# 人工智慧專題

劉瓊如

September 23, 2022

## What is Data Mining

**Definition (Wiki)**

Data mining is the process of extracting and discovering patterns in large data sets involving methods at the intersection of machine learning, statistics, and database systems.

簡而言之：從一包數據集中抽取“有用”的資訊

## How to do Data Mining

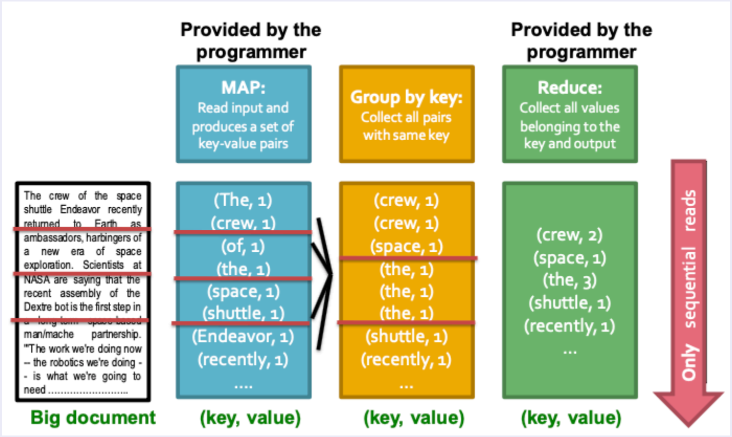
- Store (system)
- Management (databases)
- Analysis (this class)

## Data Mining Methods

- Find human-interpretable patterns that describe the data
  - Example: Statistics, Clustering
- Use some variables to predict unknown or future values of other variables
  - Example: Recommender systems

## Examples

### Example (Counting words)



Provided by the programmer
MAP: Read input and produces a set of key-value pairs
Group by key: Collect all pairs with same key
Provided by the programmer
Reduce: Collect all values belonging to the key and output
The crew of the space shuttle Endeavor recently returned to Earth as ambassadors, harbingers of a new era of space exploration. Scientists at NASA are saying that the recent assembly of the Dextre bot is the first step in a long-term space-based man/mache partnership. "The work we're doing now -- the robotics we're doing -- is what we're going to need ........................
(The, 1) (crew, 1) (of, 1) (the, 1) (space, 1) (shuttle, 1) (Endeavor, 1) (recently, 1) ....
(crew, 1) (crew, 1) (space, 1) (the, 1) (the, 1) (the, 1) (shuttle, 1) (recently, 1) ...
(crew, 2) (space, 1) (the, 3) (shuttle, 1) (recently, 1) ...
Only sequential reads
Big document
(key, value)
(key, value)
(key, value)

Figure: *Word Counting*

## How to Extract knowledge?

評估方式：

- Term frequency: 字頻愈多愈好嗎？
  Examples: the, it, at, ··· 出現愈多愈重要嗎？
- Inverse document frequency
  Examples: “the” 在每個文本都頻繁出現，那是否一個字的重要性應與“它在文本出現的次數成反比”
  那一個字的重要性是否來自於它的“稀有性”

## TF-IDF

- Term frequency (字頻）is the relative frequency of term t within document d

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}},$$

  where $f_{t,d}$ is the raw count of a term in a document

- Inverse document frequency

$$idf(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|},$$

  where

  - N: total number of documents in the corpus $N = |D|$
  - $|\{d \in D : t \in d\}|$: number of documents where the term t appears

## TF-IDF

$$tf - idf(t, d, D) = tf(t, d) \cdot idf(t, D)$$

### Example

| *Term* | *this* | *is* | *an* | *apple* |
|---|---|---|---|---|
| *Term Count* | *1* | *1* | *2* | *1* |

Table: *Document 1*

| *Term* | *this* | *is* | *a* | *cat* |
|---|---|---|---|---|
| *Term Count* | *1* | *3* | *2* | *1* |

Table: *Document 2*

## TF-IDF

$$tf(\text{"}this\text{"}, d_1) = \frac{1}{5}$$

$$tf(\text{"}this\text{"}, d_2) = \frac{1}{7}$$

$$idf(\text{"}this\text{"}, D) = \log(\frac{2}{2}) = 0$$

$$TF - IDF(\text{"}this\text{"}, d_1, D) = \frac{1}{5} \cdot 0 = 0$$

$$TF - IDF(\text{"}this\text{"}, d_2, D) = \frac{1}{7} \cdot 0 = 0$$

## TF-IDF

$$TF(\text{"}apple\text{"}, d_1) = \frac{1}{5}$$
$$TF(\text{"}apple\text{"}, d_2) = \frac{0}{7} = 0$$
$$IDF(\text{"}apple\text{"}, D) = \log(\frac{2}{1}) = \log 2$$

$$TF - IDF(\text{"}apple\text{"}, d_1, D) = \frac{1}{5} \cdot (\log 2) = \log 2/5$$
$$TF - IDF(\text{"}apple\text{"}, d_2, D) = 0 \cdot (\log 2) = 0$$

tf-idf 結論：Term $t$ 詞頻愈高愈好，但須具有稀有性．若包含 $t$ 的文本愈多，區分度愈低，愈不重要．

## 關鍵字：萃取方式

Question: TF-IDF 是否是唯一的評估方式？
Answer：不是，TF-PDF、TF-IDuF、OKapi BM25、YAKE （統計方法）
Question: 是否有不是基於統計的 knowledge extraction？且是 rule-based
Answer: 有，Text-Rank 2004, Rake, Singlerank
A survey conducted in 2015 showed that 83% of text-based recommender systems in digital libraries use tf–idf

## OKapi BM25

BM=Best Matching
Okapi BM25 (BM is an abbreviation of best matching) is a ranking function used by search engines to estimate the relevance of documents to a given search query. It is based on the probabilistic retrieval framework developed in the 1970s and 1980s by Stephen E. Robertson, Karen Spärck Jones, and others.

## OKapi BM25

**Theorem**

*Given a query Q, containing keywords $q_1, \cdots, q_n$, the BM25 score of document D is:*

$$\text{score}(D, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{\text{avgd1}})}$$

*where $f(q_i, D)$ is $q_i$'s term frequency in the document D, $|D|$ is the length of document D in words, and avgdl is the average document length in the text collection from which documents are drawn. $k_1$ and b are free parameters, usually chosen, in absence of an advanced optimization, as $k_1 \in [1.2, 2]$, $b = 0.75$.*

$q_i$: 詢問（搜尋字），D：欲比對文本（搜尋結果）

## OKapi BM25

**Theorem**

$$\mathrm{IDF}(q_i) = \ln\left(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1\right)$$

*is the IDF (inverse document frequency) weight of the query term $q_i$, where $N$ is the total number of documents in the collection, and $n(q_i)$ is the number of documents containing $q_i$.*

- $b = 1$ BM11，完全使用文本長度的權重．
- $b = 0$ BM15，不使用文本長度．
- $b$ 是用來調整文本長度估評分重要性的參數．以相同的詞頻，文本愈長，分數愈低
- $k_1$ 用來正規劃文本詞頻的範圍．$k_1 = 0$，則詞頻無影響力．

## 作業一

自行設定一個 query，在搜尋引擎上搜尋，取前三篇文章（不包括廣告）· 計算 3 documents 中，tf-idf 分數最高的 top-5，獲取"關鍵字"

The problems that you may encounter:

- 如何存資料、在那裡做計算、如何產生 output
- 斷字: Jieba (pip install jieba) jieba.cut
- (key, value)

## Value of Web Pages

- 網頁的重要性：visit counts 流量
- 流量是一個為網頁打分數的方法？如何監督每個網頁
- visit a page: follow links randomly

## Page Rank

**Definition**

PageRank (PR) is an algorithm used by Google Search to rank web pages in their search engine results. It is named after both the term ”web page” and co-founder Larry Page. PageRank works by counting the number and quality of links to a page to determine a rough estimate of the website’s importance.

網頁的” 分數 “: 計算連結數

Links as votes

## Page Rank

令 u 是一個網頁, $d_u$: degree of $u$ (#out-links)

**Simple ranking**

$$R(u) = c \sum_{v \to u} \frac{R(v)}{d_v}$$

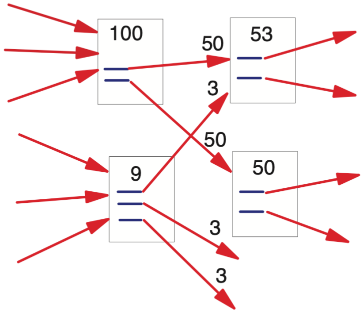


Figure: Simple Page Rank

## Page Rank: Matrix Formulation

以向量方式表示：令 A 為 web page 的 Stochastic adjacency matrix

$$A_{uv} = \begin{cases} \frac{1}{d_v} & \text{there is a link from v to u} \\ 0 & \text{if not} \end{cases}$$

If R is the vector of page rank, then

$$\mathbf{R} = cA\mathbf{R}$$

**R** is an eigenvector of A and c is the corresponding eigenvalue．

## Page Rank: Linking Problem

### Example

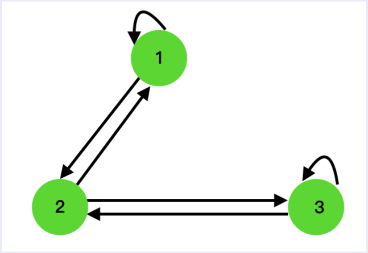


Figure: *Page Rank: Matrix Formulation*

## Solution

Let $A = \begin{pmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1/2 \\ 0 & 1/2 & 1/2 \end{pmatrix}$.

Solve $\begin{vmatrix} 1/2-\lambda & 1/2 & 0 \\ 1/2 & -\lambda & 1/2 \\ 0 & 1/2 & 1/2-\lambda \end{vmatrix} = -\frac{1}{2}(\lambda - \frac{1}{2})(2\lambda+1)(\lambda-1)$.

There are three cases,

- $\lambda = \frac{1}{2}$, $\mathbf{R} = t(1,0,0)$, for any $t \in \mathbb{R}$
- $\lambda = -\frac{1}{2}$, $\mathbf{R} = t(1,-2,1)$, for any $t \in \mathbb{R}$
- $\lambda = 1$, $\mathbf{R} = t(1,1,1)$, for any $t \in \mathbb{R}$.

Additional constrain foreces uniqueness $R_1 + R_2 + R_3 = 1$

## Large Web-Size Graphs

Encounter problem:

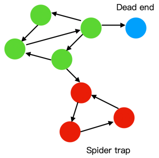
- Large adjacency matrix->Find eigenvectors (Existence)
- $\mathbf{R} = \lambda A\mathbf{R}$-> $\mathbf{R}$ is a stable vector
- idea: random $\mathbf{R}_0$, If $\lim_{n\to\infty} A^n\mathbf{R}_0 = \mathbf{r}$ exists, then $A\mathbf{r} = \mathbf{r}$.
- Power iteration
  - Let $\mathbf{R}_0 = (1/N, \cdots, 1/N)$
  - $\mathbf{R}_{n+1} = A\mathbf{R}_n$
  - Stop when $|\mathbf{R}_{n+1} - \mathbf{R}_n|_1 < \epsilon$

## Why Power iteration works?

- Suppose $A$ is an $N \times N$ matrix and has $N$ independent eigenvectors $x_1, x_2, \cdots, x_N$ with corresponding eigenvalues $\lambda_1, \lambda_2, \cdots, \lambda_N$, where $\lambda_1 > \lambda_2 > \cdots > \lambda_N$
- $\mathbf{R}_0 = \sum_{i=1}^{N} c_i x_i$, $A\mathbf{R}_0 = \sum_{i=1}^{N} c_i \lambda_i x_i$
- After $n$ iteration, $A^n\mathbf{R}_0 = \sum_{i=1}^{N} c_i \lambda_i^n x_i = \lambda_1^n \sum_{i=1}^{N} c_i \left(\frac{\lambda_i}{\lambda_1}\right)^n x_i$
- $A^n\mathbf{R}_0 \approx c_1 \lambda_1^n x_1$
  If $c_1$, then this method won't converge

## Page Rank: Problem

- Dead ends: Some pages have no out-links
- Spider traps: All out-links are within the group



若一個人在網頁上 surfer

- With probability $\beta$, follow a link
- With probability $1-\beta$, jump to some random page $E$ (preference vector)

原先 $E$ 只是為了解決 rank sink 的問題．

$$R(u) = \beta \sum_{v \to u} \frac{R(v)}{d_v} + (1-\beta)E(u)$$

In matrix form $R = \beta AR + (1-\beta)E$

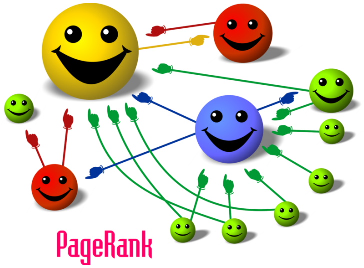
## 作業二

自行畫一個 5 個 node 的有向圖（directed graph），計算此圖中的 page rank 分數 Problems that you may encounter

- power iteration or solving by finding eigenvectors
- Is there a solution?
- Compute by hands or computer

## Directed Graph

- Nodes: Webpages
- Edges: Hyperlinks

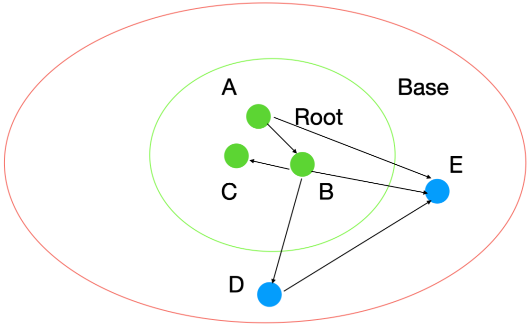


## Kleinberg's Algorithm 1997 (HITS = Hyperlink-Induced Topic Search)

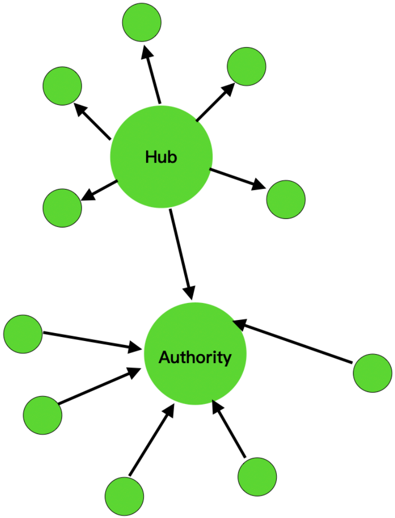
**Definition**

Kleinberg 認為網頁再給予一個主題 t 時，依照超連結的結構分成 hubs 和 authorities，$q$ 是一個 term-based search query，要在收集來的網頁裡對應 topic t

- Root set $S$：在搜尋引擎上搜尋 $q$ 會獲得的頁面（authorities 的候選）
- Base set $C = S \cup \{p | \exists p' \in S, p \to p'\} \cup \{p | \exists p'' \in S, p'' \to p\}$
- Link structure on $C$：$G = \{(C, E)\}$
- $E = \{(p, p') | p \to p'\}$: a directed edge （has a hyperlink from p to p'）
- $W \in \mathbb{R}^{|C| \times |C|}$ adjacent matrix of G


A
Root
Base
E
C
B
D

## Kleinberg's Mutual Reinforcement Approach



## Kleinberg's Mutual Reinforcement Approach

- Two distinct types of Web pages: hubs and authorities.
- a good hub will point to many authorities
- a good authority will be pointed at by many hubs
- $h(s)$: a hub weight of $s \in C$
- $a(s)$: a authority weight of $s \in C$
- $h(s) \propto \sum_{s \rightarrow p} a(p)$
- $a(s) \propto \sum_{p \rightarrow s} h(p)$

## Kleinberg's Algorithm (HITS)

- $a(s) \leftarrow 1,\ h(s) \leftarrow 1$
- $a(s) \leftarrow \sum_{p \to s} h(p) \iff a \leftarrow W^T h$
- $h(s) \leftarrow \sum_{s \to p} a(p) \iff h \leftarrow Wa$
- $a(s) \leftarrow a(s)/\sum_s a(s) \iff a \leftarrow a/\sum_i a_i$
- $h(s) \leftarrow h(s)/\sum_s h(s) \iff h \leftarrow h/\sum_i h_i$
- until converges

## Kleinberg's Algorithm (HITS)

- a 是 normalized $W^TW$ 的 principal eigenvector
- h 是 normalized $WW^T$ 的 principal eigenvector
- Let $A = W^TW$: cocitation matrix，$A_{ij}$ = page i 與 page j 共同來源 page 數

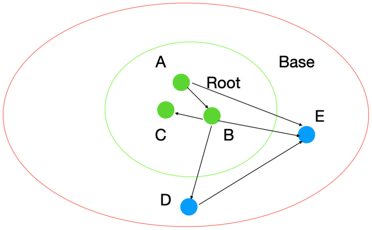
$$A_{ij} = \sum_{k=1}^{|C|} W^T_{ik} W_{kj} = \sum_{k=1}^{|C|} W_{ki} W_{kj}$$

- Let $H = WW^T$：bibliographic coupling matrix $H_{ij}$ = page i 與 page j 共同被索引 page 數

$$A_{ij} = \sum_{k=1}^{|C|} W_{ik} W^T_{kj} = \sum_{k=1}^{|C|} W_{ik} W_{jk}$$

## Example

*Let* $W = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ *be the adjacent matrix of 5 pages. Find the authority scores and hub scores for each page.*



## HITS iteration

Let $a_t$ denote the authority score of the $t$-th iteration and $h_t$ denote the hub score of the $t$-th iteration.

$$a_1 = W^T h_0 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 3 \end{pmatrix}$$

$$h_1 = W a_0 = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$a_1 \leftarrow \frac{1}{\|a_1\|_1} a_1 = \begin{pmatrix} 0 \\ 1/6 \\ 1/6 \\ 1/6 \\ 3/6 \end{pmatrix}, \ h_1 \leftarrow \frac{1}{\|h_1\|_1} h_1 = \begin{pmatrix} 2/6 \\ 3/6 \\ 0 \\ 1/6 \\ 0 \end{pmatrix}$$

## HITS iteration

$$a_2 = W^T h_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1/6 \\ 1/6 \\ 1/6 \\ 3/6 \end{pmatrix} = \begin{pmatrix} 0 \\ 1/3 \\ 1/2 \\ 1/2 \\ 1 \end{pmatrix}$$

$$h_2 = W a_1 = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1/6 \\ 1/6 \\ 1/6 \\ 3/6 \end{pmatrix} = \begin{pmatrix} 4/6 \\ 5/6 \\ 0 \\ 3/6 \\ 0 \end{pmatrix}$$

$$a_2 \leftarrow \frac{1}{\|a_2\|_1} a_2 = \begin{pmatrix} 0 \\ 1/7 \\ 3/14 \\ 3/14 \\ 3/7 \end{pmatrix}, \quad h_2 \leftarrow \frac{1}{\|h_2\|_1} h_2 = \begin{pmatrix} 4/12 \\ 5/12 \\ 0 \\ 3/12 \\ 0 \end{pmatrix}$$

Do you want to continue?

## META-ALGORITHM FOR LINK-STRUCTURE ANALYSIS

判斷 hubs 與 authorities page 的方法：

1. 給定主題 t，construct $C = \{t_{hubs}, t_{auths}\}$，$n = |C|$
2. Construct hub matrix $H$ and authorities matrix $A$，將
   $M$: 只有一個正 eigenvalue $\lambda(M)$ with multiplicity 1，且 $\lambda(M) > |\lambda'|$: $\lambda'$ any other eigenvalue
   $v_{\lambda(M)}$: unit eigenvector (principal eigenvector)
3. $k < n$: 像對於 $v_{\lambda(M)}$ 向量的 component 最大值的座標數：這形成 $C$ 裡的 principal algebraic community authorities (for $M = A$) or hubs (for $M = H$)

Let Hub matrix: $H = WW^T = \begin{pmatrix} 2 & 1 & 0 & 1 & 0 \\ 1 & 3 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

Authority matrix: $A = W^T W = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 3 \end{pmatrix}$

- The characteristic polynomial for $H$:
  $p_H(t) = \det(H - tI) = -t^2(t^3 - 6t^2 + 8t - 2)$
- roots of $p_H(t)$: That is, the eigenvalues of H：

$$\lambda_1 = 4.21 > \lambda_2 = 1.45 > \lambda_3 = 0.32 > \lambda_4 = 0 = \lambda_5 = 0$$

The unit eigenvector of$\lambda_1$：$v_{\lambda_1} = \begin{pmatrix} 0.52 \\ 0.74 \\ 0.43 \\ 0 \\ 0 \end{pmatrix}$

若 $k = 1$，則相對於 $v_{\lambda_1}$ 的 community：node $\mathrm{B}$ 為 hubs
若 $k = 2$，則相對於 $v_{\lambda_1}$ 的 community：node A, $\mathrm{B}$ 為 hubs

- The characteristic polynomial of $A$:

$$p_A(t) = \det(A - tI) = -t^2(t^3 - 6t^2 + 8t - 2)$$

- roots of $p(t)$: That is, the eigenvalues of A：

$$\lambda_1 = 4.21 > \lambda_2 = 1.45 > \lambda_3 = 0.32 > \lambda_4 = 0 = \lambda_5 = 0$$

The unit eigenvector of$\lambda_1$：$v_{\lambda_1} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$ 相對於 $v_{\lambda_1}$ 的 community：

node E: authority node

## Why HITS is less popular than PageRank?

| PageRank | HITS |
|---|---|
| Computed for all web pages stored prior to query | Performed on a query-based subset |
| It computes authorities | It computes authorities and hubs |
| Computationally fast | Easy to compute, but hard to compute realtime |

## SALSA=Stochastic Approach for Link-Structure Analysis

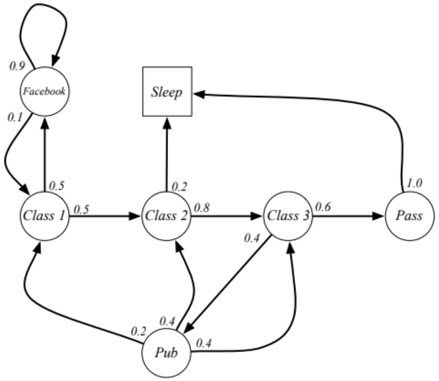
### Definition

SALSA is a web page ranking algorithm designed by R. Lempel and S. Moran to assign high scores to hub and authority web pages based on the quantity of hyperlinks among them. SALSA can be seen as an improvement of HITS. The approach is based upon the theory of Markov chains and relies on the stochastic properties of random walks performed on our collection of pages.

### Definition

A Markov chain or Markov process is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event.

# Student Markov Chain



## Markov chain

**Definition**

A state $S_t$ is Markov if and only if

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{R}[S_{t+1}|S_1, S_2, \cdots, S_t]$$

"The future is independent of the past given the present"

## Formal Definition of SALSA

- A bipartite undirected graph $\tilde{G} = (V_h, V_a, E)$
- $V_h = \{s \in C | \text{out-degree}(s) > 0\}$: the hub side of $\tilde{G}$
- $V_a = \{s \in C | \text{in-degree}(s) > 0\}$: the authorities side of $\tilde{G}$
- $E = \{(s_h, r_a) | s \to r\}$

## Example

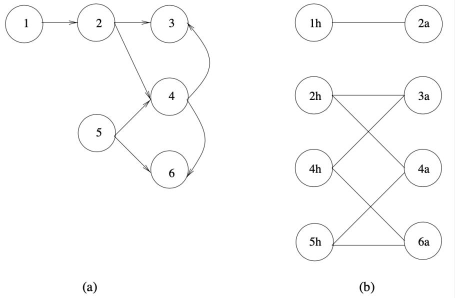


Fig. 1. Transforming (a) the collection $\mathcal{C}$ into (b) a bipartite graph $\tilde{G}$.

## SALSA

- 隨機遊走始於 $\tilde{G}$ 的其中一側，每一步經過 $\tilde{G}$ 的兩條跨兩側的邊（往、返一次），random walk 只紀錄同一側拜訪過的 node．
- Define two transition matrices corresponding to these two different Markov chain
- Hub matrix $\tilde{H}$

$$\tilde{h}_{ij} = \sum_{\{k|(i_h,k_a),(j_h,k_a)\in\tilde{G}\}} \frac{1}{\deg(i_h)} \cdot \frac{1}{\deg(k_a)}$$

- Authority matrix $\tilde{A}$

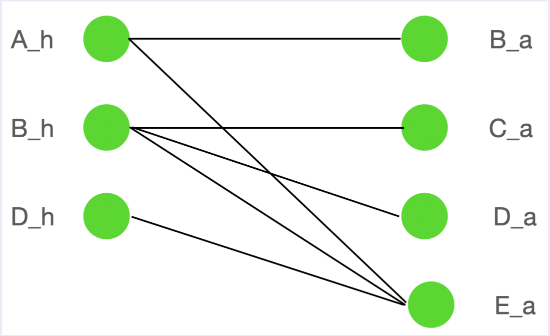
$$\tilde{a}_{ij} = \sum_{k|(k_h,i_a),(k_h,j_a)\in\tilde{G}} \frac{1}{\deg(i_a)} \cdot \frac{1}{\deg(j_a)}$$

$\tilde{a}_{ij} > 0$: ：表示至少存在 page k 個別連到 page i 與 page j

## SALSA in matrix form

- Let $W$ be adjacency matrix of the directed graph defined by its link structure.
- Let $W_r = W/\text{row sum}$, $W_r = W/\text{row sum}$
- The Hub matrix $\tilde{H} = W_r W_c^T$
- The Authority matrix $\tilde{A} = W_c^T W_r$
- $\tilde{H}$ and $\tilde{A}$ are primitive and irreducible.

## Example



## Matrix form

$$\tilde{H} = W_r W_c^T = \begin{pmatrix} 1/2 & 0 & 0 & 1/2 \\ 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1/3 & 1/3 & 1/3 \end{pmatrix}$$

$$= \begin{pmatrix} 4/6 & 1/6 & 1/6 \\ 1/9 & 7/9 & 1/9 \\ 1/3 & 1/3 & 1/3 \end{pmatrix}$$

$$\tilde{A} = W_c^T W_r = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1/3 & 1/3 & 1/3 \end{pmatrix} \begin{pmatrix} 1/2 & 0 & 0 & 1/2 \\ 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1/2 & 0 & 0 & 1/2 \\ 0 & 1/3 & 1/3 & 1/3 \\ 0 & 1/3 & 1/3 & 1/3 \\ 1/6 & 1/9 & 1/9 & 11/18 \end{pmatrix}$$

## Hub

The characteristic polynomial of $\tilde{H}$:

$$\tilde{p}_H(t) = \det(tI - \tilde{H}) = t^3 - 16/9t^2 + 8/9t - 1/9$$

The eigenvalue of $\tilde{H}$ are : $\lambda_1 = 1 > \lambda_2 = 0.59 > \lambda_3 = 0.19$

The unit eigenvector corresponding to 1 is $v_{\lambda(\tilde{H})} = \begin{pmatrix} 0.95 \\ 0.23 \\ -0.58 \end{pmatrix}$ 最大的座標是第一個 $\rightarrow$node A

## Authority

The characteristic polynomial of $\tilde{A}$

$$\tilde{p}_A(t) = \det(tI - \tilde{A}) = t^4 - 16/9t^3 + 8/9t^2 - 1/9t$$

The eigenvalue of $\tilde{A}$ are : $\lambda_1 = 1 > \lambda_2 = 0.59 > \lambda_3 = 0.19 > \lambda_4 = 0$

The unit eigenvector corresponding to 1 is $v_{\lambda(\tilde{A})} = \begin{pmatrix} 0.5 \\ 0.67 \\ -0.75 \\ 0 \end{pmatrix}$ 最大的座標

是第二個 $\rightarrow$node C

## SALSA 與 HITS 的比較

- SALSA 的算法由於不是跑整個 graph，所以運算量較低
- HITS 只考慮往前連結，但 SALSA 有將返回的現象加入機制
- SALSA 只考慮相鄰網頁對自身的影響，HITS 考慮所收集到的網頁對自身的影響

PageRank 的分數是顯示一次 random walk 在整個 WWW 的結果，與搜尋無關

## WTF: Who to follow Service at Twitter 2013

**Definition**

Twitter 是一個 Microblogging 的網際網路平台，發文字數不超過 280 characters (中、日、韓是 140 個字)．一則發文稱 tweet（推特)．截至 2012 八月共有 0.2 億個使用者，每天有 400 百萬則發文，20 億個邊，有超過 1000 個 user 有百萬粉絲，25 個 user 有千萬粉絲．在一個有巨量資料的平台上，要做有效的推薦，需要適合的方法．

論文貢獻：

- Entire graph fits in the memory on a single machine
  Big data could work efficiently
- Describe the complete end-to-end architecture of WTF (Cassovary)
- Present a user recommendation algorithm for directed graphs based on SALSA
- Outline a generation of WTF

# 模型流程

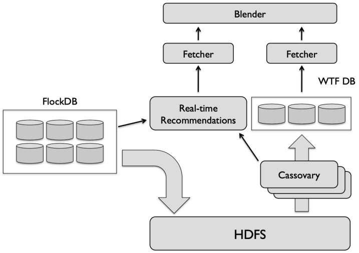
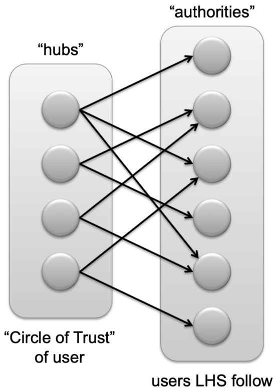


Figure: Data flow

## The Twitter Graph

若從左側開始，到右側，會回到左側；若從右側開始，會回到右側．我們使用 SALSA 做幾次迭代，分別計算 hubs scores 和 authorities score．



應用：即時推薦．例如：若一個 user $u$ 剛 follow 了 $v_1, v_2$，可選擇計算 $\{u, v_1, v_2\}$ 的 circle of trust，推薦給 $u$．

## Circle of Trust

- User's recommendation algorithm: a user's "circle of trust"
- a result of an egocentric random walk

Cassovary 使用一些參數來計算 circle of trust 分數：

- random walk 的長度
- 重新開始的機率
- 剪枝要裁掉 node 的機率值
- 控制抽樣到聯外節點 degree 太大的參數

在實際應用上，我們選擇排名在 user's circle of trust 前 500 名的 node，再推薦 authorities 排名高的給 user．hubs 的排名可以看成相似度，SALSA 本身就具有遞迴的關係，排名相近，也具有相近關係．

## 作業三

使用作業二的圖，若 $k = 1$，在 HITS 算法中，誰是 authority？誰是 hub？
在 SALSA 算法中，誰是 authority？誰是 hub？

## Related Pins at Pinterest

論文：Related Pins at Pinterest: The Evolution of a Real-World Recommender System
整體架構：建立 Pin-Borad Graph、Memboost、Ranking 的流程及算法