Tokenization

Tokenization

- Given input text, split into words or sentences
- Tokens: words, numbers, punctuation
- Example:
  - Input: Sherwood said reaction has been “very positive.”
  - Output: Sherwood said reaction has been “ very positive . ”
- Why tokenize?
  - Identify basic units for downstream processing

Tokenization

- Proposal 1: Split on whitespace
- Good enough? No
- Why not?
  - Multi-linguality:
    - Languages without white space delimiters: Chinese, Japanese
    - Agglutinative languages (Hungarian, Korean)
      - meggazdagiithatnok
    - Compounding nouns (German):
      - Lebensversicherungsgesellschaftsangestellter

      “Life insurance company employee”
  - Even with English, the approach does not handle punctuation properly.

Tokenization - again

- Proposal 2: Split on white space and punctuation
  - For English
- Good enough? No
- Problems: Non-splitting punctuation
  - 1.23 ➔ 1 . 23 1,234,456 ➔ 1 , 234 , 456
  - don’t ➔ don ‘ t E-mail ➔ E - mail
- Problems: no-splitting whitespace
  - Names: New York; Collocations: pick up
- What’s a word?

Tokenization in Chinese, Japanese, Thai, etc.

- No whitespace in written languages

- Baseline: maximum match
  - Use a dictionary
  - Cannot handle unknown words or ambiguity

- Current approach: treat it as a sequence labeling task
  - Input: c1c2c3c4c5c6
  - Output: c1c2 c3 c4c5c6
  - Output: c1/B c2/E c3/S c4/B c5/I c6/E

What counts as a word?

- In theory:
  - Phonological word, syntactic words, lexeme, etc.
  - Ex: “On the definition of word” (Di Sciullo and Williams, 1987)

- In practice:
  - $22
  - Hyphenated words
  - Named entity
  - …

Sentence Segmentation

- Proposal: Split on period, !, ?

- Problems?
  - Non-boundary periods:
    - 1.23
    - Mr. Sherword
    - U.S.A.
    - Ph.D.
    - Etc.

- Solutions?
  - Rule-based approach: e.g., using heuristics and dictionaries.
  - Labeled data + machine learning

- What if the text is ASR output?

Hw1: tokenization

- Q1: write a tokenizer
- Q2: create a vocabulary from the data
- Q3: compare vocabulary size with or without tokenization

Q1: implementing a rule-based tokenizer for English

- The shell command line that we will run is:

  cat input_file | ./eng_tokenizer.sh abbrev-list > output_file

- Abbrev-list is a list of abbreviations (one abbreviation per line)

- Read from stdin, and write to stdout: each line in stdin will be tokenized and write to stdout, independent of other lines in stdin (i.e., a word will not cross the line boundary).

- Don’t merge the tokens in the input:
  - Ex: The string “Hong Kong” is treated as two words

Shell script

- A tool called “eng_tokenizer.sh”
  - Source code: eng_tokenizer.(pl | py | java |cpp | h |…)
  - Shell script: eng_tokenizer.sh, which could look like

```
#!/bin/sh
eng_tokenizer.pl $@
```

- More examples of shell script are under ~/dropbox/18-19/570/code-samples/

Specification of the tokenizer

- An alphanumeric character is [A-Za-z0-9], and often represented as \w in a RegEx (\W for a non-alphanumeric character)

- \W should, in general, be separated from surrounding letters, but there are many exceptions.

- Your tokenizer should not separate:
  - Numbers (e.g., -4, 1.23, 1,245)
  - Email address (e.g., xx@uw.edu)
  - Url (e.g., http://www.Washington.edu/)
  - Path (e.g., C:/dropbox/17-18/570/hw1/)
  - Abbreviation (e.g., Ph.D.)

Specification of the tokenizer (cont)

- Your tokenizer should not separate \W from surrounding chars in the following cases:
  - Period: when it is part of a number, an abbreviation, an url, an email address, the ellipsis (e.g., Ph.D., 1.23, xx@uw.edu)
  - Comma: when it is part of a number (e.g., 1,245,789)
  - Colon: when it is part of an url or a path (e.g., http://washington.edu/)
  - Slash and back slash: when it is part of a fraction, an url or a path (e.g., C:\dropbox\18-19\, 3/14)
  - Hyphen: when it is used as a minus sign (e.g., -4), part of a dash (“--”)
  - Tilde: part of an url or a path (e.g., ~/hw1/)
  - %: percentage sign (e.g., -2.3%)

Other special cases

- Apostrophe: isn't ➔ is n't, I'd ➔ I 'd, doesn't ➔ does n't
  fund's ➔ fund 's, funds' ➔ funds'

- Dollar sign ($) is separated from the following number:
  $12.5 ➔ $ 12.5

- For this assignment, an abbreviation either has a form of [\w\.]+ (e.g., N.V.) or appears in a given abbrev-list
  - The tokens in abbrev-list are case-sensitive: e.g., if the list contains Inc., not inc., then only the former in the input text is treated as one token by your tokenizer.

Q2: make_voc.sh

- Given the input text:

  The teacher bought the **bike.**

  The bike is expensive.

- The output should be

  The 2
  the 1
  teacher 1
  bought 1
  bike. 1
  bike 1
  is 1
  expensive. 1

make_voc.sh does NOT lowercase or tokenize the input

Q3: run Q1 and Q2

Files on patas

- All the files are under

  ~/dropbox/18-19/570/ on patas.

- For each hwX subdir (e.g., hw1/):
  - examples/: example input and maybe output files. The output files are NOT gold standard.
  - submit-file-list: the list of files to be submitted
  - check_hwX.sh: the script to check your hw.tar.gz

- The rubric is at Canvas:assignments, not on patas.

Your submission

- Your code must run on patas.
- The submitted files should be produced when running on patas.
- Your code will be tested on new data.
- Remember to run check_hw1.sh before submitting the tar file.
- Remember that we will test your code on our end, without your specific environment setting (e.g., alias, PATH).