Natural Language Processing
CS 6320
Lecture 2
Basic Text Processing

Instructor: Sanda Harabagiu

PLAN

- *Regular Expressions*
- *Word Tokenization*
- *Word Morphology*
- *Sentence Segmentation*

Regular Expressions

- *One of the unsung successes in standardization in computer science has been the regular expression (RE), a language for specifying text search strings.*
  - *This practical language is used in every computer language, word processor, and text processing tools like the Unix tools grep or Emacs.*
- *Formally, a regular expression is an algebraic notation for characterizing a set of strings.*

❑ *REs are particularly useful for searching in texts, when we have a pattern to search for and a corpus of texts to search through.*

- *A regular expression search function will search through the corpus, returning all texts that match the pattern.*

❑ *The corpus can be a single document or a collection.*

- *For example, the Unix command-line tool **grep** takes a regular expression and returns every line of the input document that matches the expression.*

Regular expressions

- *A formal language for specifying text strings*
- *How can we search for any of these?*
  - *woodchuck*
  - *woodchucks*
  - *Woodchuck*
  - *Woodchucks*

Regular Expressions: Disjunctions

- *Letters inside square brackets []*

Pattern	Matches
`[wW]oodchuck`	Woodchuck, woodchuck
`[1234567890]`	Any digit

- Ranges *[A-Z]*

Pattern	Matches	
`[A-Z]`	An upper case letter	<u>D</u>renched Blossoms
`[a-z]`	A lower case letter	<u>m</u>y beans were impatient
`[0-9]`	A single digit	Chapter <u>1</u>: Down the Rabbit Hole

Regular Expressions: Negation in Disjunction

- *Negations* `[^Ss]`
  - *Carat means negation only when first in []*

Pattern	Matches	
`[^A-Z]`	Not an upper case letter	`Oyfn pripetchik`
`[^Ss]`	Neither ‘S’ nor ‘s’	`I have no exquisite reason”`
`[^e^]`	Neither e nor ^	`Look here`
`a^b`	The pattern a carat b	`Look up a^b now`

Regular Expressions: More Disjunction

- *Woodchucks is another name for groundhog!*
- *The pipe | for disjunction*

Pattern	Matches
groundhog|woodchuck	
yours|mine	yours mine
a|b|c	= [abc]
[gG]roundhog|[Ww]oodchuck	

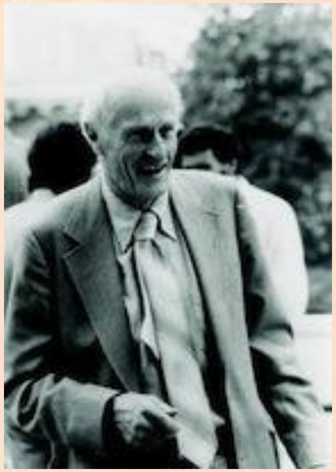
More Examples

- *Find all the instances of the word "the" in a text.*
  - */the/* → Misses capitalized examples
  - */[tT]he/* → Incorrectly returns `other` or `Theology`
  - */\b[tT]he\b/*

RE	Example Patterns Matched
/woodchucks/	"interesting links to woodchucks and lemurs"
/a/	"Mary Ann stopped by Mona's"
/Claire␣says,/	" "Dagmar, my gift please," Claire says,"
/DOROTHY/	"SURRENDER DOROTHY"
/!/	"You've left the burglar behind again!" said Nori

Regular Expressions: ?   *   +   .

Pattern	Matches	
colou?r	Optional previous char	color colour
oo*h!	0 or more of previous char	oh! ooh! oooh! ooooh!
o+h!	1 or more of previous char	oh! ooh! oooh! ooooh!
baa+		baa baaa baaaa baaaaa
beg.n		begin begun begun beg3n



Stephen C Kleene

Operators

Kleene *, Kleene +

Anchors

- *Kleene operators*
  - *Kleene *: 0 or more occurances of the previous char*
    - */[ab]*/ matches ababab or bbbb*
  - *Kleene +: 1 or more occurances of the previous char*
    - */[0-9]+/ specifies a sequence of digits*
- *Special characters that anchor regular expressions to particular places in a string:*
  - *^ specifies the start of a line*
  - *$ matches the end of a line*
  - *\b matches a word boundary: /\b35\b/ will match There are 35 dresses in the room, but it will not match There are 356 destinations.*
  - *\B matches a non-boundary*

What else???

- *The use of caret ^ for negation.*

RE	Match (single characters)	Example Patterns Matched
`[^A-Z]`	not an upper case letter	"Oyfn pripetchik"
`[^Ss]`	neither 'S' nor 's'	"I have no exquisite reason for't"
`[^\.]`	not a period	"our resident Djinn"
`[e^]`	either 'e' or '^'	"look up ^ now"
`a^b`	the pattern 'a^b'	"look up a^ b now"

- *The question mark ? Means **optionality** of the previous expression*

RE	Match	Example Patterns Matched
`woodchucks?`	woodchuck or woodchucks	"woodchuck"
`colou?r`	color or colour	"colour"

- *The use of the period . **To specify any character***

RE	Match	Example Patterns
`/beg.n/`	any character between *beg* and *n*	begin, beg'n, begun

Advanced Operators-1

- *Aliases for common set of characters*

RE	Expansion	Match	Examples
`\d`	`[0-9]`	any digit	Party␣of␣5
`\D`	`[^0-9]`	any non-digit	Blue␣moon
`\w`	`[a-zA-Z0-9_]`	any alphanumeric/underscore	Daiyu
`\W`	`[^\w]`	a non-alphanumeric	!!!!
`\s`	`[␣\r\t\n\f]`	whitespace (space, tab)	
`\S`	`[^\s]`	Non-whitespace	in␣Concord

Regular expression operators for counting:

RE	Match
`*`	zero or more occurrences of the previous char or expression
`+`	one or more occurrences of the previous char or expression
`?`	exactly zero or one occurrence of the previous char or expression
`{n}`	*n* occurrences of the previous char or expression
`{n,m}`	from *n* to *m* occurrences of the previous char or expression
`{n,}`	at least *n* occurrences of the previous char or expression

Advanced Operators-2

- *Characters that need to be backslashed*

RE	Match	Example Patterns Matched
`\*`	an asterisk "*"	"K*A*P*L*A*N"
`\.`	a period "."	"Dr. Livingston, I presume"
`\?`	a question mark	"Why don't they come and lend a hand?"
`\n`	a newline	
`\t`	a tab	

- *More examples: ELIZA simulating conversations:*

User$_1$:	Men are all alike.
ELIZA$_1$:	IN WHAT WAY
User$_2$:	They're always bugging us about something or other.
ELIZA$_2$:	CAN YOU THINK OF A SPECIFIC EXAMPLE
User$_3$:	Well, my boyfriend made me come here.
ELIZA$_3$:	YOUR BOYFRIEND MADE YOU COME HERE
User$_4$:	He says I'm depressed much of the time.
ELIZA$_4$:	I AM SORRY TO HEAR YOU ARE DEPRESSED

How is Eliza working?

- *ELIZA simulating conversations:*

User_1:	Men are all alike.
ELIZA_1:	IN WHAT WAY
User_2:	They're always bugging us about something or other.
ELIZA_2:	CAN YOU THINK OF A SPECIFIC EXAMPLE
User_3:	Well, my boyfriend made me come here.
ELIZA_3:	YOUR BOYFRIEND MADE YOU COME HERE
User_4:	He says I'm depressed much of the time.
ELIZA_4:	I AM SORRY TO HEAR YOU ARE DEPRESSED

- It employs a cascade of regular expression

```
s/.* YOU ARE (depressed|sad) .*/I AM SORRY TO HEAR YOU ARE \1/
s/.* YOU ARE (depressed|sad) .*/WHY DO YOU THINK YOU ARE \1/
s/.* all .*/IN WHAT WAY/
s/.* always .*/CAN YOU THINK OF A SPECIFIC EXAMPLE/
```

Regular Expressions: Anchors ^ $

- *In a nutshell:*

Pattern	Matches
`[A-Z]`	Palo Alto
`^[^A-Za-z]`	1 "Hello"
`\.$`	The end.
`.$`	The end? The end!

$ matches the end of a line

Errors

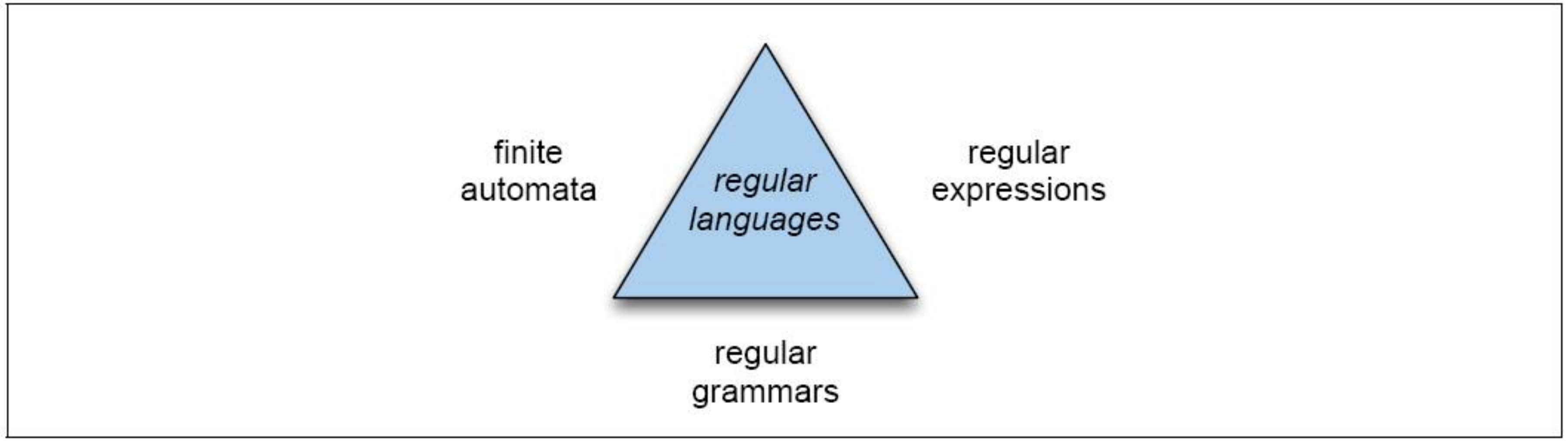
- *Two kinds of errors*
  - *Matching strings that we should not have matched (there, then, other)*
    - *False positives (Type I)*
  - *Not matching things that we should have matched (The)*
    - *False negatives (Type II)*

Errors cont.

- *In NLP we are always dealing with these kinds of errors.*
- *Reducing the error rate for an application often involves two antagonistic efforts:*
  - *Increasing accuracy or precision (minimizing false positives)*
  - *Increasing coverage or recall (minimizing false negatives).*

Finite State Automata

- *Regular expressions can be viewed as a textual way of specifying the structure of finite-state automata.*
- *FSAs and their probabilistic relatives are at the core of some NLP techniques.*
- *They also capture significant aspects of what linguists say we need for morphology and parts of syntax.*



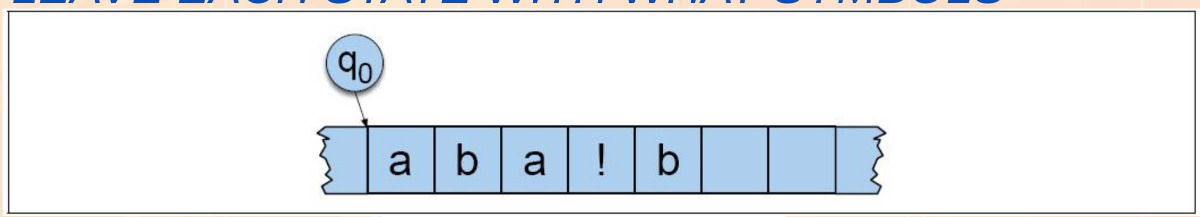
More Formally again

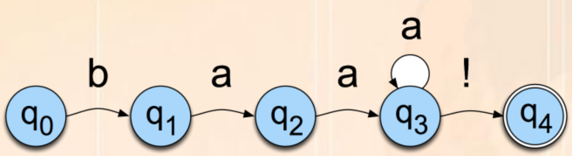
- *The definition of a finite automaton*

$Q = q_0 q_1 q_2 \ldots q_{N-1}$	a finite set of N **states**
Σ	a finite **input alphabet** of symbols
q_0	the **start state**
F	the set of **final states**, $F \subseteq Q$
$\delta(q,i)$	the **transition function** or transition matrix between states. Given a state $q \in Q$ and an input symbol $i \in \Sigma$, $\delta(q,i)$ returns a new state $q' \in Q$. δ is thus a relation from $Q \times \Sigma$ to Q;

State-Transition Table

- *An automaton can also be represented by a state-transition table ; as in the graph notation, the state transition table represents:*
  - *THE START STATE*
  - *THE ACCEPTING STATES*
  - *WHAT TRANSITION LEAVE EACH STATE WITH WHAT SYMBOLS*
- *For example:*





	Input		
State	b	a	!
0	1	∅	∅
1	∅	2	∅
2	∅	3	∅
3	∅	3	4
4:	∅	∅	∅

Summary

- *Regular expressions play a surprisingly large role*
  - *Sophisticated sequences of regular expressions are often the first model for any text processing text*
- *For many hard tasks, we use machine learning classifiers*
  - *But regular expressions are used as features in the classifiers*
  - *Can be very useful in capturing generalizations*

Words

they lay back on the San Francisco grass and looked at the stars and their

- ***Token**: an instance of that type in running text.*
- *How many?*
  - *15 tokens (or 14)*

How many words?

N *= number of tokens*

V *= vocabulary*

Church and Gale (1990): $|V| > O(N^{½})$

	Tokens = N	**|V|**
Switchboard phone conversations	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
Google N-grams	1 trillion	13 million

Issues in Tokenization

- *Finland's capital* → *Finland Finlands Finland's ?*
- *what're, I'm, isn't* → *What are, I am, is not*
- *Hewlett-Packard* → *Hewlett Packard ?*
- *state-of-the-art* → *state of the art ?*
- *Lowercase* → *lower-case lowercase lower case ?*
- *San Francisco* → *one token or two?*
- *m.p.h., PhD.* → *??*

In Natural Language Processing we care about punctuation – and do not discard it!!!!

Finland's capital → *Finland ' s capital*

Tokenization: language issues

- *French*
  - ***L'ensemble** → one token or two?*
    - ***L** ? **L'** ? **Le** ?*
    - *Want **l'ensemble** to match with **un ensemble***
- *German noun compounds are not segmented*
  - ***Lebensversicherungsgesellschaftsangestellter***
  - *'life insurance company employee'*
  - *German information retrieval needs **compound splitter***

Word Morphology

- ***Morphology*** *is the study of the way words are built from smaller units called morphemes*
  - *A morpheme is a minimal meaning-bearing unit in language.*
  - *Example: the word DOG has a single morpheme; the word CATS has two: (1) CATand (2) –S*
- *There are two classes of morphemes:* stems and affixes.
- *Stems -- are the main morphemes of words.*
- *Affixes – add additional meaning to the stems to modify their meanings and grammatical functions*

Morphology and Finite-State Transducers

- *There are four forms of affixes:*
  1. *Prefixes – precede the stem*
  2. *Suffixes – follow the stem*
  3. *Infixes – inserted inside the stem*
  4. *Circumfixes – both precede and follow the stem.*
- *Examples:*
  - *Prefixes - "un", "a"*
  - *Suffixes - plurals, "ing"*
  - *Infixes - not common in English*
  - *Circumfixes : unbelievably*

  un + believe + able + ly

Kinds of Morphology

- *The usage of prefixes and suffixes concatenated to the stem creates a concatenative morphology.*
- *When morphemes are combines in more complex ways, we have a non-concatenative morphology.*
- *Inflectional morphology - is the combination of a word stem with a grammatical morpheme usually resulting in a word of the same class*
  - *Special case: Templatic morphology, also known as root-and-pattern morphology:*
    - *Used in semitic languages*, e.g. Arabic, Hebrew

Example of Templatic Morphology

- In Hebrew, a verb is constructed using two components:
  - A root (consisting usually of 3 consonants CCC) – carrying the main meaning
  - A template which gives the ordering of consonants and vowels and specifies more semantic information about the resulting verb, e,g, the voice (active, passive)
- *Example:* the tri-consonant root lmd (meaning: learn, study)
  - can be combined with a template CaCaC for active voice to produce the word lamad = "he studied"
  - can be combined with the template CiCeC for intensive to produce the word limed = "he tought"
  - can be combined with a template CuCaC for active voice to produce the word lumad = "he was taught"

Producing words from morphemes

- *By inflection – Inflectional morphology - is the combination of a word stem with a grammatical morpheme usually resulting in a word of the same class*
- *By derivation – Derivational Morphology - is the combination of a word stem with a grammatical morpheme usually resulting in a word of a different class.*
- *By compounding – by combining multiple words together, e.g. "doghouse"*
- *By cliticization – by combining a word stem with a clitic. A clitic is a morpheme that acts syntactically like a word but it is reduced in forms and it is attached to another word. Eg.g I've.*

Clitization

- *A clitic is a unit whose status lies between that of an affix and a word!*
  - *The phonological behavior of clitics is like affixes: they then to be short and unaccented*
  - *The syntactic behavior is more like words: they often act as pronouns, articles, conjunctions or verbs.*

Full Form	Clitic	Full Form	Clitic
am	’m	have	’ve
are	’re	has	’s
is	’s	had	’d
will	’ll	would	’d

Inflectional Morphology 1/2

- *Nouns: have an affix for plural and an affix for possessive*
  - *Plural – the suffix –s and the alternative spelling –es for regular nouns*

Singular	*Plural*
dog	*dogs*
farm	*farms*
school	*schools*
car	*cars*

Singular	*Plural*
ibis	*ibises*
waltz	*waltzes*
box	*boxes*
butterfly	*butterflies*

Irregular nouns:

Singular	*Plural*
mouse	*mice*
ox	*oxen*

Possessive - for words not ending in "s" the affix is "'s" (children/ children's) and for words ending in "s" the affix is "'" (llama/llama's; llamas/llamas') (Euripides/Euripides' comedies)

Inflectional Morphology 2/2

- *Verbal inflection is more complex than nominal inflection*
- *There are three classes of verbs in English:*
  - ***Main verbs*** *(eat, sleep, walk)*
  - ***Modal verbs*** *(can will, should)*
  - ***Primary verbs*** *(be, have, do)*

Regular/ Irregular Verbs

Morphological Class	Regularly Inflected Verbs			
stem	walk	merge	try	map
-*s* form	walks	merges	tries	maps
-*ing* participle	walking	merging	trying	mapping
Past form or -*ed* participle	walked	merged	tried	mapped

Morphological Class	Irregularly Inflected Verbs		
stem	eat	catch	cut
-*s* form	eats	catches	cuts
-*ing* participle	eating	catching	cutting
preterite	ate	caught	cut
past participle	eaten	caught	cut

Spanish Verb System

	Present Indicative	Imperfect Indicative	Future	Preterite	Present Subjunctive	Conditional	Imperfect Subjunctive	Future Subjunctive
1SG	amo	amaba	amaré	amé	ame	amaría	amara	amare
2SG	amas	amabas	amarás	amaste	ames	amarías	amaras	amares
3SG	ama	amaba	amará	amó	ame	amaría	amara	amáreme
1PL	amamos	amábamos	amaremos	amamos	amemos	amaríamos	amáramos	amáremos
2PL	amáis	amabais	amaréis	amasteis	améis	amaríais	amarais	amareis
3PL	aman	amaban	amarán	amaron	amen	amarían	amaran	amaren

To love in Spanish.
Some of the inflected forms of the verb “amar” in European Spanish
50 distinct verb forms for each regular verb.

Derivational Morphology

- *Changes of word class*
  - *<u>Nominalization:</u> the formation of new nouns from verbs or adjectives.*

Suffix	Base Verb/Adjective	Derived Noun
-ation	computerize (V)	computerization
-ee	appoint (V)	appointee
-er	kill (V)	killer
-ness	fuzzy (A)	fuzziness

  - *Adjectives can also be derived from verbs.*

Suffix	Base Noun/Verb	Derived Adjective
-al	computation (N)	computational
-able	embrace (V)	embraceable
-less	clue (N)	clueless

Morphological Parsing

- *In general parsing means taking an input and producing a structure for it.*
- *Morphological parsing takes as an input words and produces a structure that reveals its morphological features.*

English		Spanish		
Input	**Morphological Parse**	**Input**	**Morphological Parse**	**Gloss**
cats	cat +N +PL	pavos	pavo +N +Masc +Pl	'ducks'
cat	cat +N +SG	pavo	pavo +N +Masc +Sg	'duck'
cities	city +N +Pl	bebo	beber +V +PInd +1P +Sg	'I drink'
geese	goose +N +Pl	canto	cantar +V +PInd +1P +Sg	'I sing'
goose	goose +N +Sg	canto	canto +N +Masc +Sg	'song'
goose	goose +V	puse	poner +V +Perf +1P +Sg	'I was able'
gooses	goose +V +3P +Sg	vino	venir +V +Perf +3P +Sg	'he/she came'
merging	merge +V +PresPart	vino	vino +N +Masc +Sg	'wine'
caught	catch +V +PastPart	lugar	lugar +N +Masc +Sg	'place'
caught	catch +V +Past			

Morpholgy and FSAs

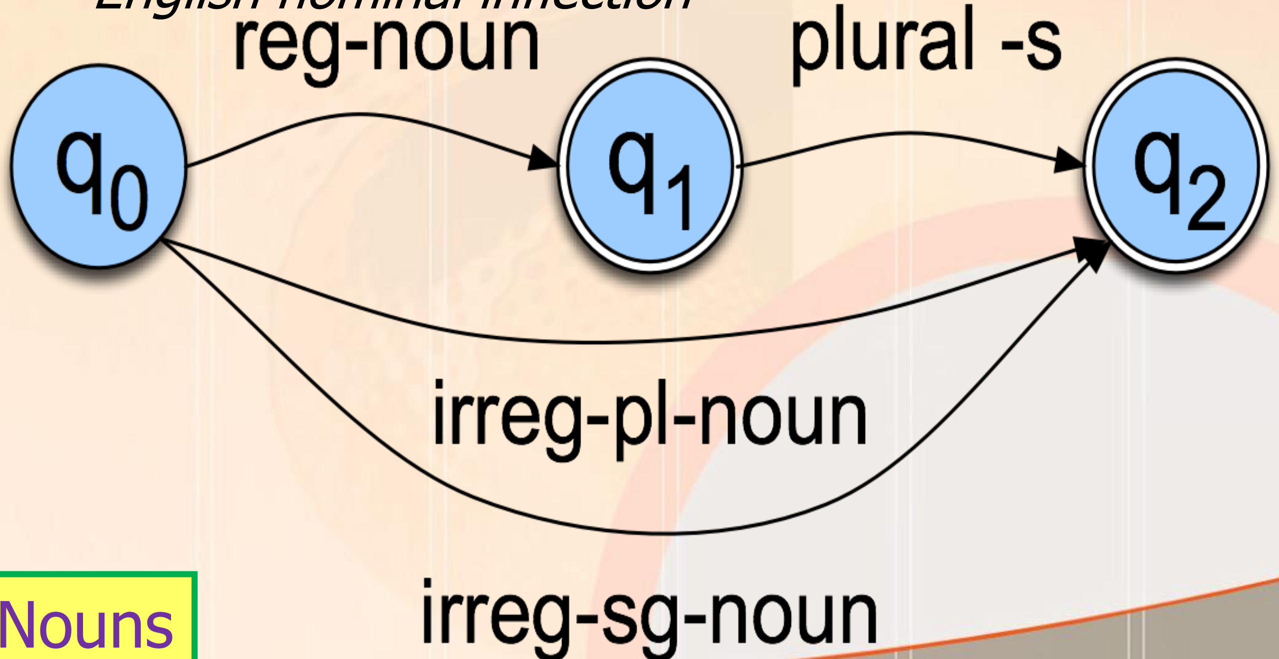
- *We'd like to use the machinery provided by FSAs to capture these facts about morphology*
  - *Accept strings that are in the language*
  - *Reject strings that are not*
  - *And do so in a way that doesn't require us to in effect list all the words in the language*

Building a finite-state **lexicon**: simple rules

- *A lexicon is a repository of words.*

➢ *Possibilities:*

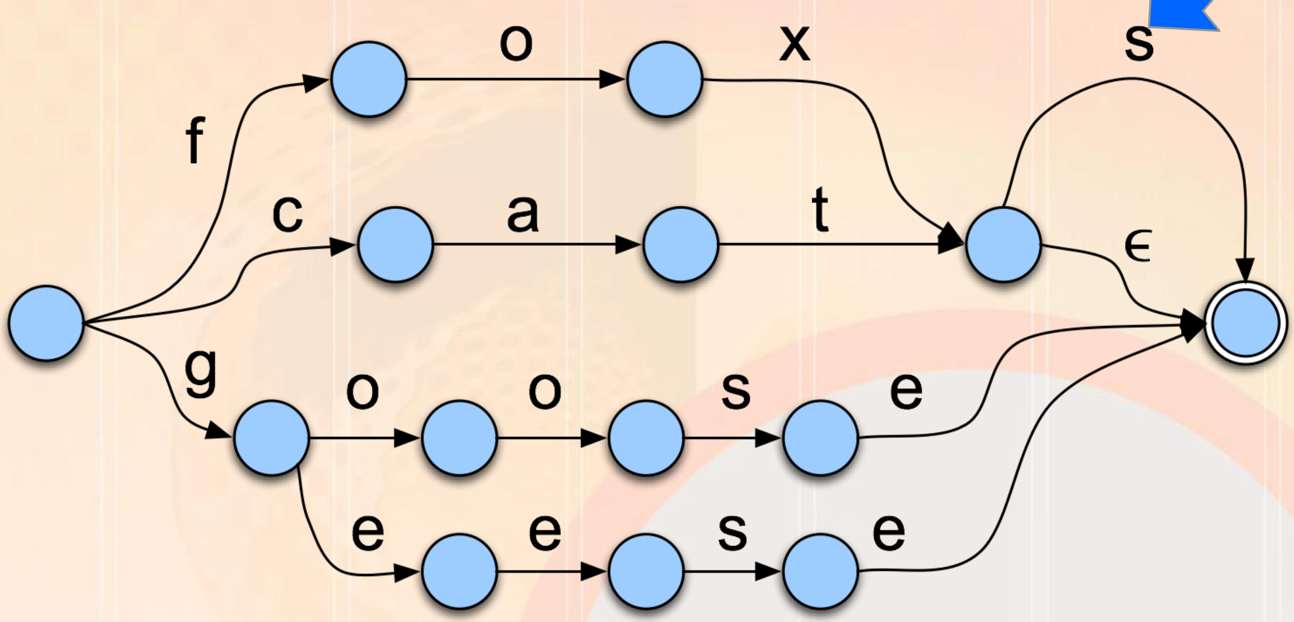
  - ❑ *List all words in the language*
  - ❑ *Computational lexicons: list all stems and affixes of a language + representation of the morphotactics that tells us how they fit together.*
    - *A example of morphotactics: the finite-state automaton (FSA) for English nominal inflection*



Nouns

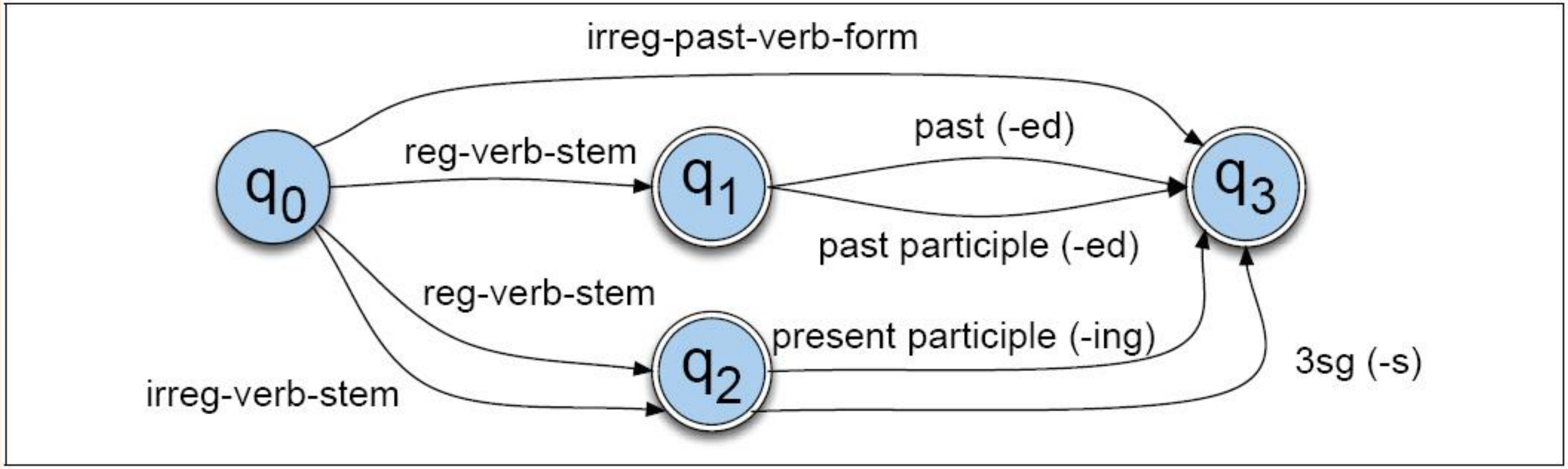
Now Plug in the Words

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



FSA for English Verb Inflection

reg-verb-stem	irreg-verb-stem	irreg-past-stem	past	past-part	pres-part	3sg
walk	cut	caught	-ed	-ed	-ing	-s
fry	speak	ate				
talk	sing	eaten				
impeach		sang				

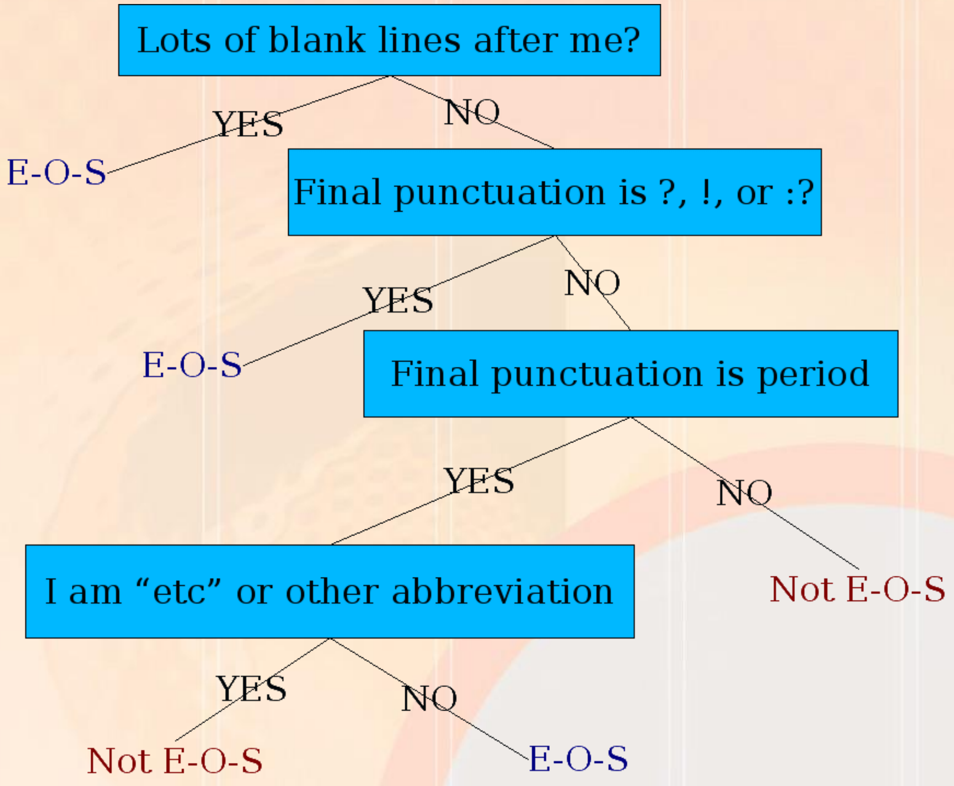


Sentence Segmentation

- *!, ? are relatively unambiguous*
- *Period "." is quite ambiguous*
  - *Sentence boundary*
  - *Abbreviations like Inc. or Dr.*
  - *Numbers like .02% or 4.3*
- *Build a binary classifier*
  - *Looks at a "."*
  - *Decides EndOfSentence/NotEndOfSentence*
  - *Classifiers: hand-written rules, regular expressions, or machine-learning*

Determining if a word is end-of-sentence:

a Decision Tree



More sophisticated decision tree features

- *Case of word with ".": Upper, Lower, Cap, Number*
- *Case of word after ".": Upper, Lower, Cap, Number*

- *Numeric features*
  - *Length of word with "."*
  - *Probability(word with "." occurs at end-of-s)*
  - *Probability(word after "." occurs at beginning-of-s)*

Implementing Decision Trees

- *A decision tree is just an if-then-else statement*
- *The interesting research is choosing the features*
- *Setting up the structure is often too hard to do by hand*
  - *Hand-building only possible for very simple features, domains*
    - *For numeric features, it's too hard to pick each threshold*
  - *Instead, structure usually learned by machine learning from a training corpus*

Decision Trees and other classifiers

- *We can think of the questions in a decision tree*
- *As features that could be exploited by any kind of classifier*
    - *Logistic regression*
    - *SVM*
    - *Neural Nets*
    - *etc.*