

Applied Text Mining in Python

Regular Expressions

Processing Free-text

```
>>> text10 = '"Ethics are built right into the ideals and  
objectives of the United Nations" #UNSG @ NY Society for Ethical  
Culture bit.ly/2guVelr @UN @UN_Women'  
>>> text11 = text10.split(' ')  
>>> text11  
['"Ethics', 'are', 'built', 'right', 'into', 'the', 'ideals',  
'and', 'objectives', 'of', 'the', 'United', 'Nations"', '#UNSG',  
'@', 'NY', 'Society', 'for', 'Ethical', 'Culture', 'bit.ly/  
2guVelr', '@UN', '@UN_Women']
```

- **How do you find all Hashtags? Callouts?**

Finding Specific Words

- **Hashtags**

```
>>> [w for w in text11 if w.startswith('#')]
['#UNSG']
```

- **Callouts**

```
>>> [w for w in text11 if w.startswith('@')]
['@', '@UN', '@UN_Women']
```



Finding patterns with regular expressions

- Callouts are more than just tokens beginning with '@'

@UN_Spokesperson

@katyperry

@coursera

- Match something after '@'

- Alphabets
- Numbers
- Special symbols like '_'

@**[A-Za-z0-9_]+**

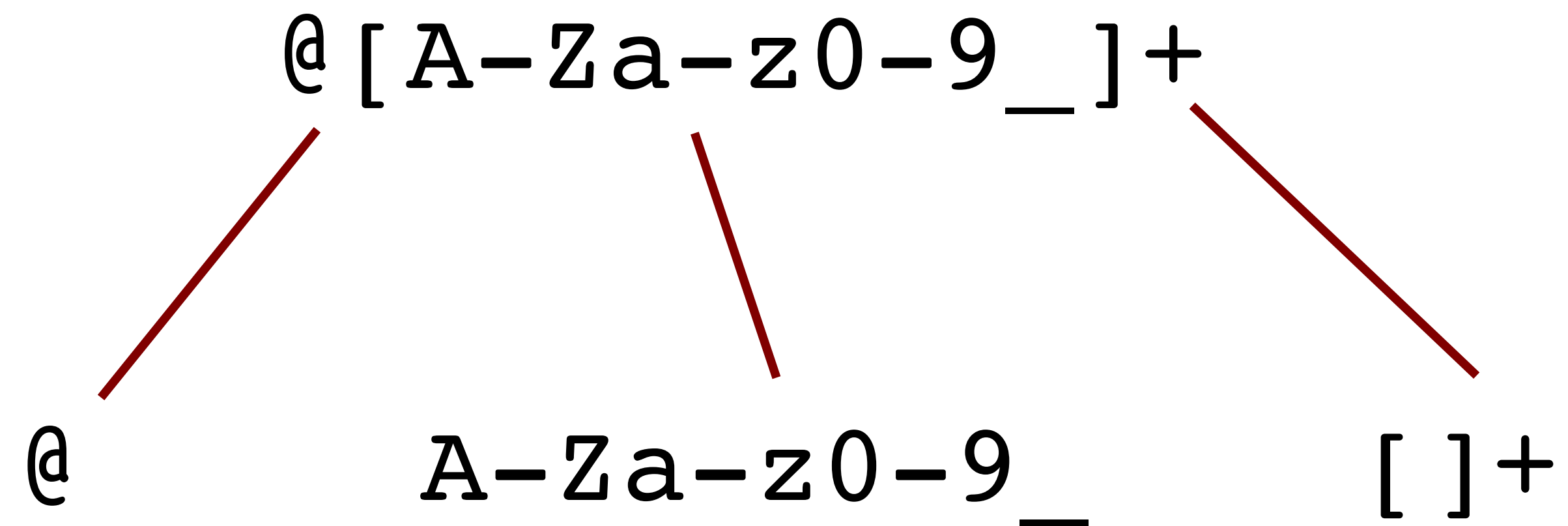
Let's try it out!

```
>>> text10 = '"Ethics are built right into the ideals and objectives of the  
United Nations" #UNSG @ NY Society for Ethical Culture bit.ly/2guVelr @UN  
@UN_Women'  
>>> text11 = text10.split(' ')  
>>> [w for w in text11 if w.startswith('@')]  
['@', '@UN', '@UN_Women']
```

Import regular expressions first!

```
>>> import re  
>>> [w for w in text11 if re.search('@[A-Za-z0-9_]+', w)]  
['@UN', '@UN_Women']
```

Parsing the callout regular expression



- starts with @
- followed by any alphabet (upper or lower case), digit, or underscore
- that repeats at least once, but any number of times

Meta-characters: Character matches

- .** : wildcard, matches a single character
- ^** : start of a string
- \$** : end of a string
- []** : matches one of the set of characters within []
- [a-z]** : matches one of the range of characters a, b, ..., z
- [^abc]** : matches a character that is not a, b, or, c
- a | b** : matches either a or b, where a and b are strings
- ()** : Scoping for operators
- ** : Escape character for special characters (`\t`, `\n`, `\b`)

Meta-characters: Character symbols

\b : Matches word boundary

\d : Any digit, equivalent to **[0-9]**

\D : Any non-digit, equivalent to **[^0-9]**

\s : Any whitespace, equivalent to **[\t\n\r\f\v]**

\S : Any non-whitespace, equivalent to **[^ \t\n\r\f\v]**

\w : Alphanumeric character, equivalent to **[a-zA-Z0-9_]**

\W : Non-alphanumeric, equivalent to **[^a-zA-Z0-9_]**

Meta-characters: Repetitions

***** : matches zero or more occurrences

+ : matches one or more occurrences

? : matches zero or one occurrences

{n} : exactly n repetitions, $n \geq 0$

{n,} : at least n repetitions

{,n} : at most n repetitions

{m,n} : at least m and at most n repetitions

Recall the callout regular expression

```
>>> text10 = '"Ethics are built right into the ideals and  
objectives of the United Nations" #UNSG @ NY Society for Ethical  
Culture bit.ly/2guVelr @UN @UN_Women'  
>>> text11 = text10.split(' ')  
  
>>> [w for w in text11 if re.search('@[A-Za-z0-9_]+', w)]  
['@UN', '@UN_Women']  
  
>>> [w for w in text11 if re.search('@\w+', w)]  
['@UN', '@UN_Women']
```

Let's look at some more examples!

- Finding specific characters

```
>>> text12 = 'ouagadougou'
```

```
>>> re.findall(r'[aeiou]', text12)
['o', 'u', 'a', 'a', 'o', 'u', 'o', 'u']
```

```
>>> re.findall(r'^[aeiou]', text12)
['g', 'd', 'g']
```

Case study: Regular expression for Dates

- Date variations for 23rd October 2002

23-10-2002

23/10/2002

23/10/02

10/23/2002

23 Oct 2002

23 October 2002

Oct 23, 2002

October 23, 2002

`\d{2}[/-]\d{2}[/-]\d{4}`

Regular Expression for Dates (contd.)

```
>>> dateStr = '23-10-2002\n23/10/2002\n23/10/02\n10/23/2002\n23 Oct 2002\n23  
October 2002\nOct 23, 2002\nOctober 23, 2002\n'
```

```
>>> re.findall(r'\d{2}[/-]\d{2}[/-]\d{4}', dateStr)  
['23-10-2002', '23/10/2002', '10/23/2002']
```

```
>>> re.findall(r'\d{2}[/-]\d{2}[/-]\d{2,4}', dateStr)  
['23-10-2002', '23/10/2002', '23/10/02', '10/23/2002']
```

```
>>> re.findall(r'\d{1,2}[/-]\d{1,2}[/-]\d{2,4}', dateStr)  
['23-10-2002', '23/10/2002', '23/10/02', '10/23/2002']
```

23-10-2002
23/10/2002
23/10/02
10/23/2002

Regex for Dates (contd.)

23 Oct 2002
23 October 2002
Oct 23, 2002
October 23, 2002

```
>>> re.findall(r'\d{2} (Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec) \d{4}', dateStr)
['Oct']
```

```
>>> re.findall(r'\d{2} (? :Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec) \d{4}', dateStr)
['23 Oct 2002']
```

```
>>> re.findall(r'\d{2} (? :Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec) [a-z]* \d{4}', dateStr)
['23 Oct 2002', '23 October 2002']
```

Regex for Dates (contd.)

```
23 Oct 2002  
23 October 2002  
Oct 23, 2002  
October 23, 2002
```

```
>>> re.findall(r'(?:\d{2} )?(?:Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|  
Dec)[a-z]* (?:\d{2}, )?\d{4}', dateStr)  
['23 Oct 2002', '23 October 2002', 'Oct 23, 2002', 'October 23, 2002']
```

```
>>> re.findall(r'(?:\d{1,2} )?(?:Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|  
Dec)[a-z]* (?:\d{1,2}, )?\d{4}', dateStr)  
['23 Oct 2002', '23 October 2002', 'Oct 23, 2002', 'October 23, 2002']
```


Take Home Concepts

- **What are regular expressions?**
- **Regular expression meta-characters**
- **Building a regular expression to identify dates**

Regular Expressions

<https://docs.python.org/3/library/re.html>

Tips and tricks of the trade for cleaning text in Python

https://stanford.edu/~rjweiss/public_html/IRiSS2013/text2/notebooks/cleaningtext.html

<https://www.analyticsvidhya.com/blog/2014/11/text-data-cleaning-steps-python/>

<http://ieva.rocks/2016/08/07/cleaning-text-for-nlp/>

https://chrisalbon.com/python/cleaning_text.html