# Word Tokenization and Distribution

Manish Shrivastava

# Tokenization

- The task of separating ‘Tokens’ in a given input sentence
- Example
  - “This is my brother’s cat”
  - Should  be
  - “ This is my brother’s cat ”
  - Or  should it be
  - “ This is my brother ‘ s cat ”

# Terms

- Token : a ‘Token’ is a single surface form word
  - Example :
    - My cat is afraid of other cats .
  - Here, ‘cat’ and ‘cats’ are both tokens
    - Giving a total of 8 tokens in the above sentence
    - ‘ . ’ or any punctuation mark is counted as a token

# Terms

- Type : Type is a vocabulary word
  - A word which might be present in its root form in the dictionary
  - ‘cat’ is the type for both ‘cat’ and ‘cats’
- The set of all types is the vocabulary of a language
- Ques: What would a high token to type ratio tell you about a language ?

# Tokenization

- Identifying individual tokens in a given input
  - Types are not of concern at this point
- Ques: Is the task of tokenization difficult?
  - ??

# Challenges

- Hyphenation
  - I am a hard-working student .
  - We would deal with the state-of-the-art .
  - I am not going to sho-

# Challenges

- Number
  - The value of gravity is 9.8 m/s/s
  - He got 1,000,000 dollars in VC funding .
  - My number is +918451990342
  - Take ½ cups of milk

# Challenges

- Dates
  - My birth date is 03/09/1982
  - I joined on 22nd July

# Challenges

- Abbreviation
  - Dr. S. P. Kishore is the primary faculty of this course
  - We are in IIIT-H campus

# Challenges

- Punctuations
  - This, hands-on experience, is rare in pedagogy .
  - I … uh … not sure how to proceed :-)

# Challenges

- URLs
  - The site for course materials for this class is http://tts.iiit.ac.in/~kishore/mediawiki/index.php/NLP:Tokens_and_Words

# Challenges

- Sentencification
  - “This is a presentation. It could also be a video”

# Regular Expressions

- Example
  - $ls *.txt
    - All files ending in “.txt”
- Abbreviation
  - “[A-Za-z]+.”
  - “[A-Za-z][A-Za-z]*\.”
    - “ . “ matches any character , hence needs to be escaped to match character “ . “
    - [A-Za-z] matches a single upper- or lower-case character

# Regular Expressions

- \s :matches any white-space eg. Tab,newline or space
- \t :tab
- \n :newline
- “^a” :matches strings beginning with letter ‘a’
- [^A-Z] : (NOTE the square braces) any character NOT in the sequence within the braces
- [aeiou] : will match one of the vowels (Think of this as the OR operator within the square braces)
- A||B : matches ‘A’ or ‘B’

# Regular Expressions

- Each programming language implements slightly differently
  - Java : Pattern and Matcher classes
  - C : include <regex.h>
  - Perl : inbuilt
  - Python : import re;

# Assignment & The Way Forward

# Assignment 1 - Twitter Tokenisation (A[1])

- What should you consider addressing ?
  - All the tokenisation challenges and the design considerations in them.
  - And … More
- The text in twitter is far from the perfect world of News Corpora.
- Here are some of the concerns:
  - Emoticons : Standardisation [ :) or :-) ]
  - Mentions (@Somebody)
  - HashTags (#Something)
  - Ellipsis and excessive punctuation (!!!)  (Lots of it)
  - URLS/Emails
  - Unicode Glyphs

# What do you need to do ?

- Look at the data !
  - Find out where the problem is harder than the vanilla tokenisation and figure out what you need to do.
- Develop a tokeniser
- Prepare a report on the challenges of this task and how your algorithm attempts to face them.
- In many instances, it might come down to design choices. It is okay. Just defend your choices.

# The next P&P : Part 1

- You have two choices : Choice 1
  - Take the text from P&P-1. Ideally it should be a continuous text if not, add a few more tokenised sentences to it. Now, for each word in that text, count the number of times it occurs.
    - PS : You can keep a threshold count say (count = 3 or 5 depending upon your corpus)
  - Now plot the words vs frequency in an ascending order (sort them before and then plot)
- Choice 2: Use your tokeniser.
  - Use the twitter data or some big enough corpus (see gutenberg.org) and tokenise
  - Just as above count how many times a word occurs
    - You can use this linux command sequence : cat <filename> | tr ‘ ‘ ‘\n’ | sort | uniq -c | sort -k1nr
  - Same as above plot the words-vs-freq in ascending order (above command gives you that)
- How does the plot look like ? and Why ?

# The next P&P : Part 2

- Look at this corpus :
  - Once upon a time, there were a little old woman and a little old man who lived in a little cottage near the river . The little old woman and the little old man were hungry, so the little old woman decided to bake a gingerbread man .
  - The little old woman made a big batch of gingerbread dough, then rolled it flat and cut it in the shape of a gingerbread man . She gave him raisins for eyes , a cinnamon drop for a mouth , and chocolate chips for buttons . Then she put the gingerbread man in the oven to bake.
- If I say :
  - The gingerbread <X> what should be the X ?
  - The little old <Y> what should be the Y ?
- How are we able to do this ?. Can you formalise it ?
- In general, given a word $w_i$ can we predict $w_{i+1}$