Basic Text Processing: Regular Expressions and Text Normalization

Regular expressions

- A formal language for specifying text strings
- They are particularly useful for searching in texts, when we have a pattern to search.
- 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.
- How can we search for any of these?
  - woodchuck
  - woodchucks
  - Woodchuck
  - Woodchucks

Regular Expressions: Disjunctions

- The string of characters inside the braces specifies a disjunction of characters to match.

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

- Ranges [A-Z]

Pattern	Matches	
[A-Z]	An upper case letter	Drenched Blossoms
[a-z]	A lower case letter	my beans were impatient
[0-9]	A single digit	Chapter 1: Down the Rabbit Hole

Regular Expressions: Negation in Disjunction

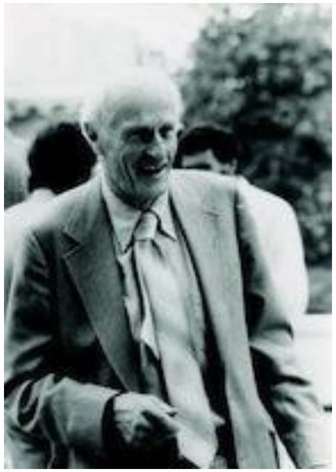
- The pattern /[2-5]/ specifies any one of the characters 2, 3, 4, or 5. The pattern /[b-g]/ specifies one of the characters b, c, d, e, f, or g.
- Negations `[^Ss]`
  - Carat means negation only when first in []

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: ? * + .

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

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



Stephen C Kleene

Kleene *, Kleene +

Regular Expressions: Anchors ^ $

- Anchors are special characters that anchor regular expressions to particular places in a string. The most common anchors are the caret ^ and the dollar sign $.
  - The caret matches the start of a line. The pattern /^The/ matches the word The only at the start of a line
- The caret ^ has three uses:
  - to match the start of a line,
  - to indicate a negation inside of square brackets,
  - and just to mean a caret.

Regular Expressions: Anchors ^ $

- The dollar sign $ matches the end of a line. So the pattern $ is a useful pattern for matching a space at the end of a line
  - /^The dog\.$/ matches a line that contains only the phrase The dog.
- There are also two other anchors: \b matches a word boundary, and \B matches a non-boundary.
  - \bthe\b/ matches *the* word the but not the word o*the*r.

Disjunction, Grouping, and Precedence

- The disjunction operator, also called the pipe symbol |. The pattern /cat|dog/ matches either the string cat or the string dog.
- To make the disjunction operator apply only to a specific pattern, we need to use the parenthesis operators (and).
  - the pattern /gupp(y|ies)/ would specify that we meant the disjunction only to apply to the suffixes y and ies.
  - we could write the expression /(Column [0-9]+ *)*/ to match the word Column, followed by a number and optional spaces, the whole pattern repeated any number of times

operator precedence hierarchy

- Parenthesis ()
- Counters * + ? {}
- Sequences and anchors the ^my end$
- Disjunction |

  - because counters have a higher precedence than sequences,/the*/ matches theeeee but not thethe.
  - Because sequences have a higher precedence than disjunction, /the|any/ matches the or any but not theny.

Greedy pattern

- Patterns can be ambiguous in another way.
  - The expression /[a-z]*/ when matching against the text once upon a time.
  - Since /[a-z]*/ matches zero or more letters, this expression could match nothing, or just the first letter o, on, onc, or once.
- In these cases regular expressions always match the largest string they can;
- we say that patterns are greedy, expanding to cover as much of a string as they can.

Example

- Find me all instances of the word "the" in a text.

`the`	**Misses capitalized examples**
`[tT]he`	Incorrectly returns `other` or `theology`
/\b[tT]he\b/	some context where it might also have underlines or numbers nearby (the or the25)
/[^a-zA-Z][tT]he[^a-zA-Z]/	won't find the word the when it begins a line
/(^|[^a-zA-Z])[tT]he([^a-zA-Z]|$)/	

Errors

- The process we just went through was based on fixing 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).

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

Text Normalization

Text Normalization

- Normalizing text means converting it to a more convenient, standard form.
  - Ex. most of what we are going to do with language relies on first separating out or tokenizing words
- English words are often separated from each other by whitespace, but whitespace is not always sufficient.
  - New York and rock ’n’ roll are sometimes treated as large words despite the fact that they contain spaces,
  - sometimes we’ll need to separate I’m into the two words I and am.
  - For processing tweets or texts we’ll need to tokenize emoticons like :) or hashtags like #nlproc

Lemmatization

- Another part of text normalization is lemmatization
  - The task of determining that two words have the same root, despite their surface differences.
  - For example, the words sang, sung, and sings are forms of the verb sing
- A lemmatizer maps from all of these to sing

Stemming and Segmentation

- Stemming refers to a simpler version of lemmatization in which we mainly just strip suffixes from the end of the word.
- Breaking up a text into individual sentences, using cues like periods or exclamation points.

Text Normalization

Every NLP task needs to do text normalization

1. Segmenting/tokenizing words
2. Normalizing word formats
3. Segmenting sentences in running text

Words

- What counts as a word?
  - Look one particular corpus (plural corpora), computer-readable collection of text or speech
  - Brown corpus is a million-word collection of samples from 500 written English texts from different genres (newspaper, fiction, non-fiction etc.)
  - How many words are in the following Brown sentence?

  He stepped out into the hall, was delighted to encounter a water brother.
- 13 words if we don't count punctuation marks as words, 15 if we count punctuation.

  Whether we treat period ("."), comma (","), and so on as words depends on the task. Punctuation is critical for finding boundaries.

Utterance

- Utterance is the spoken correlate of a sentence

  I do uh main- mainly business data processing

Two utterance

- Fragments : broken-off word main
- filled pauses : words like uh and um

How many words?

- **Types** are the number of distinct words in a corpus;
- **Tokens** are the total number of running words.

They picnicked by the pool, then lay back on the grass and looked at the stars.

- How many?
  - 16 tokens and 14 types

Herdan's Law/ Heaps' Law

- The larger the corpora we look at, the more word types we find
  - This relationship between the number of types |V| and number of tokens N is called Herdan's Law

$$|V| = kN^{\beta}$$

where k and β are positive constants, and $0 < \beta < 1$.

  - The value of β depends on the corpus size and the genre, for large β ranges from 0.67 to 0.75
  - V for a text goes up significantly faster than the square root of its tokens.

How many words?

N = number of tokens

V = vocabulary = set of types

$|V|$ is the size of the vocabulary

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

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

Lemma

- A lemma is a set of lexical forms having the same stem, the same major part-of-speech, and the same word sense.
  - Seuss's cat in the hat is different from other cats!
    - cat and cats = same lemma
- Word form: the full inflected surface form
    - cat and cats = different word forms

Lemmatization and Stemming

- Lemmatization is the task of determining that two words have the same root, despite their surface differences.
  - The words *am, are,* and *is* have the shared lemma *be*;
  - The words *dinner* and *dinners* both have the lemma *dinner*.
- Representing a word by its lemma is important for web search. This is especially important in morphologically complex languages.
  - Ex. *He is reading detective stories* would thus be *He be read detective story*.

How is lemmatization done?

- The most sophisticated methods for lemmatization involve complete morphological parsing of the word.
- Morphology is the study of morpheme the way words are built up from smaller meaning-bearing units called morphemes.
- Two broad classes of morphemes can be distinguished:
  - Stems : the central morpheme of the word, supplying the main meaning
  - Affixes : adding “additional” meanings of various kinds

Lemmatization

- Reduce inflections or variant forms to base form
  - *am, are, is* → *be*
  - *car, cars, car's, cars'* → *car*
- *the boy's cars are different colors* → *the boy car be different color*
- Lemmatization: have to find correct dictionary headword form

The Porter Stemmer

- One of the most widely used stemming algorithms is the simple and efficient Porter (1980) algorithm.

  - *ATIONAL -> ATE (e.g., relational->relate)*
  - *ING -> ε if stem contains vowel (e.g., motoring!->motor)*
  - *SSES -> SS (e.g., grasses -> grass)*

Porter's algorithm

Step 1a

```
sses → ss    caresses → caress
ies  → i     ponies   → poni
ss   → ss    caress   → caress
s    → ø     cats     → cat
```

Step 1b

```
(*v*)ing → ø  walking   → walk
              sing      → sing
(*v*)ed  → ø  plastered →
plaster
…
```

Step 2 (for long stems)

```
ational→ ate  relational→ relate
izer→ ize     digitizer → digitize
ator→ ate     operator  → operate
…
```

Step 3 (for longer stems)

```
al    → ø  revival    → reviv
able  → ø  adjustable → adjust
ate   → ø  activate   → activ
…
```

Viewing morphology in a corpus
Why only strip –ing if there is a vowel?


```
(*v*)ing → ø   walking    → walk
                sing      → sing
```

1312 King	548 being
548 being	541 nothing
541 nothing	152 something
388 king	145 coming
375 bring	130 morning
358 thing	122 having
307 ring	120 living
152 something	117 loving
145 coming	116 Being
130 morning	102 going

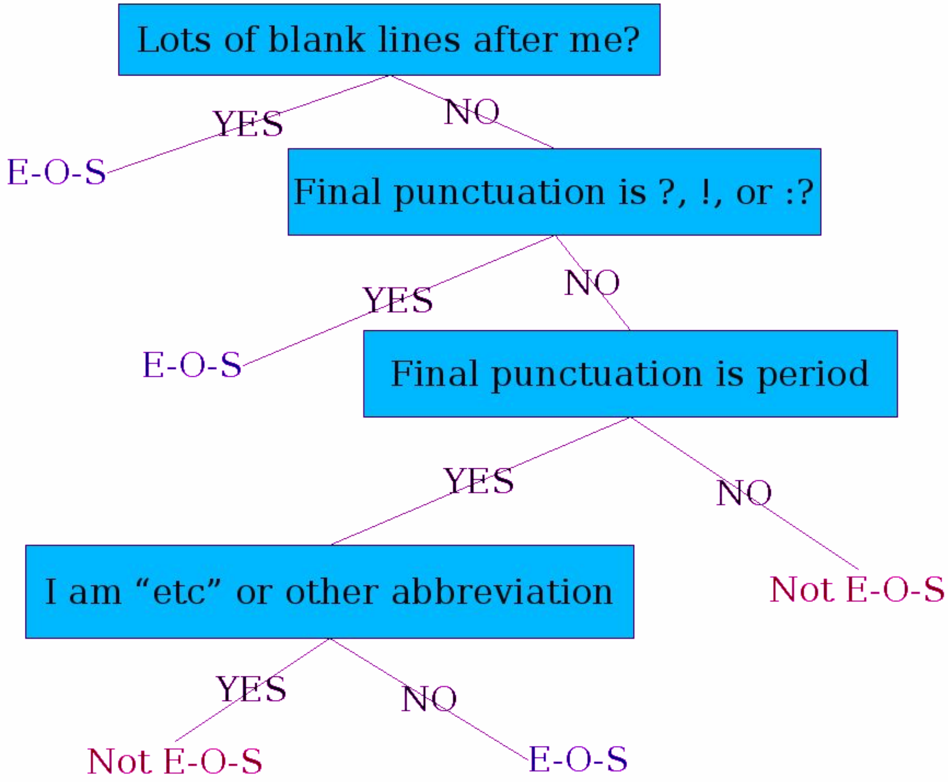
Case folding

- Applications like IR: reduce all letters to lower case
  - Since users tend to use lower case
  - Possible exception: upper case in mid-sentence?
    - e.g., ***General Motors***
    - ***Fed*** vs. ***fed***
    - ***SAIL*** vs. ***sail***
- For sentiment analysis, MT, Information extraction
  - Case is helpful (***US*** versus ***us*** is important)

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



THANK YOU