# Text Pre-Processing — 2



Faculty of Information Technology, Monash University, Australia

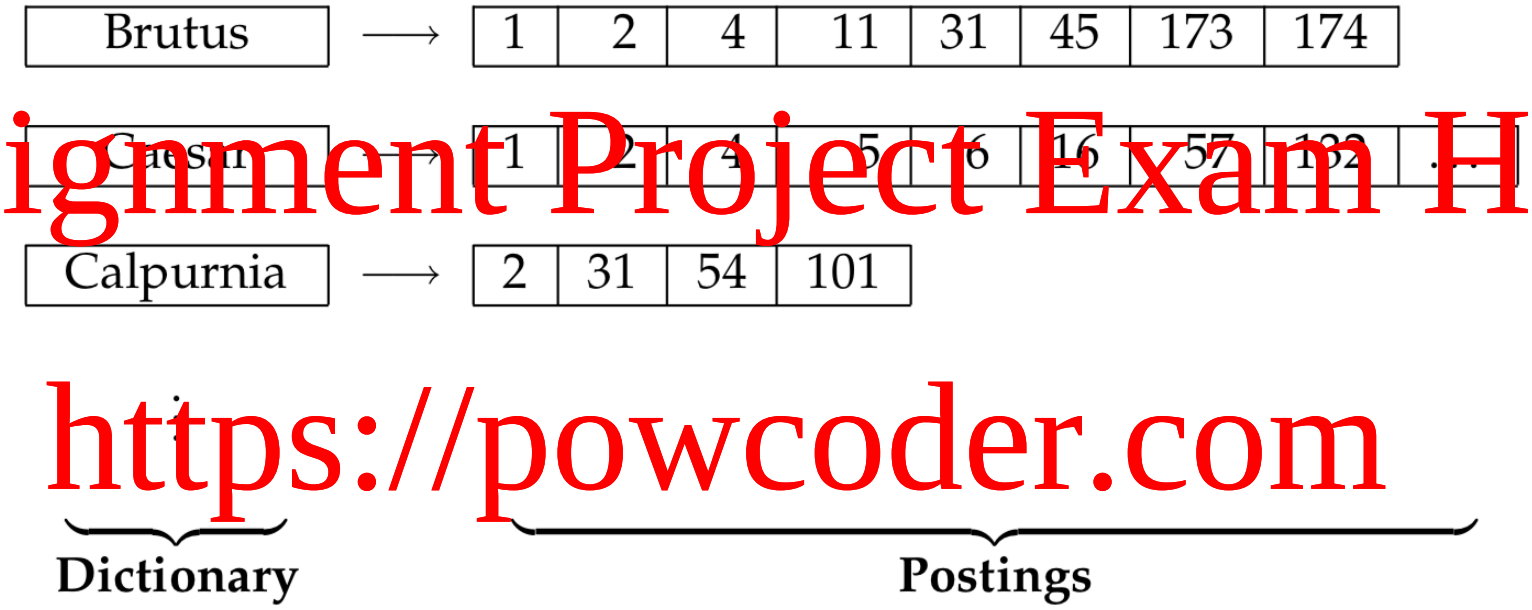


FIT5196 week 5

# Outline

# Inverted Index










▶ **Figure 1.3** The two parts of an inverted index. The dictionary is commonly kept in memory, with pointers to each postings list, which is stored on disk.

Figure: This figure is adopted from the book called "Introduction to Information Retrieval", which can be viewed here http://nlp.stanford.edu/IR-book/

- A dictionary of terms (referred to as a vocabulary or lexicon)
- Each term is associated with a list that records which documents the term occurs in.

# Inverted Index



To build an inverted index

1. Collect the documents to be indexed.
2. Tokenize the text, turning each document into a list of tokens
3. Do linguistic preprocessing, producing a list of normalized tokens, which are the indexing terms
4. Index the documents that each term occurs in by creating an inverted index, consisting of a dictionary and postings

# Inverted Index



**Doc 1**
I did enact Julius Caesar: I was killed i' the Capitol; Brutus killed me.

**Doc 2**
So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious:

| term | docID |
|---|---|
| I | 1 |
| did | 1 |
| enact | 1 |
| julius | 1 |
| caesar | 1 |
| I | 1 |
| was | 1 |
| killed | 1 |
| i' | 1 |
| the | 1 |
| capitol | 1 |
| brutus | 1 |
| killed | 1 |
| me | 1 |
| so | 2 |
| let | 2 |
| it | 2 |
| be | 2 |
| with | 2 |
| caesar | 2 |
| the | 2 |
| noble | 2 |
| brutus | 2 |
| hath | 2 |
| told | 2 |
| you | 2 |
| caesar | 2 |
| was | 2 |
| ambitious | 2 |

⟹

| term | docID |
|---|---|
| ambitious | 2 |
| be | 2 |
| brutus | 1 |
| brutus | 2 |
| capitol | 1 |
| caesar | 1 |
| caesar | 2 |
| caesar | 2 |
| did | 1 |
| enact | 1 |
| hath | 1 |
| I | 1 |
| I | 1 |
| i' | 1 |
| it | 2 |
| julius | 1 |
| killed | 1 |
| killed | 1 |
| let | 2 |
| me | 1 |
| noble | 2 |
| so | 2 |
| the | 1 |
| the | 2 |
| told | 2 |
| you | 2 |
| was | 1 |
| was | 2 |
| with | 2 |

⟹

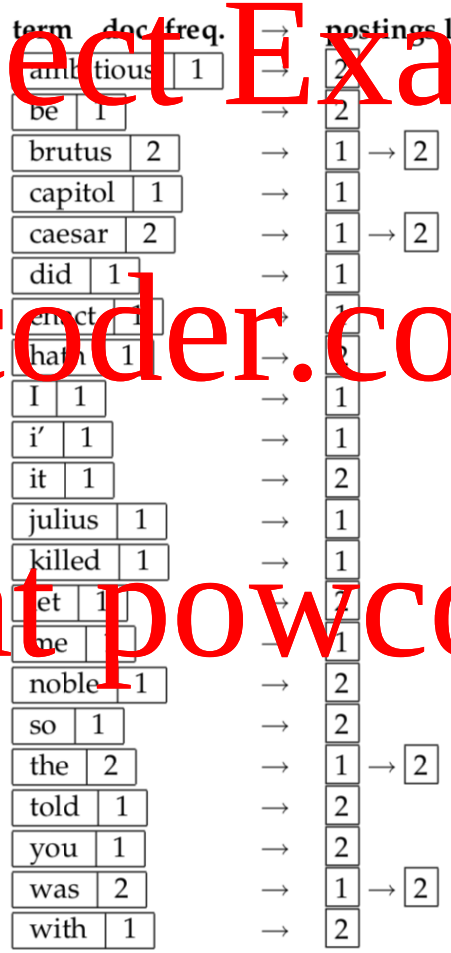



Figure: This figure is adopted from the book called "Introduction to Information Retrieval", which can be viewed here

# Vector Space Model



- VSM: each text document is represented as a vector where the elements of the vector indicate the occurrence of words within the text

$$V(d) = [w_{1,d}, w_{2,d}, w_{3,d}, \ldots, w_{N,d}]$$

where

  - $N$: the size of the vocabulary
  - $w_{n,d}$: the weight of the $n$-th term (in the vocabulary) in document $d$

- Examples:
  - document_1: "Data analysis is important."
  - document_2: "Data wrangling is as important as data analysis."
  - document_3: "Data science contains data analysis and data wrangling."

# Vector Space Model



- Examples:
  - document_1: "Data analysis is important."
  - document_2: "Data wrangling is as important as data analysis."
  - document_3: "Data science contains data analysis and data wrangling."
- Vector representation



| | 'analysis' | 'and' | 'as' | 'contains' | 'data' | 'important' | 'is' | 'science' | 'wrangling' |
|---|---|---|---|---|---|---|---|---|---|
| document_1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| document_2 | 1 | 0 | 2 | 0 | 2 | 1 | 1 | 0 | 1 |
| document_3 | 1 | 1 | 0 | 1 | 3 | 0 | 0 | 1 | 1 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| document_1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| document_2 | 1 | 0 | 2 | 0 | 2 | 1 | 1 | 0 | 1 |
| document_3 | 1 | 1 | 0 | 1 | 3 | 0 | 0 | 1 | 1 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| document_1 | 0 | 0 | 0 | 0 | 0 | 0.176 | 0.176 | 0 | 0 |
| document_2 | 0 | 0 | 0.954 | 0 | 0 | 0.176 | 0.176 | 0 | 0.176 |
| document_3 | 0 | 0.477 | 0 | 0.477 | 0 | 0 | 0 | 0.477 | 0.176 |

# Vector Space Model — Euclidean Distance

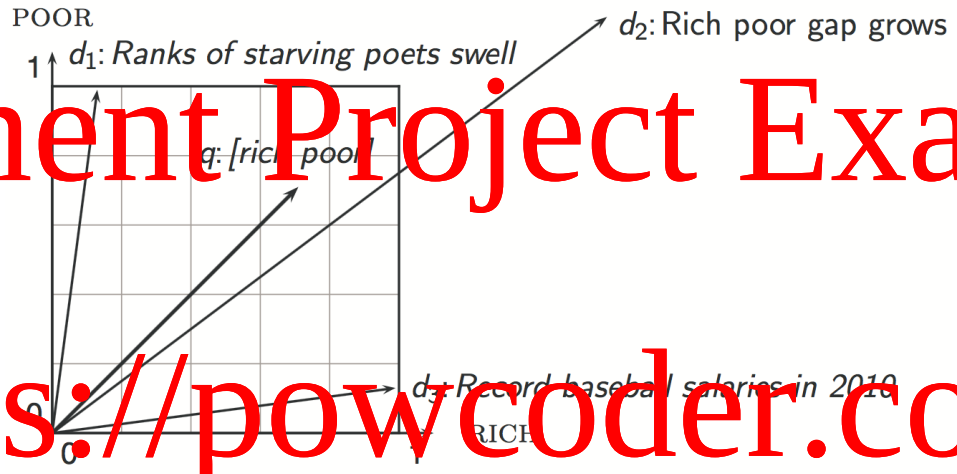





Figure: This figure is adopted from the lecture slides on https://www.cl.cam.ac.uk/teaching/1314/InfoRtrv/lecture4.pdf

# Vector Space Model — Cosine Similarity



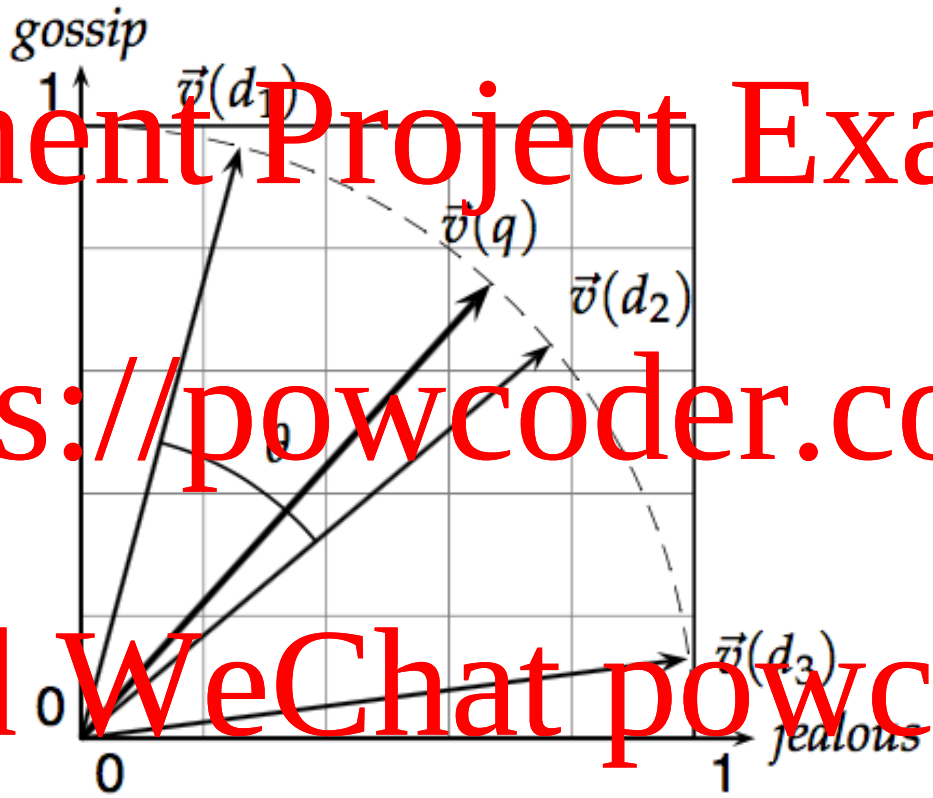
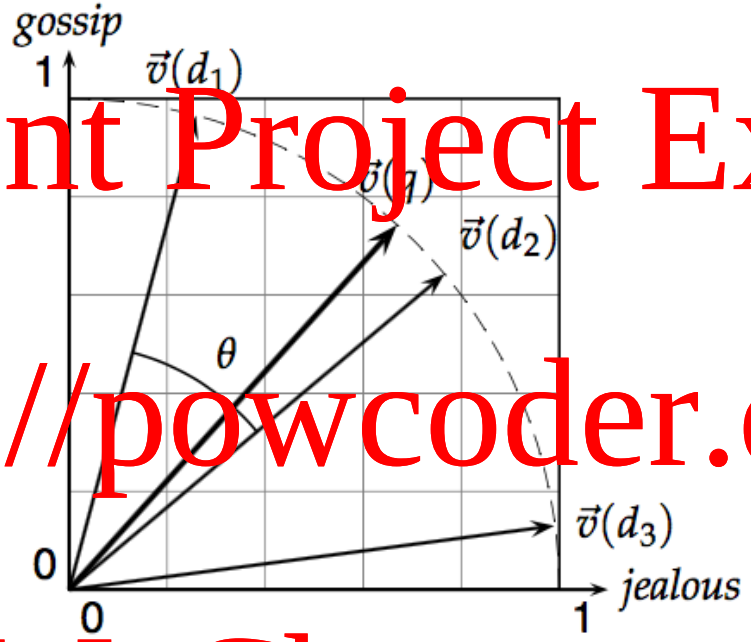



Figure: This figure is adopted from the book called "Introduction to Information Retrieval", which can be viewed here http://nlp.stanford.edu/IR-book/

# Vector Space Model — Cosine Similarity










$$sim(d_1, d_2) = \frac{V(d_1) \cdot V(d_2)}{|V(d_1)||V(d_2)|}$$

$$= \frac{\sum_{n=1}^{N} V_n(d_1) V_n(d_2)}{\sqrt{\sum_{n=1}^{N} V_n^2(d_1)} \sqrt{\sum_{n=1}^{N} V_n^2(d_2)}}$$

# Vector Space Model — Cosine Similarity



$$sim(d_1, d_2) = \frac{V(d_1) \cdot V(d_2)}{|V(d_1)||V(d_2)|} = \frac{\sum_{n=1}^{N} V_n(d_1) V_n(d_2)}{\sqrt{\sum_{n=1}^{N} V_n^2(d_1)} \sqrt{\sum_{n=1}^{N} V_n^2(d_2)}}$$



- Notation:
  - $V_n(d)$: the weight of the n-th vocabulary term in document $d$
  - $\cdot$: indicates inner product
  - $|V(d)|$: Euclidean length
- The cosine similarity of $d_1$ and $d_2$ is equivalent to the cosine of the angle between $d_1$ and $d_2$.
  - Cosine is a monotonically decreasing function of the angle for the interval $[0°, 180°]$

# Vector Space Model — Bag of Words



- The Bag-of-Words assumption:
  - The order of the words in a document does not matter.
  - Not a problem for many text analysis tasks, such as document classification and clustering
    - A collection of words is usually sufficient to differentiate between semantic concepts of documents.
  - Not a universal solution for, such as information extraction, POS tagging and many other NLP tasks, where the order of words does matter.



- Consideration of using VSM:
  - The dimensionality of the vectors in VSM.
  - How the compute the weights of words in each document.
    - binary values
    - counts
    - TF-IDF weights

## Term Frequency

- TF: the number of occurrences of a word in a document.
- How well does a word represent its document?
  - frequent (non stop) words are thematic
  - if a words t appears often in a document, then a document containing t should be similar to that document.
- Different ways of computing TF

| weighting scheme | TF |
|---|---|
| binary | 0,1 |
| raw frequency | $f_{t,d}$ |
| log normalisation | $1 + log(f_{t,d})$ |
| double normalisation K | $K + (1 - K) \times \frac{f_{t,d}}{max(t' \in d) f_{t',d}}$ |

## Term Frequency

- TF: the number of occurrences of a word in a document.
- How well does a word represent its document?
  - frequent (non stop) words are thematic
  - if a words t appears often in a document, then a document containing t should be similar to that document.
- Problems:
  - All words are considered equally important.
  - Certain terms have little or no discriminating power in, for example, text classification.
    - All the medical documents contain words "patient" and "disease"
    - All the patents contain words like "claim", "embodiment", etc.
  - Can we scale down the term weights of terms ?

# Inverse Document Frequency



- IDF: Inverse document frequency
  - The idf of a rare term is high, whereas the idf of a frequent term is likely to be low

$$idf_t = log\left(\frac{D}{n_t}\right)$$

where $D$ is the total number of documents in the corpus, and $n_t = |\{d \in D : t \in d\}|$



  - if a word/term appears in every document, $n_t = D$, then $idf_t = 0$
  - if a word/term appears in a document, $n_t = 1$, then $idf_t = log(D)$
- Different ways of computing IDF



| weighting scheme | IDF weight |
|---|---|
| IDF | $log\left(\frac{D}{n_t}\right)$ |
| IDF smoothed | $log\left(1 + \frac{D}{n_t}\right)$ |
| IDF max | $log\left(1 + \frac{max_{t' \in d} n_{t'}}{n_t}\right)$ |
| IDF probability | $log\left(\frac{D - n_t}{n_t}\right)$ |

# TF-IDF

- The TF-IDF weight schema



$$tf\text{-}idf_t = tf_t \times idf_t = n_{d,t} \times log\left(\frac{D}{n_t}\right)$$



where $n_{d,t}$: #occurrences of word $t$ in document $d$, $m_t$: the document frequency of term $t$.

- most frequent words
- less frequent words

# Colllocations

- Collocations: multi-word expressions consisting of two or more words that occur more frequently than by chance, and correspond to some conventional way of saying things.
  - Noun phrases: “strong tea” and “weapons of mass destruction”
  - Verb phrases: “make up”, “wind up”
- Collocations are characterised by limited compositionality.
  - Non-compositional phrases:
    - Difficult to predict the meaning of collocation from the meaning of its parts
    - Add an element of meaning to the combination
    - Example:

      "He is known for his fair and square dealings and everybody trusts his work"[1]

      "I had a shooting pain near the heart."



[1] Example from http://language.worldofcomputing.net/nlp-overview/collocations.html

# Colllocations

- Collocations: multi-word expressions consisting of two or more words that occur more frequently than by chance, and correspond to some conventional way of saying things.
  - Noun phrases: “strong tea” and “weapons of mass destruction”
  - Verb phrases: “make up”, “wind up”
- Collocations are characterised by limited compositionality.
  - Compositional phrases
    - the meaning of the expression can be predicted from the meaning of the parts.
    - Example:

      "I had a stomach pain yesterday."

# Colllocations

- Collocations: multi-word expressions consisting of two or more words that occur more frequently than by chance, and correspond to some conventional way of saying things.
  - Noun phrases: “strong tea” and “weapons of mass destruction”
  - Verb phrases: “make up”, “wind up”
- Collocations are characterised by limited compositionality.
- How to extract collocation
  - NLTK Collocation package
  - Simple tutorial: http://www.nltk.org/howto/collocations.html

## Summary: what do you need to do in this week?

- What we have covered:
  - Vector Space Model
  - Term weighting
  - Collocations
  - Count vocabulary
- What you need to do:
  - Download and read the chapters provided in Moodle
  - Attend tutorial 5 next week.
  - Remember: **assessment 1** is due next week.