Words and Transducers (Chapter 3)

- English Morphology
- Stemming: Normalizing the words
- Tokenizing: Getting the words (or word-like elements)
- Segmentation: Getting sentences
- Edit distance

English Morphology

- Morphology is the study of the ways that words are built up from smaller meaningful units called morphemes
- We can usefully divide morphemes into two categories
  - Stems: The core meaning-bearing units
  - Affixes: Bits and pieces that adhere to stems to change their meanings and grammatical functions
  - E.g., cat → cats

English Morphology

- We can further divide morphology up into two broad classes
  - Inflectional
  - Derivational
- Word Classes
  - By word class, we have in mind familiar notions like noun and verb
  - We'll go into the details in Chapter 5
  - Right now we're concerned with word classes because the way that stems and affixes combine is based to a large degree on the word class of the stem

Inflectional Morphology

- Inflectional morphology concerns the combination of stems and affixes where the resulting word:
  - Has the same word class as the original
  - Nouns are simple
    - Markers for plural and possessive
    - E.g. table, tables
  - Verbs are only slightly more complex
    - Markers appropriate to the tense of the verb
    - E.g. Walk, walks, walking

Regulars and Irregulars

- It is a little complicated by the fact that some words misbehave (refuse to follow the rules)
  - Mouse/mice, goose/geese, ox/oxen
  - Go/went, fly/flew
- The terms regular and irregular are used to refer to words that follow the rules and those that don't

Regular and Irregular Verbs

- Regulars…
  - Walk, walks, walking, walked, walked
- Irregulars
  - Catch, catches, catching, caught, caught
  - Cut, cuts, cutting, cut, cut
- So inflectional morphology in English is fairly straightforward
- But is complicated by the fact that are irregularities

Derivational Morphology

- Derivational morphology
  - More complicated.
  - Many paths are possible…
  - Start with compute
    - Computer -> computerize -> computerization
    - Computer -> computerize -> computerizable
  - Meaning change
    - E.g., care -- careless
  - Changes of word class

Derivational Examples

- Nouns and Verbs to Adjectives

-al	computation	computational
-able	embrace	embraceable
-less	clue	clueless

- Verbs and Adjectives to Nouns

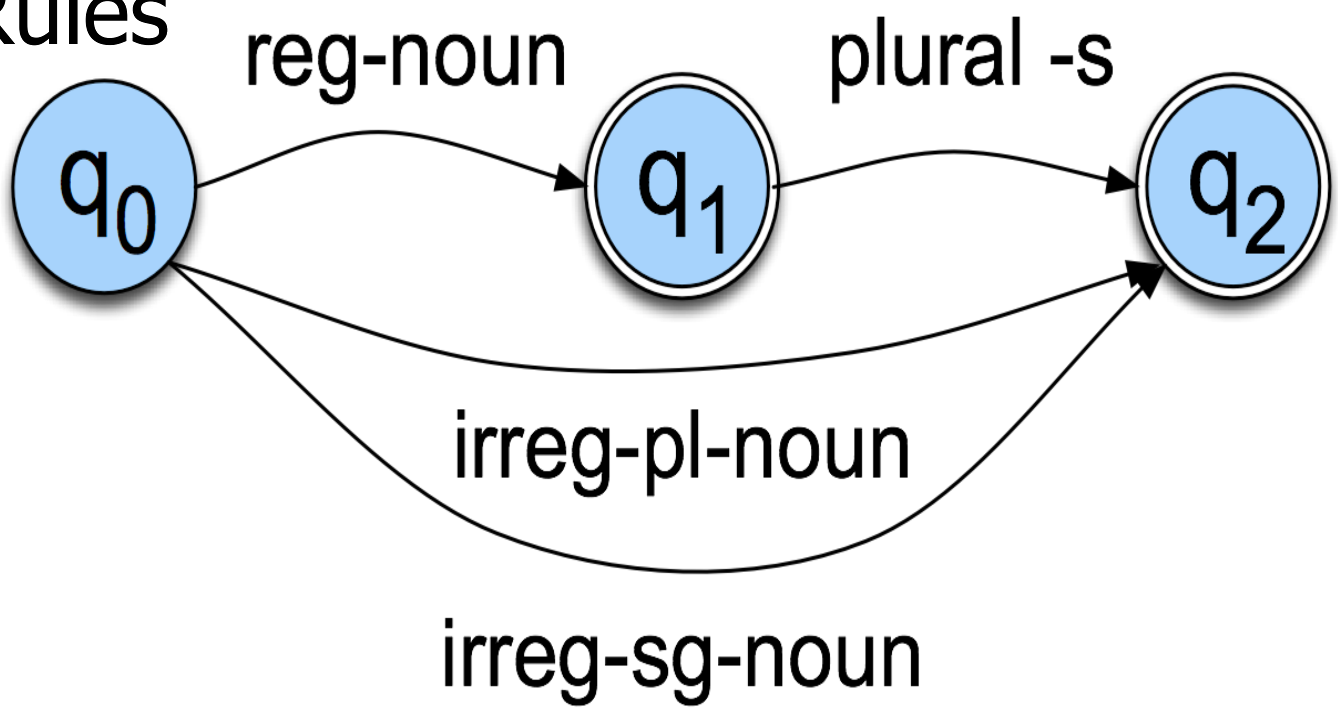
-ation	computerize	computerization
-ee	appoint	appointee
-er	kill	killer
-ness	fuzzy	fuzziness

Morphology and FSAs

- We'd like to use the machinery provided by FSAs to capture these facts about morphology
  - Accept strings that are in the language
  - Reject strings that are not
  - Determine whether an input string of letters make up a legitimate English words
- So that we do not have to list all the words in the language

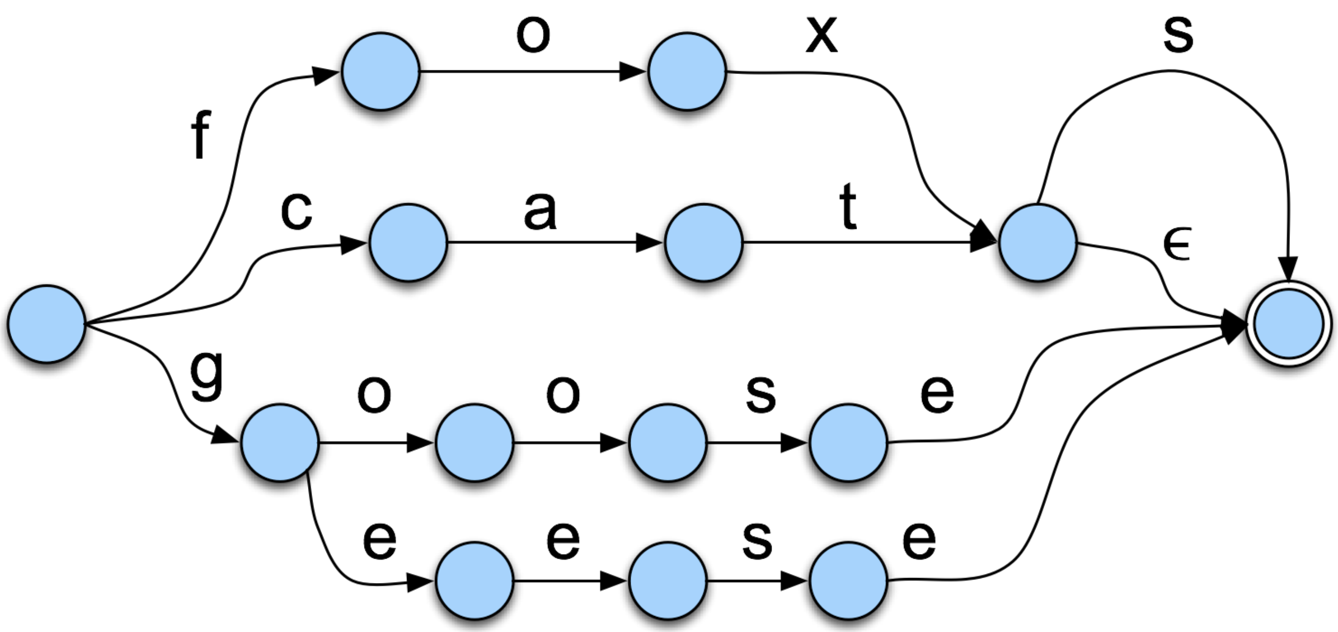
Start Simple

- Regular singular nouns are ok
- Regular plural nouns have an -s on the end
- Irregulars are ok as is
- Simple Rules



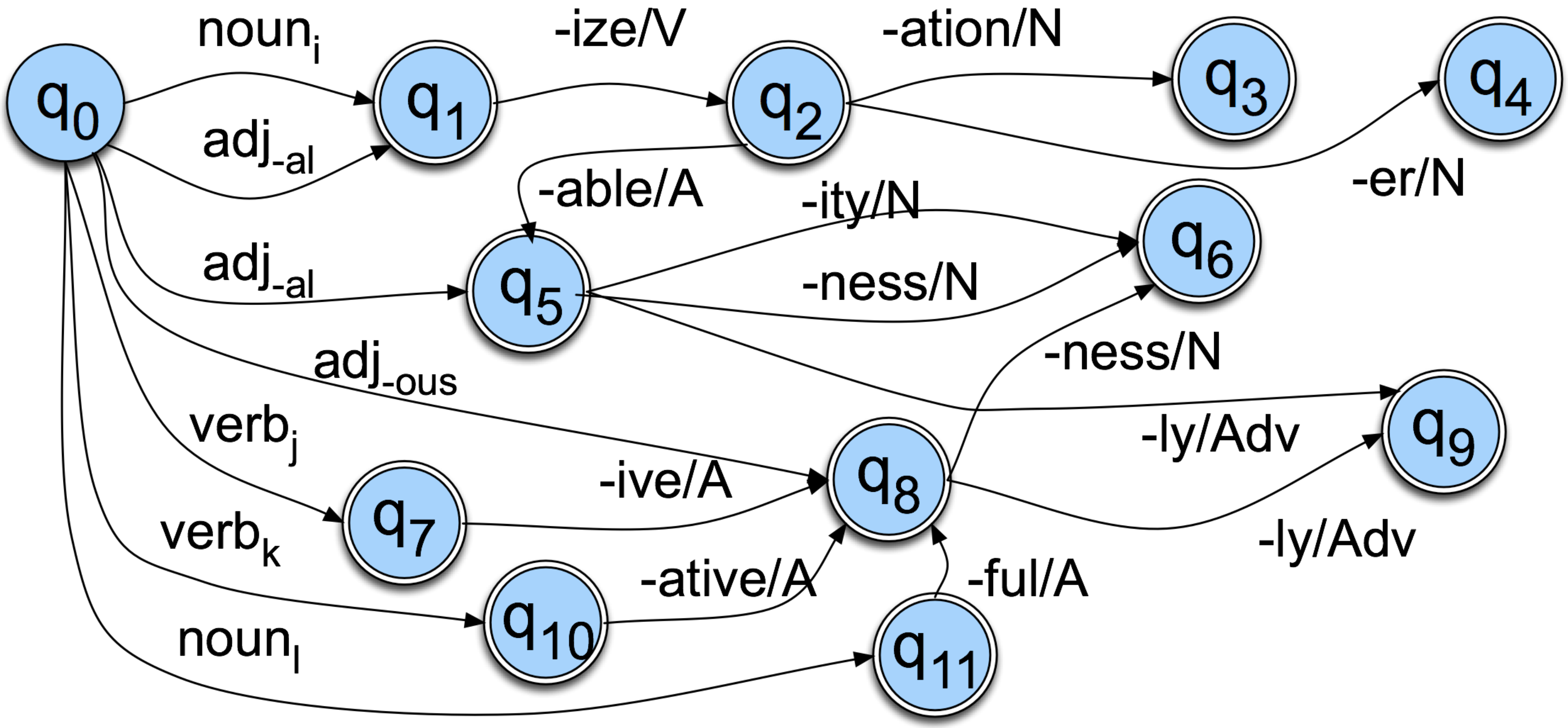
Now Plug in the Words

Replace the class names like "reg-noun" with FSAs that recognize all the words in that class.



Recognize strings, e.g. geese, goat, foxs.

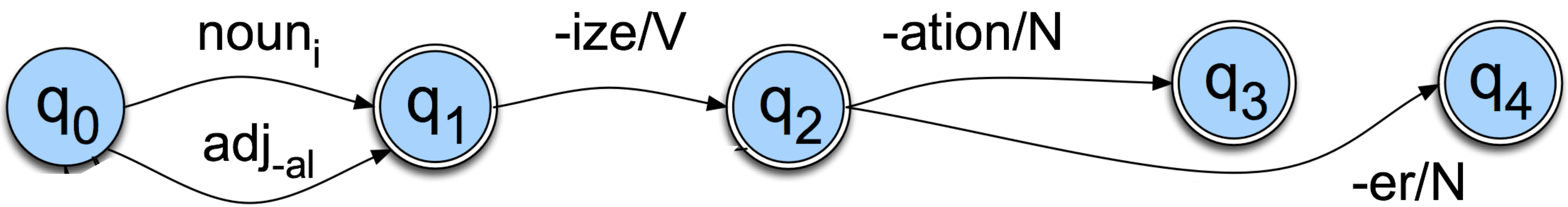
Derivational Rules



If everything is an accept state, how do things ever get rejected?

Exercise: Write a regular expression for the FSA

A|B – A or B
(ABC) – ABC as a component
A? – A is optional



Parsing

- We can now run strings through these machines to recognize strings in the language
  - Spelling checking
- Often if we find some string in the language we might like to assign a structure to it (parsing)
- Example
  - From "cats" to "cat +N +PL"
  - From "caught" to "catch+V+past"
- The kind of parsing we're talking about is normally called morphological analysis

Applications

- It can either be
  - An important stand-alone component of many applications (spelling correction, information retrieval) for complex languages, e.g., Russia
  - Or simply a link in a chain of further linguistic analysis

Light-Weight Morphology

- Sometimes you just need to know the stem of a word and you don't care about the structure.
  - E.g. camera, cameras
- In fact you may not even care if you get the right stem, as long as you get a consistent string--Stemming
  - e.g. Unknown word handling
- Stemming for Information Retrieval
  - Run a stemmer on the documents to be indexed
  - Run a stemmer on users' queries
  - Match to the index

Porter

- No lexicon needed
- Basically a set of staged sets of rewrite rules that strip suffixes
  - ING → ε (e.g., monitoring → monotor)
  - SSES→ SS (e.g., grasses → grass)
- Handles both inflectional and derivational suffixes
- Doesn’t guarantee that the resulting stem is really a stem
  - Lack of guarantee doesn’t matter for IR

Porter

- More Example (recursive)
  - Computerization
    - ization -> -ize computerize
    - ize -> ε computer

- Code:
  - http://tartarus.org/martin/PorterStemmer/
    - Implementations in C, Java, Perl, Python, C#, Lisp, Ruby, VB, javascript, php, Prolog, Haskell, matlab, tcl, D, and erlang

Caveat

reduce false negative? Recall
(Not matching things that we should have matched)
Dog-/-Dogs

reduce false positives? precision
(Matching strings that we should not have matched)

Policy—police

Query "dog"

Doc 1: I love my dog
Doc 2: I do not like dogs

Query "policy"

Doc 3: Singapore policy on gum
Doc 4: Singapore police cool

Tokenizing

- Identifying the tokens (words) in a text that we may want to deal with
- Called Word segmentation, word tokenization
  - tokenizer
- Pretty much a prerequisite to doing anything interesting

Tokenizing

- For English, why not just use white-space?
  - `Mr. Sherwood said reaction to Sea Containers' proposal has been "very positive." In New York Stock Exchange composite trading yesterday, Sea Containers closed at $62.625, up 62.5 cents.`
  - `"I said, 'what're you? Crazy?' " said Sadowsky. "I can't afford to do that.''`
- Using white-space gives you words like:
  - `cents.`
  - `said,`
  - `positive."`
  - `Crazy?'`

Punctuation Issues

- Word-internal punctuation
  - M.P.H.
  - Ph.D.
  - AT&T
  - 01/02/06
  - Google.com
  - Yahoo!
  - 555,500.50
- Clitics
  - What’re -- What are crazy?’
  - I’m
- Multi-token words (named entity detection)
  - New York
  - Rock ‘n’ roll

Language Issues

- Chinese and Japanese have no spaces between words
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
  - Sharapova now lives in US southeastern Florida
- Also Thai
- Further complicated in Japanese, with multiple alphabets intermingled
  - *e.g. フォーチュン500社は情報不足のため時間あた$500K(約6,000万円)*

Segmentation in Chinese

- Words composed of characters
- Average word is 2.4 characters long.
- Standard segmentation algorithm:
  - Maximum Matching or Maxmatch (also called greedy algorithm)

Maximum Matching Word Segmentation

Given a lexicon of Chinese, and a string

1) Start a pointer at the beginning of the string
2) Find the longest word in dictionary that matches the string starting at pointer
3) Move the pointer over the word in string
4) Go to 2

Lexicon (Dictionary)
the
table
down
there

thetabledownthere

the table down there

English Example

Lexicon (Dictionary)
the
theta
table
down
there
Bled
own

thetabledownthere

theta bled own there

- But works pretty well in Chinese
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
  - Sharapova now lives in US southeastern Florida

What are weakness?

- An annual competition for Chinese segmentation alg
- better ones based on probabilities

Practical Examples

- URL segmentation
  - www.dietsthatwork.com
  - www.choosespain.com
- Hashtag segmentation
  - #unitedbrokemyguitar
  - #manchesterunited
  - allows Twitter users to track what many people (especially people whom you aren't already following) are reporting or thinking about a particular topic or event.

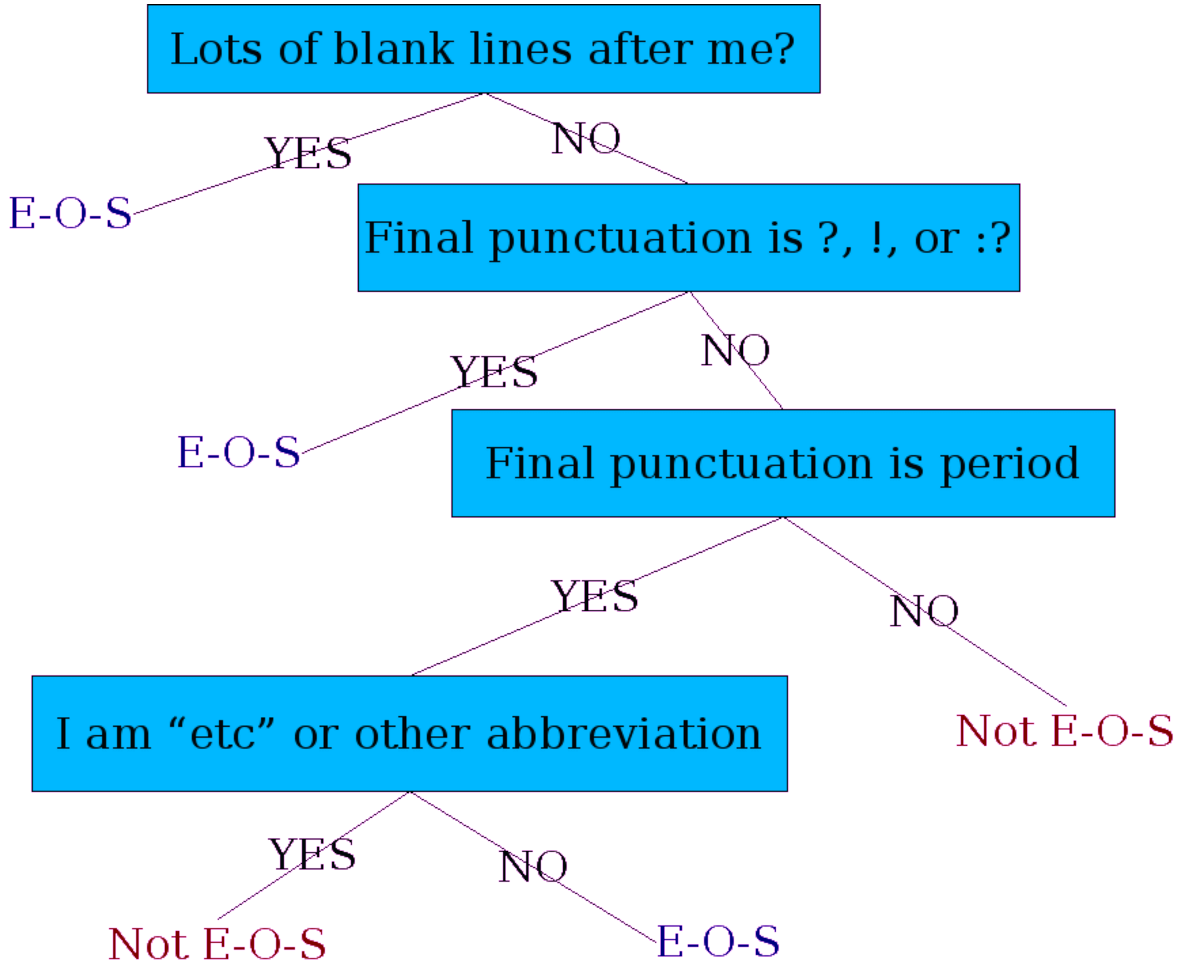
Sentence Segmentation

- Aka sentence tokenization
- Why not just use punctuation to find the sentences... Specifically break on "period space" "._"
  - `Mr. Sherwood said reaction to Sea Containers' proposal has been "very positive." In New York Stock Exchange composite trading yesterday, Sea Containers closed at $62.625, up 62.5 cents.`
  - `"I said, 'what're you? Crazy?' " said Sadowsky. "I can't afford to do that."`
- Because the '.' is ambiguous in English...
  - `… said the CEO of Apple Inc. Steve also mentioned …`

Sentence Segmentation

- !, ? are relatively unambiguous
- Period "." is quite ambiguous
  - Sentence boundary
  - Abbreviations like Inc. or Dr.
- General idea:
  - Build a binary classifier:
    - Looks at a "."
    - Decides EndOfSentence/NotEOS
    - Could be hand-written rules, sequences of regular expressions, or machine-learning

Decision Tree Version



Spelling Correction

- How do I fix “graffe”?
  - Search through all words in my lexicon
    - graf
    - craft
    - grail
    - giraffe
  - Pick the one that’s closest to graffe
  - What does “closest” mean?
  - We need a distance metric.
  - The simplest one: edit distance
    - Ala Unix diff

Edit Distance

- The edit distance between two strings is the minimum number of editing operations
  - Insertion
  - Deletion
  - Substitution
- Needed to transform one string into the other

Minimum Edit Distance

```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
d s s   i s
```

- If each operation has cost of 1
  - Distance between these is 5
- If substitutions cost 2 (Levenshtein)
  - Distance between these is 8

Min Edit Example

	i n t e n t i o n
delete i →	n t e n t i o n
substitute n by e →	e t e n t i o n
substitute t by x →	e x e n t i o n
insert u →	e x e n u t i o n
substitute n by c →	e x e c u t i o n

Defining Min Edit Distance

- For two strings S_1 of len *n*, S_2 of len *m*
  - S_1 source S_2 target
  - distance(*i,j*) or D(*i,j*)
    - means the edit distance of $S_1[1..i]$ and $S_2[1..j]$
    - i.e., the minimum number of edit operations need to transform first **i** characters of S_1 into the first **j** characters of S_2
    - The edit distance of S_1, S_2 is D(*n,m*)
- **We compute D(*n,m*) by computing D(i,j) for all *i* (0 <= *i* <= n) and *j* (0 <= j <= m)**

Defining Min Edit Distance

- Base conditions:
  - $D(i,0) = i$ (source) e.g. 'dog' → ''
  - $D(0,j) = j$ (target) e.g. '' → 'dog'

  - Recurrence Relation:
  - $$D(i,j) = \min \begin{cases} D(i-1,j) + 1 & \text{Ins} \\ D(i,j-1) + 1 & \text{Del} \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$$

Dynamic Programming

- A tabular computation of D(n,m)
- Bottom-up
  - We compute D(i,j) for small i,j
  - And compute D(i,j) for large i,j based on previously computed smaller values

The Edit Distance Table

D(1,1) ?
D(0,1) +1 = 2
D(1,0) +1 =2
D(0,0) +2 =2

i

	N	9									
	O	8									
	I	7									
	T	6									
	N	5									
4	E	4									
3	T	3									
2	N	2									
1	I	1	2	3							
0	#	0	1	2	3	4	5	6	7	8	9
		#	E	X	E	C	U	T	I	O	N

j

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$$

D(1, 2) ?
D(0,2) +1 = 3
D(1,1) +1 =3
D(0,1) +2 =3

D(1,1)

N	9	8	9	10	11	12	11	10	9	**8**
O	8	7	8	9	10	11	10	9	8	9
I	7	6	7	8	9	10	9	8	9	10
T	6	5	6	7	8	9	8	9	10	11
N	5	4	5	6	7	8	9	10	11	10
E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	**0**	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

Min Edit Distance

- Note that the result isn't all that informative
  - For a pair of strings we get back a single number
    - The min number of edits to get from here to there
- That's sort of like a map routing program that tells you the distance from here to Sentosa but doesn't tell you how to get there.

Paths/Alignments

- Keep a back pointer
  - Every time we fill a cell add a pointer back to the cell that was used to create it (the min cell that lead to it)
  - To get the sequence of operations follow the backpointer from the final cell

Backtrace

N	9	8	9	10	11	12	11	10	9	**8**
O	8	7	8	9	10	11	10	9	8	9
I	7	6	7	8	9	10	9	8	9	10
T	6	5	6	7	8	9	8	9	10	11
N	5	4	5	6	7	8	9	10	11	10
E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	**0**	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

Adding Backtrace to MinEdit

- Base conditions:
  - $D(i,0) = i$
  - $D(0,j) = j$
- Recurrence Relation:
  - $$D(i,j) = \min \begin{cases} D(i-1,j) + 1 & \text{Case 1} \\ D(i,j-1) + 1 & \text{Case 2} \\ D(i-1,j-1) + \begin{cases} 1; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} & \text{Case 3} \end{cases}$$

$$ptr(i,j) \begin{cases} \text{DOWN} & \text{Case 1} \\ \text{LEFT} & \text{Case 2} \\ \text{DIAG} & \text{Case 3} \end{cases}$$

Complexity

- Time:

  $O(nm)$

- Space:

  $O(nm)$

- Backtrace

  $O(n+m)$

Summary

- English Morphology
- Stemming: Normalizing the words
- Tokenizing: Getting the words (or word-like elements)
- Segmentation: Getting sentences
- Edit distance