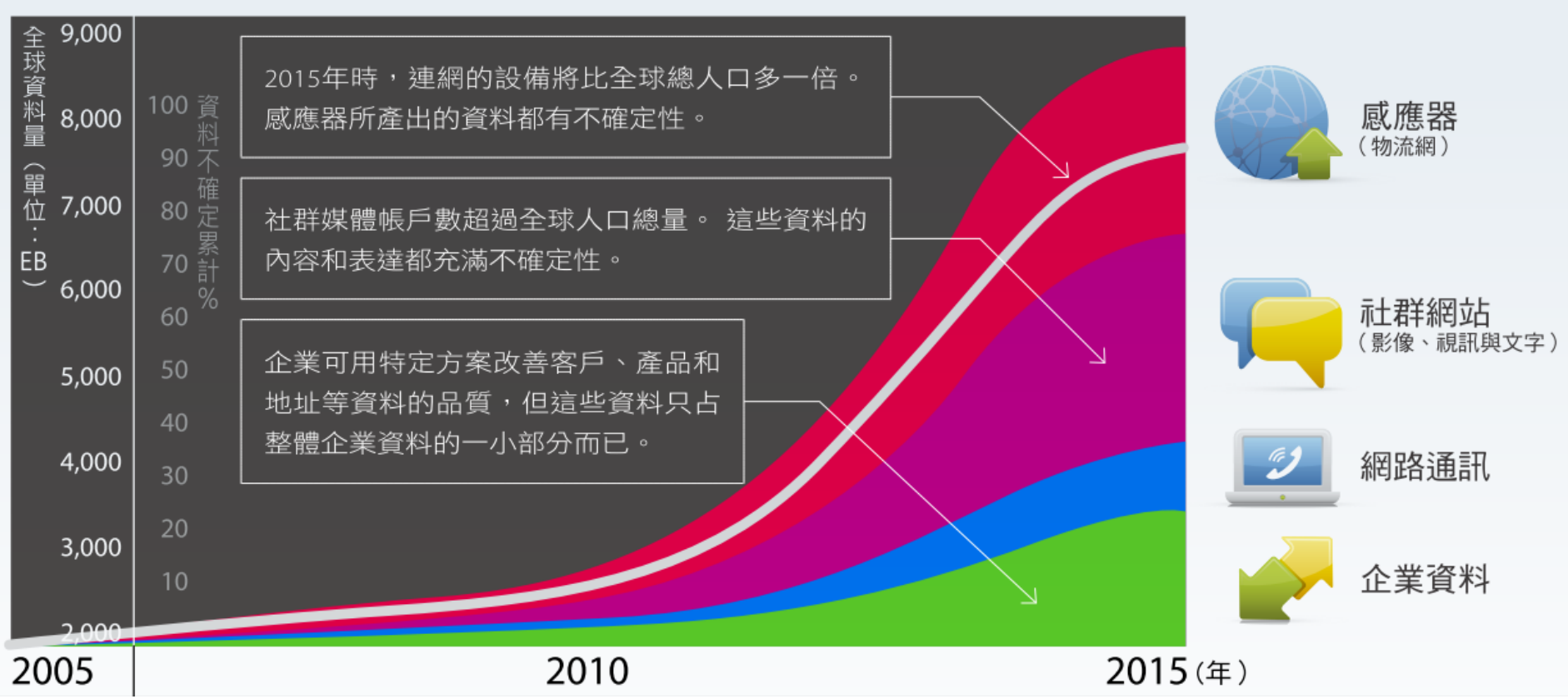
Part I : Text / Social Analytics
文字及社群數據分析

楊立偉教授

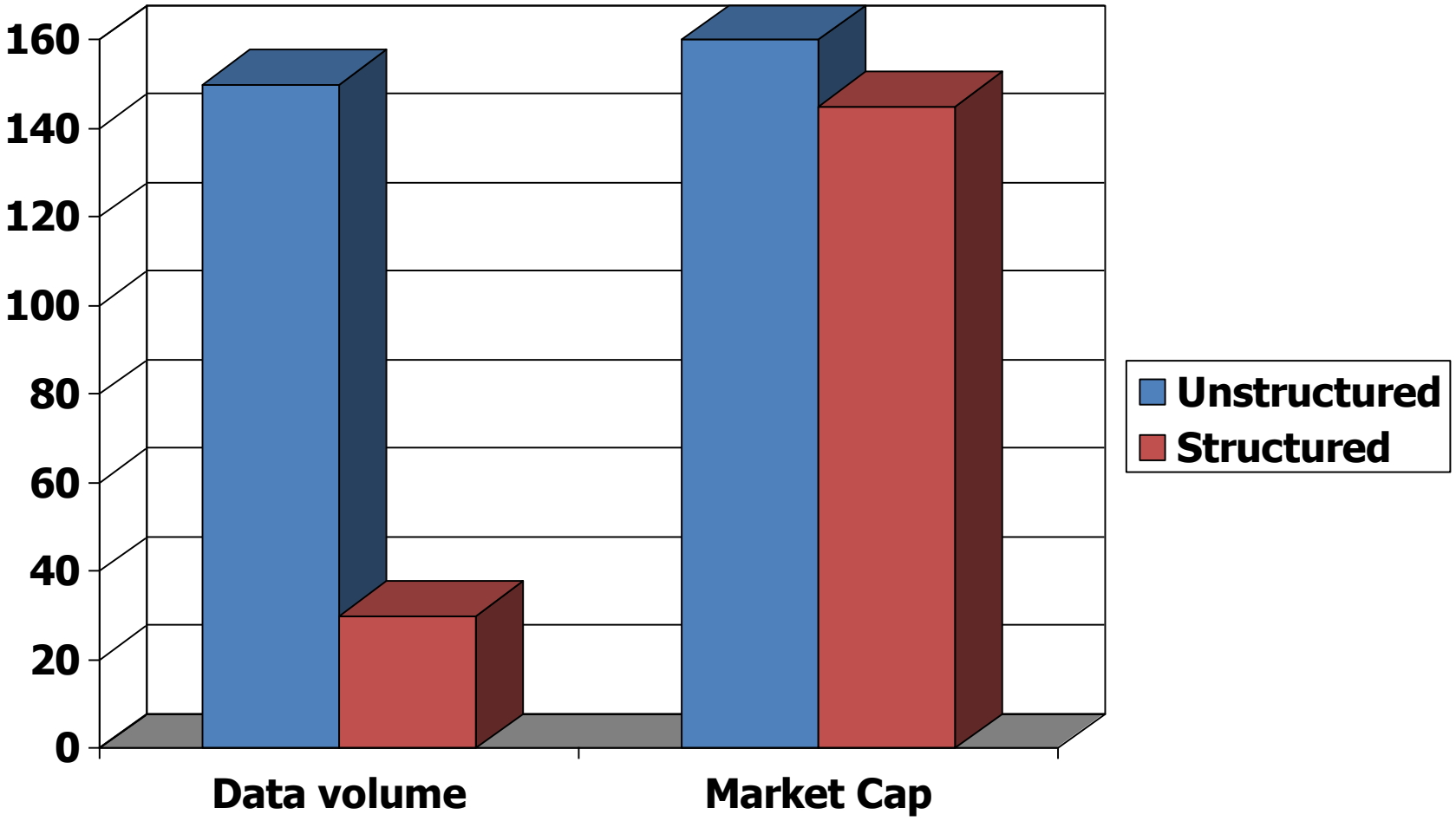
wyang@ntu.edu.tw

資料快速成長



Source : IBM 2012全球CEO調查報告
https://www-07.ibm.com/tw/blueview/2012oct/8.html

Unstructured (text) vs. Structured (database) data



Source: Stanford NLP, 2006

Text Mining

- 自然語言處理(NLP)與文字探勘(Text mining)是美國麻省理工學院MIT選為未來十大最重要技術之一
- 是重要的跨學域研究 (inter-discipline research)
  - Linguistics 語言學, and Computing Linguistics 計算語言學
  - Information Retrieval and Extraction 資訊檢索與擷取
  - Text Mining 文本探戡 and Knowledge Discovery 知識探索
  - Ontology, Domain knowledge… etc.
- 先能處理大量資訊，再將處理層次提升
  - Ex. 全文檢索 → 摘要 → 意見與觀點偵測 → 找出意見持有者
    → 找出比較性意見 → 做持續性追蹤 → 找出答案 ….

Info Retrieval & Extraction → Text Mining → Knowledge Discovery

案例：語言分析與標記

- Tagging – 人名、關鍵詞、時間、地點、情緒
- Summary – 摘要、相關詞、事件追蹤

SA memo	TEXT MINING結果								後續應用
	註記	提醒	抱怨	偏好	理財	日期	摘要	關鍵字	
美金外幣定存資金不想讓先生知道，因為是債券型基金配息累積的，想當做私房錢	✓						美金外幣定存資金不想讓先生知道		特殊背景註記
日前聽聞FP說土地信託的詐騙方式，有點擔心客戶，但是因為不了解手法，再調查，11/20客戶會來再請他小心		✓				11/20	土地信託詐騙		下名單
客戶反應 [redacted] 保險的客戶禮券太晚收到，對 [redacted] 保險印象不佳			✓				[redacted] 保險的客戶禮券太晚收到		即時pass至客經三部
客戶出國1個月帶回萬張攝影照片，由秘書代為解檔處理.喜號旅遊攝影.下星期與PA再約訪客戶見面.				✓				旅遊,攝影	特殊背景註記 [redacted]

知名案例

科技 人工智慧

機器學習告訴你：《紅樓夢》後40回到底是不是曹雪芹寫的？

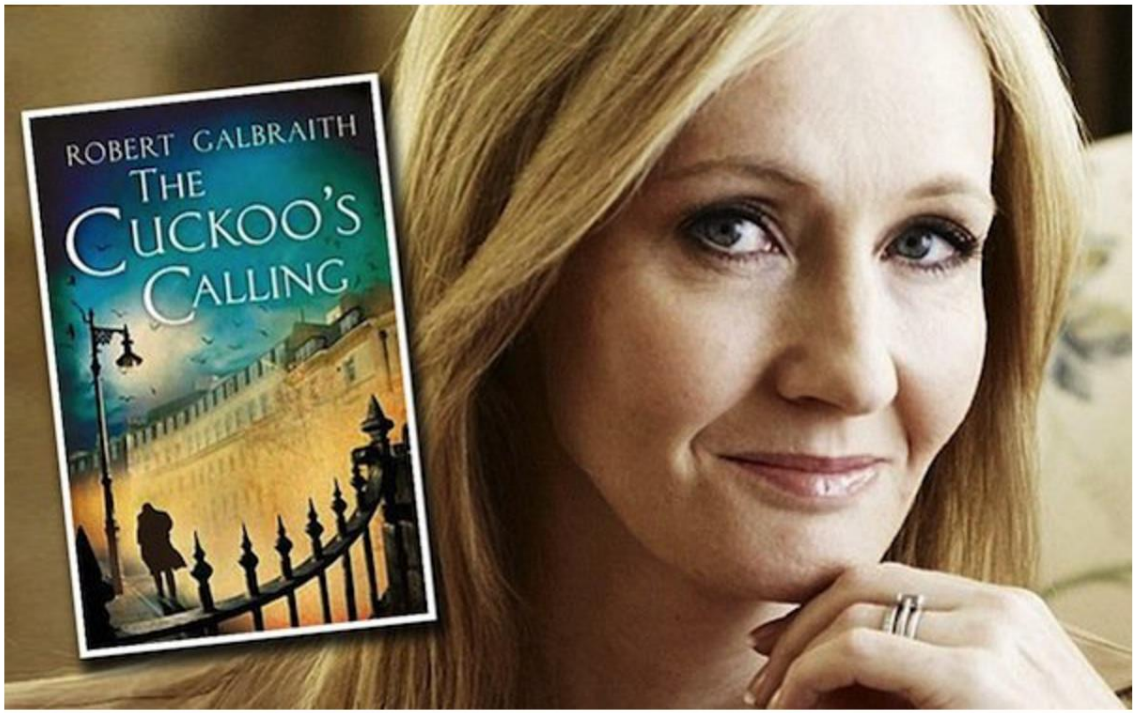
by 黎晨 2016.07.07

Source: www.bnext.com.tw/article/40151/BN-2016-07-07-160903-178

知名案例

語意分析技術，讓「哈利波特」作者羅琳改名出新書一事被曝光

2013/7/22 ▪ 【合作媒體】iFanr愛範兒 ▪ 有趣技術



Source: www.inside.com.tw/2013/07/22/how-forensic-linguistics-outed-j-k-rowling

- 署名 Robert Galbraith 的新作家寫了本偵探小說《Cuckoo》評價高但銷量慘淡。
- Twitter 有條匿名消息，宣稱 Galbraith 就是羅琳。某報紙編輯為了確認，找到牛津大學兩位電腦科學家，進行語言鑑識分析 (forensic linguistics)
- 實驗的進行，除了《Cuckoo》之外，還有羅琳的另一本小說，以及另外三本英國犯罪小說。
- 在多項測試中，其中一項是詞組的使用，因為不同的作者會使用不同的詞彙來形容某種東西，另一項測試是檢查某些常見詞彙的出現頻率。最強大的證據是單詞長度測試，從這個測試中發現了羅琳寫作的特色。各項結果證明，《Cuckoo》與羅琳的小說最為接近。
- 進行複試，另外再加上幾本書，進行六項對比：單詞長度、句子長度、段落長度、單詞出現頻率、標點出現頻率，以及單詞使用情況。結果同樣顯示《Cuckoo》與羅琳所寫的書最為接近

語意分析的應用

- 內容分析
  - 從大量(不一定精確)的文字中，找出有用的資訊
  - 擷取特徵字、特徵詞、特徵句、特徵段落(摘要)等
  - 共現分析、引文分析
- 相似性分析
  - 文獻比對、專利比對、判例比對、著作權比對等
- 自動分類或分群
  - 信件(廣告文)分類、客訴案件分類、新聞分類、專利分類等
- 還有嗎？

Lecture 1 : Term Weighting and VSM

楊立偉教授

wyang@ntu.edu.tw

本投影片修改自Introduction to Information Retrieval一書之投影片
Ch 1~3, 6

The foundation : Information Retrieval

Information retrieval (IR) is the foundation to Text mining.

IR is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

Info Retrieval & Extraction → Text Mining → Knowledge Discovery

Unstructured data in 1650

- Which plays of Shakespeare contain the words

  BRUTUS AND CAESAR, but not CALPURNIA ?

- One could scan all of Shakespeare's plays for BRUTUS and CAESAR, then strip out lines containing CALPURNIA

- Why is scan not the solution?

  - Slow (for large collections)

  - Advanced operations not feasible (e.g., find the word ROMANS near COUNTRYMAN)

Term-document incidence matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	1	1	0	0	0	1
BRUTUS	1	1	0	1	0	0
CAESAR	1	1	0	1	1	1
CALPURNIA	0	1	0	0	0	0
CLEOPATRA	1	0	0	0	0	0
MERCY	1	0	1	1	1	1
WORSER	1	0	1	1	1	0
. . .						

Entry is 1 if term occurs. Example: CALPURNIA occurs in *Julius Caesar*.
Entry is 0 if term doesn't occur. Example: CALPURNIA doesn't occur in *The tempest*.

Incidence vectors

- So we have a 0/1 vector for each term.
- To answer the query BRUTUS AND CAESAR AND NOT CALPURNIA:
  - Take the vectors for BRUTUS, CAESAR AND NOT CALPURNIA
  - Complement the vector of CALPURNIA
  - Do a (bitwise) and on the three vectors
  - 110100 AND 110111 AND 101111 = 100100

0/1 vector for BRUTUS

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	1	1	0	0	0	1
BRUTUS	1	1	0	1	0	0
CAESAR	1	1	0	1	1	1
CALPURNIA	0	1	0	0	0	0
CLEOPATRA	1	0	0	0	0	0
MERCY	1	0	1	1	1	1
WORSER	1	0	1	1	1	0
. . .						
result:	1	0	0	1	0	0

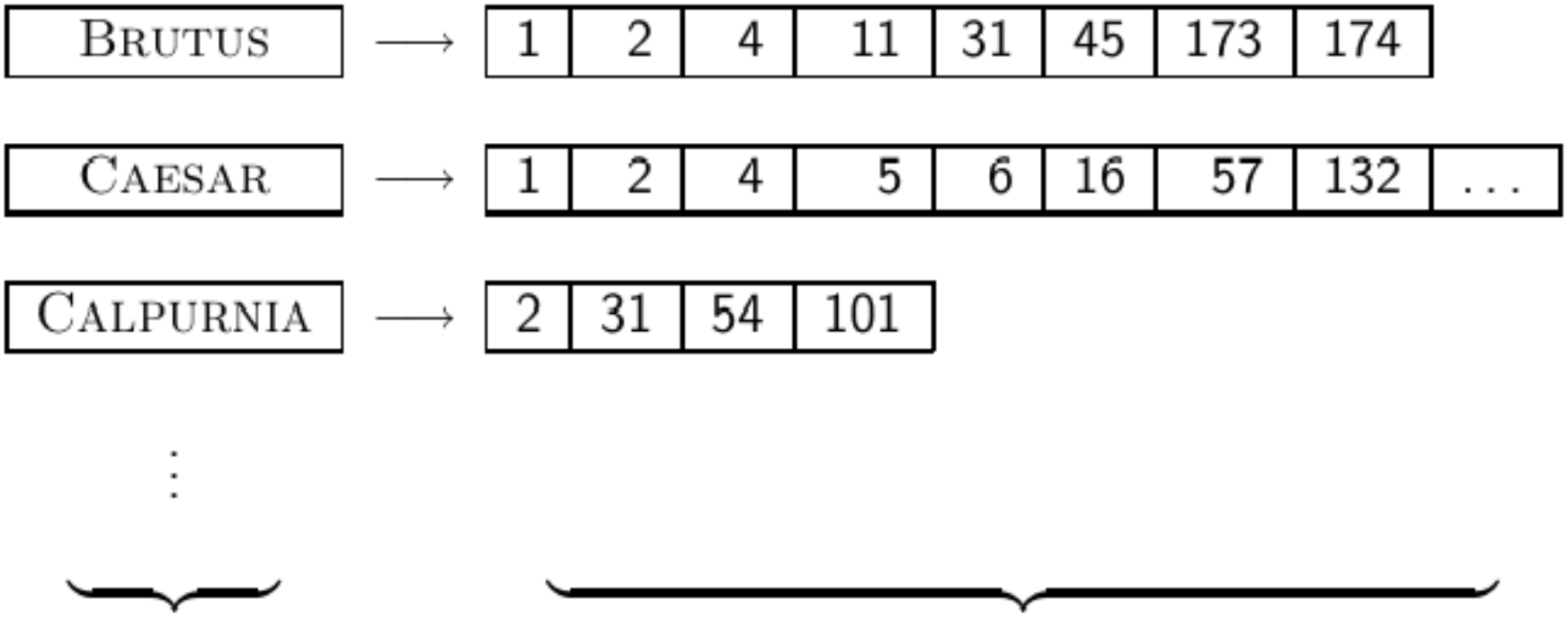
答案是Antony and Cleopatra與Hamlet

Too big to build the incidence matrix

- Consider $N = 10^6$ documents, each with about 1000 tokens
  ⇒ total of 10^9 tokens (10億)
- Assume there are M = 500,000 distinct terms in the collection
- M = 500,000 × 10^6 = half a trillion 0s and 1s. (5000億)
- But the matrix has no more than 10^9 1s.
  就算全部token不重複也不會超過10億個字
  - Matrix is extremely sparse. (only 10/5000 has values)
    因此有至少4990億個位置都是0
- What is a better representations?
  - We only record the 1s.

Inverted Index

For each term *t*, we store a list of all documents that contain *t*.



Ranked Retrieval

Problem with Boolean search

- Boolean retrieval return documents either match or don't.
- Boolean queries often result in either too few (=0) or too many (1000s) results.

  Example query : [standard user dlink 650]

  → 200,000 hits

  Example query : [standard user dlink 650 no card found]

  → 0 hits
- Good for expert users with precise understanding of their needs and of the collection. Not good for the majority of users

Ranked retrieval

- With ranking, large result sets are not an issue.
- More relevant results are ranked higher than less relevant results.
- The user may decide how many results he/she wants.

Scoring as the basis of ranked retrieval

- Assign a score to each query-document pair, say in [0, 1], to measure how well document and query “match”.
- If the query term does not occur in the document: score should be 0.
- The more frequent the query term in the document, the higher the score

Term Frequency

Binary incidence matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	1	1	0	0	0	1
BRUTUS	1	1	0	1	0	0
CAESAR	1	1	0	1	1	1
CALPURNIA	0	1	0	0	0	0
CLEOPATRA	1	0	0	0	0	0
MERCY	1	0	1	1	1	1
WORSER	1	0	1	1	1	0
. . .						

Each document is represented as a binary vector $\in \{0, 1\}^{|V|}$.

Count matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	157	73	0	0	0	1
BRUTUS	4	157	0	2	0	0
CAESAR	232	227	0	2	1	0
CALPURNIA	0	10	0	0	0	0
CLEOPATRA	57	0	0	0	0	0
MERCY	2	0	3	8	5	8
WORSER	2	0	1	1	1	5
. . .						

Each document is now represented as a count vector $\in \mathbb{N}^{|V|}$.

Here is Bag of Words (BOW) model

- Do not consider the order of words in a document.
  - c.f. Continuous model, Sequential model.
- *John is quicker than Mary , and*

  Mary is quicker than John

  are represented the same way.

Term frequency tf

- The term frequency $tf_{t,d}$ of term *t* in document *d* is defined as the number of times that *t* occurs in *d*.
- Use tf when computing query-document match scores.
- But Relevance does not increase proportionally with term frequency.
- Example

  A document with tf = 10 occurrences of the term is more relevant than a document with tf = 1 occurrence of the term, but not 10 times more relevant.

Log frequency weighting

- The log frequency weight of term t in d is defined as follows

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d} & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$$

- $\mathrm{tf}_{t,d} \rightarrow w_{t,d}$:
  0 → 0, 1 → 1, 2 → 1.3, 10 → 2, 1000 → 4, etc.
- Why use log ? 在數量少時, 差1即差很多；
  但隨著數量越多，差1的影響變得越小

- tf-matching-score(q, d) = $\sum_{t \in q \cap d} (1 + \log \mathrm{tf}_{t,d})$

Exercise

- Compute the tf matching score for the following query(Q)-document(D) pairs.
- Q: [information on cars] D: "all you have ever wanted to know about cars"

  tf = 0 + 0 + (1+log1)
- Q: [information on cars] D: "information on trucks, information on planes, information on trains"

  tf = (1+log3) + (1+log3) + 0

TF-IDF Weighting

Desired weight for frequent terms

- Frequent terms are less informative than rare terms.
- Consider a term in the query that is frequent in the collection (e.g., GOOD, INCREASE, LINE).

  → common term or 無鑑別力的詞

Desired weight for rare terms

- Rare terms are more informative than frequent terms.
- Consider a term in the query that is rare in the collection (e.g., ARACHNOCENTRIC).
- A document containing this term is very likely to be relevant.
- → We want high weights for rare terms like ARACHNOCENTRIC.

Document frequency

- We want high weights for rare terms like ARACHNOCENTRIC.
- We want low (still positive) weights for frequent words like GOOD, INCREASE and LINE.
- We will use document frequency to factor this into computing the matching score.
- The document frequency is the number of documents in the collection that the term occurs in.

idf weight

- df_t is the document frequency, the number of documents that *t* occurs in.
- df_t is an inverse measure of the informativeness of term *t*.
- We define the idf weight of term t as follows:

$$\text{idf}_t = \log_{10} \frac{N}{\text{df}_t}$$

(*N* is the number of documents in the collection.)

- idf_t is a measure of the informativeness of the term.
- [log N/df_t] instead of [N/df_t] to balance the effect of idf (i.e. use log for both tf and df)

Examples for idf

- Compute idf_t using the formula: $\text{idf}_t = \log_{10} \frac{1{,}000{,}000}{\text{df}_t}$

term	df_t	idf_t
calpurnia	1	6
animal	100	4
sunday	1000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

Collection frequency vs. Document frequency

word	collection frequency	document frequency
INSURANCE	10440	3997
TRY	10422	8760

- Collection frequency of *t*: number of tokens of *t* in the collection
- Document frequency of *t*: number of documents *t* occurs in
- Document/collection frequency weighting is computed from known collection, or estimated
- Which word is a more informative ?

Example

- cf 出現總次數 與 df 文件數。差異範例如下：

Word	*cf* 出現總次數	*df* 出現文件數
ferrari	10422	17 ←較高的稀有性 (高資訊量)
insurance	10440	3997

tf-idf weighting

- The tf-idf weight of a term is the product of its tf weight and its idf weight.

$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \cdot \log \frac{N}{\mathrm{df}_t}$$

- tf-weight
- idf-weight
- Best known weighting scheme in information retrieval

Note: the "-" in tf-idf is a hyphen, not a minus sign

Alternative names: tf.idf , tf x idf

Summary: tf-idf

- Assign a tf-idf weight for each term t in each document *d*:

$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \cdot \log \frac{N}{\mathrm{df}_t}$$

- The tf-idf weight . . .
  - . . . increases with the number of occurrences within a document. (term frequency)
  - . . . increases with the rarity of the term in the collection. (inverse document frequency)
  - tf-idf 概念上可看作 tf / df 其實就是詞的分布密度

Exercise: Term, collection and document frequency

Quantity	Symbol	Definition
term frequency	$tf_{t,d}$	number of occurrences of *t* in *d*
document frequency	df_t	number of documents in the collection that *t* occurs in
collection frequency	cf_t	total number of occurrences of *t* in the collection

- Relationship between df and cf?
- Relationship between tf and cf? cf是每篇tf的加總
- Relationship between tf and df?

Vector Space Model

Binary incidence matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	1	1	0	0	0	1
BRUTUS	1	1	0	1	0	0
CAESAR	1	1	0	1	1	1
CALPURNIA	0	1	0	0	0	0
CLEOPATRA	1	0	0	0	0	0
MERCY	1	0	1	1	1	1
WORSER	1	0	1	1	1	0
. . .						

Each document is represented as a binary vector $\in \{0, 1\}^{|V|}$.

Count matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	157	73	0	0	0	1
BRUTUS	4	157	0	2	0	0
CAESAR	232	227	0	2	1	0
CALPURNIA	0	10	0	0	0	0
CLEOPATRA	57	0	0	0	0	0
MERCY	2	0	3	8	5	8
WORSER	2	0	1	1	1	5
. . .						

Each document is now represented as a count vector $\in \mathbb{N}^{|V|}$.

Binary → count → weight matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	5.25	3.18	0.0	0.0	0.0	0.35
BRUTUS	1.21	6.10	0.0	1.0	0.0	0.0
CAESAR	8.59	2.54	0.0	1.51	0.25	0.0
CALPURNIA	0.0	1.54	0.0	0.0	0.0	0.0
CLEOPATRA	2.85	0.0	0.0	0.0	0.0	0.0
MERCY	1.51	0.0	1.90	0.12	5.25	0.88
WORSER	1.37	0.0	0.11	4.15	0.25	1.95
. . .						

Each document is now represented as a real-valued vector of tf idf weights $\in R^{|V|}$.

Documents as vectors

- Each document is now represented as a real-valued vector of tf-idf weights $\in R^{|V|}$.
- So we have a $|V|$-dimensional real-valued vector space.
- Terms are axes of the space.
- Documents are points or vectors in this space.

- Each vector is very sparse - most entries are zero.
- Very high-dimensional: tens of millions of dimensions when apply this to web (i.e. too many different terms on web)

Vector Space Model

- 將文件透過一組詞與其權重，將文件轉化為空間中的向量（或點），因此可以
  - 計算文件相似性或文件距離
  - 計算文件密度
  - 找出文件中心
  - 進行分群（聚類）
  - 進行分類（歸類）

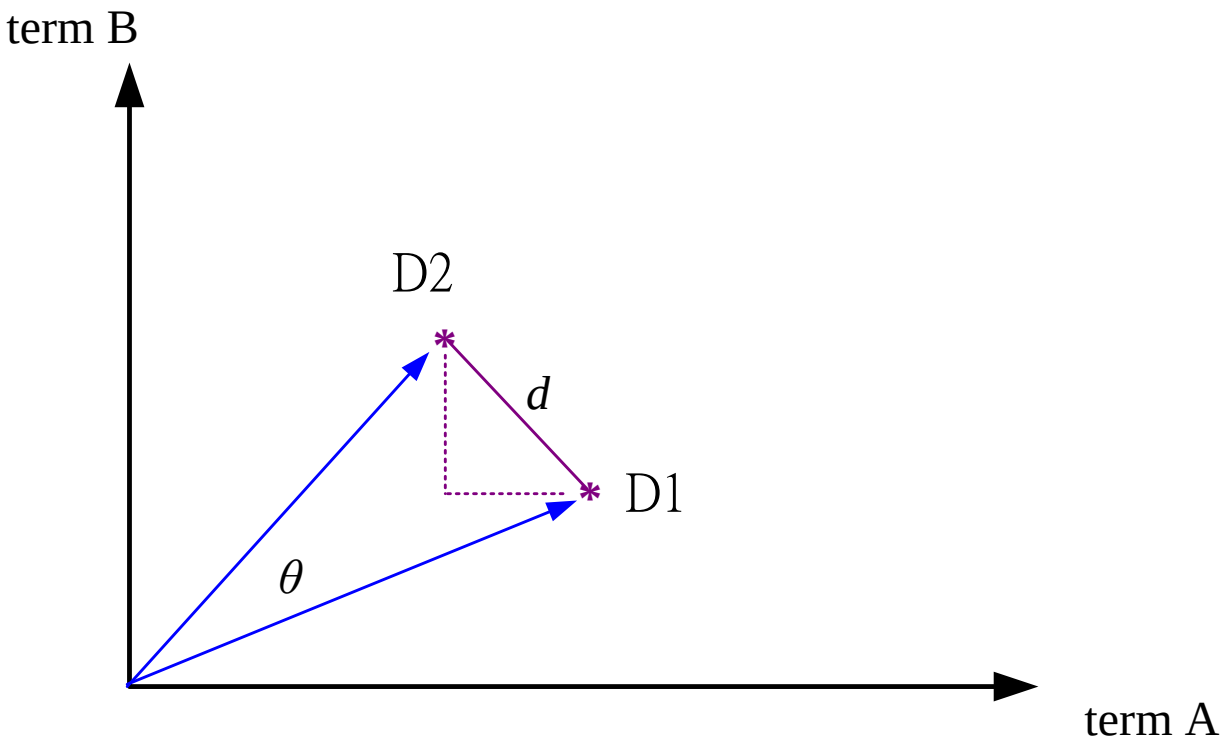
Vector Space Model

- 假設只有Antony與Brutus兩個詞，文件可以向量表示如下
  D1: Antony and Cleopatra = (5.25, 1.21)
  D2: Julius Caesar = (3.18, 6.10)

- 計算文件相似性：
  以向量夾角表示
  用內積計算
  5.25 x 3.18 + 1.21 x 6.10

- 計算文件幾何距離：

$$\sqrt{(5.25-3.18)^2+(1.21-6.10)^2}$$

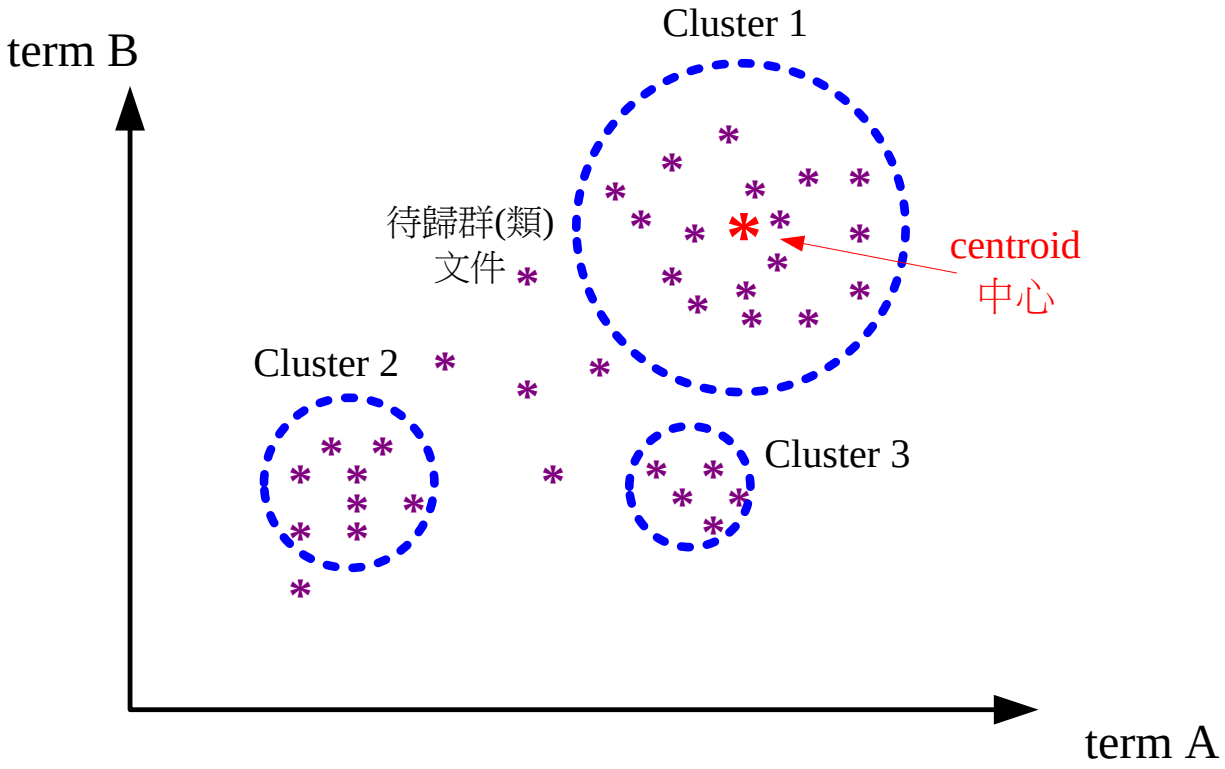


Applications of Vector Space Model

- 分群 (聚類) Clustering：由最相近的文件開始合併
- 分類 (歸類) Classification：挑選最相近的類別

- 中心 Centroid
  可做為群集之代表
  或做為文件之主題

- 文件密度
  了解文件的分布狀況



Issues about Vector Space Model (1)

- 詞之間可能存有相依性，非垂直正交 (orthogonal)
  - 假設有兩詞 tornado, apple 構成的向量空間，

    D1=(1,0) D2=(0,1)，其內積為0，故稱完全不相似
  - 但當有兩詞 tornado, hurricane 構成的向量空間，

    D1=(1,0) D2=(0,1)，其內積為0，但兩文件是否真的不相似？
  - 當詞為彼此有相依性 (dependence)
    - 挑出正交（不相依）的詞
    - 將維度進行數學轉換（找出正交軸）

Issues about Vector Space Model (2)

- 詞可能很多，維度太高，讓內積或距離的計算變得很耗時
  - 常用詞可能自數千至數十萬之間，造成高維度空間
    (運算複雜度呈指數成長, 又稱 curse of dimensionality)
  - 常見的解決方法
    - 只挑選具有代表性的詞（feature selection）
    - 將維度進行數學轉換（latent semantic indexing）

document as a vector

term as axes

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	13.1	11.4	0.0	0.0	0.0	0.0
Brutus	3.0	8.3	0.0	1.0	0.0	0.0
Caesar	2.3	2.3	0.0	0.5	0.3	0.3
Calpurnia	0.0	11.2	0.0	0.0	0.0	0.0
Cleopatra	17.7	0.0	0.0	0.0	0.0	0.0
mercy	0.5	0.0	0.7	0.9	0.9	0.3
worser	1.2	0.0	0.6	0.6	0.6	0.0

the dimensionality is 7

Queries as vectors

- Do the same for queries: represent them as vectors in the high-dimensional space
- Rank documents according to their proximity to the query

- proximity = similarity ≈ negative distance
- Rank relevant documents higher than non-relevant documents.

Use angle instead of distance

- Rank documents according to angle with query
- For example : take a document d and append it to itself. Call this document *d′*. *d′* is twice as long as *d*.
- "Semantically" *d* and *d′* have the same content.
- The angle between the two documents is 0, corresponding to maximal similarity . . .
- . . . even though the Euclidean distance between the two documents can be quite large.

- 另一個原因是向量內積計算比幾何距離計算快速

From angles to cosines

- The following two notions are equivalent.
  - Rank documents according to the angle between query and document in decreasing order
  - Rank documents according to cosine(query,document) in increasing order

Length normalization

- A vector can be (length-) normalized by dividing each of its components by its length – here we use the L_2 norm:

$$||x||_2 = \sqrt{\sum_i x_i^2}$$

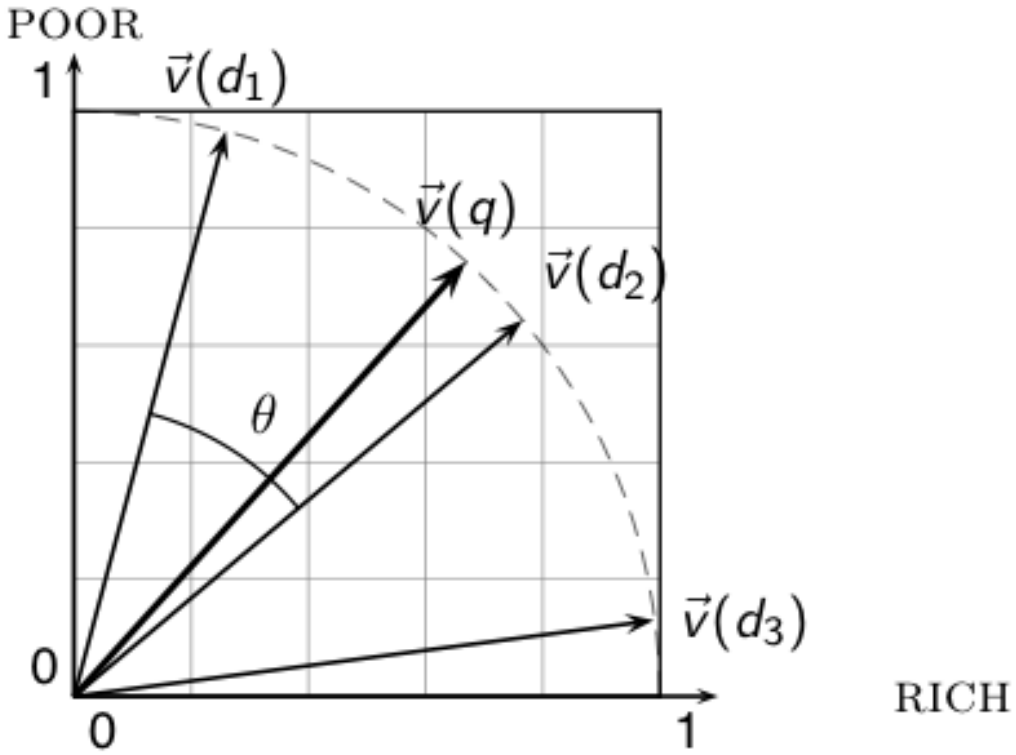
- This maps vectors onto the unit sphere . . .
- . . . since after normalization: $||x||_2 = \sqrt{\sum_i x_i^2} = 1.0$
- As a result, longer documents and shorter documents have weights of the same order of magnitude.
- Effect on the two documents *d* and *d'* (*d* appended to itself) : they have identical vectors after length-normalization.

Cosine similarity between query and document

$$\cos(\vec{q}, \vec{d}) = \text{SIM}(\vec{q}, \vec{d}) = \frac{\vec{q} \cdot \vec{d}}{|\vec{q}||\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2} \sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

- q_i is the tf-idf weight of term *i* in the query.
- d_i is the tf-idf weight of term *i* in the document.
- $|\vec{q}|$ and $|\vec{d}|$ are the lengths of $\vec{q}$ and $\vec{d}$.
- This is the cosine similarity of $\vec{q}$ anc $\vec{d}$. . . . . or, equivalently, the cosine of the angle between $\vec{q}$ and $\vec{d}$.

Cosine similarity illustrated



Cosine: Example

How similar are these novels?

SaS: Sense and Sensibility 理性與感性

PaP:Pride and Prejudice 傲慢與偏見

WH: Wuthering Heights 咆哮山莊

term frequencies (counts)

term	SaS	PaP	WH
AFFECTION	115	58	20
JEALOUS	10	7	11
GOSSIP	2	0	6
WUTHERING	0	0	38

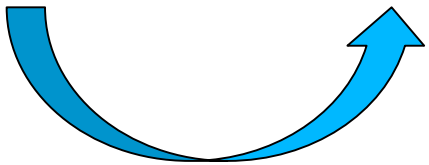
Cosine: Example

term frequencies (counts)

term	SaS	PaP	WH
AFFECTION	115	58	20
JEALOUS	10	7	11
GOSSIP	2	0	6
WUTHERING	0	0	38

log frequency weighting

term	SaS	PaP	WH
AFFECTION	3.06	2.76	2.30
JEALOUS	2.0	1.85	2.04
GOSSIP	1.30	0	1.78
WUTHERING	0	0	2.58



每一個值依 tf 公式計算
例如 $1 + log_{10}115 = 3.06$

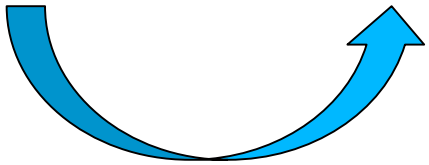
(To simplify this example, we don't do idf weighting.)
因文章數量太少，先不做 idf 加權

log frequency weighting

term	SaS	PaP	WH
AFFECTION	3.06	2.76	2.30
JEALOUS	2.0	1.85	2.04
GOSSIP	1.30	0	1.78
WUTHERING	0	0	2.58
單位長度L	3.88	3.32	4.39

cosine normalization

term	SaS	PaP	WH
AFFECTION	0.789	0.832	0.524
JEALOUS	0.515	0.555	0.465
GOSSIP	0.335	0.0	0.405
WUTHERING	0.0	0.0	0.588



每一行向量除以自身單位長度 L
例如 $3.06 \div 3.88 = 0.789$

單位長度之計算

Ex. $\sqrt{3.06^2 + 2.0^2 + 1.30^2} = 3.88$

Cosine: Example

after log frequency weighting & cosine normalization,
we have 3 document vectors. Ex. SaS=(0.789, 0.515, 0.335, 0)

term	SaS	PaP	WH
AFFECTION	0.789	0.832	0.524
JEALOUS	0.515	0.555	0.465
GOSSIP	0.335	0.0	0.405
WUTHERING	0.0	0.0	0.588

- cos(SaS,PaP) ≈
  $0.789 * 0.832 + 0.515 * 0.555 + 0.335 * 0.0 + 0.0 * 0.0 \approx 0.94.$
- cos(SaS,WH) ≈ 0.79
- cos(PaP,WH) ≈ 0.69
- Why do we have cos(SaS,PaP) > cos(SaS,WH)?

以向量夾角表示相似度，發現《理性與感性》和《傲慢與偏見》最像

Ranked retrieval in the Vector Space Model

- Represent the query as a weighted tf-idf vector
- Represent each document as a weighted tf-idf vector
- Compute the cosine similarity between the query vector and each document vector
- Rank documents with respect to the query
- Return the top K (e.g., $K = 10$) to the user

Conclusion

- Ranking search results is important (compared with unordered Boolean results)
- Term frequency
- tf-idf ranking: best known traditional ranking scheme
- Vector space model: One of the most important formal models for information retrieval

  (along with Boolean and probabilistic models)