

# Week 1 lab

## Text processing exercises

This lab exercises your ability to use your chosen programming language to perform basic string processing and counting.

I also hope you begin to consider some of the interesting and challenging features of language.

## Corpus preparation

Handling a collection of text (a **corpus**) begins by collecting the documents that constitute that corpus and breaking them into small units such as sentences and words for further processing. **Tokenisation** refers to this processing of breaking a text into words and meaningful punctuation (each word or punctuation unit is called a **token**). The case of sentences is handled by **sentence boundary detection**.

Make a simple tokenizer which:

- removes punctuation from a string (in Python, the `string` module provides a list of punctuation)
- otherwise splits text on white space (common programming languages have a `split` function or method)

## Collocations and the t-test

The **collocations** of a term are the terms that appear remarkably often in its context. We will use the t-test to identify collocations, in this case more specifically pairs of words that, frequently next to each other, may constitute a **multi-word expression**. Process the [Pride and Prejudice corpus](https://canvas.sydney.edu.au/courses/2437/files/962870/download?wrap=1)

(<https://canvas.sydney.edu.au/courses/2437/files/962870/download?wrap=1>) 

(<https://canvas.sydney.edu.au/courses/2437/files/962870/download?wrap=1>) to:

1. **Count** how often each pair of words appears together. You may want to use `collections.Counter` in Python or `table` in R. Then rank pairs of words by how often they appear together (raw frequency).

2. Rank pairs of words by their score under the t-test:

$$t = \frac{P(w_1=x, w_2=y) - P(w_1=x)P(w_2=y)}{\sqrt{P(w_1=x, w_2=y)/N}}$$

(Use maximum likelihood estimate of probability, i.e. relative frequency of the word in the corpus.  $N$  is the number of words in the corpus.)

3. Examine at the top 100 words produced by each method. What word pairs do they prefer, respectively? Do these rankings match your expectations?
4. Examine the ranking of word pairs beginning with words of interest, such as *may* or *good* or *take*.

5. Note that other *association measures* tend to be preferred for identifying collocations, notably Dunning's log-likelihood, Chi-square and pointwise mutual information. See [Manning and Schütze chapter 5](http://nlp.stanford.edu/fsnlp/promo/colloc.pdf) (<http://nlp.stanford.edu/fsnlp/promo/colloc.pdf>). Apart from the choice of association measure, the ranking can be affected by other choices, such as how we tokenize and clean the text. How? (Try some of these extensions if you wish!)
6. Homework: Rather than stripping punctuation, use a pre-existing tokeniser for English. How does this affect output?
  - Established utilities in [NLTK](http://nltk.org) (<http://nltk.org>) (Python), [OpenNLP](https://opennlp.apache.org/) (<https://opennlp.apache.org/>) (Java), [LingPipe](http://alias-i.com/lingpipe/) (<http://alias-i.com/lingpipe/>) (Java), etc.
  - [NLTK: Processing Raw Text](http://www.nltk.org/book/ch03.html) (<http://www.nltk.org/book/ch03.html>)
  - `nltk.tokenize` [API reference](http://www.nltk.org/api/nltk.tokenize.html) (<http://www.nltk.org/api/nltk.tokenize.html>)
7. Homework: Apply a list of stop-words (many English stop-word lists can be found online) to select only word-pairs that include content words.


## Example solutions

[On GitHub Enterprise](https://github.sydney.edu.au/COMP5046-Natural-Language-Processing/comp5046-labs-2018/tree/master/lab01) (<https://github.sydney.edu.au/COMP5046-Natural-Language-Processing/comp5046-labs-2018/tree/master/lab01>)

## Extension: Markov generation of abstracts

Implement a Markov chain text generator to produce pseudo-text like astrophysics abstracts: generate each word by drawing from the empirical probability distribution of words which follow the previous word (or two words). I.e. draw each word  $x$  according to the empirical distribution  $P(x|w_{t-1} = y)$  where  $y$  is the previous word.

See:

- [Nature: Gibberish Papers](http://www.nature.com/news/publishers-withdraw-more-than-120-gibberish-papers-1.14763) (<http://www.nature.com/news/publishers-withdraw-more-than-120-gibberish-papers-1.14763>)
- <http://pdos.csail.mit.edu/scigen/> (<http://pdos.csail.mit.edu/scigen/>)
- [Astro-ph collection](http://arxiv.org/archive/astro-ph) (<http://arxiv.org/archive/astro-ph>)
- [Astro-ph abstracts corpus](https://canvas.sydney.edu.au/courses/2437/files/962790/download?wrap=1) (<https://canvas.sydney.edu.au/courses/2437/files/962790/download?wrap=1>)  (<https://canvas.sydney.edu.au/courses/2437/files/962790/download?wrap=1>)