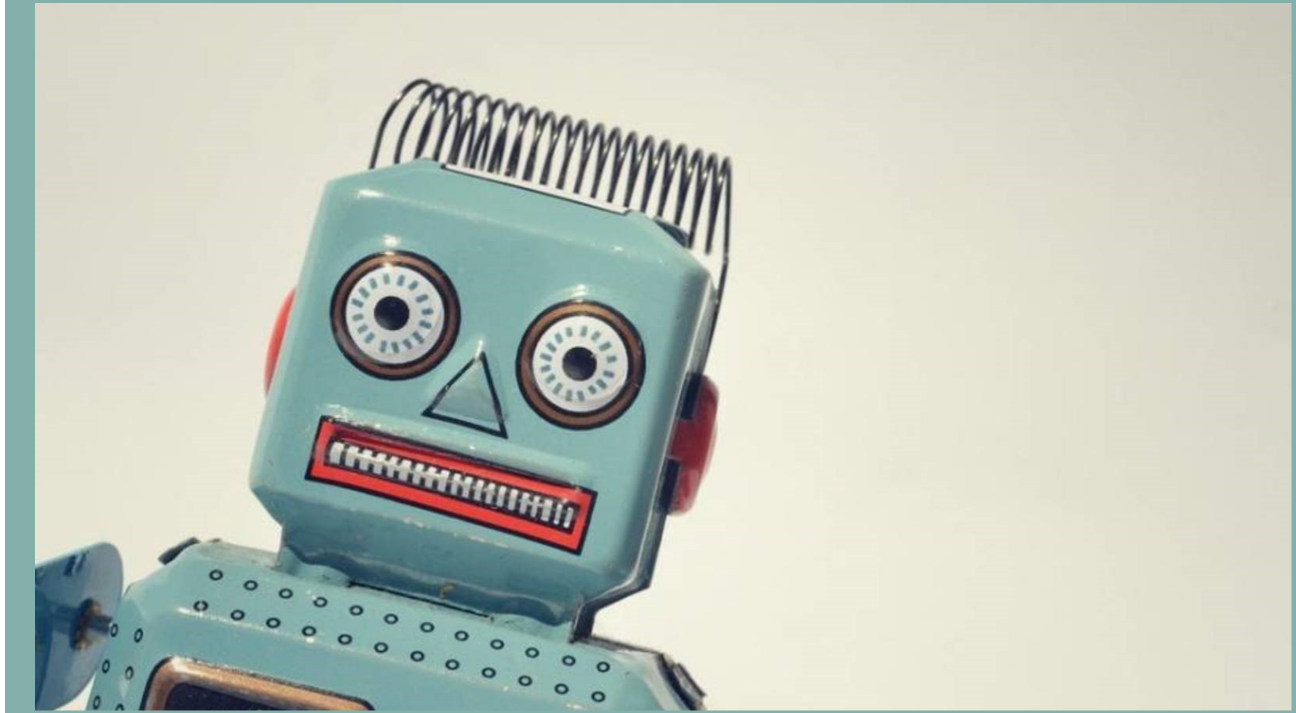
DIGITAL METHODS FOR ANALYSING TEXTS //

02_Analysing text

Ana Valdivia
Research Associate
King's College London



BEFORE WE START…//

What have we learned?

- Definition of AI and NLP.
- History of NLP.
- Language as data.

- Import unannotated and annotated corpus in Jupyter Notebook (Python).

BEFORE WE START…//

Difference between words and tokens

- If a text is 1,000 **words** long, it is said to have 1,000 **tokens**.

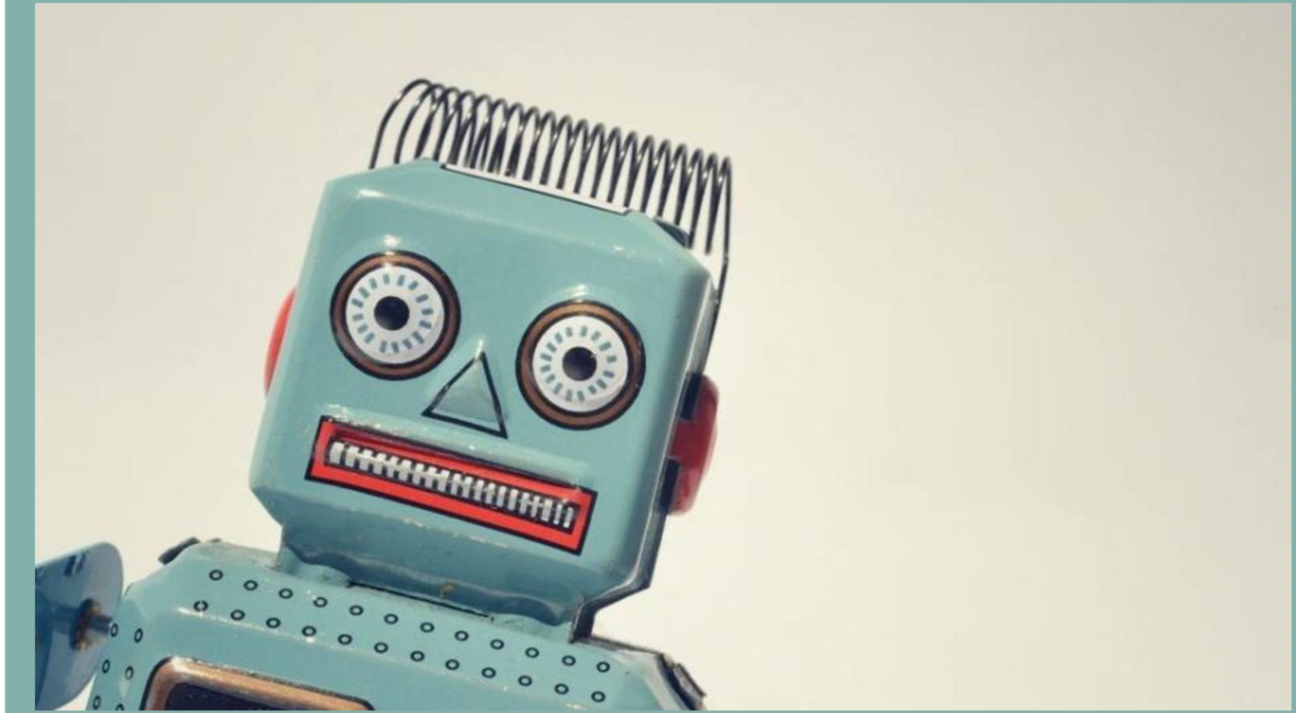
- But a lot of these words will be repeated, and there may be only say **400 different words (types)** in the texts.

ROAD MAP//

1. Corpus Preprocessing

2. Representing text
    - 2.1. Enter the matrix
    - 2.2. Discrete Representations
    - 2.3. Distributed Representations
    - 5.4. Discrete versus Continuous

CORPUS PREPROCESSING //



CORPUS PREPROCESSING//

How would you manually represent a corpus?

Click here

CORPUS PREPROCESSING//

Think like a person would do it, and translate it into code.

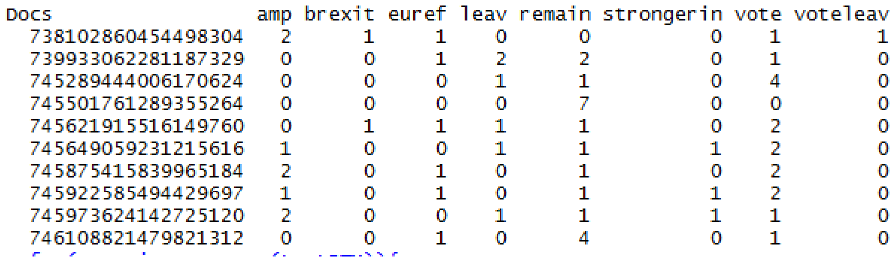
CORPUS PREPROCESSING//

Office documents

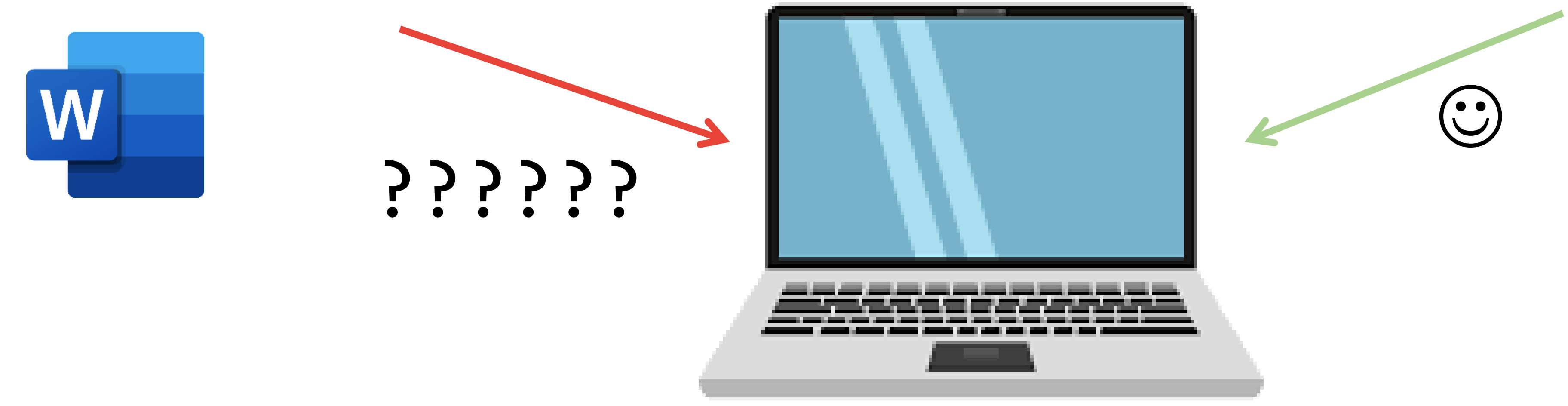
Features:

Term-matrix documents

Embeddings

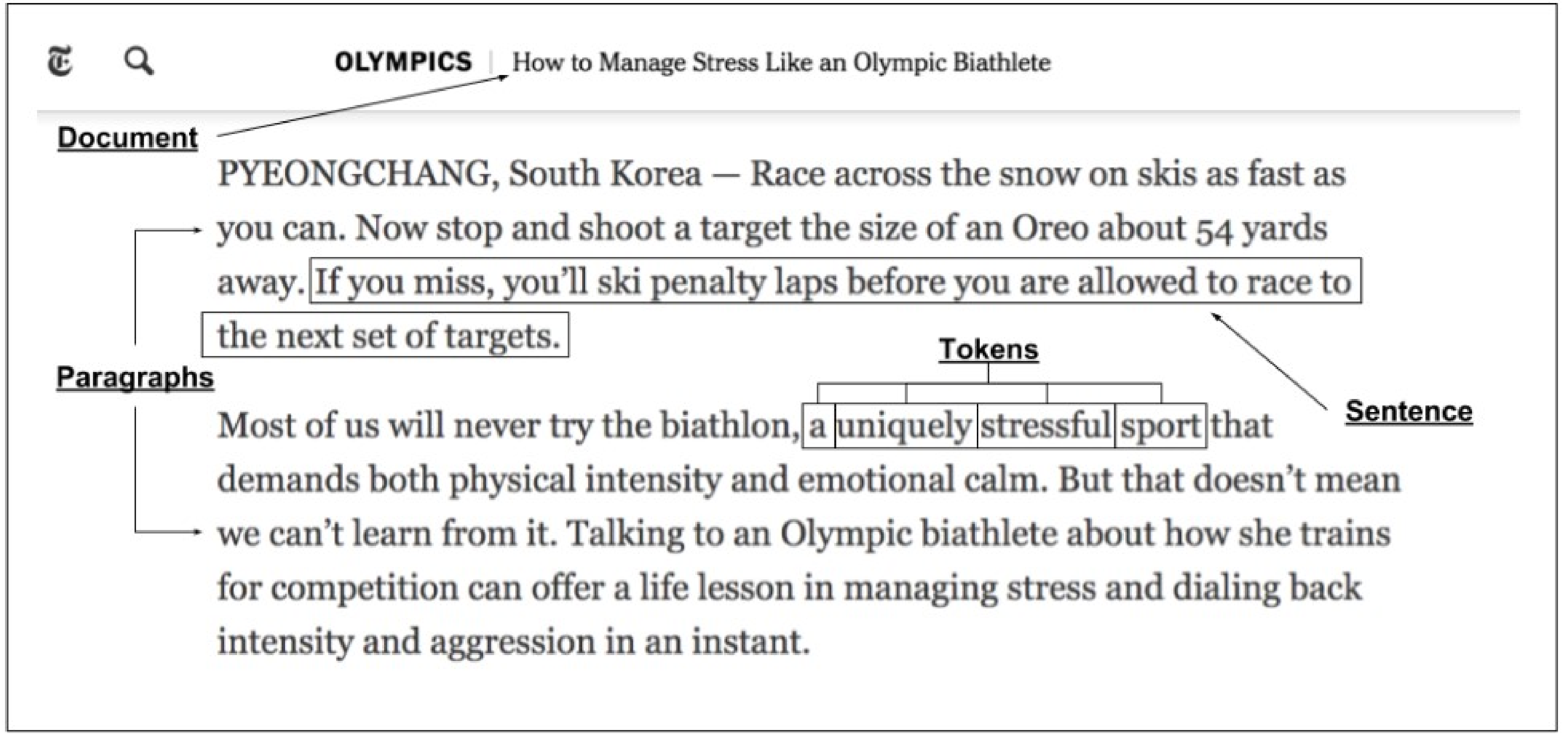


```
Docs                  amp brexit euref leav remain strongerin vote voteleav
  738102860454498304    2      1     1    0      0          0    1        1
  739933062281187329    0      0     1    2      2          0    1        0
  745289444006170624    0      0     0    1      1          0    4        0
  745501761289355264    0      0     0    0      7          0    0        0
  745621915516149760    0      1     1    1      1          0    2        0
  745649059231215616    1      0     0    1      1          1    2        0
  745875415839965184    2      0     1    0      1          0    2        0
  745922585494429697    1      0     1    0      1          1    2        0
  745973624142725120    2      0     0    1      1          1    1        0
  746108821479821312    0      0     1    0      4          0    1        0
```



CORPUS PREPROCESSING//

Document decomposition



PYEONGCHANG, South Korea — Race across the snow on skis as fast as you can. Now stop and shoot a target the size of an Oreo about 54 yards away. If you miss, you'll ski penalty laps before you are allowed to race to the next set of targets.

Most of us will never try the biathlon, a uniquely stressful sport that demands both physical intensity and emotional calm. But that doesn't mean we can't learn from it. Talking to an Olympic biathlete about how she trains for competition can offer a life lesson in managing stress and dialing back intensity and aggression in an instant.

Figure 3-2. Document decomposition illustrating the distribution of meaning across paragraphs, sentences, and individual tokens

CORPUS PREPROCESSING//

Features

Discrete features

Represents a feature with a specific meaning.
Term-document matrix.

Continuous features

Do not *mean* anything anymore.
They can not be interpreted.
Word embeddings.

Unicode

Our programs will often need to deal with different languages, and different character sets.

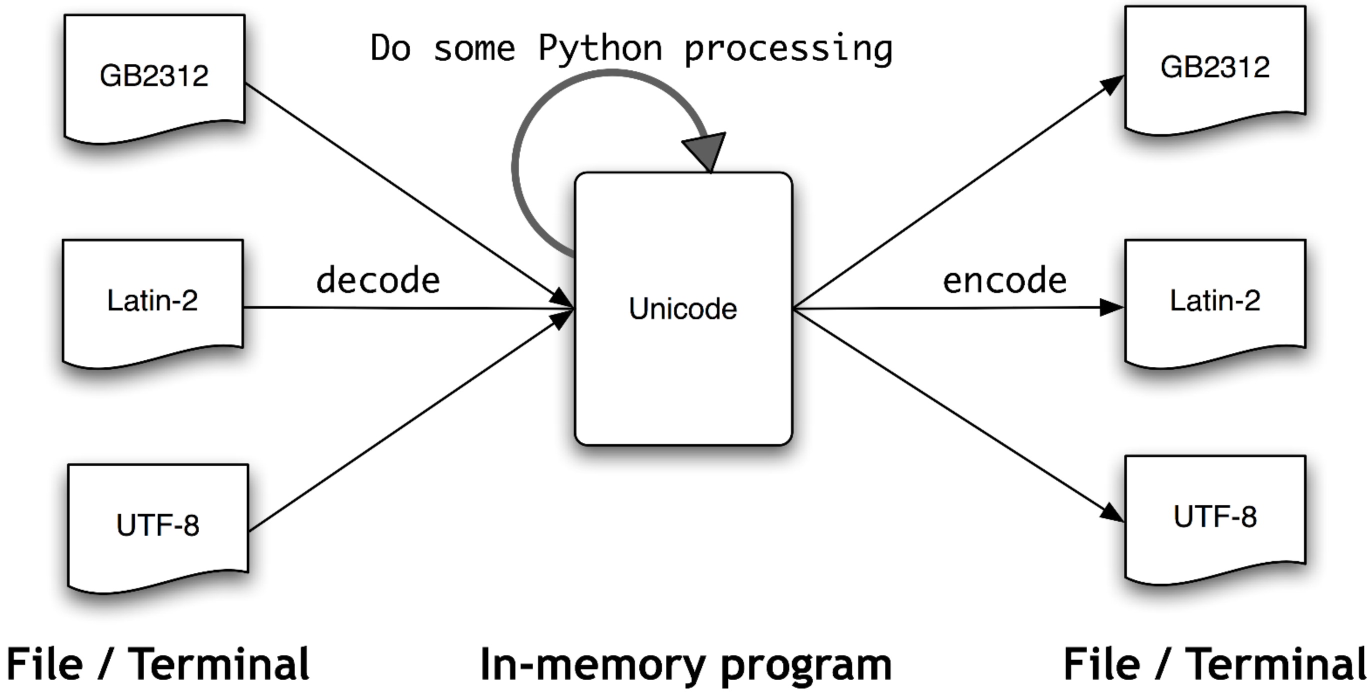
- If you live in the English-speaking world you probably use ASCIIT.
- If you live in Europe you might use one of the extended Latin character sets, containing such characters as:
  - “ø” for Danish and Norwegian,
  - “ő” for Hungarian,
  - “ñ” for Spanish and Breton, and
  - “ň” for Czech and Slovak.

CORPUS PREPROCESSING//

Unicode

Unicode supports over a million characters. Each character is assigned a number, called a code point.

In Python, code points are written in the form \uXXXX, where XXXX is the number in four-digit hexadecimal form.



Regular Expressions (regexpr)

Many linguistic processing tasks involve pattern matching. For example, we can find words ending with ed using endswith('ed').

Regular expressions give us a more powerful and flexible method for describing the character patterns we are interested in.

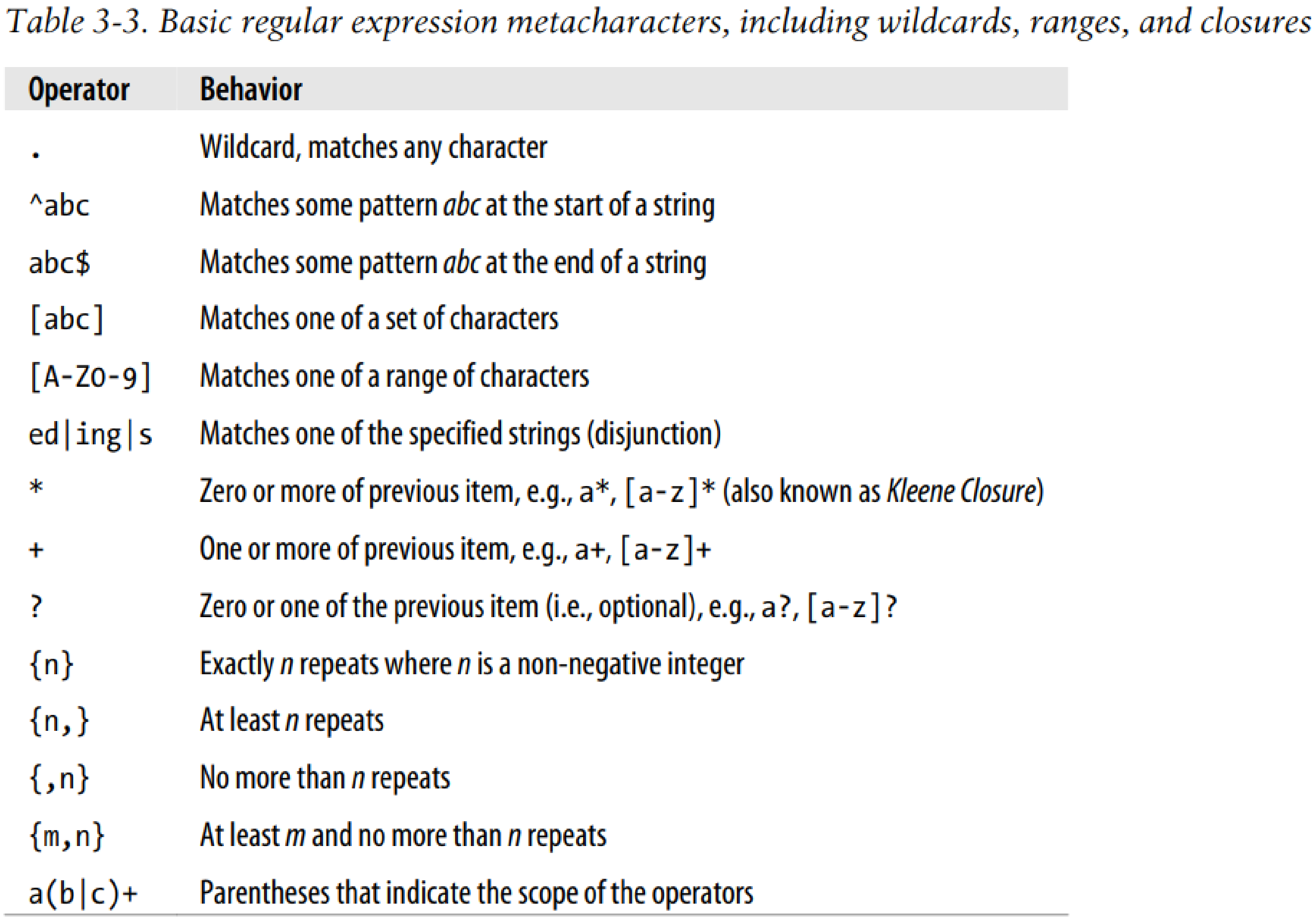
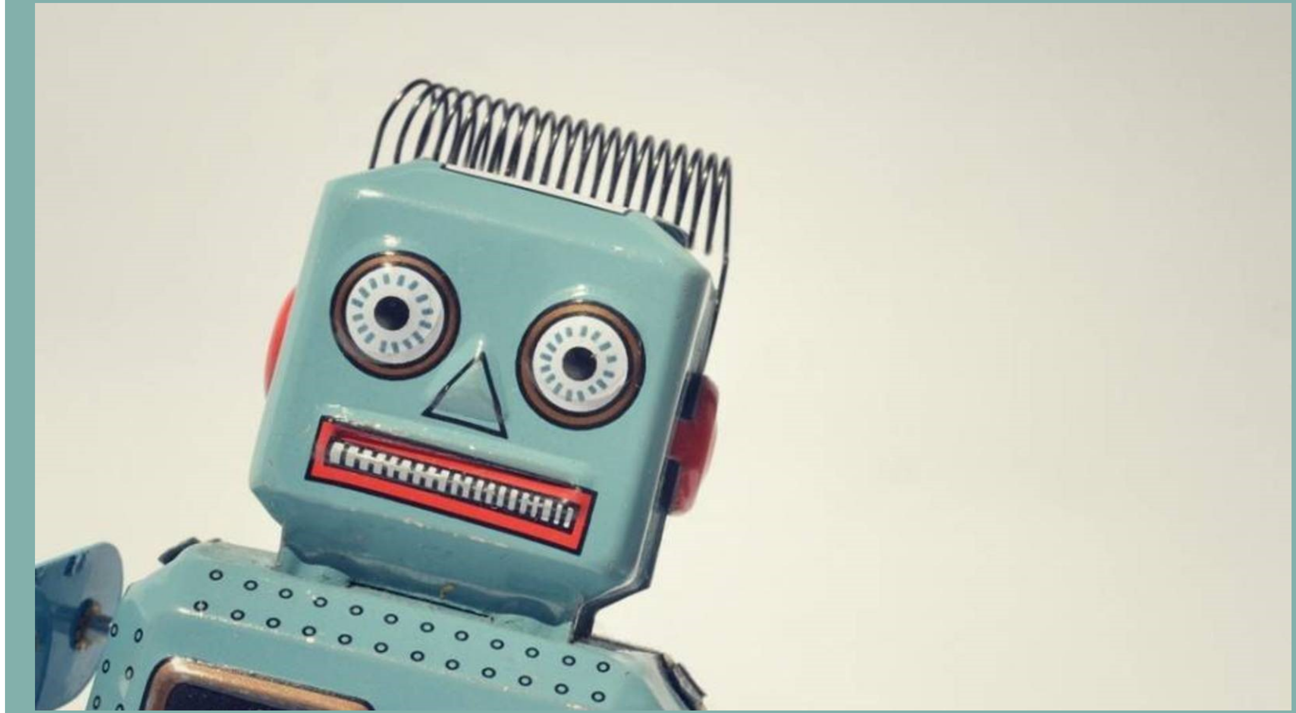


Table 3-3. Basic regular expression metacharacters, including wildcards, ranges, and closures

Operator	Behavior
.	Wildcard, matches any character
^abc	Matches some pattern *abc* at the start of a string
abc$	Matches some pattern *abc* at the end of a string
[abc]	Matches one of a set of characters
[A-Z0-9]	Matches one of a range of characters
ed|ing|s	Matches one of the specified strings (disjunction)
*	Zero or more of previous item, e.g., a*, [a-z]* (also known as *Kleene Closure*)
+	One or more of previous item, e.g., a+, [a-z]+
?	Zero or one of the previous item (i.e., optional), e.g., a?, [a-z]?
{n}	Exactly *n* repeats where *n* is a non-negative integer
{n,}	At least *n* repeats
{,n}	No more than *n* repeats
{m,n}	At least *m* and no more than *n* repeats
a(b|c)+	Parentheses that indicate the scope of the operators

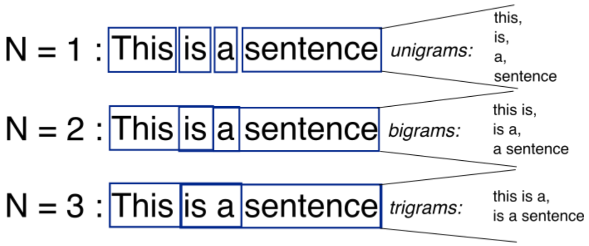
REPRESENTING TEXT //



REPRESENTING TEXT//

Discrete features

N-gran features



N = 1 : This is a sentence unigrams: this, is, a, sentence
N = 2 : This is a sentence bigrams: this is, is a, a sentence
N = 3 : This is a sentence trigrams: this is a, is a sentence

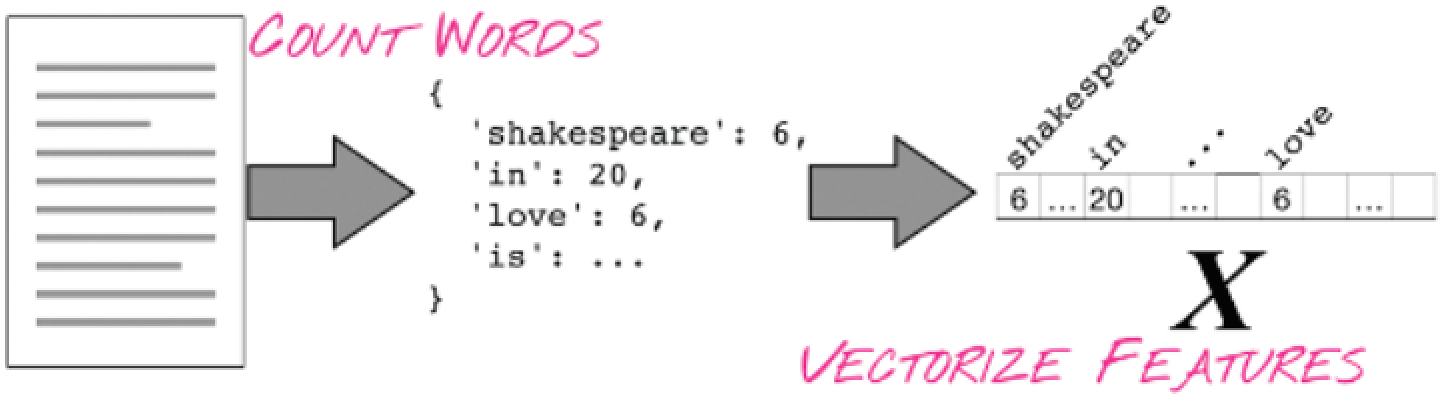
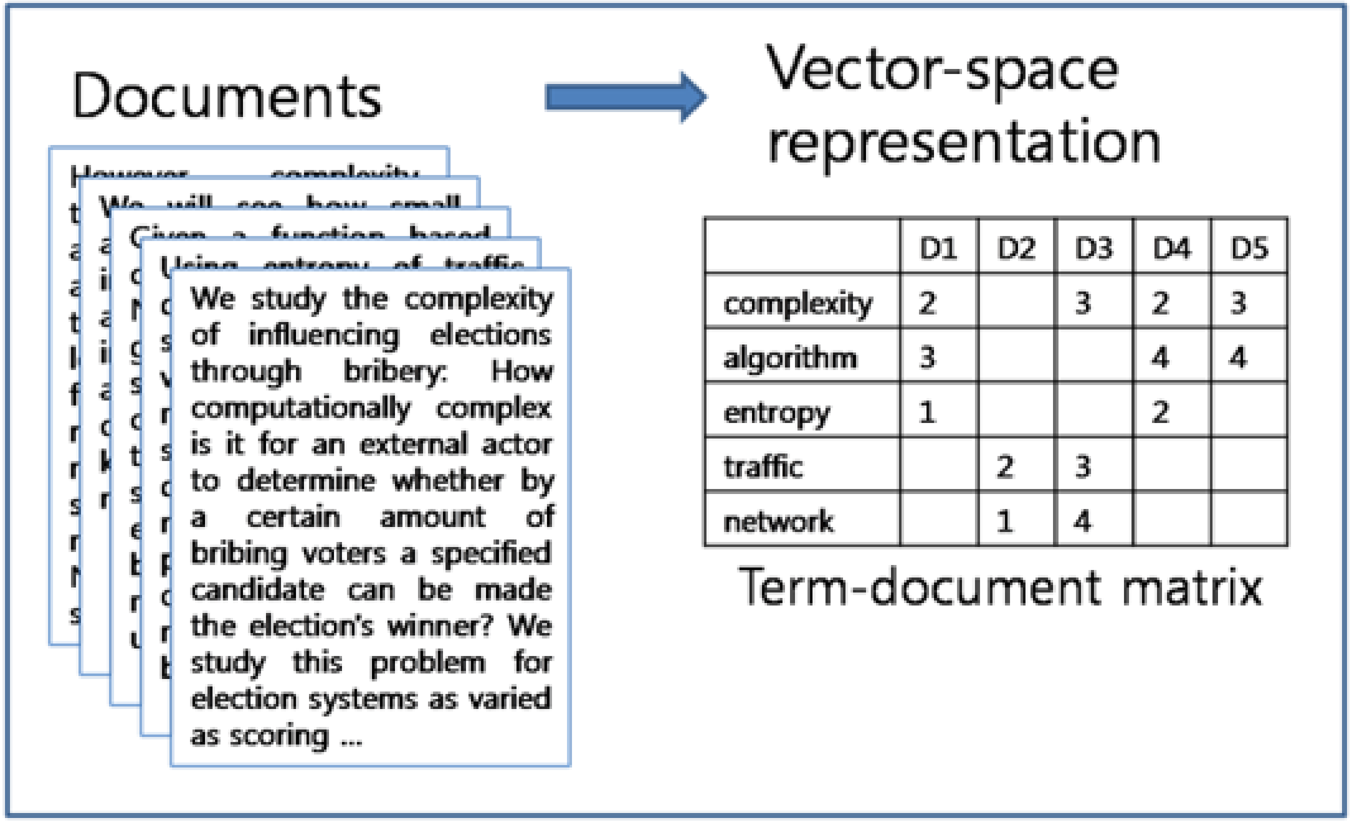


Figure 4 Schematic of a bag-of-words representation.

The Term-Document matrix

A **document-term matrix** is a mathematical matrix that describes the frequency of terms that occur in a collection of documents. In a document-term matrix, rows correspond to documents in the collection and columns correspond to terms.



We study the complexity of influencing elections through bribery: How computationally complex is it for an external actor to determine whether by a certain amount of bribing voters a specified candidate can be made the election's winner? We study this problem for election systems as varied as scoring ...

	D1	D2	D3	D4	D5
complexity	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

Term-document matrix

REPRESENTING TEXT//

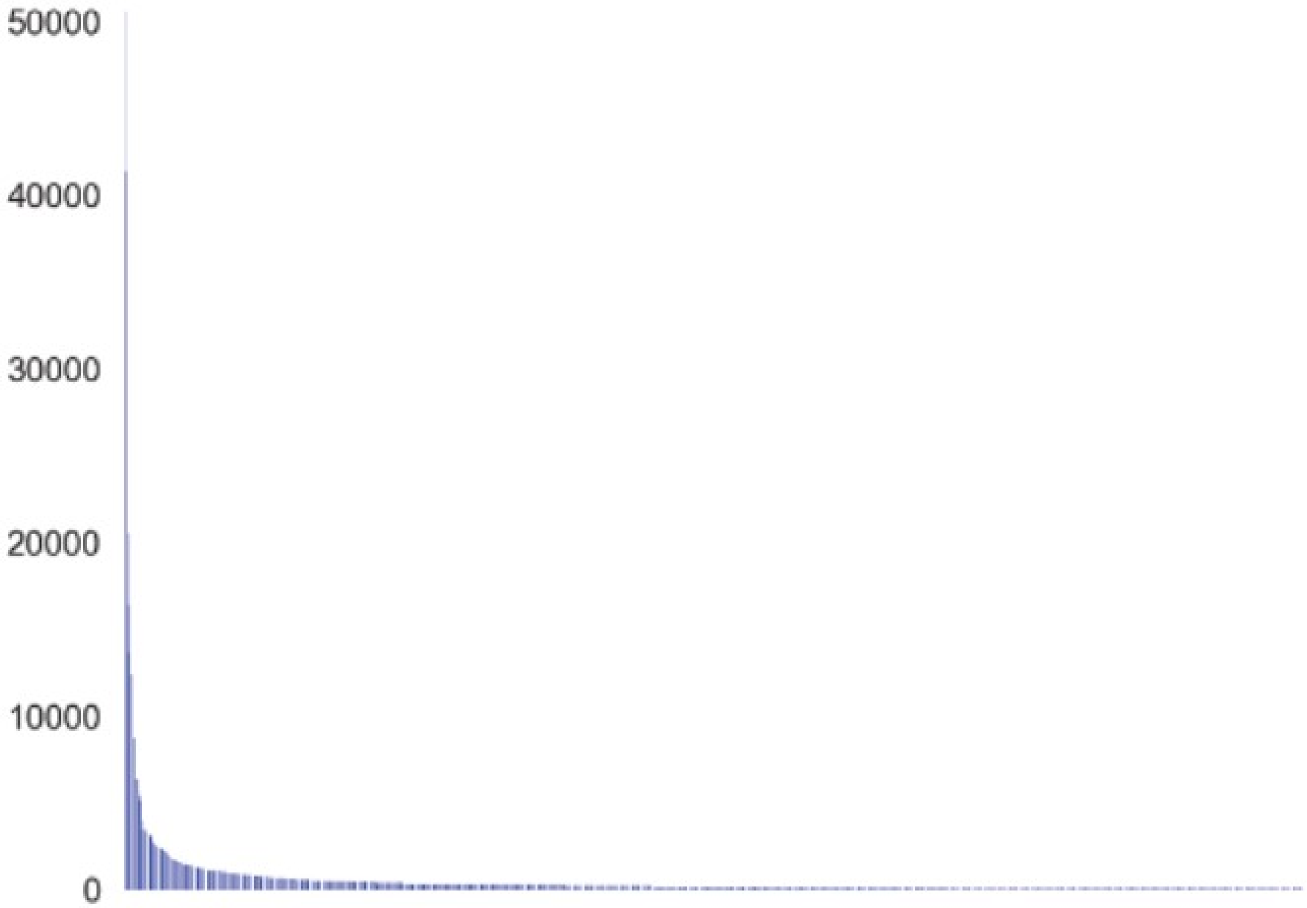
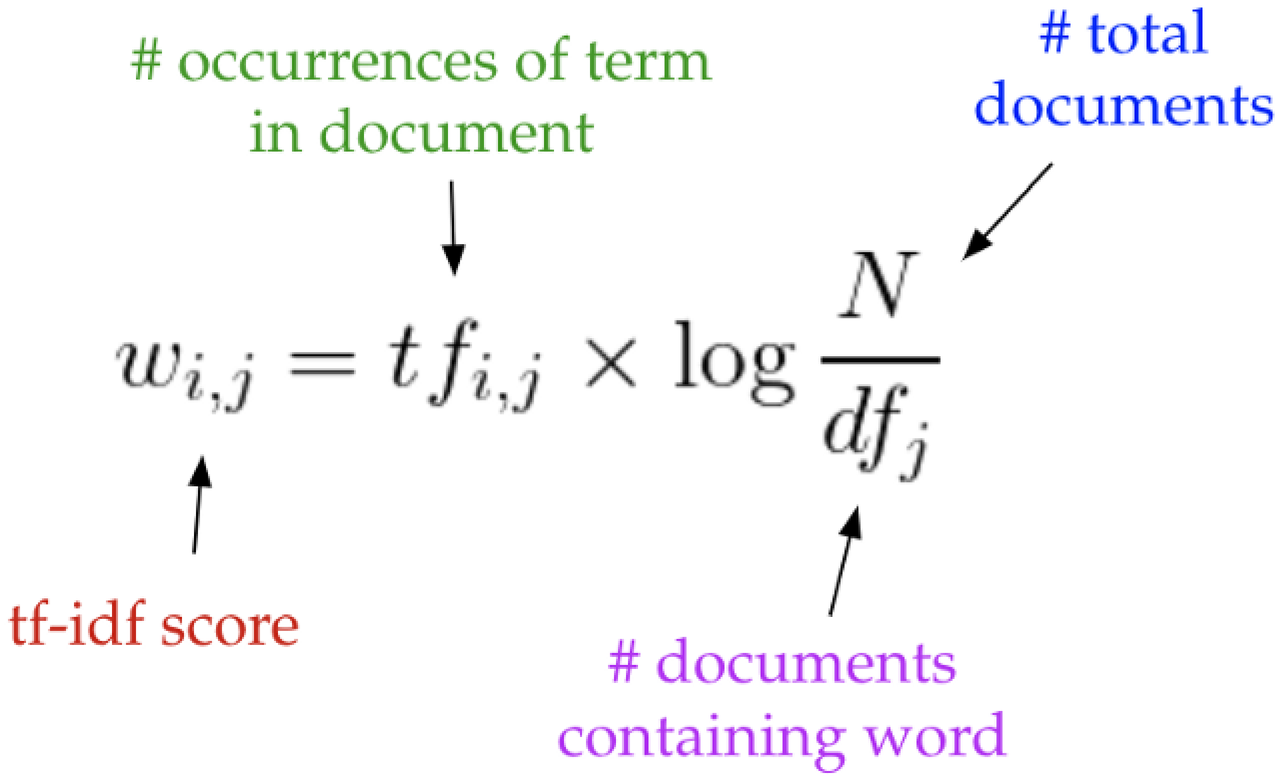


Figure 3 Frequency distribution of the top 1,000 words in a random sample of tweets, following Zipf's law.

TF-IDF Counts

The **term frequency–inverse document frequency (tf-idf)**, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.



occurrences of term in document

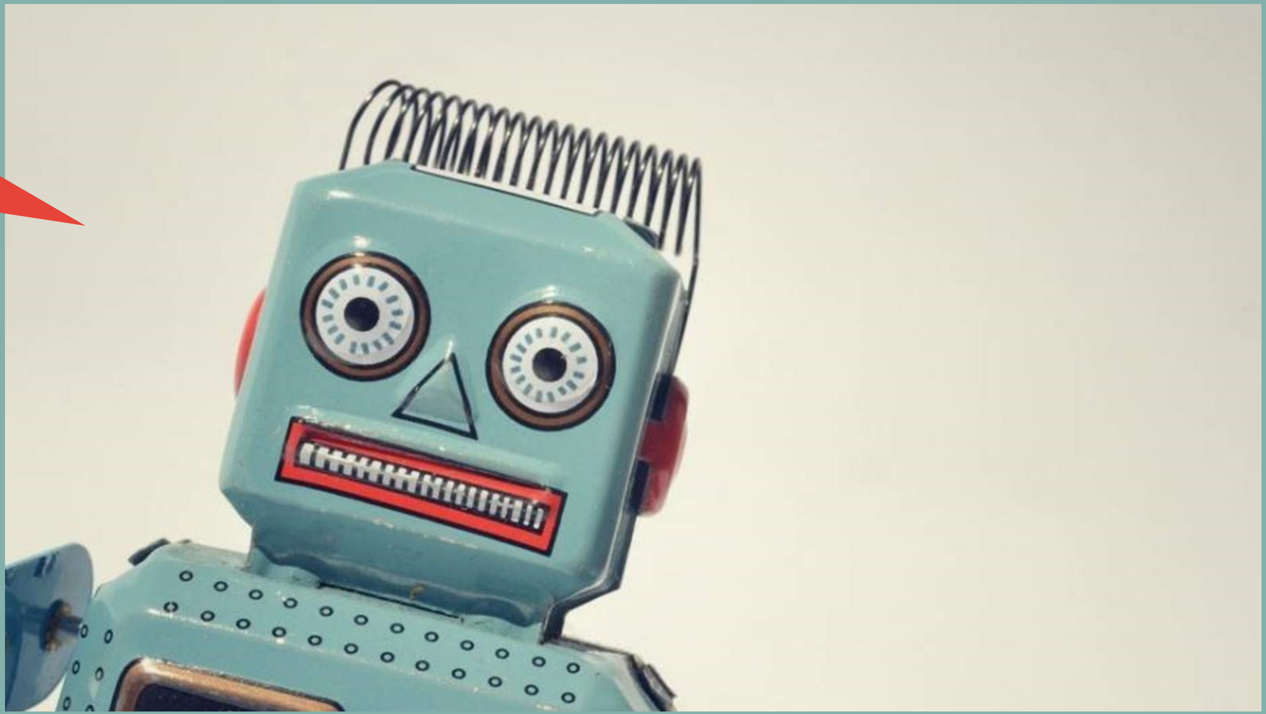
total documents

$$w_{i,j} = tf_{i,j} \times \log \frac{N}{df_j}$$

tf-idf score

documents containing word

LET’S CODE!



WE'LL BE BACK IN 15 MIN...