Information Retrieval and the Vector Space Model

CSE 6339 Introduction to Computational Linguistics
Prof. Nick Cercone
Winter. 2014

Presenters:
Farzana Yasmeen
Yasser Gonzalez-Fernandez
2014.02.24

Outline

- Typical IR System Architecture
- Document and Query Processing in IR
- IR Evaluation
- Vector Space Model (VSM)
- Similarity Measure
- Term Weighing
- Improving the Vector Space Model
- Latent Semantic Analysis

What is Information Retrieval (IR)?

- Coined by Calvin Mooers, as early as 1950's

Definition:

"Information retrieval (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)."

Process of IR:

Information Retrieval (IR) constructs an index for a given corpus and responds to queries by retrieving all the relevant documents and as few non-relevant documents as possible.

- index a collection of documents (access efficiency)
- given users query
- rank documents by importance (accuracy)

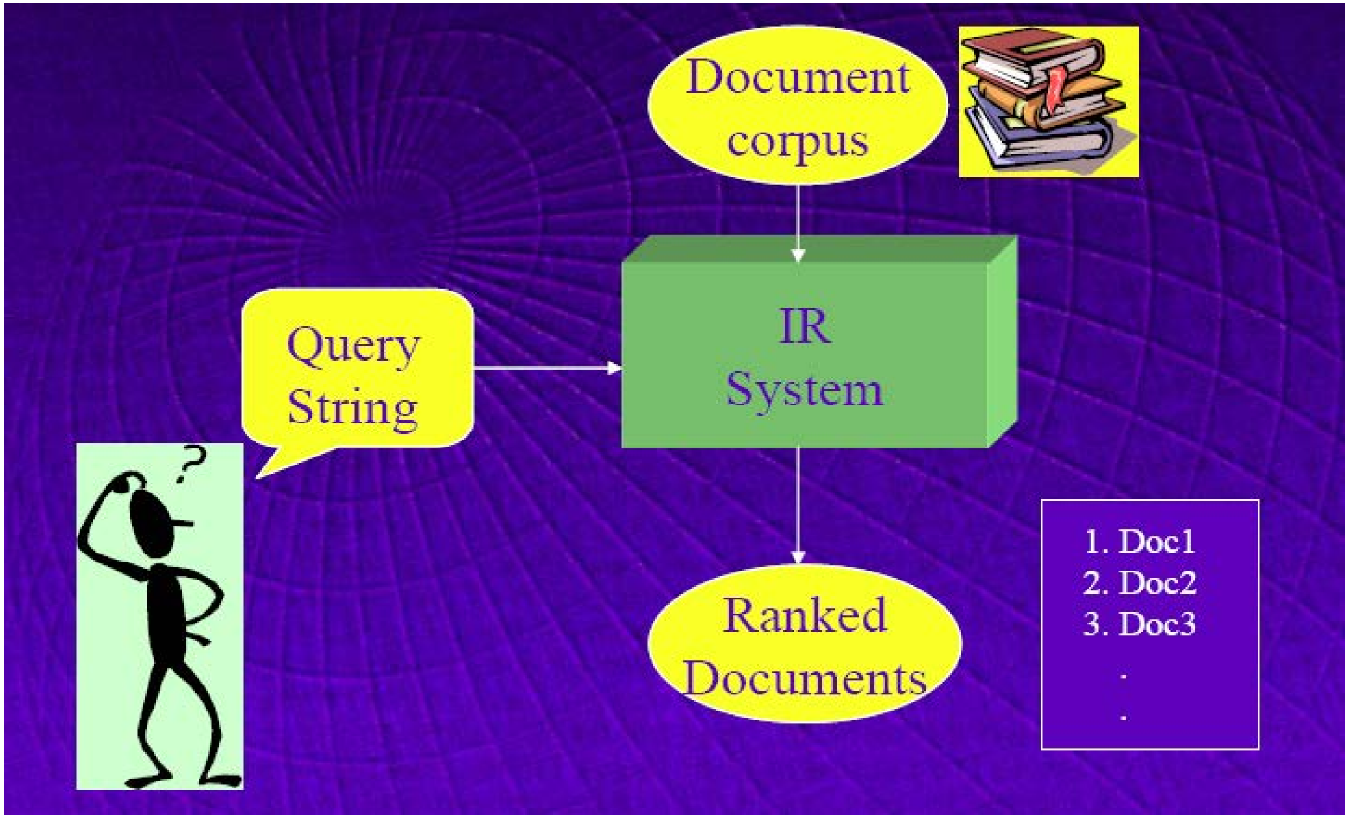
Typical IR Task

Given:

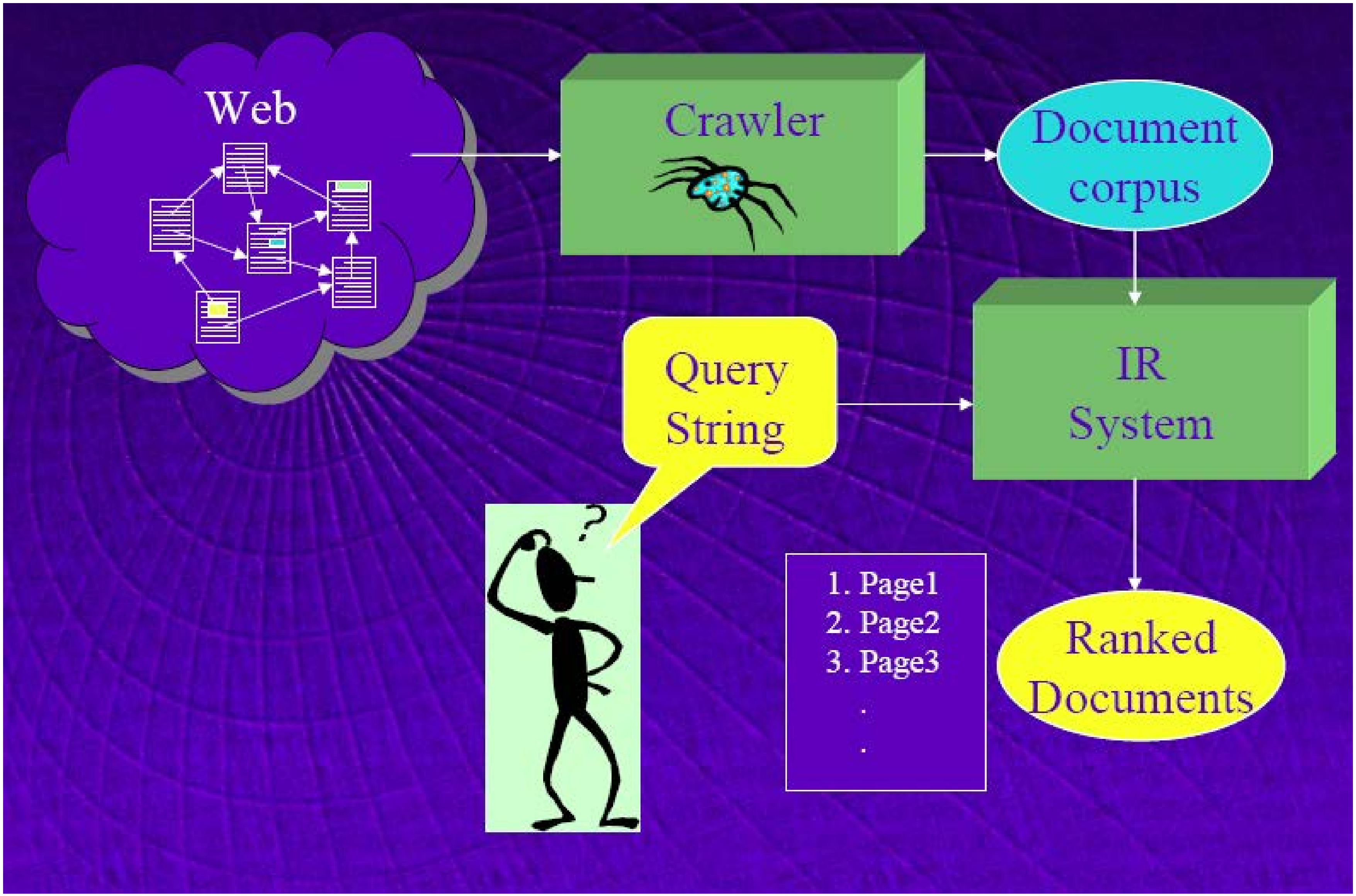
- A corpus of textual natural-language documents.
- A user query in the form of a textual string.

Find:

- A ranked set of documents that are relevant to the query

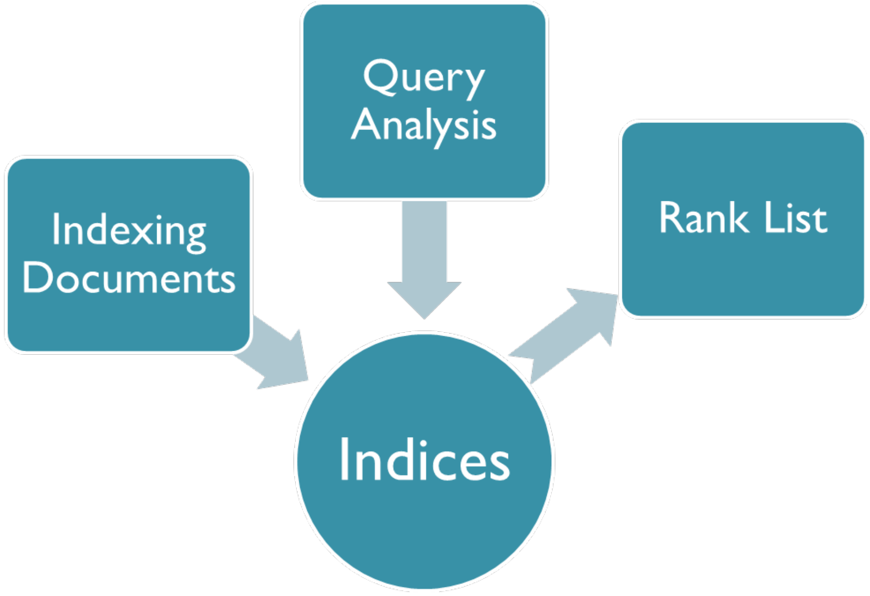


Example – Web Search System

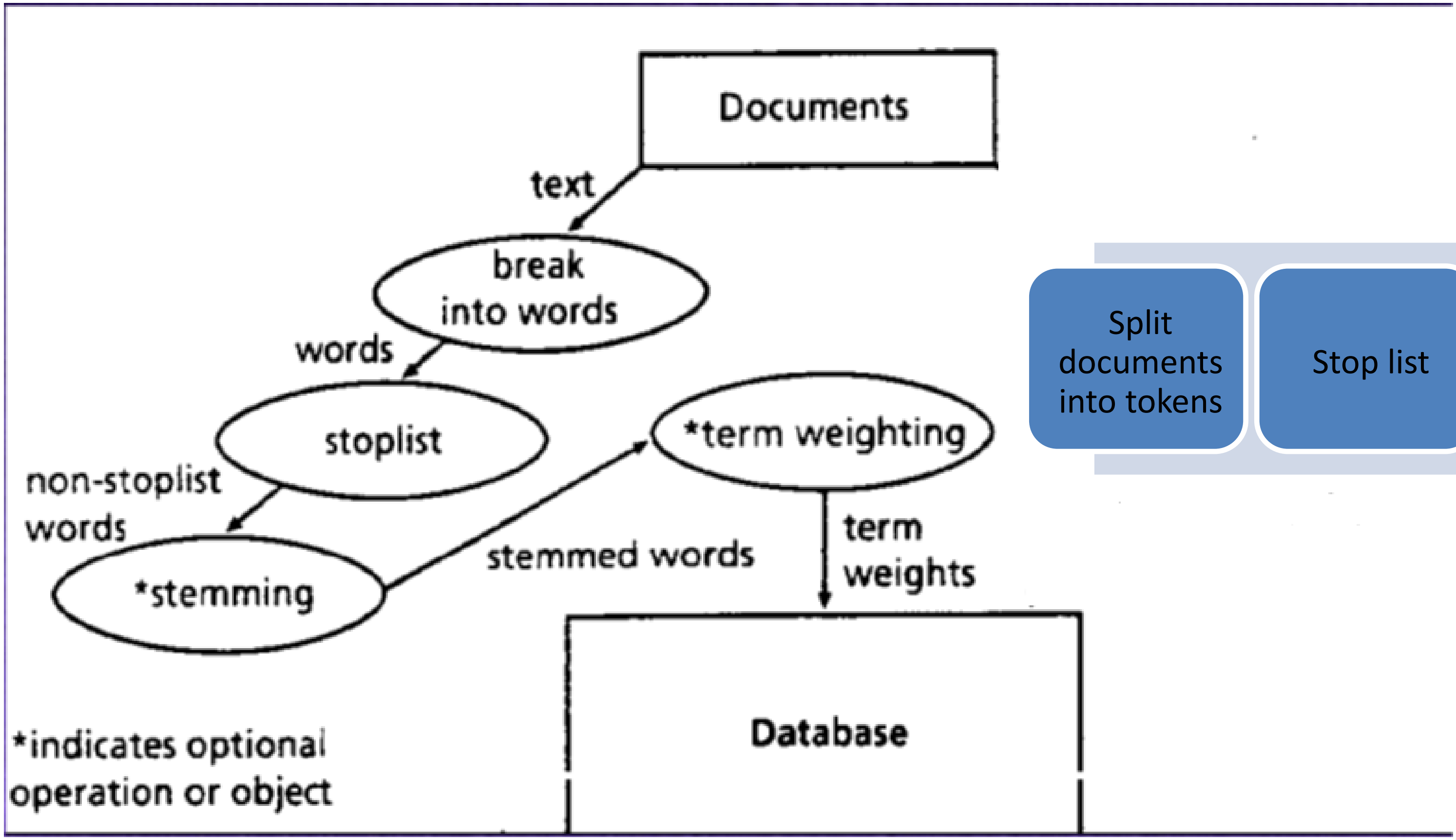


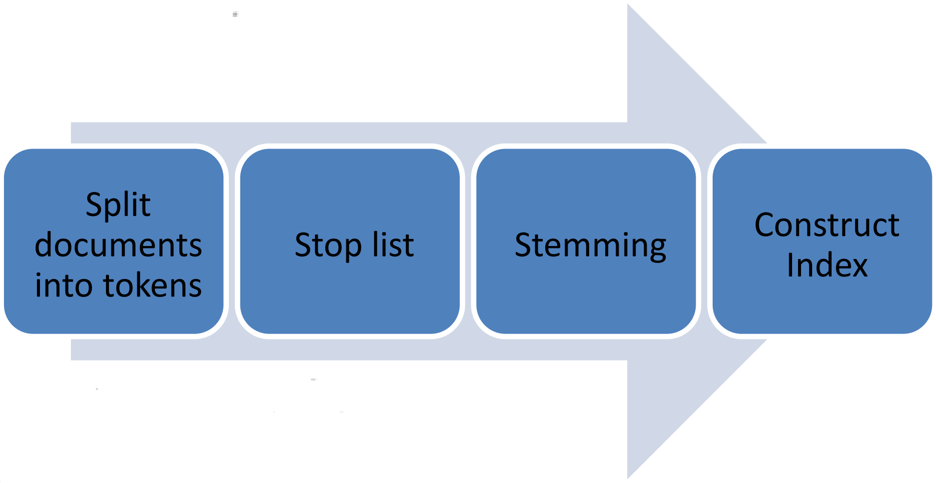
Retrieval Models

- A retrieval model specifies the details of:
  - Document Representation
  - Query Representation
  - Retrieval Function



Document Representation





Inverted Indexing

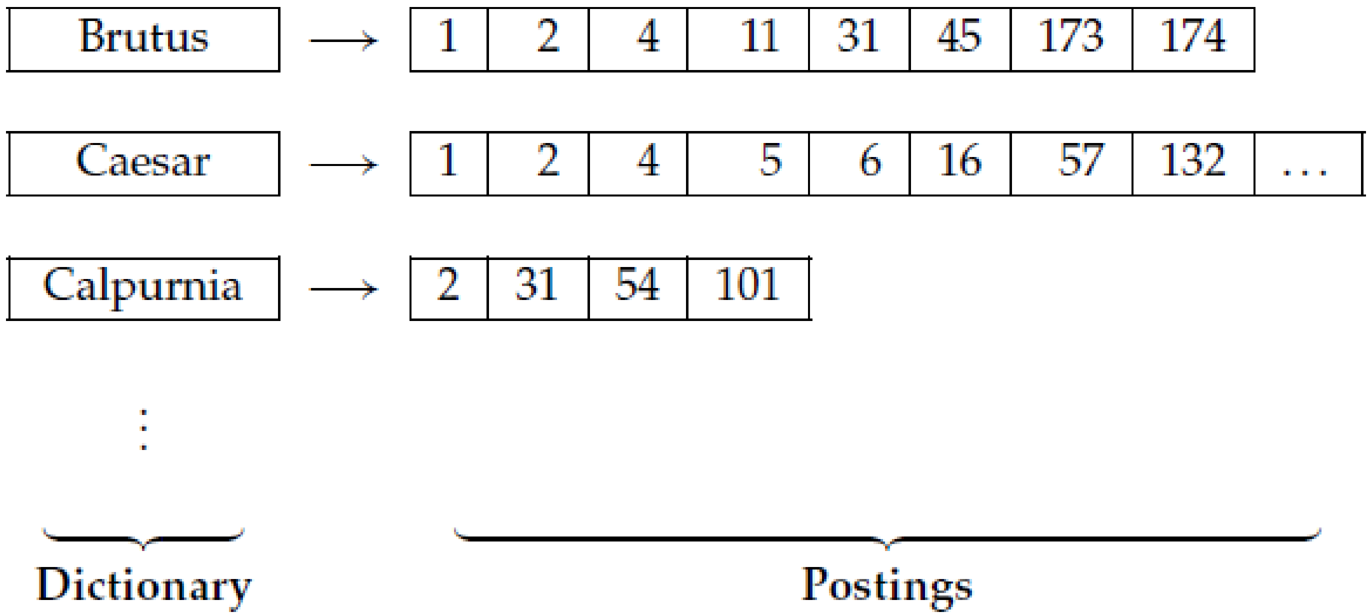


Figure: The two parts of an inverted index. The dictionary is commonly kept in memory, with pointers to each postings list, which is stored on disk.

Information Retrieval Models

Three “classic” models:

- Boolean Model
  - Documents and queries are sets of index terms
  -‘set theoretic’

- Vector Space Model
  - Documents and queries are documents in n-dimensional space
  -‘algebraic’

- Probabilistic Model

Additional models

- Extended Boolean
- Fuzzy matching
- Cluster-based retrieval
- Language models

Boolean Retrieval Model

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
Antony	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	
...							

Matrix element (*t, d*) is 1 if the play in column *d* contains the word in row *t*, and is 0 otherwise.

Query: Brutus and Caesar and not Calpurnia

110100 and 110111 and 101111 = 100100

The **answers** for this query: *Antony and Cleopatra* and *Hamlet*

Vector Space Model (VSM)

•The *vector space model* is one of the most widely used models for *ad-hoc* retrieval

•used in information filtering, information retrieval, indexing and relevancy rankings.

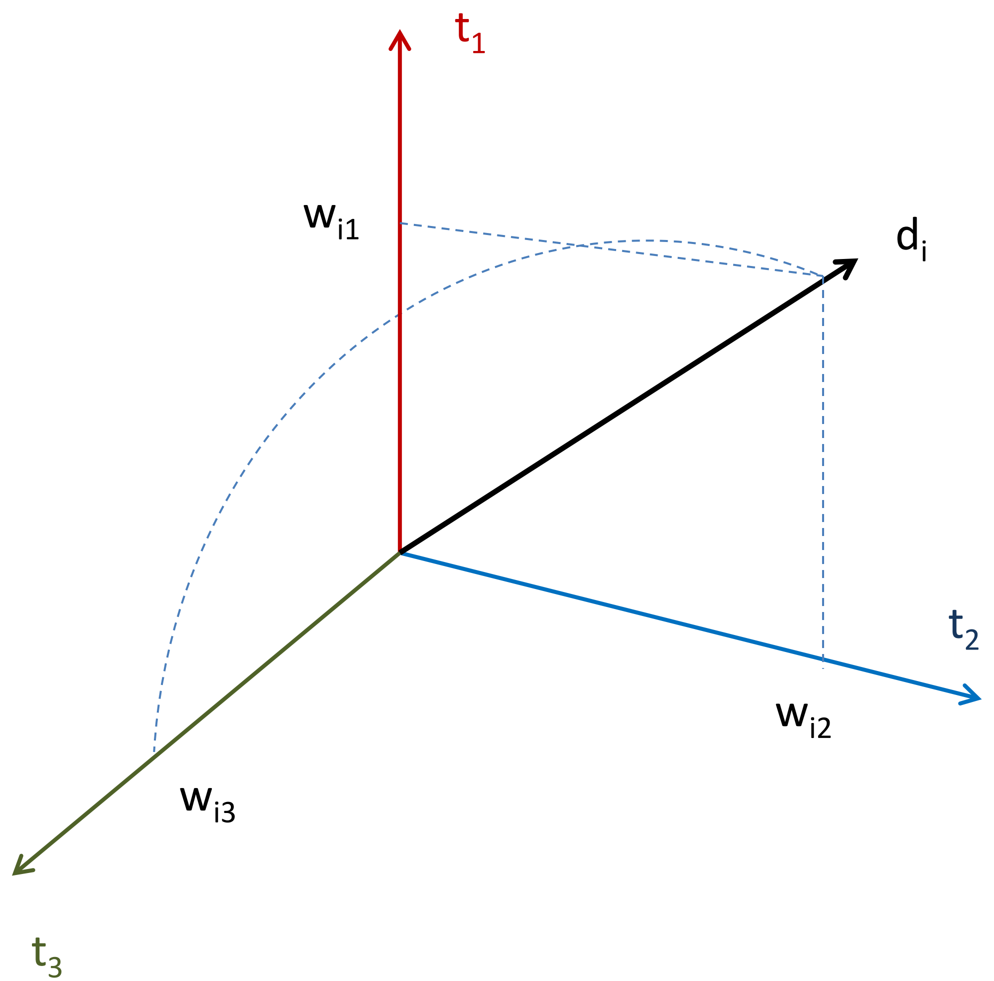
•Documents and queries are represented as vectors of weights:

$$\vec{d}_i = (w_{i,1}, w_{i,2} \ldots w_{i,t})$$

each $w_{i,j}$ is a weight for term *j* in document *i*

•Each dimension of the space corresponds to a separate term in the document collection

Vector Representation of a Document



Terms = t_1 , t_2 , t_3

$d_i = (w_{i1} , w_{i2} , w_{i3})$

$q = (t_1 , t_2 , t_3)$

How to weight words in the vector space

- Term weighting:
  - need more than simple 0 , 1 retrieval, considering large corpus
  - of 1,000 retrieved docs, which are the *most relevant*?
  - viewing each document as a vector of weights, towards ranking

- 2 things to consider:
  - The more times a term appears *within a document*, the more relevant that document is – (term frequency: *tf*)
  - If that term appears in every document *in the corpus*, it is less useful; rare terms matter more – (inverse doc frequency: *idf*)

- tf-idf: term frequency inverse document frequency

tf*idf weighting schema (1)

- tf = term frequency
  - frequency of a term/keyword in a document

$$\mathrm{tf}_{t,d}$$

  - occurrence of term *t* in document *d*

The higher the tf, the higher the importance (weight) for the doc.

	Doc1	Doc2	Doc3
car	27	4	24
auto	3	33	0
insurance	0	33	29
best	14	0	17

Table of tf values

tf*idf weighting schema (2)

- df = document frequency

$$\mathrm{df}_t.$$

  - number of documents in the collection that contain term t (posting)
  - distribution of the term

- idf = inverse document frequency
  - N is total number of documents in the corpus

$$\mathrm{idf}_t = \log \frac{N}{\mathrm{df}_t}$$

term	df_t	idf_t
car	18,165	1.65
auto	6723	2.08
insurance	19,241	1.62
best	25,235	1.5

collection of 806,791 documents

- the idf of a rare term is high, whereas the idf of a frequent term is likely to be low

tf*idf weighting schema (3)

The *tf–idf* weighting scheme assigns to term *t* a weight in document *d* given by :

$$w_{i,j} = \text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

In other words, $\text{tf–idf}_{t,d}$ assigns to term *t* a weight in document *d* that is:

1. *highest* when *t* occurs many times within a small number of documents (thus lending high discriminating power to those documents);

2. lower when the term occurs fewer times in a document, or occurs in many documents (thus offering a less pronounced relevance signal);

3. *lowest* when the term occurs in virtually all documents.

Score as ranking functions

- *score(q,d)*:

  The score of a document d is the sum, over all query terms, of the number of times each of the query terms occurs in d.

term	query				document			product
	tf	df	idf	$w_{t,q}$	tf	wf	$w_{t,d}$	
auto	0	5000	2.3	0	1	1	0.41	0
best	1	50000	1.3	1.3	0	0	0	0
car	1	10000	2.0	2.0	1	1	0.41	0.82
insurance	1	1000	3.0	3.0	2	2	0.82	2.46

$$\text{net score of } 0 + 0 + 0.82 + 2.46 = 3.28$$

  - with *N* = 1,000,000 documents

- tf-idf as a *score:*

$$\text{Score}(q, d) = \sum_{t \in q} \text{tf–idf}_{t,d}$$

Cosine Similarity Measure

- Finding similarity between documents in the vector space:

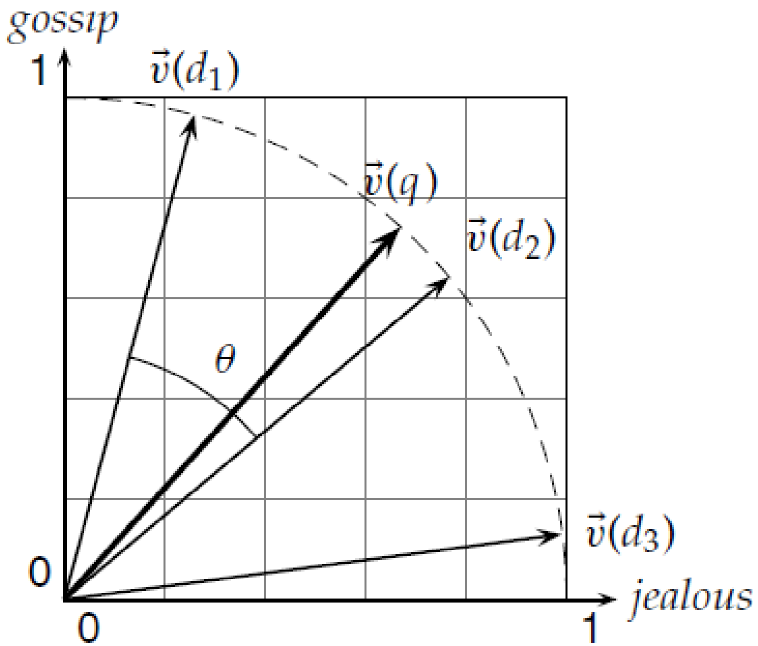
  - compute the *cosine similarity* of their vector representations

    sim(d1, d2) = $cos(\theta)$ = $\frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{|\vec{V}(d_1)||\vec{V}(d_2)|}$

  - value between 1 and -1 (cos0° = 1 , cos180° = -1)
  - compensates for the effect of document length

- Query represented as a document

$$\text{score}(q, d) = \frac{\vec{V}(q) \cdot \vec{V}(d)}{|\vec{V}(q)||\vec{V}(d)|}$$



Ranking with Cosine Similarity

term	SaS	PaP	WH
affection	115	58	20
jealous	10	7	11
gossip	2	0	6

Term frequencies in three novels.

Sense and Sensibility (SaS)
Pride and Prejudice (PaP)
Wuthering Heights (WH)

$\vec{V}(d_1)/|\vec{V}(d_1)|$

term	SaS	PaP	WH
affection	0.996	0.993	0.847
jealous	0.087	0.120	0.466
gossip	0.017	0	0.254

Term vectors for the three novels.

$$\text{sim}(\vec{v}(\text{SAS}), \vec{v}(\text{PAP})) = 0.999$$
$$\text{sim}(\vec{v}(\text{SAS}), \vec{v}(\text{WH})) = 0.888$$

Queries as vectors:

$q =$ jealous gossip

$v(q) =$ (0, 1, 1)

$\vec{v}(q) =$ (0, 1 /√2, 1 √2)

$=$ (0, 0.707, 0.707)

d	$\text{score}(q, d)$
SaS	0.074
PaP	0.085
WH	0.509

Term-document Matrix

	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
voyage	1	0	0	1	1	0
trip	0	0	0	1	0	1

The matrix is generally high-dimensional and sparse.

Dimension Reduction Techniques

- Token normalization.
- Removing stop words.
- Stemming and lemmatization.
- Latent Semantic Analysis/Indexing.

Token Normalization

- The process of standardizing tokens so that matches occur despite superficial differences in the character sequences (equivalence classes of tokens corresponding to the same term).
- Examples:
  - Capitalization/case-folding.
  - Remove hyphens, accents and diacritics, etc.
    - anti-discriminatory, antidiscriminatory => antidiscriminatory
    - U.S.A., USA => usa (but C.A.T. and cat?)

Removing Stop Words

- Dropping common terms in the entire document collection (very low *idf* value) that do not carry too much meaning.
- Generally implemented as a "curated" *stop list*:

a	an	and	are	as	at	be	by	for	from
has	he	in	is	it	its	of	on	that	the
to	was	were	will	with					

- Some reasonable queries may be disproportionately affected: *to be or not to be*, *let it be*.

Stemming and Lemmatization

- Reducing inflectional forms (e.g. verb conjugations) and other related forms of a word to a common base form.
  - *Stemming*: Usually a crude heuristic that chops off the ends of the words. It produces *pseudo-stems*.
    - believes => believ
  - *Lemmatization*: More elaborated by using a vocabulary and morphological analysis of words (requires understanding of context, POS tagging).
    - believes => believe

Latent Semantic Indexing

- Maps documents (and terms) to a low-dimensional space that encodes semantic associations (latent semantic space).
- Provides a solution to two common difficulties:
  - *Synonymy*: many terms refer to the same object (poor recall);
  - *Polysemy*: terms with more than one meaning (poor precision);
- Query-document similarity is computed in the transformed space (using e.g. cosine similarity).
- Performs a low-rank approximation of the term-document matrix (typical rank 100-300).

Singular Value Decomposition

Theorem. *Let r be the rank of the $M \times N$ matrix C. Then, there is a* singular-value decomposition (SVD *for short) of C of the form*

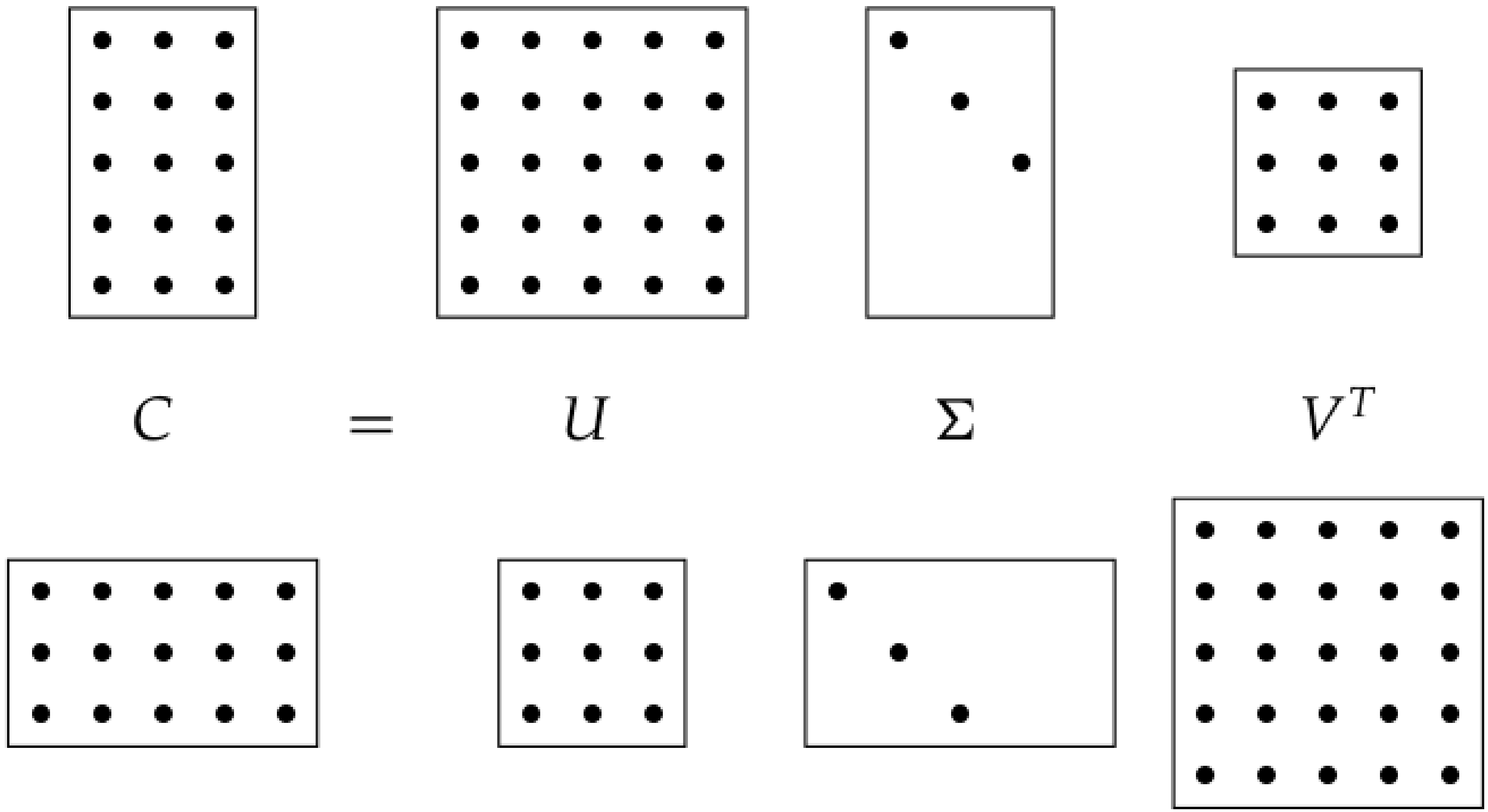
$$C = U \Sigma V^T,$$

where

1. *The eigenvalues $\lambda_1, \ldots, \lambda_r$ of CC^T are the same as the eigenvalues of $C^T C$;*
2. *For $1 \leq i \leq r$, let $\sigma_i = \sqrt{\lambda_i}$, with $\lambda_i \geq \lambda_{i+1}$. Then the $M \times N$ matrix Σ is composed by setting $\Sigma_{ii} = \sigma_i$ for $1 \leq i \leq r$, and zero otherwise.*

The values σ_i are referred to as the *singular values* of C.

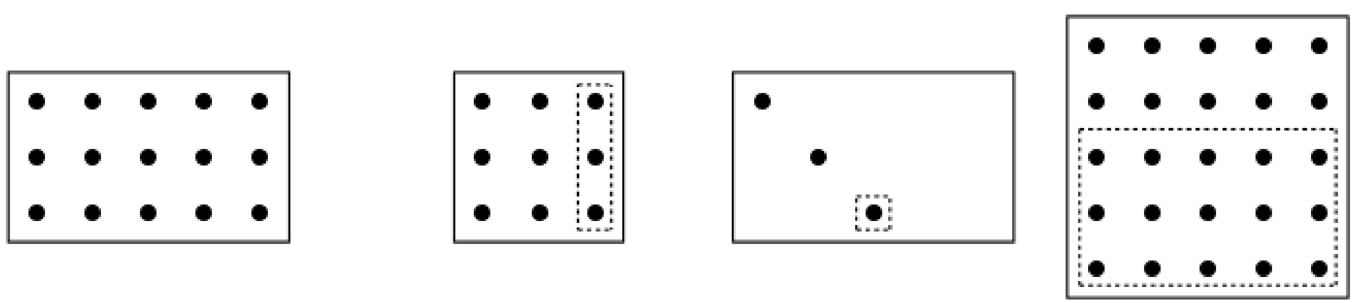
Singular Value Decomposition



(Rank-reduced) SVD

- Compute an approximation $C_k = U\Sigma_k V^T$ of C by replacing all but the first *k* singular values of Σ by zeros (which becomes Σ_k).

$$C_k = U \Sigma_k V^T$$



Latent Semantic Indexing

1) Initial term-document matrix:

	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
voyage	1	0	0	1	1	0
trip	0	0	0	1	0	1

3) Queries (and new documents) are mapped to the reduced space before computing similarity:

$$\vec{q}_k = \Sigma_k^{-1} U_k^T \vec{q}$$

2) Reduced term-document matrix (first *k* = 2 singular values):

	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41

Evaluation in IR

To measure the effectiveness of an IR system, it is necessary a test collection consisting of:

- a document collection;
- a suite of information needs, expressed as queries;
- a set of relevance judgements, generally binary assessments of relevant/non-relevant for each query-document pair.

For example, the test collections used for the Text Retrieval Conferences (TREC) since 1992.

Evaluation of the Retrieved Sets

- *Precision (P)*: fraction of retrieved documents that are relevant.
- *Recall (R)*: fraction of relevant documents that

$$\text{Precision} = \frac{\#(\text{relevant items retrieved})}{\#(\text{retrieved items})}$$

$$\text{Recall} = \frac{\#(\text{relevant items retrieved})}{\#(\text{relevant items})}$$

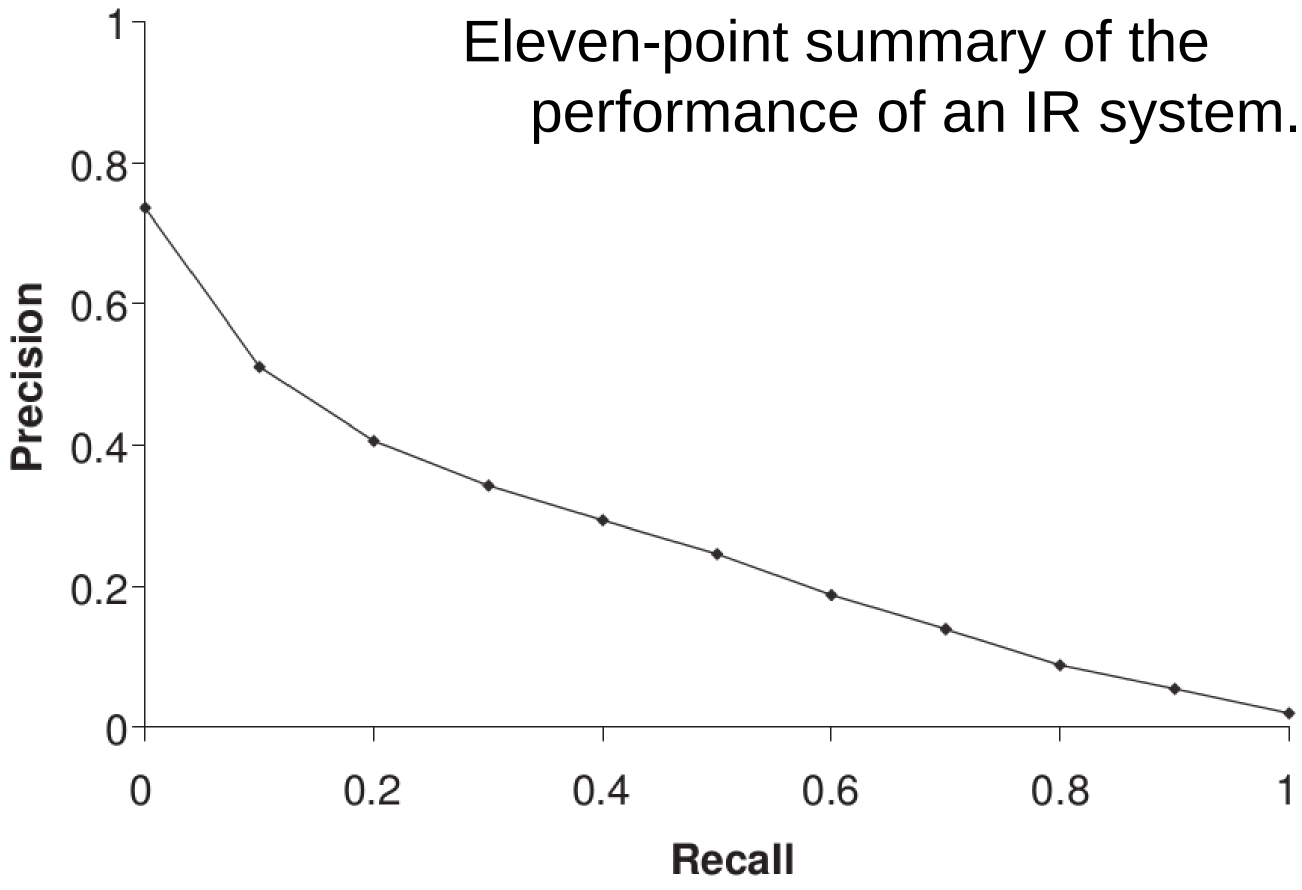
	relevant	nonrelevant
retrieved	true positives (tp)	false positives (fp)
not retrieved	false negatives (fn)	true negatives (tn)

$$P = tp/(tp + fp)$$

$$R = tp/(tp + fn)$$

•Precision-Recall Curves

Eleven-point summary of the performance of an IR system.



Evaluation of Ranked Results

- *Precision@k*: fraction of relevant documents among the first *k* retrieved documents.
- *Mean Average Precision (MAP)*: single-value measure across different recall levels.
  - For a single query, *average precision* is the average of *Precision@k* at the position *k* of every relevant document retrieved.
  - *MAP* is the *mean* of *average precision* for every query in the test collection.

R R N N N   N N N R N   R N N N R   N N N N R

Other Evaluation Measures

- *F-Measure*: a single-value measure that trades off *Precision* versus *Recall*.
- *R-Precision*: *Precision@r* in the ranking of results for a query that has *r* relevant documents.
- *Normalized Discounted Cumulative Gain (NDCG)*: designed for situations of non-binary notions of relevance.

Bibliography

- Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, *Introduction to Information Retrieval*, Cambridge University Press. 2008. Available at http://www-nlp.stanford.edu/IR-book/.

End