Tokenisation

COMP61332: Text Mining
Week 1
Riza Batista-Navarro

What is a word?

John loves Mary

John loves Mary.

John loves Mary…

John doesn’t love Mary.

John runs 5 miles.

John runs 6.3 miles.

Dr. Jones loves Mary.

What is a word?

They’re not right.

They aren’t right.

Th’Bull i’th’thorn (Name of a Pub near Buxton)

The seashore is breezy.

The sea-shore is breezy.

The band plays rock-and-roll.

The Task

Break input--usually a sentence--into tokens

3 main classes of tokens often considered

- Morphosyntactic word
- Punctuation mark or special symbol
- A number

Are these enough? Other types of tokens?

- Endings of contractions, e.g., "*'re*" in "*we're*"
- Compounds and multi-words (e.g., daughter-in-law)

Challenges

Character encoding

- ASCII only?
- Unicode (UTF-8)

ASCII-fication or romanisation of texts

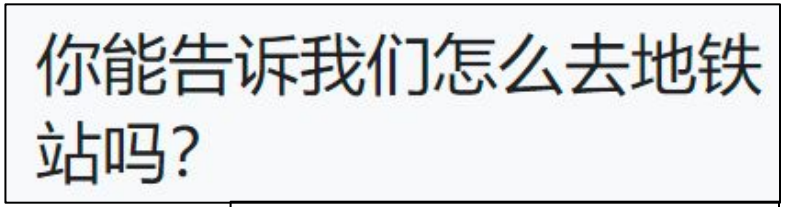
- Transliterations

Results from OCR may be poor

- Pre-processing to detect/correct errors
- OCR errors may appear as correct text (but not intended text)

Challenges: Writing systems

Token = whitespace-delimited character sequence?



Could you tell us how to get to the subway station please?

How about:

for Chinese, Japanese: no spaces between tokens?

for Arabic, Hebrew: generally written right to left, but not always

words separated, but complex ligatures used within words

numbers written left to right

Challenging examples

How many tokens in: *you're*

- 1? (you‘re)
- 2? (you + are)
- 3? (you + ‘ + re)

How about: *president's speech*

- president‘s + speech
- president + ‘s + speech

How about: *Carla's home*

- home of Carla (?)
- Carla is home (?)

More challenging examples

Hyphenation

- *Manchester-based*
- *Sister-in-law*

Telephone numbers: many different formats

- with whitespace, dots,
- slashes, hyphens, parentheses, plus signs

Dates: *04 January 2018*; *04-01-2018*; *Jan 4, 2018*

Decimals: *0.05*; *3.4*; *.6*

Monetary values: *a £5-a-dish dinner*

Domain dependence

Organism/species names, authorities, families, orders

- *E. coli*
- *Saccharum spp. L.*

Coordinates: *41032'38"N*

Protein sequence: *5'-TATGCTCGCCAGAGGATAATTA-3'*

Protein names: *MEK1/2* (Mek1 and Mek2)

Domain dependence

The importance of the phenolic quinolyl hydrazones arises from incorporating the quinoline ring with the phenolic compound; 2,4-dihydroxy benzaldehyde. The present study is planned to check the effect of the counter anions on the type and geometry of the isolated copper(II)-complexes as well as the ligational behavior of the phenolic hydrazone; 4-[(2-(4,8-dimethylquinolin-2-yl)hydrazono)methyl] benzene-1,3-diol; (H2L).

To split or not to split?

Some tokenisers attempt to handle multiwords and map these to one token

- *San Francisco*
- *12th January 2010*
- *12/01/2010*

Is this the job of tokenising? Or of named entity recognition?

To split or not to split?

- Sentence segmentation (split)
- Tokenisation (split)
- Named entity recognition (combine)

In other words: tokenisation is **knowing when to split** (not when to combine)

spaCy

During processing, spaCy first **tokenizes** the text, i.e. segments it into words, punctuation and so on. This is done by applying rules specific to each language. For example, punctuation at the end of a sentence should be split off – whereas "U.K." should remain one token. Each `Doc` consists of individual tokens, and we can iterate over them:

0	1	2	3	4	5	6	7	8	9	10
Apple	is	looking	at	buying	U.K.	startup	for	$	1	billion

First, the raw text is split on whitespace characters, similar to `text.split(' ')`. Then, the tokenizer processes the text from left to right. On each substring, it performs two checks:

1. **Does the substring match a tokenizer exception rule?** For example, "don't" does not contain whitespace, but should be split into two tokens, "do" and "n't", while "U.K." should always remain one token.

2. **Can a prefix, suffix or infix be split off?** For example punctuation like commas, periods, hyphens or quotes.