Tokenization, Normalization, and Stemming

Chapter 2 J&M'09

There are four elementary tasks in lexical processing:

- **Tokenization**: segmentation of a stream of text into words.
- **Normalization**: conversion of strings into a standard format.
- **Stemming**: removal of bound morphemes.
- **Lemmatization**: identification of the lemma (+ affixes)

Tokenization

How many words are there in the sentence below?

(1) A person from Niagara Falls wrote a five page letter to Obama's dog.

Tokenization

Types and Tokens

- type (class)
  The set of types is referred to as V (= vocabulary)
- token (occurrence)
  Number of tokens is referred to as N

Examples:

(2) a. The blue car is following the red car. ($|V| = 6$; $N = 8$)

b. The police also police the police. ($|V| = 4$; $N = 6$)

Types v.s Tokens in corpora (large collections of text)

Corpus	N	$\|V\|$
Shakespeare	884k	31k
Swichboard	2.4 million	20k

Tokenization

Tokenization: breaking the string into linguistic tokens

(3) a. He's Barack Obama → He | 's | Barack Obama
b. 7166452177 → 716 | 645 | 2177
c. 7166452177 → 7 | 166 | 452 | 177
d. #whatimissmost → # | what | i | miss | most

Harder to do in writing systems with no word spaces

(4) a. German compounding
die Hottentottenpotentatentantenattentäterin

b. 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

c. Japanese;
すべての人間は、生まれながらにして自由であり、かつ、尊厳と権利とについて平等である。人間は、理性と良心を授けられてあり、互いに同胞の精神をもって行動しなければならない。

Tokenization

The Scunthorpe problem

Mirror NEWS

Real news, real entert

FRONT PAGE | NEWS | SPORT | 3AM | TV | LIFESTYLE | MONEY | P

By Andy Rudd | 1 Comment | 8 Apr 2013 17:49

Margaret Thatcher dead: Worried Cher fans thought singer had died after Twitter #nowthatchersdead

Tributes to American singer after fans misinterpret Lady Thatcher hashtag

Tweet 20 Like 778 Send

- www.expertsexchange.com
- www.shitakemushrooms.com
- www.penisland.net
- www.therapistfinder.com
- Online file names

Tokenization

Max-match segmentation (Greedy): a tokenizer algorithm

1. $X := \epsilon$ (the empty string).
2. Read one character Y from the input string
3. $X' := X + Y$
4. Let W be the longest word in the dictionary that starts with (or is equal to) the string X'.
5. if $W = X'$ then return X', otherwise go to 2.

Example:

(5) the|cat|jumped

Not a good method for languages where words vary in length:

(6) a. thetabledownthere
    b. theta|bled|own|there
    c. the|table|down|there

Tokenization

Prolog version (not as greedy, and backtracks)

```
1  greed([],[]).
2  greed(String,[Token|TokenList]):-
3        tokenize([],String,StringRest,Token),
4        greed(StringRest,TokenList).
5
6  tokenize(String,[Char|Rest1],Rest2,Token):-
7     append(String,[Char],NewString),
8     check(NewString,Rest1,Rest2,Token).
9
10 check(NewString,Rest,Rest,NewString):- token(NewString).
11 check(NewString,Rest1,Rest2,Token):-
12   tokenize(NewString,Rest1,Rest2,Token).
13
14 token([t,h,e]).      token([t,h,e,t,a]).
15 token([t,a,b,l,e]). token([d,o,w,n]).
16 token([o,w,n]).      token([b,l,e,d]).
17 token([t,h,e,r,e]).
```

Tokenization

Execution:

```
?- greed([t,h,e,t,a,b,l,e,d,o,w,n,t,h,e,r,e],L).
L = [[t, h, e], [t, a, b, l, e], [d, o, w, n], [t, h, e, r, e]] ;
L = [[t, h, e, t, a], [b, l, e, d], [o, w, n], [t, h, e, r, e]] ;
false.

?- atom_chars('thetable',ListChar), greed(ListChar,TokenList).
ListChar = [t,h,e,t,a,b,l,e]
TokenList = [[t,h,e], [t,a,b,l,e]]
```

Tokenization

1. **Multi-word expressions**

   (7) the Netherlands, the Lakers, the London Bridge

   (Low-Level vs High-Level Tokenization)

2. **Word fusion** (clitics, contraction)

   (8) isn't, Sam's, didn't, Fred'll, I've, I'm, wanna, gonna, that'd

3. **Compounds**

   (9) a. freeze-dry, sisters-in-law, blue-green, Anglo-Cuban
   b. red head, high five, wet suit, sky scraper
   c. a [two cheese] omelet, a [six valve] engine, a [five page] letter, a [Sunday morning] walk

4. **Interwoven tokens**

   (10) Cali-freakin-fornia, abso-fucking-lutely, wel-diddly-elcome

5. **Word-part deletion**

   (11) There are 3-, 4-, and 5-year-old children here.

Other segmentation issues

- Not all periods indicate end of sentence.

  (12) a. You must talk to mr. Smith.
  b. 100 Riverview Park Dr., North Augusta
  c. He moved to the U.K. in April.

- Not all hyphens mean the same.

  (13) a. end-of-line hyphen (splits words for text justification)
  b. true hyphen: forty-seven
  c. affixal hyphen: ex-boyfriend, co-own
  d. phrasal hyphen: New York-based, K-9-like

Tokenization

NLTK (Python)
(see http://www.nltk.org/api/nltk.tokenize.html)

```
>>> import nltk
>>> sentence = """At eight o'clock on Thursday morning
... Arthur didn't feel very good."""
>>> tokens = nltk.word_tokenize(sentence)
>>> tokens
['At', 'eight', "o'clock", 'on', 'Thursday', 'morning',
'Arthur', 'did', "n't", 'feel', 'very', 'good', '.']
```

Tokenization

The Stanford NLP suite
(https://nlp.stanford.edu/software/tokenizer.shtml)

- English

```
java edu.stanford.nlp.process.DocumentPreprocessor
input.txt > output.txt
```

- Arabic

```
java -cp stanford-segmenter-2017-06-09/stanford-segmenter-3.8.0.jar
edu.stanford.nlp.international.arabic.process.ArabicSegmenter
-loadClassifier stanford-segmenter-2017-06-09/data/
arabic-segmenteratb+bn+arztrain.ser.gz -textFile input.txt > Output.txt
```

- Chinese

```
java -mx3g -cp "*" edu.stanford.nlp.pipeline.StanfordCoreNLP -props
StanfordCoreNLP-chinese.properties -file input.txt -outputFormat text >
output.txt
```

See https://stanfordnlp.github.io/CoreNLP/human-languages.html

Normalization

This leads us to **normalization**.
Necessary because there are multiple ways to spell the same thing:

(14) a. 8th-Feb, 8-Feb-2013, 02/08/13, 02.08.13, February 8th 2013, Feb 8th, ...

b. The US, The States, USA, U.S.A., U.S. of A, United States, United States of America, ...

c. ten thousand, Ten thousand, 10,000, 10K, 10^3

d. The car is near the tree. No! Near THE FENCE!

e. the dr., the doc, the doctor

f. He's here / He is here, I'd drink to that / I would drink to that, ...

Normalization & Regular Expressions

Regular expressions can describe complex text

Regular expressions:

- characters: `c`, `A100`, `30 years!`
- disjunction:
  - ordinary disjunction: `devoured|ate`, `famil(y|ies)`, `colou?r`
  - character classes: `[Tt]he`, `bec[oa]me`
  - ranges: `[A-Z]` (any capital letter), `[0-9]` (any number)
- negation: `[^a]` (any symbol but `a`)
  `[^A-Z0-9]` (not an uppercase letter or number)
- sequences:
  - any number of occurrences: `*` (Kleene star)
    `[0-9]* years`
  - at least one occurrence: `+`
    `[0-9]+ dollars`

A variety of unix tools (grep, sed, tr, . . .), and programming languages (perl, python, Java, . . .) incorporate regular expressions.

Normalization

Example: removing capitalization (with tr)

```
tr 'A-Z' 'a-z' < f1.txt > f2.txt
```

Replacing words and strings (with sed)

```
sed "s/mr./mister/g" f2.txt > f3.txt
```

De-capitalizing only a particular class of words

```
sed "s/[tT]hese/this/g" f1.txt > f2.txt
```

Removing sequences of characters

```
sed -E "s/tt*/t/g" f2.txt > f3.txt
```

Extract all emails (using egrep)

```
egrep -o '[a-zA-Z0-9]+@.+\..{3}' f1.txt  > f2.txt
```

Extract word repetitions

```
egrep -o '([a-z]+) \1' f1.txt > f2.txt
```

See the Regex tester. MS users can access linux this way.
The complete cheat sheet: here

Normalization

Obtaining the token count by sorting and counting words:

```
tr -sc 'A-Za-z' '\n' < f1.txt |sort|uniq -c |sort -n -r > f2.txt
```

31949 **the**	21207 and	16512 to
14938 of	10150 a	8499 in
8137 he	7815 that	7677 his
7342 was	5571 with	5333 had
4717 it	4642 her	4615 him
4593 not	4523 I	4402 s
4235 at	3797 as	3691 on
3412 for	3245 is	3236 you
2938 but	2842 said	2767 **The**
...	...	...

Legend:

- '-s' indicates any sequence, '-c' indicates the complement
- The symbol '\n' indicates carriage return
- The command `sort` orders the input ('-n' means numerical order, and '-r' reverses the sorted order)
- The command `uniq -c` deletes repeats and counts them

Normalization

Obtaining the normalized count instead:

```
tr 'A-Z' 'a-z' < f1.txt | tr -sc 'a-z' '\n' | sort | uniq
-c | sort -n -r > f2.txt
```

34720 the	22300 and	16753 to
15007 of	10609 a	10004 he
9036 in	8204 that	7984 his
7359 was	5710 with	5617 it
5365 had	4725 her	4697 not
4637 him	4547 at	4524 i
4415 s	4054 but	4035 as
4014 on	3871 you	3555 for
3488 she	3347 is	2842 said
2813 all	2709 from	2458 by
...	...	...

Stemming

Stemming: crude separation of the root from affixes.

Porter algorithm: cascated replace rule blocks

(15) a. sses → ss (caresses → caress)

b. ies → i (ponies → poni)

c. ss → ss (caress → caress)

d. s → ϵ (cats → cat)

(16) a. ational → ate (relational → relate)

b. izer → ize (digitzer → digitize)

c. ator → ate (operator → operate)

(17) a. al → ϵ (acquittal → acquit; revival → reviv)

b. able → ϵ (adjustable → adjust)

c. ate → ϵ (activate → activ)

Stemming

Care must be taken to deal with spelling variations

(18) a. cat + s = cats / car + s = cars / fan + s = fans

b. boss + es = bosses / mass + es = masses

c. ash + es = ashes / rash + es = rashes

d. quiz + es = quizzes / waltz + es = waltzes

(19) **im**balance, **in**complete, **ir**responsible, **il**literate

(20) a. hospital / legal / victim

b. hospitalize / legalize / victimize

c. hospitalization / legalization / victimization

Stemming

How to find good stemming rules?

- (*V*)ing $\rightarrow \epsilon$ (walking $\rightarrow$ walk; sing $\rightarrow$ sing)
- (*V*)ed $\rightarrow \epsilon$ (plastered $\rightarrow$ plaster; fed $\rightarrow$ fed)

Answer: probe text

```
tr -sc 'A-Za-z' '\n' < text.txt | tr 'A-Z' 'a-z' | grep
'ing$' | sort | uniq -c | sort -n -r | less
```

601 something	494 having	473 nothing
463 being	417 everything	415 looking
320 feeling	319 going	294 anything
233 saying	229 thing	224 during
221 sitting	216 morning	216 evening
211 drawing	203 taking	189 talking
...	...	...

Stemming vs. Lemmatization

A less crude form of stemming is **lemmatization** (identifying the lemma (root) of the token), as already discussed in the previous class.

- Lemma: **open**
  Paradigm: **ope**n, **open**s, **open**ing, **open**ed
- Lemma **be**
  Paradigm: **is**, **am**, **are**, **was**, **be**ing
  (negative counterparts **is**n't, **are**n't, **was**n't)
- Lemma: **dog**
  Paradigm: **dog**, **dog**s, **dog**'s, **dog**gy

Morphological parsing:

Stemming = (linguistically) shallow

Lemmatization = (linguistically) deep