Analyzing Tweets: Introducing Text Analysis

Entities, parsing, regular expressions, frequency analysis

Outline

- What are entities?
- Parsing text
  - Simple parsing
  - Regular expressions
- Frequency analysis

Entities

- Usernames, Tags, URL
  - @dwmcphd, #HotFaculty, http://bit.ly/ur12c
  - Simple case, just pull them from the JSON
- Other "entites"
  - Dates, Place Names, People, emoticons
- General "entity resolution" is a difficult problem
  - When is a "thing" in text a meaningful thing?

Simple Text Parsing

- Built-in string functions – work on all strings
  - split(), rsplit()
  - strip(), rstrip(), lstrip()
  - endswith(), startswith()
  - find(), rfind(), index(), rindex()
  - upper(), lower()
  - replace()
  - partition(), rpartition()

Simple Text Parsing - example

```
s1 = "In this string a date might look like 1/27/2014 but sometimes people write that
14/01/27. Recognizing string dates can be tricky. For example some people use 01-27-14
as a date."
>>> s1.find(“this")
3
>>> s1.endswith(”date")
False
>>> s1.endswith(”date.”)
True
>>> s1.partition(".")
('In this string a date might look like 1/27/2014 but sometimes people write that
14/01/27', '.', ' Recognizing string dates can be tricky. For example some people use
01-27-14 as a date.')
>>> s1.rpartition(". ")[2]
'For example some people use 01-27-14 as a date.'
```

Simple Twitter Entity Identification

```
t1 = "#SEAHAWK SUNDAY. #Hawks over the 40whiner's and to the
SuperBowl!!!!!! GO #HAWKS, UTAH IS BEHIND YA!!!!"

t2 = "The 49ers will be no match for my hawks. #Seahawks
#NFLPlayoffs"

t3 = "RT @sixflagsDK: Whose ready? Seems OUR hawks are 49er fans!
#QuestforSix #GoNiners #NFC ##Seahawks #49ers http://t.co/C0Xbv3v7ss"
```

- Some issues here - maybe just find the hashtag entities?
  - Break each tweet into tokens with split()
  - Look for each token that startswith() a hash tag “#”

simple_parse_hashtags()

```
def simple_parse_hashtags(tweet=""):
    hash_tags = list()
    if tweet:
        token_list = tweet.split()
        for token in token_list:
            if token.startswith('#'):
                hash_tags.append(token)
    return hash_tags
```

Hashtag

```
t1 = "#SEAHAWK SUNDAY. #Hawks over the 40whiner's and to the SuperBowl!!!!!! GO
#HAWKS, UTAH IS BEHIND YA!!!!"
t2 = "The 49ers will be no match for my hawks. #Seahawks #NFLPlayoffs"
t3 = "RT @sixflagsDK: Whose ready? Seems OUR hawks are 49er fans! #QuestforSix
#GoNiners #NFC ##Seahawks #49ers http://t.co/C0Xbv3v7ss"

>>> tl1 = simple_parse_hashtags(t1)
>>> print tl1
['#SEAHAWK', '#Hawks', '#HAWKS,']
>>> tl2 = simple_parse_hashtags(t2)
>>> print tl2
['#Seahawks', '#NFLPlayoffs']
>>> tl3 = simple_parse_hashtags (t3)
>>> print tl3
['#QuestforSix', '#GoNiners', '#NFC', '##Seahawks', '#49ers']
>>>
```

Non-Trivial Parsing

- What if we wanted to get dates from our first sample string or find dates in tweets?
  - `s1 = "In this string a date might look like 1/27/2014 but sometimes people write that 14/01/27. Recognizing string dates can be tricky. For example some people use 01-27-14 as a date."`
  - Break it into tokens (words) with split
  - Check each word to see if the "/" character was there
  - Not impossible, but could be tedious
  - Matches a small number of cases
- Regular expressions are a more general solution

Regular Expression Concept

- Regular expressions are designed to find & match a pattern (a regular sequence of characters & digits).
  - http://
  - ftp://
  - Jan. 27, 2014
  - & ã ©
  - Figure 3
  - #seahawks, #gohawks, #hawks

Sample Regular Expression Strings

- r'(\d{1,2}?/\d{1,2}?/\d{4}) | (\d{1,2}?/\d{1,2}?/\d{2})'
  - Matches and captures dates like 1/5/2017 or 1/2/17
  - Two possible ways of matching the year
- r'&(#?x?\d+ | [^;]+);'
  - Matches and captures HTML entities like ã ¾ ©
- ur'(?i)\b((?:https?:// | www\d{0,3}[.] | [a-z0-9.\-]+[.][a-z]{2,4}/)(?:[^\s()<>]+ | \(([^\s()<>]+ | (\([^\s()<>]+\)))*\))+(?:\(([^\s()<>]+ | (\([^\s()<>]+\)))*\) | [^\s`!()\[\]{};:\'".,<>?\xab\xbb\u201c\u201d\u2018\u2019]))'
  - Matches a URL!!!

Patterns in Regular Expressions

- Patterns are built from strings of characters
  - Special Characters
  - Quantifiers
  - Special Sequences/Positions

Patterns in Regular Expressions

- Special Characters

\	escapes special characters.
.	matches any character
^	matches start of the string
$	matches end of the string
[5b-d]	matches any chars '5', 'b', 'c' or 'd'
[^a-c6]	matches any char except 'a', 'b', 'c' or '6'
R|S	matches either regex R or regex S
()	a capture group,

- Quantifiers
- Special Sequences/Positions

Patterns in Regular Expressions

- Special Characters
- Quantifiers

*	0 or more
+	1 or more
?	0 or 1
{m}	exactly 'm'
{m,n}	from m to n.
{m,n}?	from m to n, as few as possible

- Special Sequences/Positions

Patterns in Regular Expressions

- Special Characters
- Quantifiers
- Special Sequences/Positions

\A	Start of string
\b	Matches at word boundary
\B	Matches not at word boundary
\d	Digit
\D	Non-digit
\s	Whitespace
\S	Non-whitespace
\w	Alphanumeric
\W	Non-alphanumeric
\Z	End of string

Regular Expression Methods

- First you need to import the `re` module

```
>>> import re
```

- This is the default regular expression module in Python

Regular Expression Methods

- `search(<expression>,<text>)`
  - Find the expression, return a Match object
- `match(<expression>,<text>)`
  - Find expression at start of text
- `findall(<expression>,<text>)`
  - Find all instances, list of strings or tuples ordered by optional match position
- `finditer(<expression>,<text>)`
  - Find all instances in an iterator of Match objects

Try out some regular expressions

```
>>> import re
>>> s1 = "In this string a date might look like 1/27/2014 but sometimes people
write that 14/01/27. Recognizing string dates can be tricky. For example some
people use 01-27-14 as a date."
>>> match = re.findall(r'(\d{1,2}?/\d{1,2}?/\d{4})|(\d{1,2}?/\d{1,2}?/\d{2})',s1)
>>> print match
[('1/27/2014', ''), ('', '14/01/27')]
>>>  match = re.findall(r'&(#?x?\d+|[^;]+);',txt)
>>> print match
['amp', '#227', '#x000A9', 'lt', 'gt']
>>>
```

Try out some regular expressions

```
>>> import re
>>> t1 = "#SEAHAWK SUNDAY. #Hawks over the 40whiner's and to the SuperBowl!!!!!!
GO #HAWKS, UTAH IS BEHIND YA!!!!"
>>> t2 = "The 49ers will be no match for my hawks. #Seahawks #NFLPlayoffs"
>>> t3 = "RT @sixflagsDK: Whose ready? Seems OUR hawks are 49er fans!
#QuestforSix #GoNiners #NFC ##Seahawks #49ers http://t.co/C0Xbv3v7ss"
>>> m1 = re.findall(r'(#\S*#|#\S*|\S*#)',t1)
>>> m2 = re.findall(r'(#\S*#|#\S*|\S*#)',t2)
>>> m3 = re.findall(r'(#\S*#|#\S*|\S*#)',t3)
>>> print m1
['#SEAHAWK', '#Hawks', '#HAWKS,']
>>> print m2
['#Seahawks', '#NFLPlayoffs']
>>> print m3
['#QuestforSix', '#GoNiners', '#NFC', '##', '#49ers']
>>>
```

Regular Expressions

- Python 2.7 RE Cheatsheet
  - http://tartley.com/?p=1349
  - http://www.cheatography.com/davechild/cheat-sheets/python/
- Always tricky
- What should you do if you can't figure it out?

Regular Expressions

- Python 2.7 RE Cheatsheet
  - http://tartley.com/?p=1349
  - http://www.cheatography.com/davechild/cheat-sheets/python/
- Always tricky
- What should you do if you can't figure it out?
  - Just have to try it

Counting Tokens - Frequency

Frequency

- Different things that we might count
  - Counting words
  - Counting hashtags
  - Counting mentions

Frequency

- Building off the examples from last time
  - Demo counting words
  - Top 100
  - Chart frequency