary.

Byte Pair Encoding (BPE) token learner

Let vocabulary be the set of all individual characters

= {A, B, C, D,…, a, b, c, d….}

- Repeat:
  - Choose the two symbols that are most frequently adjacent in the training corpus (say 'A', 'B')
  - Add a new merged symbol 'AB' to the vocabulary
  - Replace every adjacent 'A' 'B' in the corpus with 'AB'.
- Until k merges have been done. *(Or until you hit vocabulary size.)*

BPE token learner algorithm

function BYTE-PAIR ENCODING(strings C, number of merges k) **returns** vocab V

$V \leftarrow$ all unique characters in C # initial set of tokens is characters
for $i = 1$ **to** k **do** # merge tokens til k times
  $t_L, t_R \leftarrow$ Most frequent pair of adjacent tokens in C
  $t_{NEW} \leftarrow t_L + t_R$ # make new token by concatenating
  $V \leftarrow V + t_{NEW}$ # update the vocabulary
  Replace each occurrence of t_L, t_R in C with t_{NEW} # and update the corpus
return V

Byte Pair Encoding (BPE) Addendum

Most subword algorithms are run inside space-separated tokens.

So we commonly first add a special end-of-word symbol '__' before space in training corpus

Next, separate into letters.

BPE token **learner**

Original (very fascinating🙄) corpus:

low low low low low lowest lowest newer newer newer newer newer newer wider wider wider new new

Add end-of-word tokens, resulting in this vocabulary:

vocabulary

_, d, e, i, l, n, o, r, s, t, w

BPE token learner

corpus

```
5    l o w _
2    l o w e s t _
6    n e w e r _
3    w i d e r _
2    n e w _
```

vocabulary

```
_, d, e, i, l, n, o, r, s, t, w
```

Merge e r to er

corpus

```
5    l o w _
2    l o w e s t _
6    n e w er _
3    w i d er _
2    n e w _
```

vocabulary

```
_, d, e, i, l, n, o, r, s, t, w, er
```

BPE

corpus

```
5   l o w _
2   l o w e s t _
6   n e w er _
3   w i d er _
2   n e w _
```

vocabulary

_, d, e, i, l, n, o, r, s, t, w, er

Merge er _ to er_

corpus

```
5   l o w _
2   l o w e s t _
6   n e w er_
3   w i d er_
2   n e w _
```

vocabulary

_, d, e, i, l, n, o, r, s, t, w, er, er_

BPE

corpus		**vocabulary**
5	l o w _	_, d, e, i, l, n, o, r, s, t, w, er, er_
2	l o w e s t _	
6	n e w er_	
3	w i d er_	
2	n e w _	

Merge n e to ne

corpus		**vocabulary**
5	l o w _	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne
2	l o w e s t _	
6	ne w er_	
3	w i d er_	
2	ne w _	

BPE

The next merges are:

Merge	**Current Vocabulary**
(ne, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new
(l, o)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo
(lo, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low
(new, er_)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_
(low, _)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_, low_

BPE token **segmenter** algorithm

On the test data, run each merge learned from the training data:

- Greedily
- In the order we learned them
- (test frequencies don't play a role)

So: merge every e r to er, then merge er _ to er_, etc.

- Result:
  - Test set "n e w e r _" would be tokenized as a full word
  - Test set "l o w e r _" would be two tokens: "low er_"

Properties of BPE tokens

Usually include frequent words

And frequent subwords

- Which are often morphemes like *-est* or *–er*

Questions?

Types of Lexical and Morphological Processing

- Tokenization
  - Input: raw text
  - Output: sequence of **tokens** normalized for further processing

- Recognition
  - Input: a string of characters
  - Output: is it a legal word? (yes or no)
- Morphological Parsing
  - Input: a word
  - Output: an analysis of the structure of the word

- Morphological Generation
  - Input: an analysis of the structure of the word
  - Output: a word

Conclusion

- Finite state methods provide a simple and powerful means of generating and analyzing words (as well as the phonological alternations that accompany word formation/inflection).
- Straightforward concatenative morphology is easy to implement using finite state methods.
- Other phenomena are easiest to capture with extensions to the finite state paradigm.
  - Co-occurrence restrictions—flag diacritics.
  - Non-concatenative morphology—compile-replace algorithm. Pure finite state, but computed in a novel fashion.

Tools

- There are special finite state toolkits for building morphological tools (and other linguistic tools).
- The best-known of these is the **Xerox Finite State Tool** or **XFST**, which originated at Xerox PARC.
- There are open source reimplementations of XFST called **HFST** (Helsinki Finite State Technology) and **Foma**, which are not as fully optimized as XFST but which are sometimes more pleasant to use.
- None of these tools allow the construction of weighted FSTs.

Can you make a list of all the words in a language?

Productivity

In the Oxford English Dictionary (OED)

(www.oed.com, accessible for free from CMU machines)

- drinkable
- visitable

Not in the OED

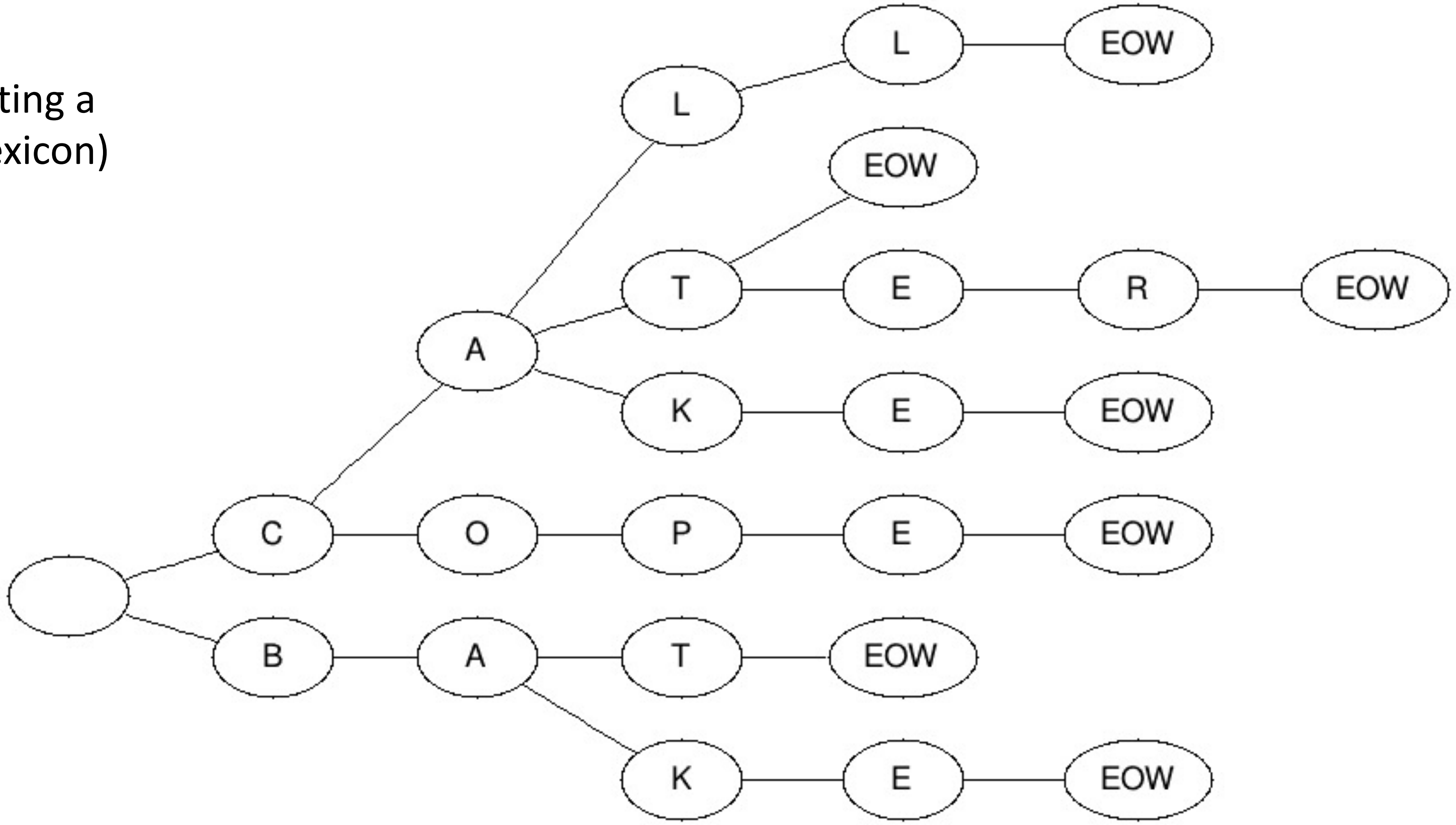
- mous(e)able
- stapl(e)able

In NLP, you need to be able to process words that are not in the dictionary.

But could you make a list of all possible words, taking productivity into account?

Can you make a list of all the words in a language?

A trie representing a list of words (lexicon)



Telugu, Tamil, Kannada, Malayalam

Dravidian languages

- Agglutinating like Turkish, Finnish, and Swahili

Hindi, Urdu, Bengali, Marathi, Punjabi, etc. Indo-european

- A little richer than English
- Like English, uses auxiliary verbs and separate words to express things that are affixes on the verbs in Dravidian languages.
  - want, have, be, make, etc.

Mapudungun compared to Spanish

Mapudungun is polysynthetic

Spanish is fusional