



BITS Pilani
Pilani Campus

Information Retrieval

Dr. Vijayalakshmi Anand
BITS Pilani



Session 1: Introduction

Date – 20 May 2023

Time – 11am to 1pm

These slides are prepared by the instructor Yates,, with grateful acknowledgement of and many others who made their course materials freely available online.

Session Content

- Objective of course
 - Evaluation Plan
 - What will we learn in this course?
 - Information Vs Data Retrieval
 - Retrieval process
 - Taxonomy of IR
 - Classic IR and Alternative models
-

Course Objectives

No	Course Objective
CO1	To understand structure and organization of various components of an IR system
CO2	To understand information representation models, term scoring mechanisms, etc. in the complete search system
CO3	To understand architecture of search engines, crawlers and the web search
CO4	To understand cross lingual retrieval and multimedia information retrieval

TEXT BOOK

C. D. Manning, P. Raghavan and H. Schutze. Introduction to Information Retrieval, Cambridge University Press, 2008.

<http://nlp.stanford.edu/IR-book/>

[https://nlp.stanford.edu/IR book/newsldes.html](https://nlp.stanford.edu/IR%20book/newsldes.html)

REFERENCE BOOKS

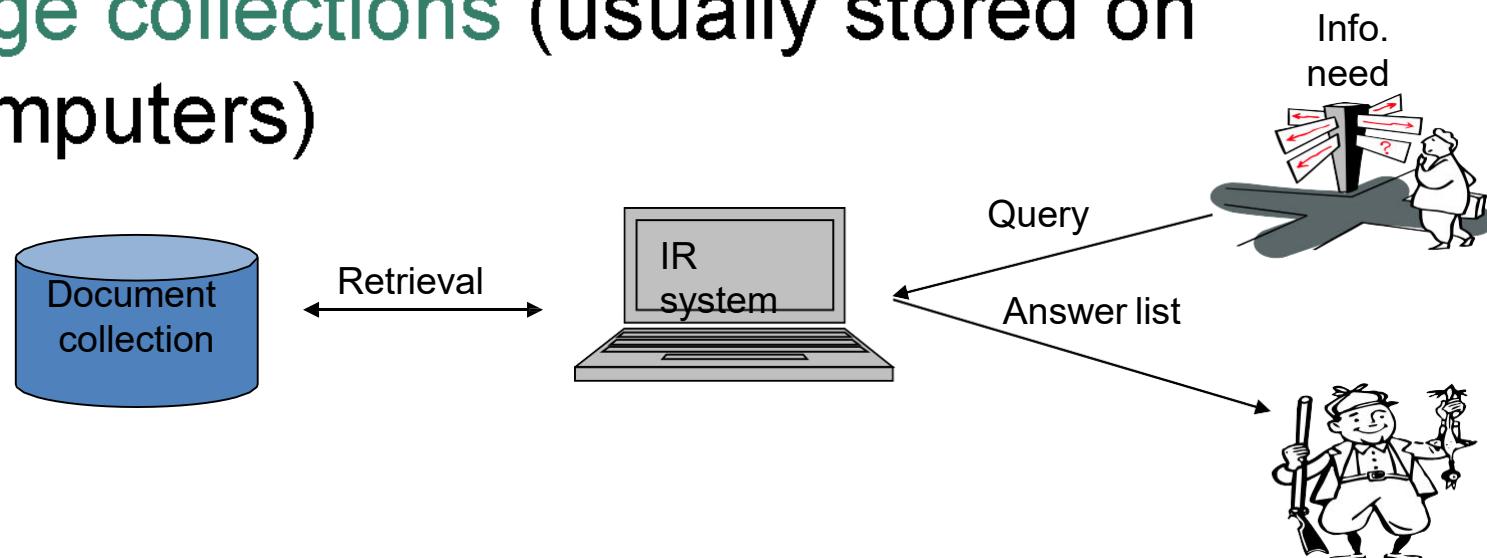
R1	Modern Information Retrieval, Ricardo Baeza-Yates and Berthier Ribeiro-Neto, Addison-Wesley, 2000. http://people.ischool.berkeley.edu/~hearst/irbook/
R2	Ricci, F.; Rokach, L.; Shapira, B.; Kantor, P.B. (Eds.), Recommender Systems Handbook. 1st Edition., 2011, 845 p. 20 illus., Hardcover, ISBN: 978-0-387-85819-7
R3	Cross-Language Information Retrieval by By Jian-Yun Nie Morgan & Claypool Publisher series 2010
R4	Multimedia Information Retrieval by Stefan M. Rüger Morgan & Claypool Publisher series 2010.
R5	Information Retrieval: Implementing and Evaluating Search Engines by S. Buttcher, C. Clarke and G. Cormack, MIT Press, 2010.
R6	Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data by B. Liu, Springer, Second Edition, 2011.

Evaluation Plan

Name	Type	Weight
Assignment	Online	15%
Quiz	Online	10%(Best2 out of 3)
Midsem exam	Online	30%
Comprehensive exam	Online	45%

Information Retrieval

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)



Document collection

➤ Document

Collection: text units

we have built an IR
system over.

- Usually documents
- But could be memos
 - book chapters
 - paragraphs
 - scenes of a movie
 - turns in a conversation...
 - Lots of them



Application

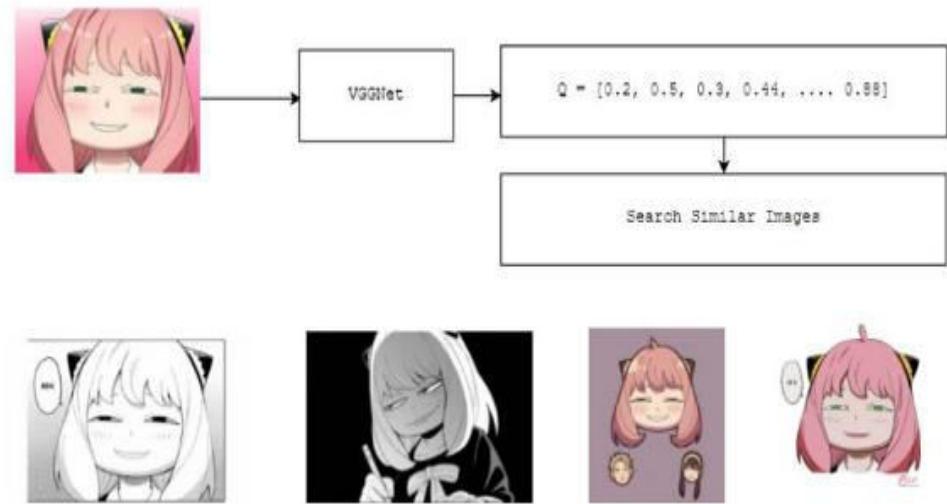


Search engine



Digital library

Automatic Extraction Order



Different types of data :Unstructured data

- Unstructured data means that a formal, easy for computer structure is missing,in contrast to the rigidly structured data used in DB style searching(product inventories ,personal records)
- This does not mean that there is no structure in the data
- Document structure (heading ,paragraphs, lists)
- Explicit markup formatting(html,XML)
- Linguistic structure (latent ,hidden)



Example

- Email
- Text files
- Social media and websites
- Mobile and communications data
- Media

- Scientific data
- Digital surveillance
- Satellite imagery

			
Text files and documents	Server, website and application logs	Sensor data	Images
			
Video files	Audio files	Emails	Social media data

Structured data

- Information stored in databases is known as **structured** data because it is represented in a strict format.

- The DBMS then checks to ensure that all data follows the structures and constraints specified in the schema.



Examples

- Library catalogues (date ,author ,place ,subject etc)
- Census records(birth, income,place ,employment etc)
- Inventory management system (id,name details)

id	name	age
1	Jim	28
2	Pam	26
3	Michael	42

id	subject
1	Languages
2	Track
3	Swimming
4	Computers

student_id	subject_id	grade
2	1	98
1	2	100
3	4	75
2	3	60
3	4	76
3	2	88

Differences

Structured data

Structured data stands for information that is highly organized, factual, and to-the-point.

Quantitative

Data warehouses

Relational databases

Several predetermined formats

Unstructured data

Unstructured data doesn't have any predefined structure to it and comes in all its diversity of forms.

Qualitative

Data lakes

Non-relational databases

A huge array of formats

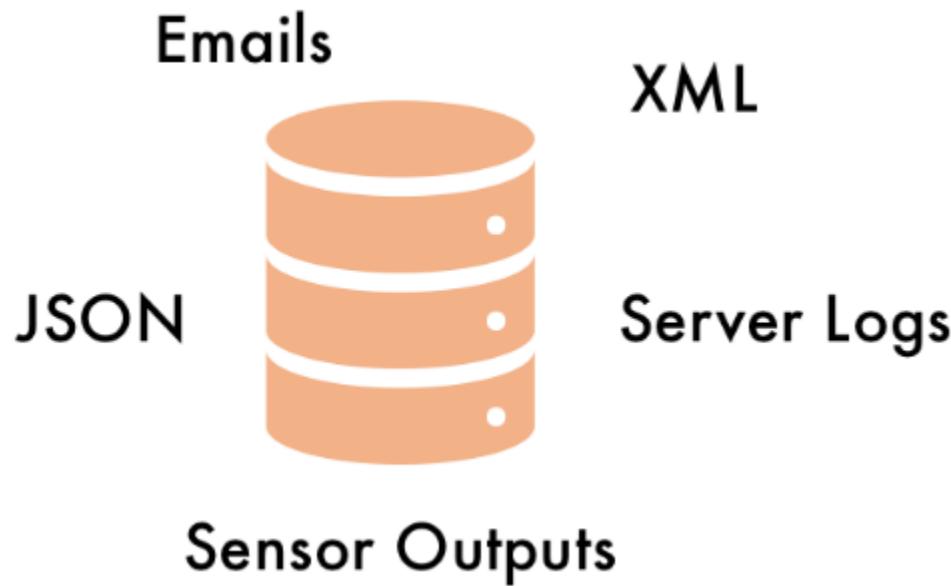


DATA SCIENCE

Semi-Structured Data

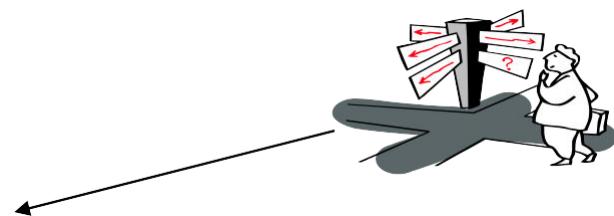
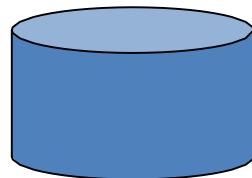
- Doesn't have relational and tabular data model
 - Will have tags
 - More complex than structured and less complex than unstructured
 - Easy to store
-

Example



Information Retrieval

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)



Information need

An information need is the topic about which the user desires to know more about.

- Known-item search:

Google

[Google Book Search](#) Google Book Search includes many recent publications published by governments worldwide.

- Precise information seeking search
 - Open-ended search (“topical search”)
-

Types of Information Needs

Retrospective

- “Searching the **past**”
- Different queries posed against a static collection

Prospective (Filtering)

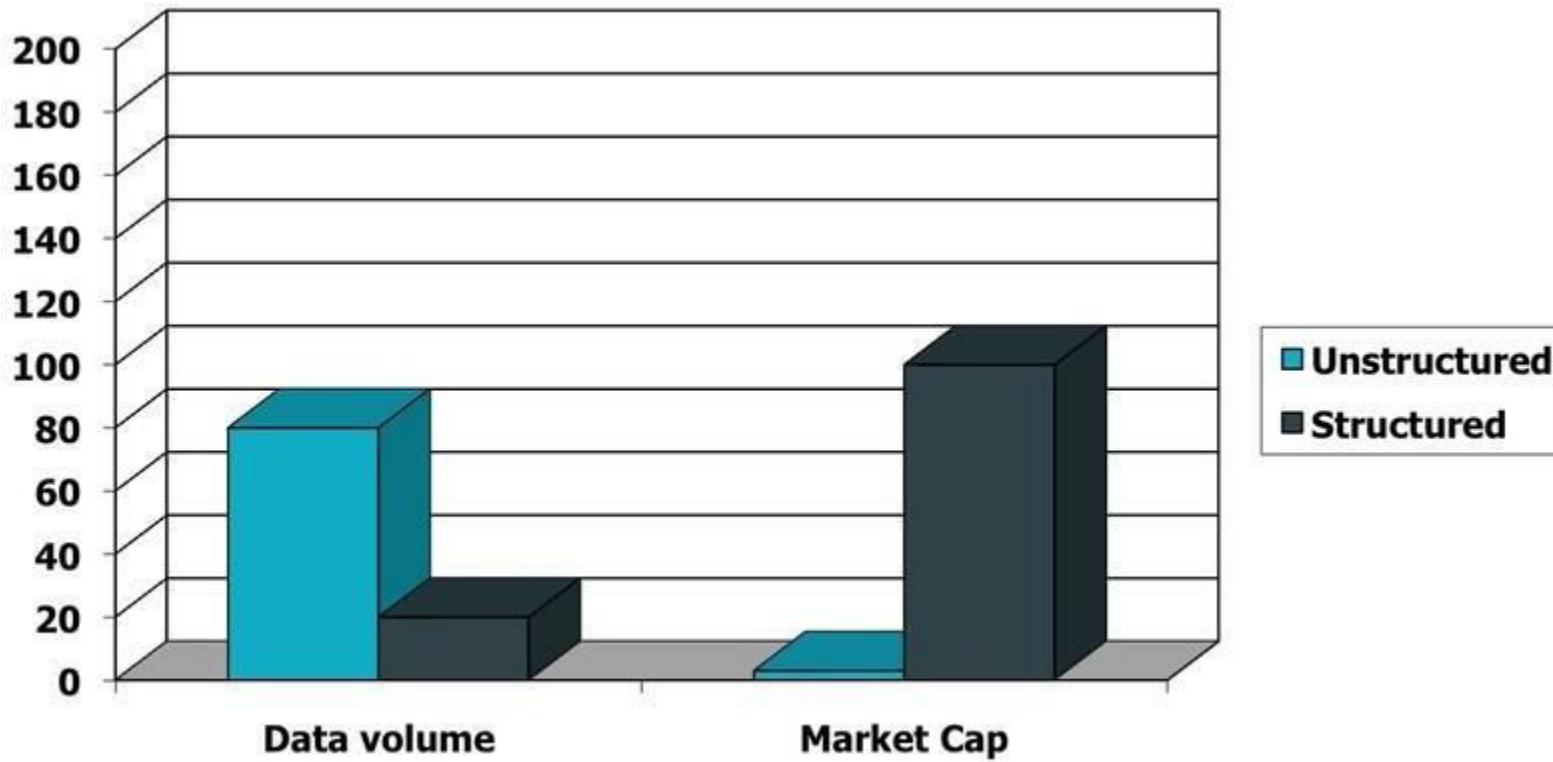
- “Searching the **future**”
- Static query posed against a dynamic collection
- Time dependent

Why information retrieval?

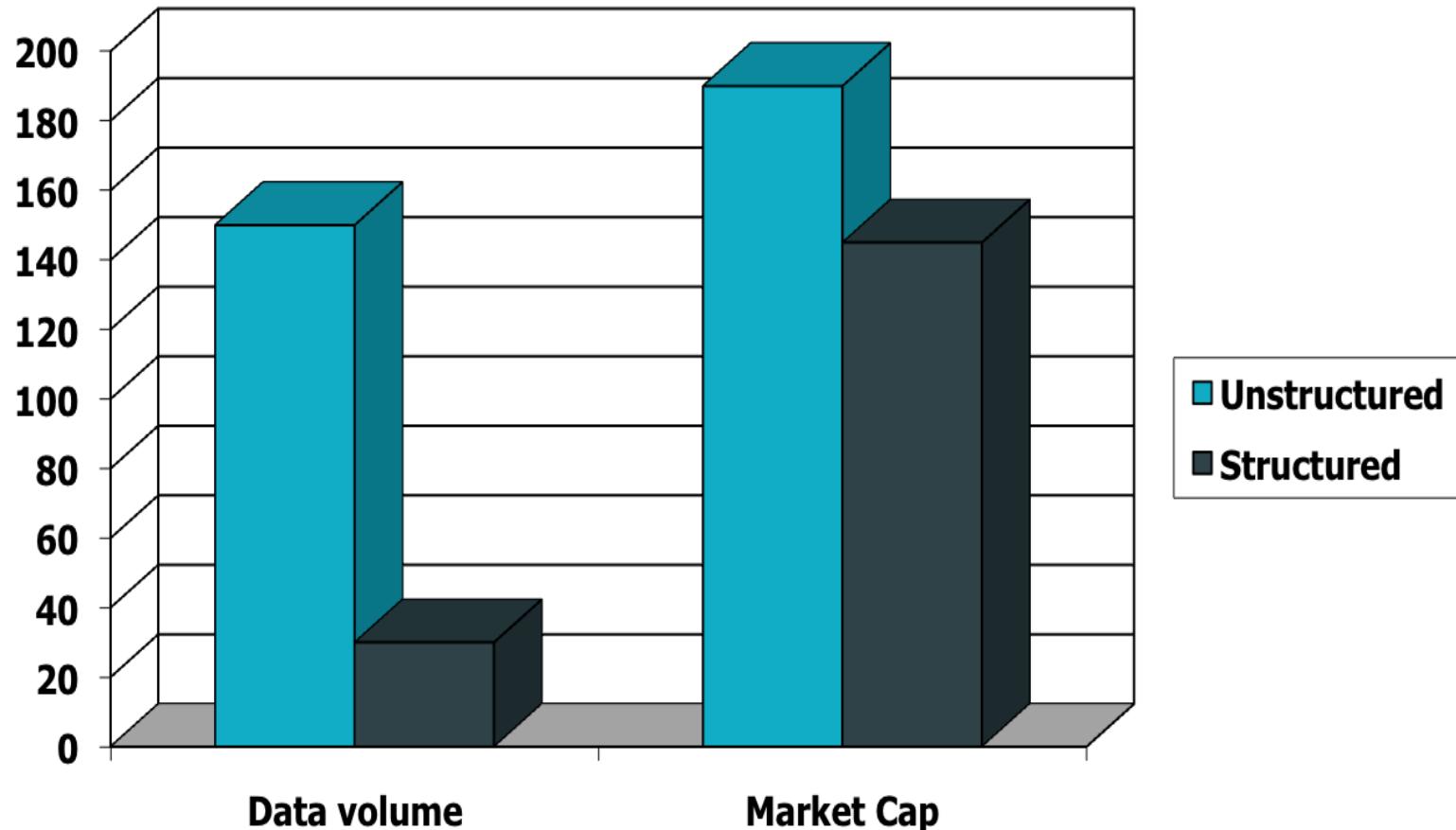
- Every online database ,every search engine everything that is searched online is based in some way or another on principles developed in IR

 - Understanding the basic of IR is a prerequisite for understanding how searching of online systems works
-

Unstructured (text) vs. structured (database) data in 1996



Unstructured (text) vs. structured (database) data in 2019



History of IR

1960-70's:

- Initial exploration of text retrieval systems for “small” corpora of scientific abstracts, and law and business documents.
- Development of the basic Boolean and vector-space models of retrieval.
- Prof. Salton and his students at Cornell University are the leading researchers in the area.

Contd...

1980's:

Large document database systems, many run by companies:

Lexis-Nexis

Dialog

MEDLINE

1990's:

Searching FTPable documents on the Internet

Archie server

WAIS(Wide Area Information Servers
)

Searching the World Wide Web

Yahoo

Altavista

1990's continued:

Organized Competitions

NIST TREC

Recommender Systems

Ringo

Amazon

NetPerceptions

2000's

Link analysis for Web Search
Google

Automated Information Extraction

Parallel Processing

Map/Reduce

Question Answering

TREC Q/A track

2000's continued:

Multimedia IR

Image

Video

Audio and music

Cross-Language IR

DARPA Tides

Document Summarization

Learning to Rank

Contd..

2010's

Intelligent Personal Assistants

Siri

Cortana

Google Now

Alexa

Complex Question Answering

IBM Watson

Deep Learning

Information Retrieval Vs. Data Retrieval

	Data Retrieval	Information Retrieval
Matching	Exact match	Partial match, best match
Inference	Deduction	Induction
Model	Deterministic	Probabilistic
Classification	Monothetic	Polythetic
Data	Database tables, structured	Free text, unstructured
Query language	Artificial, SQL, relational algebras.	Natural, Keywords, free text
Query specification	Complete	Incomplete
Items wanted	Matching	Relevant

Information Retrieval Vs. Data Retrieval

	Information Retrieval	Data Retrieval
Error Response	Insensitive	Sensitive
Results	Approximate matches	Exact matches
Results	Ordered by relevance	Unordered
Accessibility	Non-expert humans	Knowledgeable users or automatic processes

Information Retrieval Vs. Data Retrieval

1. Matching.

- In **data retrieval** we are normally looking for an **exact match**, that is, we are checking to see whether an item is or is not present in the file.
- **Ex: Select * from Student where per >= 8.0**
- In **information retrieval** more generally we want to find those items which **partially match** the request and then select from those a few of the best matching ones.
- **Ex: Student having 8 or > 8 CGPA**

Information Retrieval Vs. Data Retrieval

2. Inference

- In data retrieval is of the simple deductive kind, that is, $a \in b$ and $b \in c$ then $a \in c$.
- In information retrieval it is of inductive inference; relations are only specified with a degree of certainty or uncertainty and hence our confidence in the inference is variable.

3. Model

- Data retrieval is deterministic but information retrieval is probabilistic.
- Frequently Bayes' Theorem is invoked to carry out inferences in IR, but in DR probabilities do not enter into the processing.

Information Retrieval Vs. Data Retrieval

4 .Classification:

- In DR most likely monothetic classification is used.
 - That is, one with classes defined by objects possessing attributes both necessary and sufficient to belong to a class.
-
- In IR, polythetic classification is mostly used.
 - Each individual in a class will possess only a proportion of all the attributes possessed by all the members of that class..

Information Retrieval Vs. Data Retrieval

5.Query Language:

- The query language for DR is one with restricted syntax and vocabulary.
- In IR we prefer to use natural language although there are some notable exceptions.

6.Query Specification:

- In DR the query is generally a complete specification of what is wanted,
- In IR it is invariably incomplete.

Information Retrieval Vs. Data Retrieval

7. Items wanted :

- In IR we are searching for relevant documents as opposed to exactly matching items in DR.

8. Error response:

- DR is more sensitive to error in the sense that, an error in matching will not retrieve the wanted item which implies a total failure of the system.
- In IR small errors in matching generally do not affect performance of the system significantly

Issues with Information Retrieval?

Information Retrieval deals with **uncertainty** and **vagueness** in information systems.

- **Uncertainty:** available representation does typically not reflect true semantics/meaning of objects (text, images, video, etc.)
 - **Vagueness:** information need of user lacks clarity, is only vague expressed in query, feedback or user actions.
 - Differs conceptually from database queries!
-

Interesting web search facts

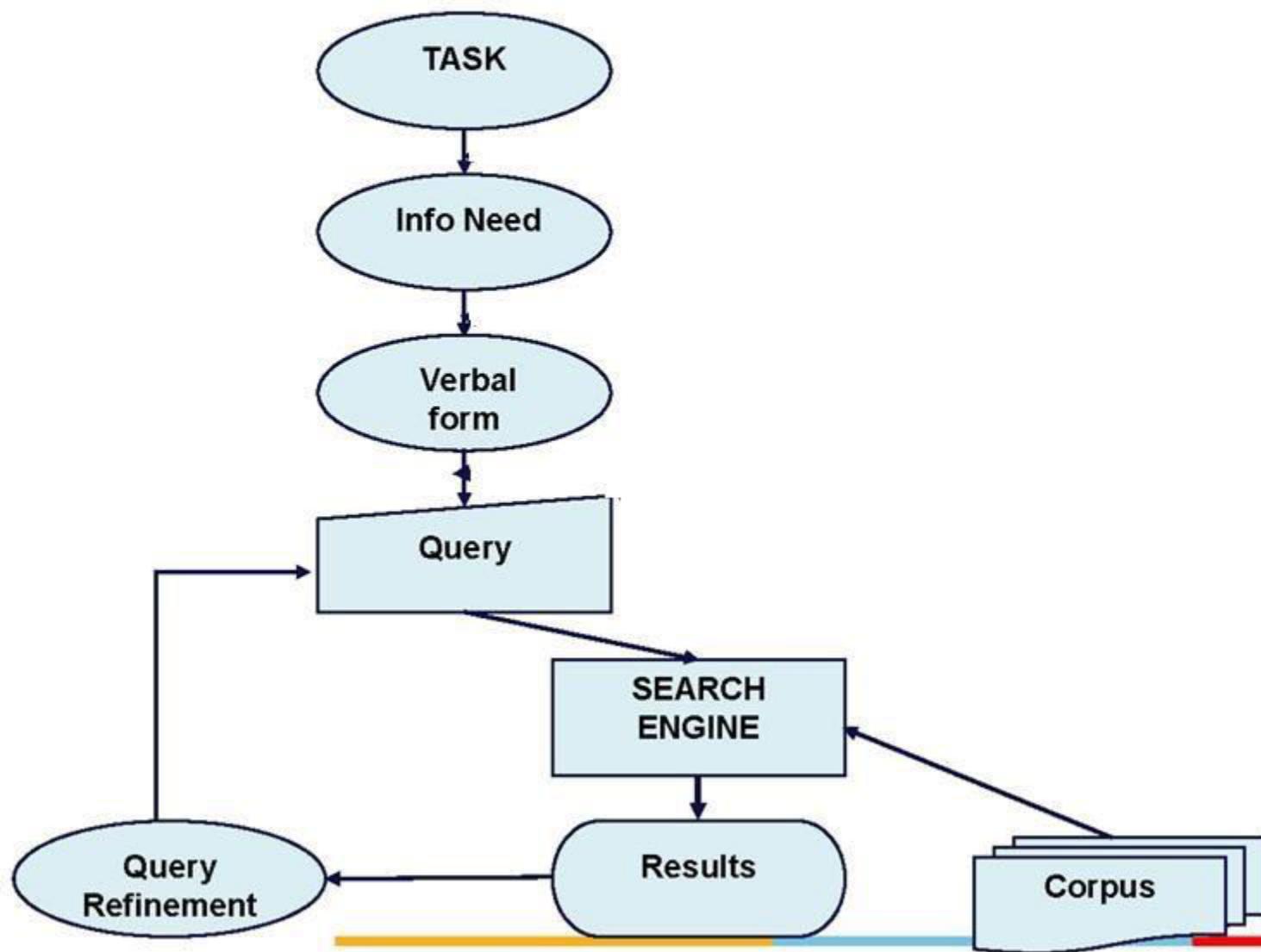
- 3.5 billion Google searches are made every day. ([Internet Live Stats](#))
- Volume of Google searches grows by roughly 10% every year. ([Internet Live Stats](#))
- Every year, somewhere between 16% and 20% of Google searches are new— they've never been searched before. ([Internet Live Stats](#))
- 90% of searches made on desktops are done via Google. ([Statista](#))
- 35% of product searches start on Google. ([eMarketer](#))
- 34% of “near me” searches done via desktop and tablets result in store visits. ([HubSpot](#))
- The average Google search session lasts just under a minute. ([Moz](#))
- Dating & Personal Services advertisers drive the highest CTRs on paid Google results. Just over 6% of their impressions turn into clicks! ([WordStream](#))
- Organic Google results with 3-4 words in the title drive higher CTRs than organic results with 1-2 words in the title. ([Smart Insights](#))
- Google has indexed hundreds of billions of web pages. All told, the index is about 100,000,000 GB large. ([Google](#))

Information Retrieval

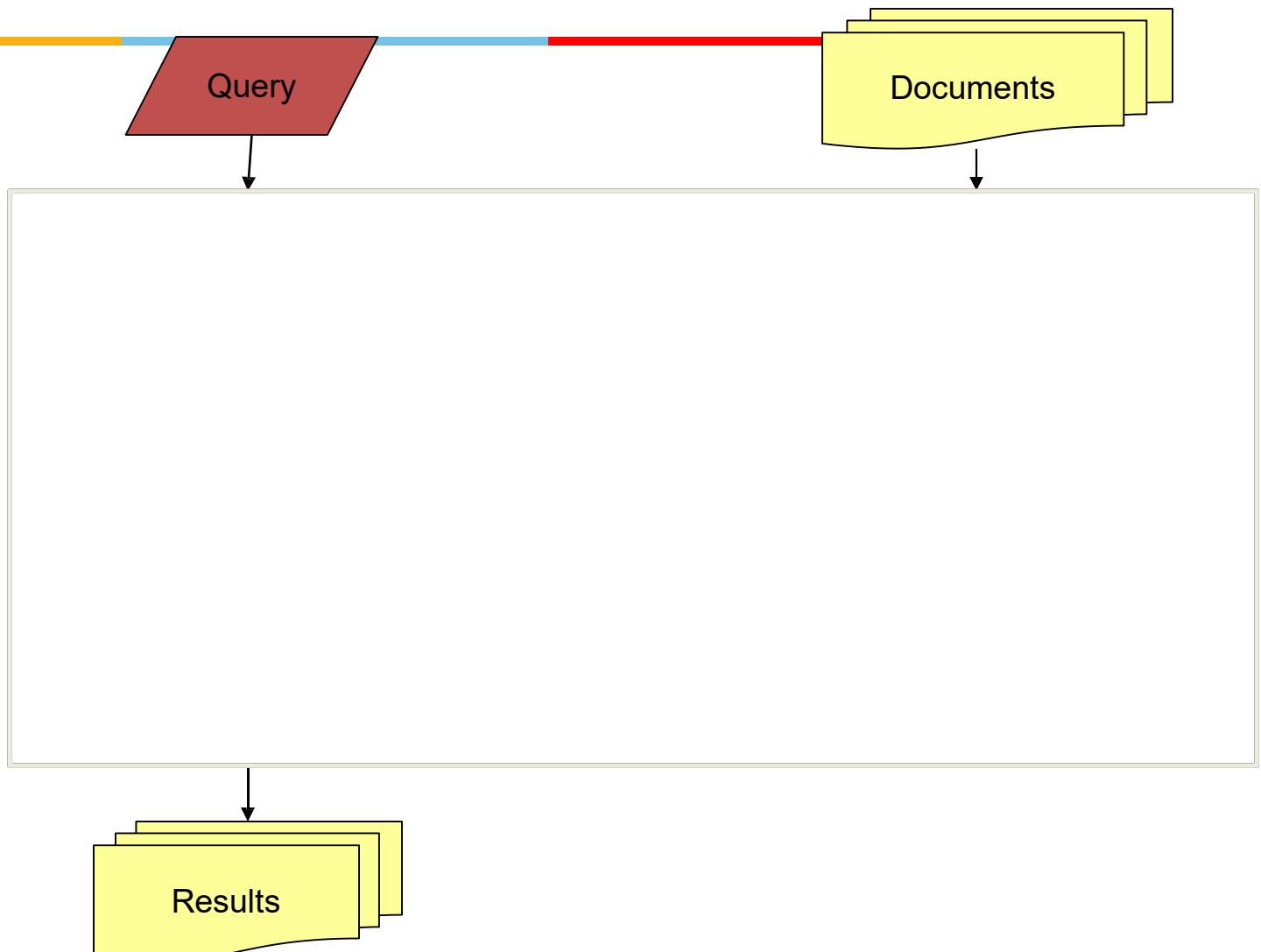
IR systems can also be distinguished by the scale at which they operate

- Three prominent scales
 - Web Search
 - Personal Information Retrieval
 - Enterprise, institutional, domain-specific search

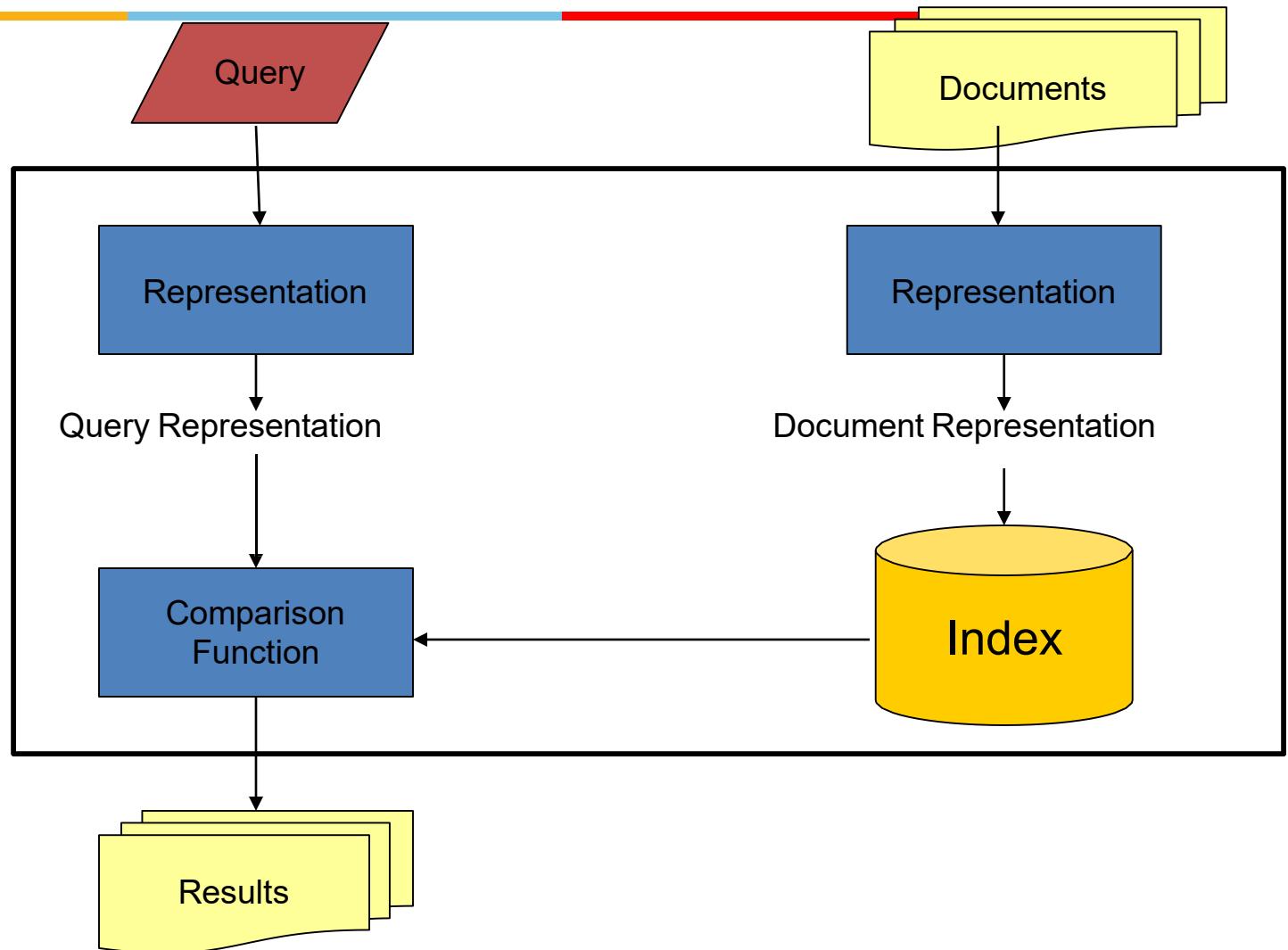
Classic Search Model



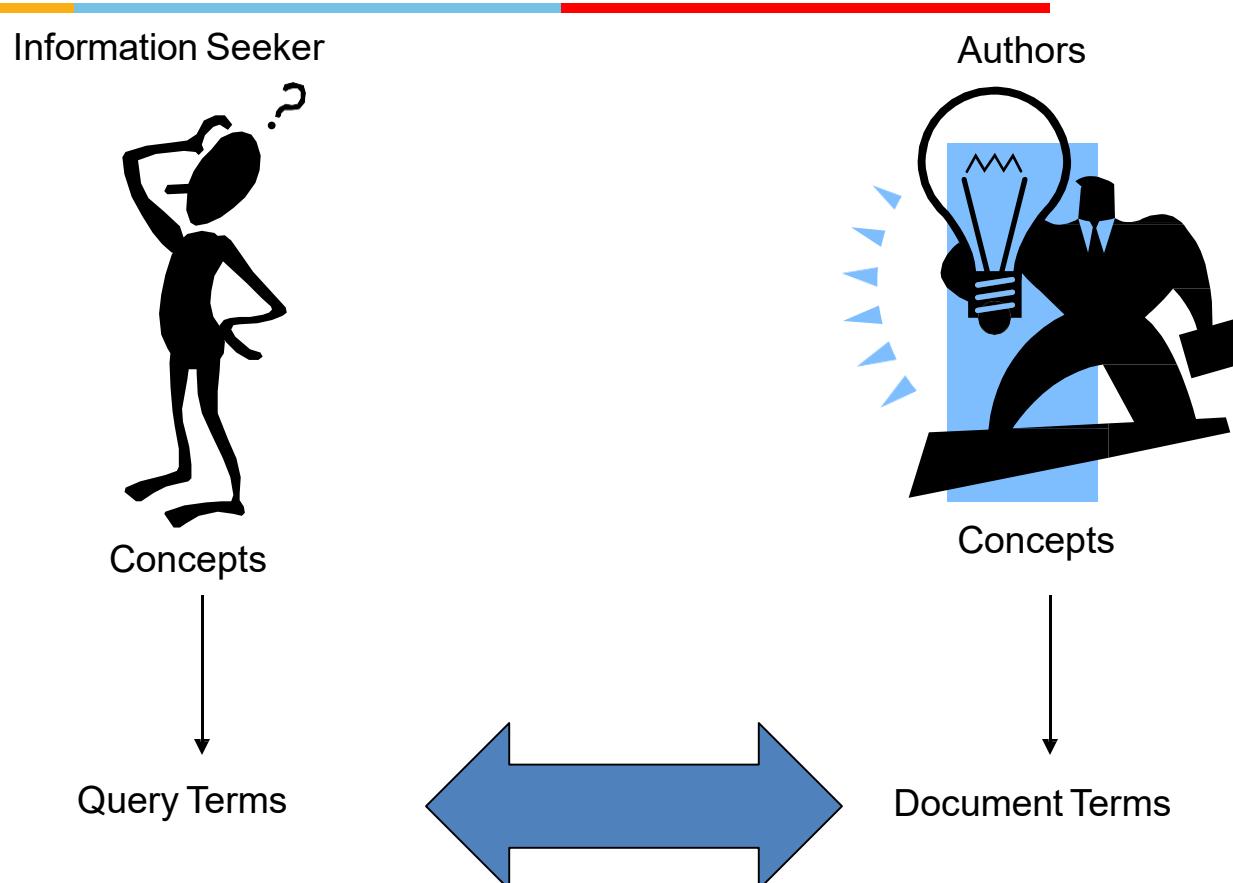
The IR Black Box



Inside The IR Black Box



The Central Problem in IR



Do these represent the same concepts?

Tasks of IR

Index the documents in the collection (offline)

Process the query

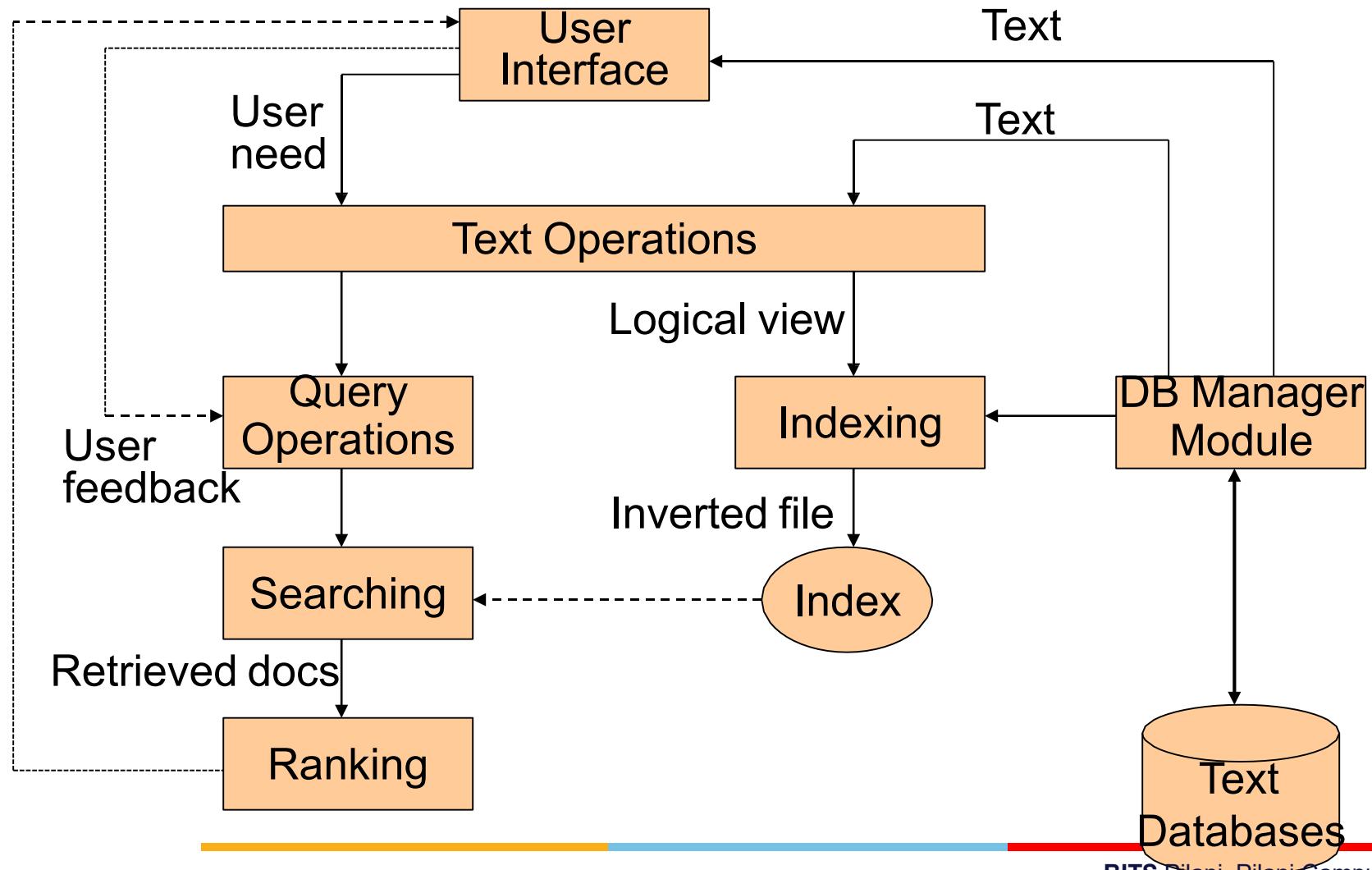
Evaluate **Similarity** and find RANKs

Find documents most closely matching the query (relevant documents)

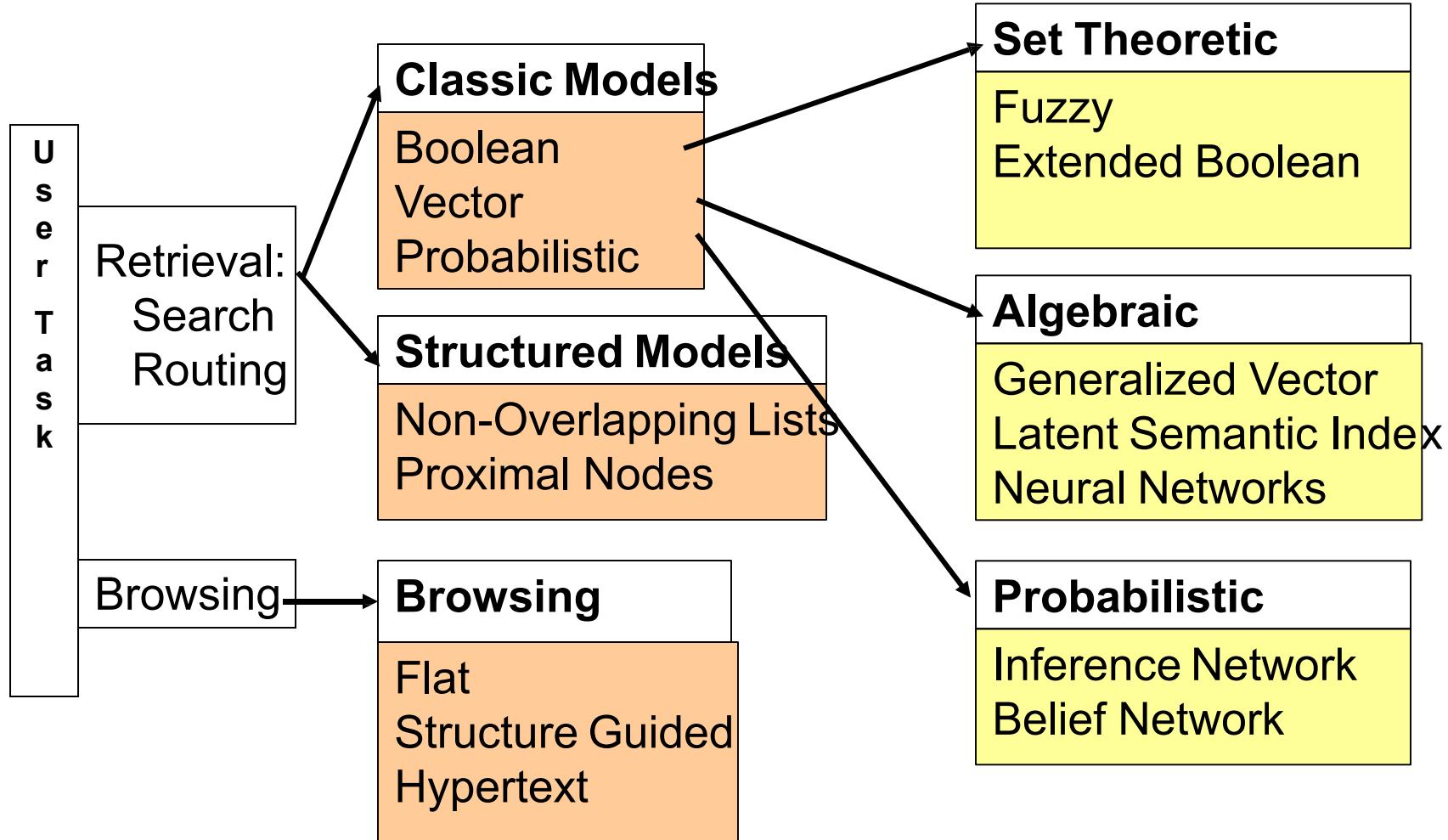
Display results

E.g., user may refine the query (feedback)

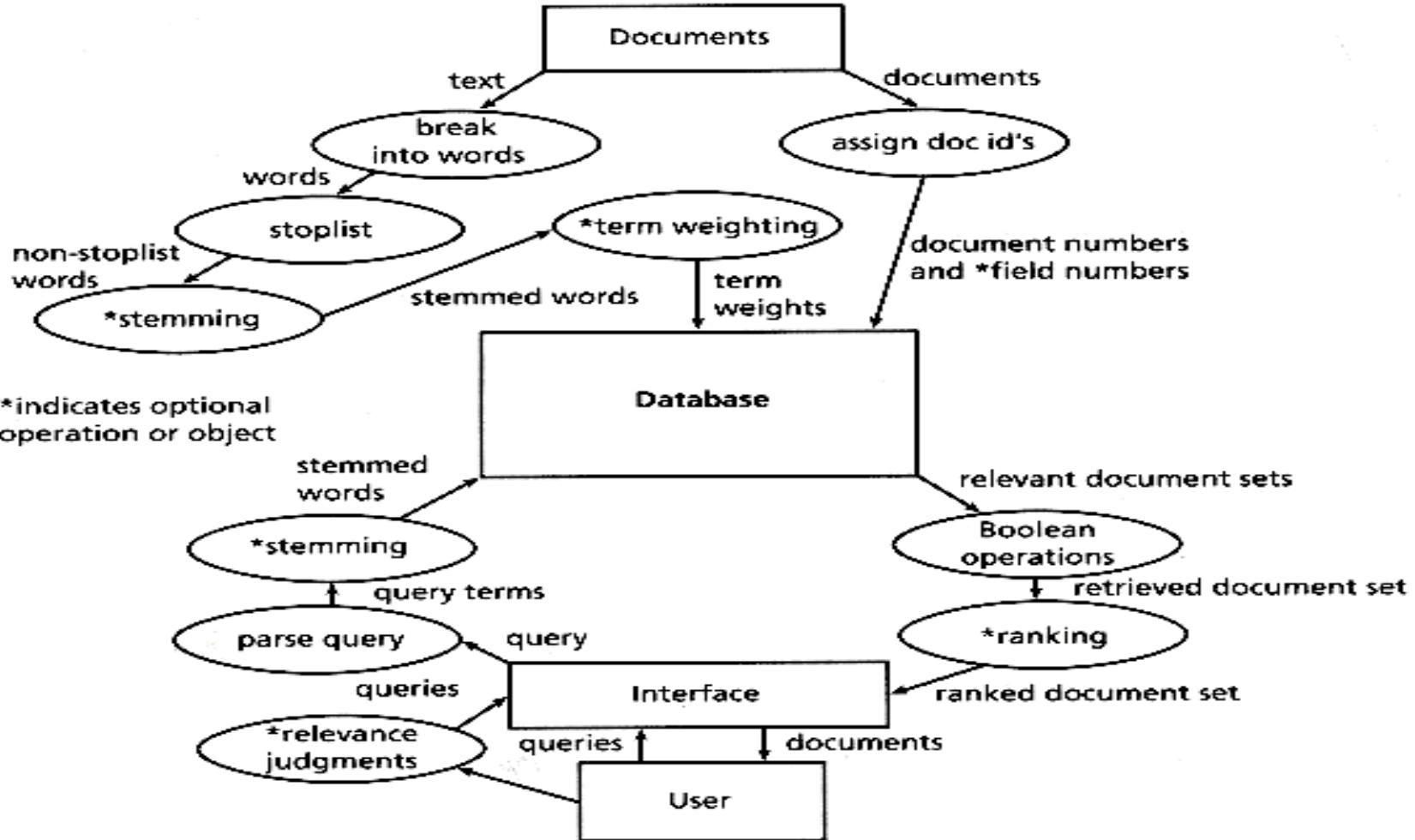
The Process of Retrieving Information



A Taxonomy of IR Models



Functional View of Paradigm IR System



Other IR related tasks

- Automated document categorization
 - Information filtering (spam filtering)
 - Information routing
 - Automated document clustering
 - Recommending information or products
 - Information extraction
 - Information integration
 - Question answering
-

Related Areas

- Database Management
- Library and Information Science
- Artificial Intelligence
- Natural Language Processing
- Machine Learning

What will we learn in this course?

- Boolean Retrieval
- Index Construction
- Vector Space Models
- Text Mining
- Ranking
- Web crawlers
- Cross Lingual IR
- Multimedia IR
- Recommendation Systems
- NeuralIR

Evaluation measures

Retrieved	True positives (tp)	False positives (fp)
Not Retrieved	False negatives (fn)	True negatives (tn)
Relevant		Non Relevant

$$recall = \frac{\text{Number of relevant documents retrieved}}{\text{Total number of relevant documents}} = tp/(tp + fn)$$

$$precision = \frac{\text{Number of relevant documents retrieved}}{\text{Total number of documents retrieved}} = tp/(tp + fp)$$

(How many correct selections?) Accuracy = $(tp + tn)/(tp + fp + fn + tn)$



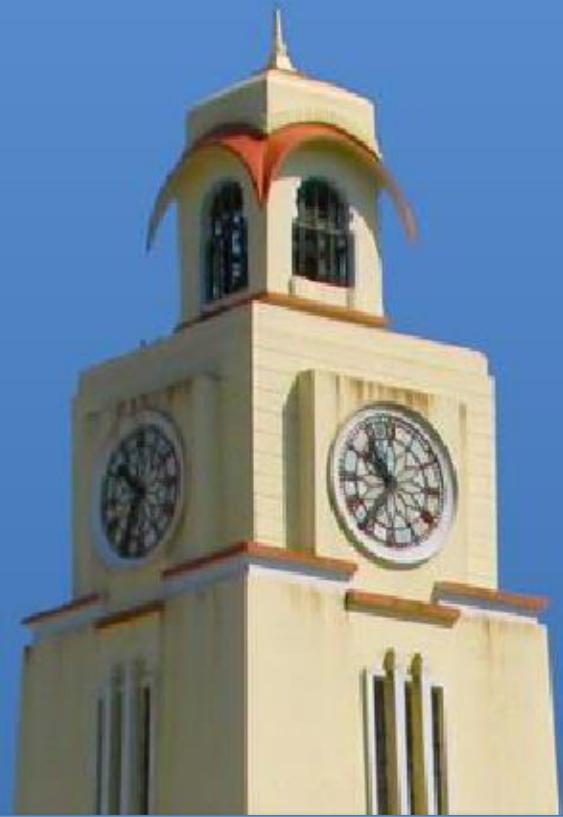
Thank you



BITS Pilani
Pilani Campus

Information Retrieval

Dr. Vijayalakshmi Anand
BITS Pilani



Session 2: Boolean Retrieval

Date – 27th May 2023

Time – 11.00am to 1.00 pm

These slides are prepared by the instructor Prof. Manning,, with grateful acknowledgement of and many others who made their course materials freely available online.

Agenda

Boolean Retrieval (T1 Chapter 1 and 2)

- Inverted index
- Processing Boolean queries
- Boolean Vs Ranked retrieval
- Term vocabulary and postings lists
- Phrase queries

Boolean Model

- Binary decision criterion
- Data retrieval model
- Advantage
 - clean formalism, simplicity
- Disadvantage
 - It is not simple to translate an information need into a Boolean expression
 - exact matching may lead to retrieval of too few or too many documents

Information Retrieval: Example

Which plays of Shakespeare contain the words ***Brutus AND Caesar*** but ***NOT Calpurnia***?

grep all of Shakespeare's plays for ***Brutus*** and ***Caesar***, then strip out lines containing ***Calpurnia***?

Why is that not the answer?

- Slow (for large corpora)
- **NOT *Calpurnia*** is non-trivial
- Other operations (e.g., find the word ***Romans*** near ***countrymen***) not feasible
- Ranked retrieval (best documents to return)

Term-document incidence

	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

*Brutus AND Caesar BUT NOT
Calpurnia*

1 if play contains word,
0 otherwise

Boolean retrieval Example

We take the vectors for Brutus, Caesar and Calpurnia, complement the last, and then do a bitwise AND:

110100 AND 110111 AND 101111 = 100100

The answers for this query are thus Antony and Cleopatra and Hamlet

How good are the retrieved docs?

Precision : Fraction of retrieved docs that are relevant to user's information need

Recall : Fraction of relevant docs in collection that are retrieved

The Inverted Index

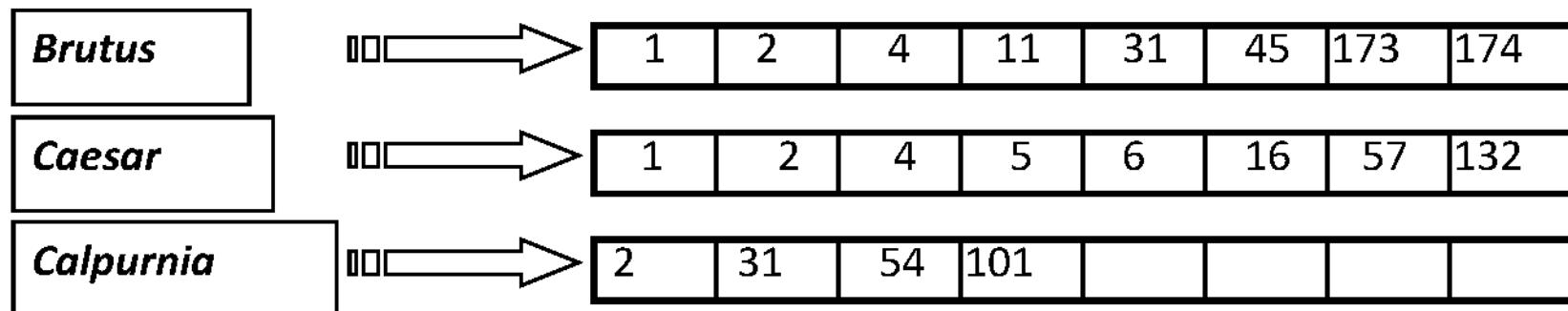
The key data structure underlying modern IR

Inverted index

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

- Identify each by a **docID**, a document serial number

Can we use fixed-size arrays for this?

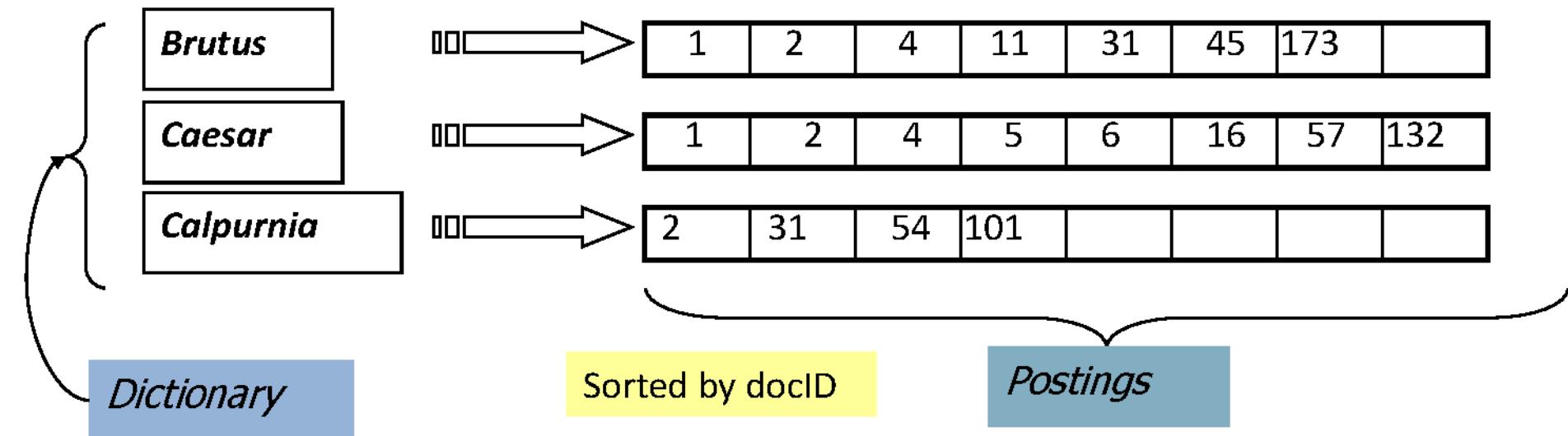


What happens if the word **Caesar** is added to document 14?

Inverted index

We need variable-size postings lists

- On disk, a continuous run of postings is normal and best
- In memory, can use linked lists or variable length arrays
 - Some tradeoffs in size/ease of insertion



Inverted index construction

Documents to be indexed



Friends, Romans, countrymen.

⋮

Tokenizer

Token stream

Friends

Romans

Countrymen

Linguistic modules

Modified tokens

friend

roman

countryman

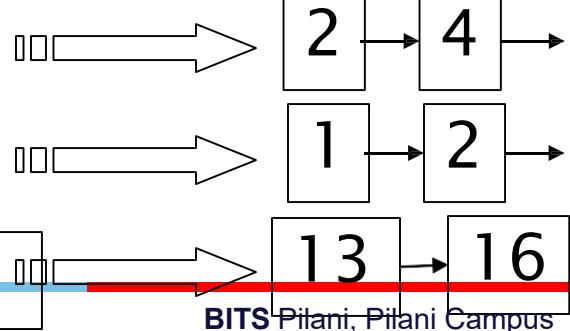
Indexer

Inverted index

friend

roman

countryman



Initial stages of text processing

- Tokenization
 - Cut character sequence into word tokens
 - Deal with “*John’s*”, a **state-of-the-art solution**
- Normalization
 - Map text and query term to same form
 - You want **U.S.A.** and **USA** to match
- Stemming
 - We may wish different forms of a root to match
 - **authorize, authorization**
- Stop words
 - We may omit very common words (or not)
 - **the, a, to, of**

Indexer steps: Token sequence

Sequence of (Modified token, Document ID) pairs.

Doc 1

Doc 2

I did enact Julius
Caesar I was killed
i' the Capitol;
Brutus killed me.

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

Indexer steps: Sort

Sort by terms

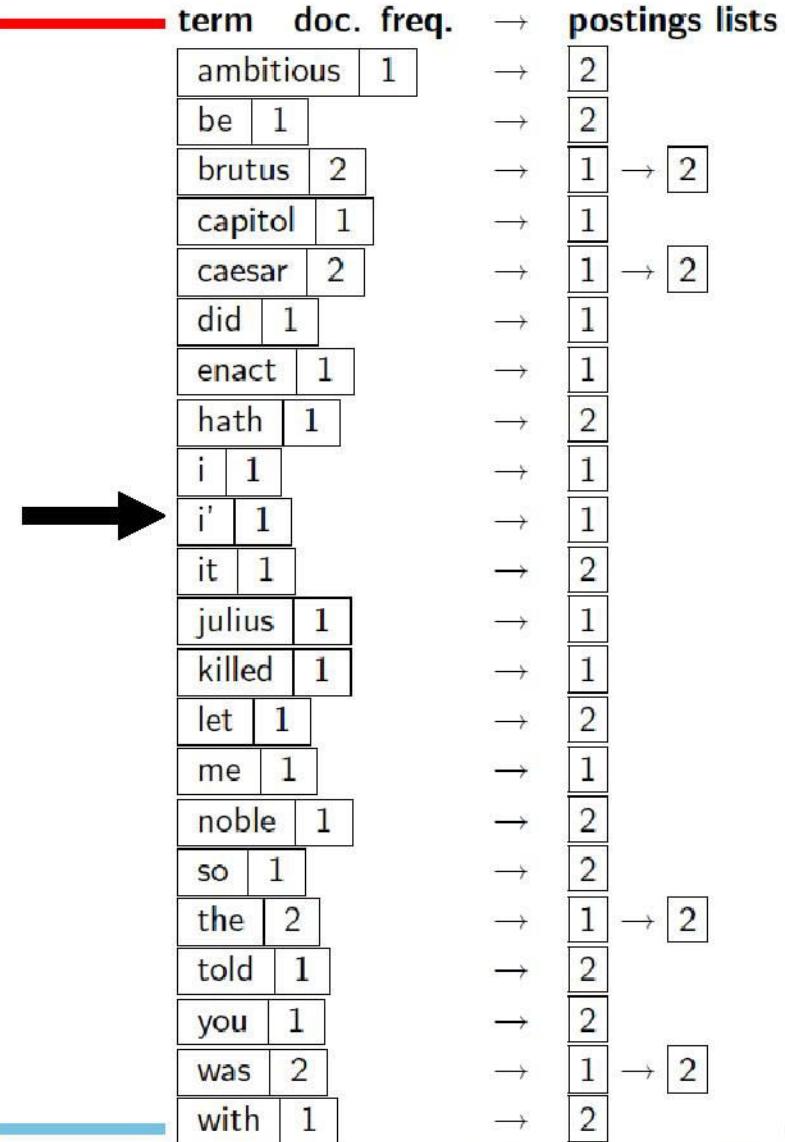
- And then docID

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

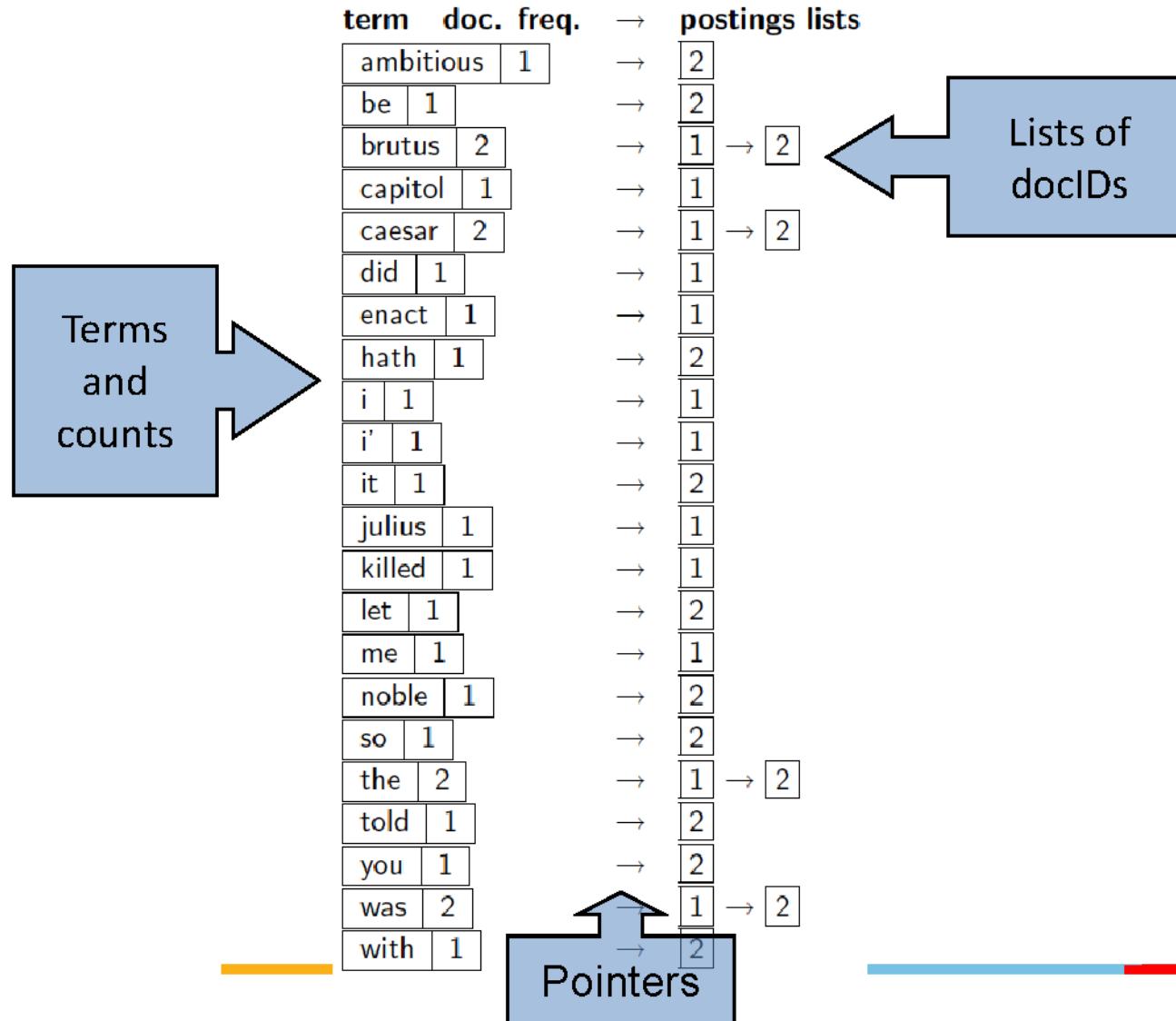
Indexer steps: Dictionary & Postings

- Multiple term entries in a single document are merged
- Split into Dictionary and Postings
- Doc frequency information is added

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

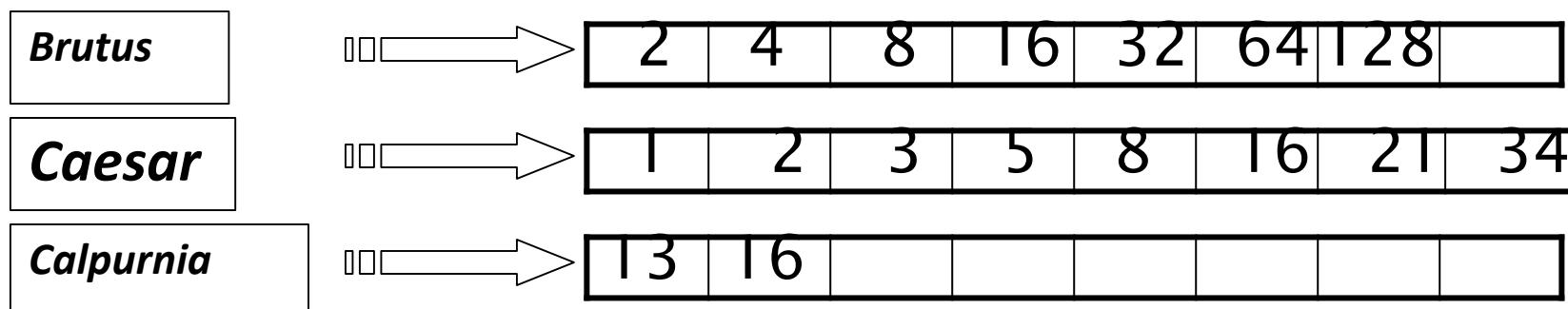


Where do we pay in storage?



Query optimization

What is the best order for query processing?
Consider a query that is an *AND* of n terms.
For each of the n terms, get its postings, then
AND them together.



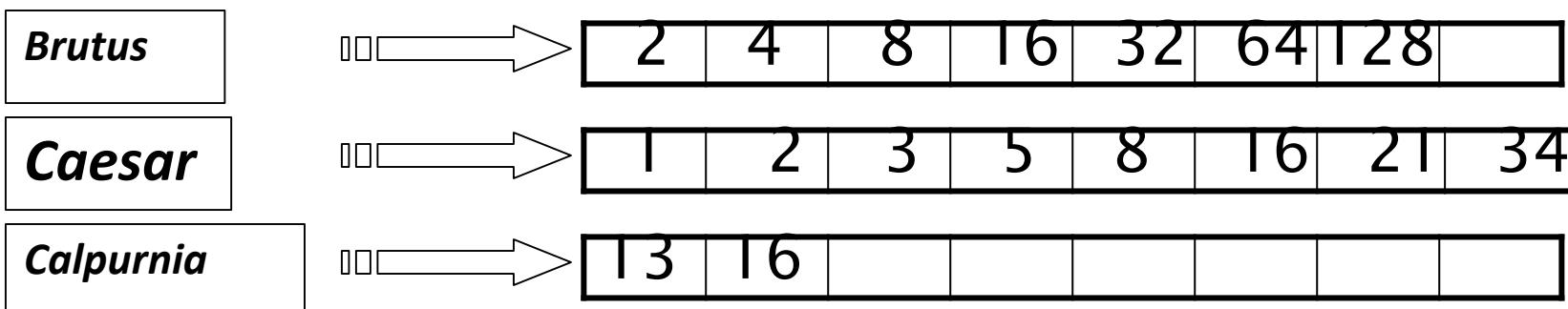
Query: *Brutus AND Calpurnia AND Caesar*

Query optimization example

Process in order of increasing freq:

start with smallest set, then keep cutting further.

This is why we kept
document freq. in dictionary



Execute the query as (**Calpurnia AND Brutus**) AND **Caesar**.

More general optimization

e.g., *(madding OR crowd) AND (ignoble OR strife)*

Get doc. freq.'s for all terms.

Estimate the size of each *OR* by the sum of its doc. freq.'s (conservative).

Process in increasing order of *OR* sizes.

Query processing with an inverted index

The index we just built

How do we process a query?



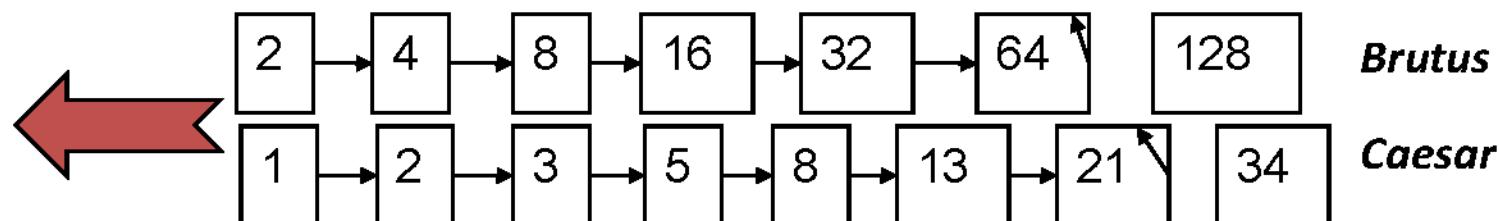
Later - what kinds of queries can we process?

Query processing: AND

Consider processing the query:

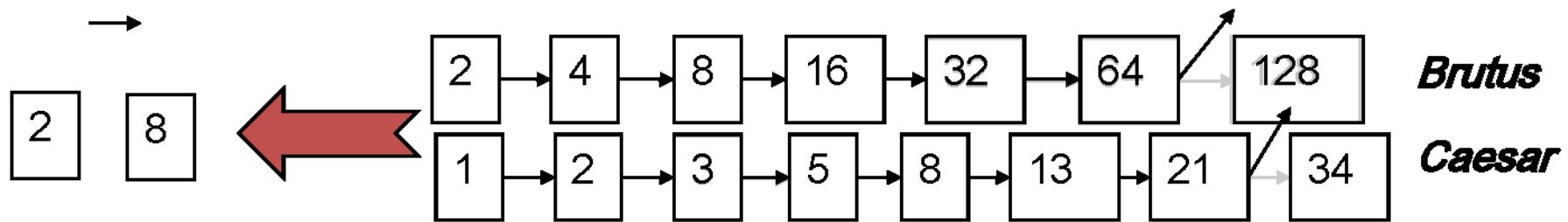
Brutus AND Caesar

- Locate ***Brutus*** in the Dictionary;
 - Retrieve its postings.
- Locate ***Caesar*** in the Dictionary;
 - Retrieve its postings.
- “Merge” the two postings:



The merge

Walk through the two postings simultaneously, in time linear in the total number of postings entries



If list lengths are x and y , merge takes $O(x+y)$ operations.
Crucial: postings sorted by docID.

Query Optimizarion

- Brutus AND Caesar AND Calipurnia

Intersecting two postings lists (a “merge” algorithm)



INTERSECT(p_1, p_2)

```
1  answer ← ⟨ ⟩  
2  while  $p_1 \neq \text{NIL}$  and  $p_2 \neq \text{NIL}$   
3  do if  $\text{docID}(p_1) = \text{docID}(p_2)$   
4      then ADD(answer,  $\text{docID}(p_1)$ )  
5           $p_1 \leftarrow \text{next}(p_1)$   
6           $p_2 \leftarrow \text{next}(p_2)$   
7      else if  $\text{docID}(p_1) < \text{docID}(p_2)$   
8          then  $p_1 \leftarrow \text{next}(p_1)$   
9          else  $p_2 \leftarrow \text{next}(p_2)$   
10 return answer
```

EXERCISE1

Consider these documents:

Doc 1 breakthrough drug for schizophrenia

Doc 2 new schizophrenia drug

Doc 3 new approach for treatment of schizophrenia

Doc 4 new hopes for schizophrenia patients

- Draw the term-document incidence matrix for this document collection.
- Draw the inverted index representation for this collection

EXERCISE 2

For the same document collection in the exercise1, consider the following queries and state the returned results for the following queries:

- a. schizophrenia AND drug
- b. for AND NOT(drug OR approach)

Boolean queries: Exact match

- The Boolean retrieval model is being able to ask a query that is a Boolean expression:
 - Boolean Queries are queries using *AND*, *OR* and *NOT* to join query terms
 - . Views each document as a set of words
 - . Is precise: document matches condition or not.
 - Perhaps the simplest model to build an IR system on
- Primary commercial retrieval tool for 3 decades.
- Many search systems you still use are Boolean:
 - Email, library catalog, Mac OS X Spotlight

The term vocabulary and postings lists

Parsing a document

- We need to deal with format and language of each document.
- What format is it in? pdf, word, excel, html etc.
- What language is it in?
- What character set is in use?
- Each of these is a classification problem, which we will study later in this course (IIR 13).
- Alternative: Use heuristics

Format/Language: Complications

- A single index usually contains terms of several languages.
- Sometimes a document or its components contain multiple languages / formats.
- French email with Spanish pdf attachment
- What is the document unit for indexing?

A file?

An email?

An email with 5 attachments?

A group of files (ppt or latex in HTML)?

Also: XML

Definitions

- Word – A delimited string of characters as it appears in the text.
- Term – A “normalized” word (case, morphology, spelling etc); an equivalence class of words.
- Token – An instance of a word or term occurring in a document.
- Type – The same as a term in most cases: an equivalence class of tokens.

Normalization

- Need to “normalize” terms in indexed text as well as query terms into the same form.
- Example: We want to match *U.S.A.* and *USA*
- We most commonly implicitly define **equivalence classes** of terms.
- Alternatively: do asymmetric expansion
window → window, windows windows
→ Windows, windows Windows (no expansion)
- More powerful, but less efficient
- Why don't you want to put *window*, *Window*, *windows*, and *Windows* in the same equivalence class?

Normalization

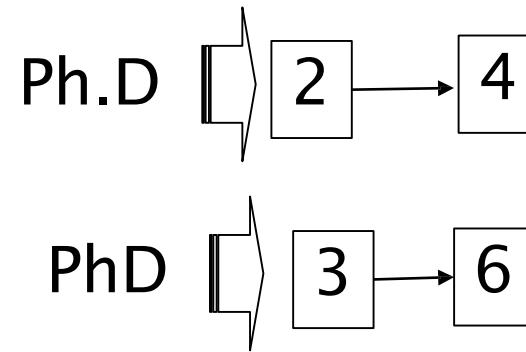
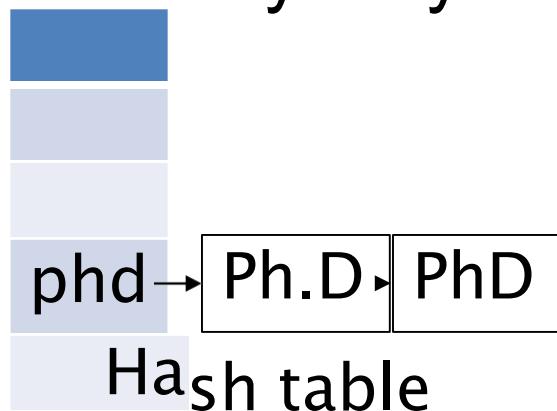
- The process of normalization is to **reduce multiple tokens to the same canonical term**, such that matches occur despite superficial differences.
- For example suppose you have variant forms of writing the same word like Ph.D, PhD which gets mapped to a single token as phd. Deleting the periods, hyphens, Accents etc..

Normalization: Other Languages

- Normalization and language detection interact.
- PETER WILL NICHT MIT. → MIT = mit
- He got his PhD from MIT. → MIT ≠ mit

Query expansion / Asymmetric expansion

- For each term, t , in a query, expand the query with synonyms and related words of t from the thesaurus.
- Powerful but less efficient in terms of space
- These hand crafted terms is called as **Thesauri**.
- Handle Synonyms and Homonyms



Exercises

In June, the dog likes to chase the cat in the barn.

- How many word tokens?
- How many word types?
- Why tokenization is difficult

– even in English.

Tokenize: Mr. O'Neill thinks that the boys' stories about Chile's capital aren't amusing.

Tokenization problems: One word or two? (or several)

- Hewlett-Packard
- State-of-the-art
- co-education
- the hold-him-back-and-drag-him-away maneuver
- data base
- San Francisco
- Los Angeles-based company
- Cheap San Francisco-Los Angeles fares
- York University vs. New York University

Numbers

- 3/20/91
- 20/3/91
- Mar 20, 1991
- B-52
- 100.2.86.144
- (800) 234-2333
- 800.234.2333
- Older IR systems may not index numbers . . .
- . . . but generally it's a useful feature.

Chinese: No whitespace

莎拉波娃现在居住在美国东南部的佛罗里达。今年4月9日，莎拉波娃在美国第一大城市纽约度过了18岁生日。生日派对上，莎拉波娃露出了甜美的微笑。

Ambiguous segmentation in Chinese



The two characters can be treated as one word meaning ‘monk’ or as a sequence of two words meaning ‘and’ and ‘still’.

Other cases of “no whitespace”

- Compounds in Dutch, German, Swedish
- ✓ Computerlinguistik → Computer + Linguistik
- ✓ Lebensversicherungsgesellschaftsangestellter
- → leben + versicherung + gesellschaft + angestellter
- Inuit: tusaatsiarunnanngittualuuujunga (I can't hear very well.)
- Many other languages with segmentation difficulties: Finnish, Urdu, . . .

Japanese

ノーベル平和賞を受賞したワンガリ・マータイさんが名誉会長を務めるMOTTA I NA I キャンペーンの一環として、毎日新聞社とマガジンハウスは「私の、もったいない」を募集します。皆様が日ごろ「もったいない」と感じて実践していることや、それによつわるエピソードを800字以内の文章にまとめ、簡単な写真、イラスト、図などを添えて10月20日までにお送りください。大賞受賞者には、50万円相当の旅行券とエコ製品2点の副賞が贈られます。

4 different “alphabets”: Chinese characters, hiragana syllabary for inflectional endings and functional words, katakana syllabary for transcription of foreign words and other uses, and latin. No spaces (as in Chinese). End user can express query entirely in hiragana!

Arabic script: Bidirectionality

استقلت الجزائر في سنة 1962 بعد 132 عاما من الاحتلال الفرنسي.

← →

← START

- ‘Algeria achieved its independence in 1962 after 132 years of French occupation.’
- Bidirectionality is not a problem if text is coded in Unicode.

Accents and diacritics

- Accents: résumé vs. resume (simple omission of accent)
- Umlauts: Universität vs. Universitaet (substitution with special letter sequence “ae”)
- Most important criterion: How are users likely to write their queries for these words?
- Even in languages that standardly have accents, users often do not type them. (Polish?)

Case folding

- Reduce all letters to lower case
- Possible exceptions: capitalized words in mid-sentence
- MIT vs. mit
- Fed vs. fed
- It's often best to lowercase everything since users will use lowercase regardless of correct capitalization.

Stop words

- stop words = extremely common words which would appear to be of little value in helping select documents matching a user need
- Examples: 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
- Stop word elimination used to be standard in older IR systems.
- But you need stop words for phrase queries, e.g. “King of Denmark”
- Most web search engines index stop words.

Lemmatization

- Reduce inflectional/variant forms to base form
- Example: *am, are, is* → *be*
- Example: *car, cars, car's, cars'* → *car*
- Example: *the boy's cars are different colors* → *the boy car be different color*
- Lemmatization implies doing “proper” reduction to dictionary headword form (the **lemma**).

Stemming

- Definition of stemming: Crude heuristic process that **chops off the ends of words** in the hope of achieving what “principled” lemmatization attempts to do with a lot of linguistic knowledge.
- Language dependent
- Example : *automate, automatic, automation* all reduce to *automat*

Porter algorithm

- Most common algorithm for stemming English
- Results suggest that it is at least as good as other stemming options
- Conventions + 5 phases of reductions
- Phases are applied sequentially
- Each phase consists of a set of commands.
- Sample command: Delete final *ement* if what remains is longer than 1 character
 - replacement → replac
 - cement → cement
- Sample convention: Of the rules in a compound command, select the one that applies to the longest suffix.

Porter Stemmer

- A consonant in a word is a letter other than A, E, I, O or U
- Any letter not a consonant is a Vowel.
- All the words in English are of the form C(VC)^mV where m is measure of any word or word part when represented in this form (VC).

Examples:

m=0 TR, EE, TREE, Y, BY.

m=1 TROUBLE, OATS, TREES

m=2 TROUBLES, PRIVATE, OATEN, ORRERY.

Porter Stemmer (contd..)

- The rules for removing a suffix will be given in the form (condition) S1 -> S2
- This means that if a word ends with the suffix S1, and the stem before S1 satisfies the given condition, S1 is replaced by S2.

(m > 1) EMENT ->

Porter Stemmer (contd..)

Step 1a

SSES -> SS

caresses -> caress

IES -> I

ponies -> poni

SS -> SS

ties -> ti

S ->

caress -> caress

cats -> cat

- Step 1 deals with plurals and past participles.

- The subsequent steps are much more straightforward.

Step 1b

(m>0) EED -> EE

feed -> feed

(*v*) ED ->

agreed -> agree

plastered -> plaster

bled -> bled

(*v*) ING ->

motoring -> motor

sing -> sing

Stemmers: A comparison

Sample text:

Such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation

Porter stemmer:

such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation

Lovins stemmer:

such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation

Paice stemmer:

such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation

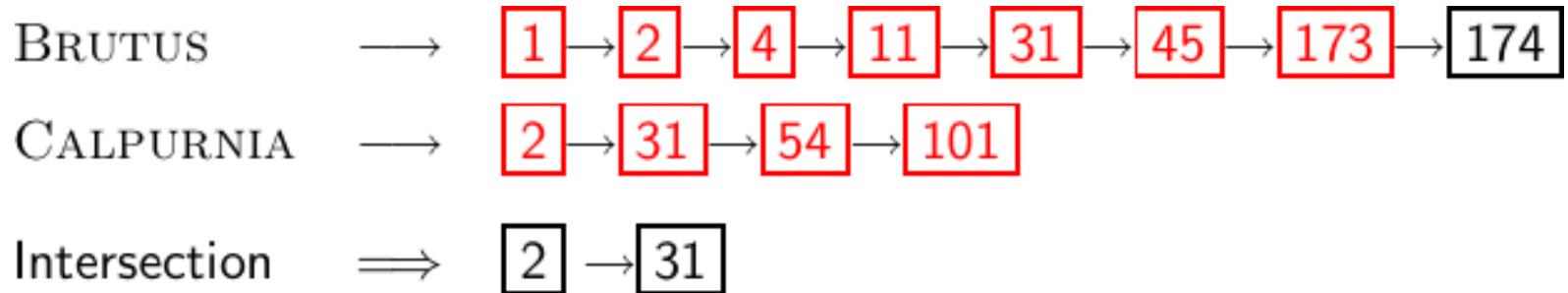
Does stemming improve effectiveness?

- In general, stemming increases effectiveness for some queries, and decreases effectiveness for others.
- Queries where stemming is likely to help: [tartan sweaters], [sightseeing tour san francisco] (equivalence classes: {sweater,sweaters}, {tour,tours})
- Porter Stemmer equivalence class oper contains all of operate operating operates operation operative operatives operational.
- Queries where stemming hurts: [operational AND research], [operating AND system], [operative AND dentistry]

Exercise: What does Google do?

- Stop words
- Normalization
- Tokenization
- Lowercasing
- Stemming
- Non-latin alphabets
- Umlauts
- Compounds
- Numbers

Recall basic intersection algorithm



- Linear in the length of the postings lists.
- Can we do better?

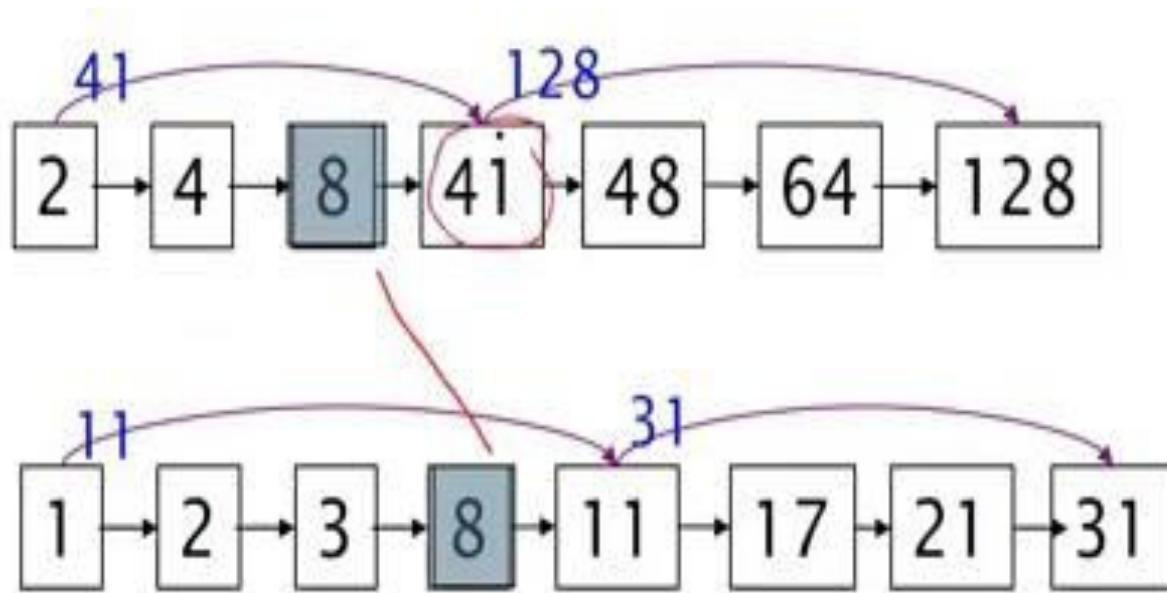
Skip pointers

- Skip pointers allow us to skip postings that will not figure in the search results.
- This makes intersecting postings lists more efficient.
- Some postings lists contain several million entries – so efficiency can be an issue even if basic intersection is linear.
- Where do we put skip pointers?

Where do we place skips?

- Simple heuristic: for postings list of length P , use \sqrt{P} evenly-spaced skip pointers.
- This ignores the distribution of query terms.
- Easy if the index is static; harder in a dynamic environment because of updates.
- They used to help a lot. With today's fast CPUs, they don't help that much anymore.

Skip lists: example





THANK YOU



BITS Pilani
Pilani Campus

Information Retrieval

Dr. Vijayalakshmi Anand
BITS-Pilani



Session 3: Tolerant Retrieval

Date – 3rd November 2022

Time – 4.15 to 6.15pm

These slides are prepared by the instructor Prof. Manning,, with grateful acknowledgement of and many others who made their course materials freely available online.

Agenda

Dictionary and Tolerant Retrieval (T1 Chapter 3)

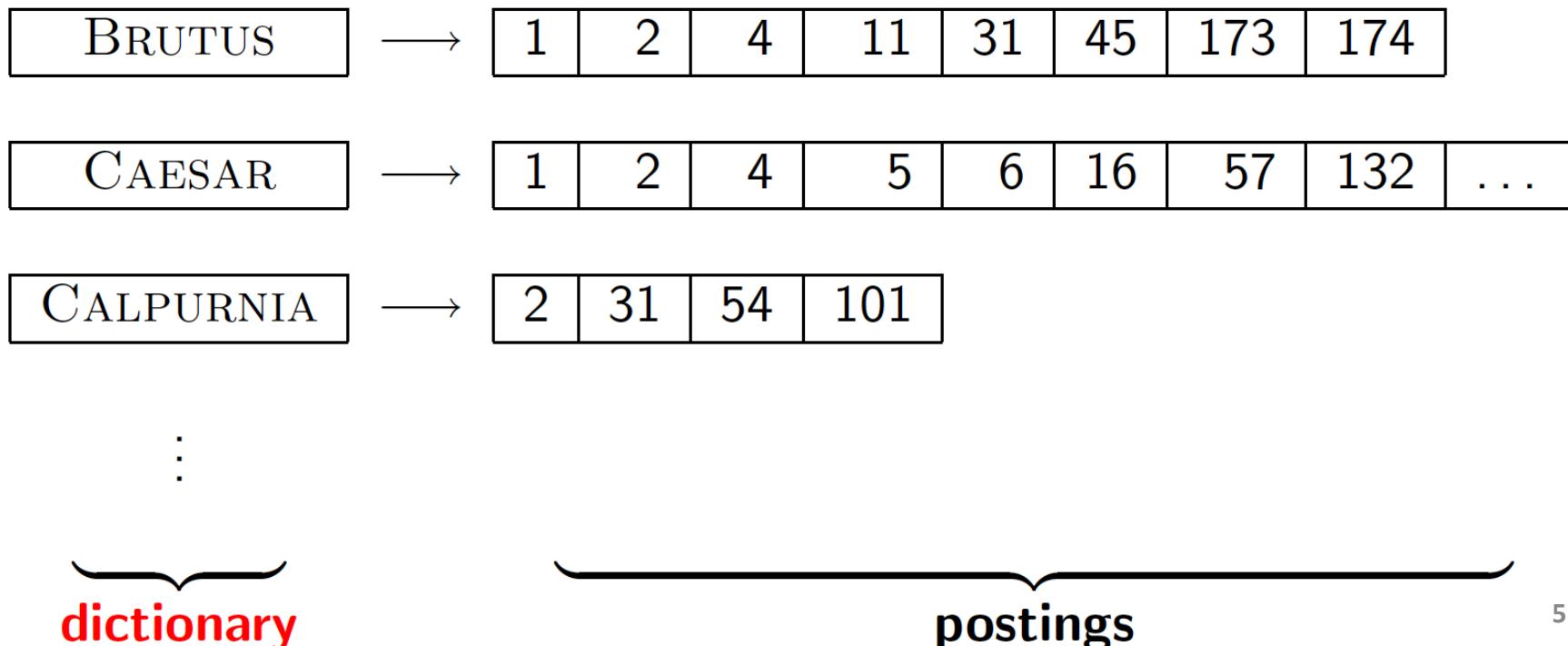
- Search Structures for dictionaries
- Wildcard queries
- Spell Check Algorithms
- Phonetic Correction
- Soundex Algorithm

Recap of the previous lecture

- The type/token distinction
 - Terms are normalized types put in the dictionary
- Tokenization problems:
 - Hyphens, apostrophes, compounds, CJK
- Term equivalence classing:
 - Numbers, case folding, stemming, lemmatization
- Skip pointers
 - Encoding a tree-like structure in a postings list
- Biword indexes for phrases
- Positional indexes for phrases/proximity queries

Dictionary data structures for inverted indexes

The dictionary data structure stores the term vocabulary, document frequency, pointers to each postings list ... in what data structure?



A naïve dictionary

An array of struct:

term	document frequency	pointer to postings list
a	656,265	→
aachen	65	→
...
zulu	221	→

How do we quickly look up elements at query time?

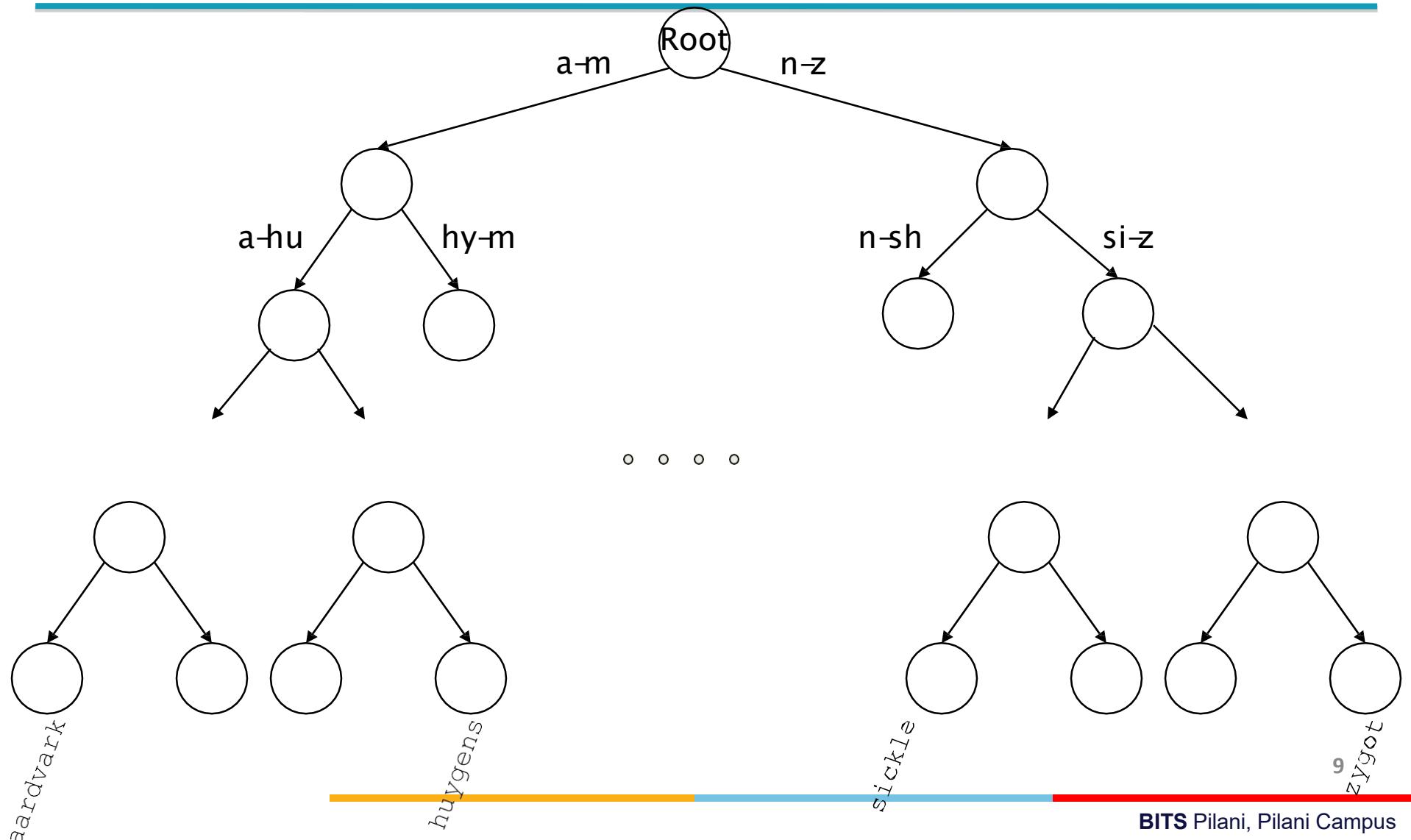
Dictionary data structures

- . Two main choices:
 - . Hashtables
 - . Trees
- . Some IR systems use hashtables, some trees

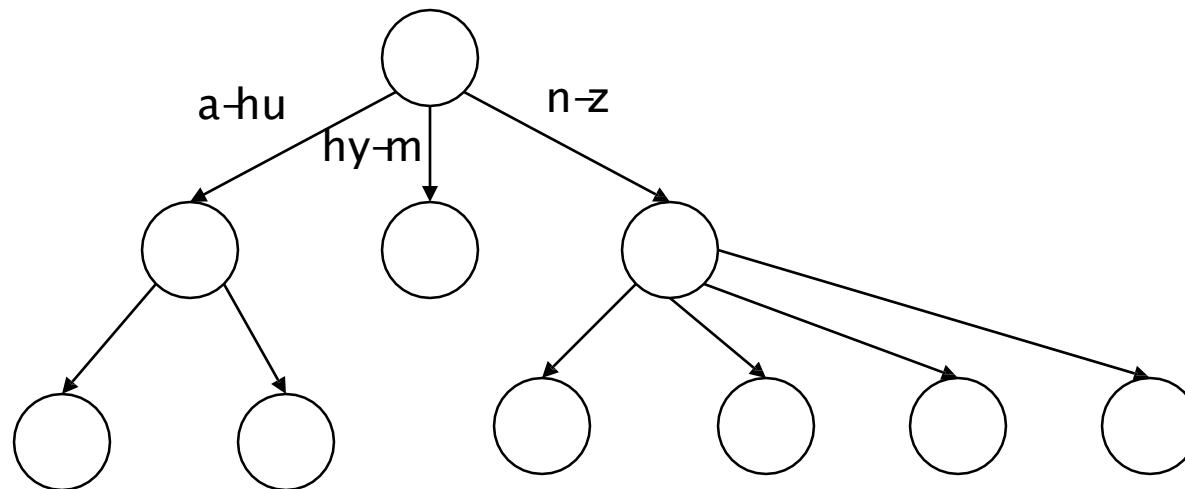
Hashtables

- Each vocabulary term is hashed to an integer
- Pros:
 - Lookup is faster than for a tree: $O(1)$
- Cons:
 - No easy way to find minor variants:
 - judgment/judgement
 - No prefix search [tolerant retrieval]
 - If vocabulary keeps growing, need to occasionally do the expensive operation of rehashing *everything*
 - *May result into collisions*

Tree: binary tree



Tree: B-tree



Definition: Every internal node has a number of children in the interval $[a,b]$ where a, b are appropriate natural numbers, e.g., $[2,4]$.

Trees

- Simplest: binary tree
- More usual: B-trees
- Trees require a standard ordering of characters and hence strings ... but we typically have one
- Pros:
 - Solves the prefix problem (terms starting with *hyp*)
- Cons:
 - Slower: $O(\log M)$ [and this requires *balanced* tree]
 - Rebalancing binary trees is expensive
 - But B-trees mitigate the rebalancing problem

“Tolerant” retrieval

- Wild-Card Queries
- Permuterm Index
- N-gram Index
- Spell Correction
- Soundex

WILD-CARD QUERIES

Wild-card queries: *

mon*: find all docs containing any word beginning with “mon”.

Easy with binary tree (or B-tree) lexicon: retrieve all words in range: ***mon ≤ w < moo***

***mon**: find words ending in “mon”: harder

- Maintain an additional B-tree for terms *backwards*.
- Can retrieve all words in range: ***nom ≤ w < non***.

Exercise: from this, how can we enumerate all terms meeting the wild-card query ***pro*cent*** ?

Query processing

- At this point, we have an enumeration of all terms in the dictionary that match the wild-card query.
- We still have to look up the postings for each enumerated term.
- E.g., consider the query:
- **se*ate AND fil*er**
- This may result in the execution of many Boolean AND queries.

B-trees handle *'s at the end of a query term

- How can we handle *'s in the middle of query term?
 - **co*tion**
- We could look up **co*** AND ***tion** in a B-tree and intersect the two term sets
 - Expensive
- The solution: transform wild-card queries so that the *'s occur at the end
 - This gives rise to the **Permuterm Index**.
We use 2 indexes
 1. Inverted index (posting list of documents)
 2. Permuterm Index (terms matching wild-card queries)OR
K-gram Index (terms matching wild-card queries)

Permuterm index

For term ***hello***, index under:

***hello\$*, *ello\$h*, *llo\$he*, *lo\$hel*, *o\$hell*,
\$hello**

where \$ is a special symbol.

Queries:

X lookup on **X\$**

X** lookup on **X\$

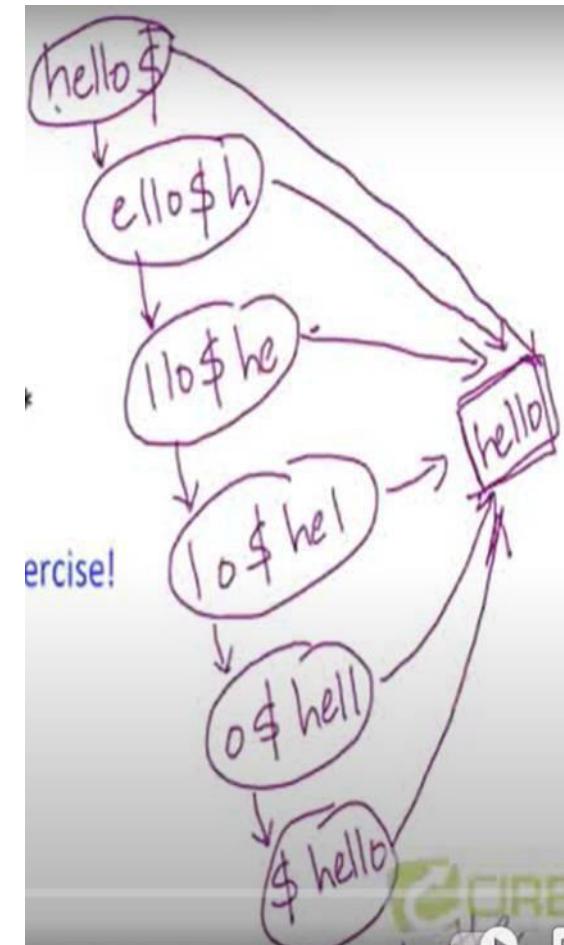
X*Y lookup on **Y\$X***

X*Y*Z ??? Exercise!

Query = ***hel*o***

X=hel, Y=o

Lookup ***o\$hel****



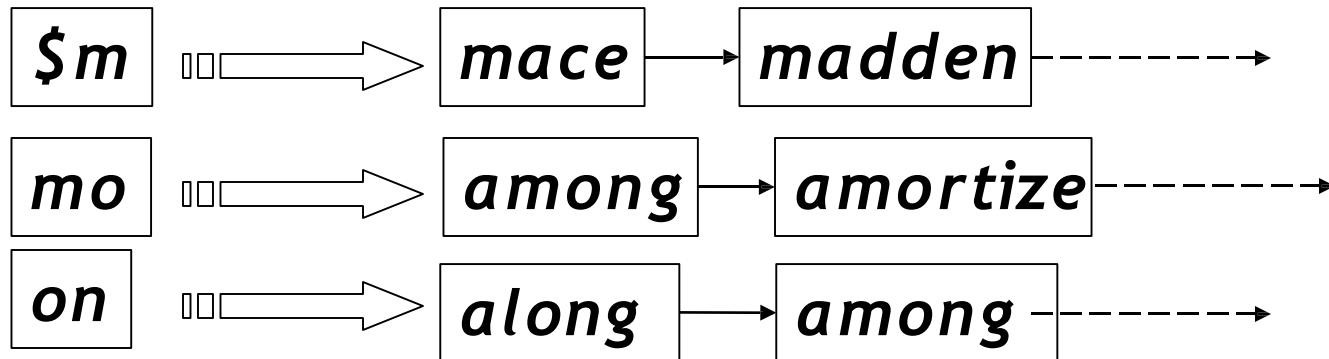
Bigram (k -gram) indexes

- Enumerate all k -grams (sequence of k chars) occurring in any term
- e.g., from text “***April is the cruelest month***” we get the 2-grams (*bigrams*)
 - \$ is a special word boundary symbol
- Maintain a second inverted index from bigrams to dictionary terms that match each bigram.

```
$a,ap,pr,ri,il,I$, $i,is,s$, $t,th,he,e$, $c,cr,ru,  
ue,el,le,es,st,t$, $m,mo,on,nt,h$
```

Bigram index example

- The k -gram index finds *terms* based on a query consisting of k -grams (here $k=2$).



Processing wild-cards

- Query ***mon**** can now be run as
 - **\$m AND mo AND on**
- Gets terms that match AND version of our wildcard query.
- But we'd enumerate ***moon***.
- Must post-filter these terms against query.
- Surviving enumerated terms are then looked up in the term-document inverted index.
- Fast, space efficient (compared to permuterm).

SPELLING CORRECTION

Spell correction

- Two principal uses
 - Correcting document(s) being indexed
 - Correcting user queries to retrieve “right” answers
- Two main flavors:
 - Isolated word
 - Check each word on its own for misspelling
 - Will not catch typos resulting in correctly spelled words
 - e.g., *from* → *form*
 - Context-sensitive
 - Look at surrounding words,
 - e.g., *I flew form Heathrow to Narita.*

<https://www.theverge.com/2020/10/15/21518034/google-search-ai-machine-learning-spelling-video-chapters-better-results>

Document correction

- Especially needed for OCR'ed documents
 - Correction algorithms are tuned for this: rn/m
 - Can use domain-specific knowledge
 - E.g., OCR can confuse O and D more often than it would confuse O and I (adjacent on the QWERTY keyboard, so more likely interchanged in typing).
- But also: web pages and even printed material have typos

Goal: the dictionary contains fewer misspellings

- But often we don't change the documents and instead fix the query-document mapping

Query mis-spellings

- Our principal focus here
 - E.g., the query ***Alanis Morisett***
- We can either
 - Retrieve documents indexed by the correct spelling,
OR
 - Return several suggested alternative queries with
the correct spelling
 - *Did you mean ... ?*

Isolated word correction

- Fundamental premise – there is a lexicon from which the correct spellings come
- Two basic choices for this
 - A standard lexicon such as
 - Webster's English Dictionary
 - An “industry-specific” lexicon – hand-maintained
 - The lexicon of the indexed corpus
 - E.g., all words on the web
 - All names, acronyms etc.
 - (Including the mis-spellings)

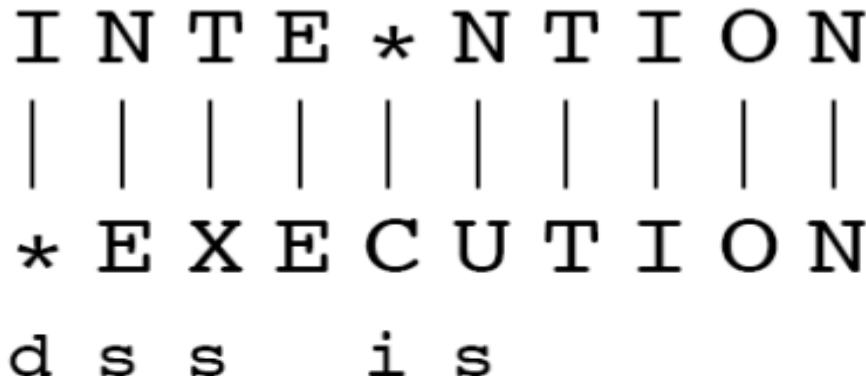
Isolated word correction

- Distance between words
- Given a lexicon and a character sequence Q, return the words in the lexicon closest to Q
- What's “closest”?
- We'll study several alternatives
 - Edit distance (Levenshtein distance)
 - Weighted edit distance
 - n -gram overlap (Jaccard Coefficient)

Edit distance

- The minimum edit distance between two strings
- Is the minimum number of editing operations
 - Insertion
 - Deletion
 - Substitution
- Needed to transform one into the other

Minimum edit distance



I	N	T	E	*	N	T	I	O	N
*	E	X	E	C	U	T	I	O	N
d	s	s		i	s				

- If each operation has cost of 1
 - Distance between these is 5
- If substitutions cost 2 (Levenshtein)
 - Distance between them is 8

Defining Min Edit Distance



- For two strings
 - X of length n
 - Y of length m
- We define $D(i,j)$
 - the edit distance between $X[1..i]$ and $Y[1..j]$
 - i.e., the first i characters of X and the first j characters of Y
 - The edit distance between X and Y is thus $D(n,m)$

Dynamic Programming for Minimum Edit Distance

- **Dynamic programming:** A tabular computation of $D(n,m)$
 - Solving problems by combining solutions to subproblems.
 - Bottom-up
 - We compute $D(i,j)$ for small i,j
 - And compute larger $D(i,j)$ based on previously computed smaller values
 - i.e., compute $D(i,j)$ for all i ($0 < i < n$) and j ($0 < j < m$)
-

- Initialization

$$D(i, 0) = i$$

$$D(0, j) = j$$

- Recurrence Relation:

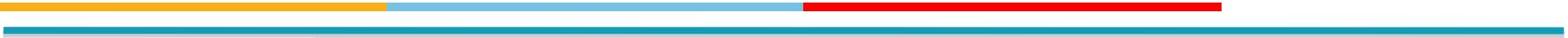
For each $i = 1 \dots M$

For each $j = 1 \dots N$

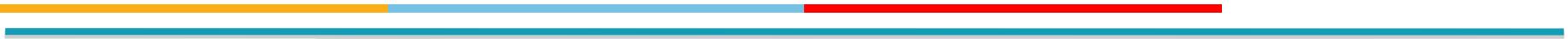
$$D(i, j) = \min \begin{cases} D(i-1, j) + 1 \\ D(i, j-1) + 1 \\ D(i-1, j-1) + 2; & \text{if } X(i) \neq Y(j) \\ 0; & \text{if } X(i) = Y(j) \end{cases}$$

- Termination:

$D(N, M)$ is distance



-	0	E	X	E	C	U	T	I	O	N
0	0	1	2	3	4	5	6	7	8	9
I	1	1	2	3	4	5	6	6	7	8
N	2	2	2	3	4	5	6	7	7	7
T	3	3	3	3	4	5	5	6	7	8
E	4	3	4	3	4	5	6	6	7	8
N	5	4	4	4	4	5	6	7	7	7
T	6	5	5	5	5	5	5	6	7	8
I	7	6	6	6	6	6	6	5	6	7
O	8	7	7	7	7	7	7	6	5	6
N	9	8	8	8	8	8	8	7	6	5



Weighted edit distance

- As above, but the weight of an operation depends on the character(s) involved
 - Meant to capture OCR or keyboard errors
Example: **m** more likely to be mis-typed as **n** than as **q**
 - Therefore, replacing **m** by **n** is a smaller edit distance than by **q**
 - This may be formulated as a probability model
- Requires weight matrix as input
- Modify dynamic programming to handle weights

Using edit distances

- Given query, first enumerate all character sequences within a preset (weighted) edit distance (e.g., 2)
- Intersect this set with list of “correct” words
- Show terms you found to user as suggestions
- Alternatively,
 - We can look up all possible corrections in our inverted index and return all docs ... slow
 - We can run with a single most likely correction
- The alternatives disempower the user, but save a round of interaction with the user

Edit distance to all dictionary terms?

- Given a (mis-spelled) query – do we compute its edit distance to every dictionary term?
 - Expensive and slow
 - Alternative?
- How do we cut the set of candidate dictionary terms?
- One possibility is to use n -gram overlap for this
- This can also be used by itself for spelling correction.

n-gram overlap

-
- Enumerate all the *n*-grams in the query string as well as in the lexicon
 - Use the *n*-gram index (recall wild-card search) to retrieve all lexicon terms matching any of the query *n*-grams
 - Threshold by number of matching *n*-grams
 - Variants – weight by keyboard layout, etc.

Example with trigrams

- Suppose the text is *november*
 - Trigrams are *nov*, *ove*, *vem*, *emb*, *mbe*, *ber*.
- The query is *december*
 - Trigrams are *dec*, *ece*, *cem*, *emb*, *mbe*, *ber*.
- So 3 trigrams overlap (of 6 in each term)
- How can we turn this into a normalized measure of overlap?

One option – Jaccard coefficient

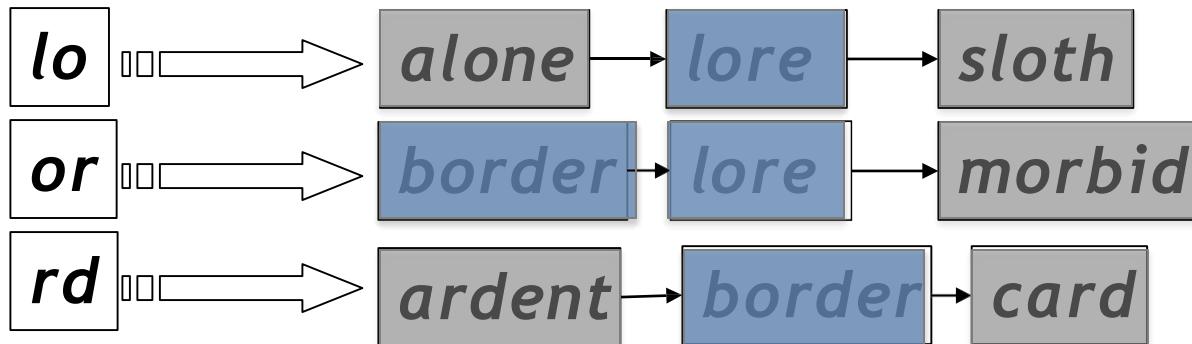
- A commonly-used measure of overlap
- Let X and Y be two sets; then the J.C. is

$$|X \cap Y| / |X \cup Y|$$

- Equals 1 when X and Y have the same elements and zero when they are disjoint
- X and Y don't have to be of the same size
- Always assigns a number between 0 and 1
 - Now threshold to decide if you have a match
 - E.g., if J.C. > 0.8, declare a match

Matching trigrams

- Consider the query ***lord*** – we wish to identify words matching 2 of its 3 bigrams (***lo, or, rd***)



Standard postings “merge” will enumerate ...

Adapt this to using Jaccard (or another) measure.

Context-sensitive spell correction

- Text: *I flew from Heathrow to Narita.*
- Consider the phrase query “*flew form Heathrow*”
- We’d like to respond
 - Did you mean “*flew from Heathrow*”?
- because no docs matched the query phrase.

Context-sensitive correction

- Need surrounding context to catch this.
- First idea: retrieve dictionary terms close (in weighted edit distance) to each query term
- Now try all possible resulting phrases with one word “fixed” at a time
 - ***flew from heathrow***
 - ***fled form heathrow***
 - ***flea form heathrow***
- **Hit-based spelling correction:** Suggest the alternative that has lots of hits.
 - Query-log frequency
(more relevant since we look at query terms)
 - Corpus frequency

Exercise

- Suppose that for “**flew form Heathrow**” we have 7 alternatives for flew, 19 for form and 3 for heathrow.
- How many “corrected” phrases will we enumerate in this scheme?

General issues in spell correction

- We enumerate multiple alternatives for “Did you mean?”
- Need to figure out which to present to the user
 - The alternative hitting most docs
 - Query log analysis

SOUNDEX

Soundex

- Class of heuristics to expand a query into **phonetic equivalents**
 - Language specific – mainly for names
 - E.g., **chebyshev** → **tchebycheff**
 - Invented for the U.S. census ... in 1918

Soundex – algorithm

- Turn every token to be indexed into a 4-character reduced form
- Do the same with query terms
- Build and search an index on the reduced forms
 - (when the query calls for a soundex match)

<http://www.creativyst.com/Doc/Articles/SoundEx1/SoundEx1.htm#Top>

Soundex – algorithm

Retain the first letter of the word.

Change all occurrences of the following letters to '0' (zero):

'A', 'E', 'I', 'O', 'U', 'H', 'W', 'Y'.

Change letters to digits as follows:

B, F, P, V → 1

C, G, J, K, Q, S, X, Z → 2

D, T → 3

L → 4

M, N → 5

R → 6

Example ***Herman***

Soundex continued

4. Remove all pairs of consecutive digits.
5. Remove all zeros from the resulting string.
6. Pad the resulting string with trailing zeros and return the first four positions, which will be of the form <uppercase letter> <digit> <digit> <digit>.

E.g., **Herman** becomes H655. (horman/hermon/Herman/Hermen)

Will *hermann* generate the same code?

Soundex

- Soundex is the classic algorithm, provided by most databases (Oracle, Microsoft, ...)
- How useful is soundex?

What queries can we process?

We have

- Positional inverted index
- Permuterm Index
- K-gram Index
- Wild-card index
- Spell-correction
- Soundex

Queries such as

- **(SPELL(moriset) /3 toron*to) OR
SOUNDEX(chaiikofski)**

Exercise

- Draw yourself a diagram showing the various indexes in a search engine incorporating all the functionality we have talked about
- Identify some of the key design choices in the index pipeline:
 - Does stemming happen before the Soundex index?
 - What about n -grams?
- Given a query, how would you parse and dispatch sub-queries to the various indexes?

Resources

IIR 3, MG 4.2

Efficient spell retrieval:

K. Kukich. Techniques for automatically correcting words in text.
ACM Computing Surveys 24(4), Dec 1992.

J. Zobel and P. Dart. Finding approximate matches in large
lexicons. Software - practice and experience 25(3), March
1995. <http://citeseer.ist.psu.edu/zobel95finding.html>

Mikael Tillenius: Efficient Generation and Ranking of Spelling Error
Corrections. Master's thesis at Sweden's Royal Institute of
Technology. <http://citeseer.ist.psu.edu/179155.html>

Nice, easy reading on spell correction:

Peter Norvig: How to write a spelling corrector
<http://norvig.com/spell-correct.html>



THANK YOU



BITS Pilani
Pilani Campus

Information Retrieval

Dr.Vijayalakshmi Anand
BITS Pilani



Session 4: Index Construction

Date – 17th dec 2022

Time – 4.15 pm to 6.15 pm

These slides are prepared by the instructor Prof. Manning,, with grateful acknowledgement of and many others who made their course materials freely available online.

Agenda

Index Construction and Compression (T1 Chapter 4, 5)

- Blocked sort-based Indexing
- Single pass in-memory indexing
- Distributed and dynamic indexing
- Dictionary comparison
- Postings file compression

Index construction

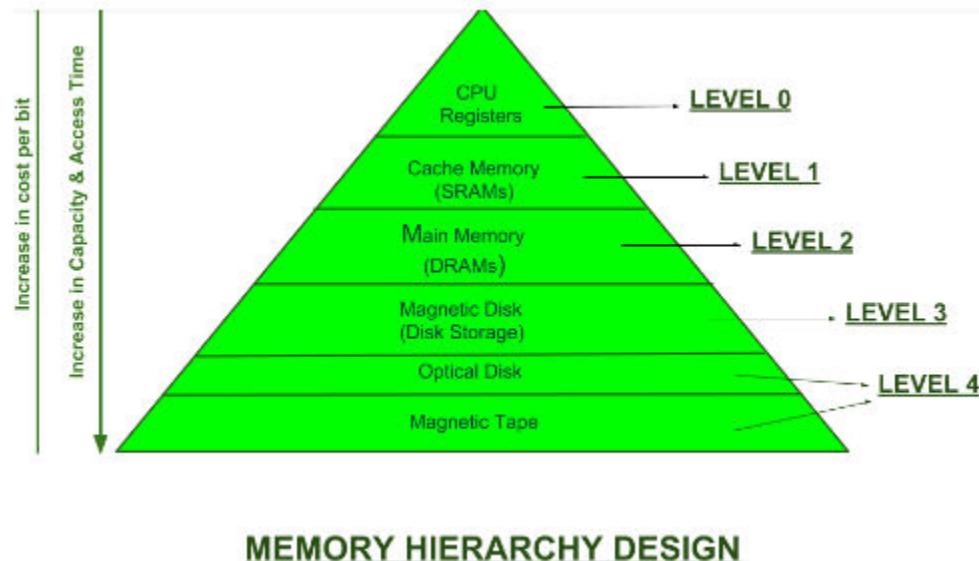
- How do we construct an index?
- What strategies can we use with limited main memory?

Hardware basics

- . Many design decisions in information retrieval are based on the characteristics of hardware
- . We begin by reviewing hardware basics

Hardware basics

- Access to data in memory is much faster than access to data on disk.



Hardware basics

- Disk seeks: No data is transferred from disk while the disk head is being positioned.
- Therefore: Transferring one large chunk of data from disk to memory is faster than transferring many small chunks.
- Disk I/O is block-based: Reading and writing of entire blocks (as opposed to smaller chunks).
- Block sizes: 8KB to 256 KB.

Hardware basics

- Servers used in IR systems now typically have several GB of main memory, sometimes tens of GB.
- Available disk space is several (2–3) orders of magnitude larger.
- Fault tolerance is very expensive: It's much cheaper to use many regular machines rather than one fault tolerant machine.

Hardware assumptions for this lecture

symbol	statistic	value
s	average seek time	$5 \text{ ms} = 5 \times 10^{-3} \text{ s}$
b	transfer time per byte	$0.02 \mu\text{s} = 2 \times 10^{-8} \text{ s}$
	processor's clock rate(disk seeks)	10^9 s^{-1}
p	low-level operation (e.g., compare & swap a word)	$0.01 \mu\text{s} = 10^{-8} \text{ s}$
	size of main memory	several GB
	size of disk space	1 TB or more

RCV1: collection

- Shakespeare's collected works definitely aren't large enough for demonstrating many of the points in this course.
- The collection we'll use isn't really large enough either, but it's publicly available and is at least a more plausible example.
- As an example for applying scalable index construction algorithms, we will use the Reuters RCV1 collection.
- This is one year of Reuters newswire (part of 1995 and 1996)

A Reuters RCV1 document



You are here: Home > News > Science > Article

Go to a Section: U.S. International Business Markets Politics Entertainment Technology Sports Oddly Enough

Extreme conditions create rare Antarctic clouds

Tue Aug 1, 2006 3:20am ET

[Email This Article](#) | [Print This Article](#) | [Reprints](#)



SYDNEY (Reuters) - Rare, mother-of-pearl colored clouds caused by extreme weather conditions above Antarctica are a possible indication of global warming, Australian scientists said on Tuesday.

[+] Text [-]

Known as nacreous clouds, the spectacular formations showing delicate wisps of colors were photographed in the sky over an Australian meteorological base at Mawson Station on July 25.

Reuters RCV1 statistics

symbol	statistic	value
N	documents	800,000
L	avg. # tokens per doc	200
M	terms (= word types)	400,000
	avg. # bytes per token (incl. spaces/punct.)	6
	avg. # bytes per token (without spaces/punct.)	4.5
	avg. # bytes per term	7.5
	non-positional postings	100,000,000

4.5 bytes per word token vs. 7.5 bytes per word type: why?

Recall Index construction



Documents are parsed to extract words and these are saved with the Document ID.

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



Key step

After all documents have been parsed, the inverted file is sorted by terms.



Term	Doc #	Term	Doc #
I	1	ambitious	2
did	1	be	2
enact	1	brutus	1
julius	1	brutus	2
caesar	1	capitol	1
I	1	caesar	1
was	1	caesar	2
killed	1	caesar	2
i'	1	did	1
the	1	enact	1
capitol	1	hath	1
brutus	1	I	1
killed	1	I	1
me	1	i'	1
so	2	it	2
let	2	julius	1
it	2	killed	1
be	2	killed	1
with	2	let	2
caesar	2	me	1
the	2	noble	2
noble	2	so	2
brutus	2	the	1
hath	2	the	2
told	2	told	2
you	2	you	2
caesar	2	was	2
was	2	was	1
ambitious	2	with	2

Scaling index construction

- . In-memory index construction does not scale
 - . Can't stuff entire collection into memory, sort, then write back
- . How can we construct an index for very large collections?
- . Taking into account the hardware constraints we just learned about . . .
- . Memory, disk, speed, etc.

Sort-based Index Construction

- As we build the index, we parse docs one at a time.
 - While building the index, we cannot easily exploit compression tricks
- The final postings for any term are incomplete until the end.
- At 8 bytes per non-positional postings entry (*term, doc, freq*), demands a lot of space for large collections.
- For example of no. of terms $T = 100,000,000$
- So ... we can do this in memory, but typical collections are much larger.
E.g., the *New York Times* provides an index of >150 years of newswire

Thus: We need to store intermediate results on disk.

Sort using disk as “memory”?

- Can we use the same index construction algorithm for larger collections, but by using disk instead of memory?

Sort using disk as “memory”?

- Can we use the same index construction algorithm for larger collections, but by using disk instead of memory?
- No: Sorting $T = 100,000,000$ records on disk is too slow
 - – too many disk seeks.
- We need an *external* sorting algorithm.
 - External to main memory, data stored on disk

BSBI: Blocked sort-based Indexing (Sorting with fewer disk seeks)

12-byte (4+4+4) records (*term, doc, freq*).
These are generated as we parse docs.
Must now sort 100M such 12-byte records by *term*.

Define a Block $\sim 10M$ such records

Can easily fit a couple into memory.

Will have 10 such blocks to start with.

Basic idea of algorithm:

Accumulate postings for each block, sort, write to disk.

Then merge the blocks into one long sorted order.



postings to be merged

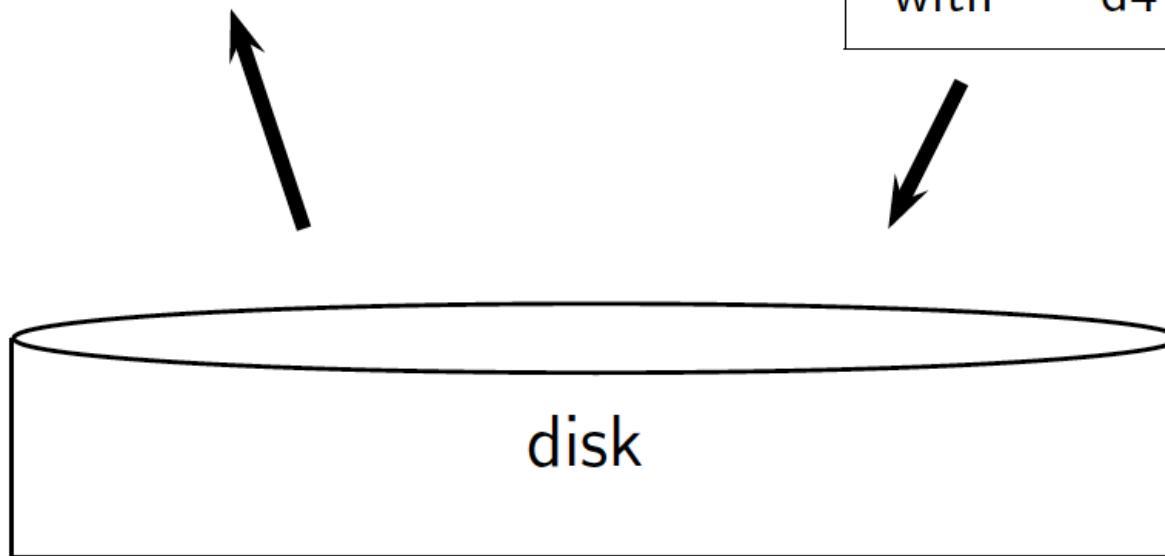
brutus	d3
caesar	d4
noble	d3
with	d4

brutus	d2
caesar	d1
julius	d1
killed	d2



brutus	d2
brutus	d3
caesar	d1
caesar	d4
julius	d1
killed	d2
noble	d3
with	d4

merged
postings



Sorting 10 blocks of 10M records



- First, read each block and sort within:
 - Quicksort takes $2N \ln N$ expected steps
 - In our case $2 \times (10M \ln 10M)$ steps
- 10 times this estimate – gives us 10 sorted runs of 10M records each.
- Done straightforwardly, need 2 copies of data on disk
 - But can optimize this

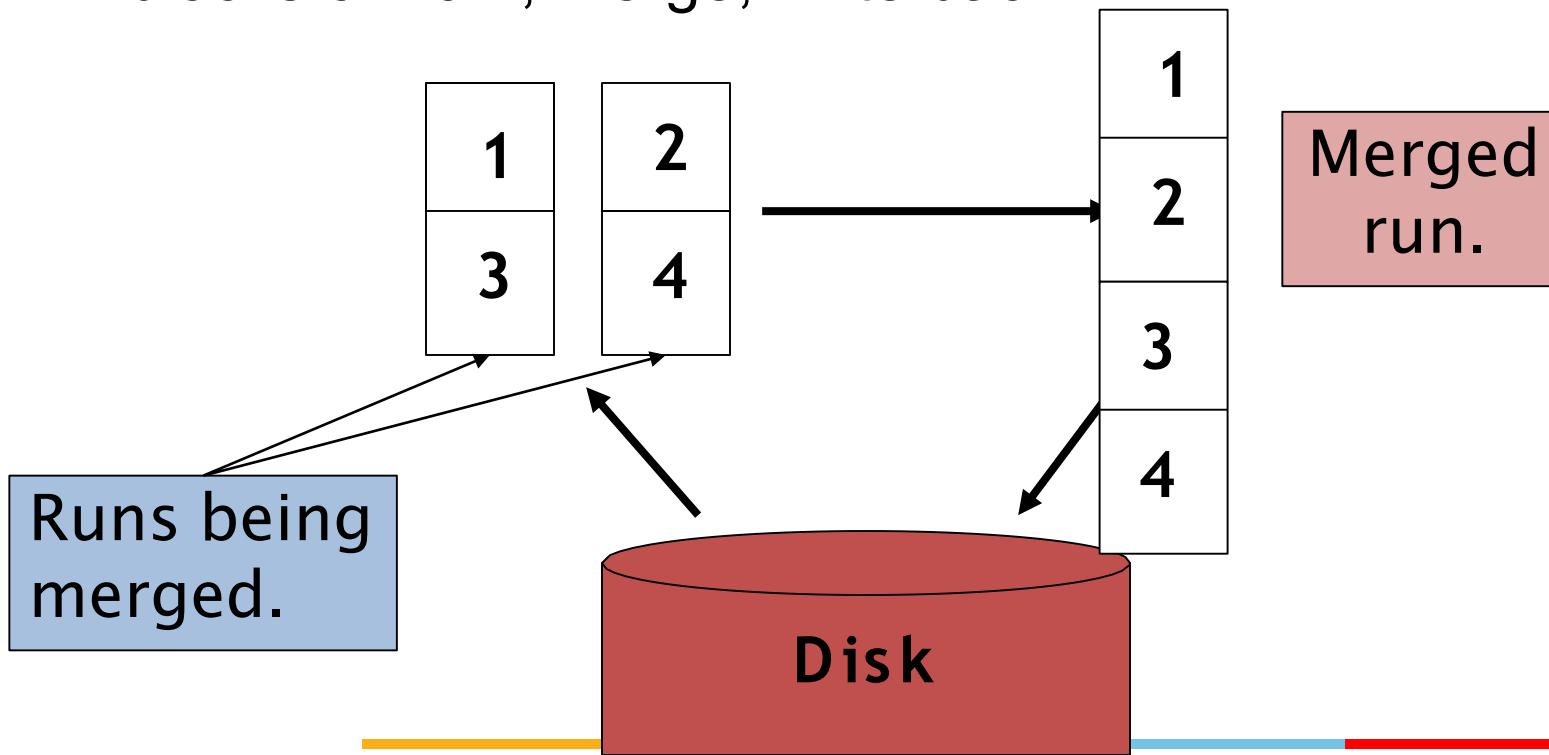
BSBINDEXCONSTRUCTION()

```
1   $n \leftarrow 0$ 
2  while (all documents have not been processed)
3  do  $n \leftarrow n + 1$ 
4       $block \leftarrow \text{PARSENEXTBLOCK}()$ 
5      BSBI-INVERT( $block$ )
6      WRITEBLOCKTODISK( $block, f_n$ )
7      MERGEBLOCKS( $f_1, \dots, f_n; f_{\text{merged}}$ )
```

How to merge the sorted runs?

Can do binary merges, with a merge tree of $\log_2 10 = 4$ layers.

During each layer, read into memory runs in blocks of 10M, merge, write back.



How to merge the sorted runs?

But it is more efficient to do a multi-way merge, where you are reading from all blocks simultaneously

Providing you read decent-sized chunks of each block into memory and then write out a decent-sized output chunk, then you're not killed by disk seeks

Remaining problem with sort-based algorithm

Our assumption was: we can keep the dictionary in memory.

We need the dictionary (which grows dynamically) in order to implement a term to termID mapping.

Actually, we could work with term,docID postings instead of termID,docID postings . . .

. . . but then intermediate files become very large.

SPIMI:

Single-pass in-memory indexing

Key idea 1: Generate separate dictionaries for each block – no need to maintain term-termID mapping across blocks.

Key idea 2: Don't sort. Accumulate postings in postings lists as they occur.

With these two ideas we can generate a complete inverted index for each block.

These separate indexes can then be merged into one big index.

SPIMI-Invert

```
SPIMI-INVERT(token_stream)
```

```
1  output_file = NEWFILE()
2  dictionary = NEWHASH()
3  while (free memory available)
4  do token  $\leftarrow$  next(token_stream)
5    if term(token)  $\notin$  dictionary
6      then postings_list = ADDToDICTIONARY(dictionary, term(token))
7      else postings_list = GETPOSTINGSLIST(dictionary, term(token))
8      if full(postings_list)
9        then postings_list = DOUBLEPOSTINGSLIST(dictionary, term(token))
10       ADDTOPOSTINGSLIST(postings_list, docID(token))
11   sorted_terms  $\leftarrow$  SORTTERMS(dictionary)
12   WRITEBLOCKTODISK(sorted_terms, dictionary, output_file)
13 return output_file
```

Merging of blocks is analogous to BSBI.

SPIMI: Compression

Compression makes SPIMI even more efficient.

Compression of terms

Compression of postings

We will discuss about the same later in this course...

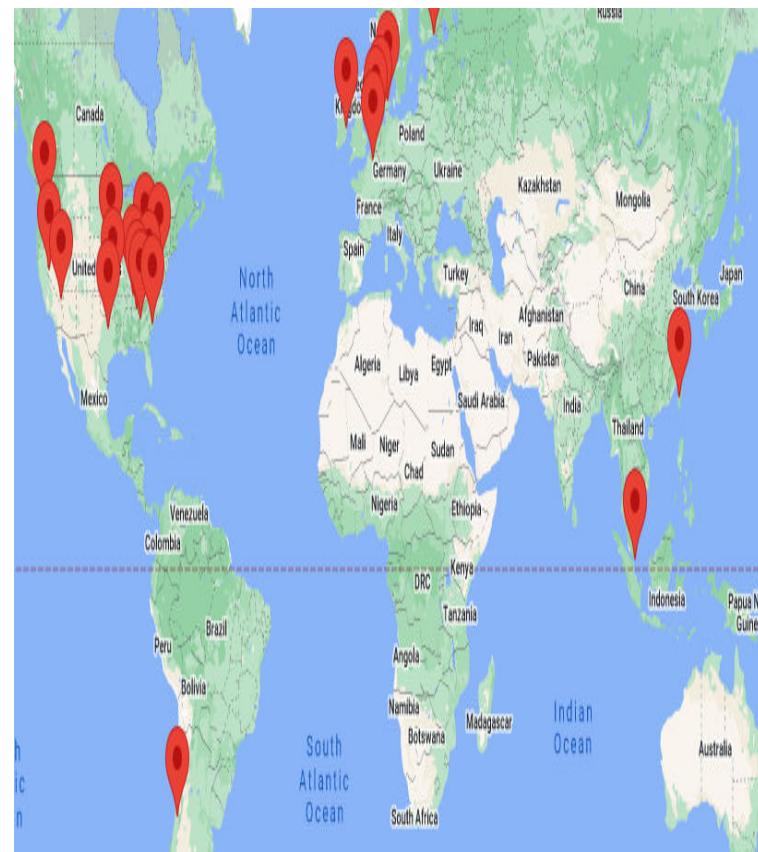
Distributed indexing

- For web-scale indexing:
- must use a distributed computing cluster
 - Individual machines are fault-prone
 - Can unpredictably slow down or fail
- How do we exploit such a pool of machines?

Web search engine data centers



- Web search data centers (Google, Bing, Baidu) mainly contain commodity machines.
- Data centers are distributed around the world.
- Estimate: Google ~1 million servers, 3 million processors/cores (Gartner 2007)



Distributed indexing

- Maintain a *master* machine directing the indexing job – considered “safe”.
- Break up indexing into sets of (parallel) tasks.
- Master machine assigns each task to an idle machine from a pool.

Parallel tasks

- We will use two sets of parallel tasks
- Parsers
- Inverters
- Break the input document collection into *splits*
- Each split is a subset of documents (corresponding to blocks in BSBI/SPIMI)

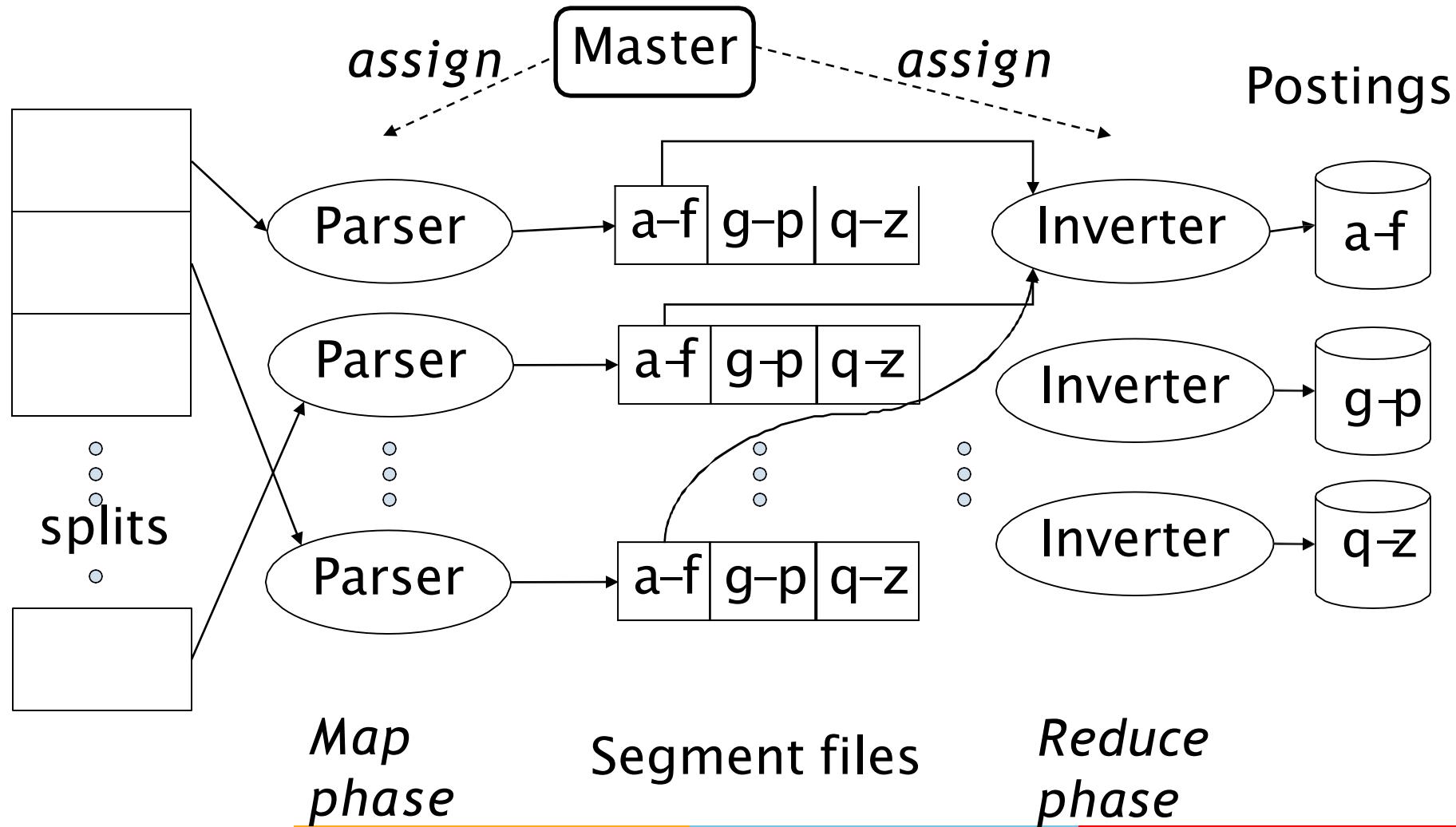
Parsers

- Master assigns a split to an idle parser machine
- Parser reads a document at a time and emits (term, doc)pairs
- Parser writes pairs into j partitions
- Each partition is for a range of terms' first letters (e.g., **a-f**, **g-p**, **q-z**) – here $j = 3$.
- Now to complete the index inversion

Inverters

- An inverter collects all (term,doc) pairs (= postings) for one term-partition.
- Sorts and writes to postings lists

Data flow



MapReduce

- The index construction algorithm we just described is an instance of *MapReduce*.
- MapReduce (Dean and Ghemawat 2004) is a robust and conceptually simple framework for distributed computing ...
- ... without having to write code for the distribution part.
- They describe the Google indexing system (ca. 2002) as consisting of a number of phases, each implemented in MapReduce.

MapReduce

- Index construction was just one phase.
- Another phase: transforming a term-partitioned index into a document-partitioned index.
 - *Term-partitioned*: one machine handles a subrange of terms
 - *Document-partitioned*: one machine handles a subrange of documents
- As we'll discuss in the web part of the course, most search engines use a document-partitioned index ... better load balancing, etc.

Schema for index construction in MapReduce

Schema of map and reduce functions

map: input → list(k, v) reduce: (k, list(v)) → output

Instantiation of the schema for index construction

map: collection → list(termID, docID)

reduce: (<termID1, list(docID)>, <termID2, list(docID)>, ...) → (postings list1, postings list2, ...)

Example for index construction

Map:

d1 : C came, C c'ed.

d2 : C died. →

<C,d1>, <came,d1>, <C,d1>, <c'ed, d1>, <C, d2>,
<died,d2>

Reduce:

(<C,(d1,d2,d1)>, <died,(d2)>, <came,(d1)>, <c'ed,(d1)>)
→ (<C,(d1:2,d2:1)>, <died,(d2:1)>, <came,(d1:1)>,
<c'ed,(d1:1)>)

Dynamic indexing

- Up to now, we have assumed that collections are static.
- They rarely are:
 - Documents come in over time and need to be inserted.
 - Documents are deleted and modified.
- This means that the dictionary and postings lists have to be modified:
 - Postings updates for terms already in dictionary
 - New terms added to dictionary

Simplest approach

- Maintain “big” main index
- New docs go into “small” auxiliary index
- Search across both, merge results
 - Invalidation bit-vector for deleted docs
Filter docs output on a search result by this invalidation bit-vector
- Periodically, re-index into one main index

Issues with main and auxiliary indexes

Problem of frequent merges – you touch stuff a lot

Poor performance during merge

Actually:

Merging of the auxiliary index into the main index is efficient if we keep a separate file for each postings list.

Merge is the same as a simple append.

But then we would need a lot of files – inefficient for OS.

Assumption for the rest of the lecture: The index is one big file.

In reality: Use a scheme somewhere in between (e.g., split very large postings lists, collect postings lists of length 1 in one file etc.)

Logarithmic merge

Maintain a series of indexes, each twice as large as the previous one

At any time, some of these powers of 2 are instantiated

Keep smallest (Z_0) in memory

Larger ones (I_0, I_1, \dots) on disk

If Z_0 gets too big ($> n$), write to disk as I_0

or merge with I_0 (if I_0 already exists) as Z_1

Either write merge Z_1 to disk as I_1 (if no I_1)

Or merge with I_1 to form Z_2



LMERGEADDTOKEN(*indexes*, Z_0 , *token*)

```
1   $Z_0 \leftarrow \text{MERGE}(Z_0, \{\text{token}\})$ 
2  if  $|Z_0| = n$ 
3    then for  $i \leftarrow 0$  to  $\infty$ 
4      do if  $I_i \in \text{indexes}$ 
5        then  $Z_{i+1} \leftarrow \text{MERGE}(I_i, Z_i)$ 
6          ( $Z_{i+1}$  is a temporary index on disk.)
7           $\text{indexes} \leftarrow \text{indexes} - \{I_i\}$ 
8        else  $I_i \leftarrow Z_i$  ( $Z_i$  becomes the permanent index  $I_i$ .)
9           $\text{indexes} \leftarrow \text{indexes} \cup \{I_i\}$ 
10         BREAK
11          $Z_0 \leftarrow \emptyset$ 
```

LOGARITHMICMERGE()

```
1   $Z_0 \leftarrow \emptyset$  ( $Z_0$  is the in-memory index.)
2   $\text{indexes} \leftarrow \emptyset$ 
3  while true
4  do LMERGEADDTOKEN(indexes,  $Z_0$ , GETNEXTTOKEN())
```

Logarithmic merge

- Auxiliary and main index: index construction time is $O(T^2)$ as each posting is touched in each merge.
- Logarithmic merge: Each posting is merged $O(\log T)$ times, so complexity is $O(T \log T)$
- So logarithmic merge is much more efficient for index construction
- But query processing now requires the merging of $O(\log T)$ indexes
 - Whereas it is $O(1)$ if you just have a main and auxiliary index

Further issues with multiple indexes



- Collection-wide statistics are hard to maintain
 - E.g., when we spoke of spell-correction: which of several corrected alternatives do we present to the user?
 - We said, pick the one with the most hits
 - How do we maintain the top ones with multiple indexes and invalidation bit vectors?
 - One possibility: ignore everything but the main index for such ordering
 - Will see more such statistics used in results ranking
-

Dynamic indexing at search engines

- All the large search engines now do dynamic indexing
- Their indices have frequent incremental changes
 - News items, blogs, new topical web pages
- But (sometimes/typically) they also periodically reconstruct the index from scratch
 - Query processing is then switched to the new index, and the old index is deleted

Get Search News Recaps!

Email:

Daily Monthly

Subscribe

 Feeds and more info



Google
Land

YAHOO!
Land

Microsoft
Land

Columns
Land

Marketing
Land

Searching
Land

Ask, AOL &
More Lands

Newsletters
& Feeds 

Confe
& Wel

« [Local Store And Inventory Data Poised To Transform "Online Shopping"](#) | [Main](#) | [SEO Company, Fathom Online, Acquired By Geary Interactive](#) »

Mar 31, 2008 at 8:45am Eastern by Barry Schwartz

Google Dance Is Back? Plus Google's First Live Chat Recap & Hyperactive Yahoo Slurp

Is the Google Dance back? Well, not really, but I [am noticing](#) Google Dance-like behavior from Google based on reading some of the feedback at a [WebmasterWorld](#) thread.

The Google Dance refers to how years ago, a change to Google's ranking algorithm often began showing up slowly across data centers as they reflected different results, a sign of coming changes. These days Google's data centers are typically always showing small changes and differences, but the differences between [this data center](#) and [this one](#) seem to be more like the extremes of the past Google Dances.

So either Google is preparing for a massive update or just messing around with our heads. As of now, these results have not yet moved over to the main Google.com results.

Search:

netklix

Click here for
\$40 Free
Advertising

ONWARD
search ▾

the leading
provider of search
marketing jobs

seomoz
PREMIUM MEMBERSHIP

Resources for today's lecture

Chapter 4 of IIR

MG Chapter 5

Original publication on MapReduce: Dean and Ghemawat (2004)

Original publication on SPIMI: Heinz and Zobel (2003)

41-30 mins <https://www.youtube.com/watch?v=dH0pEySEoQo>

42-13 mins [WDM 42: Index Construction Using InMemory Sorting - YouTube](#)

43- 43 mins

44-12 mins(repeated part of 43) no need to watch

45-10 mins(repeated part of 43) no need to watch

46-25 mins (skip 8 mins) https://www.youtube.com/watch?v=WHS9f_wbU-E

47-QA of 46 not required [WDM 47: Q & A on Distributed Index – YouTube](#)

48 – 24 mins [WDM 48: Map Reduce – YouTube](#)

49 – 11 mins [WDM 49: Dynamic indexing using naive approach - YouTube](#)

50- 17 mins [WDM 50: Dynamic indexing using logarithmic merge – YouTube](#)

51- 3 minutes [WDM 51: Issues With Multiple Indexes - YouTube](#)



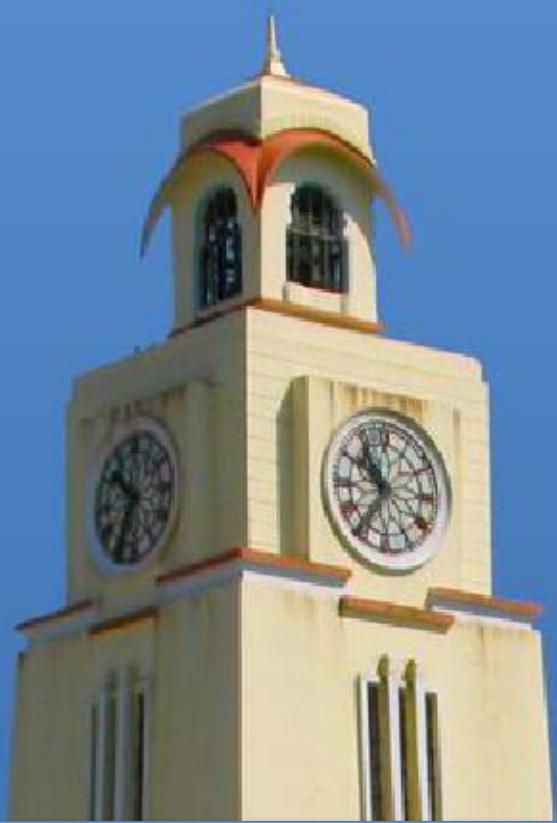
THANK YOU



BITS Pilani
Pilani Campus

Information Retrieval

Dr. Vijayalakshmi Anand,
BITS Pilani



Session 5: Index Compression

Date – 24th December 2022

Time – 4.15 pm to 6.15 pm

These slides are prepared by the instructor Prof. Manning,, with grateful acknowledgement of and many others who made their course materials freely available online.

Last lecture – Index construction

- Sort-based indexing
 - Naïve in-memory inversion
 - Blocked Sort-Based Indexing
 - Merge sort is effective for disk-based sorting
(avoid seeks!)
- Single-Pass In-Memory Indexing
 - No global dictionary
 - Generate separate dictionary for each block
 - Don't sort postings
 - Accumulate postings in postings lists as they occur
- Distributed indexing using MapReduce
- Dynamic indexing

Agenda

Index Compression (T1 Chapter 5)

- Why Index Compression?
- Collection statistics
 - How big will the dictionary and postings be?
- Dictionary Compression
- Posting list compression

Why compression (in general)?

BRUTUS →

1	2	4	11	31	45	173	174
---	---	---	----	----	----	-----	-----

CAESAR →

1	2	4	5	6	16	57	132	...
---	---	---	---	---	----	----	-----	-----

CALPURNIA →

2	31	54	101
---	----	----	-----

- Use less disk space
 - Saves a little money
- Keep more stuff in memory
 - Increases speed
- Increase speed of data transfer from disk to memory
 - [read compressed data | decompress] is faster than [read uncompressed data]
 - Premise: Decompression algorithms are fast
 - True of the decompression algorithms we use

Why compression for inverted indexes?

- Dictionary
 - Make it small enough to keep in main memory
 - Make it so small that you can keep some postings lists in main memory too
- Postings file(s)
 - Reduce disk space needed
 - Decrease time needed to read postings lists from disk
 - Large search engines keep a significant part of the postings in memory.
 - Compression lets you keep more in memory
- We will devise various IR-specific compression schemes

Reuters RCV1 statistics

symbol	statistic	value
N	documents	800,000
L	avg. # tokens per doc	200
M	terms (= word types)	400,000
	avg. # bytes per token (incl. spaces/punct.)	6
	avg. # bytes per token (without spaces/punct.)	4.5
	avg. # bytes per term	7.5
	non-positional postings	100,000,000

Index parameters vs. what we index (details IIR Table 5.1, p.80)

size of	word types (terms)			non-positional postings			positional postings		
	dictionary			non-positional index			positional index		
	Size (K)	Δ%	cumul %	Size (K)	Δ %	cumul %	Size (K)	Δ %	cumul %
Unfiltered	484			109,971			197,879		
No numbers	474	-2	-2	100,680	-8	-8	179,158	-9	-9
Case folding	392	-17	-19	96,969	-3	-12	179,158	0	-9
30 stopwords	391	-0	-19	83,390	-14	-24	121,858	-31	-38
150 stopwords	391	-0	-19	67,002	-30	-39	94,517	-47	-52
stemming	322	-17	-33	63,812	-4	-42	94,517	0	-52

Lossless vs. lossy compression

- Lossless compression: All information is preserved.
 - What we mostly do in IR.
- Lossy compression: Discard some information
- Several of the preprocessing steps can be viewed as lossy compression: case folding, stop words, stemming, number elimination.

Vocabulary vs. collection size

- How big is the term vocabulary?
 - That is, how many distinct words are there?
- Can we assume an upper bound?
 - Not really: At least $70^{20} = 10^{37}$ different words of length 20
- In practice, the vocabulary will keep growing with the collection size

DICTIONARY COMPRESSION

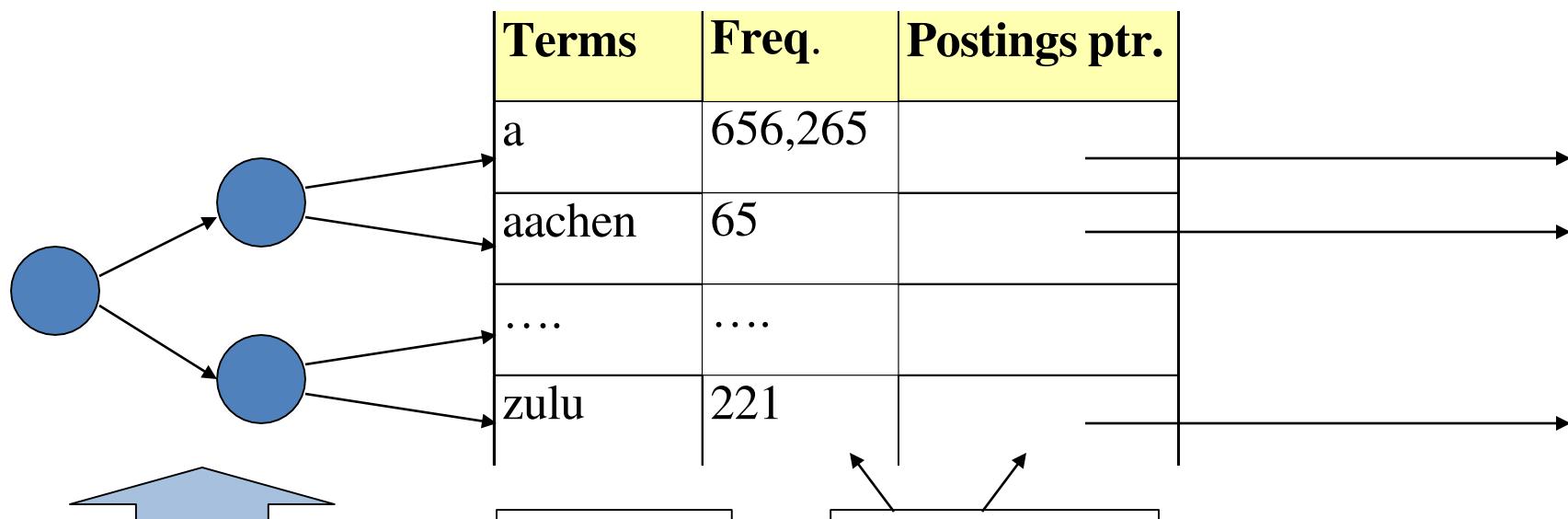
Why compress the dictionary?

- Search begins with the dictionary
- We want to keep it in memory
- Memory footprint competition with other applications
- Embedded/mobile devices may have very little memory
- Even if the dictionary isn't in memory, we want it to be small for a fast search startup time
- So, compressing the dictionary is important

Dictionary storage - first cut

Array of fixed-width entries

~400,000 terms; 28 bytes/term = 11.2 MB.



Dictionary search
structure

20 bytes

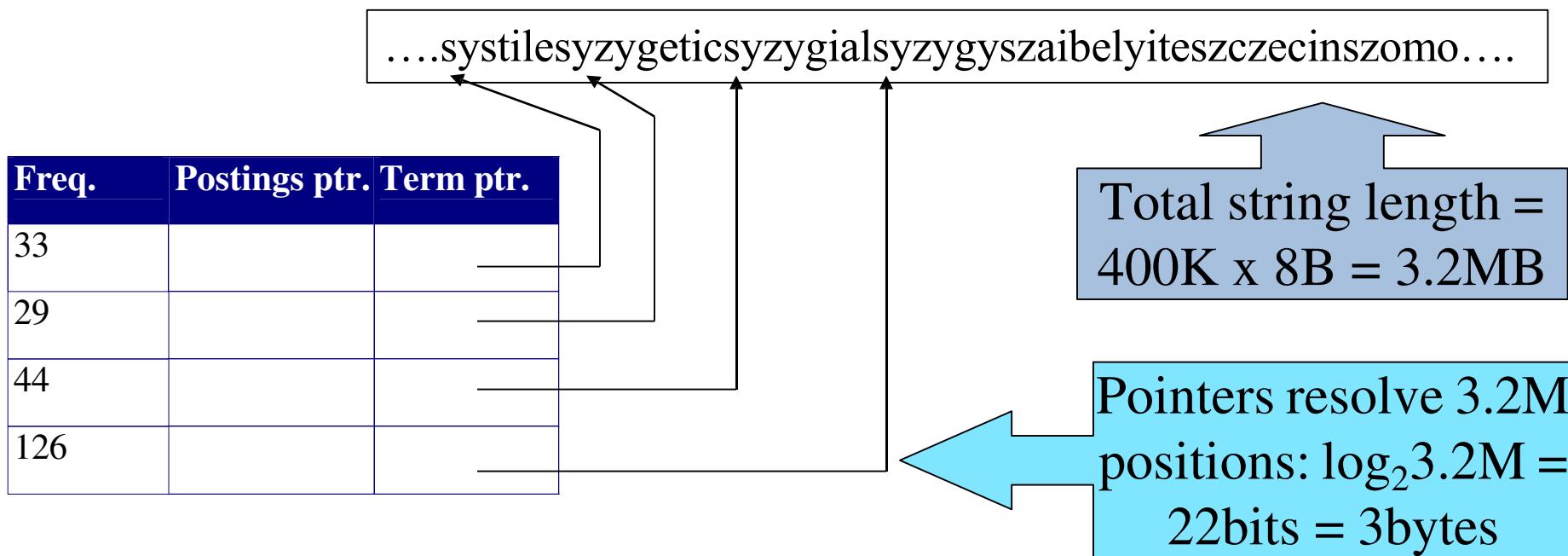
4 bytes each

Fixed-width terms are wasteful

- Most of the bytes in the **Term** column are wasted – we allot 20 bytes for 1 letter terms.
 - And we still can't handle *supercalifragilisticexpialidocious* or *hydrochlorofluorocarbons*.
- Written English averages ~4.5 characters/word.
 - Exercise: Why is/isn't this the number to use for estimating the dictionary size?
- Ave. dictionary word in English: ~8 characters
 - How do we use ~8 characters per dictionary term?
- Short words dominate token counts but not type average.

Compressing the term list: Dictionary-as-a-String

- Store dictionary as a (long) string of characters:
- Pointer to next word shows end of current word
- Hope to save up to 60% of dictionary space.



Space for dictionary as a string

4 bytes per term for Freq.

4 bytes per term for pointer to Postings.

3 bytes per term pointer

Avg. 8 bytes per term in term string

400K terms x 19 \Rightarrow 7.6 MB (against 11.2MB for fixed width)

Blocking

Store pointers to every k th term string.

Example below: $k=4$.

Need to store term lengths (1 extra byte)

....7systile9syzygetic8syzygial6syzygy11szaibelyite8szczecin9szomo....

Freq.	Postings ptr.	Term ptr.
33		—
29		
44		
126		
7		—

} Save 9 bytes
on 3
pointers.

Lose 4 bytes on
term lengths.

Net

Example for block size $k = 4$

Where we used 3 bytes/pointer without blocking

$3 \times 4 = 12$ bytes,
now we use $3 + 4 = 7$ bytes.

Shaved another ~ 0.5 MB. This reduces the size of the dictionary from 7.6 MB to 7.1 MB.

We can save more with larger k .

Why not go with larger k ?

Front coding

Front-coding:

Sorted words commonly have long common prefix –
store differences only
(for last $k-1$ in a block of k)

8automata8automate9automatic10automation
→ **8automat*^a1^e2ⁱc3^{ion}**

Encodes *automat*

Extra length
beyond *automat*.

Begins to resemble general string compression.

RCV1 dictionary compression summary

Technique	Size in MB
Fixed width	11.2
Dictionary-as-String with pointers to every term	7.6
Also, blocking $k = 4$	7.1
Also, Blocking + front coding	5.9

POSTINGS COMPRESSION

Postings compression

- The postings file is much larger than the dictionary, factor of at least 10.
- Key desideratum: store each posting compactly.
- A posting for our purposes is a docID.
- For Reuters (800,000 documents), we would use 32 bits per docID when using 4-byte integers.
- Alternatively, we can use $\log_2 800,000 \approx 20$ bits per docID.
- Our goal: use far fewer than 20 bits per docID.

Postings: two conflicting forces

- . A term like ***arachnocentric*** occurs in maybe one doc out of a million – we would like to store this posting using $\log_2 1M \sim 20$ bits.
- . A term like ***the*** occurs in virtually every doc, so 20 bits/posting is too expensive.
- . Prefer 0/1 bitmap vector in this case

Postings file entry

- We store the list of docs containing a term in increasing order of docID.

computer 33,47,154,159,202 ...

33,14,107,5,43 ...

Consequence: it suffices to store *gaps*
(Delta-Encoding / Run-length encoding)

Hope: most gaps can be encoded/stored with far fewer than 20 bits.

Three postings entries

	encoding	postings list						
THE	docIDs	...	283042	283043	283044	283045	...	
	gaps			1	1	1	...	
COMPUTER	docIDs	...	283047	283154	283159	283202	...	
	gaps			107	5	43	...	
ARACHNOCENTRIC	docIDs	252000	500100					
	gaps	252000	248100					

Variable length encoding

Aim:

For ***arachnocentric***, we will use ~20 bits/gap entry.

For ***the***, we will use ~1 bit/gap entry.

If the average gap for a term is G , we want to use $\sim \log_2 G$ bits/gap entry.

Key challenge: encode every integer (gap) with about as few bits as needed for that integer.

This requires a ***variable length encoding***

Variable length codes achieve this by using short codes for small numbers

Variable Byte (VB) codes

- For a gap value G , we want to use close to the fewest bytes needed to hold $\log_2 G$ bits
- Begin with one byte to store G and dedicate 1 bit in it to be a continuation bit c
- If $G \leq 127$, binary-encode it in the 7 available bits and set $c = 1$
- Else encode G 's lower-order 7 bits and then use additional bytes to encode the higher order bits using the same algorithm
- At the end set the continuation bit of the last byte to 1 ($c = 1$) – and for the other bytes $c = 0$.

Example

docIDs	824	829	215406
gaps	824	5	214577
VB code	00000110 10111000	10000101	00001101 00001100 10110001

Postings stored as the byte concatenation

000001101011100010000101000011010000110010110001



Key property: VB-encoded postings are uniquely prefix-decodable.

Index compression summary

- We can now create an index for highly efficient Boolean retrieval that is very space efficient
- 10-15% of the total size of the text in the collection
- However, we've ignored positional information
- Hence, space savings are less for indexes used in practice
 - But techniques substantially the same.

References

- T1 Chapter 5
- F. Scholer, H.E. Williams and J. Zobel. 2002. Compression of Inverted Indexes For Fast Query Evaluation. *Proc. ACM-SIGIR 2002*.
 - Variable byte codes
- V. N. Anh and A. Moffat. 2005. Inverted Index Compression Using Word-Aligned Binary Codes. *Information Retrieval* 8: 151–166.
 - Word aligned codes
 - WDM 56: Various Posting Compression Techniques (first 28 minutes)



BITS Pilani
Pilani Campus

Information Retrieval

Dr. Vijayalakshmi Anand
BITS Pilani



Session 6: Text Classification

Date – 29th Dec 2022

Time – 4.15 pm to 6.15 pm

These slides are prepared by the instructor Prof. Manning., with grateful acknowledgement of and many others who made their course materials freely available online.

What we covered in last session

Vector Space Model (T1 Chapter 6)

- Ranked Retrieval
- Scoring documents
- Term frequency
- Collection statistics
- Weighting schemes
- Vector space scoring

What we will cover in this session

Text Classification (T1 Chp 13 and 14)

- Text classification problem
- Naïve Bayes
- Feature Selection
- Vector space classification
- Document Representation
- Rocchio classification
- Evaluating Classification

Standing queries

- The path from IR to text classification:
 - You have an information need to monitor, say:
 - [Covid vaccination in India](#)
 - You want to rerun an appropriate query periodically to find new news items on this topic
 - You will be sent new documents that are found
 - I.e., it's not ranking but classification (relevant vs. not relevant)
- Such queries are called **standing queries**
 - Long used by “information professionals”
 - A modern mass instantiation is **Google Alerts**
- Standing queries are (hand-written) text classifiers

Formal definition of TC: Application/Testing

Given: a description $d \in X$ of a document
Determine: $\gamma(d) \in C$, that is, the class that is most appropriate for d

Examples of how search engines use classification

- Language identification (classes: English vs. French etc.)
- The automatic detection of spam pages (spam vs. nonspam)
- Topic-specific or *vertical* search – restrict search to a “vertical” like “related to health” (relevant to vertical vs. not)
- Sentiment detection: is a movie or product review positive or negative (positive vs. negative)

Categorization/Classification

Given:

A representation of a document d

Issue: how to represent text documents.

Usually some type of high-dimensional space – bag of words

A fixed set of classes:

$$C = \{c_1, c_2, \dots, c_J\}$$

Determine:

The category of d : $\gamma(d) \in C$, where $\gamma(d)$ is a classification function

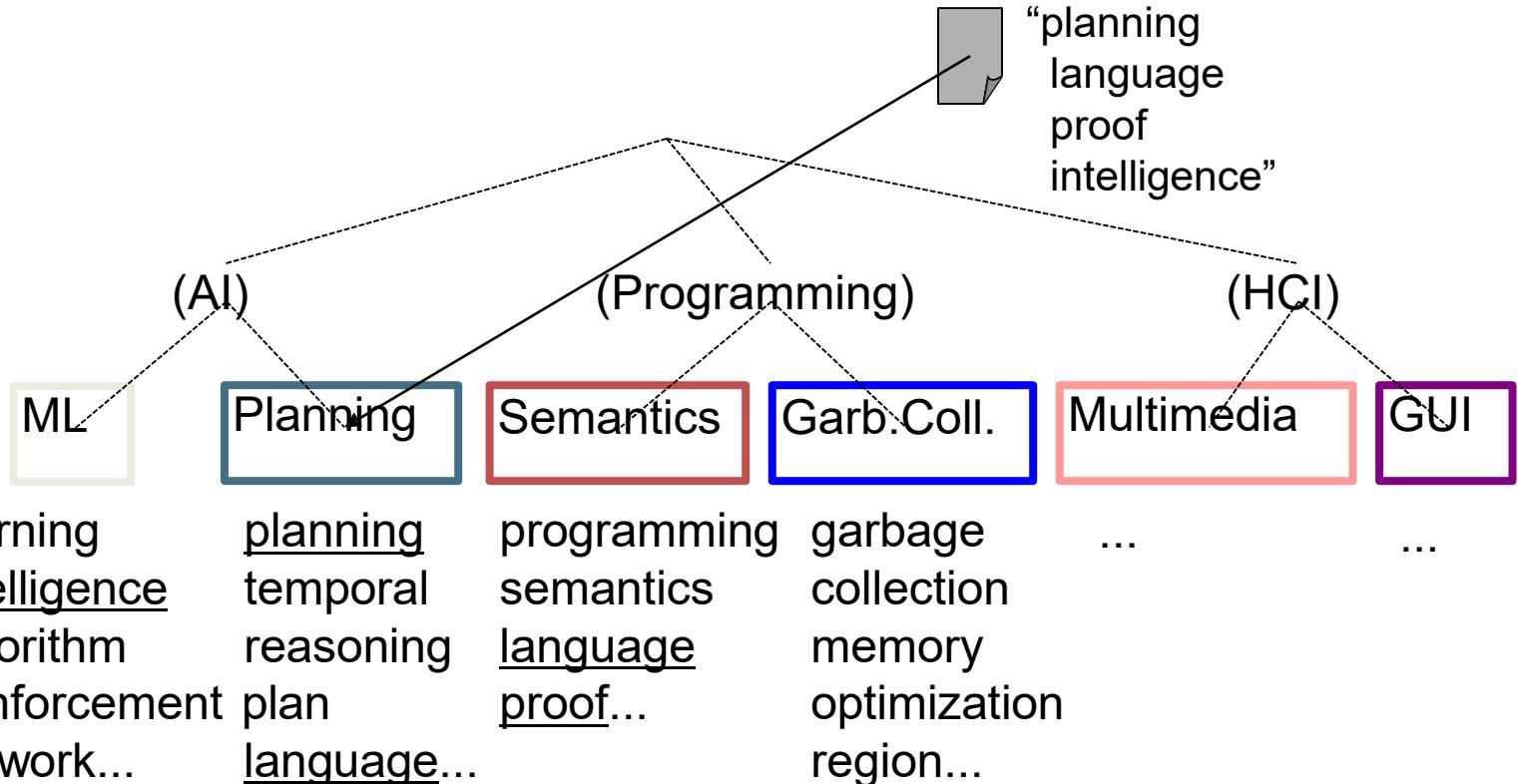
We want to build classification functions (“classifiers”).

Document Classification

**Test
Data:**

Classes:

**Training
Data:**



Classification Methods

- Manual classification
 - Used by the original Yahoo! Directory
 - Looksmart, about.com, ODP, PubMed
 - Accurate when job is done by experts
 - Consistent when the problem size and team is small
 - Difficult and expensive to scale
 - Means we need automatic classification methods for big problems

Classification Methods

- Hand-coded rule-based classifiers
 - One technique used by new agencies, intelligence agencies, etc.
 - Widely deployed in government and enterprise
 - Vendors provide “IDE” for writing such rules

Classification Methods

- . Hand-coded rule-based classifiers
 - . Commercial systems have complex query languages
 - . Accuracy can be high if a rule has been carefully refined over time by a subject expert
 - . Building and maintaining these rules is expensive

Classification Methods: Supervised learning

- Given:
 - A document d
 - A fixed set of classes:
 - $C = \{c_1, c_2, \dots, c_J\}$
 - A training set D of documents each with a label in C
- Determine:
 - A learning method or algorithm which will enable us to learn a classifier γ
 - For a test document d , we assign it the class $\gamma(d) \in C$

Classification Methods

- Supervised learning
 - Naive Bayes (simple, common)
 - k-Nearest Neighbors (simple, powerful)
 - Support-vector machines (new, generally more powerful)
 - ... plus many other methods
 - Requires hand-classified training data
 - But data can be built up (and refined) by amateurs
- Many commercial systems use a mixture of methods

The bag of words representation

Y

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet.

)=C

The bag of words representation

great	2
love	2
recommend	1
laugh	1
happy	1
...	...

Y

)=C

Features

- Supervised learning classifiers can use any sort of feature
 - URL, email address, punctuation, capitalization, dictionaries, network features
- In the bag of words view of documents
 - We use **only** word features
 - we use **all** of the words in the text (not a subset)

Feature Selection: Why?

- Text collections have a large number of features
 - 10,000 – 1,000,000 unique words ... and more
- Selection may make a particular classifier feasible
 - Some classifiers can't deal with 1,000,000 features
- Reduces training time
 - Training time for some methods is quadratic or worse in the number of features
- Makes runtime models smaller and faster
- Can improve generalization (performance)
 - Eliminates noise features
 - Avoids overfitting

Feature Selection: Frequency

- The simplest feature selection method:
 - Just use the commonest terms
 - No particular foundation
 - But it make sense why this works
 - They're the words that can be well-estimated and are most often available as evidence
 - In practice, this is often 90% as good as better methods

Bayesian Learning

- Bayesian learning algorithms that calculate explicit probabilities for hypotheses, such as the naive Bayes classifier, are among the most practical approaches to certain types of learning problems
- For example: Problem of learning to classify text documents such as electronic news articles.
- For such learning tasks, the naive Bayes classifier is among the most effective algorithms known

Independence

- Two events A and B are independent if and only if $P(A \cap B) = P(A)P(B)$.
- if two events A and B are independent then $P(A|B) = P(A)$

$$\begin{aligned} P(A|B) &= \frac{P(A \cap B)}{P(B)} \\ &= \frac{P(A)P(B)}{P(B)} \\ &= P(A). \end{aligned}$$

- In general, for n events A_1, A_2, \dots, A_n to be independent we must have

$$P(A_i \cap A_j) = P(A_i)P(A_j), \text{ for all distinct } i, j \in \{1, 2, \dots, n\};$$

$$P(A_i \cap A_j \cap A_k) = P(A_i)P(A_j)P(A_k), \text{ for all distinct } i, j, k \in \{1, 2, \dots, n\};$$

Conditional Probability

- Conditional Probability is a measure of the probability of an event given that another event has already occurred.
- If the event of interest is A and the event B is known or assumed to have occurred.
- This probability is written $P(A|B)$, notation for the *probability of A given B*.

$$P(A | B) = \frac{P(A \cap B)}{P(B)}$$

Chain Rule for Conditional Probability

$$P(A \cap B) = P(A)P(B|A) = P(B)P(A|B)$$

Now we can extend this formula to three or more events:

$$P(A \cap B \cap C) = P(A \cap (B \cap C)) = P(A)P(B \cap C|A)$$

$$P(B \cap C) = P(B)P(C|B).$$

Conditioning both sides on A , we obtain

$$P(B \cap C|A) = P(B|A)P(C|A, B)$$

Combining Equation 1.6 and 1.7 we obtain the following chain rule:

$$P(A \cap B \cap C) = P(A)P(B|A)P(C|A, B).$$

Chain rule for conditional probability:

$$P(A_1 \cap A_2 \cap \cdots \cap A_n) = P(A_1)P(A_2|A_1)P(A_3|A_2, A_1) \cdots P(A_n|A_{n-1}A_{n-2} \cdots A_1)$$

https://www.probabilitycourse.com/chapter1/1_4_0_conditional_probability.php

Conditional independence

Definition: X is conditionally independent of Y given Z , if the probability distribution governing X is independent of the value of Y , given the value of Z

$$(\forall i, j, k) P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z_k)$$

$$P(X|Y, Z) = P(X|Z)$$

Example:

$$P(\text{Thunder}|\text{Rain, Lightning}) = P(\text{Thunder}|\text{Lightning})$$

Slide credit: Tom Mitchell

Naïve Bayes Classifier

Bayes rule:

$$P(Y = y_k | X_1, \dots, X_n) = \frac{P(Y = y_k)P(X_1, \dots, X_n | Y = y_k)}{\sum_j P(Y = y_j)P(X_1, \dots, X_n | Y = y_j)}$$

Assume conditional independence among X_i 's:

$$P(Y = y_k | X_1, \dots, X_n) = \frac{P(Y = y_k)\prod_i P(X_i | Y = y_k)}{\sum_j P(Y = y_j)\prod_i P(X_i | Y = y_j)}$$

Pick the most probable (MAP) Y

$$\hat{Y} \leftarrow \operatorname{argmax} P(Y = y_k)\prod_i P(X_i | Y = y_k)$$

y_k



Prior
Probability

MLE



Slide credit: Tom
Mitchell

Naïve Bayes Classifier – Example 1



Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

A: attributes

M: mammals,

N: non-mammals

$$P(A | M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$P(A | N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$P(A | M)P(M) = 0.06 \times \frac{7}{20} = 0.021$$

$$P(A | N)P(N) = 0.004 \times \frac{13}{20} = 0.0027$$

$P(A|M)P(M) > P(A|N)P(N)$

=> Mammals

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

Naïve Bayes Classifier – Example 2



- Use case is to predict the chances of playing tennis given a few parameters

- Weather outlook
- Temperature
- Humidity
- Wind

• Data is given for 14 days

Day	<i>Outlook</i>	<i>Temperature</i>	<i>Humidity</i>	<i>Wind</i>	<i>PlayTennis</i>
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Naïve Bayes Classifier – Example 2

New input set to be classified is –

{*Outlook = sunny, Temperature = cool, Humidity = high, Wind = strong*}

Output parameter (*PlayTennis*) has 2 classes – Yes, No

Step-1 Calculate conditional probabilities of each input parameter given each of the classes (likelihood)

$$P(\text{Outlook}=\text{sunny} \mid \text{PlayTennis}=\text{yes}) = 2 / 9$$

$$P(\text{Outlook}=\text{sunny} \mid \text{PlayTennis}=\text{no}) = 3 / 5$$

$$P(\text{Temp}=\text{cool} \mid \text{PlayTennis}=\text{yes}) = 3 / 9$$

$$P(\text{Temp}=\text{cool} \mid \text{PlayTennis}=\text{no}) = 1 / 5$$

$$P(\text{Humidity}=\text{high} \mid \text{PlayTennis}=\text{yes}) = 3 / 9$$

$$P(\text{Humidity}=\text{high} \mid \text{PlayTennis}=\text{no}) = 4 / 5$$

$$P(\text{Wind}=\text{strong} \mid \text{PlayTennis}=\text{yes}) = 3 / 9$$

$$P(\text{Wind}=\text{strong} \mid \text{PlayTennis}=\text{no}) = 3 / 5$$

Step-2 Calculate total probability of each of the classes (prior)

$$P(\text{PlayTennis}=\text{yes}) = 9/14$$

$$P(\text{PlayTennis}=\text{no}) = 5/14$$

Naïve Bayes Classifier – Example 2

Step-3 Calculate probability for each class given the input conditions in new input set (posterior)

$P(\text{PlayTennis=yes} \mid \text{Outlook=sunny, Temp=cool, Humidity=high, Wind=strong})$

$$\begin{aligned}&= P(\text{yes}). P(\text{sunny} \mid \text{yes}). P(\text{cool} \mid \text{yes}). P(\text{high} \mid \text{yes}). P(\text{strong} \mid \text{yes}) \\&= 9/14 * 2/9 * 3/9 * 3/9 * 3/9 = 0.00529\end{aligned}$$

$P(\text{PlayTennis=no} \mid \text{Outlook=sunny, Temp=cool, Humidity=high, Wind=strong})$

$$\begin{aligned}&= P(\text{no}). P(\text{sunny} \mid \text{no}). P(\text{cool} \mid \text{no}). P(\text{high} \mid \text{no}). P(\text{strong} \mid \text{no}) \\&= 5/14 * 3/5 * 1/5 * 4/5 * 3/5 = 0.02057\end{aligned}$$

Using Naïve Bayes classifier, $\text{argmax}(P) = \text{no}$, which mean the classifier assign $\text{PlayTennis} = \text{No}$

Discrete and Continuous value attributes

To compute, $P(x_k|C_i)$

- A_k is categorical:

the number of tuples of class C_i in D having the value x_k for A_k

$$P(x_k|C_i) = \frac{\text{the number of tuples of class } C_i \text{ in } D.}{\text{the number of tuples of class } C_i \text{ in } D.}$$

- A_k is continuous:

A continuous-valued attribute is typically assumed to have a Gaussian distribution with a mean μ and standard deviation σ

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}).$$

Estimate Probabilities from Continuous Data

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Normal distribution:

$$P(A_i | c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(A_i - \mu_j)^2}{2\sigma_{ij}^2}}$$

One for each (A_i, c_i) pair

For (Income, Class=No):

If Class=No

sample mean = 110

sample variance = 3179.16

$$P(\text{Income} = 120 | \text{No}) = \frac{1}{\sqrt{2\pi(56.38)}} e^{-\frac{(120-110)^2}{2(3179)}} = 0.0069$$

Example of Naïve Bayes Classifier

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$

- $$\begin{aligned} P(X|\text{Class}=\text{No}) &= P(\text{Refund}=\text{No}|\text{Class}=\text{No}) \\ &\quad \times P(\text{Married}|\text{Class}=\text{No}) \\ &\quad \times P(\text{Income}=120\text{K}|\text{Class}=\text{No}) \\ &= 4/7 \times 4/7 \times 0.0069 = 0.0023 \end{aligned}$$
- $$\begin{aligned} P(X|\text{Class}=\text{Yes}) &= P(\text{Refund}=\text{No}|\text{Class}=\text{Yes}) \\ &\quad \times P(\text{Married}|\text{Class}=\text{Yes}) \\ &\quad \times P(\text{Income}=120\text{K}|\text{Class}=\text{Yes}) \\ &= 1 \times 0 \times 1.2 \times 10^{-9} = 0 \end{aligned}$$

Since $P(X|\text{No})P(\text{No}) > P(X|\text{Yes})P(\text{Yes})$

Therefore $P(\text{No}|X) > P(\text{Yes}|X)$

$\Rightarrow \text{Class} = \text{No}$

Issues with Naïve Bayes Classifier

Consider the table with Tid = 6 deleted

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
7	Yes	Divorced	220K	yes
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Naïve Bayes Classifier:

$$P(\text{Refund} = \text{Yes} | \text{No}) = 2/6$$

$$P(\text{Refund} = \text{No} | \text{No}) = 4/6$$

$$P(\text{Refund} = \text{Yes} | \text{Yes}) = 0$$

$$P(\text{Refund} = \text{No} | \text{Yes}) = 1$$

$$P(\text{Marital Status} = \text{Single} | \text{No}) = 2/6$$

$$P(\text{Marital Status} = \text{Divorced} | \text{No}) = 0$$

$$P(\text{Marital Status} = \text{Married} | \text{No}) = 3/6$$

$$P(\text{Marital Status} = \text{Single} | \text{Yes}) = 2/3$$

$$P(\text{Marital Status} = \text{Divorced} | \text{Yes}) = 2/3$$

$$P(\text{Marital Status} = \text{Married} | \text{Yes}) = 0/3$$

For Taxable Income:

If class = No: sample mean = 91
sample variance = 685

If class = Yes: sample mean = 90
sample variance = 25

Naïve Bayes will not be able to classify X as Yes or No!

Given X = (Refund = Yes, divorced, 120K)

$$P(X | \text{No}) = 2/6 \times 0 \times 0.0083 = 0$$

$$P(X | \text{Yes}) = 0 \times 1/3 \times 1.2 \times 10^{-9} =$$

0

Issues with Naïve Bayes Classifier



- If one of the conditional probabilities is zero, then the entire expression becomes zero
- Need to use other estimates of conditional probabilities than simple fractions
- Probability estimation:

$$\text{Original : } P(A_i | C) = \frac{N_{ic}}{N_c}$$

c: number of classes / vocabulary

$$\text{Laplace : } P(A_i | C) = \frac{N_{ic} + 1}{N_c + c}$$

p: prior probability of the class

$$\text{m - estimate : } P(A_i | C) = \frac{N_{ic} + mp}{N_c + m}$$

N_c : number of instances in the class

N_{ic} : number of instances having attribute value A_i in class c

Naïve Bayes for Text Classification

- Naïve Bayes is commonly used for **text classification**
- For a document with **k** terms $d = (t_1, \dots, t_k)$

Fraction of documents in c

$$P(c|d) = P(c)P(d|c) = P(c) \prod_{t_i \in d} P(t_i|c)$$

- $P(t_i|c)$ = Fraction of terms from **all documents** in c that are t_i .

Number of times t_i appears in some document in c

$$P(t_i|c) = \frac{N_{ic} + 1}{N_c + T}$$

Laplace Smoothing

Total number of terms in all documents in c

Number of unique words (vocabulary size)

- Easy to implement and works relatively well

A Simple Example

Text	Tag	Which tag does the sentence <i>A very close game</i> belong to? i.e. $P(\text{sports} \mid A \text{ very close game})$
“A great game”	Sports	
“The election was over”	Not sports	Feature Engineering: Bag of words i.e use word frequencies without considering order
“Very clean match”	Sports	Using Bayes Theorem:
“A clean but forgettable game”	Sports	$P(\text{sports} \mid A \text{ very close game})$
“It was a close election”	Not sports	$= P(A \text{ very close game} \mid \text{sports}) P(\text{sports})$
		<hr/> $P(A \text{ very close game})$

We assume that every word in a sentence is **independent** of the other ones

$$P(a \text{ very close game}) = P(a) \times P(\text{very}) \times P(\text{close}) \times P(\text{game})$$

$$P(a \text{ very close game} \mid \text{Sports}) = P(a \mid \text{Sports}) \times P(\text{very} \mid \text{Sports}) \times P(\text{close} \mid \text{Sports}) \times P(\text{game} \mid \text{Sports})$$

“close” doesn’t appear in sentences of sports tag, So $P(\text{close} \mid \text{sports}) = 0$, which makes product 0

Laplace smoothing

- Laplace smoothing: we add 1 or in general constant k to every count so it's never zero.
- To balance this, we add the number of possible words to the divisor, so the division will never be greater than 1
- In our case, the possible words are ['a', 'great', 'very', 'over', 'it', 'but', 'game', 'election', 'clean', 'close', 'the', 'was', 'forgettable', 'match'].

Apply Laplace Smoothing

Word	P(word Sports)	P(word Not Sports)
a	2+1 / 11+14	1+1 / 9+14
very	1+1 / 11+14	0+1 / 9+14
close	0+1 / 11+14	1+1 / 9+14
game	2+1 / 11+14	0+1 / 9+14

$$\begin{aligned}
 & P(a|Sports) \times P(very|Sports) \times P(close|Sports) \times P(game|Sports) \times \\
 & P(Sports) \\
 & = 2.76 \times 10^{-5} \\
 & = 0.0000276
 \end{aligned}$$

$$\begin{aligned}
 & P(a|Not\ Sports) \times P(very|Not\ Sports) \times P(close|Not\ Sports) \times \\
 & P(game|Not\ Sports) \times P(Not\ Sports) \\
 & = 0.572 \times 10^{-5} \\
 & = 0.00000572
 \end{aligned}$$

Multinomial and Bernoulli model



Using conditional independence assumption

$$\text{Multinomial} \quad P(d|c) = P(\langle t_1, \dots, t_{n_d} \rangle | c) = \prod_{1 \leq k \leq n_d} P(X_k = t_k | c)$$

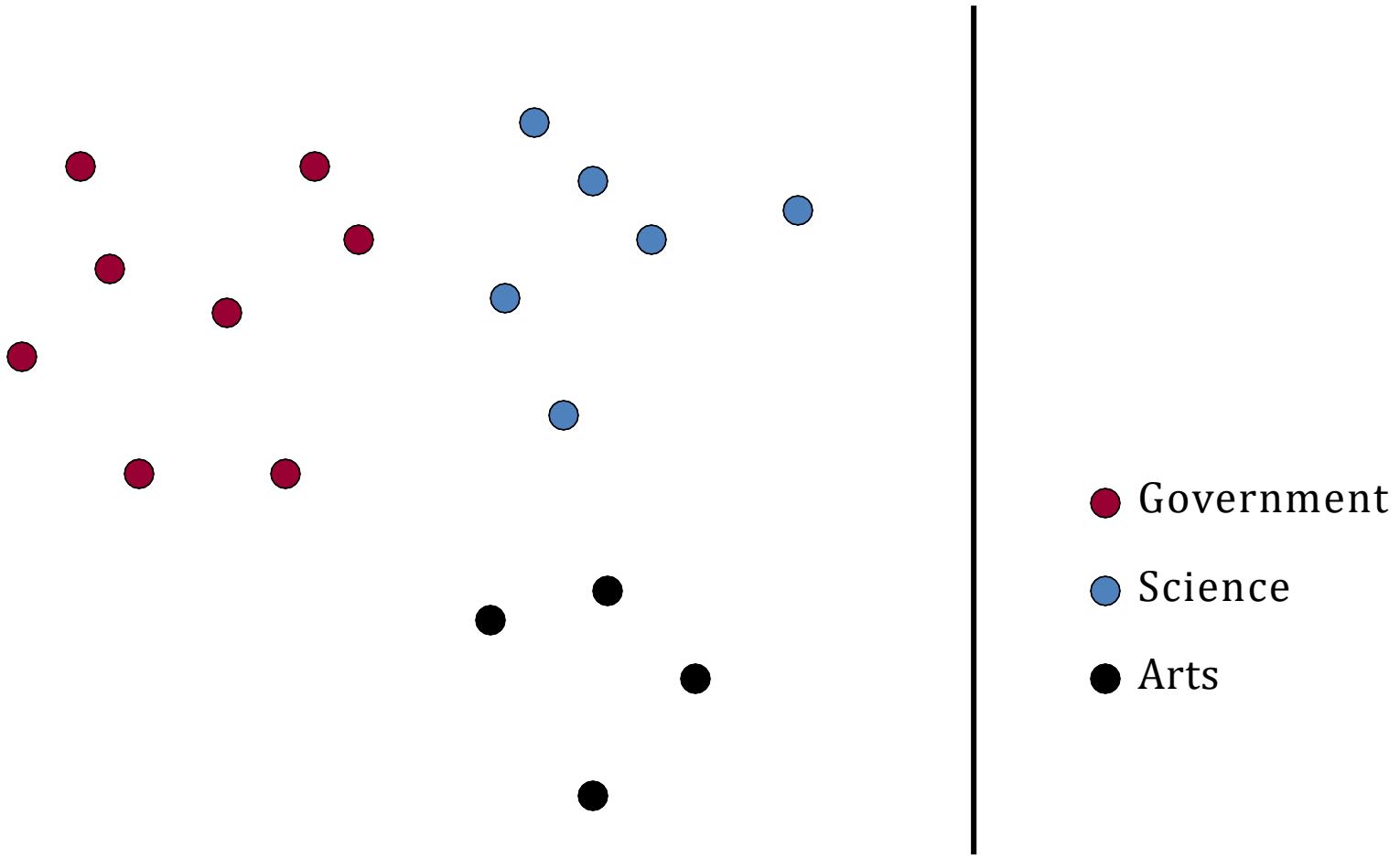
$$\text{Bernoulli} \quad P(d|c) = P(\langle e_1, \dots, e_M \rangle | c) = \prod_{1 \leq i \leq M} P(U_i = e_i | c).$$

- t_1, \dots, t_n and e_1, \dots, e_M are two different document representations.
- In the first case, X is the set of all term sequences (or, more precisely, sequences of term tokens).
- In the second case, X is $\{0, 1\}^M$

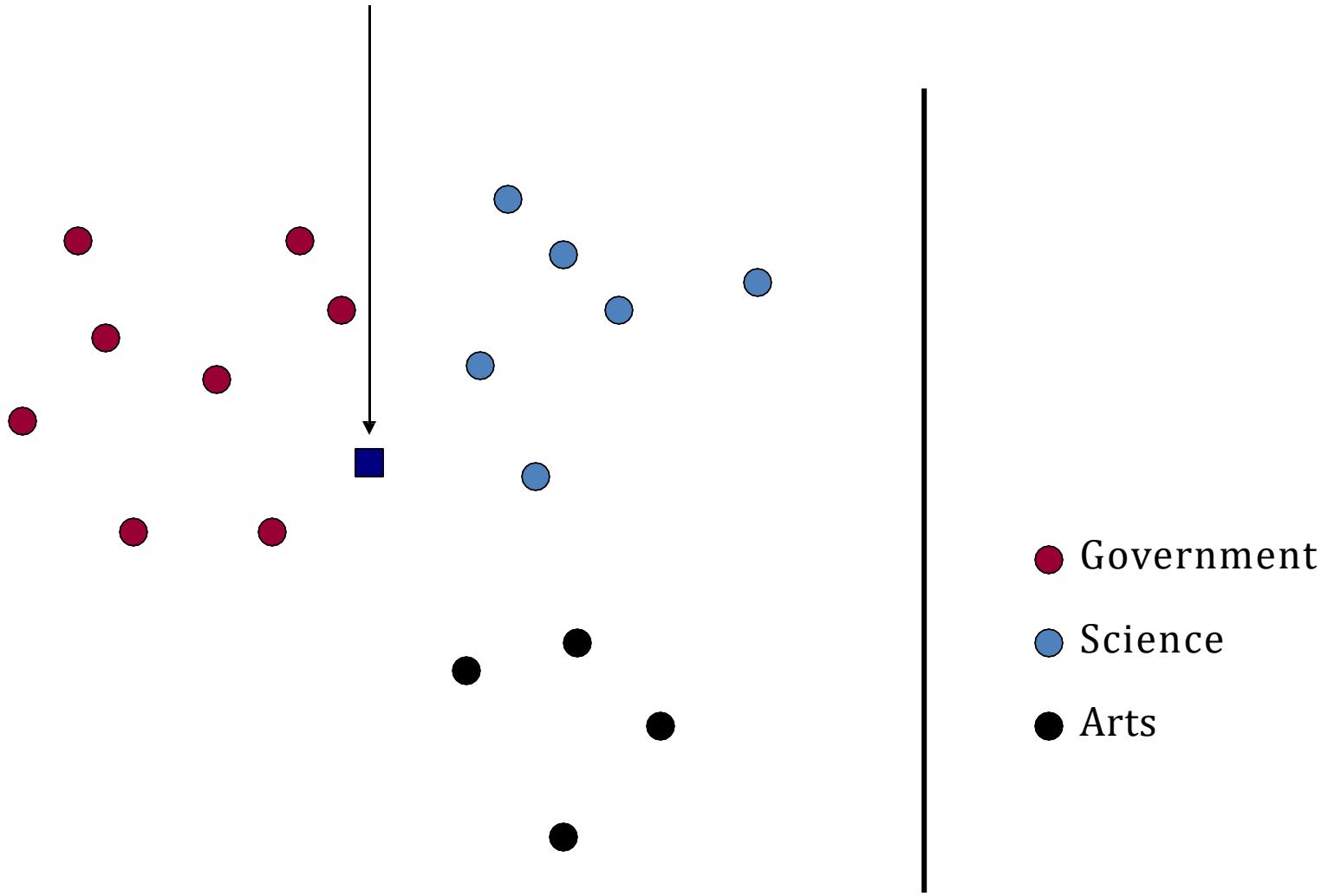
Classification Using Vector Spaces

- In vector space classification, training set corresponds to a labeled set of points (equivalently, vectors)
- **Premise 1:** Documents in the same class form a contiguous region of space
- **Premise 2:** Documents from different classes don't overlap (much)
- Learning a classifier: build surfaces to delineate classes in the space

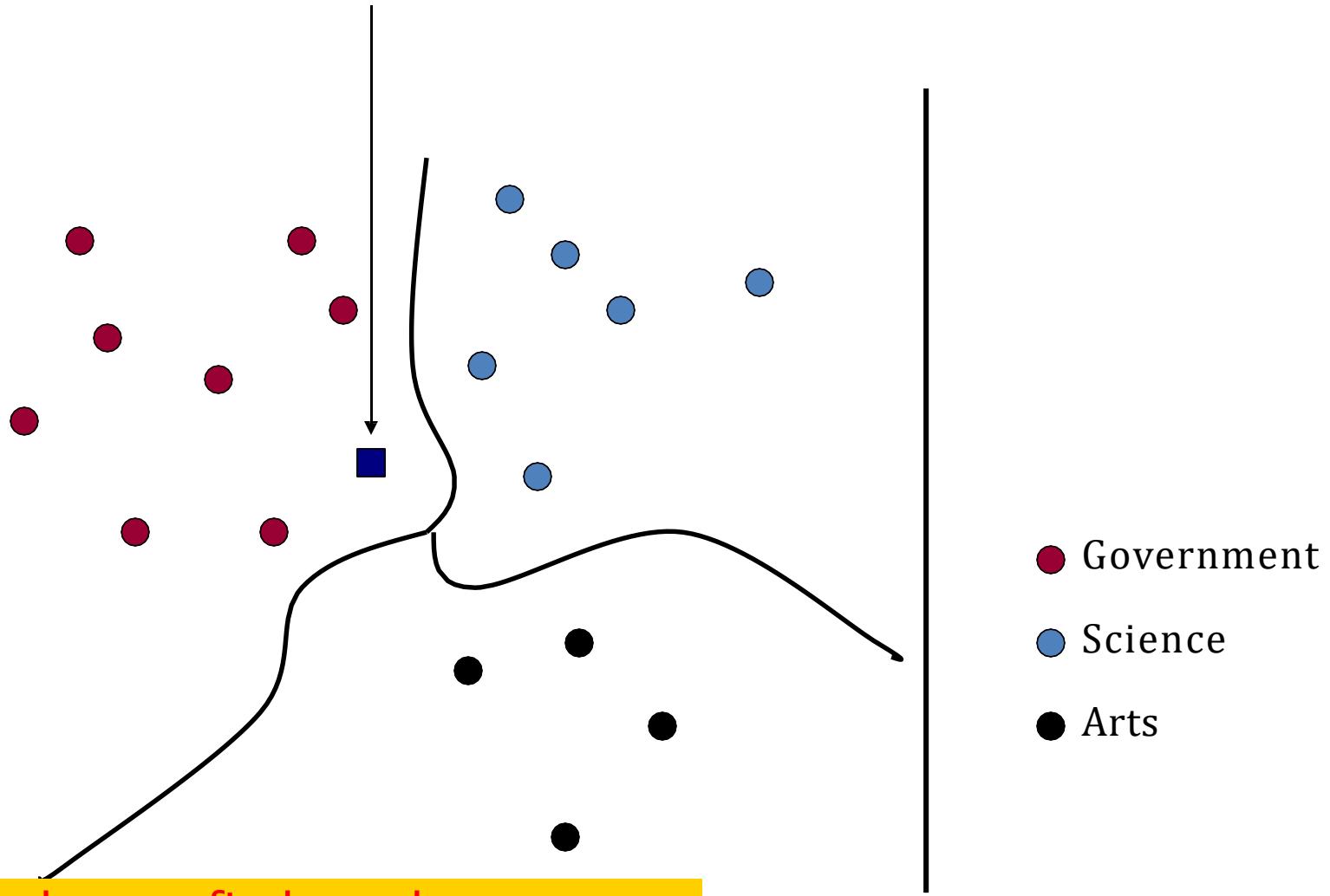
Documents in a Vector Space



Test Document of what class?



Test Document = Government



Our focus: how to find good separators

Definition of centroid

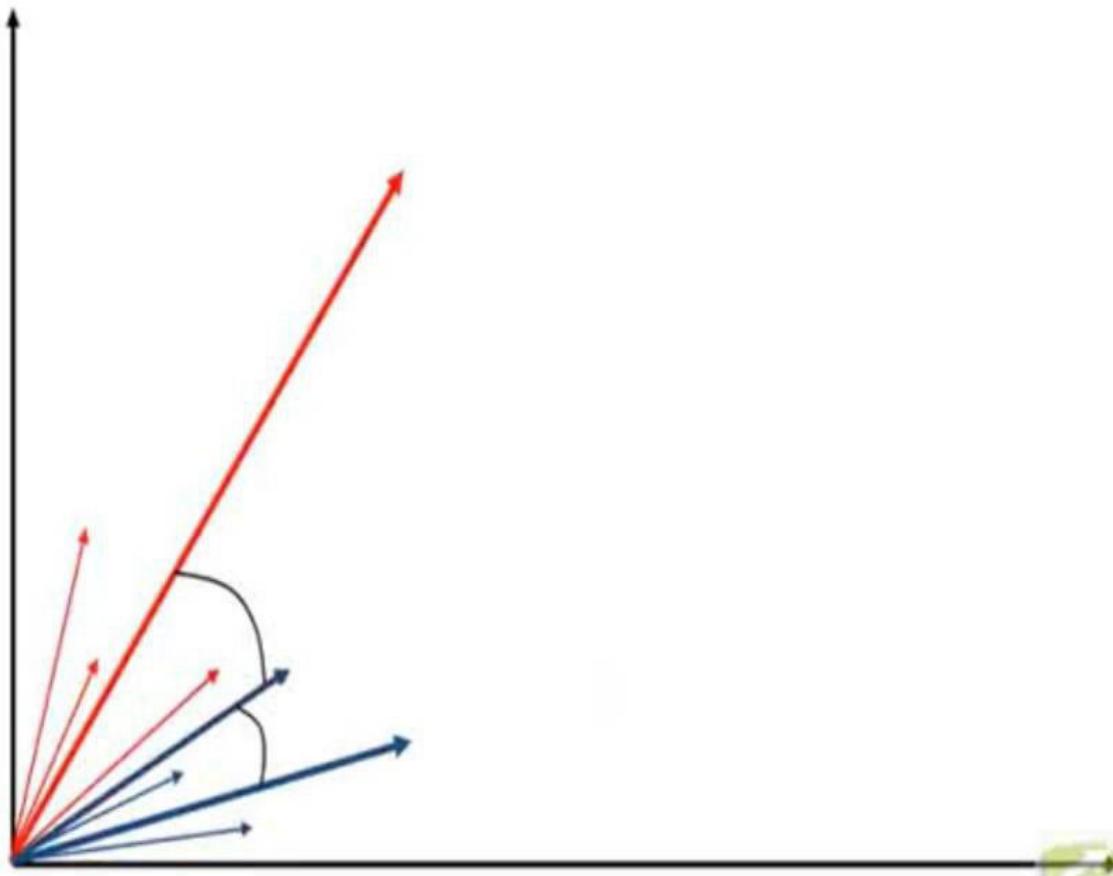
$$\vec{\mu}(c) = \frac{1}{|D_c|} \sum_{d \in D_c} \vec{v}(d)$$

- Where D_c is the set of all documents that belong to class c and $v(d)$ is the vector space representation of d .
- Note that centroid will in general not be a unit vector even when the inputs are unit vectors.*

Rocchio classification

- Rocchio forms a simple representative for each class: the centroid/prototype
- Classification: nearest prototype/centroid
- It does not guarantee that classifications are consistent with the given training data

Rocchio classification



Rocchio classification



- Little used outside text classification
 - It has been used quite effectively for text classification
 - But in general worse than Naïve Bayes
- Again, cheap to train and test documents

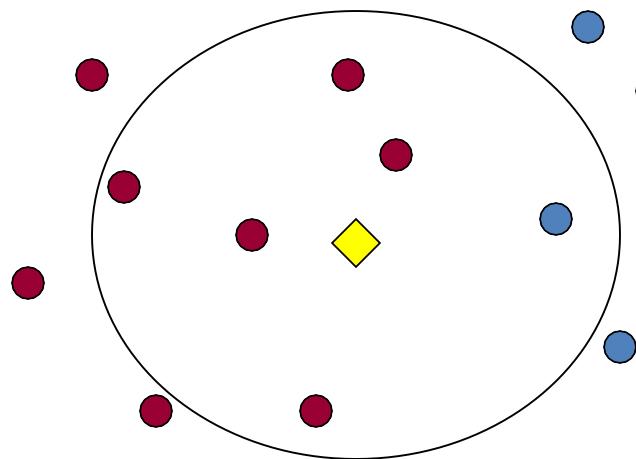
k Nearest Neighbor Classification



$k\text{NN}$ = k Nearest Neighbor

- To classify a document d :
- Define k -neighborhood as the k nearest neighbors of d
- Pick the majority class label in the k -neighborhood

Example: k=6 (6NN)



$P(\text{science}|\diamondsuit)?$

- Government
- Science
- Arts

Nearest-Neighbor Learning

- Learning: just store the labeled training examples D
- Testing instance x (*under 1NN*):
 - Compute similarity between x and all examples in D .
 - Assign x the category of the most similar example in D .
- Does not compute anything beyond storing the examples
- Also called:
 - Case-based learning
 - Memory-based learning
 - Lazy learning

Example



We have data from the questionnaires survey (to ask people opinion) and objective testing with two attributes (acid durability and strength) to classify whether a special paper tissue is good or not. Here is four training samples

X2 = Strength

X1 = Acid Durability (seconds)

Y = Classification
(kg/square meter)

7	7	Bad
---	---	-----

7	4	Bad
---	---	-----

3	4	Good
---	---	------

1	4	Good
---	---	------

Now the factory produces a new paper tissue that pass laboratory test with X1 = 3 and X2 = 7. Without another expensive survey, can we guess what the classification of this new tissue is?

7

7

$$(7-3)^2 + (7-7)^2 = 16$$

7

4

$$(7-3)^2 + (4-7)^2 = 25$$

3

4

$$(3-3)^2 + (4-7)^2 = 9$$

1

4

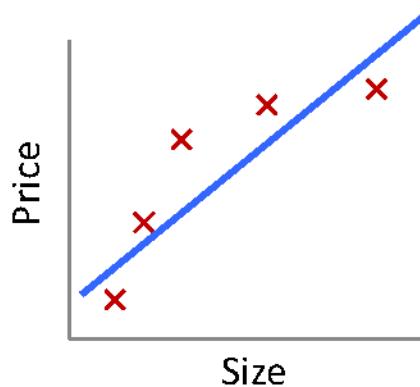
$$(1-3)^2 + (4-7)^2 = 13$$

3. Sort the distance and determine nearest neighbors based on the K-th minimum distance

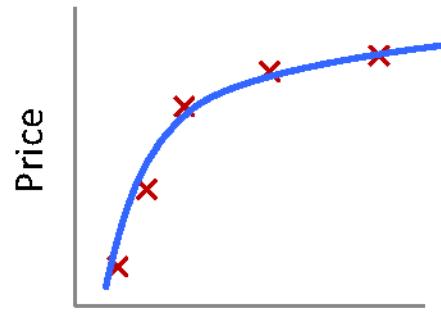
X1 = Acid Durability (seconds)	X2 = Strength (kg/square meter)	Square Distance to query instance (3, 7)	Rank minimum distance	Is it included in 3- Nearest neighbors?
7	7	$(7-3)^2 + (7-7)^2 = 16$	3	Yes
7	4	$(7-3)^2 + (4-7)^2 = 25$	4	No
3	4	$(3-3)^2 + (4-7)^2 = 9$	1	Yes
1	4	$(1-3)^2 + (4-7)^2 = 13$	2	Yes

X1 = Acid Durability (seconds)	X2 = Strength (kg/square meter)	Square Distance to Rank query instance (3, 7)	Is it included in 3-Nearest neighbors?	Y = Category of nearest Neighbor
7	7	$(7-3)^2 + (7-7)^2 = 16$ 3	Yes	Bad
7	4	$(7-3)^2 + (4-7)^2 = 25$ 4	No	-
3	4	$(3-3)^2 + (4-7)^2 = 9$ 1	Yes	Good
1	4	$(1-3)^2 + (4-7)^2 = 13$ 2	Yes	Good

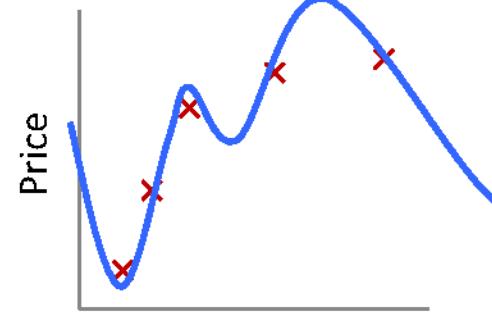
Quality of Fit



$\theta_0 + \theta_1 x$
Underfitting
(high bias)



$\theta_0 + \theta_1 x + \theta_2 x^2$
Correct fit



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$
Overfitting
(high variance)

Overfitting:

- The learned hypothesis may fit the training set very well ($J(\theta) \approx 0$)
- ...but fails to generalize to new examples

Addressing overfitting

x_1 = size of house

x_2 = no. of bedrooms

x_3 = no. of floors

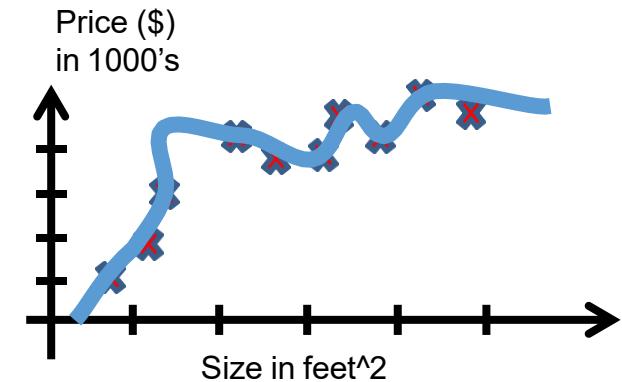
x_4 = age of house

x_5 = average income in neighborhood

x_6 = kitchen size

⋮

x_{100}



Addressing overfitting

1. Reduce number of features.

Manually select which features to keep.
Model selection algorithm

2. Regularization.

Keep all the features, but reduce magnitude/values of parameters θ_j .

Works well when we have a lot of features, each of which contributes a bit to predicting y .

Evaluating Classification

- Evaluation must be done on test data that are independent of the training data
 - Sometimes use cross-validation (averaging results over multiple training and test splits of the overall data)
- Easy to get good performance on a test set that was available to the learner during training (e.g., just memorize the test set)

Evaluating Classification

- Measures: precision, recall, F1, classification accuracy
- **Classification accuracy**: r/n where n is the total number of test docs and r is the number of test docs correctly classified

Evaluating classification

- Evaluation must be done on test data that are independent of the training data (usually a disjoint set of instances).
- It's easy to get good performance on a test set that was available to the learner during training (e.g., just memorize the test set).
- Measures: Precision, recall, F_1 , classification accuracy

Precision P and recall R

	in the class	not in the class
predicted to be in the class	true positives (TP)	false positives (FP)
predicted to not be in the class	false negatives (FN)	true negatives (TN)

$$P = TP / (TP + FP)$$

$$R = TP / (TP + FN)$$

A combined measure: F

- F_1 allows us to trade off precision against recall.

$$F_1 = \frac{1}{\frac{\frac{1}{P} + \frac{1}{R}}{2}} = \frac{2PR}{P + R}$$

- This is the harmonic mean of P and R :

Take-away today

- Text classification: definition & relevance to information retrieval
 - Naive Bayes: simple baseline text classifier
 - Theory: derivation of Naive Bayes classification rule & analysis
 - Vector space classification
 - Rocchio classification
 - K-Nearest Neighbour classification
 - Evaluation of text classification: how do we know it worked / didn't work?



THANK YOU

WDM 116: Various Classification Methods

<https://www.youtube.com/watch?v=lWdfh-xOm8Y>

[WDM 117: Bayes Rules Of Text Classification – YouTube](#)

[WDM 118: Example of Multivariate Bernoulli Model - YouTube](#)



BITS Pilani
Pilani Campus

Information Retrieval

Dr. Vijayalakshmi Anand,
BITS Pilani



Session 5: Ranked Retrieval

Date – 24th Dec 2022

Time – 4.15 pm to 6.15 pm

These slides are prepared by the instructor Prof. Manning., with grateful acknowledgement of and many others who made their course materials freely available online.

Agenda

Vector Space Model (T1 Chapter 6)

- Ranked Retrieval
- Scoring documents
- Term frequency
- Collection statistics
- Weighting schemes
- Vector space scoring

Ranked Retrieval

- Thus far, our queries have all been **Boolean**.
- Documents either match or don't.
- **Good for expert users** with precise understanding of their needs and of the collection.
- **Also good for applications**: Applications can easily consume 1000s of results.
- **Not good for the majority of users**
- Most users are not capable of writing Boolean queries . . .
. . or they are, but they think it's too much work.
- Most users don't want to wade through 1000s of results.
- This is particularly true of web search.

Boolean search Problem :



Feast or famine

- Boolean queries often result in either too few ($=0$) or too many (1000s) results.
- Query 1 (boolean disjunction OR):

[standard user dlink 650]

→ 200,000 hits – **feast**

- Query 2 (boolean conjunction AND):

[standard user dlink 650 no card found]

→ 0 hits – **famine**

- In Boolean retrieval, it takes a lot of skill to come up with a query that produces a manageable number of hits.

Feast or famine:

No problem in ranked retrieval

- With ranking, large result sets are not an issue.
- Just show the top 10 results
- Doesn't overwhelm the user
- Premise: the ranking algorithm works: **More relevant results are ranked higher than less relevant results.**

Scoring as the basis of Ranked Retrieval

- We wish to rank documents that are more relevant higher than documents that are less relevant.
- How can we accomplish such a ranking of the documents in the collection with respect to a query?
- Assign a score to each query-document pair, say in [0, 1].
- This score measures how well document and query “match”.

Query-document matching scores

- How do we compute the score of a query-document pair?
- Let's start with a one-term query.
- 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
- We will look at a number of alternatives for doing this.

Jaccard coefficient

- A commonly used measure of overlap of two sets
- Let A and B be two sets
- Jaccard coefficient:

$$\text{JACCARD}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

$(A \neq \emptyset \text{ or } B \neq \emptyset)$

- $\text{JACCARD}(A, A) = 1$
- $\text{JACCARD}(A, B) = 0$ if $A \cap B = \emptyset$
- A and B don't have to be the same size.
- Always assigns a number between 0 and 1.

Jaccard coefficient: Example

- What is the query-document match score that the Jaccard coefficient computes for:
- Query: “ides of March”
- Document1 “Caesar died in March”
- Document2 “long march”
- $\text{JACCARD}(q, d1) = 1/6$
- $\text{Jaccard}(q, d2) = 1/5$

What's wrong with Jaccard?

- It doesn't consider term frequency (how many occurrences a term has).
- Rare terms are more informative than frequent terms. Jaccard does not consider this information.
- Ex: Birla Institute of Technology and Science
- “Birla Institute” versus “of Technology”
- We need a more sophisticated way of normalizing for the length of a document. $|A \cap B| / \sqrt{|A \cup B|}$
- Later in this lecture, we'll use (cosine) instead of $|A \cap B| / |A \cup B|$ (Jaccard) for length normalization.

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 $\in \{0, 1\}^{|\mathcal{V}|}$.

Binary incidence 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

Bag of words model

- We do not consider the **order** of words in a document.
- *John is quicker than Mary and Mary is quicker than John* are represented the same way.
- This is called a **bag of words model**.
- In a sense, this is a step back: The positional index was able to distinguish these two documents.
- For now: bag of words model

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** .
- We want to use tf when computing query-document match scores.
- But how?
- Raw term frequency is not what we want because:
- 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.
- Relevance does not increase proportionally with term frequency.

Frequency in document vs. frequency in collection

- In addition, to term frequency (the frequency of the term in the document) . . .
- . . . we also want to use the frequency of the term **in the collection** for weighting and ranking.

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.

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).
- A document containing this term is more likely to be relevant than a document that doesn't . . .
- . . . but words like GOOD, INCREASE and LINE are not sure indicators of relevance.
- → **For frequent terms** like GOOD, INCREASE and LINE, we want positive weights . . .
- . . . but **lower weights** than for rare terms.

Document frequency

- We want **high weights** for rare terms like ARACHNOCENTRIC.
- We want **low (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:

$$idf_t = \log_{10} \frac{N}{df_t}$$

(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 “dampen” the effect of idf

Examples for idf_t

- 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

Effect of idf on ranking

- idf affects the ranking of documents for **queries with at least two terms**.
- For example, in the query “arachnocentric line”, idf weighting **increases** the relative weight of ARACHNOCENTRIC and **decreases** the relative weight of LINE.
- idf has **little effect** on ranking for **one-term queries**.

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
- Why these numbers?
- Which word is a better search term (and should get a higher weight)?
 - This example suggests that df (and idf) is better for weighting than cf (and “icf”).

tf-idf weighting

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

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

- 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 :

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

- 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)

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 $\in \{0, 1\}^M$.

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 $\in \mathbb{R}^{|V|}$.

Documents as vectors

- Each document is now represented as a real-valued vector of tf-idf weights $\in \mathbb{R}^{|V|}$.
- So we have a $|V|$ -dimensional real-valued vector space.
- Terms are **axes** of the space.
- Documents are **points** or **vectors** in this space.
- Very high-dimensional: tens of millions of dimensions when you apply this to web search engines
- Each vector is very sparse - most entries are zero.

Queries as vectors

- Key idea 1: do the same for queries: represent them as vectors in the high-dimensional space
- Key idea 2: Rank documents according to their proximity to the query
 - proximity = similarity
 - proximity \approx negative distance
 - rank relevant documents higher than nonrelevant documents

How do we formalize vector space similarity?

- First cut: (negative) distance between two points
- ($=$ distance between the end points of the two vectors)
- Euclidean distance?
- Euclidean distance is a bad idea . . .
- . . . because Euclidean distance is **large** for vectors of **different lengths**.

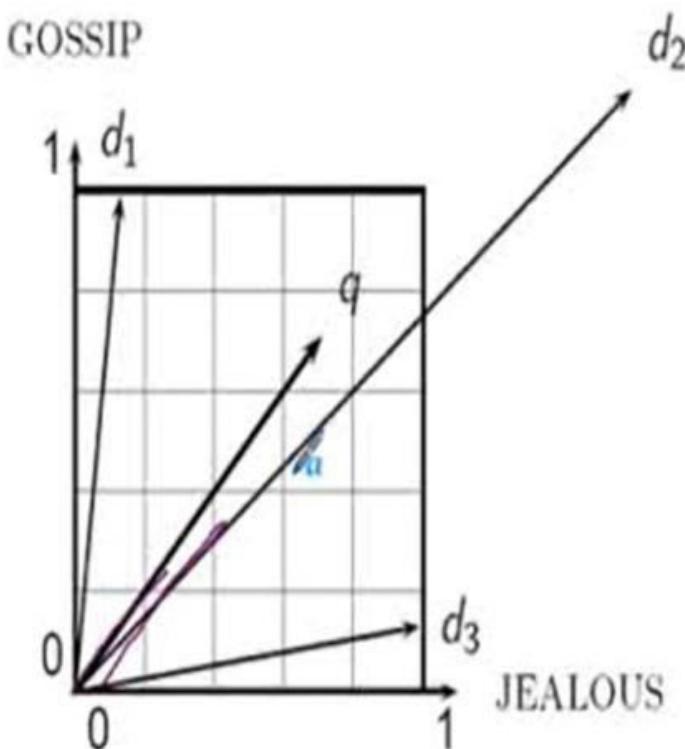
$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2}.$$

Use angle instead of distance

- Rank documents according to angle with query
- Thought experiment: 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.

Why distance is a bad idea

The Euclidean distance between \vec{q} and $\vec{d_2}$ is large even though the distribution of terms in the query \vec{q} and the distribution of terms in the document $\vec{d_2}$ are very similar.



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} and \vec{d} . Or equivalently, the cosine of the angle between \vec{q} and \vec{d} .
- Larger cosine score(small angle), larger is similarity.

Cosine for normalized vectors

- For normalized vectors, the cosine is equivalent to the dot product or scalar product.

$$\cos(\vec{q}, \vec{d}) = \vec{q} \cdot \vec{d} = \sum_i q_i \cdot d_i$$

- (if \vec{q} and \vec{d} are length-normalized).

Cosine: Example

term frequencies (counts)

How similar are
these novels?

SaS:
Sense and
Sensibility

PaP:
Pride and
Prejudice

WH:
Wuthering
Heights

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

Cosine: Example

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

log frequency weighting & 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

- $\cos(\text{SaS}, \text{PaP}) \approx$
 - $0.789 * 0.832 + 0.515 * 0.555 + 0.335 * 0.0 + 0.0 * 0.0 \approx 0.94.$
- $\cos(\text{SaS}, \text{WH}) \approx 0.79$
- $\cos(\text{PaP}, \text{WH}) \approx 0.69$

Some formulas for Sim

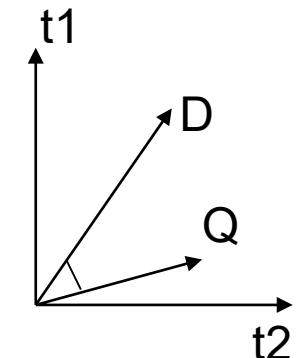
Dot product

$$Sim(D, Q) = \sum (a_i * b_i)$$

Cosine

$$Sim(D, Q) = \frac{\sum (a_i * b_i)}{\sqrt{\sum_i a_i^2 * \sum_i b_i^2}}$$

$$2 \sum (a_i * b_i)$$



Dice

$$Sim(D, Q) = \frac{\sum_i}{\sum_i a_i^2 + \sum_i b_i^2}$$

Jaccard

$$Sim(D, Q) = \frac{\sum (a_i * b_i)}{\sum_i a_i^2 + \sum_i b_i^2 - \sum_i (a_i * b_i)}$$

Summary: 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

Comments on Vector Space Models



- Simple, mathematically based approach.
- Considers both local (tf) and global (idf) word occurrence frequencies.
- Provides partial matching and ranked results.
- Tends to work quite well in practice despite obvious weaknesses.
- Allows efficient implementation for large document collections.

Problems with Vector Space Model

- Missing semantic information (e.g. word sense).
- Missing syntactic information (e.g. phrase structure, word order, proximity information).
- Assumption of term independence (e.g. ignores synonymy).
- Lacks the control of a Boolean model (e.g., *requiring* a term to appear in a document).
 - Given a two-term query “A B”, may prefer a document containing A frequently but not B, over a document that contains both A and B, but both less frequently.

57 Ranked Retrieval (13 mins)

<https://www.youtube.com/watch?v=CTwqDj5gtLo>

58 Jaccard Score (14 mins)

<https://www.youtube.com/watch?v=eqpzVedX1kY&t=605s>

59 TF and bag of words (24 mins)

https://www.youtube.com/watch?v=n_ul9GIOIkQ

WDM 60: Inverse Document Frequency (16 mins)

<https://www.youtube.com/watch?v=97Ey6aCeask>

WDM 61: TF-IDF Score (16 mins)

<https://www.youtube.com/watch?v=r8zeOv638-A>

WDM 62: Documents AS TF-IDF Vectors (16 mins)

<https://www.youtube.com/watch?v=zUyWj7Bx8HM>

WDM 63: Length Normalization (28 mins)

<https://www.youtube.com/watch?v=70n2xxgEk7w>

64 Cosine Similarity Example (24 mins)

<https://www.youtube.com/watch?v=HW9W6EBytLg>

WDM 66: Variants of TF IDF Weights - YouTube



THANK YOU

Here is a simplified example of the vector space retrieval model. Consider a very small collection C that consists in the following three documents:

- d1: “new york times”
- d2: “new york post”
- d3: “los angeles times”

Some terms appear in two documents, some appear only in one document. The total number of documents is $N=3$. Therefore, the idf values for the terms are:

angles	$\log_2(3/1)=1.584$
los	$\log_2(3/1)=1.584$
new	$\log_2(3/2)=0.584$
post	$\log_2(3/1)=1.584$
times	$\log_2(3/2)=0.584$
york	$\log_2(3/2)=0.584$

For all the documents, we calculate the tf scores for all the terms in C. We assume the words in the vectors are ordered alphabetically.

	angeles	los	new	post	times	york
d1	0	0	1	0	1	1
d2	0	0	1	1	0	1
d3	1	1	0	0	1	0

Now we multiply the tf scores by the idf values of each term, obtaining the following matrix of documents-by-terms: (All the terms appeared only once in each document in our small collection, so the maximum value for normalization is 1.)

	angeles	los	new	post	times	york
d1	0	0	0.584	0	0.584	0.584
d2	0	0	0.584	1.584	0	0.584
d3	1.584	1.584	0	0	0.584	0

Given the following query: “new new times”, we calculate the $tf-idf$ vector for the query, and compute the score of each document in C relative to this query, using the cosine similarity measure. When computing the $tf-idf$ values for the query terms we divide the frequency by the maximum frequency (2) and multiply with the idf values.

q	0	0	$(2/2)*0.584=0.584$	0	$(1/2)*0.584=0.292$	0
---	---	---	---------------------	---	---------------------	---

We calculate the length of each document and of the query:

$$\text{Length of } d_1 = \sqrt{0.584^2 + 0.584^2 + 0.584^2} = 1.011$$

$$\text{Length of } d_2 = \sqrt{0.584^2 + 1.584^2 + 0.584^2} = 1.786$$

$$\text{Length of } d_3 = \sqrt{1.584^2 + 1.584^2 + 0.584^2} = 2.316$$

$$\text{Length of } q = \sqrt{0.584^2 + 0.292^2} = 0.652$$

Then the similarity values are:

$$\text{cosSim}(d_1, q) = (0*0 + 0*0 + 0.584*0.584 + 0*0 + 0.584*0.292 + 0.584*0) / (1.011*0.652) = 0.776$$

$$\text{cosSim}(d_2, q) = (0*0 + 0*0 + 0.584*0.584 + 1.584*0 + 0*0.292 + 0.584*0) / (1.786*0.652) = 0.292$$

$$\text{cosSim}(d_3, q) = (1.584*0 + 1.584*0 + 0*0.584 + 0*0 + 0.584*0.292 + 0*0) / (2.316*0.652) = 0.112$$

According to the similarity values, the final order in which the documents are presented as result to the query will be: d_1, d_2, d_3 .



BITS Pilani
Pilani Campus

Information Retrieval

Dr. Vijayalakshmi anand, Ph.D,
BITS Pilani



Session 9: Evaluation of IR System

Date – 11 Feb 2023

Time – 4.15 pm-6.15 pm

These slides are prepared by the instructor Prof. Manning,, with grateful acknowledgement of and many others who made their course materials freely available online.

Measures for a search engine

- How fast does it index?
 - Number of documents/hour
 - [Average document size]
- How fast does it search
 - Latency as a function of index size
- Expressiveness of query language
 - Ability to express complex information needs
 - Speed on complex queries

How do you tell if users are happy?

- All of the preceding criteria are *measurable*: we can quantify speed/size; we can make expressiveness precise
- The key measure: user happiness
 - What is this?
 - Speed of response/size of index are factors
 - But blindingly fast, useless answers won't make a user happy
- Need a way of quantifying user happiness

Measuring relevance

- Three elements:
 1. A benchmark document collection
 2. A benchmark suite of queries
 3. An assessment of either Relevant or Nonrelevant for each query and each document

Measuring relevance

Information need i

"I am looking for information on whether drinking red wine is more effective at reducing your risk of heart attacks than white wine."

Query q

[red wine white wine heart attack]

Document d'

At the heart of his speech was an attack on the wine industry lobby for downplaying the role of red and white wine in drunk driving.

Benchmark

A benchmark collection contains:

A set of standard documents and queries/topics.

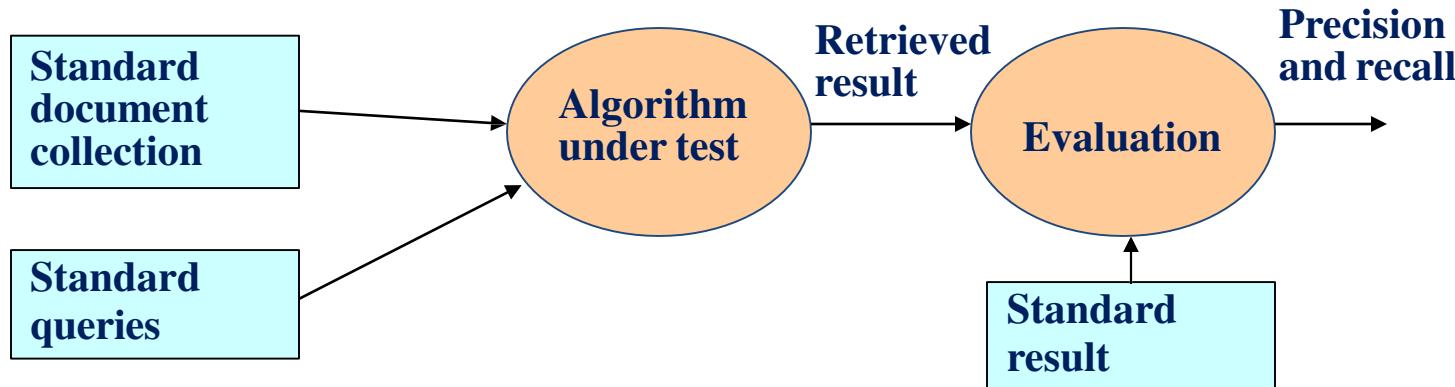
A list of relevant documents for each query.

Standard collections for traditional IR:

Smart collection:

<ftp://ftp.cs.cornell.edu/pub/smarter>

TREC: <http://trec.nist.gov/>



Early Test Collections

- Previous experiments were based on the SMART collection which is fairly small.
[\(ftp://ftp.cs.cornell.edu/pub/smart\)](ftp://ftp.cs.cornell.edu/pub/smart)

Collection Name	Number Of Documents	Number Of Queries	Raw Size (Mbytes)
CACM	3,204	64	1.5
CISI	1,460	112	1.3
CRAN	1,400	225	1.6
MED	1,033	30	1.1
TIME	425	83	1.5

- Different researchers used different test collections and evaluation techniques.

The TREC Benchmark

- TREC: Text REtrieval Conference (<http://trec.nist.gov/>)
Originated from the TIPSTER program sponsored by Defense Advanced Research Projects Agency (DARPA).
- Became an annual conference in 1992, co-sponsored by the National Institute of Standards and Technology (NIST) and DARPA.
- Participants submit the P/R values for the final document and query corpus and present their results at the conference.

Characteristics of the TREC Collection

- Both long and short documents (from a few hundred to over one thousand unique terms in a document).
- Test documents consist of:

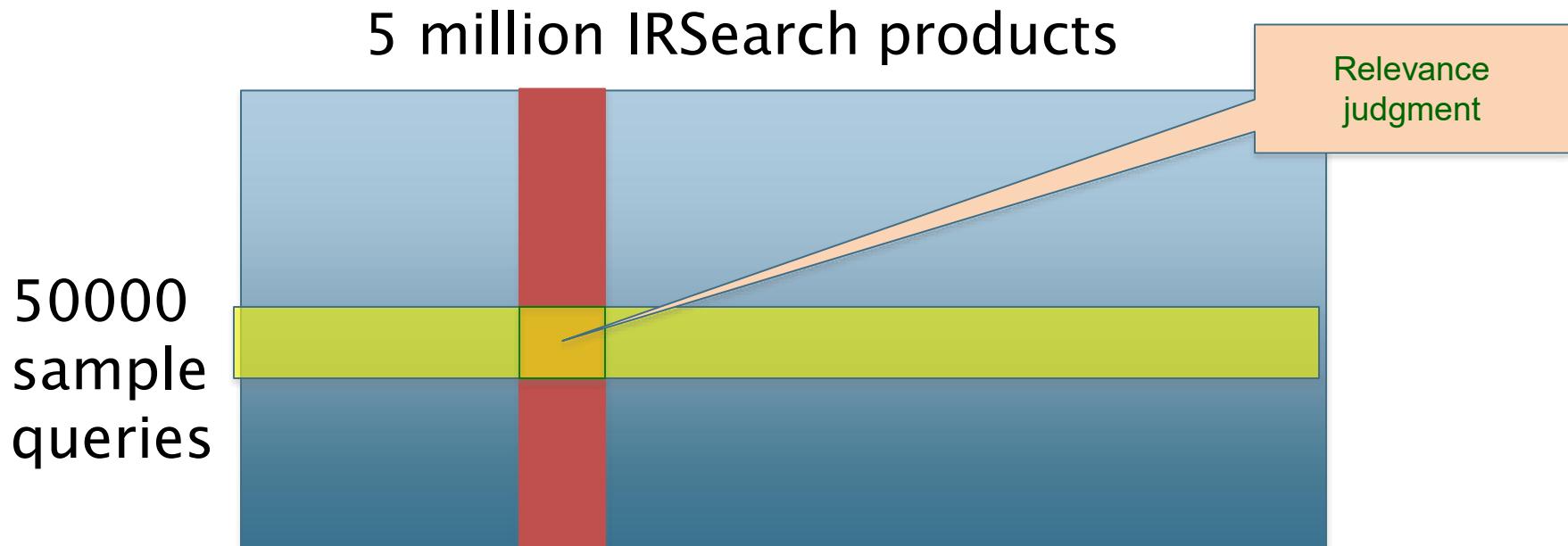
WSJ	Wall Street Journal articles (1986-1992)	550 M
AP	Associate Press Newswire (1989)	514 M
ZIFF	Computer Select Disks (Ziff-Davis Publishing)	493 M
FR	Federal Register	469 M
DOE	Abstracts from Department of Energy reports	190 M

So you want to measure the quality of a new search algorithm?

Benchmark documents

Benchmark query suite

Judgments of document relevance for each query



Relevance judgments

- Binary (relevant vs. non-relevant) in the simplest case
 - More nuanced relevance levels also used(0, 1, 2, 3...)
- What are some issues already?
- 5 million times 50K takes us into the range of a quarter trillion judgments
 - If each judgment took a human 2.5 seconds, we'd still need 10¹¹ seconds, or nearly \$300 million if you pay people \$10 per hour to assess 10K new products per day

Crowd source relevance judgments?

- Present query-document pairs to low-cost labor on online crowd-sourcing platforms
 - Hope that this is cheaper than hiring qualified assessors
- Lots of literature on using crowd-sourcing for such tasks
 - You get fairly good signal, but the variance in the resulting judgments is quite high

Evaluating an IR system

- Note: user need is translated into a query
- Relevance is assessed relative to the user need, not the query
- E.g., Information need: My swimming pool bottom is becoming black and needs to be cleaned.
- Query: pool cleaner
- Assess whether the doc addresses the underlying need, not whether it has these words

System Evaluation

- There are many retrieval models/ algorithms/ systems, which one is the best?
- What is the best component for:
 - Ranking function (dot-product, cosine, ...)
 - Term selection (stopword removal, stemming...)
 - Term weighting (TF, TF-IDF,...)
- How far down the ranked list will a user need to look to find some/all relevant documents?

Difficulties in Evaluating IR Systems

- Effectiveness is related to the *relevancy* of retrieved items.
- Relevancy is not typically binary but continuous.
- Even if relevancy is binary, it can be a difficult judgment to make.
- Relevancy, from a human standpoint, is:
 - Subjective: Depends upon a specific user's judgment.
 - Situational: Relates to user's current needs.
 - Cognitive: Depends on human perception and behavior.
 - Dynamic: Changes over time.

Human Labeled Corpora (Gold Standard)

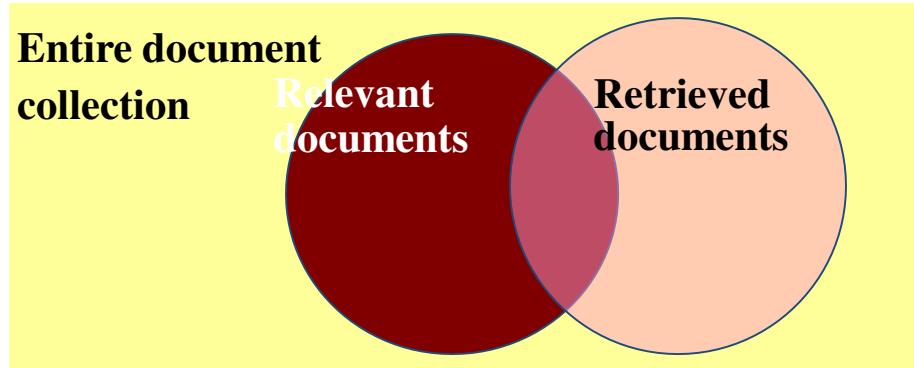


- Start with a corpus of documents.
- Collect a set of queries for this corpus.
- Have one or more human experts exhaustively label the relevant documents for each query.
- Typically assumes binary relevance judgments.
- Requires considerable human effort for large document/query corpora.



Unranked Retrieval Evaluation

Precision and Recall



	irrelevant		
	retrieved & irrelevant	Not retrieved & irrelevant	
relevant	retrieved & relevant	not retrieved but relevant	
retrieved			not retrieved

$$\text{recall} = \frac{\text{Number of relevant documents retrieved}}{\text{Total number of relevant documents}}$$

$$\text{precision} = \frac{\text{Number of relevant documents retrieved}}{\text{Total number of documents retrieved}}$$

- **Precision:** fraction of retrieved docs that are relevant
 $= P(\text{relevant} | \text{retrieved})$
- **Recall:** fraction of relevant docs that are retrieved
 $= P(\text{retrieved} | \text{relevant})$

	Relevant	Nonrelevant
Retrieved	tp	fp
Not Retrieved	fn	tn

- Precision $P = \frac{\text{tp}}{\text{tp} + \text{fp}}$
- Recall $R = \frac{\text{tp}}{\text{tp} + \text{fn}}$

Precision and Recall

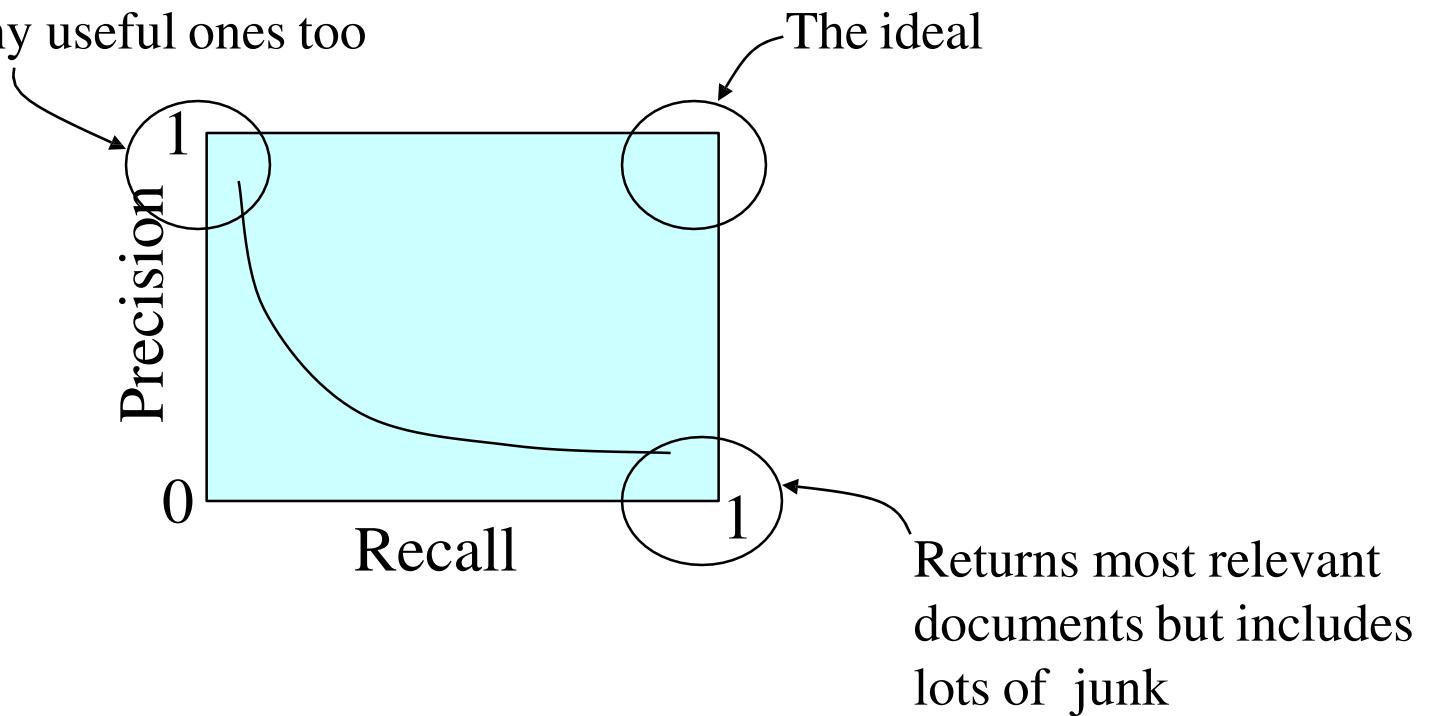
- Precision
 - The ability to retrieve top-ranked documents that are mostly relevant.
- Recall
 - The ability of the search to find all of the relevant items in the corpus.

Determining Recall is Difficult

- Total number of relevant items is sometimes not available:
 - Sample across the database and perform relevance judgment on these items.
 - Apply different retrieval algorithms to the same database for the same query. The aggregate of relevant items is taken as the total relevant set.

Trade-off between Recall and Precision

Returns relevant documents but misses many useful ones too



Computing Recall/Precision Points

- For a given query, produce the ranked list of retrievals.
- Adjusting a threshold on this ranked list produces different sets of retrieved documents, and therefore different recall/precision measures.
- Mark each document in the ranked list that is relevant according to the gold standard.
- Compute a recall/precision pair for each position in the ranked list that contains a relevant document.

Computing Recall/Precision Points: Example 1



n	doc #	relevant
1	588	x
2	589	x
3	576	
4	590	x
5	986	
6	592	x
7	984	
8	988	
9	578	
10	985	
11	103	
12	591	
13	772	x
14	990	

Let total # of relevant docs = 6
Check each new recall point:

$$R=1/6=0.167; P=1/1=1$$

$$R=2/6=0.333; P=2/2=1$$

$$R=3/6=0.5; P=3/4=0.75$$

$$R=4/6=0.667; P=4/6=0.667$$

$$R=5/6=0.833; P=5/13=0.38$$

Missing one
relevant document.
Never reach
100% recall

Computing Recall/Precision Points: Example 2



n	doc #	relevant
1	588	x
2	576	
3	589	x
4	342	
5	590	x
6	717	
7	984	
8	772	x
9	321	x
10	498	
11	113	
12	628	
13	772	
14	592	x

Let total # of relevant docs = 6
Check each new recall point:

$$R=1/6=0.167; P=1/1=1$$

$$R=2/6=0.333; P=2/3=0.667$$

$$R=3/6=0.5; P=3/5=0.6$$

$$R=4/6=0.667; P=4/8=0.5$$

$$R=5/6=0.833; P=5/9=0.556$$

$$R=6/6=1.0; P=6/14=0.429$$

F-Measure

- One measure of performance that takes into account both recall and precision.
- Harmonic mean of recall and precision:
- Compared to arithmetic mean, both need to be high for harmonic mean to be high.

$$F = \frac{2PR}{P+R} = \frac{2}{\frac{1}{R} + \frac{1}{P}}$$

Example

	relevant	not relevant	
retrieved	20	40	60
not retrieved	60	1,000,000	1,000,060
	80	1,000,040	1,000,120

- $P = 20 / (20 + 40) = 1/3$
- $R = 20 / (20 + 60) = 1/4$
- $F_1 = 2 \frac{1}{\frac{1}{3} + \frac{1}{4}} = 2/7$

E Measure (parameterized F Measure)

A variant of F measure that allows weighting emphasis on precision over recall:

Value of β controls trade-off:

$\beta = 1$: Equally weight precision and recall ($E=F$).

$\beta > 1$: Weight recall more.

$\beta < 1$: Weight precision more.

$$E = \frac{(1 + \beta^2)PR}{\beta^2P + R} = \frac{(1 + \beta^2)}{\frac{\beta^2}{R} + \frac{1}{P}}$$



Ranked Retrieval Evaluation

Rank-Based Measures

- Binary relevance
 - Precision@K (P@K)
 - Mean Average Precision (MAP)
 - Mean Reciprocal Rank (MRR)
- Multiple levels of relevance
 - Normalized Discounted Cumulative Gain (NDCG)

Precision@K

- Set a rank threshold K
- Compute % relevant in top K
- Ignores documents ranked lower than K
- Ex: Green-relevant
 - Prec@3 of 2/3
 - Prec@4 of 2/4
 - Prec@5 of 3/5
- In similar fashion we have Recall@K



Precision@K

■ Precision (**P@k**)

- The P@k for a given result list for a given query is the percentage of relevant documents among the top-k

Query: matrix movies

Relevant: 10, 582, 877, 10003

Result list: 582, 17, 5666, 10003, 10, 37, ...
1. 2. 3. 4. 5. 6. REL 877

P@1: 1/1 = 100%

P@2: 1/2 = 50%

P@3: 1/3 = 33%

P@4: 2/4 = 50%

P@5: 3/5 = 60%

Average Precision

■ Average Precision (AP)

- Let R_1, \dots, R_k be the sorted list of positions of the relevant document in the result list of a given query
- Then AP is the average of the $k P@R_i$ values

Query: matrix movies

Relevant: 10, 582, 877, 10003

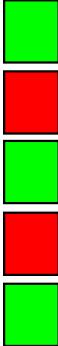
Result list: 582, 17, 5666, 10003, 10, ..., 877
REL NOT NOT REL REL REL
1. 2. 3. 4. 5. 40.

R_1, \dots, R_4 : 1, 4, 5, 40

$P@R_1, \dots, P@R_4$: 100%, 50%, 60%, 10%

AP: $(100\% + 50\% + 60\% + 10\%) / 4 = 55\%$

Mean Average Precision

- Consider rank position of each ***relevant*** doc
 - $K_1, K_2, \dots K_R$
- Compute Precision@K for each $K_1, K_2, \dots K_R$
- Average precision = average of P@K
- Ex:  has average precision of $\frac{1}{3} \cdot \left(\frac{1}{1} + \frac{2}{3} + \frac{3}{5} \right) \approx 0.76$
- MAP is Average Precision across multiple queries/rankings

Mean Average Precision

■ Mean Precisions (**MP@k**, **MP@R**, **MAP**)

- Given a benchmark with several queries + ground truth
- Then one can capture the quality of a system by taking the **mean** (average) of a given measure over all queries

MP@k = mean of the P@k values over all queries

MP@R = mean of the P@R values over all queries

MAP = mean of the AP values over all queries

Mean Reciprocal Rank

- Consider rank position, K, of first relevant doc
 - Could be – only clicked doc
- Reciprocal Rank score = $\frac{1}{K}$
- MRR is the mean RR across multiple queries

Mean Reciprocal Rank

- For each query return a ranked list of M candidate answers.
- Query score is 1/Rank of the first correct answer
 - *If first answer is correct: 1*
 - *else if second answer is correct: ½*
 - *else if third answer is correct: ⅓, etc.*
 - *Score is 0 if none of the M answers are correct*
- Take the mean over all N queries

$$MRR = \frac{\sum_{i=1}^N \frac{1}{rank_i}}{N}$$

BEYOND BINARY RELEVANCE



Web Images Video Local Shopping More ▾

Toyota safety

Search

Options ▾



Search Pad



108,000,000 results for
Toyota safety:



Toyota



CarsDirect



Shopping Sites

Also try: [toyota safety ratings](#), [toyota safety recall](#), [More...](#)

Toyota Recall

Toyota Takes Care of its Customers. Read the FAQs at [Toyota.com](#).
[www.Toyota.com/Recall](#)

Toyota Safety

& Latest Prices. Free Info. Toyota Research, Reviews.
[www.Toyota.Edmunds.com](#)

TOYOTA | Car Safety Innovation and Technology

Toyota home page for car safety and car technology Prius model.
[www.safetytoyota.com](#) - [Cached](#)

fair

Toyota home page for car safety and car technology ...

We are presenting Toyota's safety technologies for cars. We clearly explain about car safety and car technology using movies and more.
[www.safetytoyota.com/en-gb](#) - [Cached](#)

fair

Toyota Safety Ratings - Toyota Safety Features - Motor Trend ...

MotorTrend offers Toyota safety ratings, comprehensive auto safety reports, and more. View all of the standard Toyota safety features. ...
[motortrend.com/new_cars/07/toyota/safety_ratings/index.html](#) - 149k - [Cached](#)

Good

Toyota Motor Europe Corporate Site Safety

Our approach. Toyota believes that all stakeholders in the road safety equation share a responsibility to reduce the frequency of road accidents. ...
[www.toyota.eu/Safety](#) - [Cached](#)

Safety Toyota

Explore 5,000+ Pro Sports Choices. Save On Safety Toyota.
[BaseballGear.Shopzilla.com](#)

[PDF] pdf European Safety Brochure 2005

4047k - Adobe PDF - [View as html](#)
not guarantee that all accidents or injuries will be avoided when driving a Toyota and/or Lexus brand motor vehicle equipped with the safety systems ...
[www.toyota.no/Images/Safety_Brochure_tcm308-344461.pdf](#)

See your message here...

Toyota - Star Safety System

Star Safety System ... Toyota Mobility Program. Careers. Contact Us. Home. contact us. site map. your privacy rights. legal terms. Toyota Newsroom. sign up for info ...
[www.toyota.com/vehicles/demos/star-safety.html](#) - 58k - [Cached](#)

Toyota Prius Safety Ratings - CarsDirect

Get overall safety ratings and NHTSA crash test results for the Toyota Prius at CarsDirect.

Non-Binary Relevance

- Documents are rarely entirely relevant or non-relevant to a query
- Many sources of *graded relevance judgments*
 - Relevance judgments on a 5-point scale
 - Multiple judges
 - Click distribution and deviation from expected levels
(but click-through != relevance judgments)

Discounted Cumulative Gain

- Popular measure for evaluating web search and related tasks
- Two assumptions:
 - Highly relevant documents are more useful than marginally relevant documents
 - the lower the ranked position of a relevant document, the less useful it is for the user, since it is less likely to be examined

Cumulative Gain

- With graded relevance judgments, we can compute the *gain* at each rank.
- Cumulative Gain** at rank n:

$$CG_n = \sum_{i=1}^n rel_i$$

- (Where rel_i is the graded relevance of the document at position i)

n	doc #	relevance (gain)	CG _n
1	588	1.0	1.0
2	589	0.6	1.6
3	576	0.0	1.6
4	590	0.8	2.4
5	986	0.0	2.4
6	592	1.0	3.4
7	984	0.0	3.4
8	988	0.0	3.4
9	578	0.0	3.4
10	985	0.0	3.4
11	103	0.0	3.4
12	591	0.0	3.4
13	772	0.2	3.6
14	990	0.0	3.6

Discounted Cumulative Gain

- Uses *graded relevance* as a measure of usefulness, or *gain*, from examining a document
- Gain is accumulated starting at the top of the ranking and may be reduced, or *discounted*, at lower ranks
- Typical discount is $1/\log(rank)$
 - With base 2, the discount at rank 4 is $1/2$, and at rank 8 it is $1/3$

Summarize a Ranking: DCG

- What if relevance judgments are in a scale of [0,r]? $r > 2$
- Cumulative Gain (CG) at rank n
 - Let the ratings of the n documents be r_1, r_2, \dots, r_n (in ranked order)
 - $CG = r_1 + r_2 + \dots + r_n$
- Discounted Cumulative Gain (DCG) at rank n
 - $DCG = r_1 + r_2 / \log_2 2 + r_3 / \log_2 3 + \dots + r_n / \log_2 n$
 - We may use any base for the logarithm

Discounted Cumulative Gain

- DCG is the total gain accumulated at a particular rank p :
- Alternative formulation:

$$DCG_p = rel_1 + \sum_{i=2}^p \frac{rel_i}{\log_2 i}$$

- used by some web search companies
- emphasis on retrieving highly relevant documents

$$DCG_p = \sum_{i=1}^p \frac{2^{rel_i} - 1}{\log(1+i)}$$

Discounting Based on Position

- Users care more about high-ranked documents, so we **discount** results by $1/\log_2(\text{rank})$

- Discounted Cumulative Gain:**

$$DCG_n = rel_1 + \sum_{i=2}^n \frac{rel_i}{\log_2 i}$$

n	doc #	rel (gain)	CG _n	log _n	DCG _n
1	588	1.0	1.0	-	1.00
2	589	0.6	1.6	1.00	1.60
3	576	0.0	1.6	1.58	1.60
4	590	0.8	2.4	2.00	2.00
5	986	0.0	2.4	2.32	2.00
6	592	1.0	3.4	2.58	2.39
7	984	0.0	3.4	2.81	2.39
8	988	0.0	3.4	3.00	2.39
9	578	0.0	3.4	3.17	2.39
10	985	0.0	3.4	3.32	2.39
11	103	0.0	3.4	3.46	2.39
12	591	0.0	3.4	3.58	2.39
13	772	0.2	3.6	3.70	2.44
14	990	0.0	3.6	3.81	2.44

NDCG for summarizing rankings

- Normalized Discounted Cumulative Gain (NDCG) at rank n
 - Normalize DCG at rank n by the DCG value at rank n of the ideal ranking
 - The ideal ranking would first return the documents with the highest relevance level, then the next highest relevance level, etc
- Normalization useful for contrasting queries with varying numbers of relevant results
- NDCG is now quite popular in evaluating Web search

Normalized Discounted Cumulative Gain (NDCG)

- To compare DCGs, normalize values so that a *ideal ranking* would have a **Normalized DCG** of 1.0
- Ideal ranking:

n	doc #	rel (gain)	CG _n	log _n	DCG _n
1	588	1.0	1.0	0.00	1.00
2	589	0.6	1.6	1.00	1.60
3	576	0.0	1.6	1.58	1.60
4	590	0.8	2.4	2.00	2.00
5	986	0.0	2.4	2.32	2.00
6	592	1.0	3.4	2.58	2.39
7	984	0.0	3.4	2.81	2.39
8	988	0.0	3.4	3.00	2.39
9	578	0.0	3.4	3.17	2.39
10	985	0.0	3.4	3.32	2.39
11	103	0.0	3.4	3.46	2.39
12	591	0.0	3.4	3.58	2.39
13	772	0.2	3.6	3.70	2.44
14	990	0.0	3.6	3.81	2.44



n	doc #	rel (gain)	CG _n	log _n	IDCG _n
1	588	1.0	1.0	0.00	1.00
2	592	1.0	2.0	1.00	2.00
3	590	0.8	2.8	1.58	2.50
4	589	0.6	3.4	2.00	2.80
5	772	0.2	3.6	2.32	2.89
6	576	0.0	3.6	2.58	2.89
7	986	0.0	3.6	2.81	2.89
8	984	0.0	3.6	3.00	2.89
9	988	0.0	3.6	3.17	2.89
10	578	0.0	3.6	3.32	2.89
11	985	0.0	3.6	3.46	2.89
12	103	0.0	3.6	3.58	2.89
13	591	0.0	3.6	3.70	2.89
14	990	0.0	3.6	3.81	2.89

Normalized Discounted Cumulative Gain (NDCG)

- Normalize by DCG of the ideal ranking:

$$NDCG_n = \frac{DCG_n}{IDCG_n}$$

- $NDCG \leq 1$ at all ranks
- $NDCG$ is comparable across different queries

n	doc #	rel (gain)	DCG _n	IDCG _n	NDCG _n
1	588	1.0	1.00	1.00	1.00
2	589	0.6	1.60	2.00	0.80
3	576	0.0	1.60	2.50	0.64
4	590	0.8	2.00	2.80	0.71
5	986	0.0	2.00	2.89	0.69
6	592	1.0	2.39	2.89	0.83
7	984	0.0	2.39	2.89	0.83
8	988	0.0	2.39	2.89	0.83
9	578	0.0	2.39	2.89	0.83
10	985	0.0	2.39	2.89	0.83
11	103	0.0	2.39	2.89	0.83
12	591	0.0	2.39	2.89	0.83
13	772	0.2	2.44	2.89	0.84
14	990	0.0	2.44	2.89	0.84

NDCG Summary



- Discounted Cumulative Gain (DCG, nDCG)
 - Sometimes relevance comes in more than one shade, e.g.
0 = not relevant, 1 = somewhat rel, 2 = very relevant
 - There should be a "discount" in the score if very relevant documents come after only somewhat relevant documents
 - **Cumulative gain:** $CG@k = \sum_{i=1..k} rel_i$
 - **Discounted CG:** $DCG@k = rel_1 + \sum_{i=2..k} rel_i / \log_2 i$
 - Problem: CG and DCG are larger for larger result lists
 - Solution: normalize by maximally achievable value
 - **Ideal DCG:** $iDCG@k = DCG@k$ of ideal ranking

■ Discounted Cumulative Gain (DCG, nDCG), example

- Consider the following result list and relevances, assuming only 3 relevant documents overall (for this query)

Hit #1: very relevant 2

Hit #2: relevant 1

Hit #3: not relevant 0

Hit #4: very relevant 2

Hit #5: not relevant 0

- Then **DCG@5** = $2 + \frac{1}{\log_2 2} + \frac{2}{\log_2 4} = 2 + 1 + 1 =$
- And **iDCG@5** = $2 + \frac{2}{\log_2 2} + \frac{1}{\log_2 3} = 2 + 2 + 0.63 =$
- Hence **nDCG@5** = $\frac{4}{4.63} = 0.86 = 86\%$

NDCG – Example 2

4 documents: d_1, d_2, d_3, d_4

i	Ground Truth		Ranking Function ₁		Ranking Function ₂	
	Document Order	r_i	Document Order	r_i	Document Order	r_i
1	d_4	2	d_3	2	d_3	2
2	d_3	2	d_4	2	d_2	1
3	d_2	1	d_2	1	d_4	2
4	d_1	0	d_1	0	d_1	0
	$NDCG_{GT}=1.00$		$NDCG_{RF1}=1.00$		$NDCG_{RF2}=0.9203$	

$$DCG_{GT} = 2 + \left(\frac{2}{\log_2 2} + \frac{1}{\log_2 3} + \frac{0}{\log_2 4} \right) = 4.6309$$

$$DCG_{RF1} = 2 + \left(\frac{2}{\log_2 2} + \frac{1}{\log_2 3} + \frac{0}{\log_2 4} \right) = 4.6309$$

$$DCG_{RF2} = 2 + \left(\frac{1}{\log_2 2} + \frac{2}{\log_2 3} + \frac{0}{\log_2 4} \right) = 4.2619$$

$$MaxDCG = DCG_{GT} = 4.6309$$

Average Precision- Example



= the relevant documents

Ranking #1



	Recall	0.17	0.17	0.33	0.5	0.67	0.83	0.83	0.83	1.0
--	--------	------	------	------	-----	------	------	------	------	-----

	Precision	1.0	0.5	0.67	0.75	0.8	0.83	0.71	0.63	0.56	0.6
--	-----------	-----	-----	------	------	-----	------	------	------	------	-----

Ranking #2



	Recall	0.0	0.17	0.17	0.17	0.33	0.5	0.67	0.67	0.83	1.0
--	--------	-----	------	------	------	------	-----	------	------	------	-----

	Precision	0.0	0.5	0.33	0.25	0.4	0.5	0.57	0.5	0.56	0.6
--	-----------	-----	-----	------	------	-----	-----	------	-----	------	-----

$$\text{Ranking } \#1: (1.0 + 0.67 + 0.75 + 0.8 + 0.83 + 0.6) / 6 = 0.78$$

$$\text{Ranking } \#2: (0.5 + 0.4 + 0.5 + 0.57 + 0.56 + 0.6) / 6 = 0.52$$

Mean Average Precision (MAP)



= relevant documents for query 1

Ranking #1



Recall 0.2 0.2 0.4 0.4 0.4 0.6 0.6 0.6 0.8 1.0

Precision 1.0 0.5 0.67 0.5 0.4 0.5 0.43 0.38 0.44 0.5



= relevant documents for query 2

Ranking #2



Recall 0.0 0.33 0.33 0.33 0.67 0.67 1.0 1.0 1.0 1.0

Precision 0.0 0.5 0.33 0.25 0.4 0.33 0.43 0.38 0.33 0.3

$$\text{average precision query 1} = (1.0 + 0.67 + 0.5 + 0.44 + 0.5)/5 = 0.62$$

$$\text{average precision query 2} = (0.5 + 0.4 + 0.43)/3 = 0.44$$

$$\text{mean average precision} = (0.62 + 0.44)/2 = 0.53$$



THANK YOU



BITS Pilani
Pilani Campus

Information Retrieval

Dr. Vijayalakshmi Anand,
BITS -Pilani



Session 10: Web Search

Date – 18th February 2022

Time – 4.15 pm-6.15pm

These slides are prepared by the instructor Prof. Manning,, with grateful acknowledgement of and many others who made their course materials freely available online.

What we will cover in this session

Web Search (T1 Chapter 19)

- Web characteristics
- The search user experience
- Index
- Personalized Search

Web Search

Provide information discovery for large amounts of open access material on the web

Search Engine

- Without search, content is hard to find.
- Without search, there is no incentive to create content.
 - Why publish something if nobody will read it?
 - Why publish something if I don't get ad revenue from it?
- Somebody needs to pay for the web.
 - Servers, web infrastructure, content creation
 - A large part today is paid by search ads.

Interesting web search facts

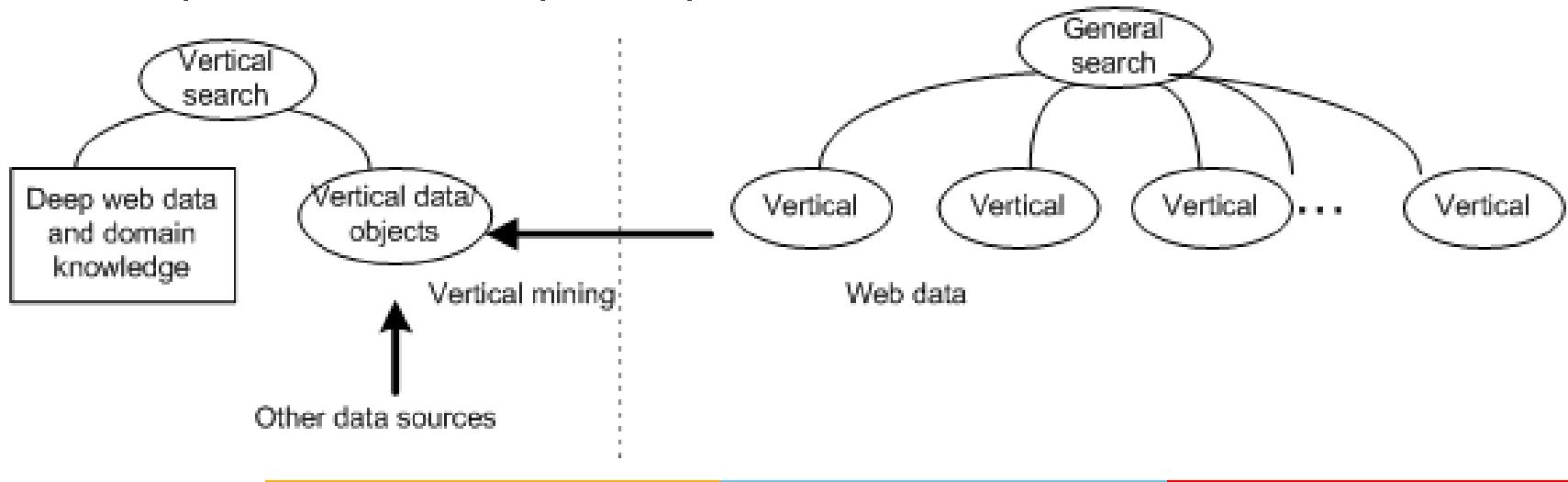
- 3.5 billion Google searches are made every day. ([Internet Live Stats](#))
- Volume of Google searches grows by roughly 10% every year. ([Internet Live Stats](#))
- Every year, somewhere between 16% and 20% of Google searches are new—they've never been searched before. ([Internet Live Stats](#))
- 90% of searches made on desktops are done via Google. ([Statista](#))
- 35% of product searches start on Google. ([eMarketer](#))
- 34% of “near me” searches done via desktop and tablets result in store visits. ([HubSpot](#))
- The average Google search session lasts just under a minute. ([Moz](#))
- Dating & Personal Services advertisers drive the highest CTRs on paid Google results. Just over 6% of their impressions turn into clicks! ([WordStream](#))
- Organic Google results with 3-4 words in the title drive higher CTRs than organic results with 1-2 words in the title. ([Smart Insights](#))
- Google has indexed hundreds of billions of web pages. All told, the index is about 100,000,000 GB large. ([Google](#))

Search Engine : Broad Classes

- **Subject hierarchies or Directory Based Search Engine**
 - e.g. Open Directory Project (ODP), Yahoo!
 - Created and maintained by human editors
- **Web crawling + automatic indexing or Crawler Based Search Engine**
 - Compile their own databases
 - • eg.-- Google, Ask, Exalead, Bing
- **Metasearch**
 - Do not compile their own databases
 - Search databases of different search engines
 - eg. Excite, DogPile

General Search vs. Vertical Search

- **General Search:** identify relevant information with a horizontal/exhaustive view of the world.
- **Vertical Search:**
 - Focus on specific segment of web content
 - Integrate domain knowledge (e.g. taxonomies /ontology), & deep web
 - Examples: travel in Expedia, products in Amazon.



Advanced Search

Advanced Search

Find pages with...

all these words:

Type the important words: tri-colour rat terrier

this exact word or phrase:

Put exact words in quotes: "rat terrier"

any of these words:

Type OR between all the words you want: miniature OR

none of these words:

Put a minus sign just before words that you don't want:
-rodent, -"Jack Russell"

numbers ranging from:

to

Put two full stops between the numbers and add a unit of measurement: 10..35 kg, £300..£500, 2010..2011

Web Challenges for IR

- **Distributed Data:** Documents spread over millions of different web servers.
- **Volatile Data:** Many documents change or disappear rapidly (e.g. dead links).
- **Large Volume:** Billions of separate documents.
- **Unstructured and Redundant Data:** No uniform structure, HTML errors, up to 30% (near) duplicate documents.
- **Quality of Data:** No editorial control, false information, poor quality writing, typos, etc.
- **Heterogeneous Data:** Multiple media types (images, video, VRML), languages, character sets, etc.

Measuring the Web

- As of July 2021, there are more than **1.2 billion hostnames** registered online.
- **1969** is the year when the **first 4 hosts** were registered.
- In 2019, the web hosting industry **market size was valued at \$56.7 billion**.
- The **shared hosting** segment dominated the market, with a **revenue share of 37.64%** in 2019.
- By **2027**, the web hosting industry market size is forecasted to **grow to \$171.4 billion**.

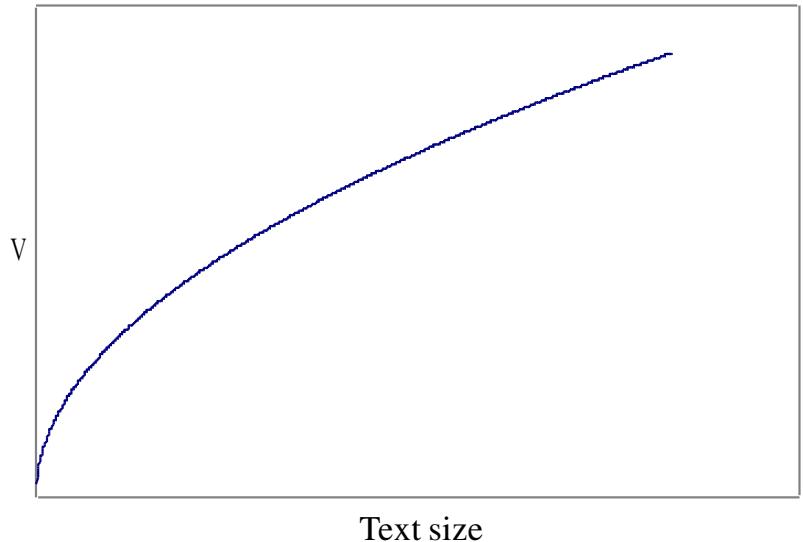
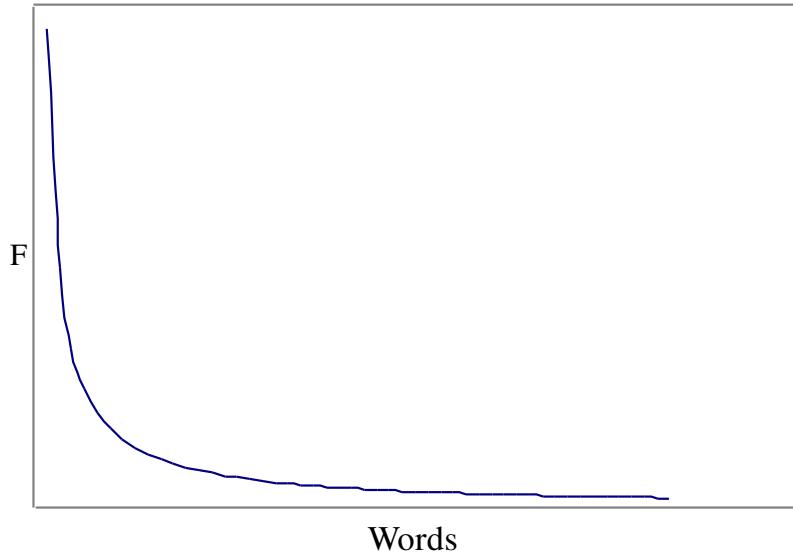
Measuring the Web

- The global web hosting services market is expected to grow annually at a readjusted CAGR(compound annual growth rate) of 18% from 2020 to 2027 due to the increased demand spurred by the COVID-19 pandemic.
 - In total, there are 507,743,033 registered domains today, and the number is increasing at a steady rate (Domain Name Stat, 2021).
 - On average, 900,000 domains are registered every week (HostSorter, 2020).
-

Modeling the Web

- Heaps' and Zipf's laws are also valid in the Web.
 - » In particular, the vocabulary grows faster (larger) and the word distribution should be more biased (larger)
- Heaps' Law
 - » An empirical rule which describes the vocabulary growth as a function of the text size.
 - » It establishes that a text of n words has a vocabulary of size $O(n^\beta)$ for $0 < \beta < 1$
- Zipf's Law
 - » An empirical rule that describes the frequency of the text words.
 - » It states that the i -th most frequent word appears as many times as the most frequent one divided by i^β , for some $\beta > 1$

Zipf's and Heaps' Law



Distribution of sorted word frequencies (left) and size of the vocabulary (right)

User Needs

Informational – want to learn about something (~40%)

Low hemoglobin

Navigational – want to go to that page (~25%)

United Airlines

Transactional – want to do something (~35%)

Access a service

Downloads

Seattle weather

Shop

Mars surface images

Canon S410

Table 3

Definitions of classifications of Web queries

Levels	Examples of queries
<i>Level one</i>	
• (I) Informational: queries meant to obtain data or information in order to address an information need, desire, or curiosity	• Child labor law
• (N) Navigational: queries looking for a specific URL	• Capitalone
• (T) Transactional: queries looking for resources that require another step to be useful	• Buy table clocks
<i>Level two</i>	
• (I, D) Directed: specific question	• Registering domain name
• (I, U) Undirected: tell me everything about a topic	• Singers in the 1980s
• (I, L) List: list of candidates	• Things to do in hollywood ca
• (I, F) Find: locate where some real world service or product can be obtained	• PVC suit for overweight men
• (I, A) Advice: advice, ideas, suggestions, instructions	• What to serve with roast pork tenderloin
• (N, T) Navigation to transactional: the URL the user wants is a transactional site	• match.com
• (N, I) Navigation to informational: the URL the user wants is an informational site	• yahoo.com
• (T, O) Obtain: obtain a specific resource or object	• Music lyrics
• (T, D) Download: find a file to download	• mp3 downloads
• (T, R) Results page: obtain a resource that one can printed, save, or read from the search engine results page	• (The user enters a query with the expectation that 'answer' will be on the search engine results page and not require browsing to another Website)
• (T, I) Interact: interact with program/resource on another Website	• Buy table clock
<i>Level three</i>	
• (I,D, C) Closed: deals with one topic; question with one, unambiguous answer	• Nine supreme court justices
• (I,D, O) Open: deals with two or more topics	• The excretory system of arachnids
• (T, O, O) Online: the resource will be obtained online	• Airline seat map
• (T, O, F) Off-line: the resource will be obtained off-line and may require additional actions by the user	• Full metal alchemist wallpapers
• (T, D, F) Free: the downloadable file is free	• Free online games
• (T, D, N) Not free: the downloadable file is not necessarily free	• Family guy episode download
• (T, R, L) Links: the resources appears in the title, summary, or URL of one or more of the results on the search engine results page	• (As an example, a user enters the title of a conference paper in order to locate the page numbers, which usually appear in one or more of the results)
• (T, R, O) Other: the resources does not appear one of the results but somewhere else on the search engine results page	• (As an example, a user enters a query term to check for spelling with no interest in the results listing)

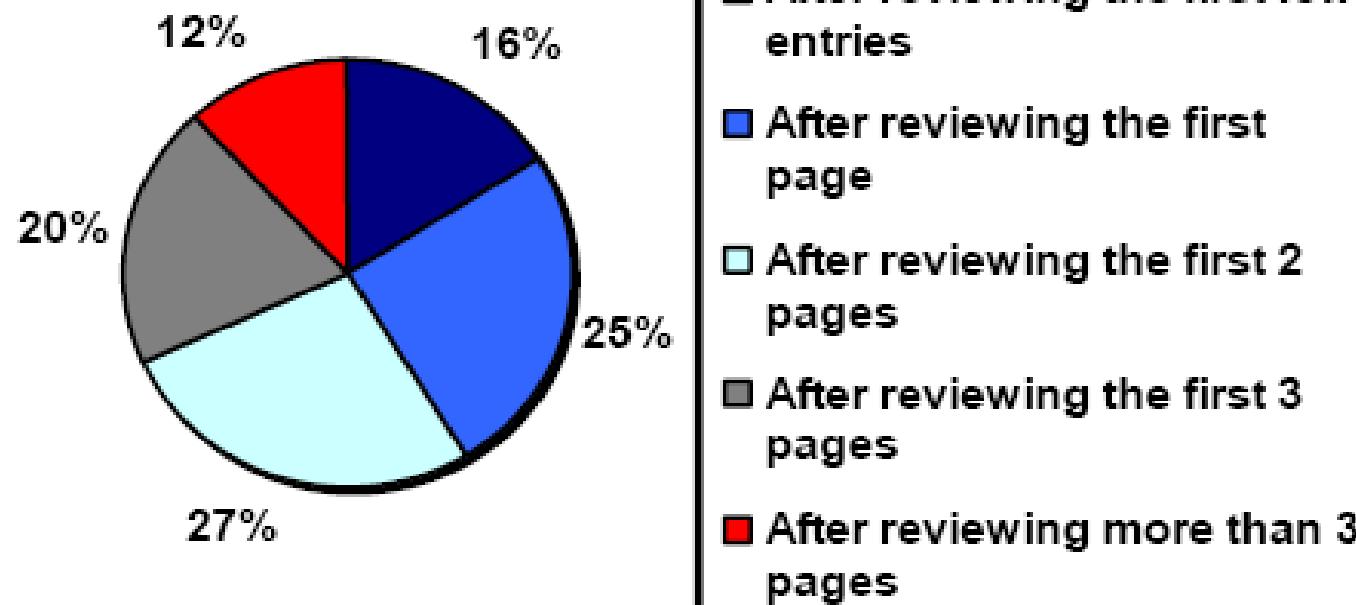


Essential Characteristics for user-friendliness

- Mobile Compatibility. ...
 - Accessible to All Users. ...
 - Well Planned Information Architecture. ...
 - Well-Formatted Content That Is Easy to Scan. ...
 - Fast Load Times. ...
 - Browser Consistency. ...
 - Effective Navigation. ...
 - Good Error Handling
 - Contrasting Color Scheme
 - Usable forms
-

How far do people look for results?

"When you perform a search on a search engine and don't find what you are looking for, at what point do you typically either revise your search, or move on to another search engine? (Select one)"



(Source: iprospect.com/WhitePaper_2006_SearchEngineUserBehavior.pdf)

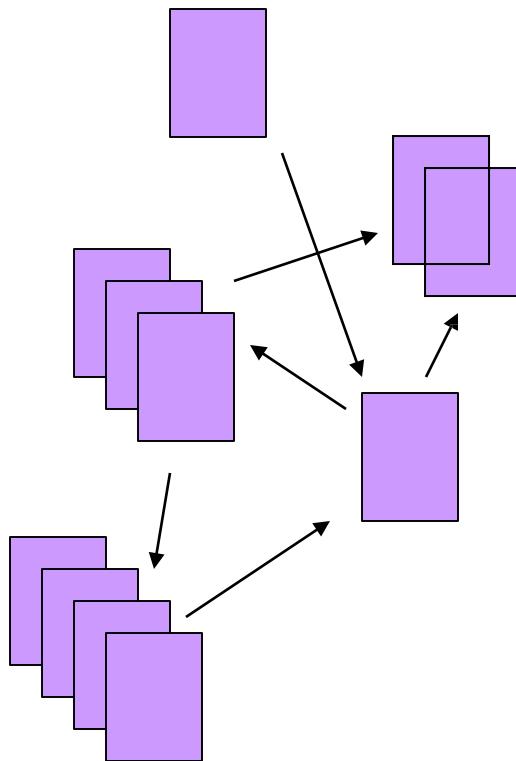
Users' empirical evaluation of results

- Quality of pages varies widely
 - Relevance is not enough
 - Other desirable qualities (non IR!!)
 - Content: Trustworthy, diverse, non-duplicated, well maintained
 - Web readability: display correctly & fast
 - No annoyances: pop-ups, etc.
- Precision vs. recall
 - What matters
 - On the web, recall seldom matters
 - Precision at 1? Precision above the fold?
 - Comprehensiveness – must be able to deal with obscure queries
 - Recall matters when the number of matches is very small

Users' empirical evaluation of engines

- Relevance and validity of results
- UI – Simple, no clutter, error tolerant
- Trust – Results are objective
- Coverage of topics for polysemic queries
- Pre/Post process tools provided
 - Mitigate user errors (auto spell check, search assist,...)
 - Explicit: Search within results, more like this, refine
 - ...
 - Anticipative: related searches

The Web document collection



- No design/co-ordination
- Content includes truth, lies, obsolete information, contradictions ...
- Unstructured (text, html, ...), semi-structured (XML, annotated photos), structured (Databases)...
- Scale much larger than previous text collections ... but corporate records are catching up
- Growth – “volume doubling every few months”
- Content can be *dynamically generated*

The Web

Relevance Judgment

- Web search engines measure relevance using human judges
- Each result to each query is judged on a scale. For example:
 - Perfect, the ideal result for this query
 - Relevant, that is, meets the information need
 - Irrelevant, that, does not meet the information need
- The judgments are used to compute various metrics that measure *recall* and *precision*

Recall and Precision

- *Recall* is the fraction of relevant documents retrieved from all relevant documents in the collection
- *Precision* is the fraction of retrieved documents that are relevant

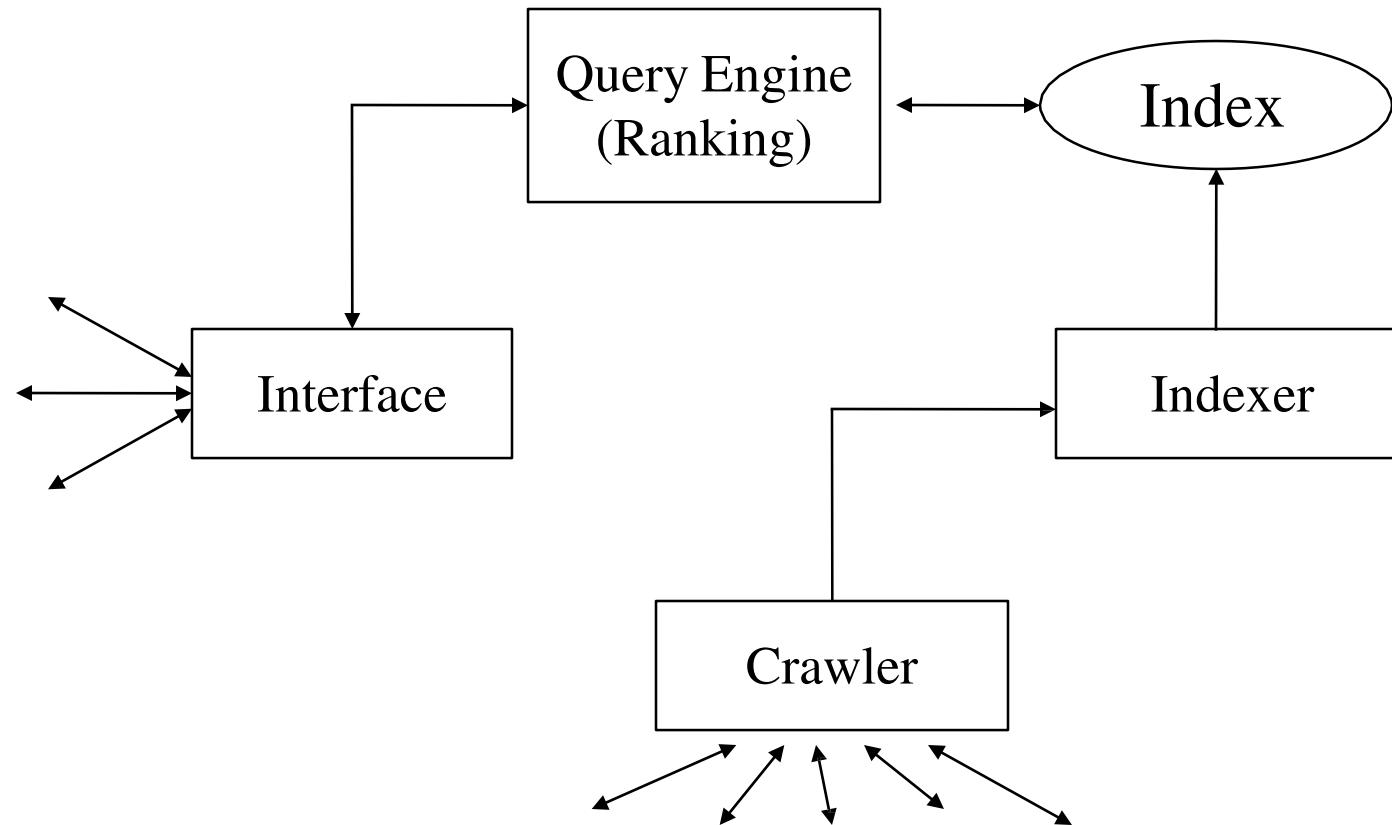
Recall and Precision

- In practice, recall is hard to measure:
 - Requires relevance judgment of all documents in the collection to each query
 - Impractical for large collections
 - Typically ignored by web search engines
- Precision is much easier to measure
- Precision may fluctuate, but typically decreases with the number of results inspected

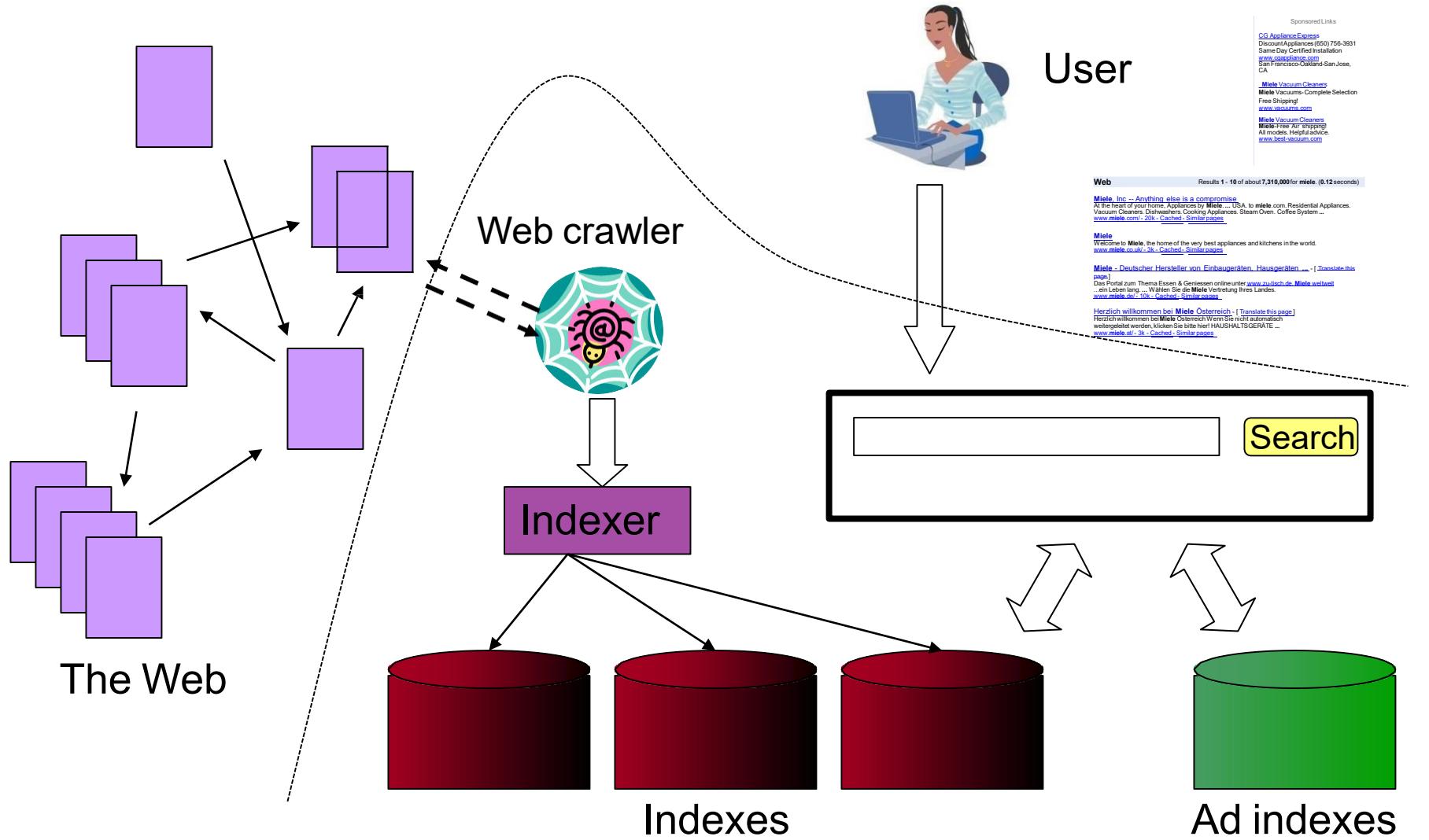
Search Engines

- Centralized Architecture
- Distributed Architecture
- User Interface
 - ❖ Ranking
 - ❖ Crawling the Web
 - ❖ Indices

Typical Crawler-Indexer Architecture



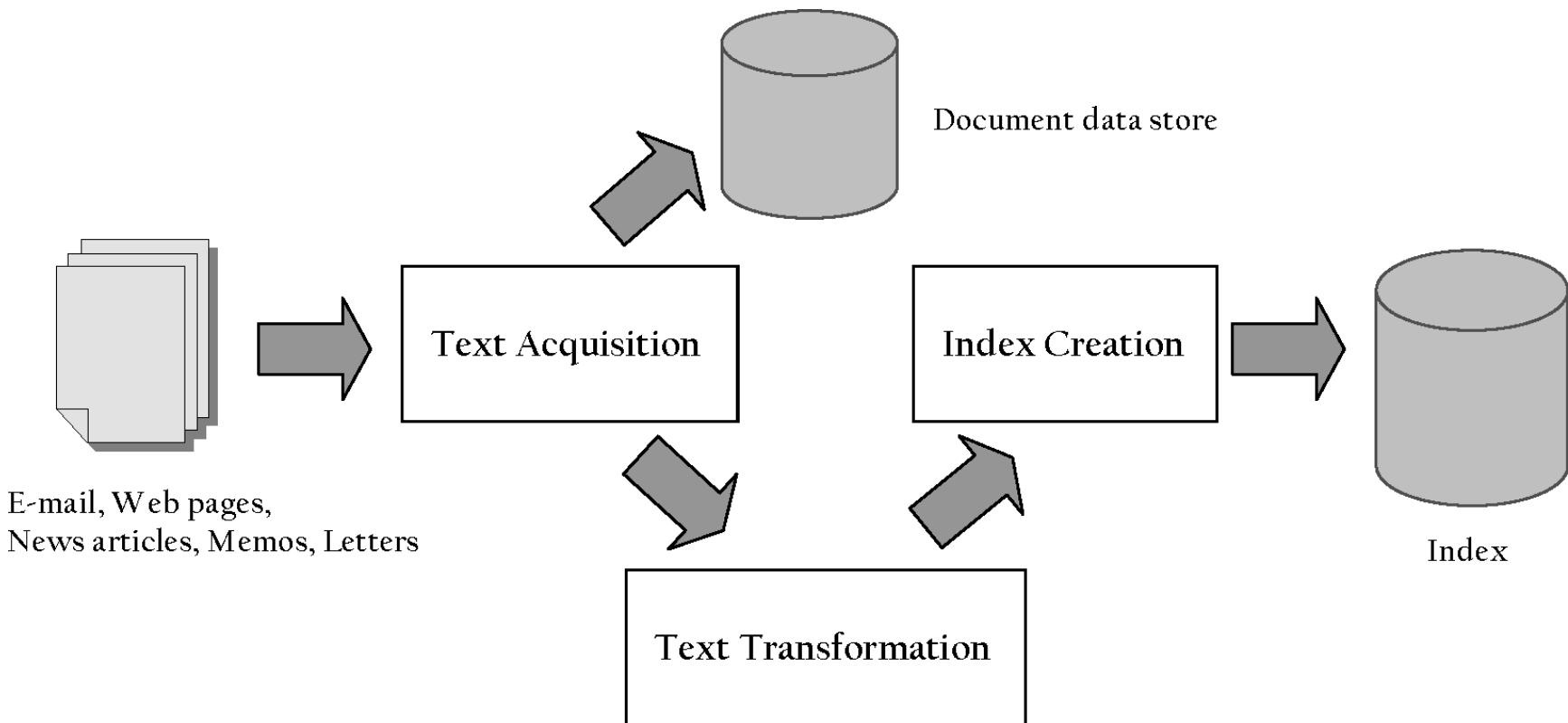
Web search basics



Search engine architecture: key pieces

- Spider (a.k.a. crawler/robot) – builds corpus
 - Collects web pages recursively
 - For each known URL, fetch the page, parse it, and extract new URLs
 - Repeat
 - Additional pages from direct submissions & other sources
- Indexer – creates inverted indexes so online system can search
- Online query process– serves query results
 - Front end – query reformulation, word processing
 - Back end – finds matching documents and ranks them

Indexing Process



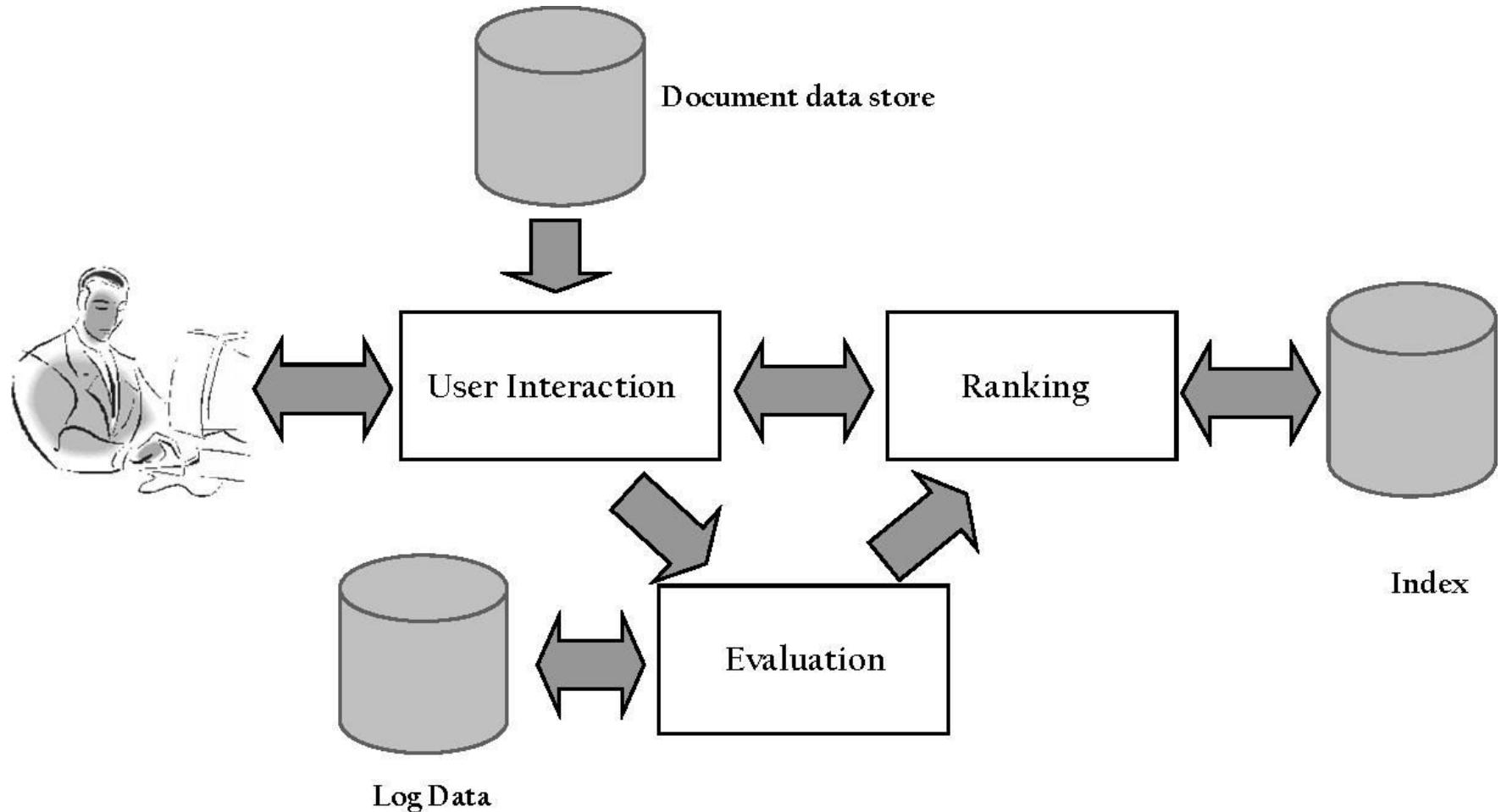
Indexing Process

- Text acquisition
 - identifies and stores documents for indexing
- Text transformation
 - transforms documents into *index terms* or *features*
- Index creation
 - takes index terms and creates data structures (*indexes*) to support fast searching

Choosing Pages for Indexing

- In practice, there are two solutions to choosing pages for the index:
 - In real time, make a yes/no decision about each page, and add to the index
 - Store the pages, and process them offline to construct an index
- The former solution is typically based on the well-known AltaVista “chunk” solution
 - Create a buffer of documents (a “chunk”)
 - Build an index on that buffer
 - Move the index and content to an index serving node
 - (After some time) Mark the chunk’s URLs for refetch
 - (After some time) Expire the chunk
- The latter approach is likely what’s used at Google:
 - Store multiple copies of the web in a document store
 - Iterate over the document store (potentially multiple times) to choose documents
 - Create an index, and ship it to the index serving nodes
 - Repeat

Query Process



Query Process

User interaction

- supports creation and refinement of query, display of results

Ranking

- uses query and indexes to generate ranked list of documents

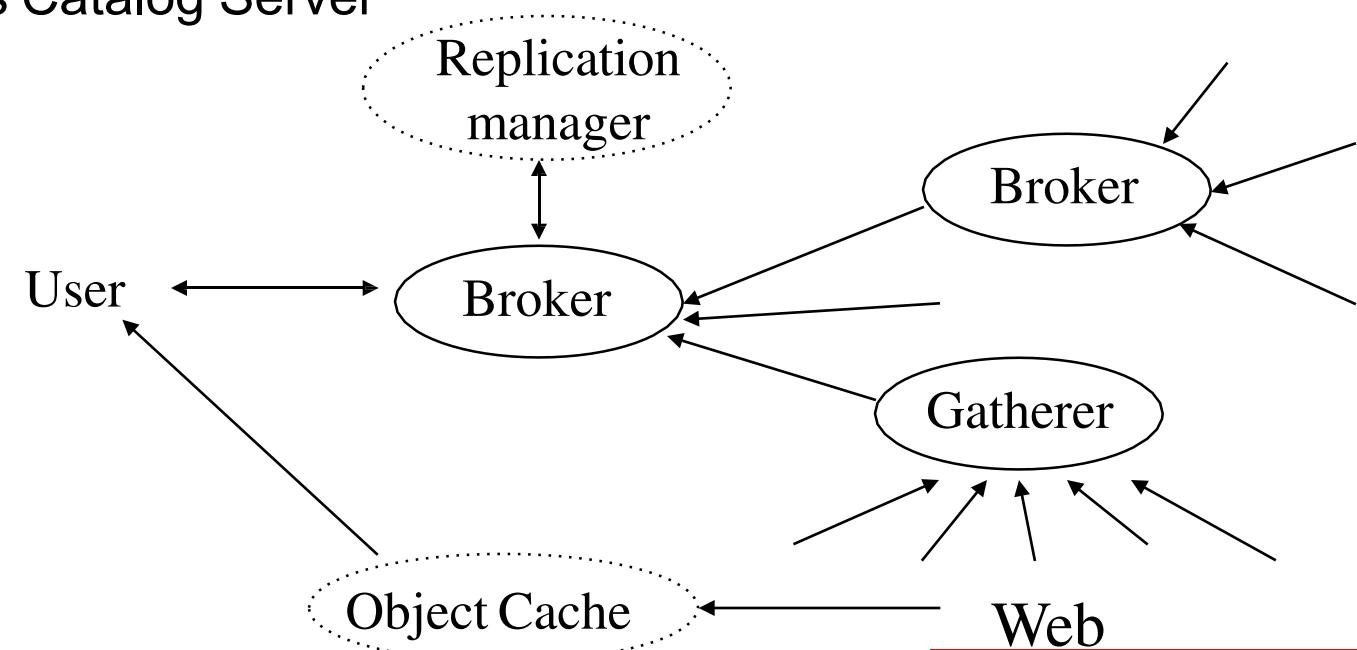
Evaluation

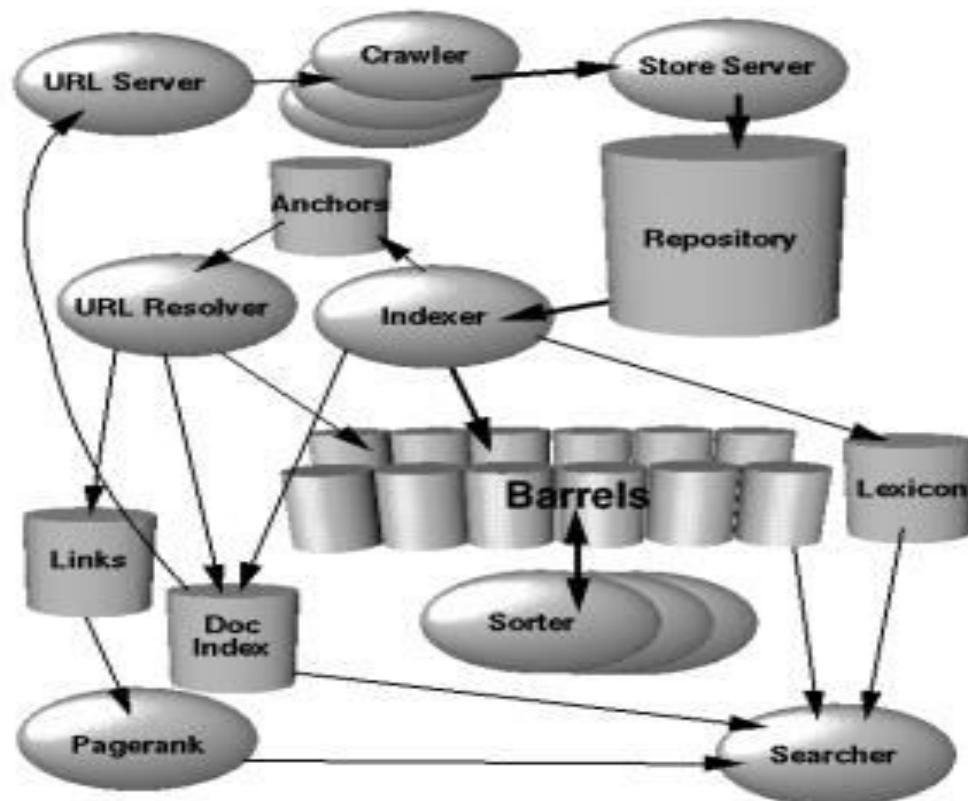
- monitors and measures effectiveness and efficiency (primarily offline)

Distributed Architecture

- Harvest

- » Gatherers: collect and extract indexing information from one or more Web servers
- » Brokers: provide the indexing mechanism and the query interface to the data gathered
- » Netscape's Catalog Server





User Interface

- Query interface
 - » AltaVista: OR
 - » HotBot: AND
- Answer interface
 - » order by relevance
 - » order by Url or date
 - » option: find documents similar to each Web page

Ranking

- Most search engines follow traditional
 - » Boolean or Vector Model
- Hyperlink Information
 - » HITS (Kleinsberg, (SIAM98))
 - » ARC (Cha98, WWW7)
 - » PageRank, Google (BP98, WWW7)

Crawling the Web

- Synonyms
 - » spider, robot, crawler, etc.
 - » Starting from a set of popular URLs
 - » Partition the Web using country codes or Internet names
- Crawling order
 - » Depth-first, breadth-first
 - » CG98, WWW7
- robot.txt
 - » Guidelines for robot behavior includes what pages should not be indexed
 - » e.g. dynamically generated pages, password protected pages

Indices

- Variants of Inverted file
 - » A short description of each Web page is complemented
 - creation data, size, the title and the first lines or a few headings
 - 500bytes for each page*100million pages=50GB
 - » 30% of the text size
 - 5KB for each page*100million pages*30%=150GB
 - » compression
 - 50GB
- Binary Search on the sorted list of words of the inverted file

Browsing in Web Directories

Search Engine	URL	Web sites	Categories
eBLAST	www.eblast.com	125	-
LookSmart	www.looksmart.com	300	24
Lycos Subjects	a2z.lycos.com	50	-
Magellan	ww.mckinley.com	60	-
NewHoo	www.newhoo.com	100	23
Netscape	www.netscape.com	-	-
Search.com	www.search.com	-	-
Snap	www.snap.com	-	-
Yahoo	www.yahoo.com	750	-

Meta Search Engine

- A meta search engine is **a specialized form of search engine that aggregates results from the data of other search engines**
- Examples:
 - » WebCompass
 - » WebSeeker
 - » EchoSearch
 - » WebFerret
 - » Inquirus (LG98, WWW7)

Personalization

- Ambiguity means that a single ranking is unlikely to be optimal for all users
- Personalized ranking is the only way to bridge the gap
- Personalization can use
 - Long term behavior to identify user interests, e.g., a long term interest in user interface research
 - Short term session to identify current task, e.g., checking on a series of stock tickers
 - User location, e.g., MTA in New York vs Baltimore
 - Social network
 - ...

Potential for Personalization

[Teevan, Dumais, Horvitz 2010]

How much can personalization improve ranking?

How can we measure this?

Ask raters to explicitly rate a set of queries

But rather than asking them to guess what a user's information need might be ...

... ask which results *they would personally consider relevant*

Use self-generated and pre-generated queries

Computing potential for personalization

For each query q

Compute average rating for each result

Let R_q be the optimal ranking according to the average rating

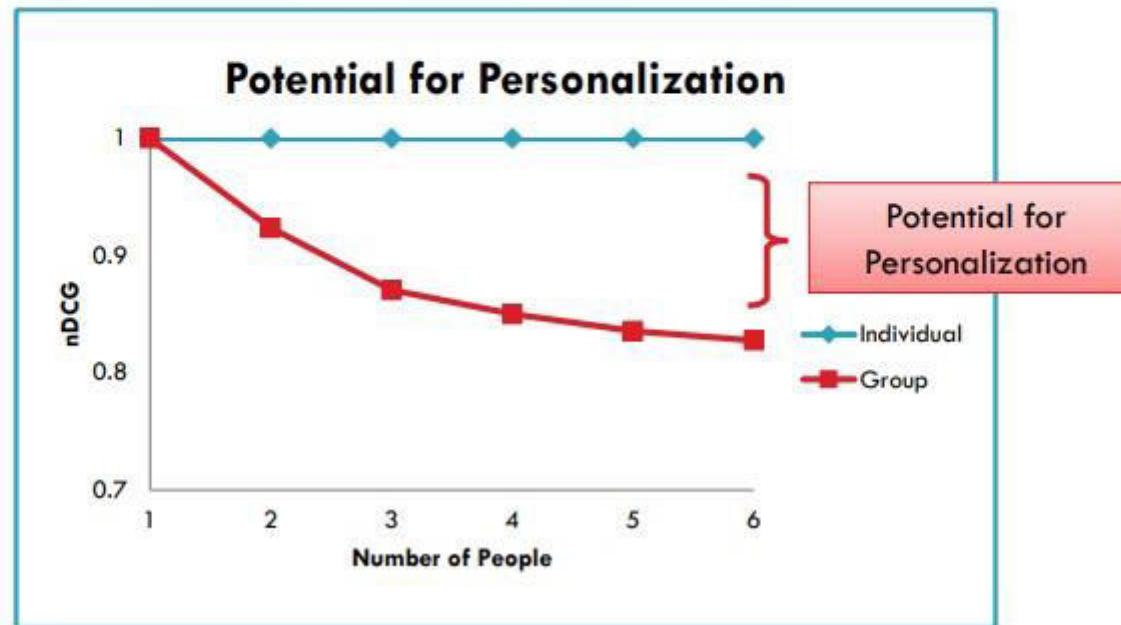
Compute the NDCG value of ranking R_q for the ratings of each rater i

Let Avg_q be the average of the NDCG values for each rater

Let Avg be the average Avg_q over all queries
Potential for personalization is $(1 - \text{Avg})$

Personalization in Ranking

- A single ranking for everyone limits search quality
- Quantify the variation in relevance for the same query across different individuals



Potential for Personalization

- Quantify the variation in relevance for the same query across different individuals
- Different ways to measure individual relevance
 - Explicit judgments from different people for the same query
 - Implicit judgments from click entropy or content analysis
- Personalization can lead to large improvements
 - Study with explicit judgments
 - 46% improvements for core ranking
 - 70% improvements with personalization

https://web.stanford.edu/class/cs276/19handouts/guest_lecture-SDumais-May2019

Potential for Personalization

- ❑ Not all queries have high potential for personalization
 - ❑ E.g., new york times vs. sigir
 - ❑ E.g., * maps
- ❑ Learn when to personalize

- ❑ Query: Stanford IR course
- ❑ What is the “potential for personalization”?



https://web.stanford.edu/class/cs276/19handouts/guest_lecture-SDumais-May2019

How can you identify different intents?

- **Past behavior**
 - Current session,
 - Longer history of actions and
 - preferences
- **Contextual metadata**
 - Location,
 - Time,
 - Device, etc

https://web.stanford.edu/class/cs276/19handouts/guest_lecture-SDumais-May2019

User Models

- Constructing user models
 - Sources of evidence
 - Content: Queries, content of web pages, desktop index, etc.
 - Behavior: Explicit feedback, implicit feedback, visited web pages
 - Context: Location, time (of day/week/year), device, etc.
 - Time frames: Short-term, long-term
 - Who: Individual, group
- Using user models
 - Where resides: Client, server
 - How used: Ranking, query support, presentation, etc.
 - When used: Always, sometimes, context learned

https://web.stanford.edu/class/cs276/19handouts/guest_lecture-SDumais-May2019

Personal Navigation

- Re-finding is common in Web search
 - 33% of queries are repeat queries
 - 39% of clicks are repeat clicks
- Many of these are navigational queries
 - E.g., *new york times* -> www.nytimes.com
 - Consistent intent across individuals
 - Identified via low click entropy, anchor text
- “Personal navigational” queries
 - Different intents across individuals ... but consistently the same intent for an individual
 - SIGIR (for Dumais) -> www.sigir.org
 - SIGIR (for Bowen Jr.) -> www.sigir.mil

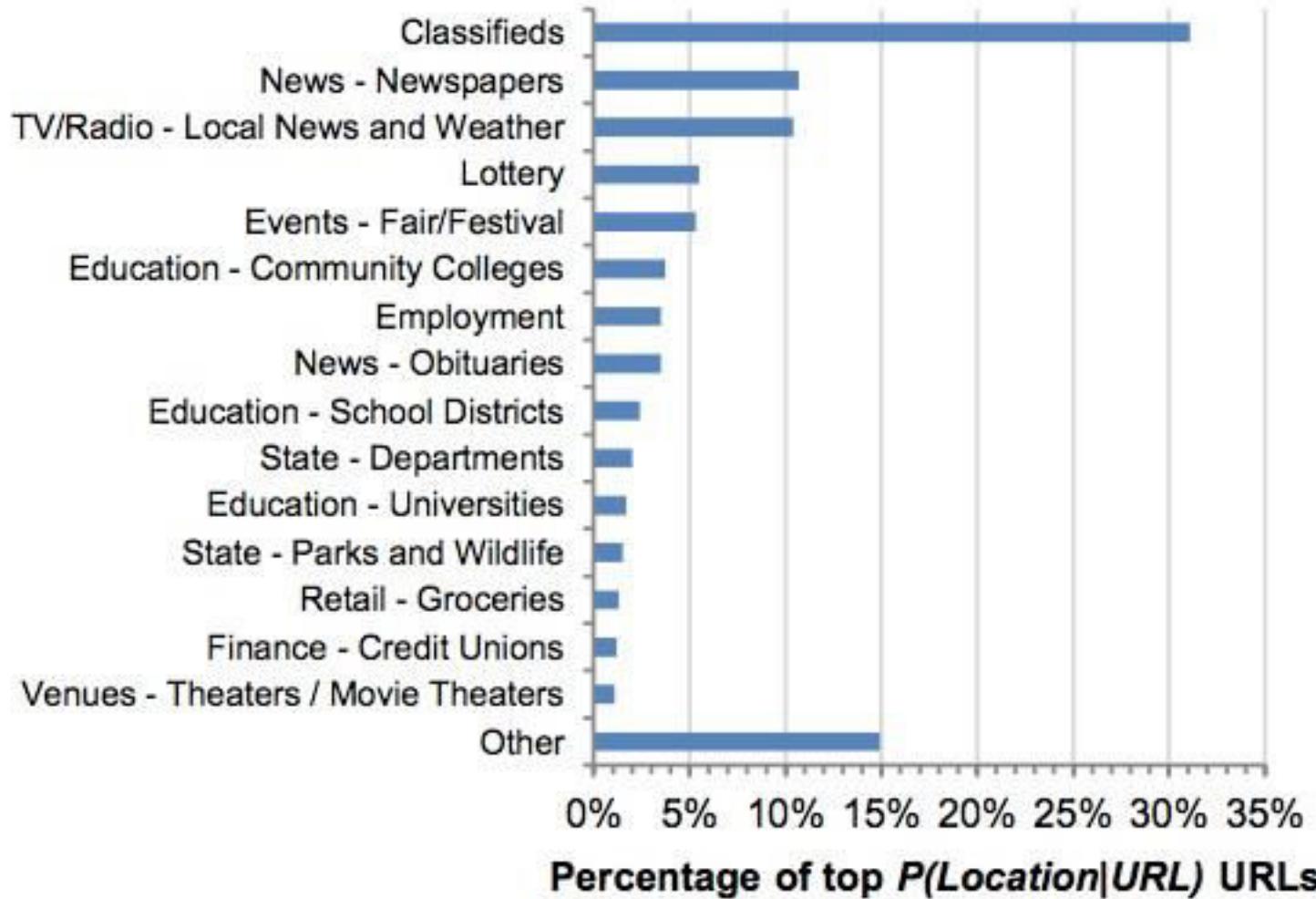
		Repeat Click	New Click
Repeat Query	33%	29%	4%
New Query	67%	10%	57%
	39%		61%



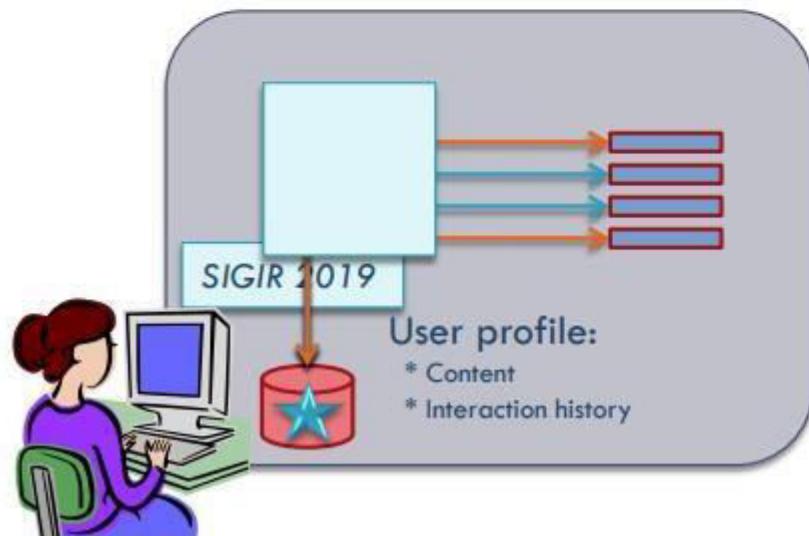
User location

- . User location is one of the most important features for personalization
 - . Country
 - . Query [football] in the US vs the UK
 - . State/Metro/City
 - . Queries like [zoo]
 - . Fine-grained location
 - . Queries like [pizza], [restaurants], [coffee shops]

Topics in URLs with high $P(\text{user location} \mid \text{URL})$



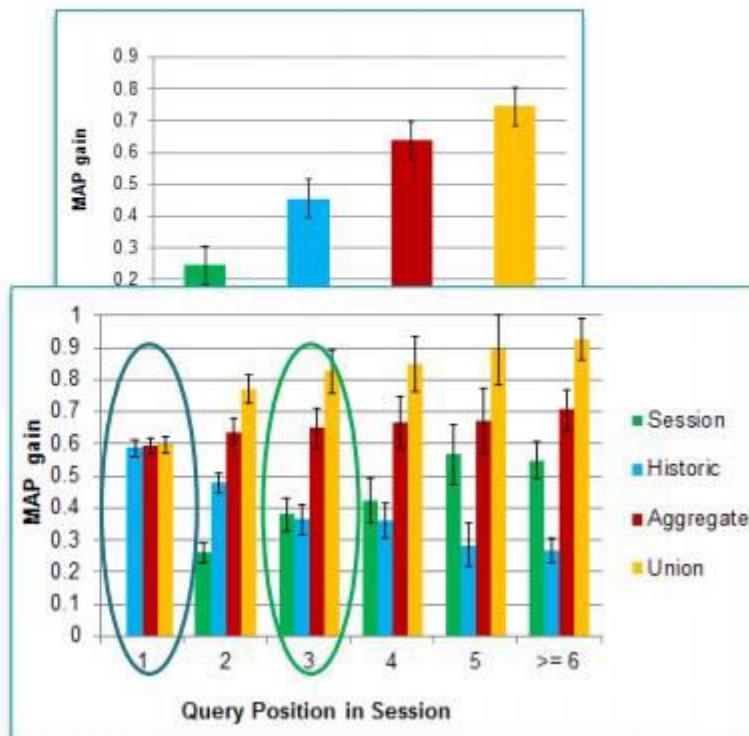
- Rich client-side model of a person's interests
 - Model: Content from desktop search index & Interaction history
Rich and constantly evolving user model
 - Client-side re-ranking of web search results using model
 - Good privacy (only the query is sent to server)
 - But, limited portability, and use of community



Short + Long term context



- User model (temporal extent)
 - Session, Historical, Combinations
 - Temporal weighting
- Large-scale log analysis
- Which sources are important?
 - Session (short-term): +25%
 - Historic (long-term): +45%
 - Combinations: +65-75%
- What happens within a session?
 - 1st query, can only use historical
 - By 3rd query, short-term features more important than long-term



Temporal Dynamics

- Queries are not uniformly distributed over time

- Often triggered by events in the world

- What's relevant changes over time

- E.g., US Open ... in 2019 vs. in 2018

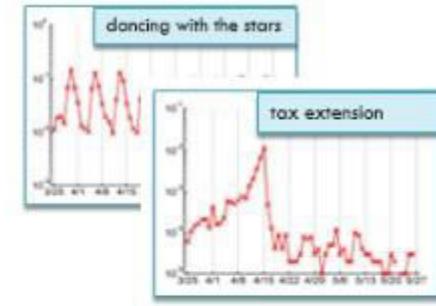
- E.g., US Open 2019 ... in May (golf) vs. in Sept (tennis)

- E.g., US Tennis Open 2019 ...

- Before event: Schedules and tickets, e.g., stubhub

- During event: Real-time scores or broadcast, e.g., espn

- After event: General sites, e.g., wikipedia, usta



Challenges in Personalization

User-centered

- Privacy
- Serendipity and novelty
- Transparency and control

Systems-centered

- Optimization
 - Storage, run-time, caching, etc.
- Evaluation
 - Measurement, experimentation

Conclusion

- “Too many attempts have been made without sufficient regard to what people really want, what they can use, and how best it should fit their needs.”
- “A major challenge to large-scale personalization is to lower the entry bar, making it easier for less-experienced users to customize their pages, and making it clear to novices that customization is possible.”

References

- [J. Teevan, S. Dumais, E. Horvitz. Potential for personalization. 2010](#)
- [J. Pitkow et al. Personalized search. 2002](#)
- [J. Teevan, S. Dumais, E. Horvitz. Personalizing search via automated analysis of interests and activities. 2005](#)
- [P. Bennett et al. Inferring and using location metadata to personalize Web search. 2011](#)
- [T. Haveliwala. Topic-sensitive pagerank. 2002.](#)
- [G. Jeh and J. Widom. Scaling personalized Web search. 2003](#)
- [M. Curtiss et al. Unicorn: A system for searching the social graph. 2013](#)
- https://web.stanford.edu/class/cs276/19handouts/guest_lecture-SDumais-May2019
- https://www.youtube.com/watch?v=Wt_pPv8udcU
- Introduction to Information Retrieval by Chris Manning and Pandu Nayak

Personalizing search

Two general ways of personalizing search

1. **Query expansion**

Modify or augment user query

E.g., query term “IR” can be augmented with either
“information retrieval” or “Ingersoll-Rand” depending on
user interest

Ensures that there are enough personalized results

2. **Reranking**

Issue the same query and fetch the same results ...

... but rerank the results based on a user profile

Allows both personalized and globally relevant results

[Pitkow et al. 2002]

Query augmentation and Reranking

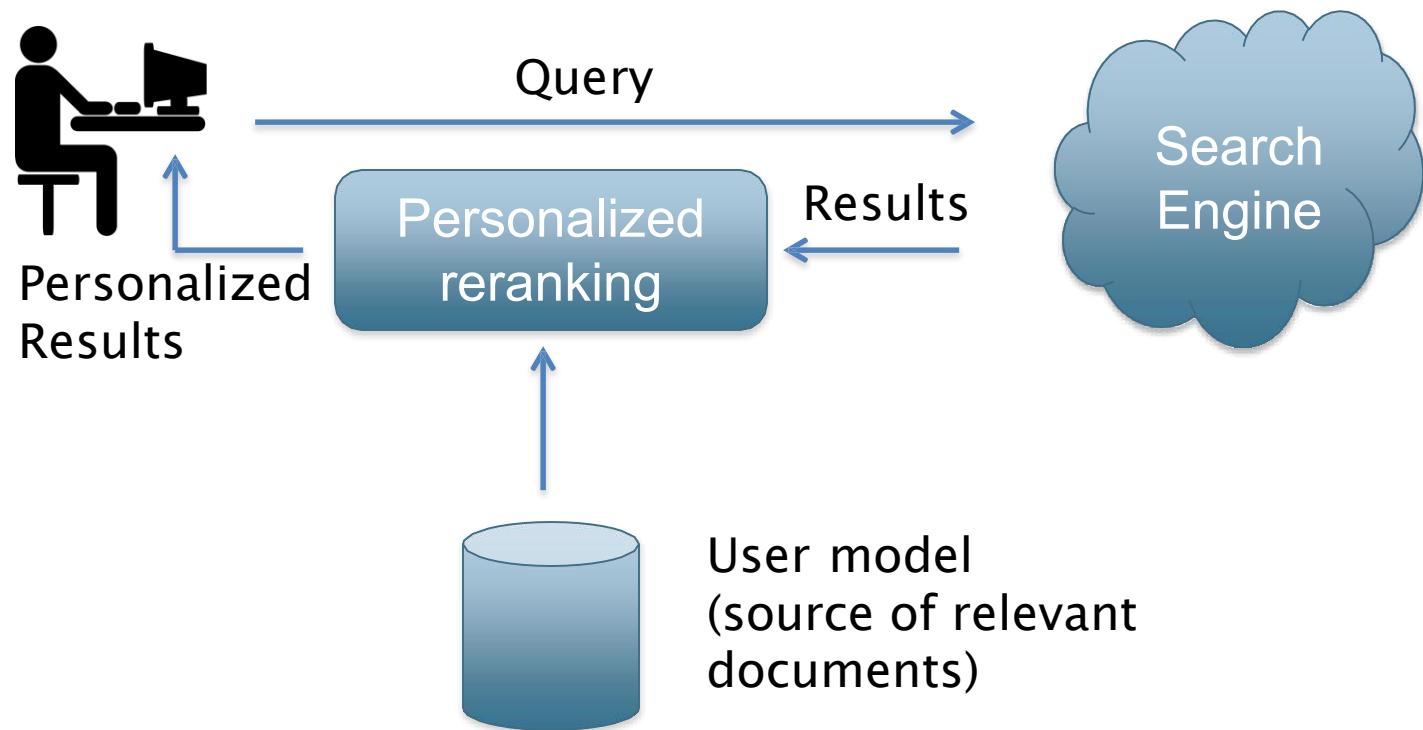
- Include other terms
- Computing the similarity between the query term and the user model
 - if the query is on a topic the user has previously seen, the system can reinforce the query with similar terms
- Concise query is then shown to the user and “submitted to a search engine for processing”
- Query augmented and processed by the search engine, the results can be “individualized”
- Information is filtered based upon information in the user’s model and/or context
- Re-rank search results based upon the similarity of the content of the pages in the results and the user’s profile”

User interests

- Explicitly provided by the user
 - Sometimes useful, particularly for new users
 - ... but generally doesn't work well
- Inferred from user behavior and content
 - Previously issued search queries
 - Previously visited Web pages
 - Personal documents
 - Emails
- Ensuring privacy and user control is very important

Relevance feedback perspective

[Teevan, Dumais, Horvitz 2005]



Personalization helps Query Understanding

- Queries are difficult to interpret in isolation



- Easier if we can model: who is asking, what they have done in the past, where they are, when it is, etc.



Searcher: (SIGIR | Susan Dumais ... an information retrieval researcher)

vs. (SIGIR | Stuart Bowen Jr. ... the Special Inspector General for Iraq Reconstruction)

Previous actions: (SIGIR | information retrieval)

vs. (SIGIR | U.S. coalitional provisional authority)



Location: (SIGIR | at SIGIR conference) vs. (SIGIR | in Washington DC)

Time: (SIGIR | Jan. submission) vs. (SIGIR | Aug. conference)

- Using a single ranking for everyone, in every context, at every point in time, limits how well a search engine can do



THANK YOU



BITS Pilani
Pilani Campus

Information Retrieval

Dr. Vijayalakshmi
BITS Pilani



Session 11: Web Crawlers

These slides are prepared by the instructor Prof. Manning,, with grateful acknowledgement of and many others who made their course materials freely available online.

What we will cover in this session

Web Crawlers(T1 Chapter 20)

- Crawling
- Crawler Architecture
- Distributed Indexes (Near) duplicate detection

Web Crawler

- A Web Crawler is a software for downloading pages from the Web
- Also known as Web Spider, Web Robot, or simply Bot

Brief History - Web Crawler

- The first known web crawler was created in 1993 by Matthew Gray, an undergraduate student at MIT
- On June of that year, Gray sent the following message to the `www-talk` mailing list:

"I have written a perl script that wanders the WWW collecting URLs, keeping tracking of where it's been and new hosts that it finds. Eventually, after hacking up the code to return some slightly more useful information (currently it just returns URLs), I will produce a searchable index of this."

- The project of this Web crawler was called WWWW (World Wide Web Wanderer)
 - It was used mostly for Web characterization studies
-

Brief History - Web Crawler

- On June 1994, Brian Pinkerton, a PhD student at the University of Washington, posted the following message to the `comp.infosystems.announce` news

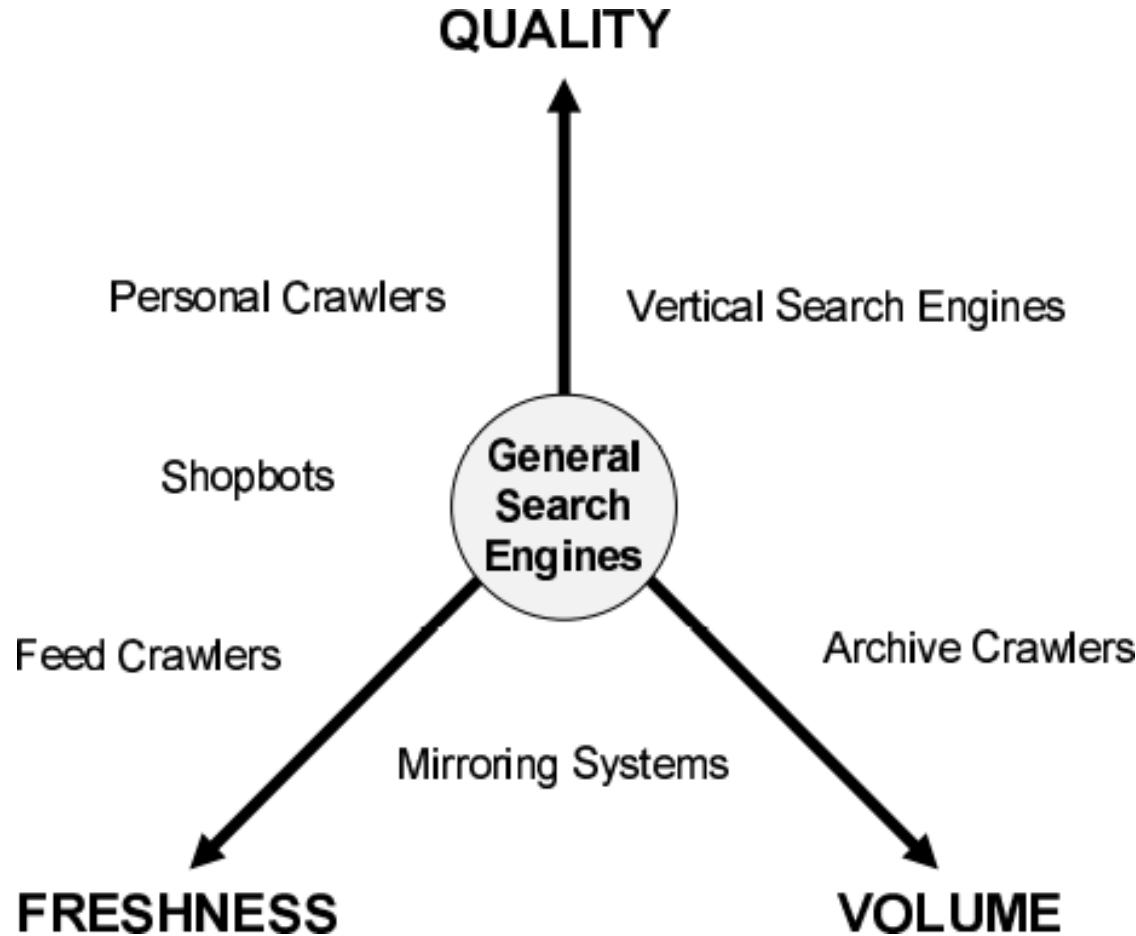
"The WebCrawler Index is now available for searching! The index is broad: it contains information from as many different servers as possible. It's a great tool for locating several different starting points for exploring by hand. The current index is based on the contents of documents located on nearly 4000 servers, world-wide."

- The WebCrawler become a commercial success
 - Other search engines based on Web crawlers appeared: Lycos (1994), Excite (1995), Altavista (1995), and Hotbot (1996)
-

Web Crawler Applications

- Create an index covering broad topics (General Web search)
- Create an index covering specific topics (Vertical Web search)
- Archive content
(Web archival, URL: <http://www.archive.org/>)
- Analyze Web sites for extracting aggregate statistics (Web characterization)
- Keep copies or replicate Web sites
(Web mirroring-daily or weekly)
- Web site analysis (broken links, site not available)

Crawler Taxonomy

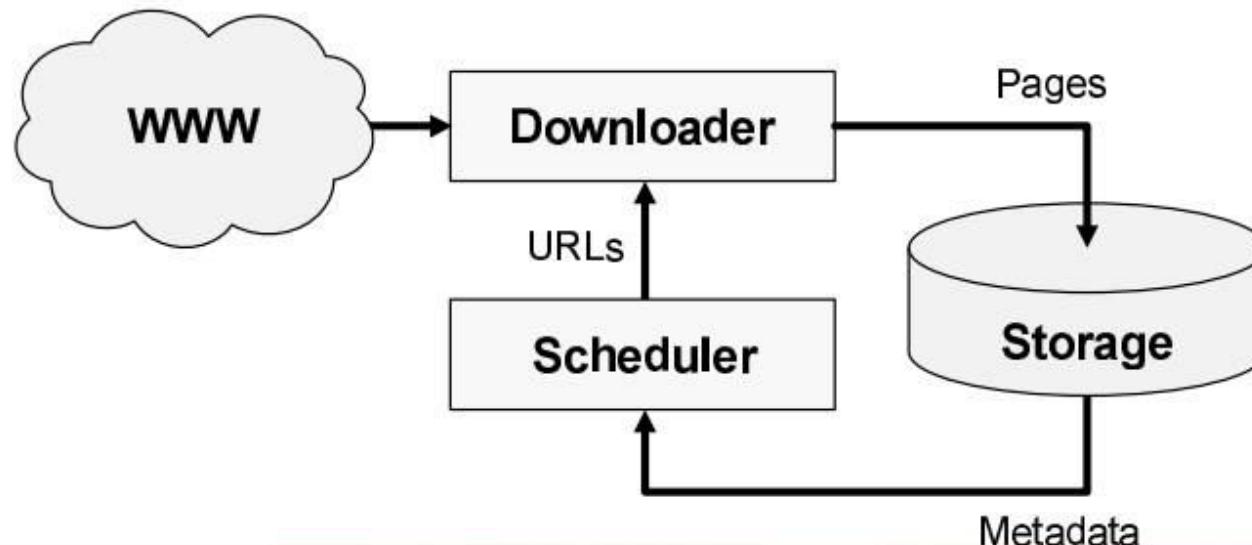


Basic crawler operation

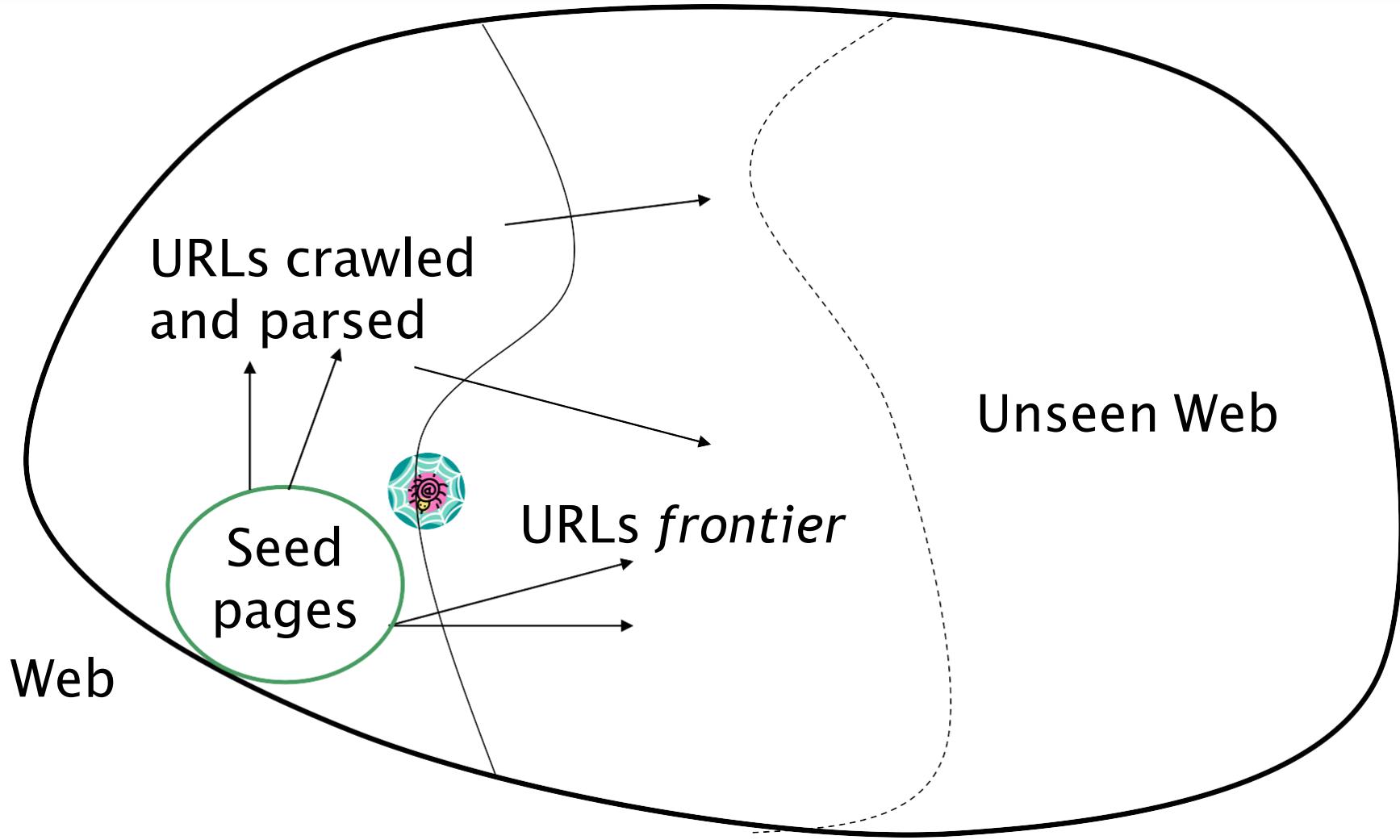
- . Begin with known “seed” URLs
- . Fetch and parse them
 - . Extract URLs they point to
 - . Place the extracted URLs on a queue
- . Fetch each URL on the queue and repeat

Basic Crawler Architecture

- The crawler is composed of three main modules: downloader, storage, and scheduler
 - **Scheduler:** maintains a queue of URLs to visit
 - **Downloader:** downloads the pages
 - **Storage:** makes the indexing of the pages, and provides the scheduler with metadata on the pages retrieved



Crawling picture



Crawler practical issues

- The implementation of a crawler involves many **practical issues**
- Most of them is due to the need to interact with many different systems
- Example: How to download maintaining the traffic produced as uniform as possible?
 - The pages are from multiple sources
 - DNS and Web server response times are highly variable
 - Web server up-time cannot be taken for granted
- Other practical issues are related with: types of Web pages, URL canonization, parsing, wrong implementation of HTML standards, and duplicates

Simple picture – complications

- Web crawling isn't feasible with one machine
 - All of the above steps distributed
- Malicious pages
 - Spam pages
 - Spider traps – incl dynamically generated
- Even non-malicious pages pose challenges
 - Latency/bandwidth to remote servers vary
 - Webmasters' stipulations
 - How “deep” should you crawl a site's URL hierarchy?
 - Site mirrors and duplicate pages
 - Politeness – don't hit a server too often

What any crawler *must* do

- Be Robust: Be immune to spider traps and other malicious behavior from web servers
- Be Polite: Respect implicit and explicit politeness considerations

Explicit and implicit politeness

- Explicit politeness: specifications from webmasters on what portions of site can be crawled
 - robots.txt
- Implicit politeness: even with no specification, avoid hitting any site too often

Robots.txt

- Protocol for giving spiders (“robots”) limited access to a website, originally from 1994
 - www.robotstxt.org/robotstxt.html
- Website announces its request on what can(not) be crawled
 - For a server, create a file /robots.txt
 - This file specifies access restrictions

Robots.txt example

- No robot should visit any URL starting with "/yoursite/temp/", except the robot called "searchengine":

```
User-agent: *
Disallow: /yoursite/temp/
```

```
User-agent: searchengine
Disallow:
```

← → C



stanford.edu/robots.txt

User-agent: *

Disallow: /wp-admin/

Allow: /wp-admin/admin-ajax.php

Sitemap: <https://www.stanford.edu/wp-sitemap.xml>

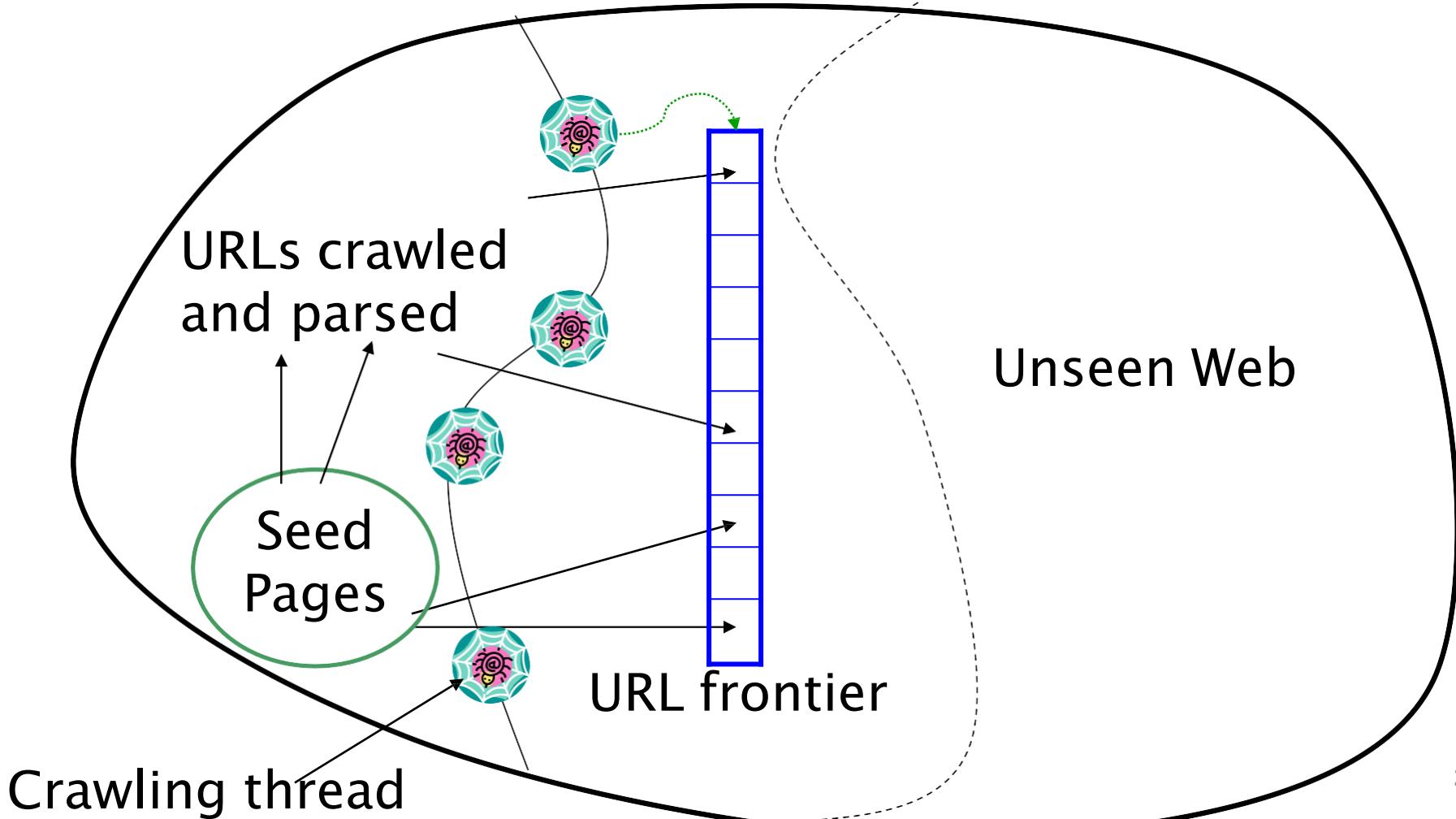
What any crawler *should* do

- Be capable of distributed operation: designed to run on multiple distributed machines
- Be scalable: designed to increase the crawl rate by adding more machines
- Performance/efficiency: permit full use of available processing and network resources

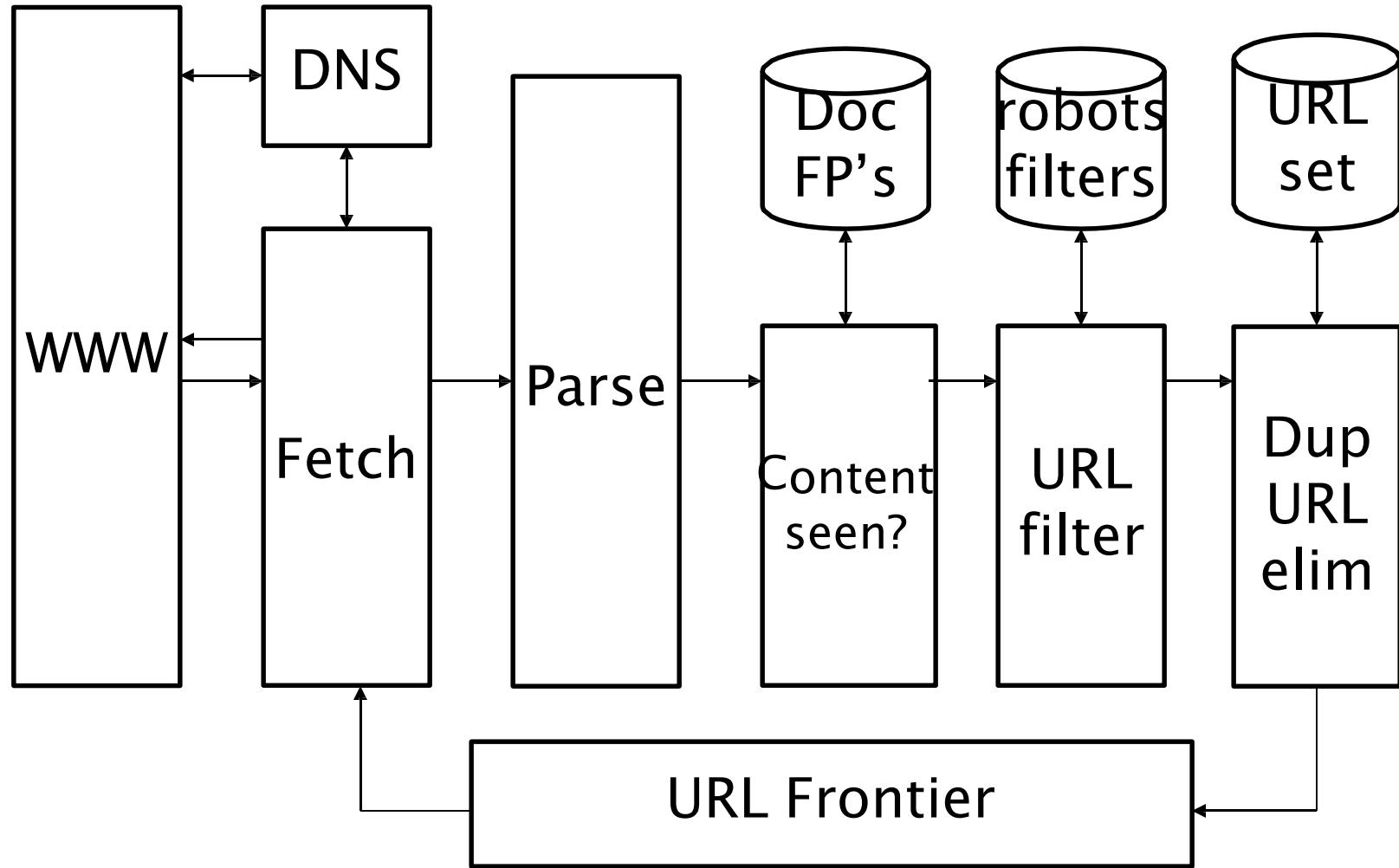
What any crawler *should* do

- Fetch pages of “higher quality” first
- Continuous operation: Continue fetching fresh copies of a previously fetched page
- Extensible: Adapt to new data formats, protocols

Updated crawling picture



Basic crawl architecture



URL frontier

- The URL frontier is the data structure that holds and manages URLs we've seen, but that have not been crawled yet.
- Can include multiple pages from the same host
- Must avoid trying to fetch them all at the same time
- Must keep all crawling threads busy

DNS (Domain Name Server)

- A lookup service on the internet
 - Given a URL, retrieve its IP address
 - Service provided by a distributed set of servers – thus, lookup latencies can be high (even seconds)
- Common OS implementations of DNS lookup are *blocking*: only one outstanding request at a time
- Solutions
 - DNS caching
 - Batch DNS resolver – collects requests and sends them out together

Parsing: URL normalization

- When a fetched document is parsed, some of the extracted links are *relative URLs*
- E.g., http://en.wikipedia.org/wiki/Main_Page has a relative link to /wiki/Wikipedia:General_disclaimer which is the same as the absolute URL
http://en.wikipedia.org/wiki/Wikipedia:General_disclaimer
- During parsing, must normalize (expand) such relative URLs

Content seen?

- . Duplication is widespread on the web
- . If the page just fetched is already in the index, do not further process it
- . This is verified using document fingerprints or shingles
 - . Second part of this lecture

Filters and robots.txt

- Filters – regular expressions for URLs to be crawled/not
- Once a robots.txt file is fetched from a site, need not fetch it repeatedly
 - Doing so burns bandwidth, hits web server
- Cache robots.txt files

Duplicate URL elimination

- For a non-continuous (one-shot) crawl, test to see if an extracted+filtered URL has already been passed to the frontier
- For a continuous crawl – frontier implementation

Distributing the crawler

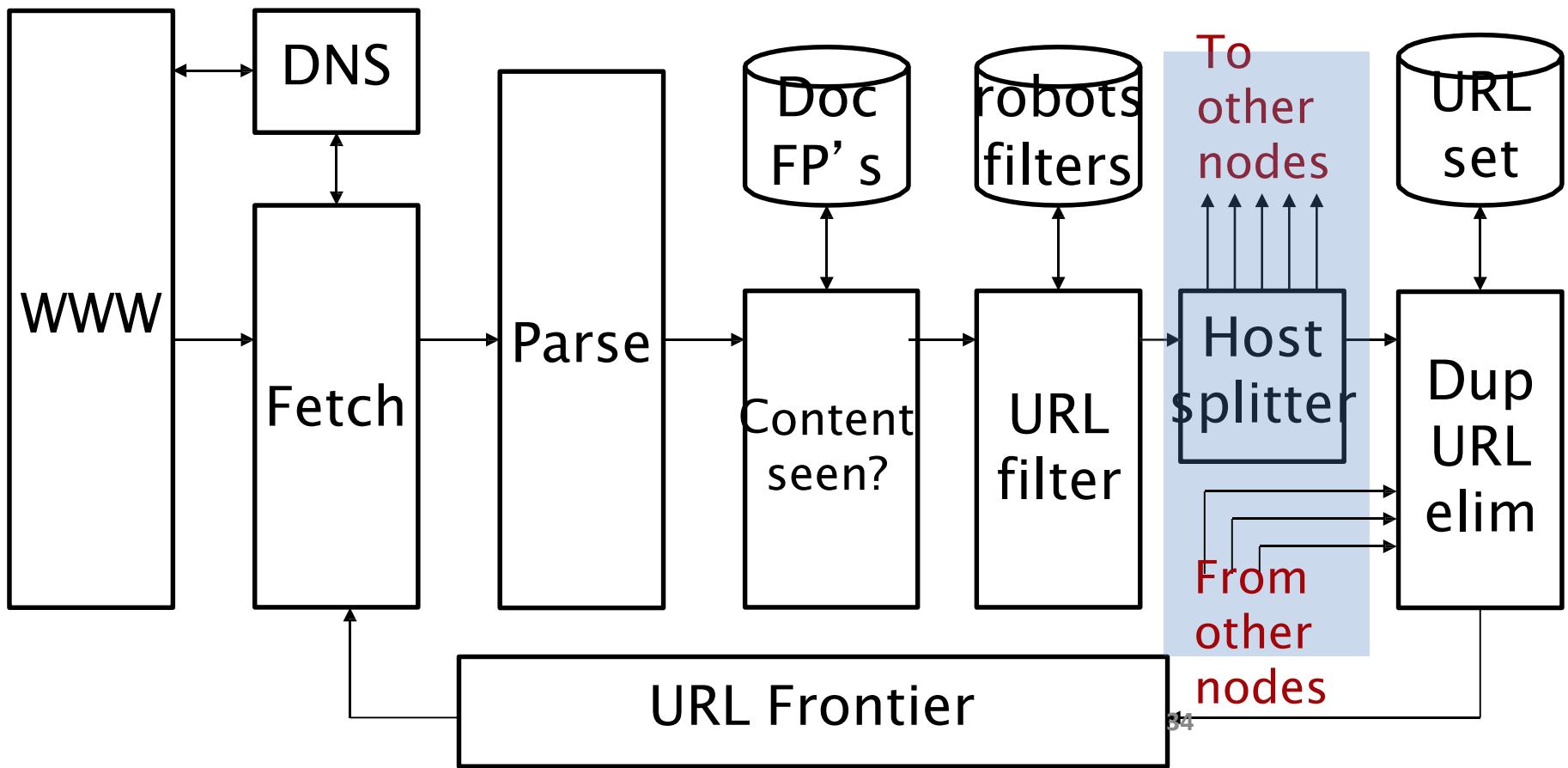
- Run multiple crawl threads, under different processes – potentially at different nodes
 - Geographically distributed nodes
- Partition hosts being crawled into nodes
 - Hash used for partition
- How do these nodes communicate and share URLs?

Google data centers (wazfaring. com)



Communication between nodes

- Output of the URL filter at each node is sent to the Dup URL Eliminator of the appropriate node



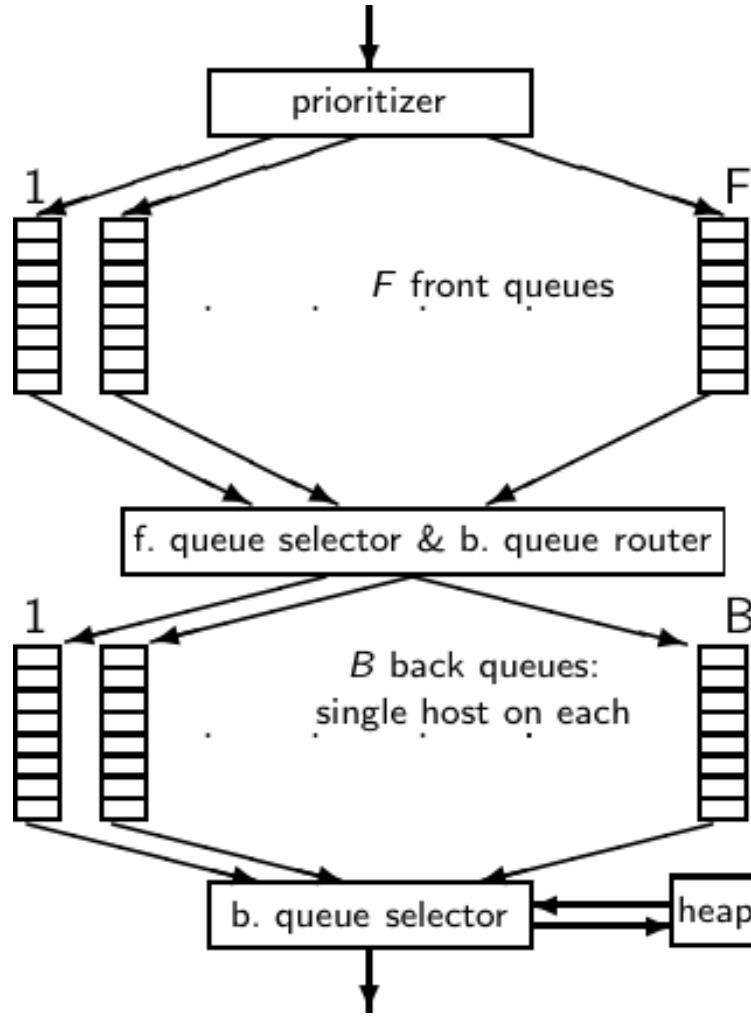
URL frontier: two main considerations

- Politeness: do not hit a web server too frequently
- Freshness: crawl some pages more often than others
 - E.g., pages (such as News sites) whose content changes often
- These goals may conflict with each other.
- (E.g., simple priority queue fails – many links out of a page go to its own site, creating a burst of accesses to that site.)

Politeness – challenges

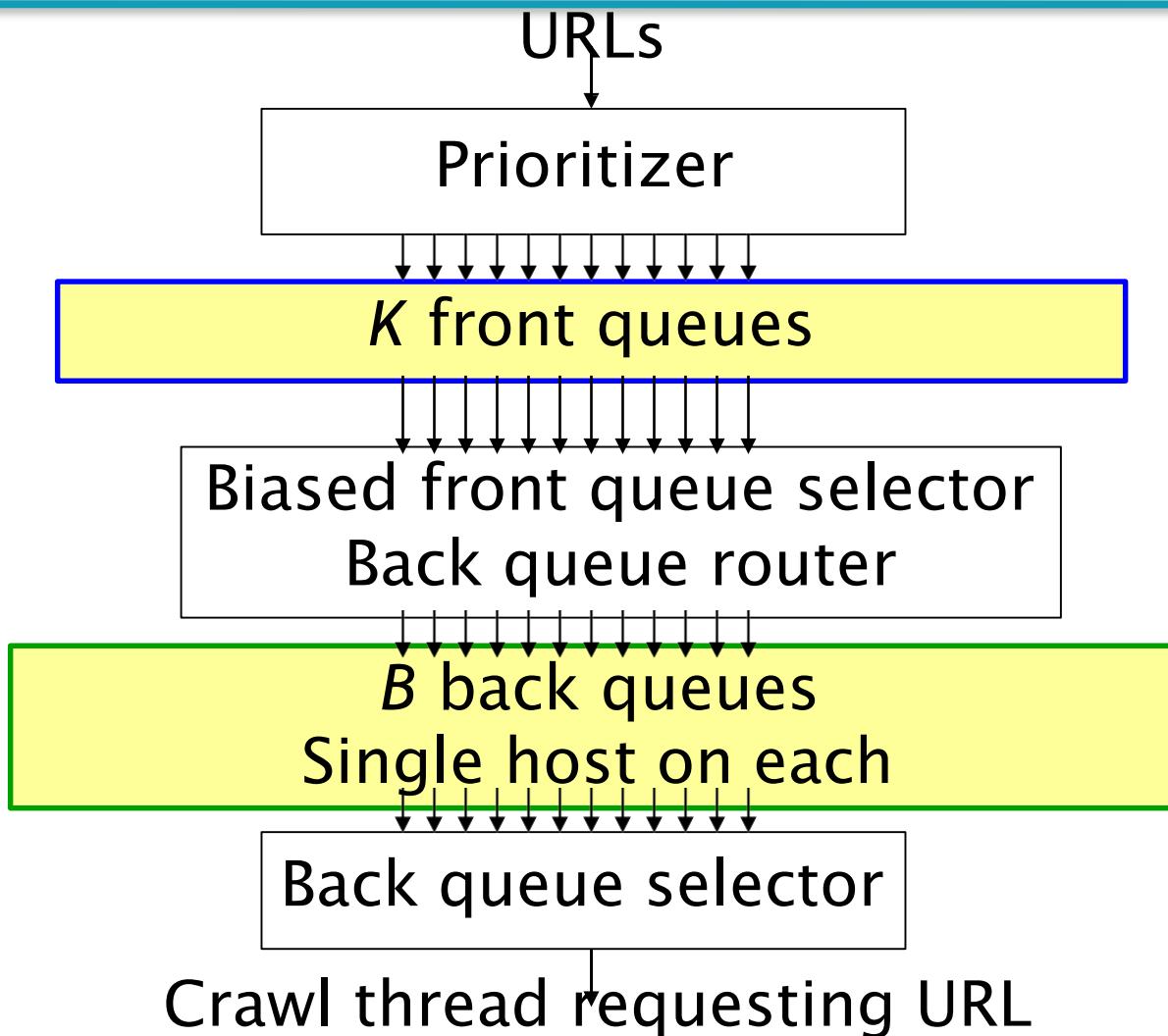
- Even if we restrict only one thread to fetch from a host, can hit it repeatedly
- Common heuristic: insert time gap between successive requests to a host that is \gg time for most recent fetch from that host

Mercator URL frontier

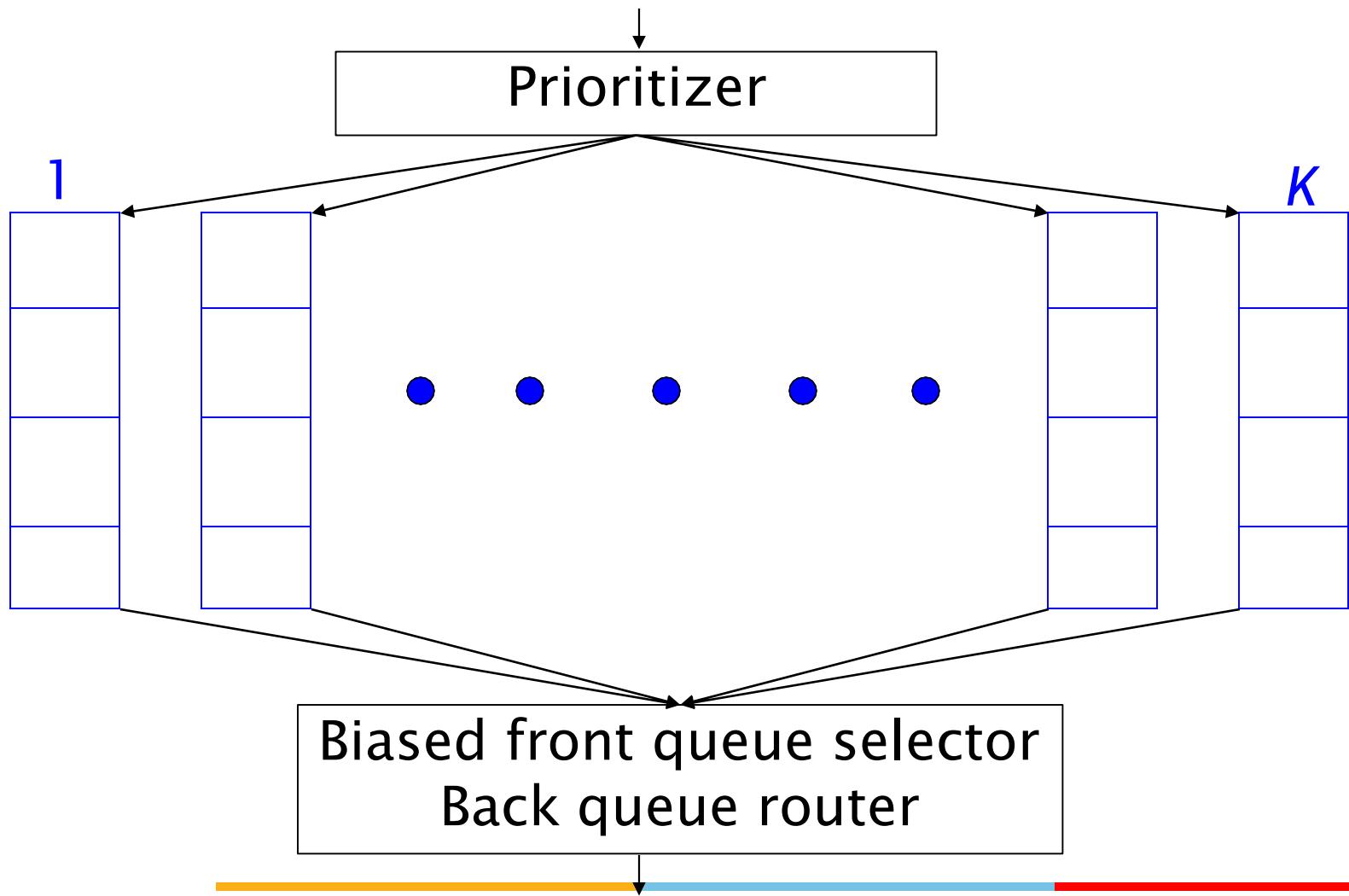


- URLs flow in from the top into the frontier.
- Front queues manage prioritization.
- Back queues enforce politeness.
- Each queue is FIFO.

URL frontier: Mercator scheme



Front queues



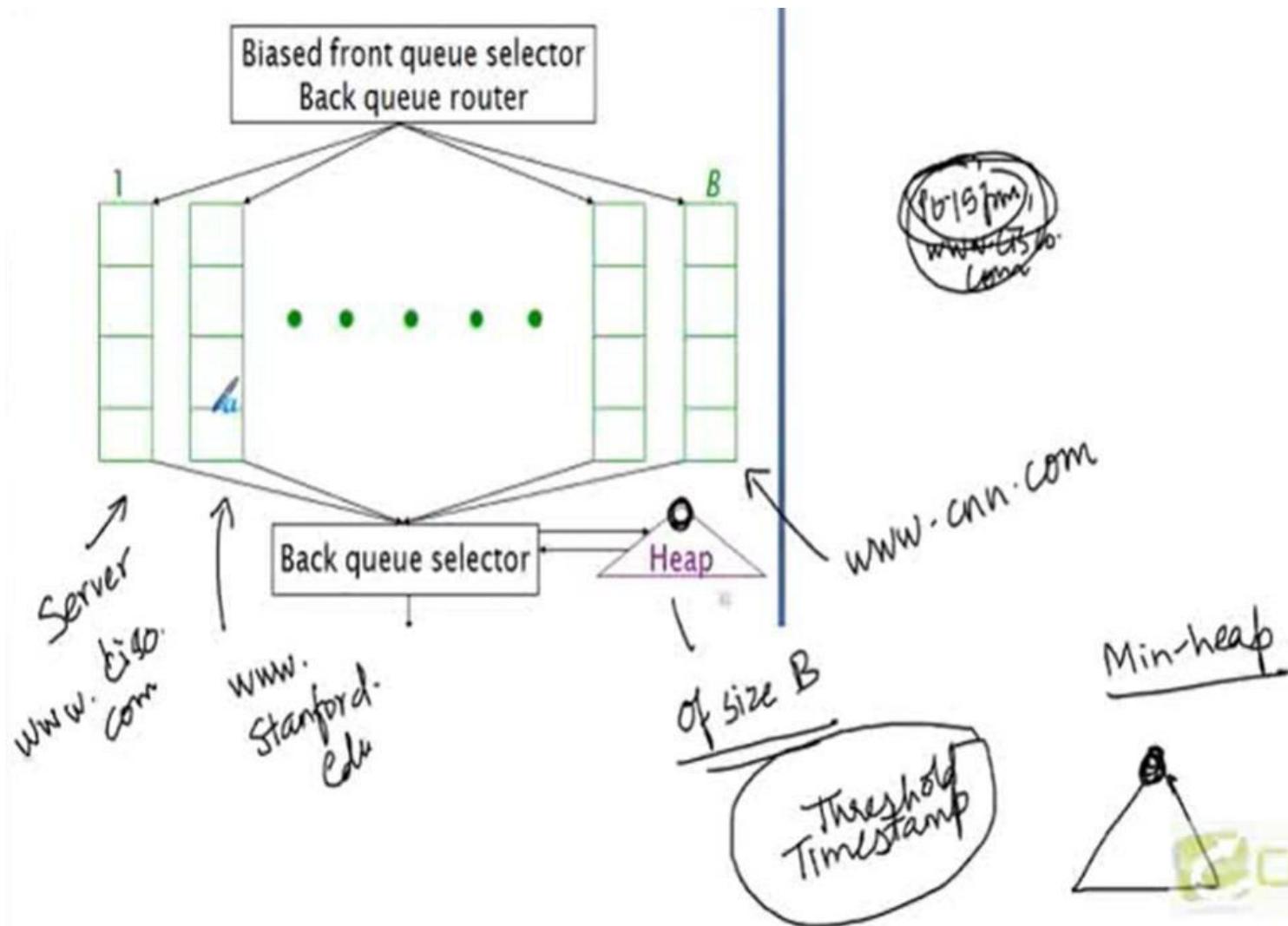
Front queues

- Prioritizer assigns to URL an integer priority between 1 and K
 - Appends URL to corresponding queue
- Heuristics for assigning priority
 - Refresh rate sampled from previous crawls
 - Application-specific (e.g., “crawl news sites more often”)

Biased front queue selector

- When a back queue requests a URL (in a sequence to be described): picks a front queue from which to pull a URL
- This choice can be round robin biased to queues of higher priority, or some more sophisticated variant
 - Can be randomized

Back queues



Back queue invariants

- Each back queue is kept non-empty while the crawl is in progress
- Each back queue only contains URLs from a single host
 - Maintain a table from hosts to back queues

Host name	Back queue
...	3
	1
	B

Back queue heap

- One entry for each back queue
- The entry is the earliest time t_e at which the host corresponding to the back queue can be hit again
- This earliest time is determined from
 - Last access to that host
 - Any time buffer heuristic we choose

Back queue processing

- A crawler thread seeking a URL to crawl:
- Extracts the root of the heap
- Fetches URL at head of corresponding back queue q (look up from table)
- Checks if queue q is now empty – if so, pulls a URL v from front queues
 - If there's already a back queue for v 's host, append v to it and pull another URL from front queues, repeat
 - Else add v to q
- When q is non-empty, create heap entry for it

Number of back queues B

- Keep all threads busy while respecting politeness
- Mercator recommendation: three times as many back queues as crawler threads

Duplicate documents

- The web is full of duplicated content
- Strict duplicate detection = exact match
 - Not as common
- But many, many cases of near duplicates
 - E.g., Last modified date the only difference between two copies of a page

Duplicate/Near-Duplicate Detection

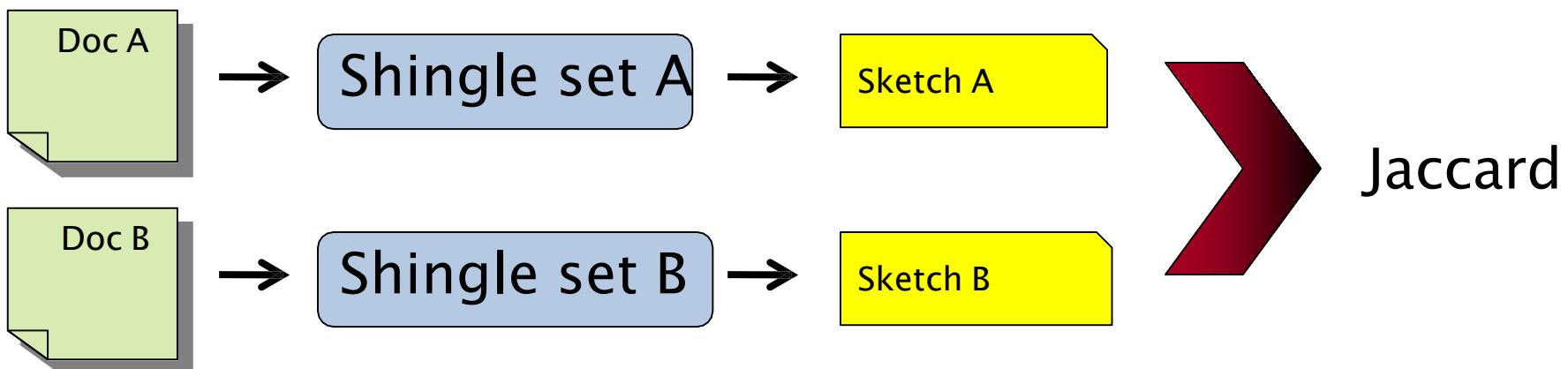
- Duplication: Exact match can be detected with fingerprints
- Near-Duplication: Approximate match
 - Overview
 - Compute syntactic similarity with an edit-distance measure
 - Use similarity threshold to detect near-duplicates
 - E.g., Similarity > 80% => Documents are “near duplicates”
 - Not transitive though sometimes used transitively

Computing Similarity

- Features:
 - Segments of a document (natural or artificial breakpoints)
 - Shingles (Word N-Grams)
 - **a rose is a rose is a rose** → 4-grams are
 - **a_rose_is_a**
 - **rose_is_a_rose**
 - **is_a_rose_is**
- Similarity Measure between two docs (= sets of shingles)
 - Jaccard coefficient: $(\text{Size_of_Intersection} / \text{Size_of_Union})$

Shingles + Set Intersection

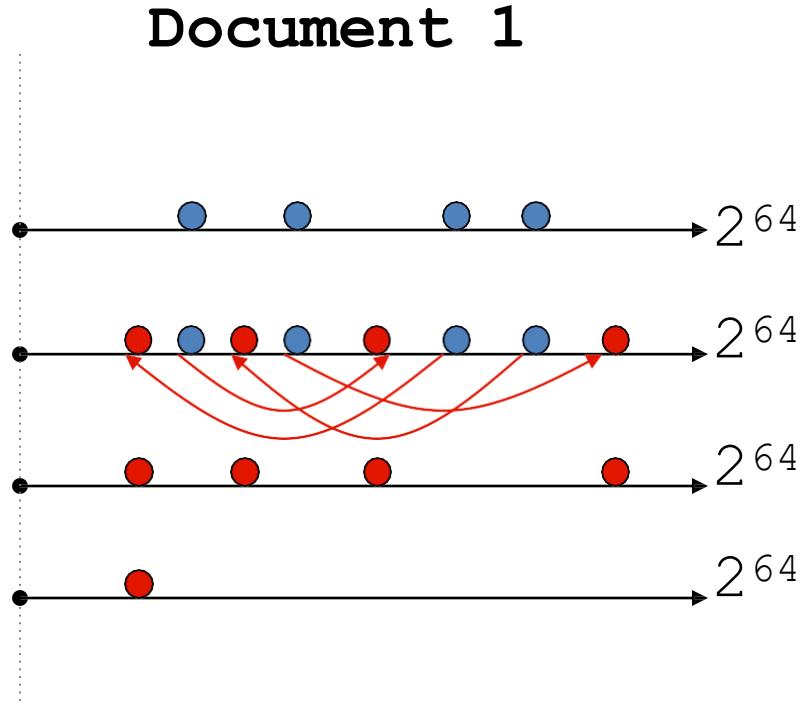
- Computing exact set intersection of shingles between all pairs of documents is expensive
- Approximate using a cleverly chosen subset of shingles from each (a *sketch*)
- Estimate $(\text{size_of_intersection} / \text{size_of_union})$ based on a short sketch



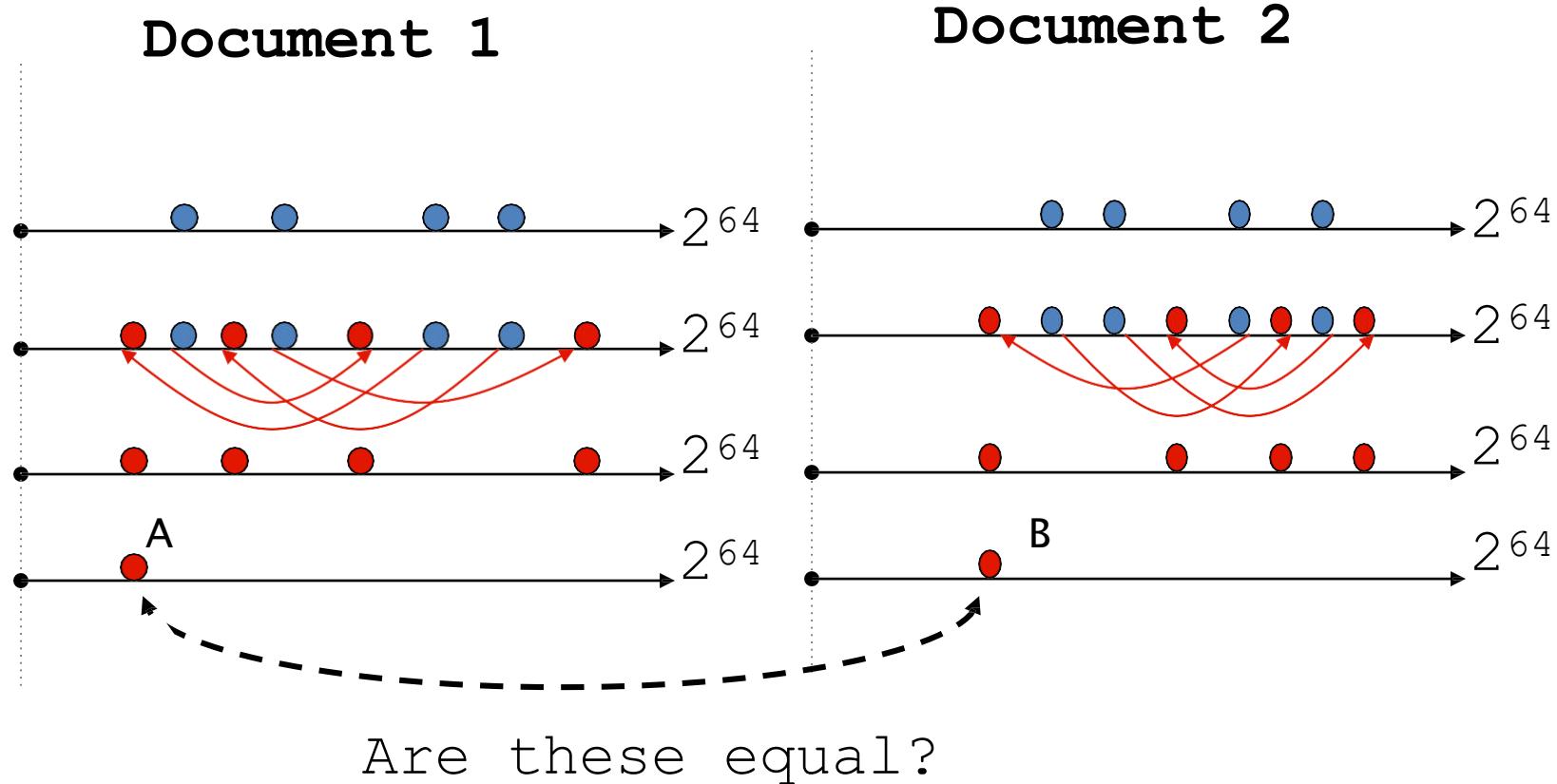
Sketch of a document

- . Create a “sketch vector” (of size ~ 200) for each document
 - . Documents that share $\geq t$ (say 80%) corresponding vector elements are deemed **near duplicates**
 - . For doc D , $\text{sketch}_D[i]$ is as follows:
 - . Let f map all shingles in the universe to $1..2^m$ (e.g., $f = \text{fingerprinting}$)
 - . Let π_i be a *random permutation* on $1..2^m$
 - . Pick $\text{MIN } \{\pi_i(f(s))\}$ over all shingles s in D

Computing Sketch[i] for Doc1



Test if $\text{Doc1.Sketch}[i] = \text{Doc2.Sketch}[i]$



Test for 200 random permutations: $\pi_1, \pi_2, \dots, \pi_{200}$

Web Crawler Coverage

- Even large search engines cover only a portion of the publicly available content
- Gulli et al showed in 2005 that the coverage of large search engines is between 58% and 76% of the Web
- It is highly desirable that such downloaded fraction contains the most authoritative pages

Crawling the Hidden Web

- The process we have described allows the indexing of all the Web that is reachable by just following links
 - *Raghavan and Garcia-Molina* observed that there is an enormous amount of information not reachable by following links, but only by querying or filling forms
 - This fraction of the Web is known as the **hidden or deep Web** (*Bright-Planet*)
 - It was estimated in the year 2000 as 550 times larger (in terms of pages) than the full Web
 - More recent studies suggest that the size of the deep Web might be even larger (*Chang et al*)
-

Practical Global Web Crawlers

- The Internet Archive uses a crawler called **Heritrix**,
 - It was designed with the purpose of archiving periodic snapshots of a large portion of the Web
 - It uses several processes in a distributed fashion, and a fixed number of Web sites are assigned to each process
- The early architecture of **Google** is described by **Sergey Brin and Lawrence Page**
 - The crawler was integrated with the indexing process
 - During parsing, the URLs found were passed to a URL server that checked if the URL have been previously seen
 - If not, the URL was added to the queue of the URL server

Practical Global Web Crawlers

- The architecture of the **FAST Search Engine** was described by Risvik and Michelsen
 - It is a distributed architecture in which each machine holds a document scheduler
 - Each scheduler maintains a queue of documents to be downloaded by a document processor
 - Each crawler communicates with the other crawlers via a distributor module

Modular Web Crawlers

- **Mercator** is a modular Web crawler written in Java
 - Its modularity arises from the usage of interchangeable protocol modules and processing modules
 - Protocols modules are related to how to acquire the Web pages Processing modules are related to how to process Web pages
 - Other processing modules can be used to index the text of the pages, or to gather statistics from the Web

Modular Web Crawlers

- **WebFountain** is a distributed, modular crawler similar to Mercator
 - It features a controller machine that coordinates a series of ant machines
 - It also includes a module for stating and solving an equation system for maximizing the freshness
- **WebSPHINX** is composed of a Java class library and a development environment for web crawlers
 - It implements multi-threaded Web page collector and HTML parser, and a graphical user interface to set the starting URLs

Open Source Web Crawlers

- **NUTCH** is an open-source crawler written in Java that is part of the Lucene search engine
 - It is sponsored by the Apache Foundation It includes a simple interface for intranet Web crawling as well as a more powerful set of commands for large-scale crawl
- **WIRE** is an open-source web crawler written in C++
Includes several policies for scheduling the page downloads
 - Also includes a module for generating reports and statistics on the downloaded pages It has been used for Web characterization
- Other crawlers described in the literature include WebBase (in C), CobWeb (in Perl), PolyBot (in C++ and Python), and WebRace (in Java)

Trends and Research Issues

- Improving the selection policy, in the sense of developing strategies to discover relevant items early during the crawl
- Improving memory and network usage
- Crawling for acquiring facts, including crawling the semantic Web
- Doing crawling in other environments, such as in peer-to-peer services, etc.

References

- <https://www.youtube.com/watch?v=940z8nTZtOw>
- <https://www.coursera.org/lecture/text-retrieval/lesson-5-4-web-search-introduction-web-crawler-qkTHD>
- <https://www.youtube.com/watch?v=u3BO0BspfM>
- <https://www.youtube.com/watch?v=X6kMrZciiTc>

- Introduction to Information Retrieval by Chris Manning and Pandu Nayak
- Modern Information Retrieval. Chapter 12. Web Crawling with Carlos Castillo.



THANK YOU



BITS Pilani
Pilani Campus

Information Retrieval

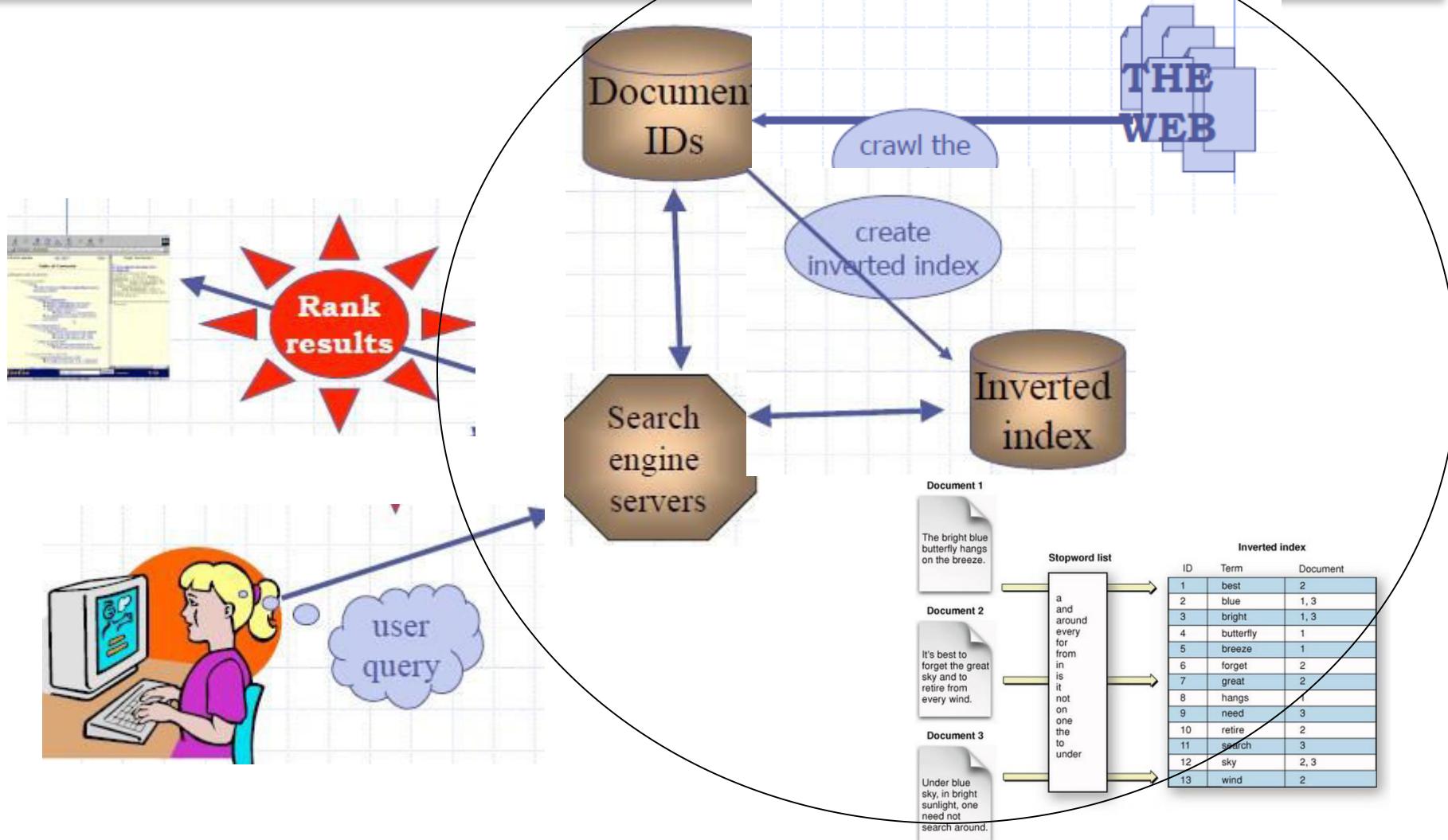
Dr. Vijayalakshmi Anand

BITS pilani

Lecture Outline

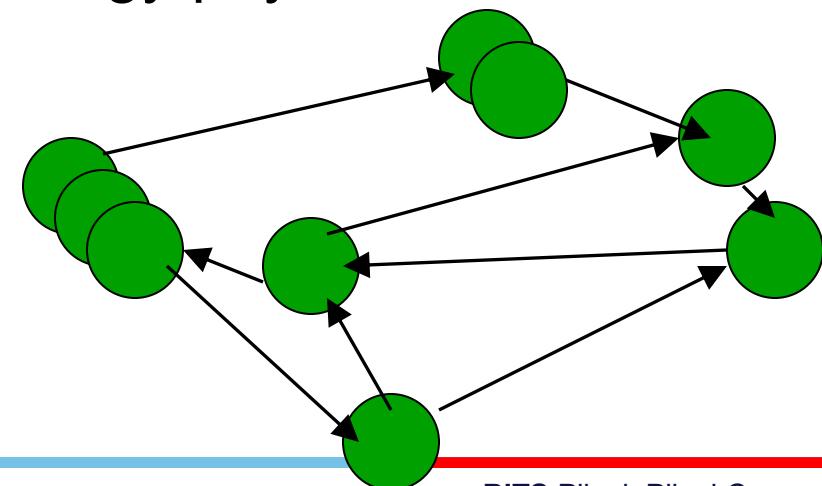
- **Link analysis**
 - The web as a graph
 - Page rank
 - Hub and Authorities

Web at a glance



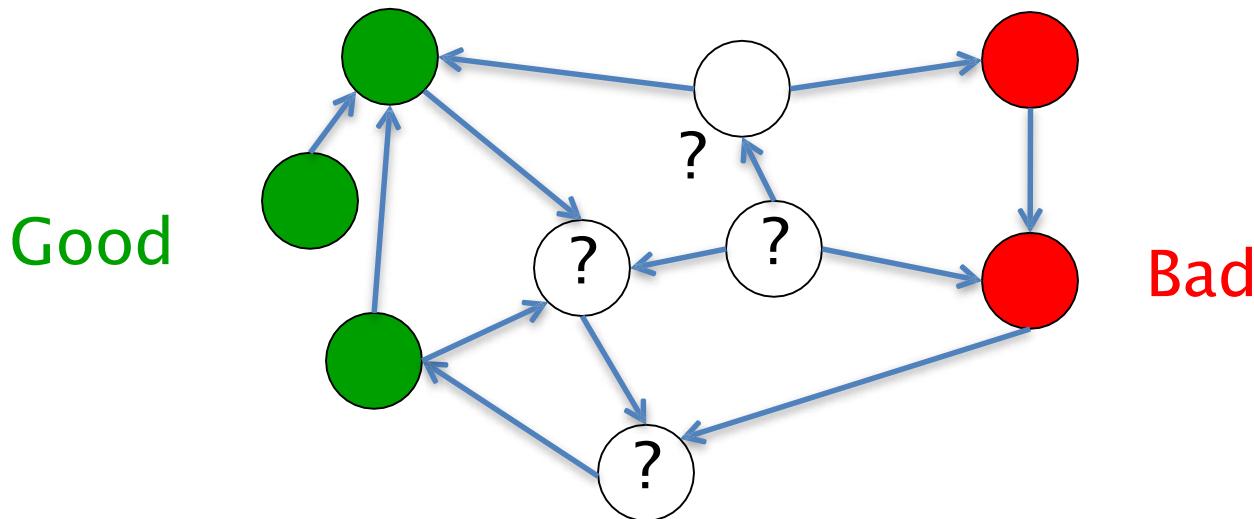
Hypertext and links

- We look beyond the *content* of documents
 - We begin to look at the hyperlinks between them
- Address questions like
 - Do the links represent a conferral of authority to some pages? Is this useful for ranking?
 - How likely is it that a page pointed to by the CERN home page is about high energy physics
- Big application areas
 - The Web
 - Email
 - Social networks



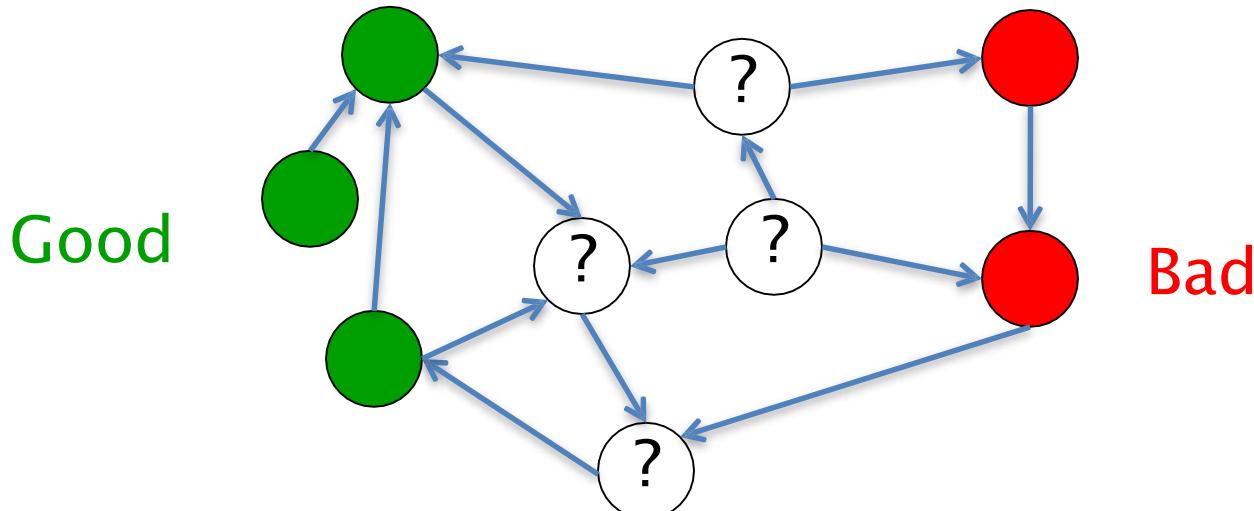
Links are everywhere

- Powerful sources of authenticity and authority
 - Mail spam – which email accounts are spammers?
 - Host quality – which hosts are “bad”?
- The **Good**, The **Bad** and The Unknown



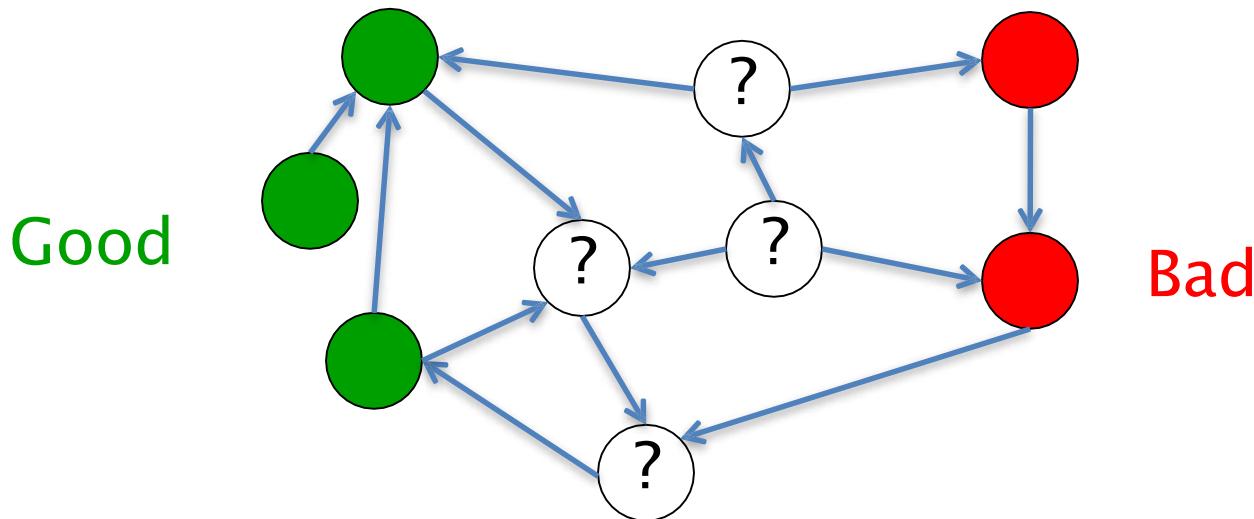
Example 1: Good/Bad/Unknown

- The **Good**, The **Bad** and The Unknown
 - **Good** nodes won't point to **Bad** nodes
 - All other combinations plausible



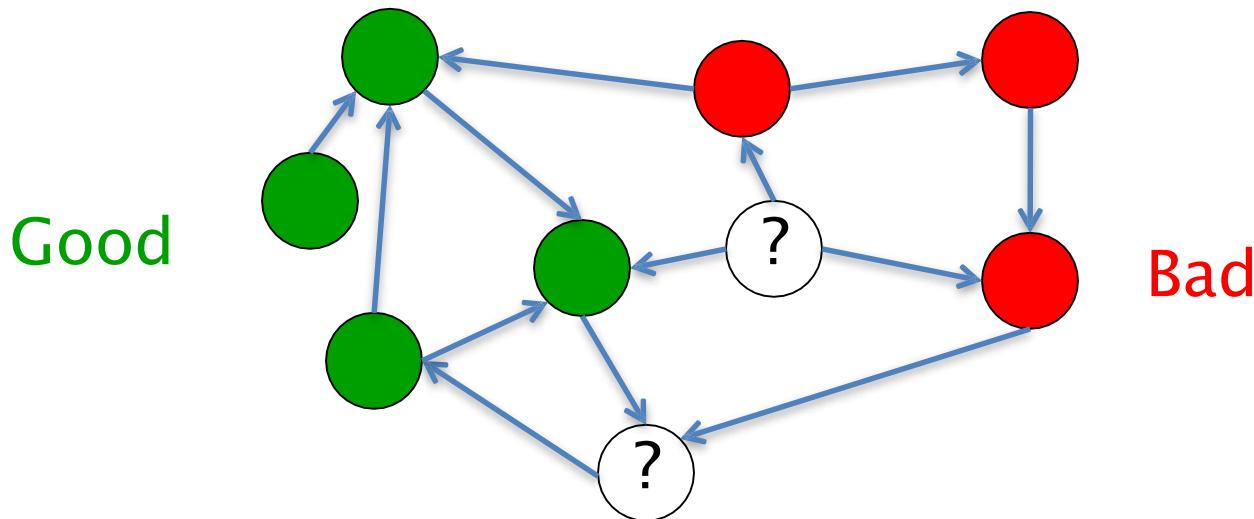
Simple iterative logic

- Good nodes won't point to Bad nodes
 - If you point to a Bad node, you're Bad
 - If a Good node points to you, you're Good



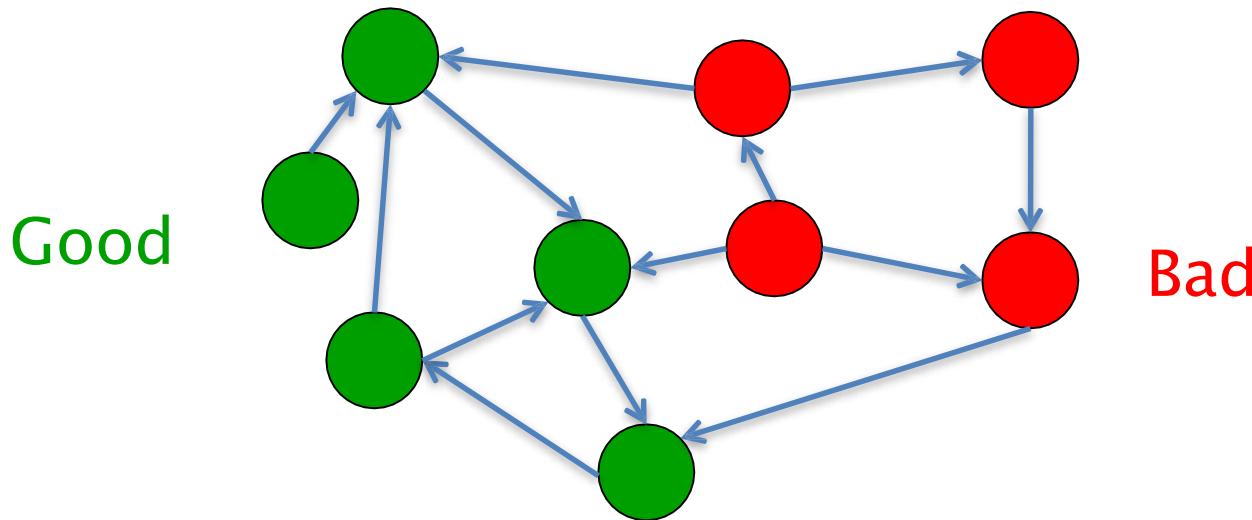
Simple iterative logic

- **Good** nodes won't point to **Bad** nodes
 - If you point to a **Bad** node, you're **Bad**
 - If a **Good** node points to you, you're **Good**



Simple iterative logic

- Good nodes won't point to Bad nodes
 - If you point to a Bad node, you're Bad
 - If a Good node points to you, you're Good



Sometimes need probabilistic analogs – e.g., mail spam

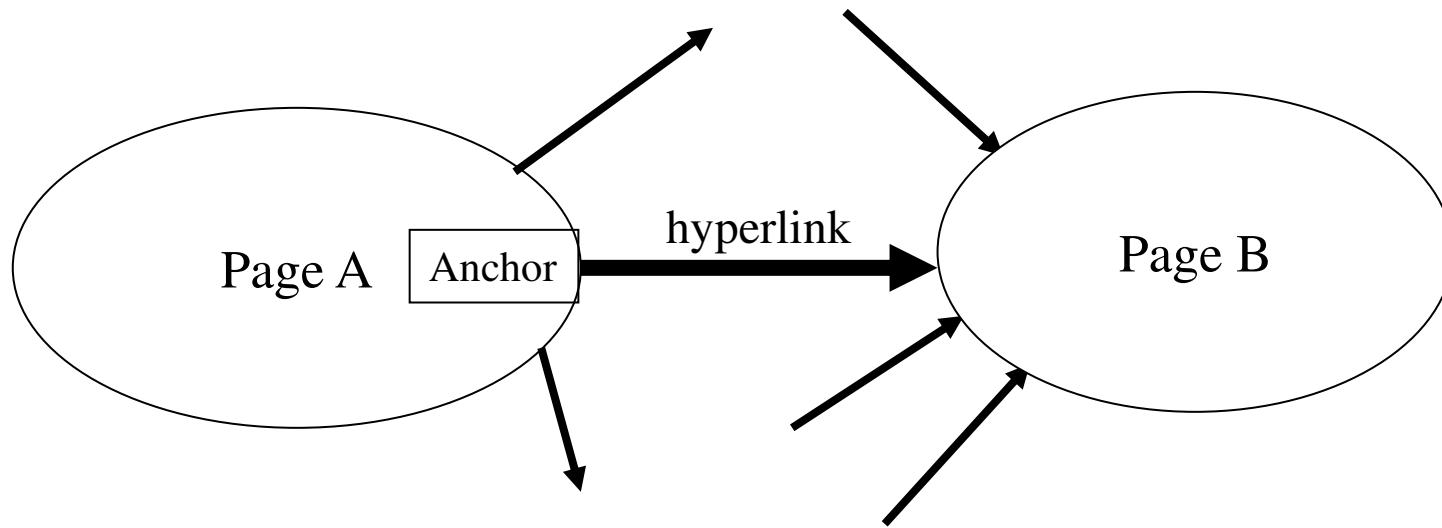
Many other examples of link analysis

- Social networks are a rich source of grouping behavior
- E.g., Shoppers' affinity – Goel+Goldstein 2010
 - **Consumers whose friends spend a lot, spend a lot themselves**
- <http://www.cs.cornell.edu/home/kleinber/networks-book/>

Link analysis

- Using hyperlinks for ranking web search results
- It is one of the factors used by search engines to compute a composite score for a web page on any given query
- Link analysis for most IR functionality thus far based purely on text
 - Scoring and ranking
 - Link-based clustering – topical structure from links
 - Links as features in classification – documents that link to one another are likely to be on the same subject
- Crawling
 - Based on the links seen, where do we crawl next?

Web as a Graph



Assumption 1: A hyperlink is a quality signal.

- The hyperlink $d_1 \rightarrow d_2$ indicates that d_1 's author deems d_2 high-quality and relevant.

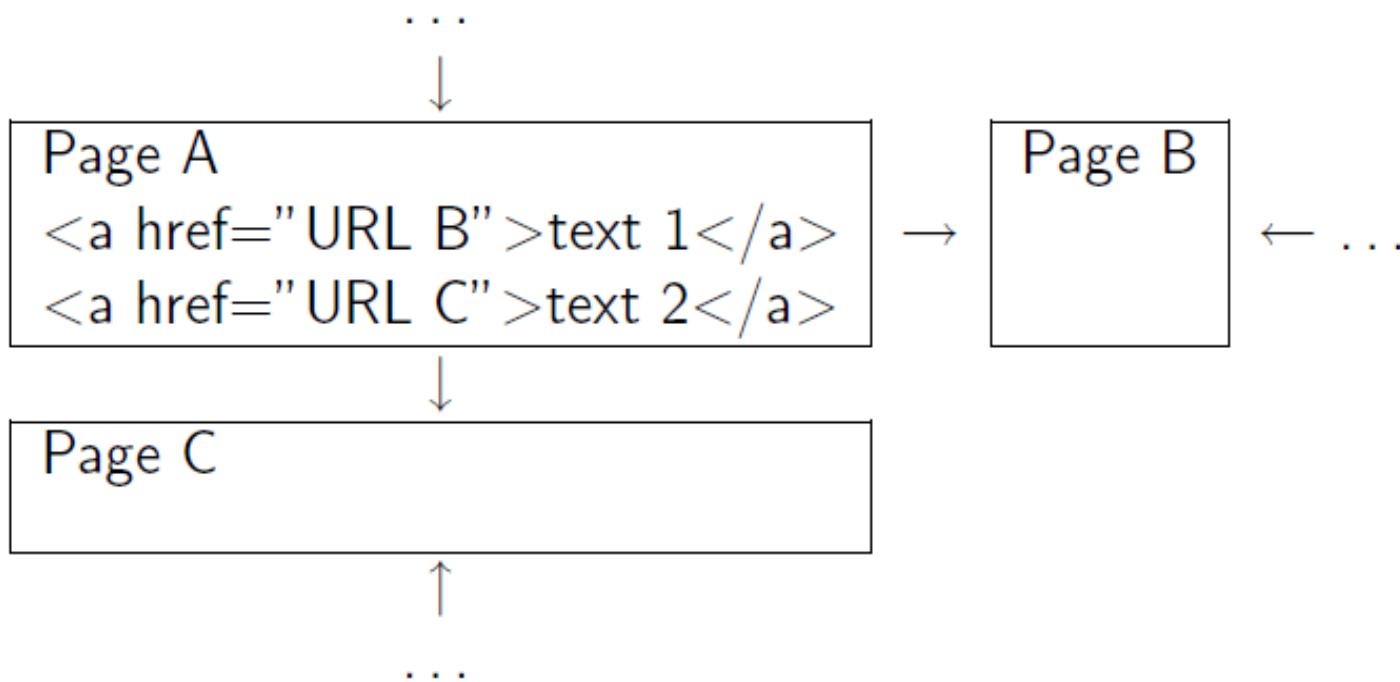
Assumption 2: The anchor text describes the content of d_2 .

- We use anchor text somewhat loosely here for: the text surrounding the hyperlink.

Example: “You can find cheap cars here.”

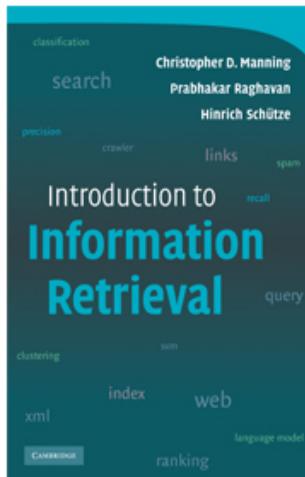
Anchor text: “You can find cheap cars here”

Web as a Graph



Assumption 1: reputed sites

Introduction to Information Retrieval



This is the companion website for the following book.

Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, *Introduction to Information Retrieval*

You can order this book at [CUP](#), at your local bookstore or on the internet. The best search

The book aims to provide a modern approach to information retrieval from a computer science perspective. It is available at [University of Cambridge](#) and at the [University of Stuttgart](#).

We'd be pleased to get feedback about how this book works out as a textbook, what is missing, and so on. Please send comments to: informationretrieval (at) yahoogroups (dot) com

Assumption 2: annotation of target

The image shows two versions of the Tohoku University website side-by-side, connected by a large green downward-pointing arrow.

Top Version (Japanese):

- Logo: 東北大学 TOHOKU UNIVERSITY
- Language Links: 中文 | 한국어 | **English** | 日本語 (The English link is circled in red.)
- Navigation: 大学概要, 学部・大学院・研究所, 教育・学生支援, 國際交流, 研究・産学連携

Bottom Version (English):

- Logo: 東北大学 TOHOKU UNIVERSITY
- Language Links: Chinese | Korean | English | Japanese
- Search Bar: Search
- Navigation: Inquiry, Access, Sitemap
- Menu: About Tohoku University, Faculties, Schools and Institutes, Campus Life, International Exchange, Research and Cooperation, Disclosure and Public Information, Entrance Exam Information
- Left Sidebar: Prospective Students, General Public, Corporations, Alumni, Current Students, Faculty and Staff (Internal use)
- Image: A group of students at a university入学式 (Admission Ceremony).
- Right Sidebar: A purple banner for the "New! Video Channel" with a "Click Here" button.

Anchor Text

- Searching on [text of d2] + [anchor text → d2] is often more effective than searching on [text of d2] only.
- Example: Query IBM
 - Matches IBM's copyright page
 - Matches many spam pages
 - Matches IBM Wikipedia article
 - May not match IBM home page, if IBM home page is mostly graphics
- Searching on [anchor text → d2] is better for the query IBM
 - In this representation, the page with the most occurrences of IBM is **www.ibm.com**.

Anchor Text

- **Anchor text containing IBM pointing to www.ibm.com**

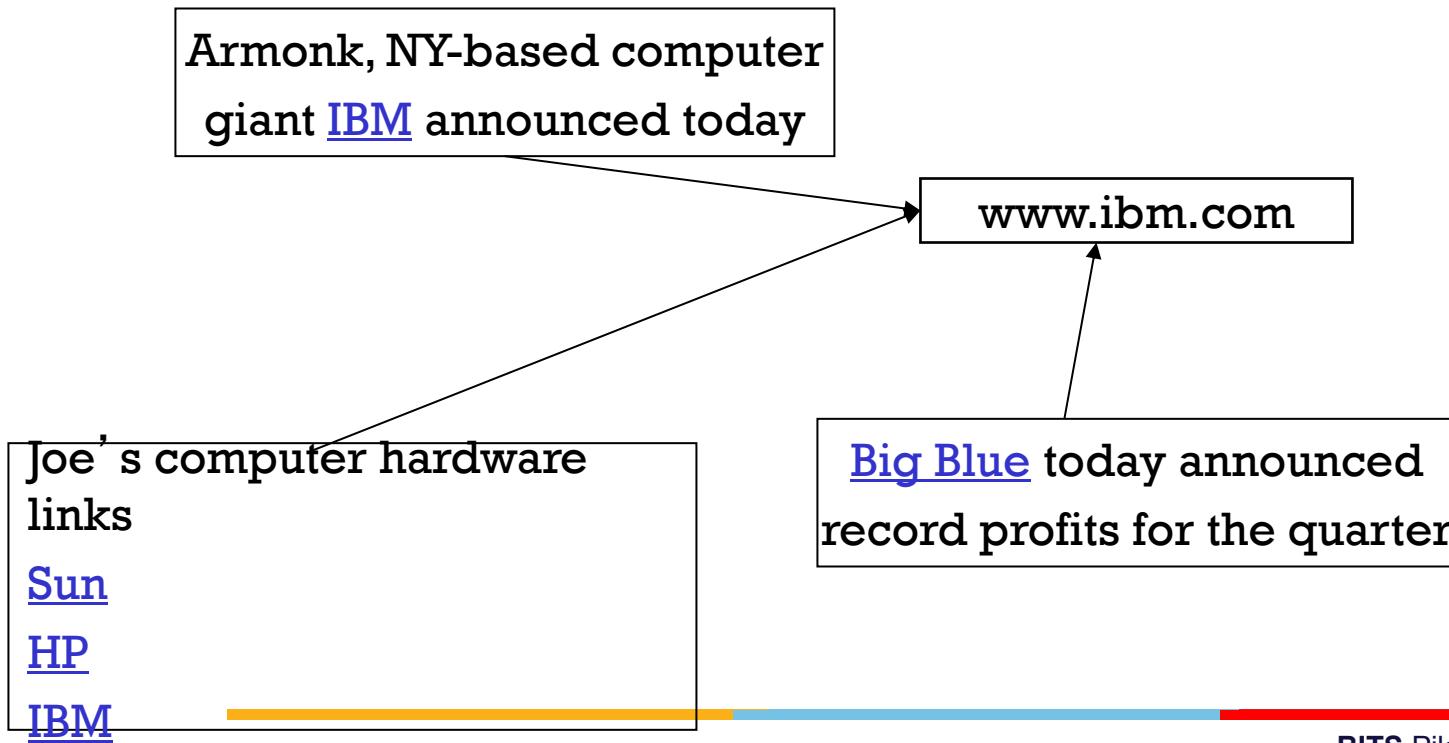
www.nytimes.com: “IBM acquires Webify”

www.slashdot.org: “New IBM optical chip”

www.stanford.edu: “IBM faculty award recipients”.

Indexing anchor text

- Thus: Anchor text is often a better description of a page's content than the page itself
- Anchor text can be weighted more highly than document text.



Indexing anchor text

- Can sometimes have unexpected effects, e.g., spam, **miserable failure**
- Can score anchor text with weight depending on the authority of the anchor page's website
 - E.g., if we were to assume that content from cnn.com or yahoo.com is authoritative, then trust (more) the anchor text from them.

Connectivity servers

- Quality of a page is a function of the links that points to it
- For link analysis, queries on the connectivity of the web graph are needed
- Support for fast queries on the web graph
 - Which URLs point to a given URL ?
 - Which URLs are pointed by a given URL ?
- Mappings stored:
 - From URL to out-links
 - From URL to in-links

Applications

- Link analysis
- Web graph analysis
 - Connectivity, crawl optimization
- Crawl control

Page Rank

- PageRank: Scoring measure based on the link structure of web pages
- Every node in the web graph is given a score between 0 and 1, depending on its in and out-links
- Given a query, a search engine combines the PageRank score with other values to compute the ranked retrieval (e.g. cosine similarity, relevance feedback, etc.)

Model behind PageRank: Random walk

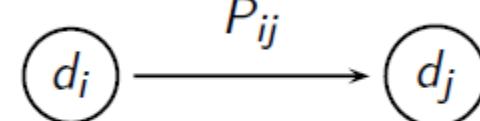
- Imagine a web surfer doing a random walk on the web
- Start at a random page
- At each step, go out of the current page along one of the links on that page, equiprobably
- In the steady state, each page has a long-term visit rate
- This long-term visit rate is the page's PageRank
- PageRank = long-term visit rate = steady state probability.

Markov chains

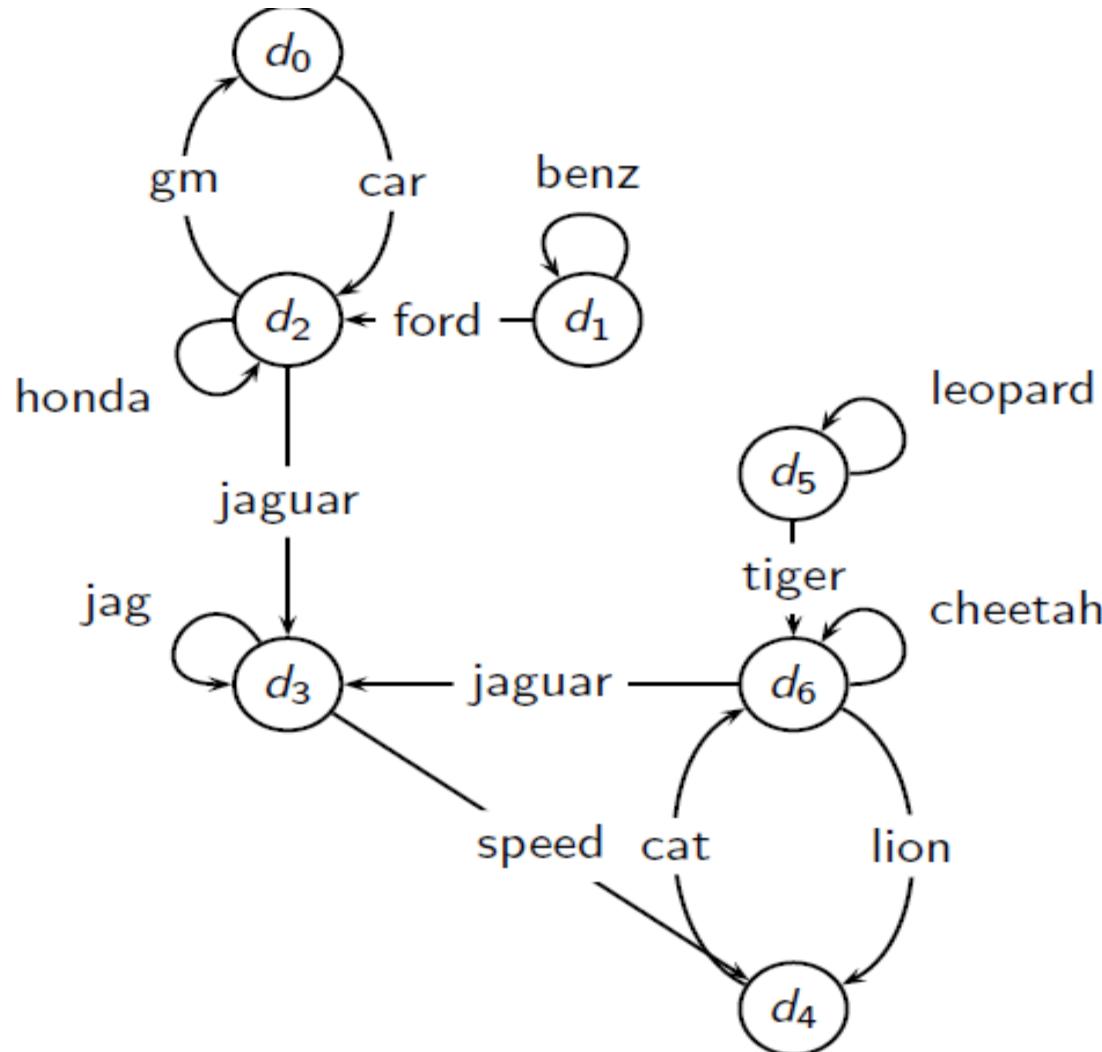
- Formalization of random walk:
- Markov chain is a ***discrete-time stochastic process***: a process that occurs in a series of time-steps in each of which a random choice is made.
- A Markov chain consists of N states, plus an $N \times N$ transition probability matrix P .
- state = page
- At each step, we are on exactly one of the pages.

For $1 \leq i, j \leq N$, the matrix entry P_{ij} tells us the probability of j being the next page, given we are currently on page i .

Clearly, for all i , $\sum_{j=1}^N P_{ij} = 1$



Example web graph



Link matrix for example

	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0	0	1	0	0	0	0
d_1	0	1	1	0	0	0	0
d_2	1	0	1	1	0	0	0
d_3	0	0	0	1	1	0	0
d_4	0	0	0	0	0	0	1
d_5	0	0	0	0	0	1	1
d_6	0	0	0	1	1	0	1

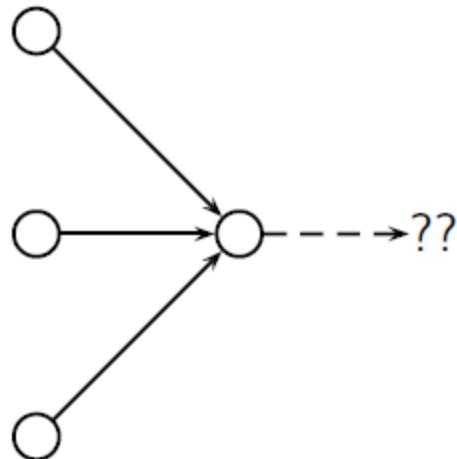
Transition probability matrix P

	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0.00	0.00	1.00	0.00	0.00	0.00	0.00
d_1	0.00	0.50	0.50	0.00	0.00	0.00	0.00
d_2	0.33	0.00	0.33	0.33	0.00	0.00	0.00
d_3	0.00	0.00	0.00	0.50	0.50	0.00	0.00
d_4	0.00	0.00	0.00	0.00	0.00	0.00	1.00
d_5	0.00	0.00	0.00	0.00	0.00	0.50	0.50
d_6	0.00	0.00	0.00	0.33	0.33	0.00	0.33

Long-term visit rate

- PageRank = long-term visit rate
- Long-term visit rate of page **d** is the probability that a web surfer is at page **d** at a given point in time
- The web graph must correspond to an ergodic Markov chain
- First a special case: The web graph must not contain dead ends.

Dead ends



- The web is full of dead ends
- Random walk can get stuck in dead ends
- If there are dead ends, long-term visit rates are not well-defined.

Teleporting

- Teleporting – to get us out of dead ends
- At a dead end, jump to a random web page with prob. $1/N$
- At a non-dead end, the surfer invokes the teleport operation with probability $0 < \alpha < 1$ and the standard random walk (follow an out-link chosen uniformly at random as in) with probability $1 - \alpha$, where α is a fixed parameter chosen in
 - For example, if the page has 4 outgoing links: randomly choose one with probability $(1-0.10)/4=0.225$
- 10% is a parameter, the teleportation rate
- “jumping” from dead end is independent of teleportation rate.

Teleporting

- With teleporting, we cannot get stuck in a dead end
- But even without dead ends, a graph may not have well-defined long-term visit rates
- More generally, we require that the Markov chain be ergodic.

Two formulas

$$PR(A) = (1-d) + d(PR(T_1)/L(T_1) + \dots + PR(T_n)/L(T_n))$$

$$PR(A) = (1-d)/N + d(PR(T_1)/L(T_1) + \dots + PR(T_n)/L(T_n))$$

Where T_1, T_2, \dots, T_n are pages that are pointing to A.

How is calculated?

- The PR of each page depends on the PR of the pages pointing to it.
- But we won't know what PR those pages have until the pages pointing to them have their PR calculated and so on.
- So what we do is make a guess.



- Each page has one outgoing link. So that means $L(A) = 1$ and $L(B) = 1$.

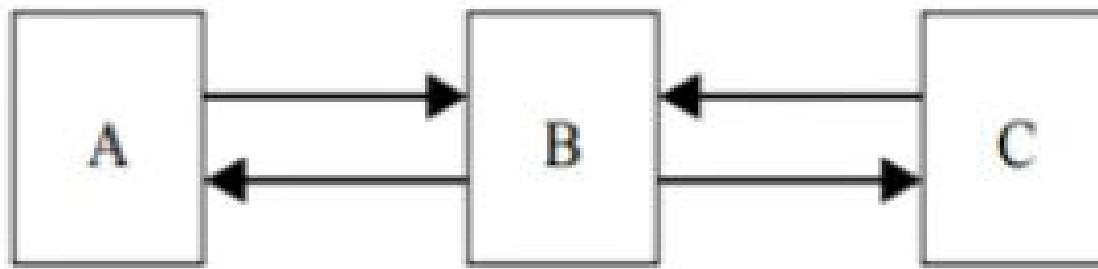
- d (damping factor) = 0.85
- $PR(A) = (1 - d) + d(PR(B)/1)$
- $PR(B) = (1 - d) + d(PR(A)/1)$

- $\text{PR(A)} = 0.15 + 0.85 * 0$
= 0.15
= $0.15 + 0.85 * 1$
 $\text{PR(B)} = 1$
- Now we have calculated a "next best guess" so we just plug it in the equation again...
- $\text{PR(A)} = 0.15 + 0.85 * 1$
= 1
- $\text{PR(B)} = 0.15 + 0.85 * 0.15$
= 0.2775

And again...

- $\text{PR(A)} = 0.15 + 0.85 * 0.2775$
= 0.385875
- $\text{PR(B)} = 0.15 + 0.85 * 1$
= 1

Example2



The number of web pages $N = 3$.

Consider the damping parameter $d = 0.7$. (Consider the page rank calculations by the second formula.)

$$PR(A) = (1 - d) \times (1 / N) + d \times (PR(B) / 2)$$

$$PR(B) = (1 - d) \times (1 / N) + d \times (PR(A) / 1 + PR(C) / 1)$$

$$PR(C) = (1 - d) \times (1 / N) + d \times (PR(B) / 2)$$

$$\text{So } PR(A) = 0.1 + 0.35 \times PR(B)$$

$$PR(B) = 0.1 + 0.70 \times PR(A) + 0.70 \times PR(C)$$

$$PR(C) = 0.1 + 0.35 \times PR(B)$$

By solving the above system of linear equations, we get

$$PR(A) = 0.2647$$

$$PR(B) = 0.4706$$

$$PR(C) = 0.2647.$$

Ergodic Markov chains

- **Definition:** A Markov chain is said to be *ergodic* if there exists a positive integer T_0 such that for all pairs of states i, j in the Markov chain, if it is started at time 0 in state i then for all $t > T_0$, the probability of being in state j at time t is greater than 0
- A Markov chain is ergodic iff it is **irreducible** and **aperiodic**
- **Irreducibility.** Roughly: there is a path from any page to any other page
- **Aperiodicity.** Roughly: The pages cannot be partitioned such that the random walker visits the partitions sequentially

Ergodic Markov chains

- **Theorem:** For any ergodic Markov chain, there is a unique long-term visit rate for each state
- This is the steady-state probability distribution
- Over a long time period, we visit each state in proportion to this rate
- It doesn't matter where we start
- Teleporting makes the web graph ergodic
 - Web-graph+teleporting has a steady-state probability distribution
 - Each page in the web-graph+teleporting has a PageRank.

Issues with Page rank algorithm

1. PageRank scores do not reflect current events.
2. inability to handle queries containing natural language and information outside of keywords.

HITS – Hyperlink-Induced Topic Search

There are two different types of relevance on the web

- **Hubs:** A hub page is a good list of links to pages answering the information need
 - E.g., for query [chicago bulls]: Alice's list of recommended resources on the Chicago Bulls sports team
- **Authorities:** An authority page is a direct answer to the information need
 - The home page of the Chicago Bulls sports team
 - By definition: Links to authority pages occur repeatedly on hub pages
- Most approaches to search (including PageRank ranking) don't make the distinction between these two types of relevance.

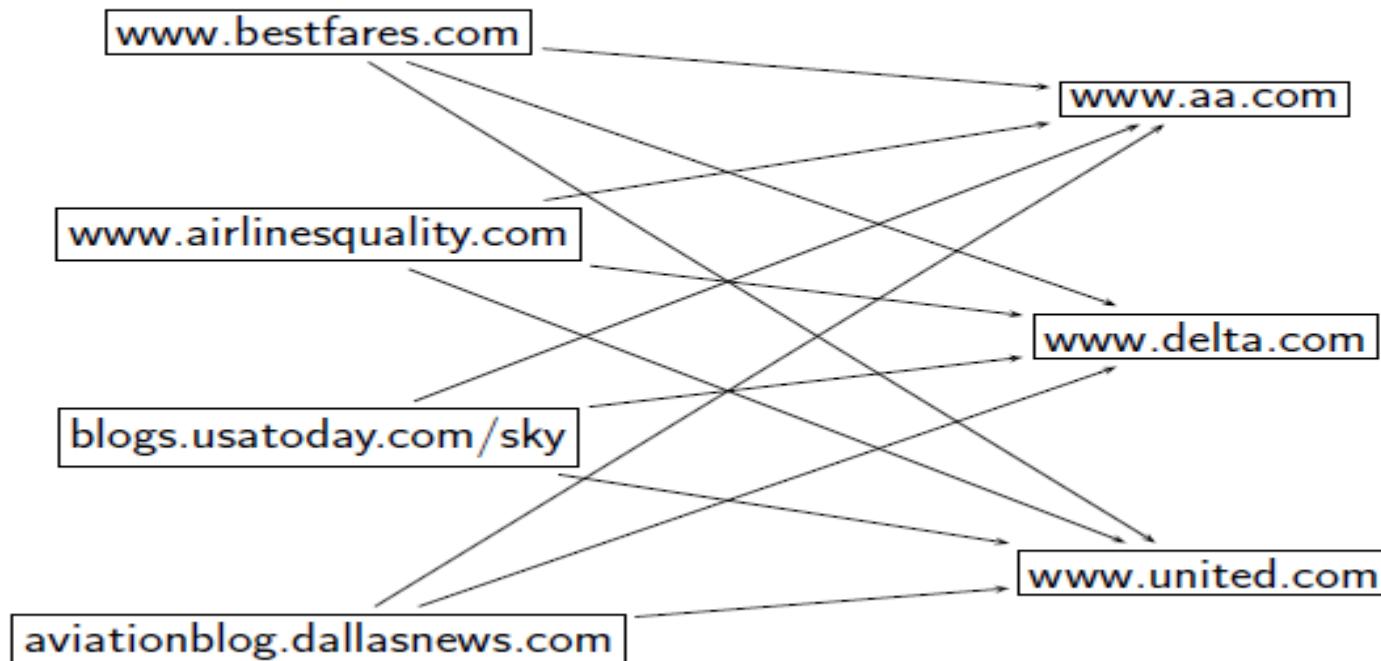
Hubs and authorities: Definition

- A good hub page for a topic links to many authority pages for that topic
- A good authority page for a topic is linked to by many hub pages for that topic
- Appear to have a circular definition of hubs and authorities; we will turn this into an iterative computation.

Example

hubs

authorities



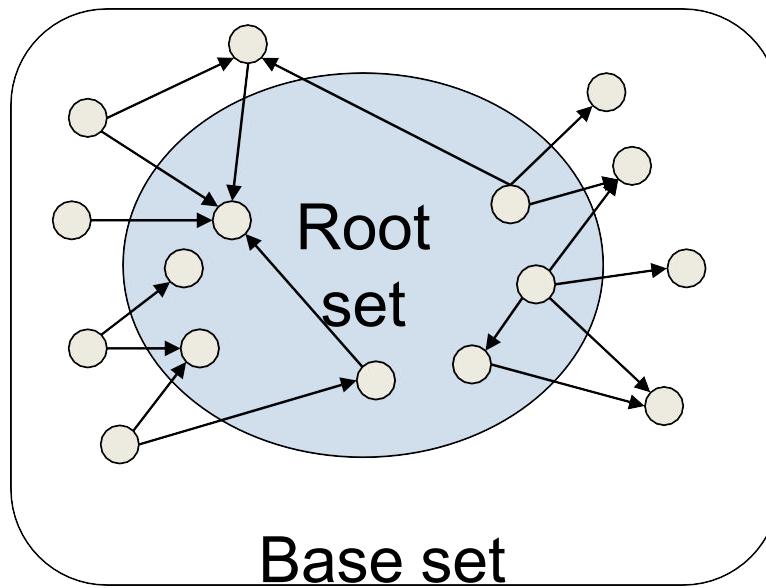
High-level scheme

- . Extract from the web a base set of pages that *could* be good hubs or authorities.
- . From these, identify a small set of top hub and authority pages;
→ iterative algorithm.

Base set

- . Given text query (say ***browser***), use a text index to get all pages containing ***browser***.
 - . Call this the root set of pages.
- . Add in any page that either
 - . points to a page in the root set, or
 - . is pointed to by a page in the root set.
- . Call this the base set.

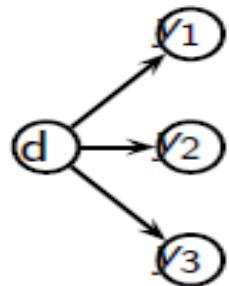
Visualization



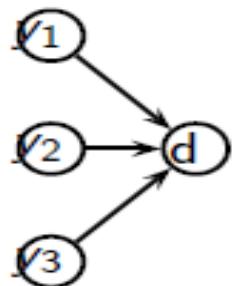
Get in-links (and out-links) from a *connectivity server*

Hub and authority scores

For all d : $h(d) = \sum_{d \rightarrow y} a(y)$



For all d : $a(d) = \sum_{y \rightarrow d} h(y)$



Iterate these two steps until convergence

Hub and authority scores

- Compute for each page d , a hub score $h(d)$ and an authority score $a(d)$
- Initialization: for all d : $h(d) = 1$, $a(d) = 1$
- Iteratively update all $h(d)$, $a(d)$
- After convergence:
 - Output pages with highest h scores as top hubs
 - Output pages with highest a scores as top authorities
 - So we output two ranked lists

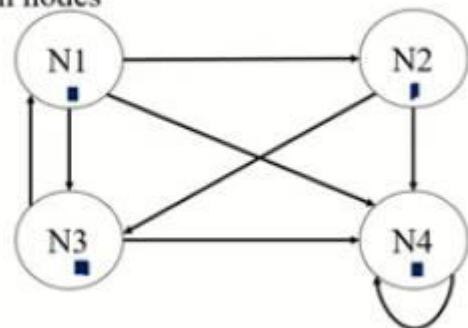
Calculate the hub and authority score using HITS algorithm k=3 the adjancy matrix is

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Adjacency matrix with nodes

	N1	N2	N3	N4
N1	0	1	1	1
N2	0	0	1	1
N3	1	0	0	1
N4	0	0	0	1

Graph with nodes



Ranks using out degree and in degree

Nodes	Out-degree(Hub)	In-degree(Authority)
N1	3	1
N2	2	1
N3	2	2
N4	1	4

Adjacency matrix A with nodes

	N1	N2	N3	N4
N1	0	1	1	1
N2	0	0	1	1
N3	1	0	0	1
N4	0	0	0	1

Transpose of Matrix A^T

	N1	N2	N3	N4
N1	0	0	1	0
N2	1	0	0	0
N3	1	1	0	0
N4	1	1	1	1

Authority weight vector , $v = A^T * u$

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} * \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Authority, v

$$\begin{pmatrix} 1 \\ 1 \\ 2 \\ 4 \end{pmatrix} \blacksquare$$

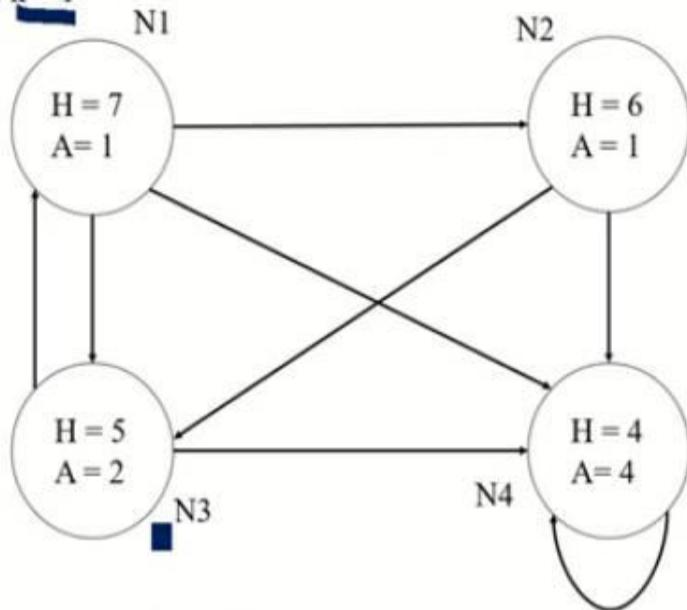
Updated Hub weight vector , $u = A * v$

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} 1 \\ 1 \\ 2 \\ 4 \end{pmatrix}$$

Hub, u

$$\begin{pmatrix} 7 \\ 6 \\ 5 \\ 4 \end{pmatrix}$$

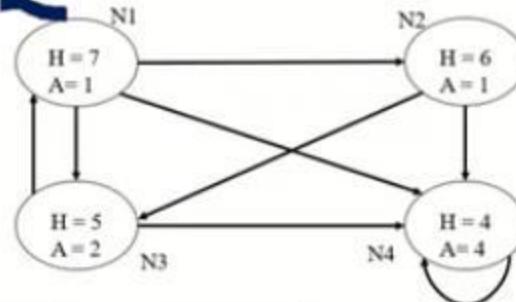
For $k = 1$



HUB: N1, N2, N3, N4

AUTHORITY: N4, N3, N2, N1 {TIE}

For $k = 1$



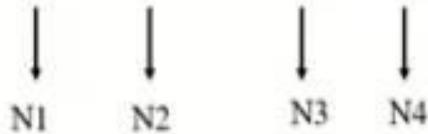
Nodes	Hub	Authority
N1	7	1
N2	6	1
N3	5	2
N4	4	4

Calculate new Authority from K = 1

$$v_1 = 1^2 + 1^2 + 2^2 + 4^2 = 22$$

$$= \frac{1}{\sqrt{22}}, \frac{1}{\sqrt{22}}, \frac{2}{\sqrt{22}}, \frac{4}{\sqrt{22}}$$

$$v_1 = 0.213, 0.213, 0.426, 0.853$$



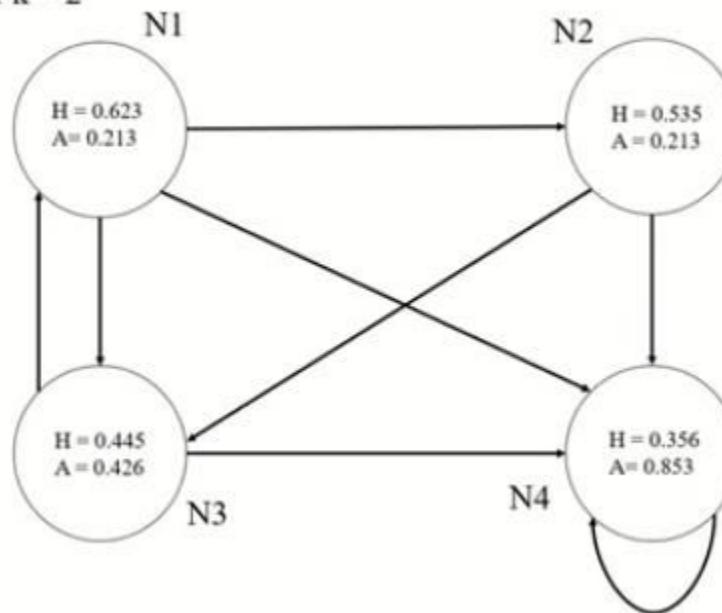
For k = 2

Calculate new hub from K = 1

$$u_1 = 7^2 + 6^2 + 5^2 + 4^2 = 126$$

$$= \frac{7}{\sqrt{126}}, \frac{6}{\sqrt{126}}, \frac{5}{\sqrt{126}}, \frac{4}{\sqrt{126}}$$

$$u_1 = 0.623, 0.535, 0.445, 0.356$$



For $k = 2$

Nodes	Hub	Authority
N1	0.623	0.213
N2	0.535	0.213
N3	0.445	0.426
N4	0.356	0.853

HUB: N1, N2, N3, N4

AUTHORITY: N4, N3, N2, N1 {TIE}

Calculate new Authority from K = 2

$$v_1 = 0.213^2 + 0.213^2 + 0.426^2 + 0.853^2 = 0.999$$
$$= \frac{0.213}{\sqrt{0.999}}, \frac{0.213}{\sqrt{0.999}}, \frac{0.426}{\sqrt{0.999}}, \frac{0.853}{\sqrt{0.999}}$$

$$v_1 = 0.213, 0.213, 0.426, 0.853$$



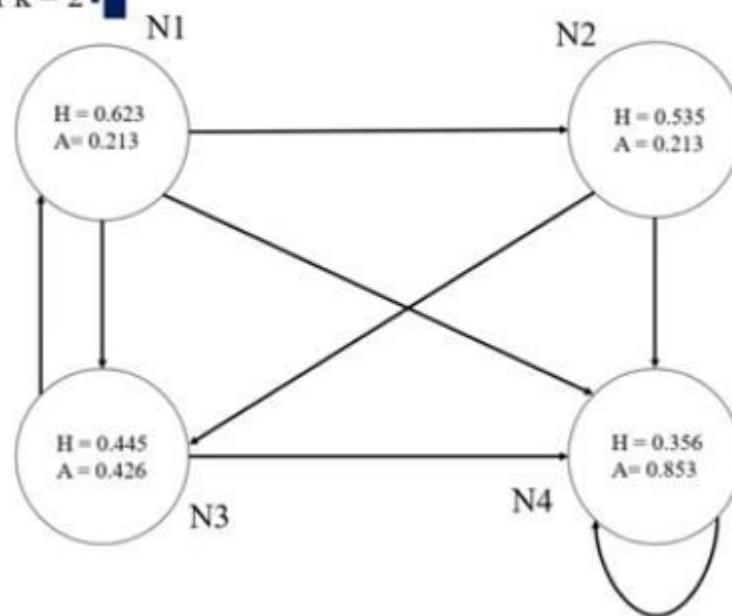
Calculate new hub from K = 2

$$u_1 = 0.623^2 + 0.535^2 + 0.445^2 + 0.356^2 = 0.999$$
$$= \frac{0.623}{\sqrt{0.999}}, \frac{0.535}{\sqrt{0.999}}, \frac{0.445}{\sqrt{0.999}}, \frac{0.356}{\sqrt{0.999}}$$

$$u_1 = 0.623, 0.535, 0.445, 0.356$$



For k = 2



Hubs and Authorities: Summary

- HITS can pull together good pages regardless of page content
- Once the base set is assembled, we only do link analysis, no text matching
- Pages in the base set often do not contain any of the query words.

Hubs and Authorities: Issues

- Topic Drift
 - Off-topic pages can cause off-topic “authorities” to be returned
 - E.g., the neighborhood graph can be about a “super topic”
- Mutually Reinforcing Affiliates
 - Affiliated pages/sites can boost each others’ scores
 - Linkage between affiliated pages is not a useful signal.



Thank You!



BITS Pilani
Pilani Campus

Information Retrieval

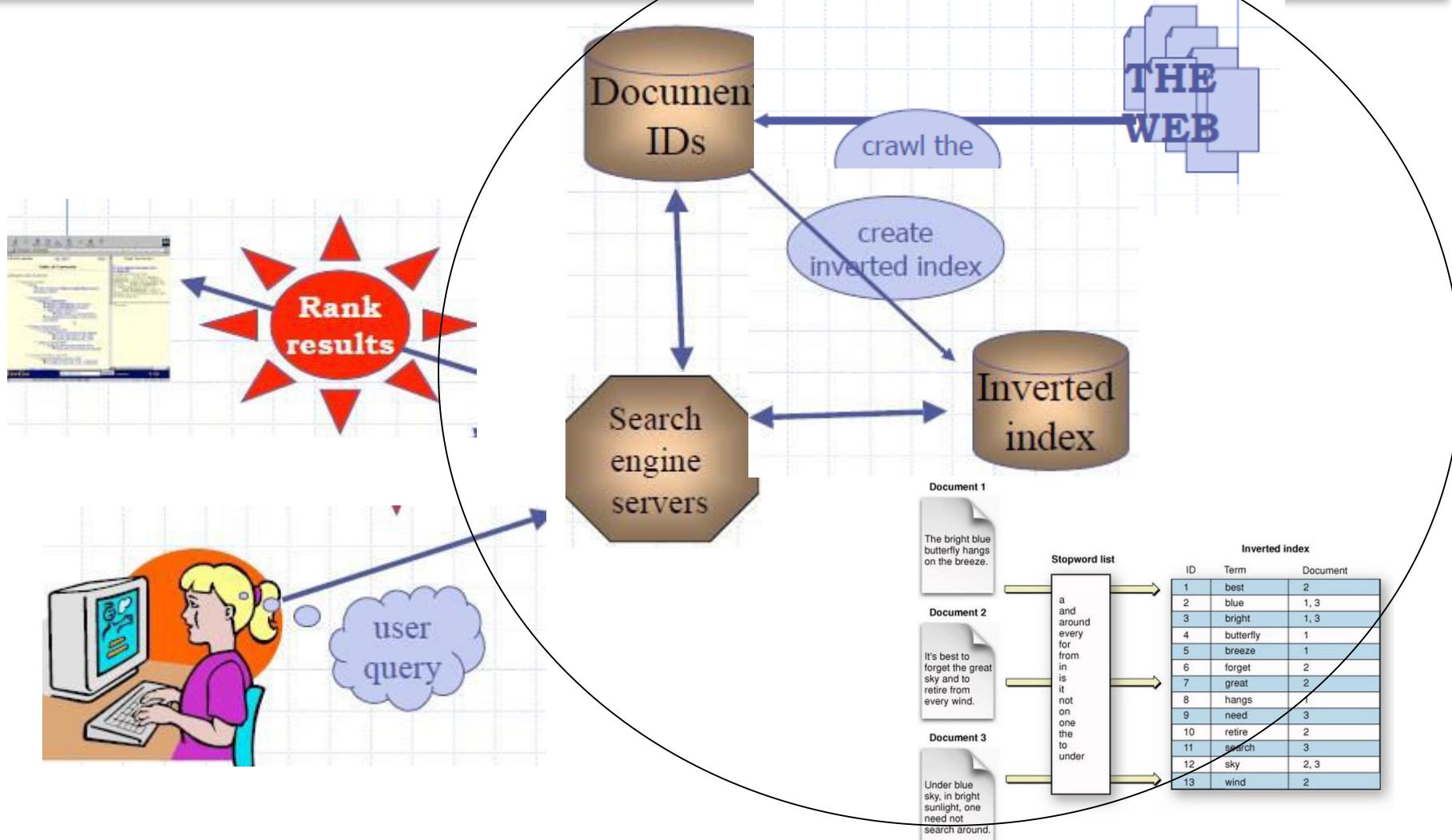
Dr. Vijayalakshmi Anand

BITS pilani

Lecture Outline

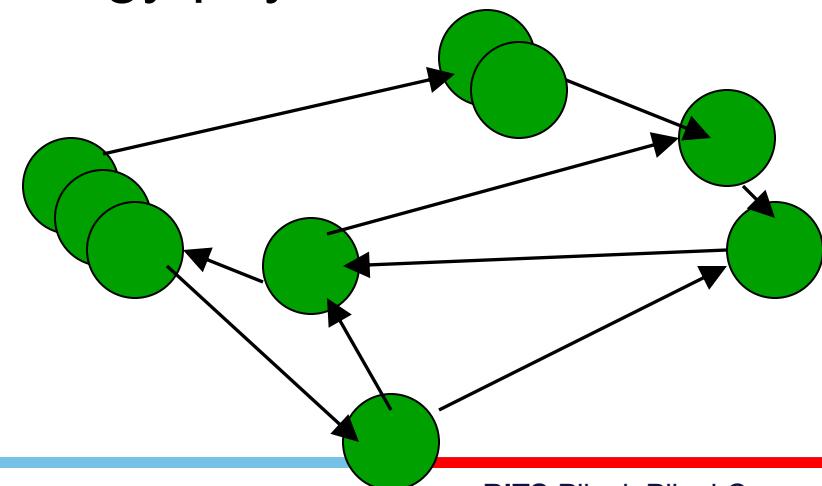
- **Link analysis**
 - The web as a graph
 - Page rank
 - Hub and Authorities

Web at a glance



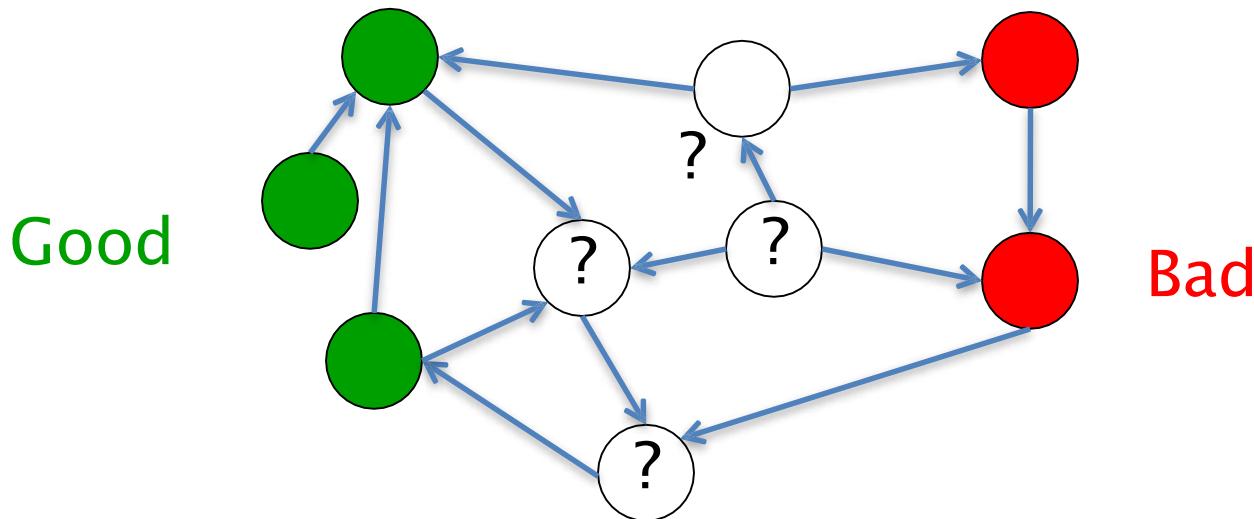
Hypertext and links

- We look beyond the *content* of documents
 - We begin to look at the hyperlinks between them
- Address questions like
 - Do the links represent a conferral of authority to some pages? Is this useful for ranking?
 - How likely is it that a page pointed to by the CERN home page is about high energy physics
- Big application areas
 - The Web
 - Email
 - Social networks



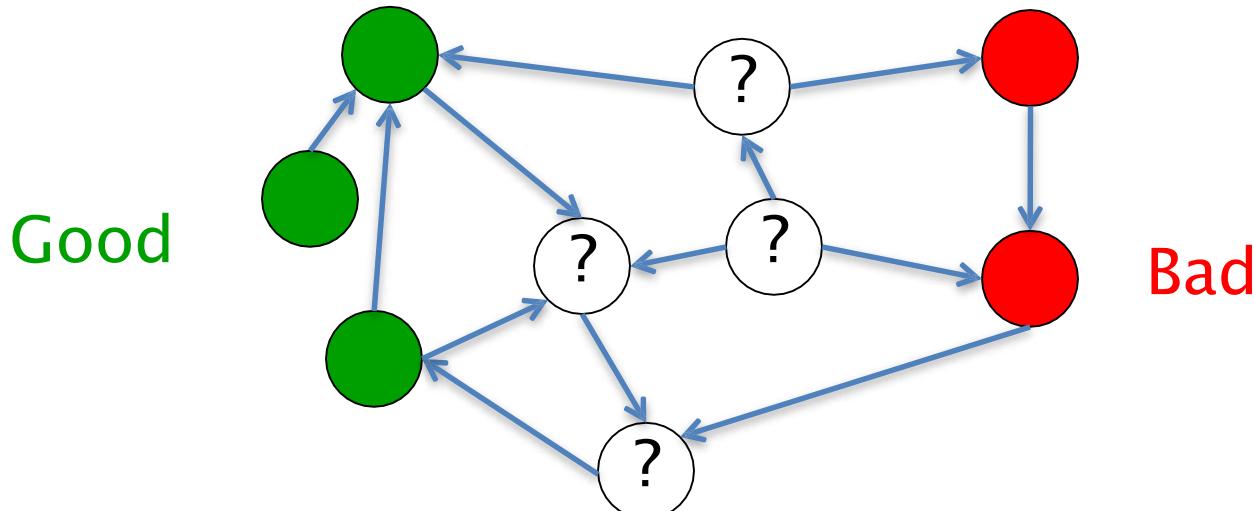
Links are everywhere

- Powerful sources of authenticity and authority
 - Mail spam – which email accounts are spammers?
 - Host quality – which hosts are “bad”?
- The **Good**, The **Bad** and The Unknown



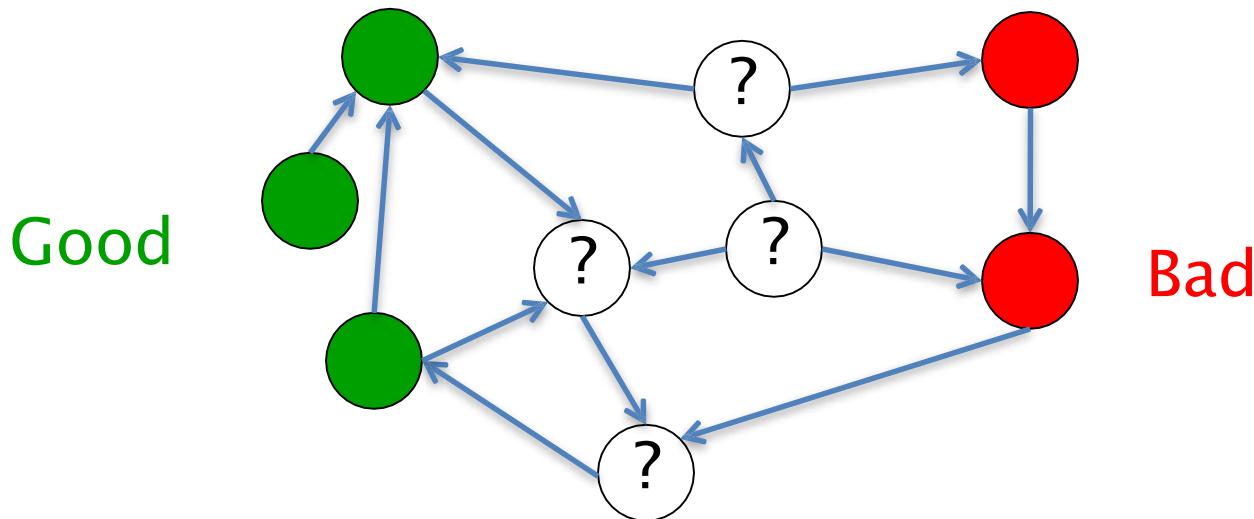
Example 1: Good/Bad/Unknown

- The **Good**, The **Bad** and The Unknown
 - **Good** nodes won't point to **Bad** nodes
 - All other combinations plausible



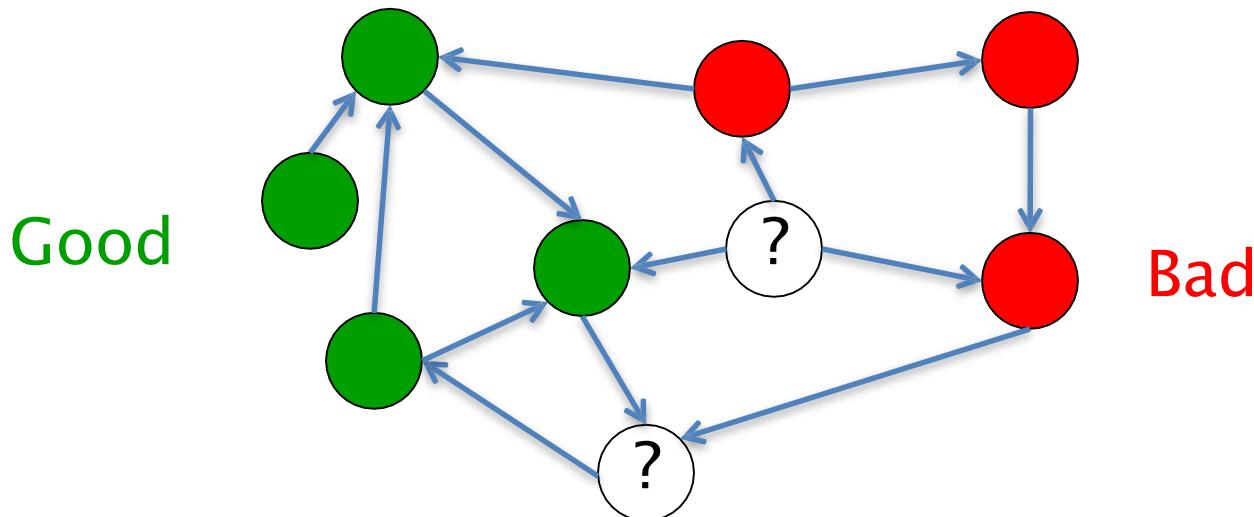
Simple iterative logic

- Good nodes won't point to Bad nodes
 - If you point to a Bad node, you're Bad
 - If a Good node points to you, you're Good



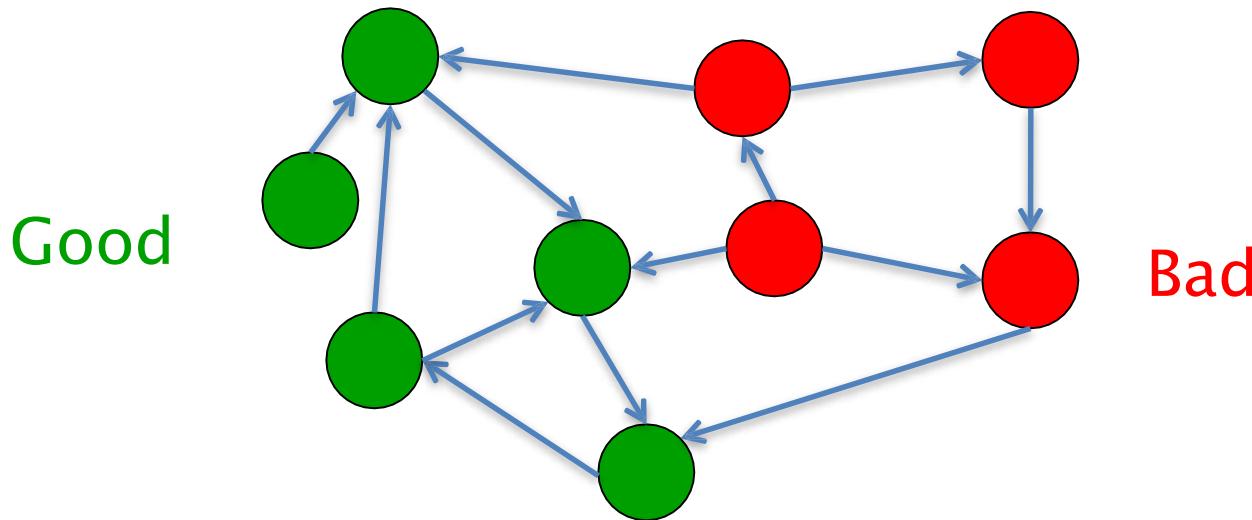
Simple iterative logic

- **Good** nodes won't point to **Bad** nodes
 - If you point to a **Bad** node, you're **Bad**
 - If a **Good** node points to you, you're **Good**



Simple iterative logic

- Good nodes won't point to Bad nodes
 - If you point to a Bad node, you're Bad
 - If a Good node points to you, you're Good



Sometimes need probabilistic analogs – e.g., mail spam

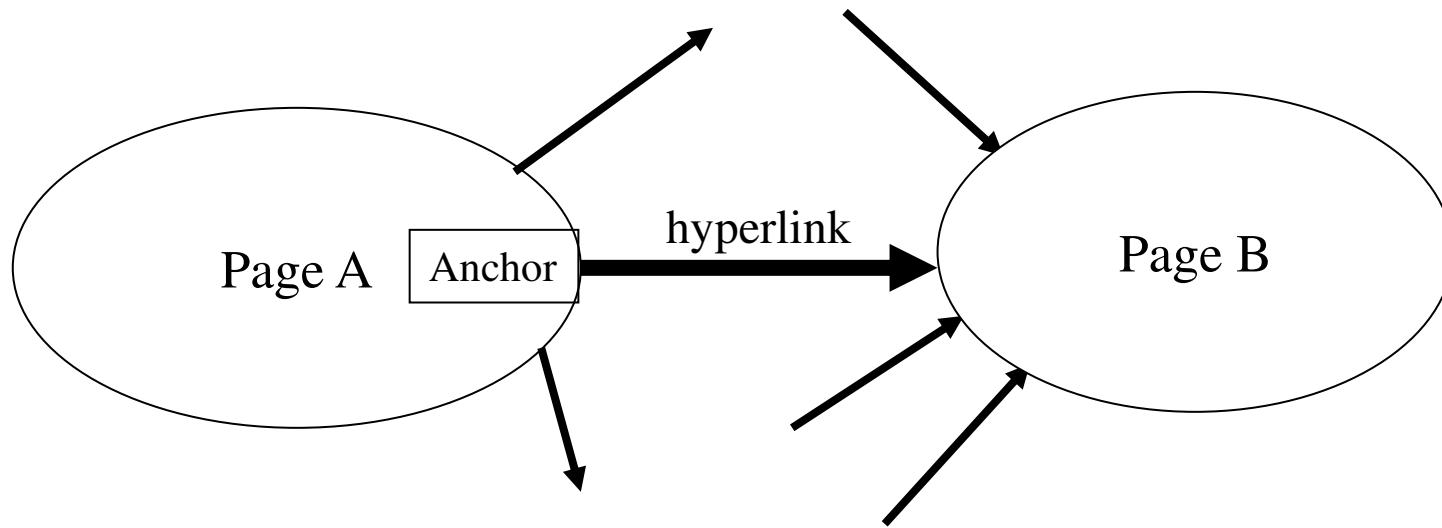
Many other examples of link analysis

- Social networks are a rich source of grouping behavior
- E.g., Shoppers' affinity – Goel+Goldstein 2010
 - **Consumers whose friends spend a lot, spend a lot themselves**
- <http://www.cs.cornell.edu/home/kleinber/networks-book/>

Link analysis

- Using hyperlinks for ranking web search results
- It is one of the factors used by search engines to compute a composite score for a web page on any given query
- Link analysis for most IR functionality thus far based purely on text
 - Scoring and ranking
 - Link-based clustering – topical structure from links
 - Links as features in classification – documents that link to one another are likely to be on the same subject
- Crawling
 - Based on the links seen, where do we crawl next?

Web as a Graph



Assumption 1: A hyperlink is a quality signal.

- The hyperlink $d_1 \rightarrow d_2$ indicates that d_1 's author deems d_2 high-quality and relevant.

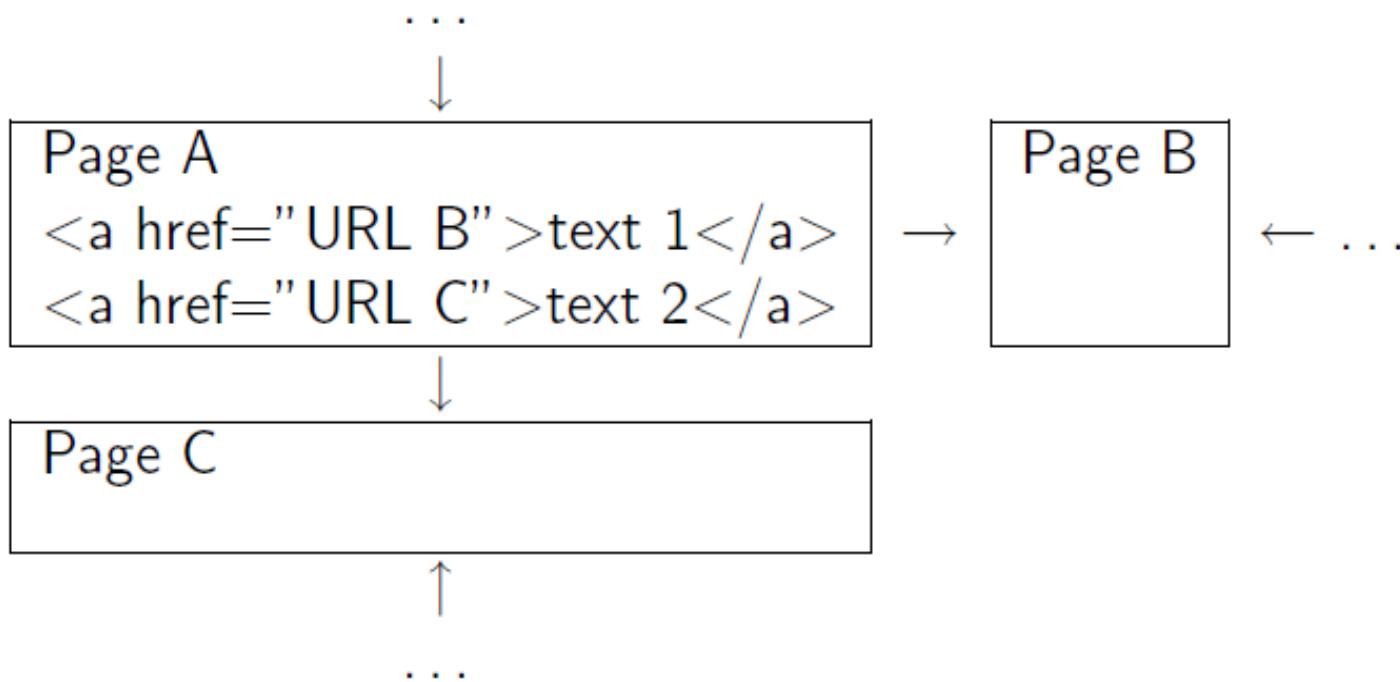
Assumption 2: The anchor text describes the content of d_2 .

- We use anchor text somewhat loosely here for: the text surrounding the hyperlink.

Example: “You can find cheap cars here.”

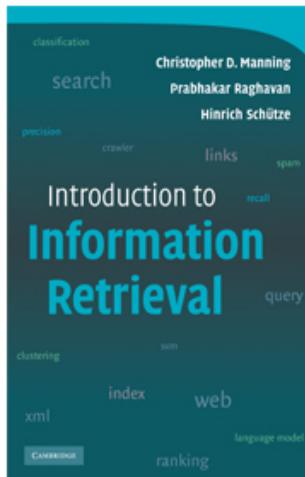
Anchor text: “You can find cheap cars here”

Web as a Graph



Assumption 1: reputed sites

Introduction to Information Retrieval



This is the companion website for the following book.

Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, *Introduction to Information Retrieval*

You can order this book at [CUP](#), at your local bookstore or on the internet. The best search

The book aims to provide a modern approach to information retrieval from a computer science perspective. It is available at [University of Cambridge](#) and at the [University of Stuttgart](#).

We'd be pleased to get feedback about how this book works out as a textbook, what is missing, and so on. Please send comments to: informationretrieval (at) yahoogroups (dot) com

Assumption 2: annotation of target

東北大学
TOHOKU UNIVERSITY

中文 | 한국어 | **English** | 日本語

お問い合わせ

大学概要 学部・大学院・研究所 教育・学生支援 國際交流 研究・産学連携

東北大学
TOHOKU UNIVERSITY

Chinese | Korean | English | Japanese

Inquiry Access Sitemap

About Tohoku University Faculties, Schools and Institutes Campus Life International Exchange Research and Cooperation Disclosure and Public Information Entrance Exam Information

Prospective Students

General Public

Corporations

Alumni

Current Students

Faculty and Staff (Internal use)

New! Video Channel

TOHOKU UNIVERSITY

Click Here

東北大学入学式(平成23年5月)

Anchor Text

- Searching on [text of d2] + [anchor text → d2] is often more effective than searching on [text of d2] only.
- Example: Query IBM
 - Matches IBM's copyright page
 - Matches many spam pages
 - Matches IBM Wikipedia article
 - May not match IBM home page, if IBM home page is mostly graphics
- Searching on [anchor text → d2] is better for the query IBM
 - In this representation, the page with the most occurrences of IBM is **www.ibm.com**.

Anchor Text

- **Anchor text containing IBM pointing to www.ibm.com**

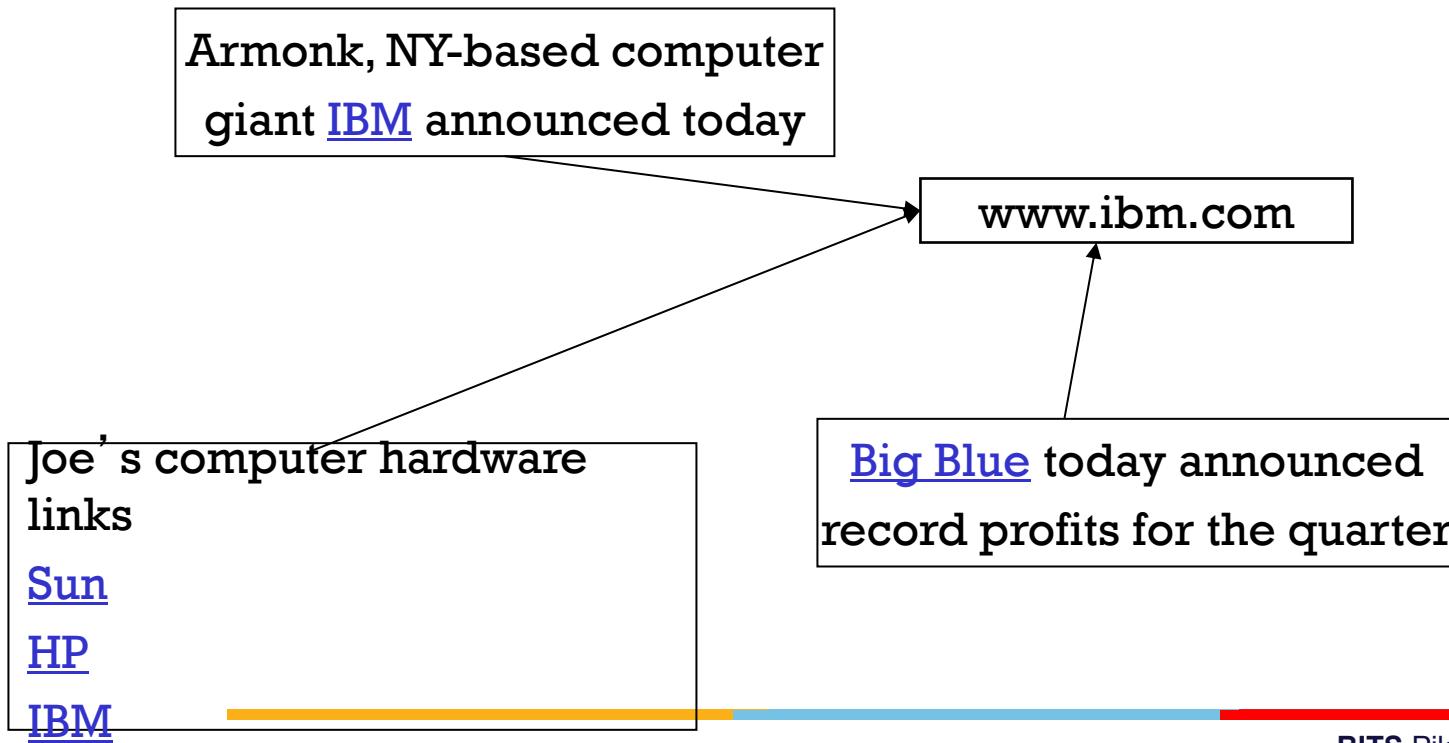
www.nytimes.com: “IBM acquires Webify”

www.slashdot.org: “New IBM optical chip”

www.stanford.edu: “IBM faculty award recipients”.

Indexing anchor text

- Thus: Anchor text is often a better description of a page's content than the page itself
- Anchor text can be weighted more highly than document text.



Indexing anchor text

- Can sometimes have unexpected effects, e.g., spam, **miserable failure**
- Can score anchor text with weight depending on the authority of the anchor page's website
 - E.g., if we were to assume that content from cnn.com or yahoo.com is authoritative, then trust (more) the anchor text from them.

Connectivity servers

- Quality of a page is a function of the links that points to it
- For link analysis, queries on the connectivity of the web graph are needed
- Support for fast queries on the web graph
 - Which URLs point to a given URL ?
 - Which URLs are pointed by a given URL ?
- Mappings stored:
 - From URL to out-links
 - From URL to in-links

Applications

- Link analysis
- Web graph analysis
 - Connectivity, crawl optimization
- Crawl control

Page Rank

- PageRank: Scoring measure based on the link structure of web pages
- Every node in the web graph is given a score between 0 and 1, depending on its in and out-links
- Given a query, a search engine combines the PageRank score with other values to compute the ranked retrieval (e.g. cosine similarity, relevance feedback, etc.)

Model behind PageRank: Random walk

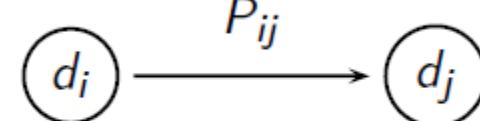
- Imagine a web surfer doing a random walk on the web
- Start at a random page
- At each step, go out of the current page along one of the links on that page, equiprobably
- In the steady state, each page has a long-term visit rate
- This long-term visit rate is the page's PageRank
- PageRank = long-term visit rate = steady state probability.

Markov chains

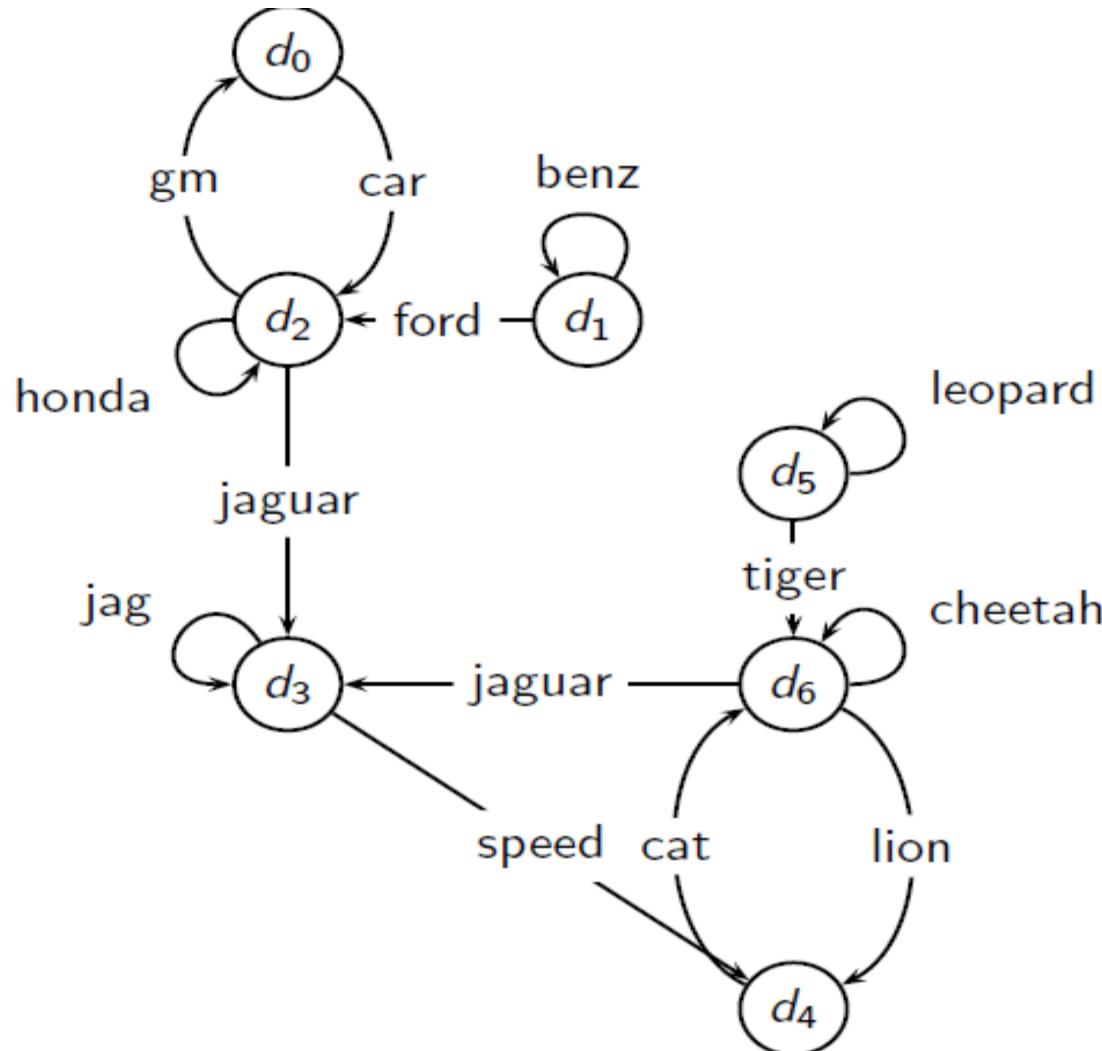
- Formalization of random walk:
- Markov chain is a ***discrete-time stochastic process***: a process that occurs in a series of time-steps in each of which a random choice is made.
- A Markov chain consists of N states, plus an $N \times N$ transition probability matrix P .
- state = page
- At each step, we are on exactly one of the pages.

For $1 \leq i, j \leq N$, the matrix entry P_{ij} tells us the probability of j being the next page, given we are currently on page i .

Clearly, for all i , $\sum_{j=1}^N P_{ij} = 1$



Example web graph



Link matrix for example

	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0	0	1	0	0	0	0
d_1	0	1	1	0	0	0	0
d_2	1	0	1	1	0	0	0
d_3	0	0	0	1	1	0	0
d_4	0	0	0	0	0	0	1
d_5	0	0	0	0	0	1	1
d_6	0	0	0	1	1	0	1

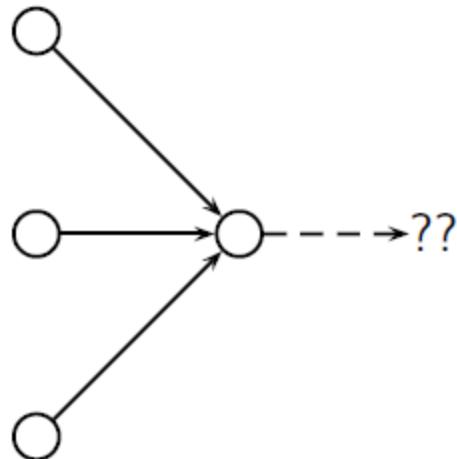
Transition probability matrix P

	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0.00	0.00	1.00	0.00	0.00	0.00	0.00
d_1	0.00	0.50	0.50	0.00	0.00	0.00	0.00
d_2	0.33	0.00	0.33	0.33	0.00	0.00	0.00
d_3	0.00	0.00	0.00	0.50	0.50	0.00	0.00
d_4	0.00	0.00	0.00	0.00	0.00	0.00	1.00
d_5	0.00	0.00	0.00	0.00	0.00	0.50	0.50
d_6	0.00	0.00	0.00	0.33	0.33	0.00	0.33

Long-term visit rate

- PageRank = long-term visit rate
- Long-term visit rate of page **d** is the probability that a web surfer is at page **d** at a given point in time
- The web graph must correspond to an ergodic Markov chain
- First a special case: The web graph must not contain dead ends.

Dead ends



- The web is full of dead ends
- Random walk can get stuck in dead ends
- If there are dead ends, long-term visit rates are not well-defined.

Teleporting

- Teleporting – to get us out of dead ends
- At a dead end, jump to a random web page with prob. $1/N$
- At a non-dead end, the surfer invokes the teleport operation with probability $0 < \alpha < 1$ and the standard random walk (follow an out-link chosen uniformly at random as in) with probability $1 - \alpha$, where α is a fixed parameter chosen in
 - For example, if the page has 4 outgoing links: randomly choose one with probability $(1-0.10)/4=0.225$
- 10% is a parameter, the teleportation rate
- “jumping” from dead end is independent of teleportation rate.

Teleporting

- With teleporting, we cannot get stuck in a dead end
- But even without dead ends, a graph may not have well-defined long-term visit rates
- More generally, we require that the Markov chain be ergodic.

Two formulas

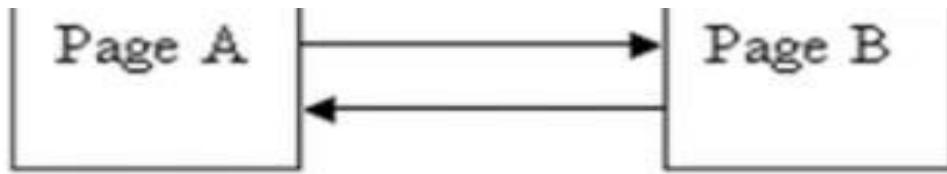
$$PR(A) = (1-d) + d(PR(T_1)/L(T_1) + \dots + PR(T_n)/L(T_n))$$

$$PR(A) = (1-d)/N + d(PR(T_1)/L(T_1) + \dots + PR(T_n)/L(T_n))$$

Where T_1, T_2, \dots, T_n are pages that are pointing to A.

How is calculated?

- The PR of each page depends on the PR of the pages pointing to it.
- But we won't know what PR those pages have until the pages pointing to them have their PR calculated and so on.
- So what we do is make a guess.



- Each page has one outgoing link. So that means $L(A) = 1$ and $L(B) = 1$.

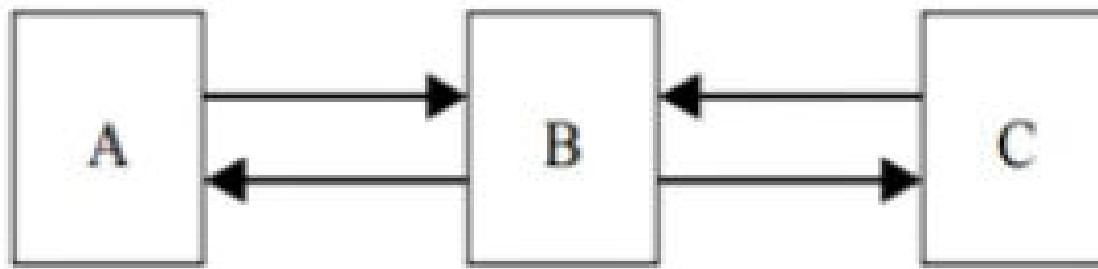
- d (damping factor) = 0.85
- $PR(A) = (1 - d) + d(PR(B)/1)$
- $PR(B) = (1 - d) + d(PR(A)/1)$

- $\text{PR(A)} = 0.15 + 0.85 * 0$
= 0.15
= $0.15 + 0.85 * 1$
 $\text{PR(B)} = 1$
- Now we have calculated a "next best guess" so we just plug it in the equation again...
- $\text{PR(A)} = 0.15 + 0.85 * 1$
= 1
- $\text{PR(B)} = 0.15 + 0.85 * 0.15$
= 0.2775

And again...

- $\text{PR(A)} = 0.15 + 0.85 * 0.2775$
= 0.385875
- $\text{PR(B)} = 0.15 + 0.85 * 1$
= 1

Example2



The number of web pages $N = 3$.

Consider the damping parameter $d = 0.7$. (Consider the page rank calculations by the second formula.)

$$PR(A) = (1 - d) \times (1 / N) + d \times (PR(B) / 2)$$

$$PR(B) = (1 - d) \times (1 / N) + d \times (PR(A) / 1 + PR(C) / 1)$$

$$PR(C) = (1 - d) \times (1 / N) + d \times (PR(B) / 2)$$

$$\text{So } PR(A) = 0.1 + 0.35 \times PR(B)$$

$$PR(B) = 0.1 + 0.70 \times PR(A) + 0.70 \times PR(C)$$

$$PR(C) = 0.1 + 0.35 \times PR(B)$$

By solving the above system of linear equations, we get

$$PR(A) = 0.2647$$

$$PR(B) = 0.4706$$

$$PR(C) = 0.2647.$$

Ergodic Markov chains

- **Definition:** A Markov chain is said to be *ergodic* if there exists a positive integer T_0 such that for all pairs of states i, j in the Markov chain, if it is started at time 0 in state i then for all $t > T_0$, the probability of being in state j at time t is greater than 0
- A Markov chain is ergodic iff it is **irreducible** and **aperiodic**
- **Irreducibility.** Roughly: there is a path from any page to any other page
- **Aperiodicity.** Roughly: The pages cannot be partitioned such that the random walker visits the partitions sequentially

Ergodic Markov chains

- **Theorem:** For any ergodic Markov chain, there is a unique long-term visit rate for each state
- This is the steady-state probability distribution
- Over a long time period, we visit each state in proportion to this rate
- It doesn't matter where we start
- Teleporting makes the web graph ergodic
 - Web-graph+teleporting has a steady-state probability distribution
 - Each page in the web-graph+teleporting has a PageRank.

Issues with Page rank algorithm

1. PageRank scores do not reflect current events.
2. inability to handle queries containing natural language and information outside of keywords.

HITS – Hyperlink-Induced Topic Search

There are two different types of relevance on the web

- **Hubs:** A hub page is a good list of links to pages answering the information need
 - E.g., for query [chicago bulls]: Alice's list of recommended resources on the Chicago Bulls sports team
- **Authorities:** An authority page is a direct answer to the information need
 - The home page of the Chicago Bulls sports team
 - By definition: Links to authority pages occur repeatedly on hub pages
- Most approaches to search (including PageRank ranking) don't make the distinction between these two types of relevance.

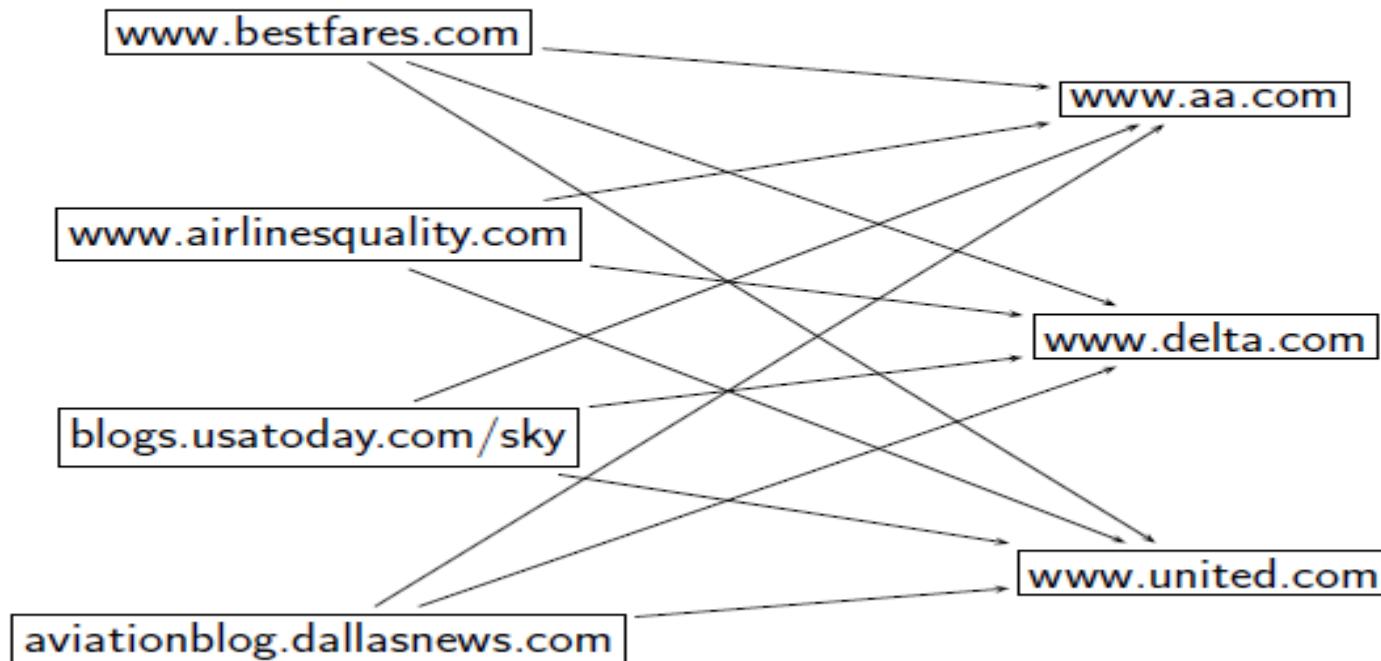
Hubs and authorities: Definition

- A good hub page for a topic links to many authority pages for that topic
- A good authority page for a topic is linked to by many hub pages for that topic
- Appear to have a circular definition of hubs and authorities; we will turn this into an iterative computation.

Example

hubs

authorities



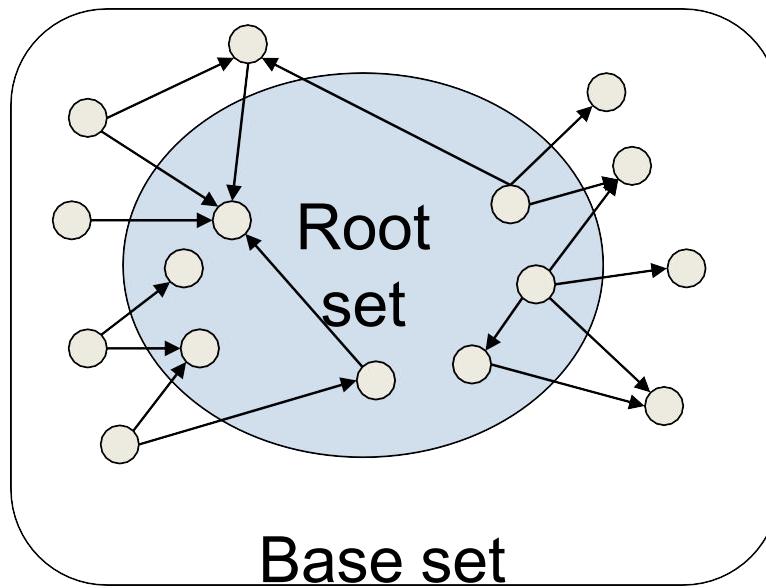
High-level scheme

- . Extract from the web a base set of pages that *could* be good hubs or authorities.
- . From these, identify a small set of top hub and authority pages;
→ iterative algorithm.

Base set

- . Given text query (say ***browser***), use a text index to get all pages containing ***browser***.
 - . Call this the root set of pages.
- . Add in any page that either
 - . points to a page in the root set, or
 - . is pointed to by a page in the root set.
- . Call this the base set.

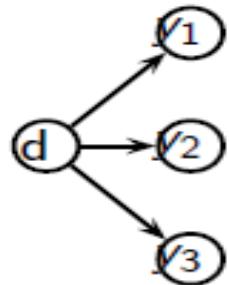
Visualization



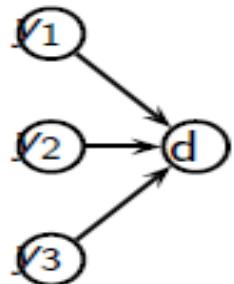
Get in-links (and out-links) from a *connectivity server*

Hub and authority scores

For all d : $h(d) = \sum_{d \rightarrow y} a(y)$



For all d : $a(d) = \sum_{y \rightarrow d} h(y)$



Iterate these two steps until convergence

Hub and authority scores

- Compute for each page d , a hub score $h(d)$ and an authority score $a(d)$
- Initialization: for all d : $h(d) = 1$, $a(d) = 1$
- Iteratively update all $h(d)$, $a(d)$
- After convergence:
 - Output pages with highest h scores as top hubs
 - Output pages with highest a scores as top authorities
 - So we output two ranked lists

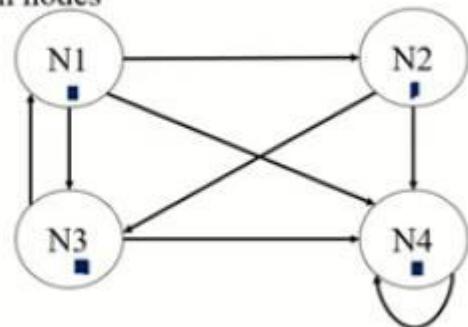
Calculate the hub and authority score using HITS algorithm k=3 the adjancy matrix is

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Adjacency matrix with nodes

	N1	N2	N3	N4
N1	0	1	1	1
N2	0	0	1	1
N3	1	0	0	1
N4	0	0	0	1

Graph with nodes



Ranks using out degree and in degree

Nodes	Out-degree(Hub)	In-degree(Authority)
N1	3	1
N2	2	1
N3	2	2
N4	1	4

Adjacency matrix A with nodes

	N1	N2	N3	N4
N1	0	1	1	1
N2	0	0	1	1
N3	1	0	0	1
N4	0	0	0	1

Transpose of Matrix A^T

	N1	N2	N3	N4
N1	0	0	1	0
N2	1	0	0	0
N3	1	1	0	0
N4	1	1	1	1

Authority weight vector , $v = A^T * u$

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} * \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Authority, v

$$\begin{pmatrix} 1 \\ 1 \\ 2 \\ 4 \end{pmatrix} \blacksquare$$

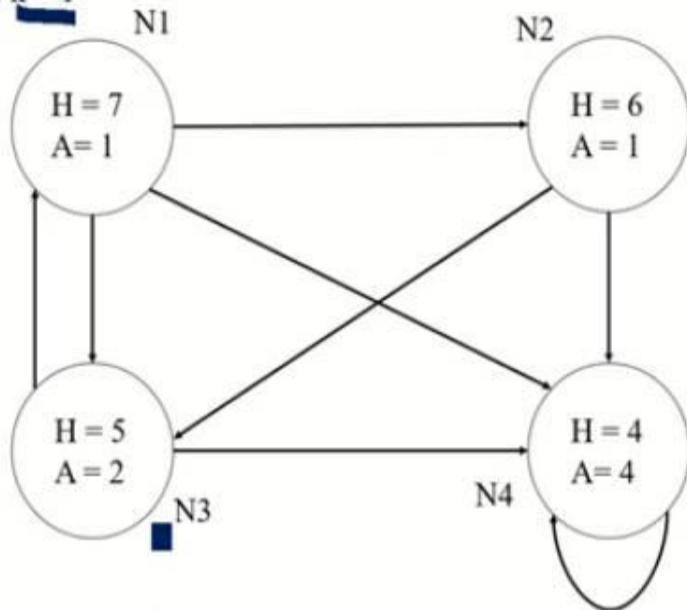
Updated Hub weight vector , $u = A * v$

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} 1 \\ 1 \\ 2 \\ 4 \end{pmatrix}$$

Hub, u

$$\begin{pmatrix} 7 \\ 6 \\ 5 \\ 4 \end{pmatrix}$$

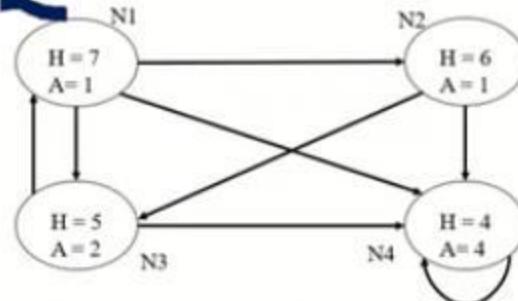
For $k = 1$



HUB: N1, N2, N3, N4

AUTHORITY: N4, N3, N2, N1 {TIE}

For $k = 1$



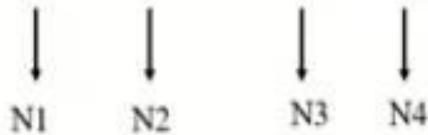
Nodes	Hub	Authority
N1	7	1
N2	6	1
N3	5	2
N4	4	4

Calculate new Authority from K = 1

$$v_1 = 1^2 + 1^2 + 2^2 + 4^2 = 22$$

$$= \frac{1}{\sqrt{22}}, \frac{1}{\sqrt{22}}, \frac{2}{\sqrt{22}}, \frac{4}{\sqrt{22}}$$

$$v_1 = 0.213, 0.213, 0.426, 0.853$$



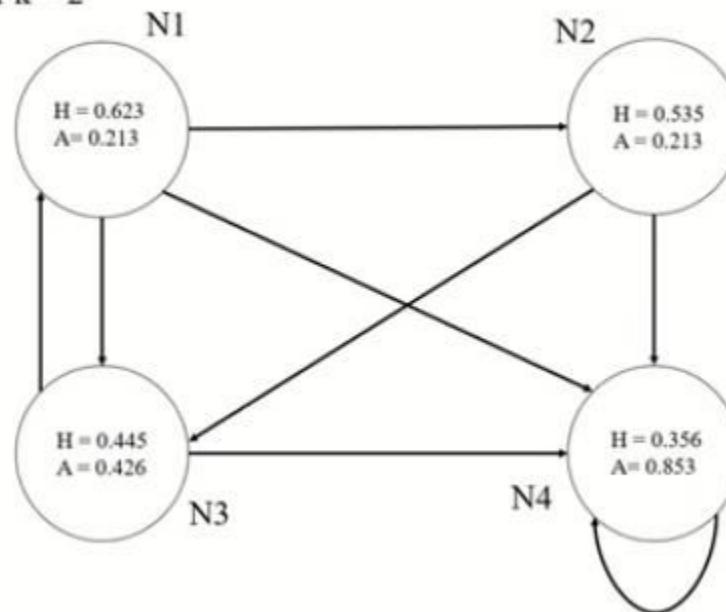
For k = 2

Calculate new hub from K = 1

$$u_1 = 7^2 + 6^2 + 5^2 + 4^2 = 126$$

$$= \frac{7}{\sqrt{126}}, \frac{6}{\sqrt{126}}, \frac{5}{\sqrt{126}}, \frac{4}{\sqrt{126}}$$

$$u_1 = 0.623, 0.535, 0.445, 0.356$$



For $k = 2$

Nodes	Hub	Authority
N1	0.623	0.213
N2	0.535	0.213
N3	0.445	0.426
N4	0.356	0.853

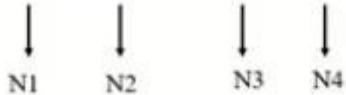
HUB: N1, N2, N3, N4

AUTHORITY: N4, N3, N2, N1 {TIE}

Calculate new Authority from K = 2

$$v_1 = 0.213^2 + 0.213^2 + 0.426^2 + 0.853^2 = 0.999$$
$$= \frac{0.213}{\sqrt{0.999}}, \frac{0.213}{\sqrt{0.999}}, \frac{0.426}{\sqrt{0.999}}, \frac{0.853}{\sqrt{0.999}}$$

$$v_1 = 0.213, 0.213, 0.426, 0.853$$



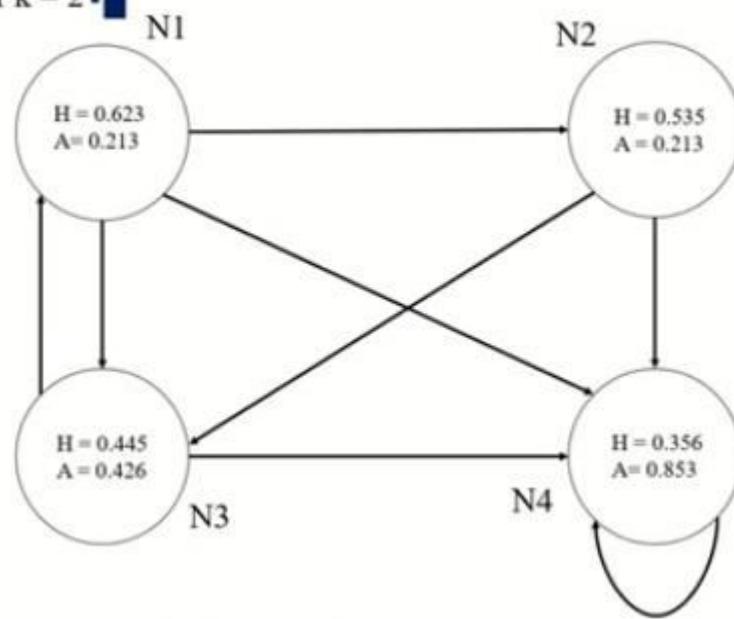
Calculate new hub from K = 2

$$u_1 = 0.623^2 + 0.535^2 + 0.445^2 + 0.356^2 = 0.999$$
$$= \frac{0.623}{\sqrt{0.999}}, \frac{0.535}{\sqrt{0.999}}, \frac{0.445}{\sqrt{0.999}}, \frac{0.356}{\sqrt{0.999}}$$

$$u_1 = 0.623, 0.535, 0.445, 0.356$$



For k = 2



Hubs and Authorities: Summary

- HITS can pull together good pages regardless of page content
- Once the base set is assembled, we only do link analysis, no text matching
- Pages in the base set often do not contain any of the query words.

Hubs and Authorities: Issues

- Topic Drift
 - Off-topic pages can cause off-topic “authorities” to be returned
 - E.g., the neighborhood graph can be about a “super topic”
- Mutually Reinforcing Affiliates
 - Affiliated pages/sites can boost each others’ scores
 - Linkage between affiliated pages is not a useful signal.



Thank You!



BITS Pilani
Pilani Campus

Information Retrieval

Dr. Vijayalakshmi anand



Session 13: Cross Language Information Retrieval

These slides are prepared by the instructor Prof. Manning,, with grateful acknowledgement of and many others who made their course materials freely available online.

What we will cover in this session

(R3 Chapter 2)

- Language problems in IR
- Translation Approaches
- Handling Many Languages
- Resources for CLIR

Cross Language Information Retrieval (CLIR)



“A subfield of information retrieval dealing with retrieving information written in a language different from the language of the user's query.”

E.g., Using Hindi queries to retrieve English documents

Also called *multi-lingual*, *cross-lingual*, or *trans-lingual* IR.

Cross Language Information Retrieval (CLIR)



- Accepting questions in one language (English) and retrieving information in a variety of other languages
 - – “questions” may be typical Web queries or full questions in across-lingual question answering (QA) system
 - – “information” could be news articles, text fragments or passages, factual answers, audio broadcasts, written documents, images, etc.
- Searching distributed, unstructured, heterogeneous, multilingual data
- Often combined with summarization, translation, and discovery technology

Why CLIR?

E.g., On the web, we have:

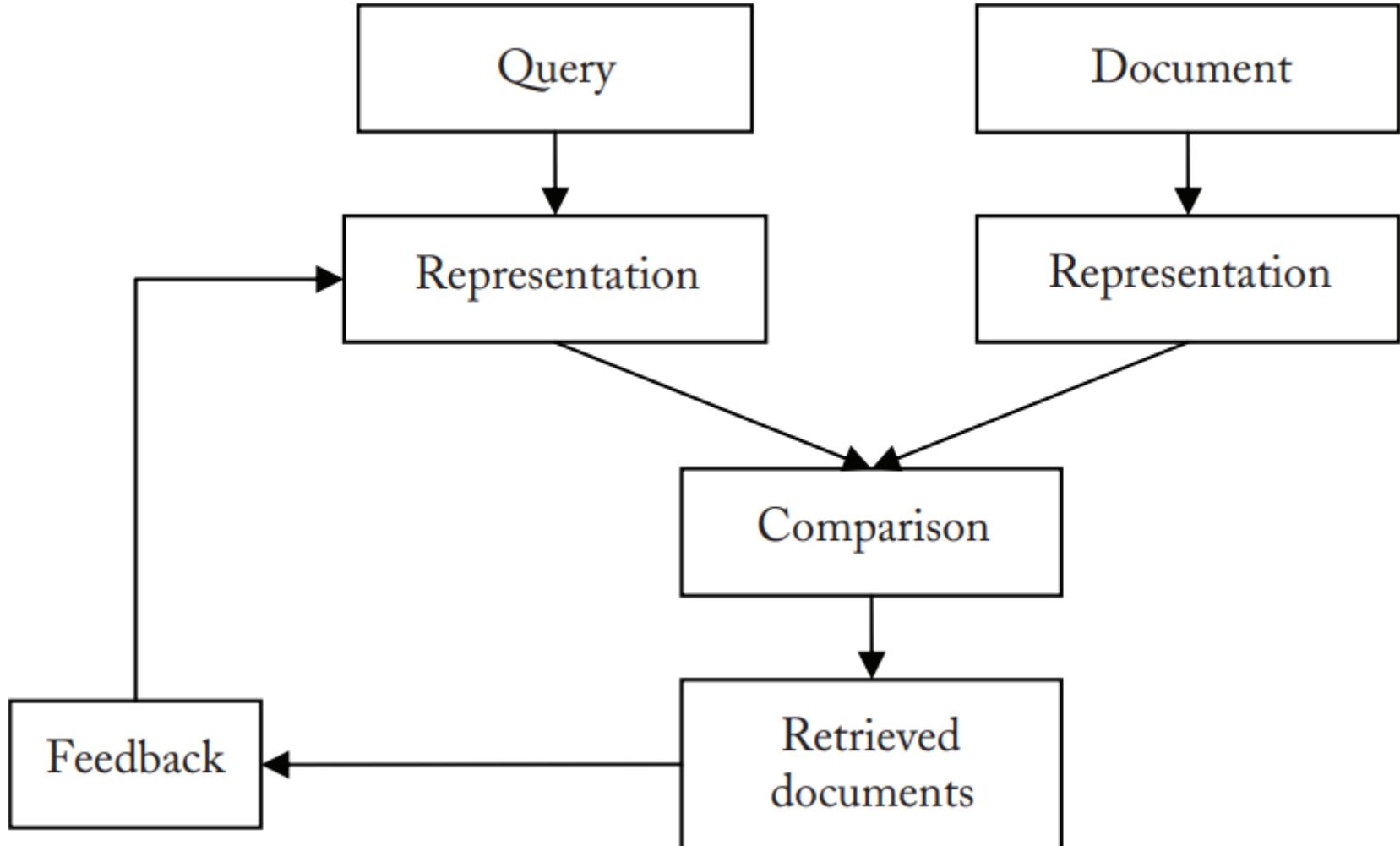
Documents in different languages

Multilingual documents

Images with captions in different languages

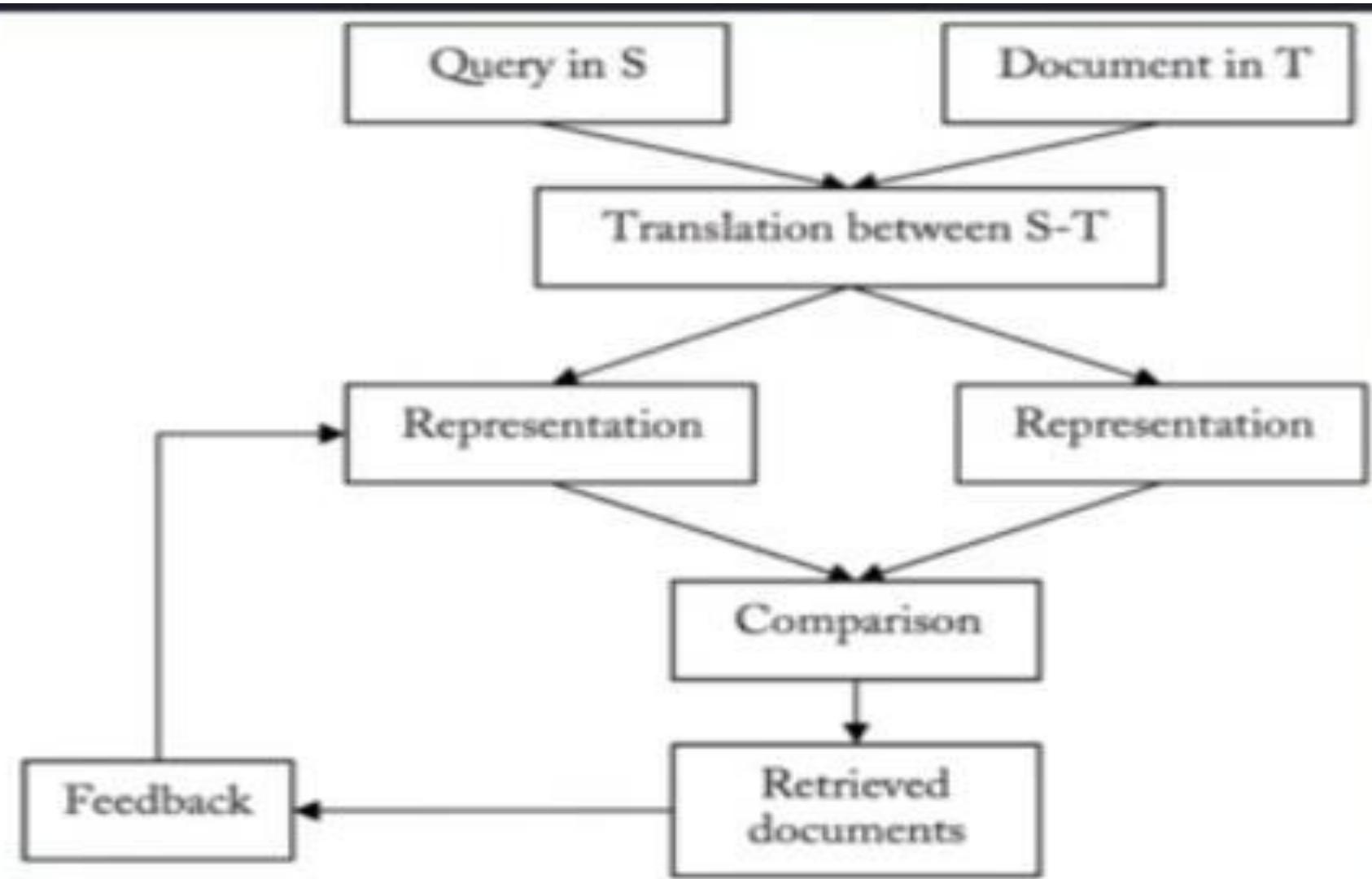
A single query should retrieve all such resources.

Typical IR System



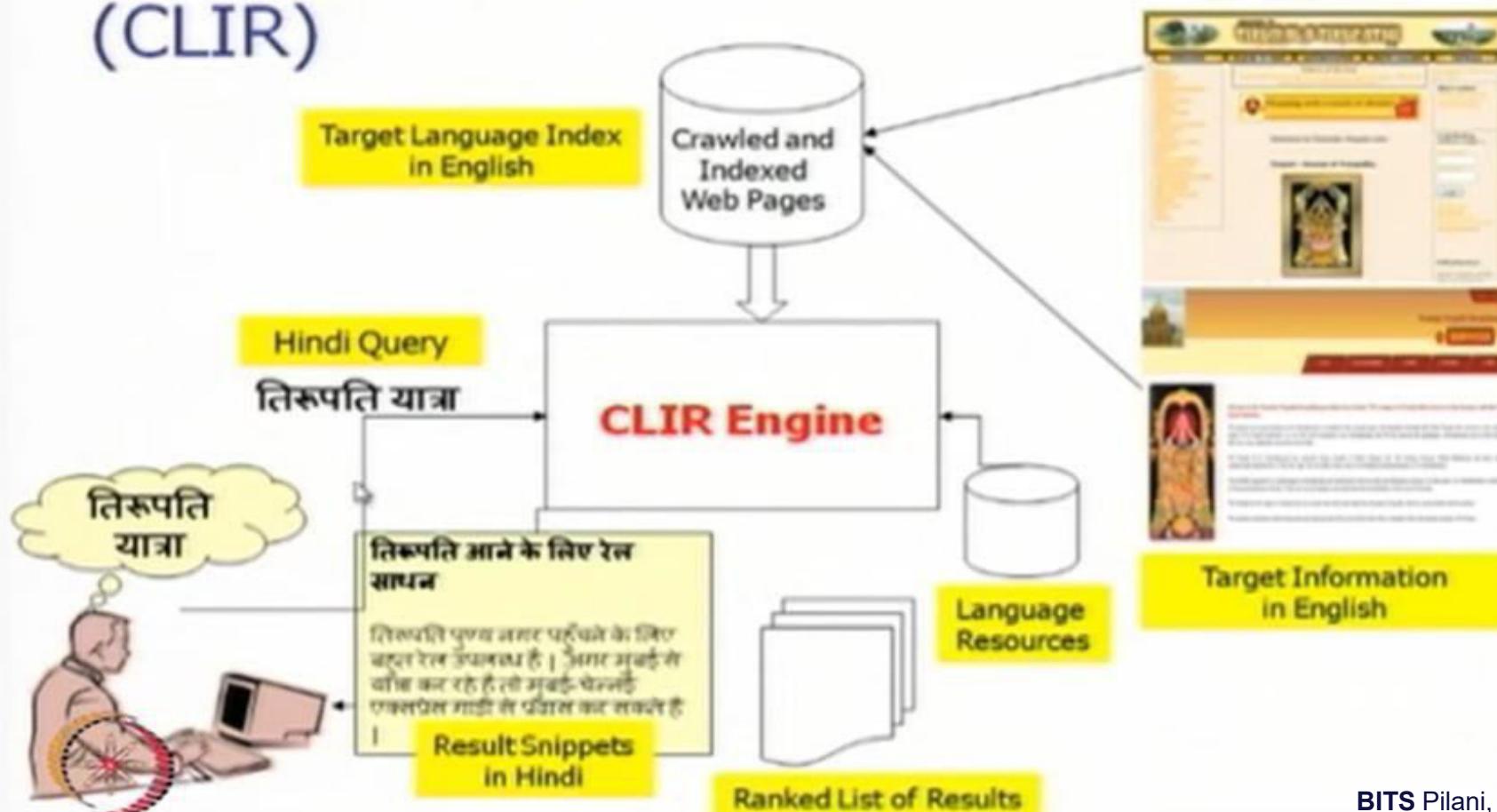
Typical IR System

- Typical queries are formed by 2–3 words.
- Both the query and the document, similar processes of indexing are carried out in order to “understand,” to some extent, what the user desires to find and what the document talks about
- Given a query representation and a document representation, a relevance score is determined for each document according to how strongly the document representation corresponds to that of the query.
- Score intends to reflect the degree of relevance of the document to the user’s information need



Example

Cross Language Information Retrieval (CLIR)



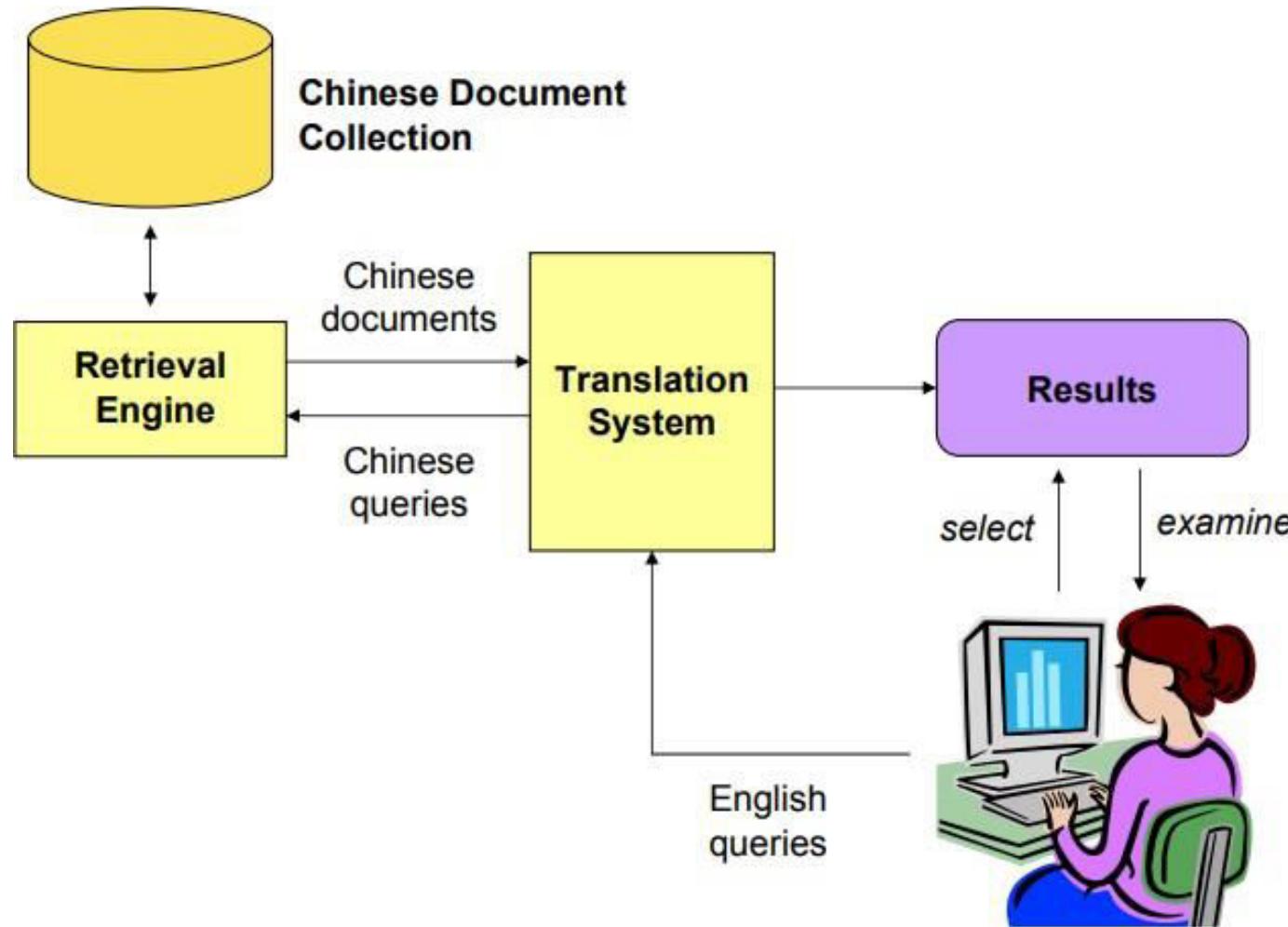
Approaches to CLIR

Document translation is to map the document representation into the query representation space, as illustrated in Figure 2.

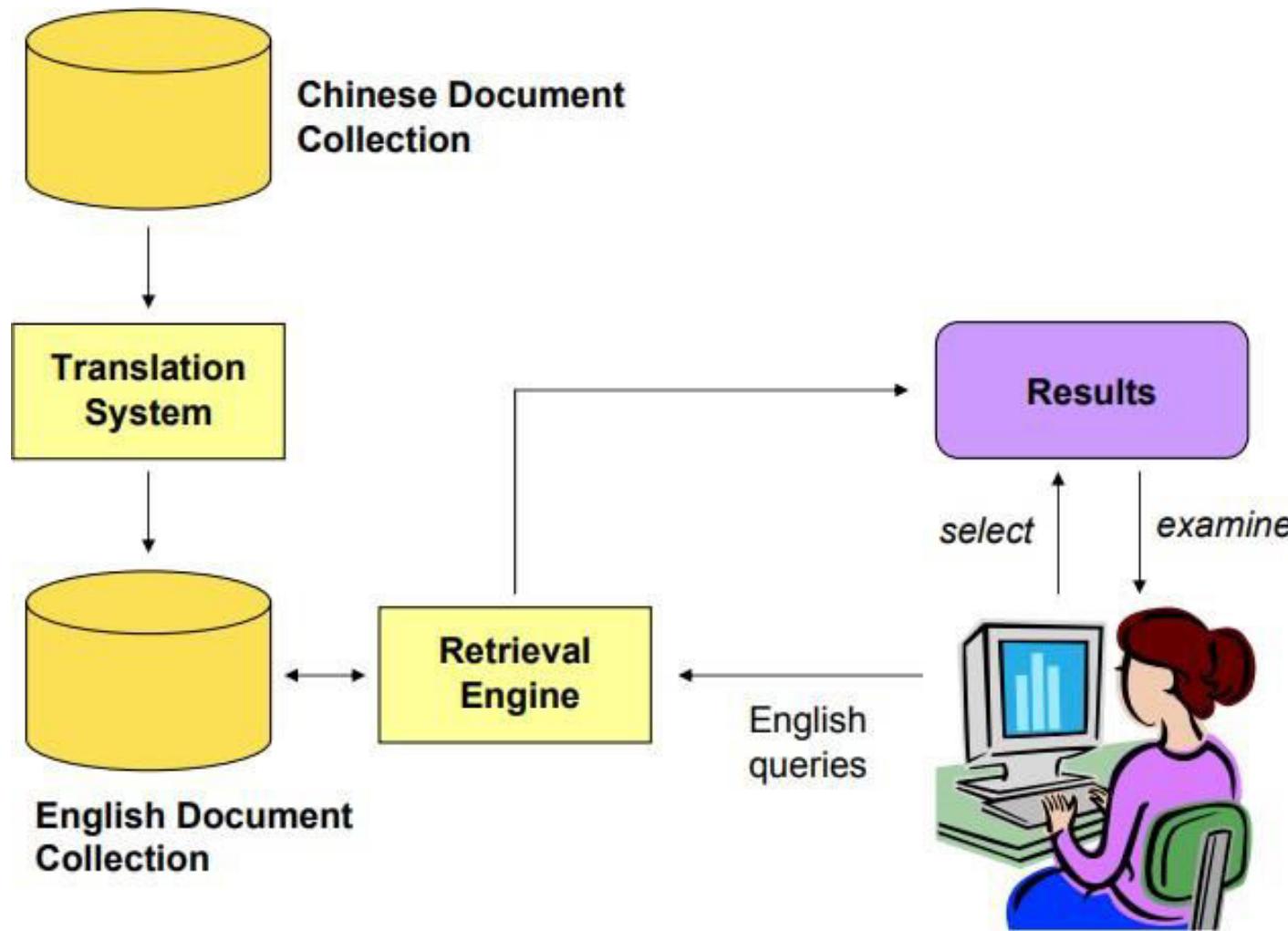
Query translation is to map the query representation into the document representation space, as illustrated in Figure 3.

Pivot language or Interlingua is to map both document and query representations to a third space.

Query Translation



Document Translation



Query versus Document Translation

Query Translation

- Often easier
- Disambiguation of query terms may be difficult with short queries
- Translation of documents must be performed at query time

Document Translation

- Documents can be translated and stored offline
- Automatic translation can be slow

Which is better?

- Often depends on the availability of language-specific resources (e.g., morphological analyzers)
- Both approaches present challenges for interaction

Non-statistical approach

Interlingua approaches

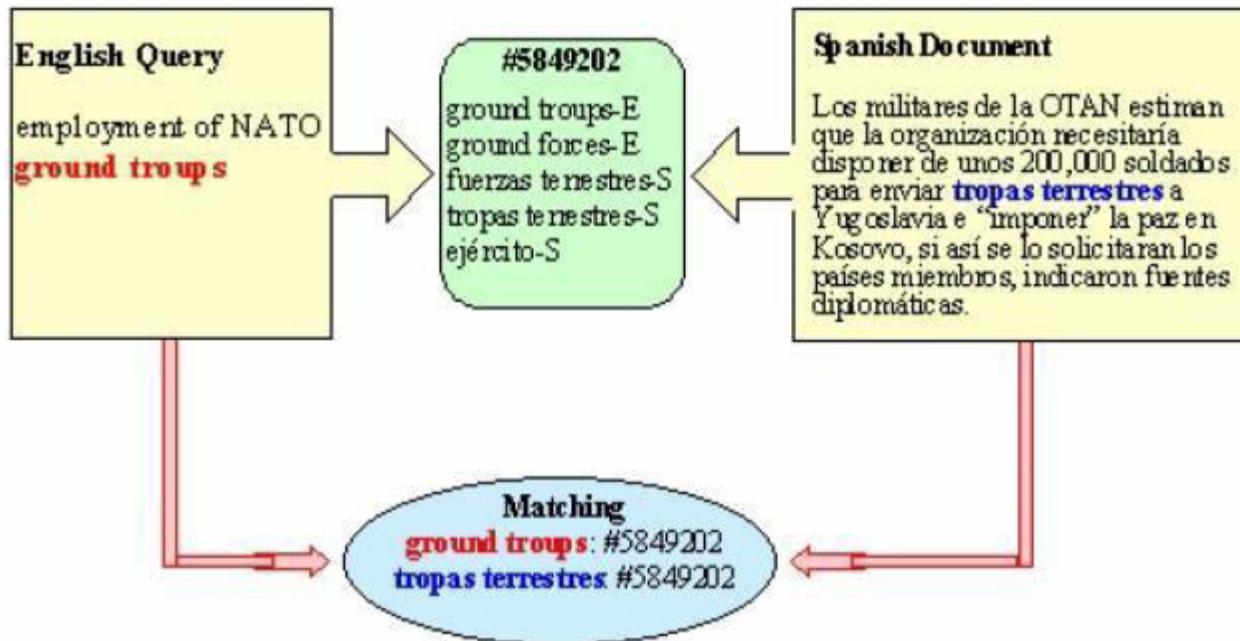
- Translate query into special language
- Translate all documents into same language
- Compare directly
- Cross-language retrieval becomes monolingual retrieval

Choice of interlingua?

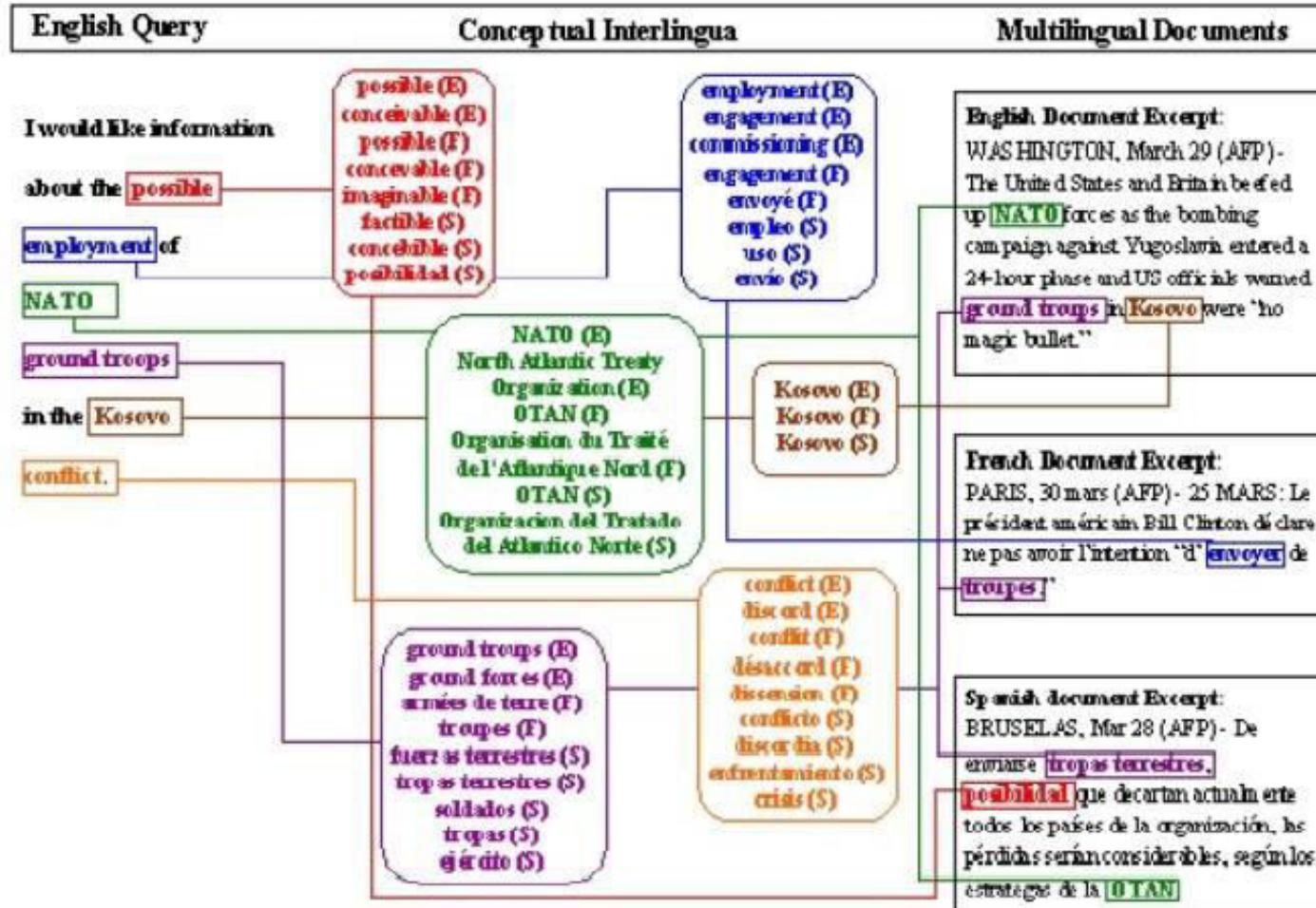
- Could use an existing language (e.g., English)
- Create own

Textwise created a “conceptual interlingua”

Conceptual Interlingua for Document Retrieval



[Liddy, Infonortics 1999]



[Liddy, Infonortics 1999]

Challenges in CLIR

- Indexing, retrieval and ranking of multilingual documents
- Web data is not clean and regular
 - ↳ ◊ Different font encodings – some of them proprietary
 - ◊ Spelling variations very common
 - ◊ Different document encodings
- Language identification needed to invoke appropriate language analyzers
- Involves a number of fundamental NLP research problems like query disambiguation, machine transliteration, named-entity recognition, multi-word recognition Ex: Good Friday does not mean Friday which is good

- Manually encoding comprehensive bilingual lexicons and transfer rules is difficult.
- SMT acquires knowledge needed for translation from a ***parallel corpus*** or ***bitext*** that contains the same set of documents in two languages.
- The Canadian Hansards (parliamentary proceedings in French and English) is a well-known parallel corpus.
- First align the sentences in the corpus based on simple methods that use coarse cues like sentence length to give bilingual sentence pairs.

Why Statistical Machine Translation?

- Insufficient amount of really good dictionaries.
 - Some linguistic information still needs to be set manually.
 - It is hard to deal with rule interactions in big systems, ambiguity, and idiomatic expressions.
 - Failure to adapt to new domains.
-

Noisy Channel Model

- Based on analogy to information-theoretic model used to decode messages transmitted via a communication channel that adds errors.
- Assume that source sentence was generated by a “noisy” transformation of some target language sentence and then use Bayesian analysis to recover the most likely target sentence that generated it.

Translate foreign language sentence $F = f_1, f_2, \dots, f_m$ to an English sentence $\hat{E} = e_1, e_2, \dots, e_I$ that maximizes $P(E | F)$

Bayesian Analysis of Noisy Channel



$$\begin{aligned}\hat{E} &= \operatorname{argmax}_{E \in English} P(E | F) \\ &= \operatorname{argmax}_{E \in English} \frac{P(F | E)P(E)}{P(F)} \\ &= \operatorname{argmax}_{E \in English} P(F | E)P(E)\end{aligned}$$

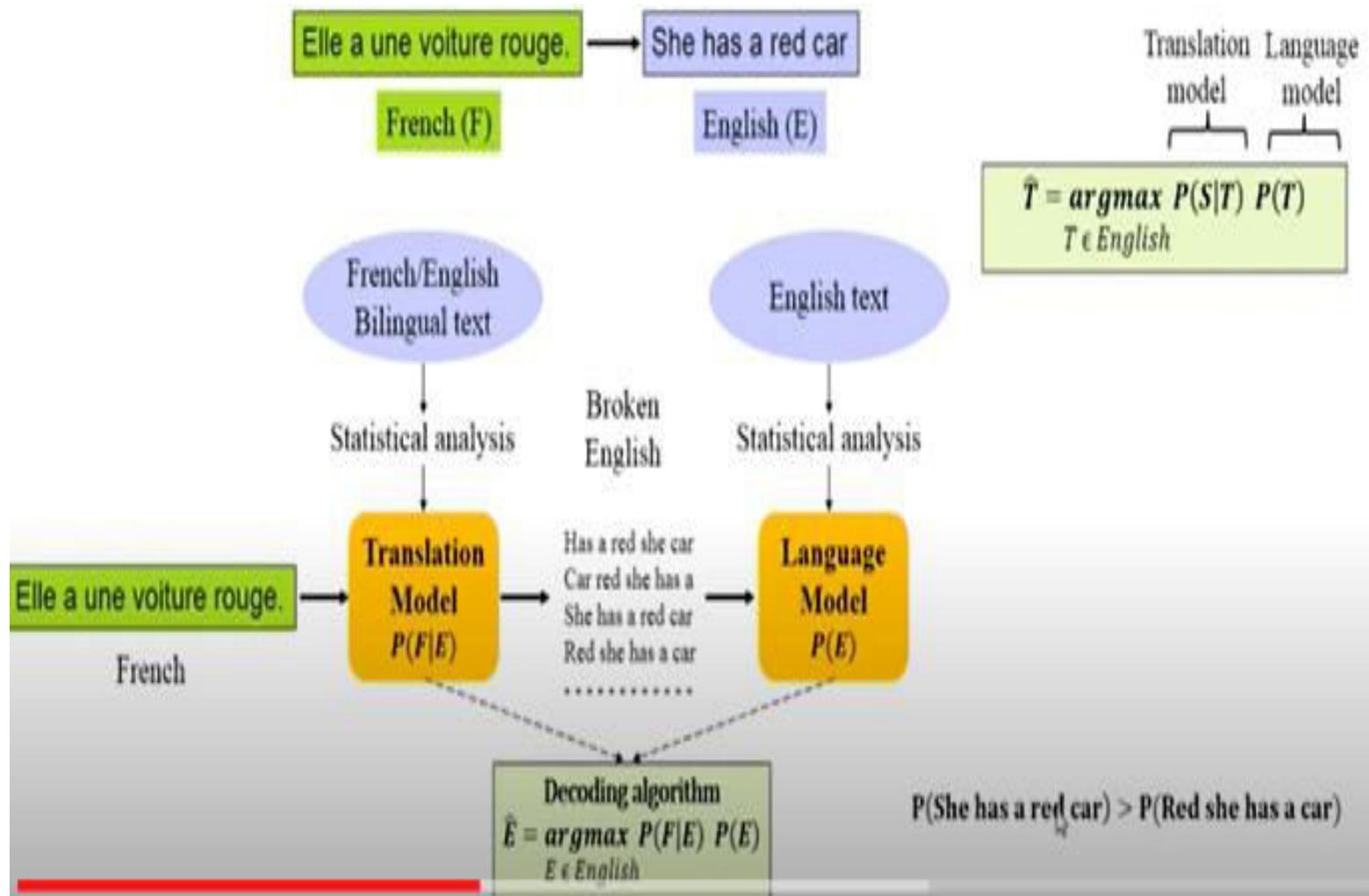
The diagram shows the term $P(F | E)P(E)$ decomposed into two parts: $P(F | E)$ and $P(E)$. Brackets under $P(F | E)$ and $P(E)$ are labeled "Translation Model" and "Language Model" respectively, written in red text below the equations.

Translation model: The goal is the translation of a text given in some language F into a target language E.

Language model: How probable the sentence T is in target language

Decoder: Given S the translation and language model produces the most probable T

Example



Three Aspects

- Modelling
 - Propose a probabilistic model for sentence translation
- Training
 - Learn the model parameters from data
- Decoding
 - Given a new sentence, use the learnt model to translate the input sentence .

Lexical Translation -Word based model



- How to translate a word → look up in dictionary
- Haus – house, building, home, household, shell.
- Multiple translations - some more frequent than others - for instance: house, and building most common

Translation model-Formulating the model

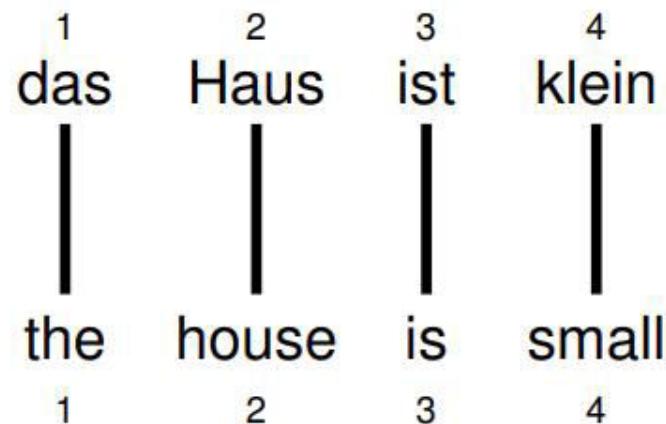
- Goal: build a model $p(\mathbf{e}|\mathbf{f})$
- where \mathbf{e} and \mathbf{f} are complete English and Foreign sentences.
- To help, we'll introduce an alignment \mathbf{a} to explain how \mathbf{f} generates \mathbf{e} in terms of word-level translation decisions.

$$p(\mathbf{e}|\mathbf{f}) = \sum_a p(\mathbf{e}, \mathbf{a}|\mathbf{f})$$

- If we can learn $p(\mathbf{e}|\mathbf{f})$ from data, we can use it to collect lexical translation probabilities, to align parallel sentences, or to translate new sentences.

Alignment

- In a parallel text (or when we translate), we align words in one language with the words in the other



- Word positions are numbered 1–4

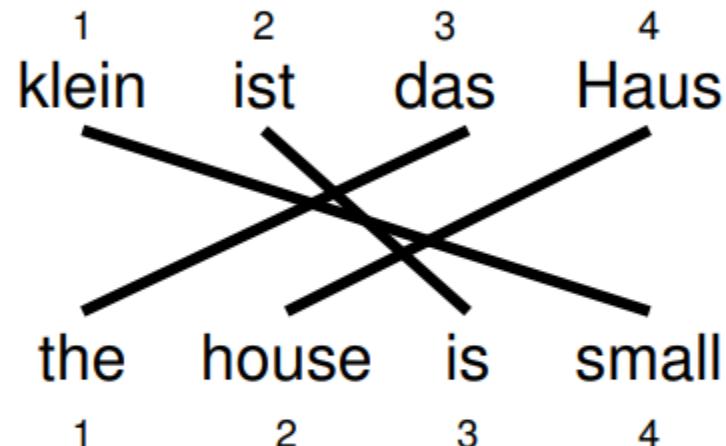
Alignment Function

- Formalizing alignment with an alignment function
- Mapping an English target word at position i to a German source word at position j with a function $a : i \rightarrow j$
- Example

$$a : \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 4\}$$

Reordering

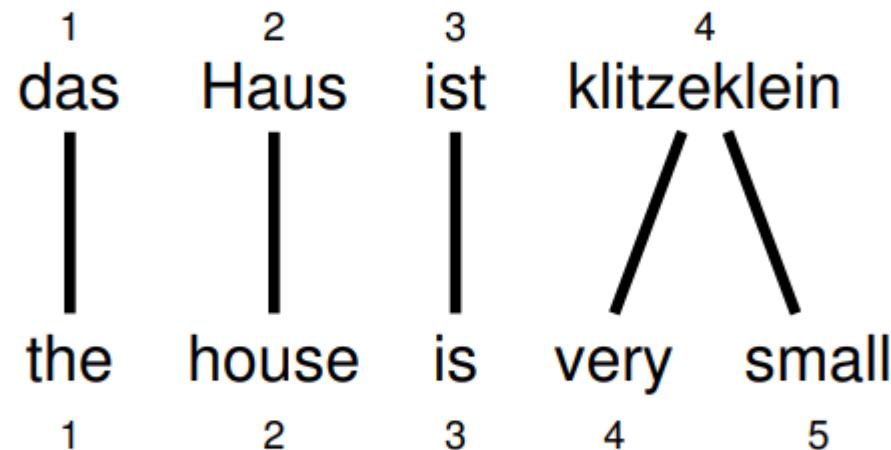
Words may be reordered during translation



$$a : \{1 \rightarrow 3, 2 \rightarrow 4, 3 \rightarrow 2, 4 \rightarrow 1\}$$

One to many translation

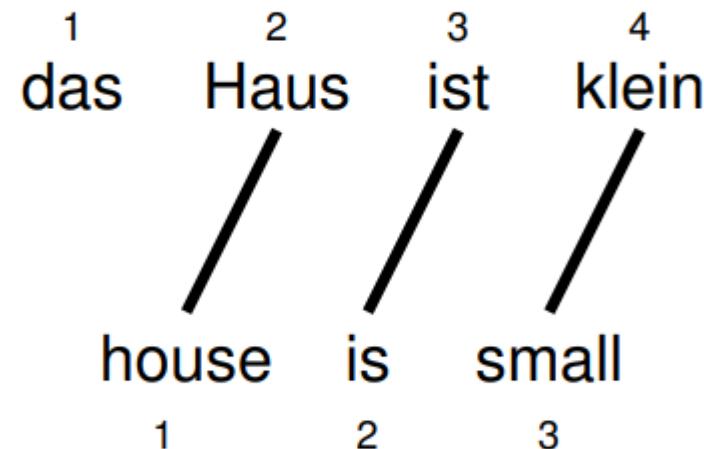
A source word may translate into multiple target words



$$a : \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 4, 5 \rightarrow 4\}$$

Dropped words

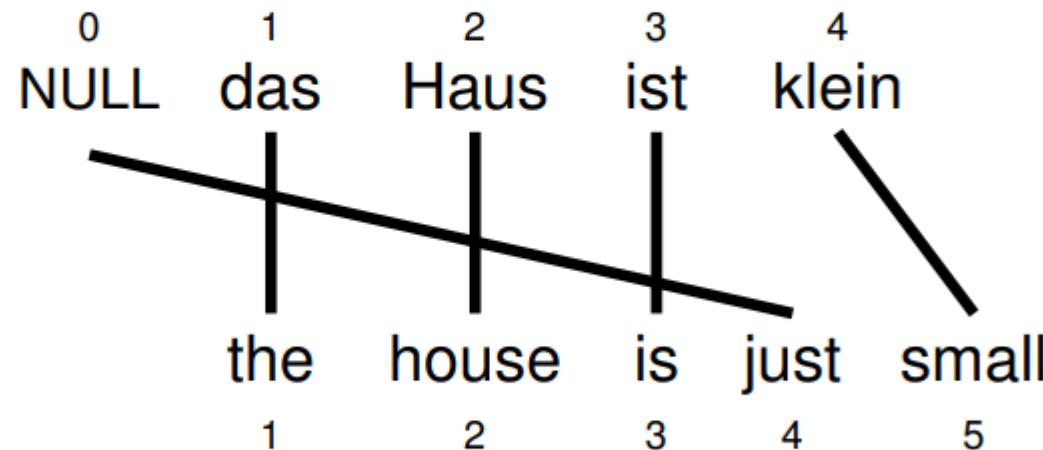
Words may be dropped when translated
(German article **das** is dropped)



$$a : \{1 \rightarrow 2, 2 \rightarrow 3, 3 \rightarrow 4\}$$

Inserting words

- Words may be added during translation
 - The English **just** does not have an equivalent in German
 - We still need to map it to something: special NULL token



$$a : \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 0, 5 \rightarrow 4\}$$

IBM model1

- Generative model: break up translation process into smaller steps
 - IBM Model 1 only uses lexical translation
- Translation probability
 - for a foreign sentence $\mathbf{f} = (f_1, \dots, f_{l_f})$ of length l_f
 - to an English sentence $\mathbf{e} = (e_1, \dots, e_{l_e})$ of length l_e
 - with an alignment of each English word e_j to a foreign word f_i according to the alignment function $a : j \rightarrow i$

$$p(\mathbf{e}, a | \mathbf{f}) = \frac{\epsilon}{(l_f + 1)^{l_e}} \prod_{j=1}^{l_e} t(e_j | f_{a(j)})$$

- parameter ϵ is a normalization constant

das

<i>e</i>	<i>t(e f)</i>
the	0.7
that	0.15
which	0.075
who	0.05
this	0.025

Haus

<i>e</i>	<i>t(e f)</i>
house	0.8
building	0.16
home	0.02
household	0.015
shell	0.005

ist

<i>e</i>	<i>t(e f)</i>
is	0.8
's	0.16
exists	0.02
has	0.015
are	0.005

klein

<i>e</i>	<i>t(e f)</i>
small	0.4
little	0.4
short	0.1
minor	0.06
petty	0.04

$$\begin{aligned}
 p(e, a|f) &= \frac{\epsilon}{5^4} \times t(\text{the}| \text{das}) \times t(\text{house}| \text{Haus}) \times t(\text{is}| \text{ist}) \times t(\text{small}| \text{klein}) \\
 &= \frac{\epsilon}{5^4} \times 0.7 \times 0.8 \times 0.8 \times 0.4 \\
 &= 0.0029\epsilon
 \end{aligned}$$

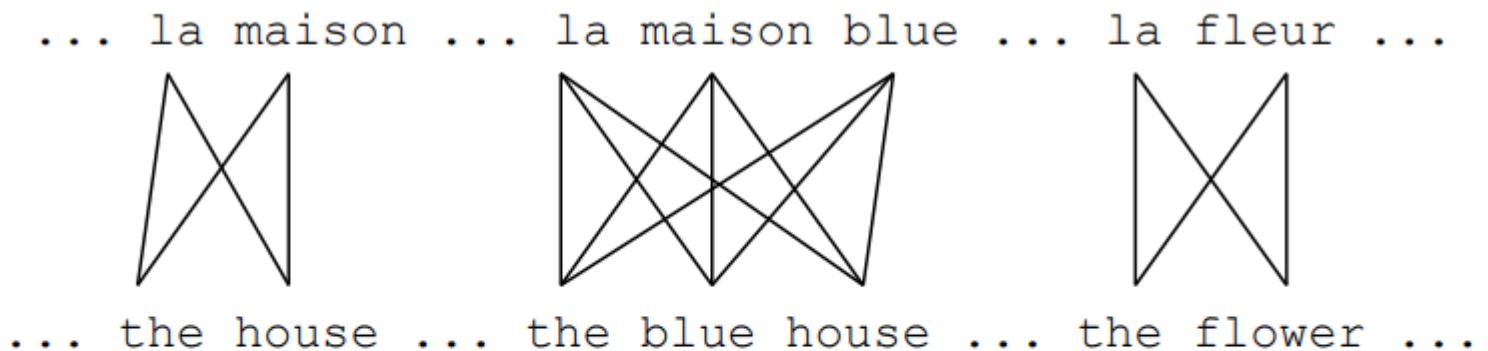
Model 1 Training

- If the alignments are known, the translation probabilities can be calculated simply by counting the aligned words.
- But, if translation probabilities were not known then the alignments could be estimated.
- **We know neither!**
- Suggests an iterative method where the alignments and translation method are refined over time.
- It is the **Expectation-Maximization** Algorithm

EM algorithm

1. initialize model parameters (e.g. uniform)
2. assign probabilities to the missing data
3. estimate model parameters from completed data
4. iterate steps 2-3 until convergence

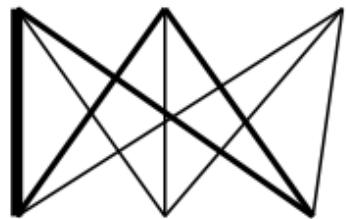
Example



- Initial step: all alignments equally likely
- Model learns that, e.g., *la* is often aligned with *the*

—... —b— ...

... la maison ... la maison blue ... la fleur ...

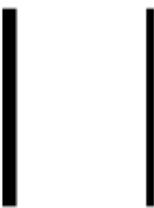
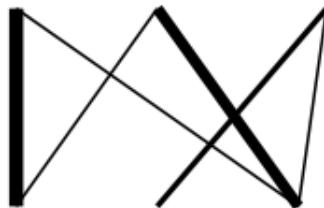


... the house ... the blue house ... the flower ...

- After one iteration
- Alignments, e.g., between *la* and *the* are more likely

— · · · · · · · ·

... la maison ... la maison bleu ... la fleur ...



... the house ... the blue house ... the flower ...

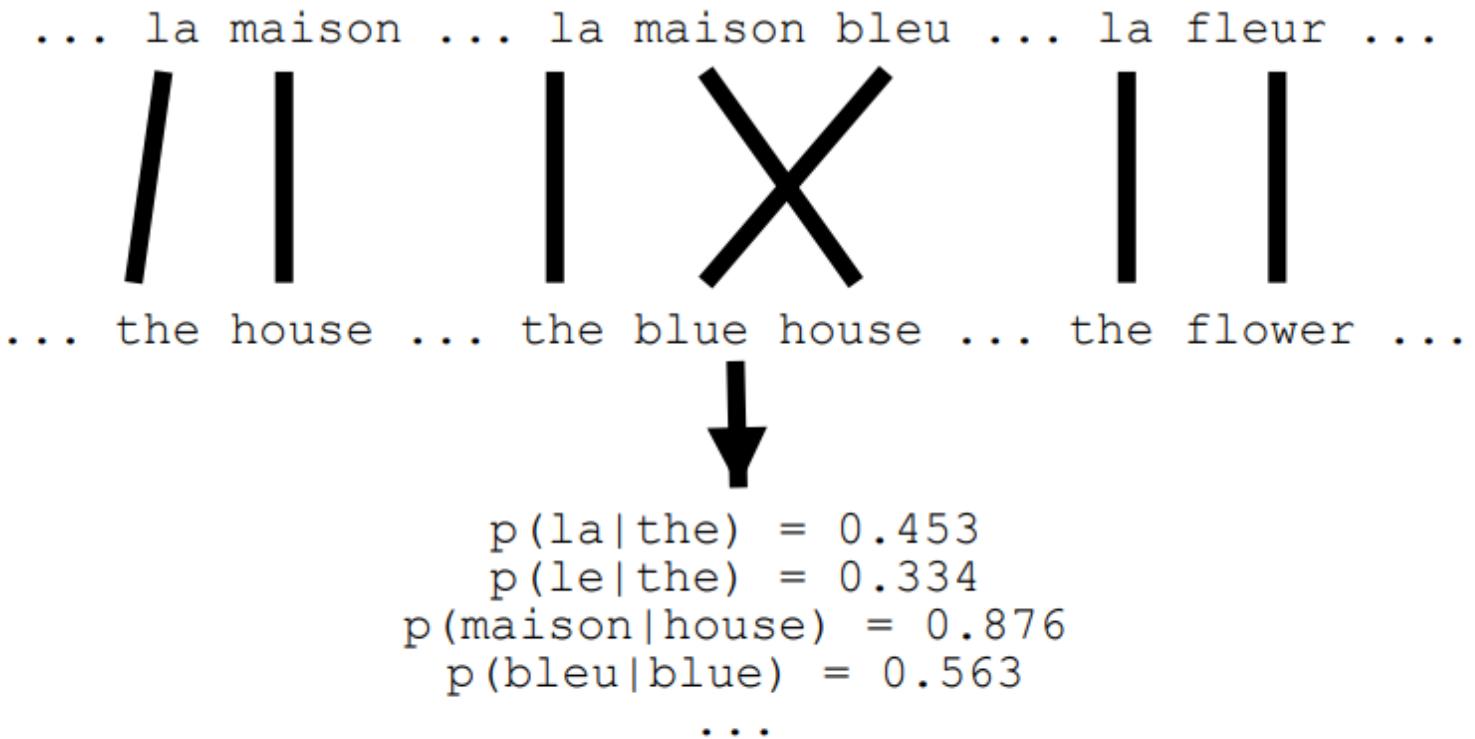
- After another iteration
- It becomes apparent that alignments, e.g., between *fleur* and *flower* are more

... la maison ... la maison bleu ... la fleur ...



... the house ... the blue house ... the flower ...

- Convergence



- Parameter estimation from the aligned corpus

Pseudo code

```
do
    set count(e|f) to 0 for all e,f
    set total(f) to 0 for all f
    for all sentence pairs (e_s,f_s)
        for all words e in e_s
            total_s = 0
            for all words f in f_s
                total_s += t(e|f)
        for all words e in e_s
            for all words f in f_s
                count(e|f) += t(e|f) / total_s
                total(f)   += t(e|f) / total_s
        for all f in domain( total(.) )
            for all e in domain( count(.|f) )
                t(e|f) = count(e|f) / total(f)
    until convergence
```

Example

Consider the following sentences as parallel corpus

Sentence 1:

f1: kutta bhouka

e1: dog barked

Sentence 2:

f2: kutta kata kutta

e2: dog bit dog

Step 1 – Initialization: Set all translation parameters uniformly

Since there are 3 English words all $p(e|f)$ is set to 1/3.

$p(\text{dog} \text{kutta}) = 1/3$	$p(\text{bit} \text{kutta}) = 1/3$	$p(\text{barked} \text{kutta}) = 1/3$
$p(\text{dog} \text{bhouka}) = 1/3$	$p(\text{bit} \text{bhouka}) = 1/3$	$p(\text{barked} \text{bhouka}) = 1/3$
$p(\text{dog} \text{kata}) = 1/3$	$p(\text{bit} \text{kata}) = 1/3$	$p(\text{barked} \text{kata}) = 1/3$

Sentence 2:

f_2 : kutta kata kutta

e_2 : dog bit dog

Step 2 – Alignments: compute $p(a,e|f)$ and $p(a|e,f)$

S1 can be generated using the following 4 different alignments:

$a1=\{<0,0><0,1>\}$ $a2=\{<1,0><1,1>\}$ $a3=\{<0,0><1,1>\}$ $a4=\{<0,1><1,0>\}$

Each $p(a,e|f)$ can be generated with a probability for ex. $a1$ can be computed by $p(0|0) \times p(0|1) = p(\text{dog}|\text{kutta}) \times p(\text{dog}|\text{bhouska}) = 1/3 \times 1/3 = 1/9$

$a1=1/9, a2=1/9, a3=1/9, a4=1/9$

Step3: Compute $p(a|e,f)$ for each alignment normalize the values

S1

$$a_1 = 1/9 \quad p(a_1|e,f) = (1/9)/(1/9+1/9+1/9+1/9) = \cancel{1/4}$$

$$a_2 = 1/9 \quad p(a_2|e,f) = (1/9)/(1/9+1/9+1/9+1/9) = \cancel{1/4}$$

$$a_3 = 1/9 \quad p(a_3|e,f) = (1/9)/(1/9+1/9+1/9+1/9) = \cancel{1/4}$$

$$a_4 = 1/9 \quad p(a_4|e,f) = (1/9)/(1/9+1/9+1/9+1/9) = 1/4$$

S2

$$a_1 = 1/27 \quad p(a_1|e,f) = (1/27)/(1/27+1/27+\dots+1/27 \text{ (13 times)}) = 1/13$$

$$a_2 = 1/27 \quad p(a_2|e,f) = 1/13$$

$$a_8 = 1/27 \quad p(a_8|e,f) = 1/13$$

$$a_3 = 1/27 \quad p(a_3|e,f) = 1/13$$

$$a_9 = 1/27 \quad p(a_9|e,f) = 1/13$$

$$a_4 = 1/27 \quad p(a_4|e,f) = 1/13$$

$$a_{10} = 1/27 \quad p(a_{10}|e,f) = 1/13$$

$$a_5 = 1/27 \quad p(a_5|e,f) = 1/13$$

$$a_{11} = 1/27 \quad p(a_{11}|e,f) = 1/13$$

$$a_6 = 1/27 \quad p(a_6|e,f) = 1/13$$

$$a_{12} = 1/27 \quad p(a_{12}|e,f) = 1/13$$

$$a_7 = 1/27 \quad p(a_7|e,f) = 1/13$$

$$a_{13} = 1/27 \quad p(a_{13}|e,f) = 1/13$$

- Step4: Take the counts for each translations over the whole corpus i.e observe in the whole corpus how many times each words have been aligned and add their $p(a|e,f)$

$c(\text{dog} \text{kutta}) = 35/26$	$c(\text{bit} \text{kutta}) = 6/13$	$c(\text{barked} \text{kutta}) = 1/2$
$c(\text{dog} \text{bhouka}) = 1/2$	$c(\text{bit} \text{bhouka}) = 0$	$c(\text{barked} \text{bhouka}) = 1/2$
$c(\text{dog} \text{kata}) = 6/13$	$c(\text{bit} \text{kata}) = 7/13$	$c(\text{barked} \text{kata}) = 0$

- Step5: Normalize fractional counts to get revised parameter values

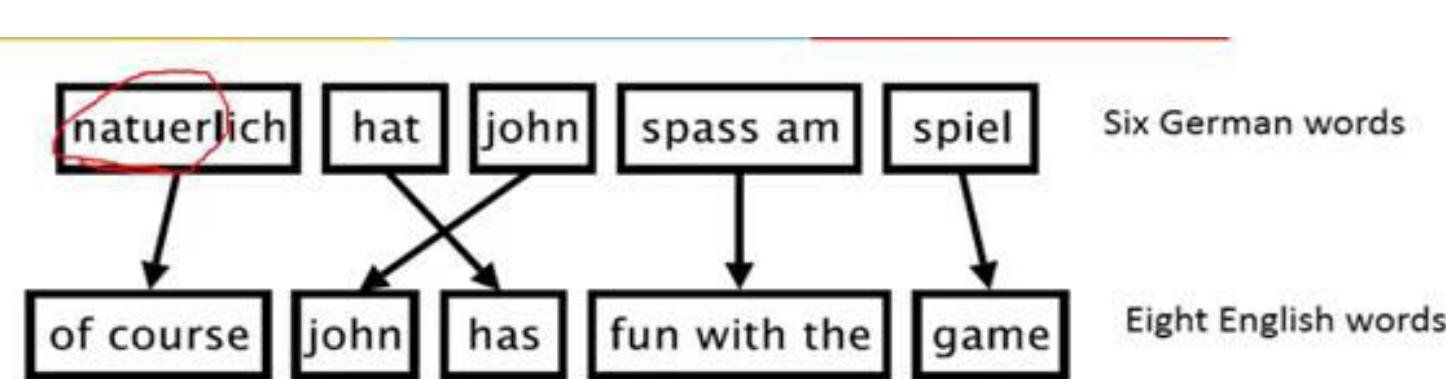
$c(\text{dog} \text{kutta}) = 35/60$	$c(\text{bit} \text{kutta}) = 1/5$	$c(\text{barked} \text{kutta}) = 13/60$
$c(\text{dog} \text{bhouka}) = 1/2$	$c(\text{bit} \text{bhouka}) = 0$	$c(\text{barked} \text{bhouka}) = 1/2$
$c(\text{dog} \text{kata}) = 6/13$	$c(\text{bit} \text{kata}) = 7/13$	$c(\text{barked} \text{kata}) = 0$

- Note: Correct translation probabilities increase for ex dog, kutta from the initial uniform probability

Phrase based model

- Word-Based Models translate words as atomic units
- Phrase-Based Models translate phrases as atomic units
- Advantages:
 - many-to-many translation can handle non-compositional phrases
 - use of local context in translation
 - the more data, the longer phrases can be learned
- "Standard Model", used by Google Translate, Microsoft and others

Phrase based model



- Foreign input is segmented in phrases(Phrase segmentation)
- Each phrase is translated into English(Phrase translation)
- Phrases are reordered so that the verb follows the subject.(Output ordering)

- Main knowledge source: table with phrase translations and their probabilities
- Example: phrase translations for natuerlich

Translation	Probability $\phi(\bar{e} \bar{f})$
of course	0.5
naturally	0.3
of course ,	0.15
, of course ,	0.05

Advantages

- Words may not be the best atomic units for translation, due to frequent one-to-many mappings (and vice versa).
- Translating word groups instead of single words helps to resolve translation ambiguities.
- if we have large training corpora, we can learn longer and longer useful phrases, sometimes even memorize the translation of entire sentences.

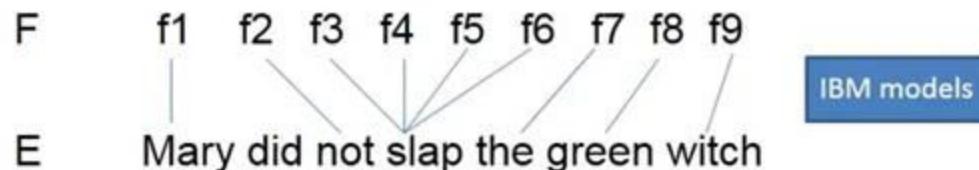
- First stage in training a phrase based model is extraction of a Phrase-based (PB) lexicon
- A PB lexicon pairs strings in one language with strings in other language.e.g.,

English	Spanish
	michael — michael
assumes —	geht davon aus / geht davon aus ,
	that — dass / , dass
	he — er
	will stay — bleibt
	in the — im
	house — haus

One or more words on either side of the source or target language and number of words could differ

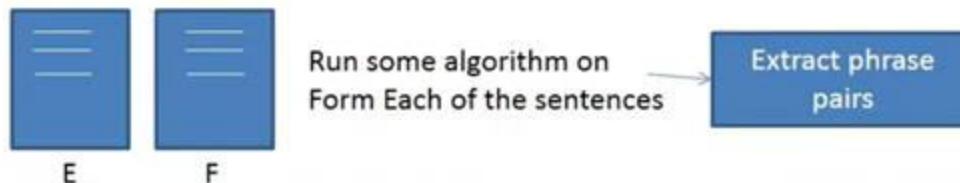
Example

A training example(E/F sentence pair)



Extract phrase pairs form the above alignment

(Mary,f1) (not,f2) (slap,f3f4f5) (the,f7)(green,f8)(witch,f9)



FOREIGN									
	f1	f2	f3	f4	f5	f6	f7	f8	f9
Mary	●								
did									
Not		●							
Slap			●	●	●	●			
the						●			
green							●		
witch							●		

Each foreign word is aligned to exactly one English word observe that there is
One single dot in each column

Finding alignment matrices

- Step 1: Train IBM Model for $P(f|e)$ and come with the most likely alignment for each (e,f) pair.
- Step 2: Train IBM Model for $P(e|f)$ and come with the most likely alignment for each (e,f) pair.
- We now have two alignments: take union of the two alignments as a starting point.

Alignment from $P(e|f)$ model

		FOREIGN								
		f1	f2	f3	f4	f5	f6	f7	f8	f9
E	Mary	●								
	did				●					
N	Not	●								
	Slap		●	●	●	●				
G	the				●					
	green						●	●		
L	witch						●	●		

Constrained by each f aligned to one e

union of alignment

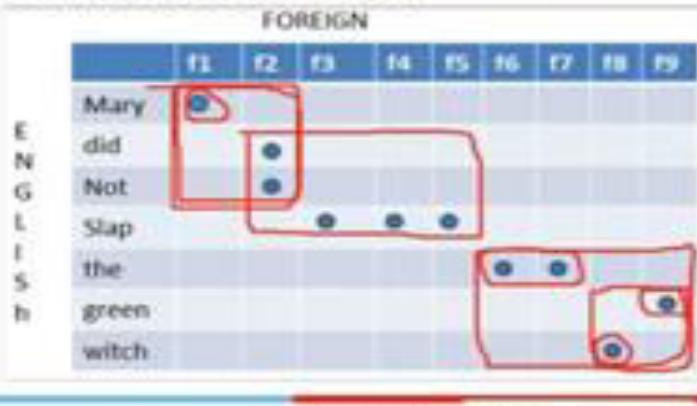
Alignment from $P(f|e)$ model

		FOREIGN								
		f1	f2	f3	f4	f5	f6	f7	f8	f9
E	Mary	●								
	did		●							
N	Not	●								
	Slap			●			●			
G	the							●		
	green								●	
L	witch								●	●

Constrained by each e aligned to one f
For ex each row has one alignment

Extracting phrase pairs from the alignment matrix

- A phrase pair consist of a sequence of English words e, paired with a sequence of foreign words, f.
- A phrase pair (e,f) is consistent if
 - 1) There is at least one word in e aligned to a word in f
 - 2)There are no words in f aligned to words outside
 - 3) There are no words in e aligned to words outside f
- (Mary did not,f1,f2) is consistent
- (Mary did,f2) is not consistent
- We extract all consistent phrase pairs for the training example.



Probabilities for Phrase Pairs

- For any phrase pair (f,e) extracted from the training data, we can calculate
- $t(f|e) = \text{count}(f,e) / \text{count } e$
- e.g.,
- $t(f_3,f_4,f_5|\text{slap}) = \text{count}(f_3 \ f_4 \ f_5)/\text{count}(\text{slap})$

Challenges in CLIR

- 1. Translation ambiguity:**
- 2. Phrase identification and translation**
- 3. Translate/transliterate a term:**

Challenges in CLIR

- 4. Transliteration errors:**
- 5. Dictionary coverage:**
- 6. Font:**
- 7. Morphological analysis (different for different languages)**
- 8. Out-of-Vocabulary (OOV) problems**

Factors affecting the performance of CLIR systems

- Limited size of Dictionary
 - New words get added to the language quite frequently and maintaining the dictionary up to date with these new words is difficult. Also compounds and phrases can be formed from existing words in the language. No dictionary can contain all possible compounds and phrases. A specific domain can generate a specific terminology which might not be present in general dictionary. Inflected word forms are not included in dictionary
- Query translation/transliteration performance
 - A search key may have more than one sense in source language which may be expressed by dictionary by providing several alternatives. During the translation process, extraneous senses may be added to the query due to the fact that the translation alternatives may also have more than one sense. Lexical ambiguity appears in both source and target language.



BITS Pilani
Pilani Campus

Information Retrieval

Dr. Vijayalakshmi Anand
BITS pilani

Information Retrieval

- *Deals* with the representation, storage and retrieval of unstructured data
- Classical *IR* deals mainly with text
- The evolution of multimedia databases and of the web have given new interest to *IR*.

Multimedia *IR*

- A multimedia information system that can store and retrieve
 - Attributes
 - Text
 - 2D grey-scale and color images
 - 1D time series
 - Digitized voice or music
 - Video

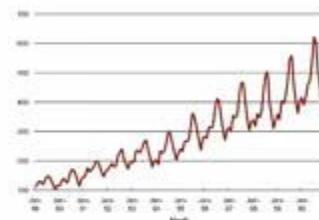
Objects in multimedia database



Objects in Multimedia database:



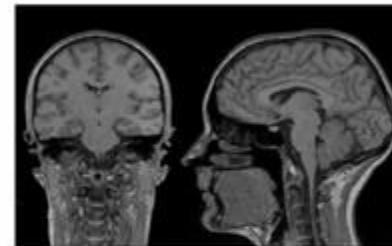
2D-color Images



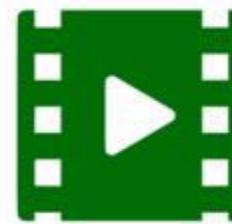
1-D time series



Audio file



Medical Images



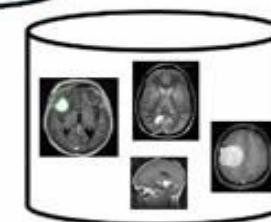
Video file

Examples of Some queries:

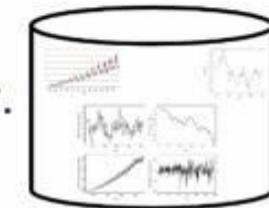
Find images that look like a sunset



Find medical X-rays that contains something
that has the texture of a tumor



Find companies whose stock prices move similarly.



What is multimedia

One or more media
Possibly interlinked
Digital
For communication
(not only entertainment)



Challenges of MMIR

- Semantic gap
- Polysemy
- Fusion problem
- Responsiveness

Semantic gap&Polysemy



- Pixels with spatial distribution
- Faces and vase like object



Basic multimedia search technology

- Metadata are pieces of information about a multimedia object that are not strictly necessary for working with it, but that are useful to
 - describe resources so they can be indexed, classified, located, browsed and found
 - store technical information, such as data formats and compression schemes
 - manage resources such as their rights or where they are currently located
 - record preservation actions
 - create usage trails, eg, which section of a video has been watched how many times

Why metadata is difficult

Simple metadata is ambiguous (e.g. DC.creator)



DC.title = "Reconstruction of Colossus Computer"

DC.creator = "Suzanne Little"

OR

DC.creator = "Tommy Flowers"

OR

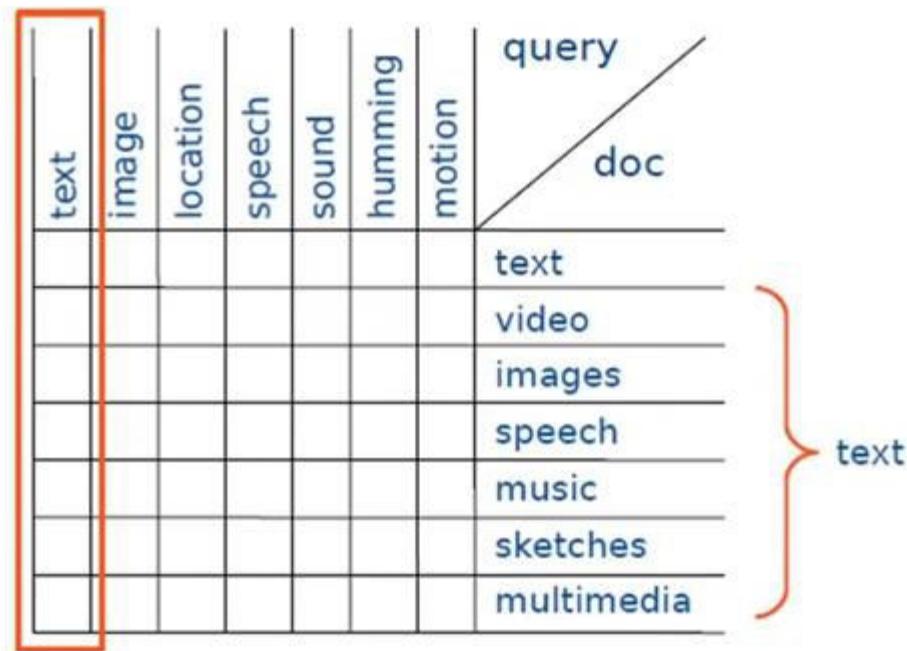
DC.creator = "Tony Sale"

Comprehensive metadata is complex

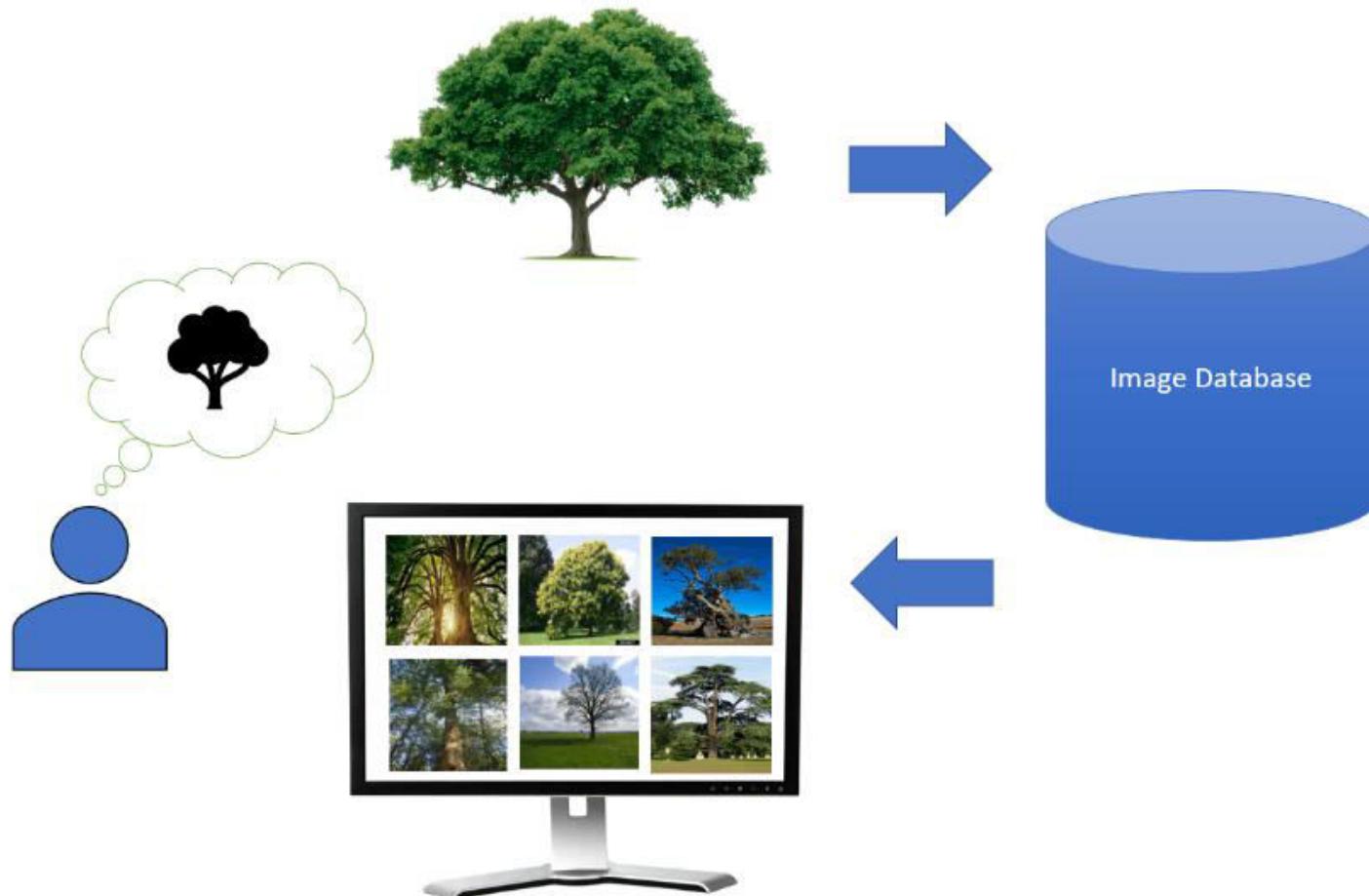
User created metadata is expensive and potentially subjective

How to create?

Piggy bag retrieval



Content base image retrieval



What is it?

- A way of querying image databases
- Uses visual properties of an image as ‘search terms’
- Returns images from a database that share the same or similar visual properties

Limitations

- **labor intensive**
- **imprecise**



Solution CBIR

How does it work?

- 'sees' the image

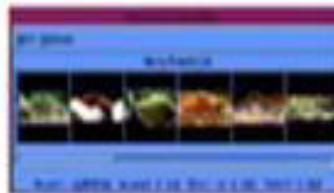
- color

Shape:



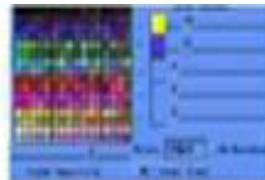
- texture

Shape:



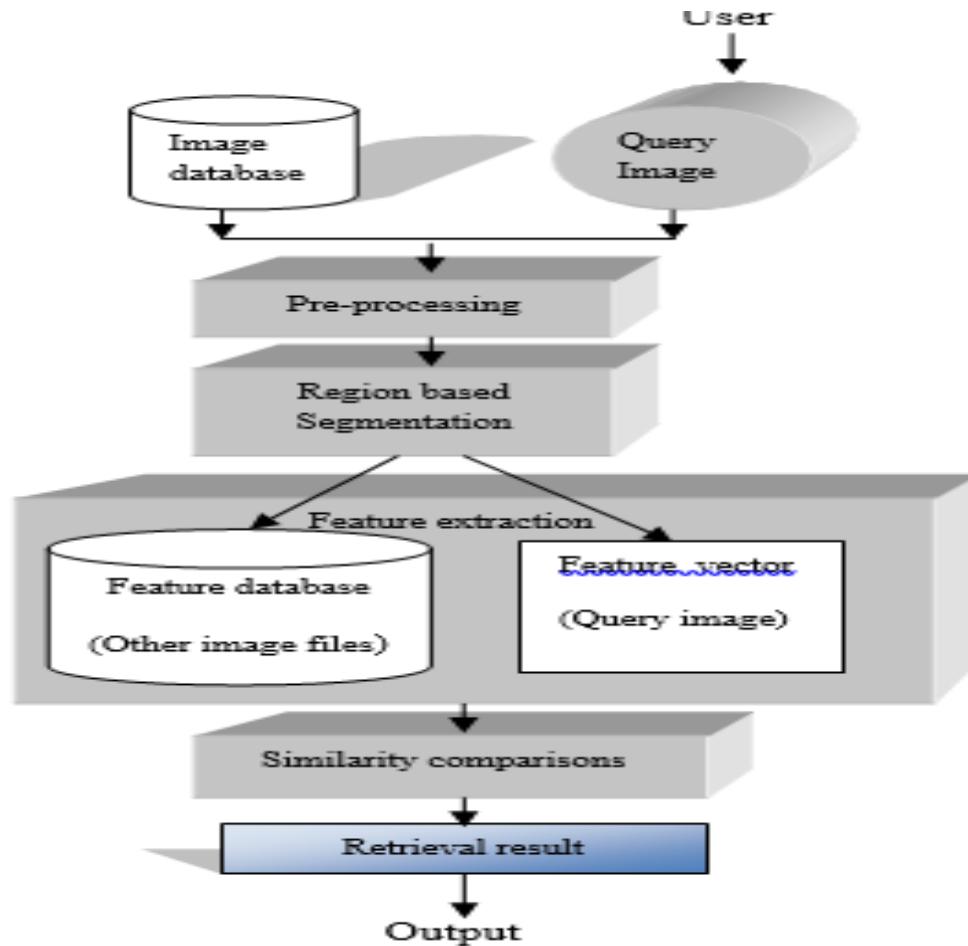
- shape

Shape:



- Compares features of image with images in database
- Computes 'distance' between features of images
- Returns set of images least 'distant' from original

How it works

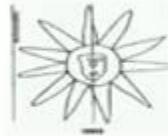


Applications

ImageMiner



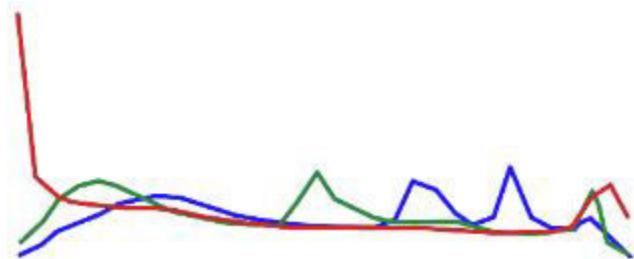
International
Association of
Paper
Historians



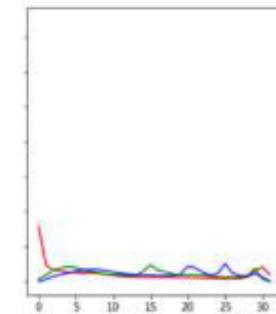
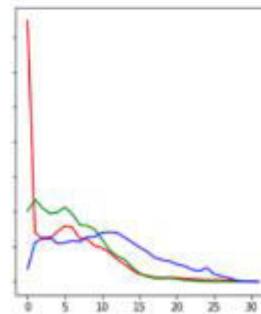
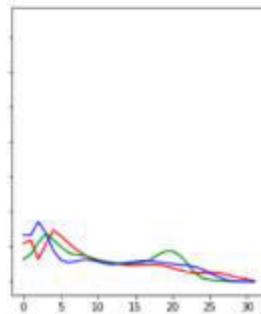
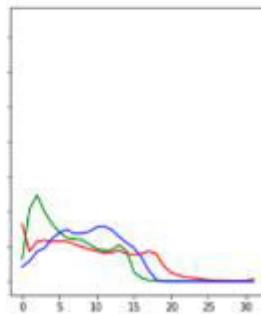
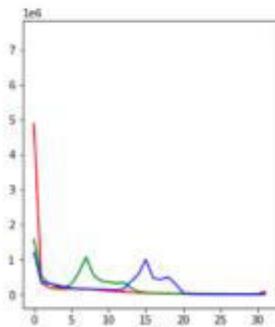
Photobook

Colour histograms

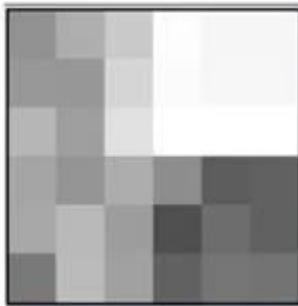
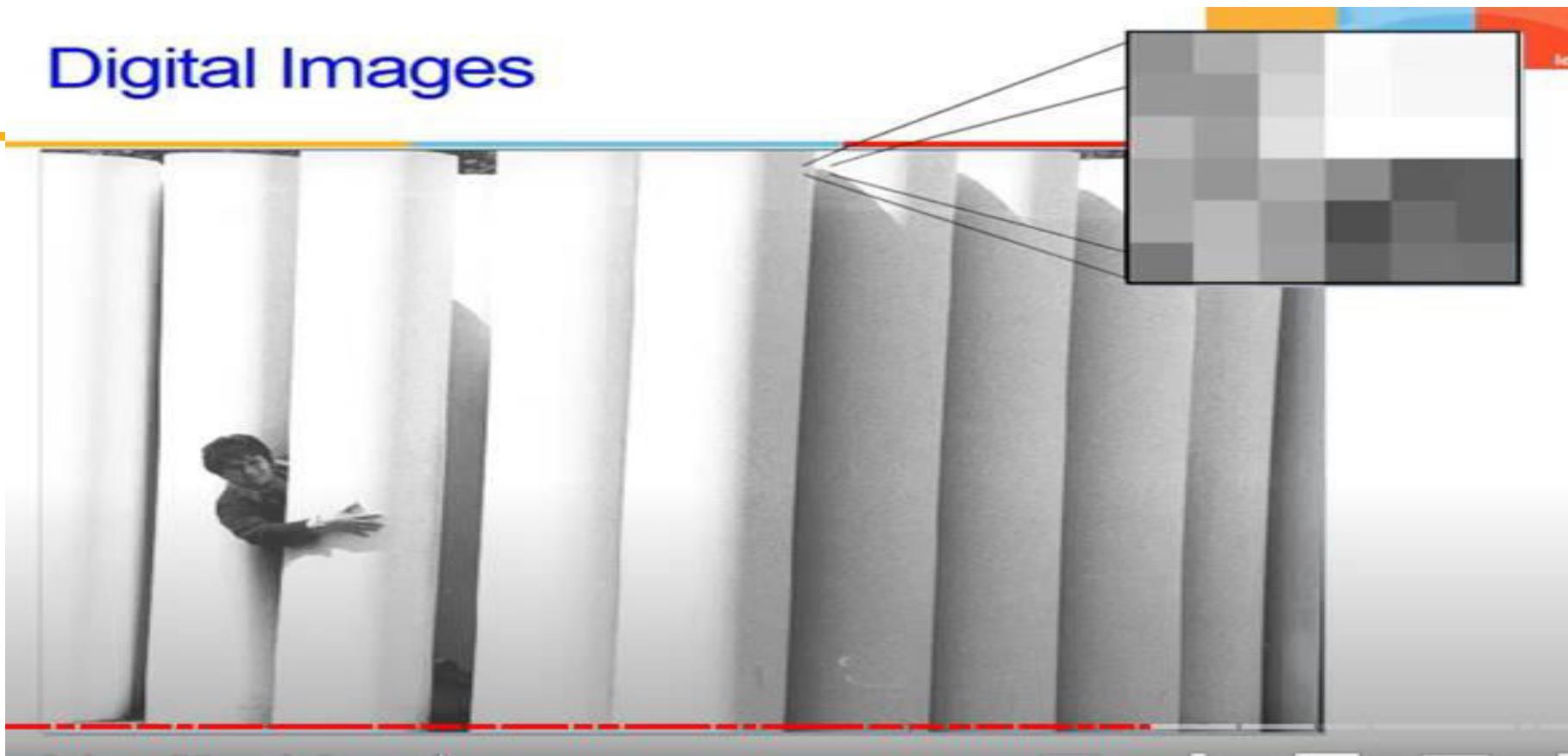
Query image:



Results:

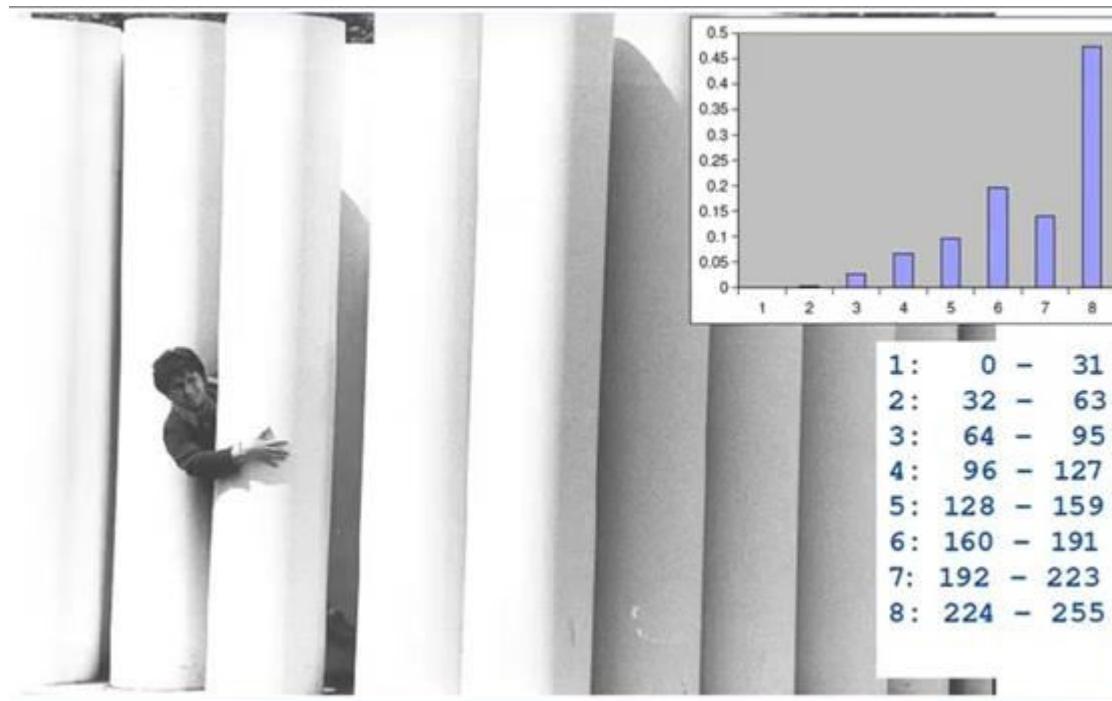


Digital Images

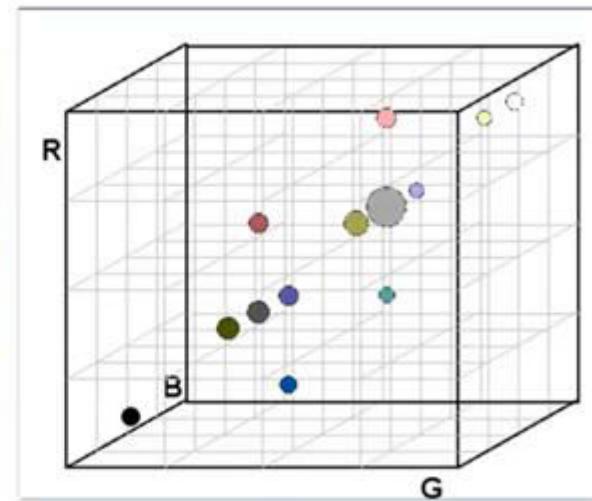


145	173	201	253	245	245
153	151	213	251	247	247
181	159	225	255	255	255
165	149	173	141	93	97
167	185	157	79	109	97
121	187	161	97	117	115

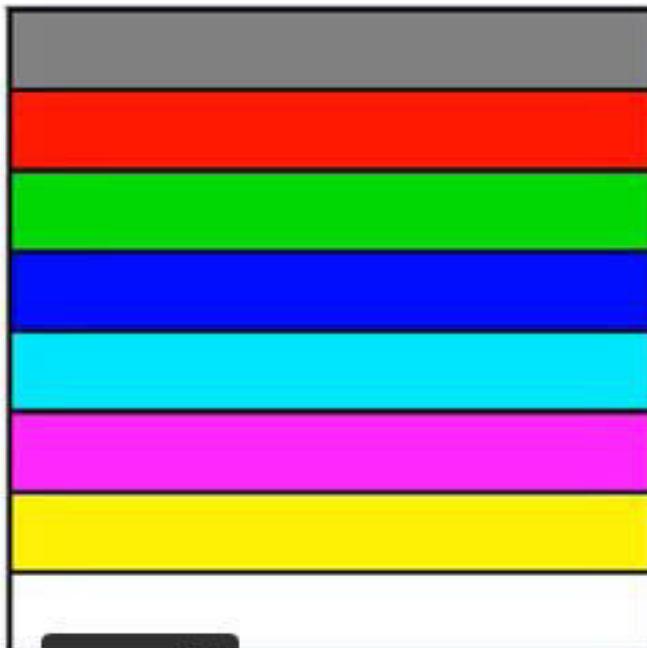
Histogram



Colour histogram

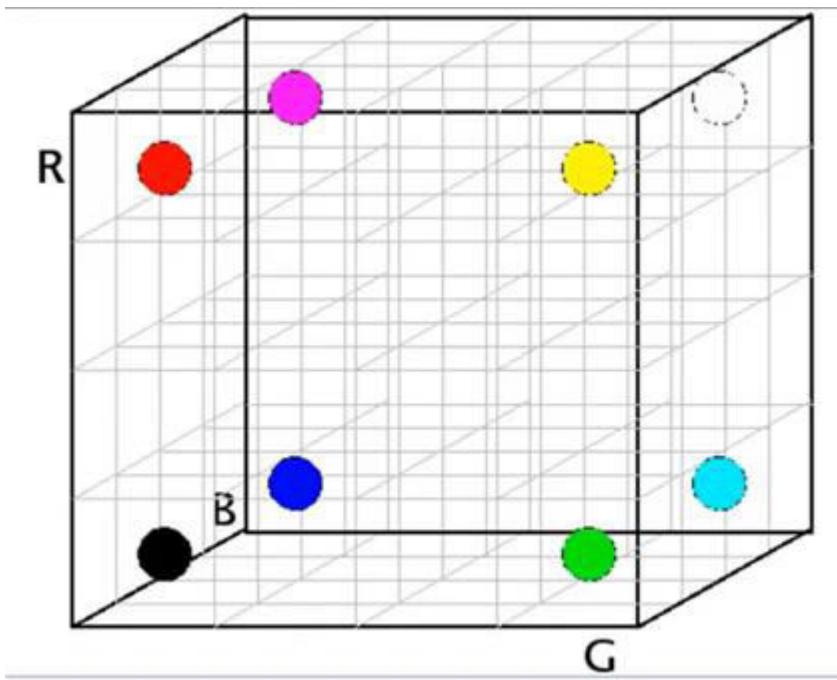


Excercise



R	G	B	
0	0	0	black
255	0	0	red
0	255	0	green
0	0	255	blue
0	255	255	cyan
255	0	255	magenta
255	255	0	yellow
255	255	255	white

solution



Colour histogram extraction

```
Off-line, for each image  
    create histogram with a bin for each color  
    initialize each bin counter = 0  
        for each pixel in image:  
            increment bin counter corresponding to pixel  
            color  
        end
```

On Line, use histograms in image similarity measure

Example: Low-level Color Features

- **Assumption:** If two images share similar colors then also their content may be similar
- Loss of information through low-level features
- Example: red sunset (orange, yellow)



Simple histogram distance measure

- The distance between the histogram of the query image and images in the database are measured
- Images with a histogram distance smaller than a predefined threshold are retrieved from the database
- The simplest distance between images I and H is the L-1 metric distance as $D(I,H) = \sum |I-H|$

Comparisons of colour

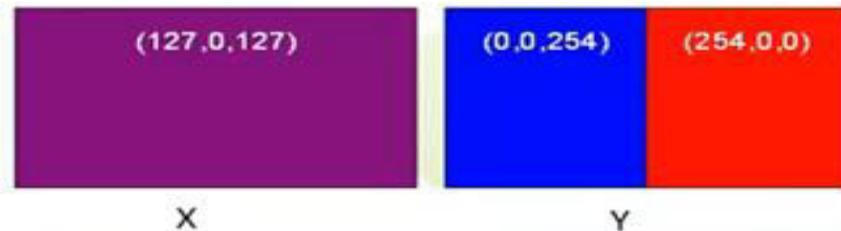
- Compare images based on the color? Extract **color features** first –
 - Each pixel of an image contains color information
- Images consist of many pixels
 - Pixel by pixel?
- Aggregation for comparisons?
 - Average color
 - Color histograms
 - Color layout (regions)

Average colour

- Comparison of 2 images x and y by using the Euclidean distance for the average color

$$d_{avg}^2(x, y) = (R_{avgx} - R_{avgy})^2 + (G_{avgx} - G_{avgy})^2 + (B_{avgx} - B_{avgy})^2$$

- Very bad similarity measure
- E.g., magenta image and red blue image are the same according to average color



Average colour

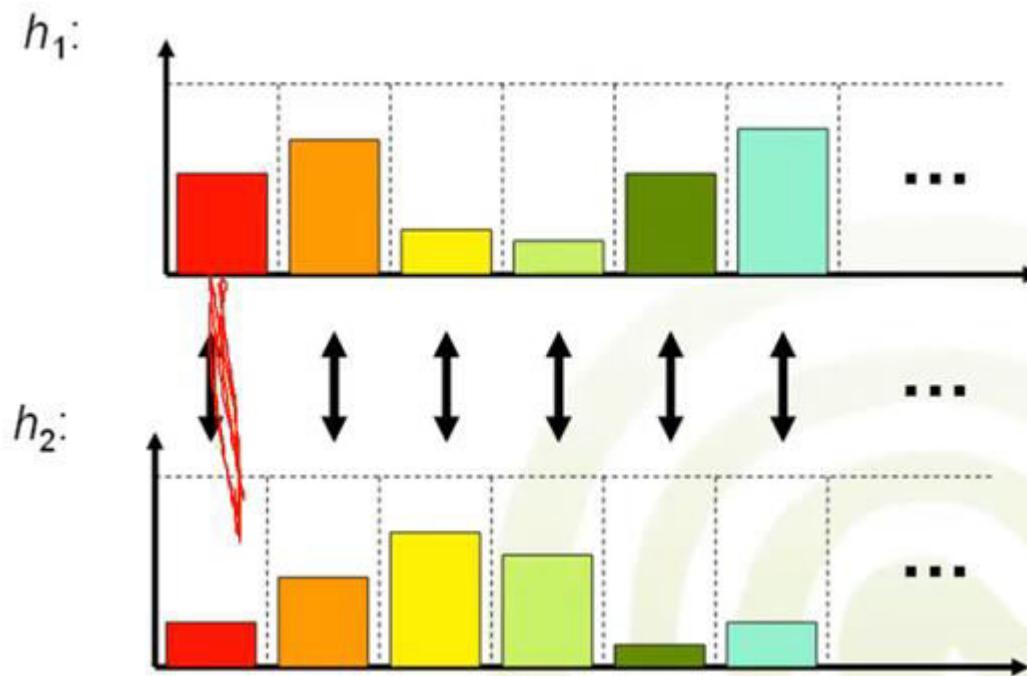
- Perceptionally somewhat **questionable** ...
- But...
 - Quick and easy to calculate and compare
- Best to use as a filter: **exclude images**
 - Dominant color influences the average color, the opposite is not valid
 - E.g., search for mostly blue images: exclude all images with red, yellow or green color averages

- Given: histograms h_1 and h_2
- **Minkowski distance** with parameter r :

$$d_r(h_1, h_2) := \sum_{i \in C} |h_1(i) - h_2(i)|^r$$

- $r=1$: Histogram-L₁-norm
(also: city block distance, Manhattan distance)
- $r=2$: Histogram-L₂-norm (Euclidean)

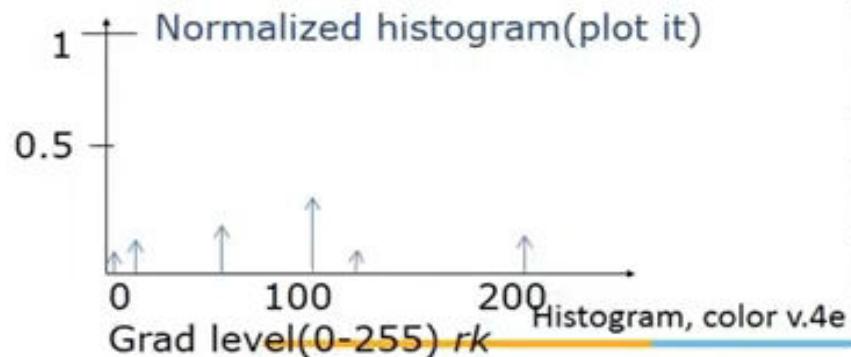
Example



Normalized histogram

- An image of 5x4 pixels K=0,1,2,...,L-1=255 M rows
- L=256 levels
- Sketch the histogram

Normalized
Count
 $P(r_k) = n_k / MN$



7	100	200	45
2	100	7	120
100	100	7	120
200	100	45	100
200	200	45	45

level	count	N columns
2	1	
7	3	
45	4	
100	6	
120	2	
200	4	

Types of queries

- Whole match

Given a collection of N objects O_1, O_2, \dots, O_n and a query object Q , we want to find those data objects that are within distance ϵ from Q .

e.g. objects are 512×512 gray scale images and query object is also 512×512 .

- Sub-pattern match

- Find parts of image that are...

In 512×512 brain scans, find pieces similar to the given 16×16 typical X-ray of a tumor.

- passage retrieval for text documents.



Genetic multimedia indexing problem

- Given a database of multimedia objects
- Design fast search algorithms that locate objects that match a query object, exactly or approximately
 - Objects:
 - 1-d time sequences
 - Digitized voice or music
 - 2-d colour images
 - 2-d or 3-d grey scale medical images
 - Video clips
- E.g.: “Find companies whose stock prices move similarly”

Sequential searching



- For each and every object O_i ($1 \leq i \leq N$)

Compute distance between Q and the object O_i i.e. $D(Q, O_i)$ and Return object if $D(Q, O_i) \leq \varepsilon$.

It is slow for 2 reasons

- Distance computation might be expensive.
- Database size N might be huge.

Generic Multimedia Object INdexIng (GEMINI) approach

Steps of Generic Multimedia Object Indexing Approach

Step-1: Determine the distance function $D()$ between two objects.

Step-2: Find one or more numerical feature-extraction functions, to provide a 'quick-and-dirty' test.

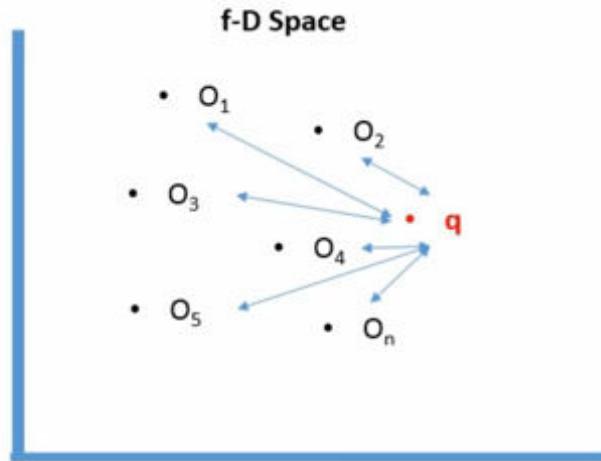
Step-3: Prove that the distance in feature space lower-bounds the actual distance $D()$, to guarantee correctness.

Step-4: Use a Spatial Access Method (e.g. R-Tree) to store and retrieve the f-D feature vectors.

Step-1: Determine the distance function D() between two objects.

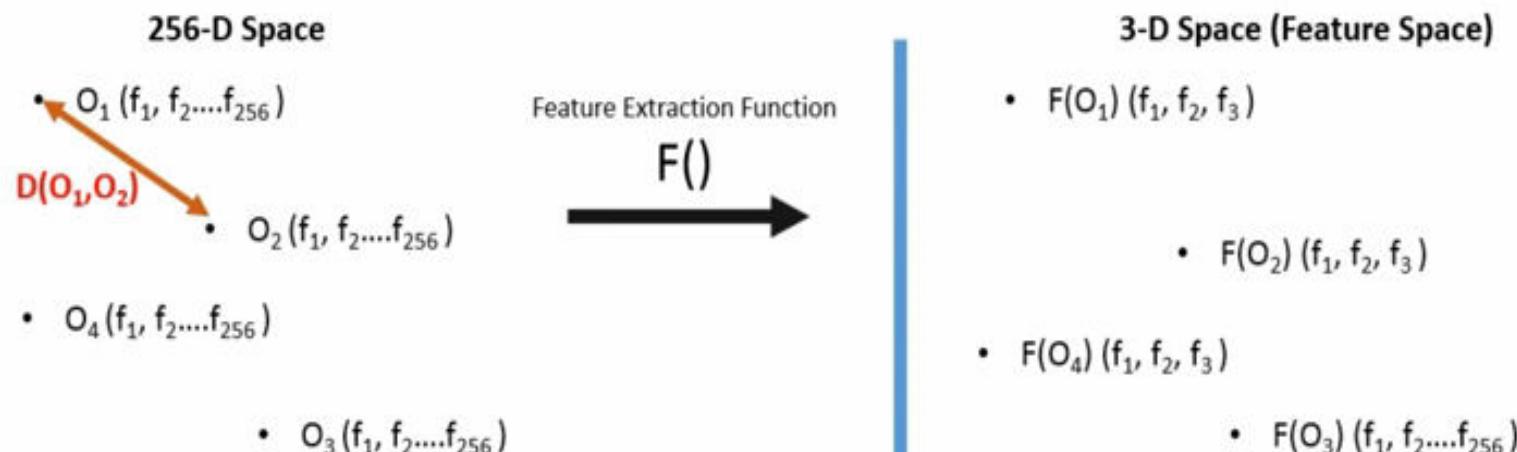
Objects

$O_1 (f_1, f_2, f_3 \dots f_f)$
 $O_2 (f_1, f_2, f_3 \dots f_f)$
 $O_3 (f_1, f_2, f_3 \dots f_f)$
 $O_4 (f_1, f_2, f_3 \dots f_f) \dots$
...
...
 $\dots O_n (f_1, f_2, f_3 \dots f_f)$



Distance = Dissimilarity

Step-2: Find one or more numerical feature-extraction functions, to provide a 'quick-and-dirty' test.



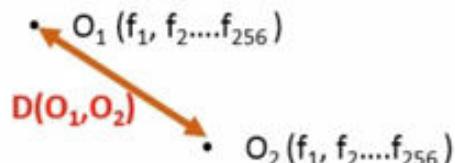
More Time to calculate Distance



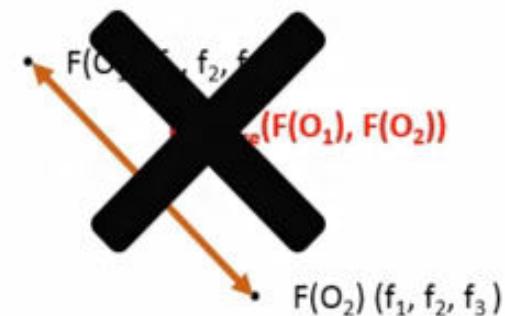
Step-3: Prove that the distance in feature space lower-bounds the actual distance $D()$, to guarantee correctness.

$$D_{\text{feature}}(F(O_1), F(O_2)) \leq D(O_1, O_2)$$

Actual Space



Feature Space (for quick calculation)



Step-4: Use a Spatial Access Method to store and retrieve the f-D feature vectors.

e.g. R-Tree

Audio information retrieval

The audio retrieval problem

The retrieval of audio tracks that match a vaguely specified audio-information need.

This problem takes many forms such as:

- **fingerprinting**: given a small snippet of sound, find an audio object that matches it
- **speech recognition**: given an audio track, recognize the text it contains
- **speaker identification**: given an audio track, recognize the speaker(s) it contains spoken
- **document retrieval**: given a text query, retrieve spoken documents that match the query

Finger printing

- Audio fingerprinting is a commercially successful IR task
- Use a small snippet of sound to query a large database and look for an exact match
- Process is complicated because the query is often corrupted
- Typical case: snippet of sound captured by a cell phone in the noisy environment of a pub



Music retrieval:

- Use hash function
- unique metadata

Speech recognition

- 1.analyze the audio;
- 2.break it into parts;
- 3.digitize it into a computer-readable format; and
- 4.use an algorithm to match it to the most suitable text representation.

Speaker identification

- Consists of determining who is speaking regardless of the words they are saying
- Two common approaches
 - Speaker-dependent speech recognition
 - GMMs density estimation
- Speaker-dependent speech recognition: unique models tuned to the pronunciation peculiarities of each speaker collecting speaker-dependent information for a large population is impractical

Speaker identification

- More general model like a single GMM used to capture all sounds produced by a speaker
- GMM might require up to 2,000 components to properly model the way each speaker speaks
large number of components is necessary because system is not trying to recognize individual words
- Speaker identification using GMMs often needs more than 10 seconds of speech to make a reliable decision

Spoken Document Retrieval

- Retrieve spoken documents that fit a text query
- Two speech-specific approaches are most commonly used
 - keyword spotting
 - phonetic recognition
- Both approaches are more robust for IR than normal speech-to-text using a speech recognizer

Keyword spotting

- Recognize pre-selected keywords inside spoken documents
 - Each keyword contains a lot of information
 - Presence of one of them is highly informative
 - Approach is limiting because users must include those keywords in their queries

Phonetic Recognition

- Perform retrieval at the phoneme level
- Key issue: needs to deal with mismatches at the level of underlying sounds
 - Using conventional IR techniques the words "bat" and "bet" are completely different
 - But phonetically the /a/ and the /i/ in these two words are very easy to confuse

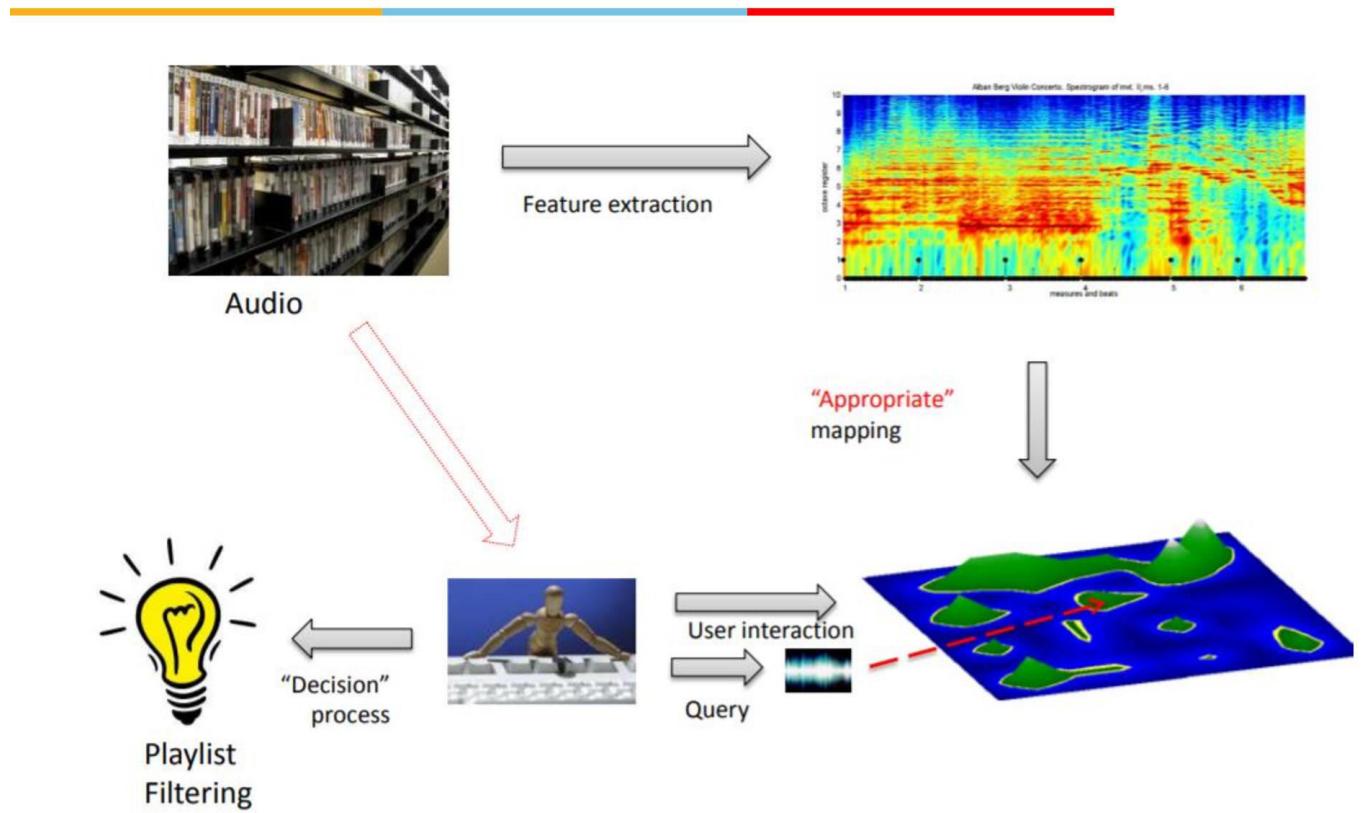
Audio analysis

- Analyzing the audio signal, to extract basic information, is an important part of an audio-retrieval system
- Audio is recorded as a waveform
 - Measures of the changes in air pressure along the wave over time
 - If sound wave is produced by combination of multiple sources, signal is complex
 - Each object in a sound landscape has three primary dimensions loudness, pitch, and timbre
 - For IR, we can ignore the overall loudness of the signal
 - Pitch and timbre carry different kinds of information

-
- **Pitch**: attribute of sound that describes musical melody
 - **Timbre**: property of the sound that allows identifying the type of musical instrument that is playing



Audio IR





Audio Information

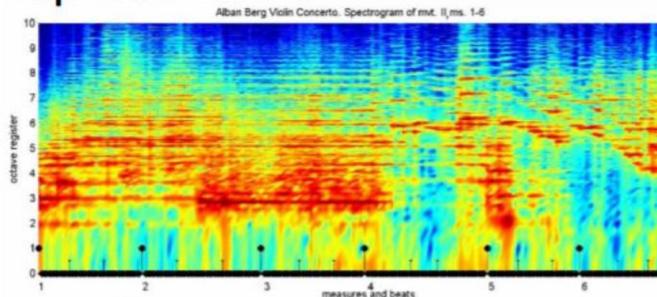
- Speech – Male, female speech
 - Signal-based speech characterization
 - Automatic speech recognition (ASR)
 - ASR (text) retrieval
- Music
 - Genre recognition (Jazz, classic,...)
 - Excerpt retrieval Query by humming
 - ..
- Sound
 - Car, water, people, animals,...
 - Often useful in relation to visual (--> video)



Audio Features

- Low-level-audio representation

- Spectrogram
 - MFCC



- Mid-level representations

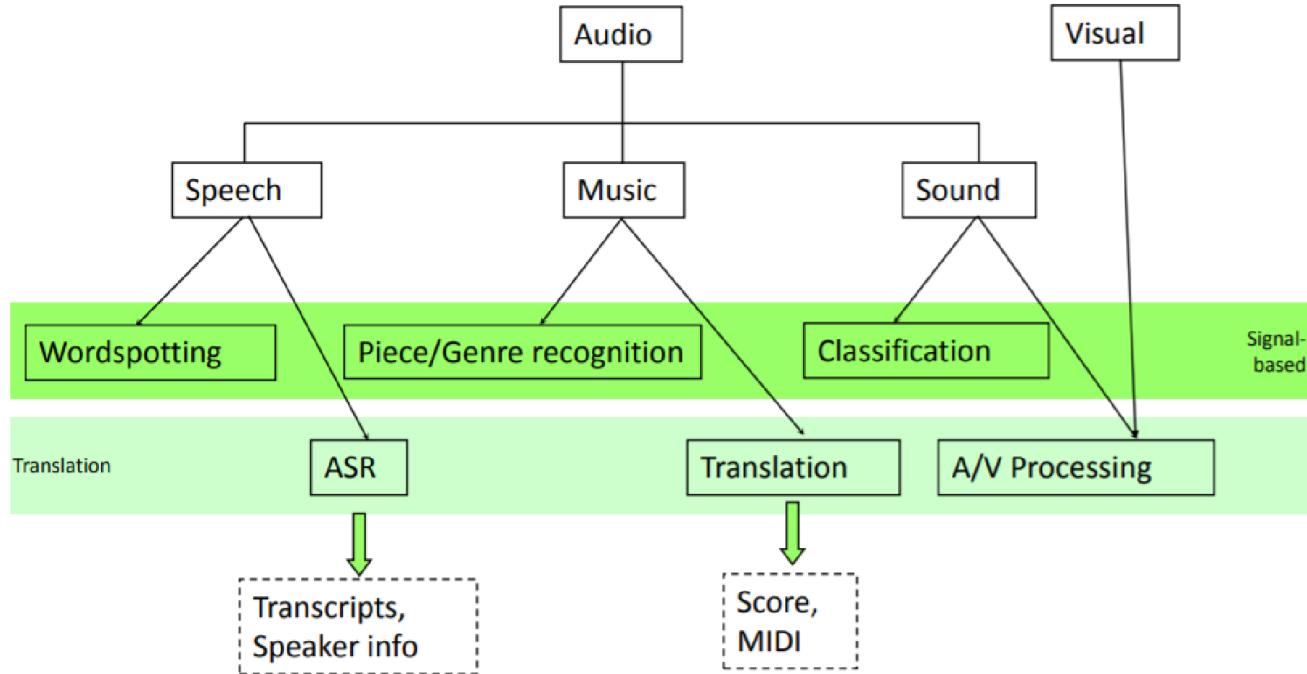
- MIDI strings
 - LPC

- High-level representation

- Automatic speech recognition (ASR)



Audio Information Indexing





Themefinder

Themefinder

[[About](#) | [Search options](#) | [Help](#)]
[[New Links](#) | [Composers](#) | [Random](#)]

[Take the Quartet Quiz](#)

Repertory type of music to search

Pitch **A-G, sharp=♯, flat=♭**
e.g. C E- G F♯

Interval
maj=H, min=m, aug=A, dim=d per=p, fifth=5,
up=+, down=-,
e.g. +m9 -P8 +M3 P1

Scale
do=1, re=2, mi=3, fa=4, so=5, la=6, ti=7 (mode
insensitive). e.g. 34554321

Degree
up=l, down=v, unison=-,
e.g. l/v-/ or vuuds

Gross Contour
up step=u, up leap=u,
down step=d, down leap=d, same=s. e.g.
uUDsdu

Refined Contour

Location beginning of theme only, or
 anywhere in theme

Key **Mode:**

Meter /

Submit Search



Video IR

Video type

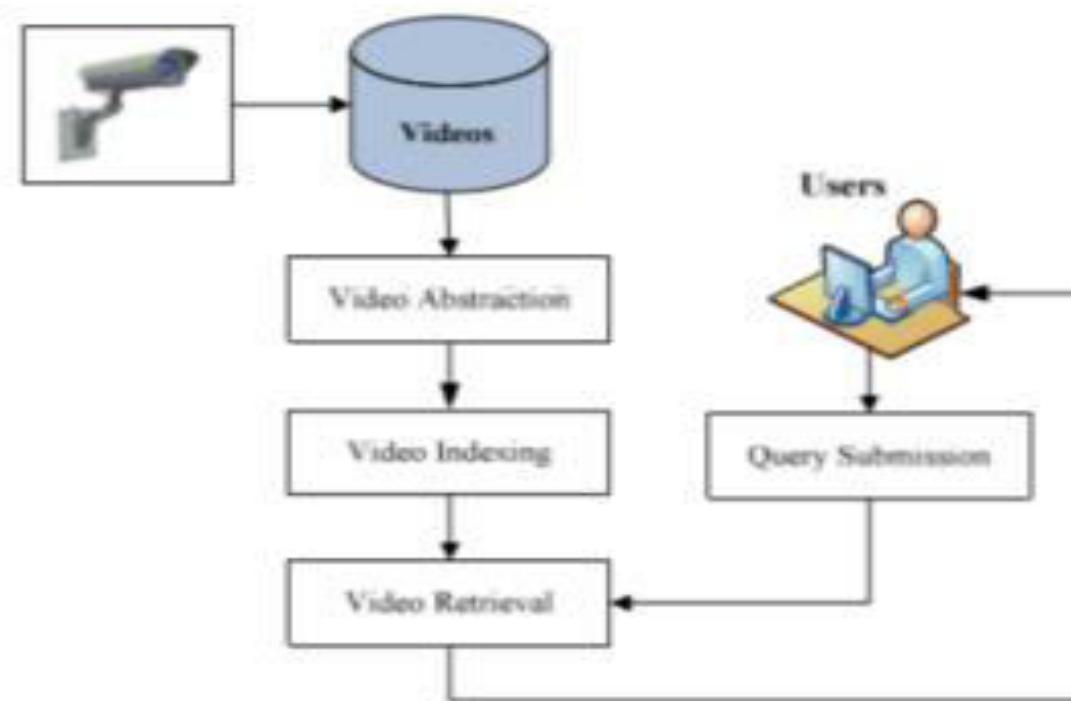
- Movies
- Amateur souvenir film
- Lectures
- YouTube (short, static)

Combining individual media (fusion)

- Each brings some information...

Going further: social multimedia

Abstraction



Challenges in Video information retrieval

■ Semantic gap

- Fusion is a way to reduce the semantic gap
- Examples
 - – Looking at a movie w/o audio
 - – Listening to a story w/o visual
 - – Voice conversation vs video conversation
 - – Disambiguation (eg „jaguar“)



Thank You!



Recommender Systems



BITS Pilani
Pilani Campus

Vijayalakshmi Anand
Prof. CSIS Department
BITS -Pilani

Outline

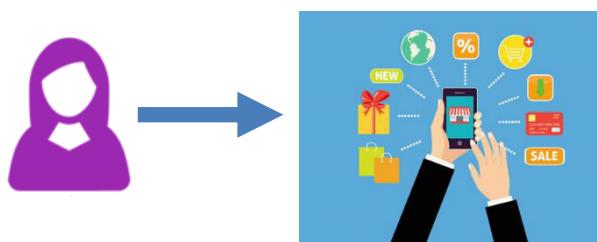
- Motivation for Recommender Systems
- What is recommender systems
- Example
- Types of recommender systems
- SVD
- Latent factor model

Motivation for Recommender Systems

1. Vast amount of digital information available
What music to listen what movie to see what book to read etc.
2. Long tail effect



- Search Vs Recommender systems





What is recommender system

- A recommender system is a type of information filtering system.
- A recommendation system, or recommender system tries to make predictions on user preferences and make recommendations which should interest customers.



Examples

Netflix-Movie recommendation



Amazon recommendation system

Recommendations for you in Grocery & Gourmet Foods



Recommended items other customers often buy again



Google news

Recommended

Manhunt underway for suspects in fatal Las Vegas strip shooting
Reuters - 7 hours ago

By Dan Whitcomb. Fri Feb 22, 2013 11:44pm EST. (Reuters) - A multi-state manhunt was under way on Friday for the men who shot dead an aspiring rapper as he drove on the Las Vegas Strip in a Maserati, touching off a fiery crash that killed a cab driver and ...

Tea party's anti-Rove 'Nazi ad'
CNN - Feb 20, 2013

Editor's note: John Avlon is a CNN contributor and senior political columnist for Newsweek and The Daily Beast. He is co-editor of the book "Deadline Artists: America's Greatest Newspaper Columns."

Tracking 66 State of the Union proposals
USA TODAY - Feb 20, 2013

President Obama delivered the first State of the Union Address of his second term Feb. 12, laying out domestic and foreign policy goals for the coming year and beyond.

Ronda Rousey's maverick ways lead to landmark UFC bout
Los Angeles Times | Written by Lance Pugmire
2 hours ago

The former Olympic judo competitor has taken independent stances en route to her fight Saturday with Liz Carmouche at Honda Center.

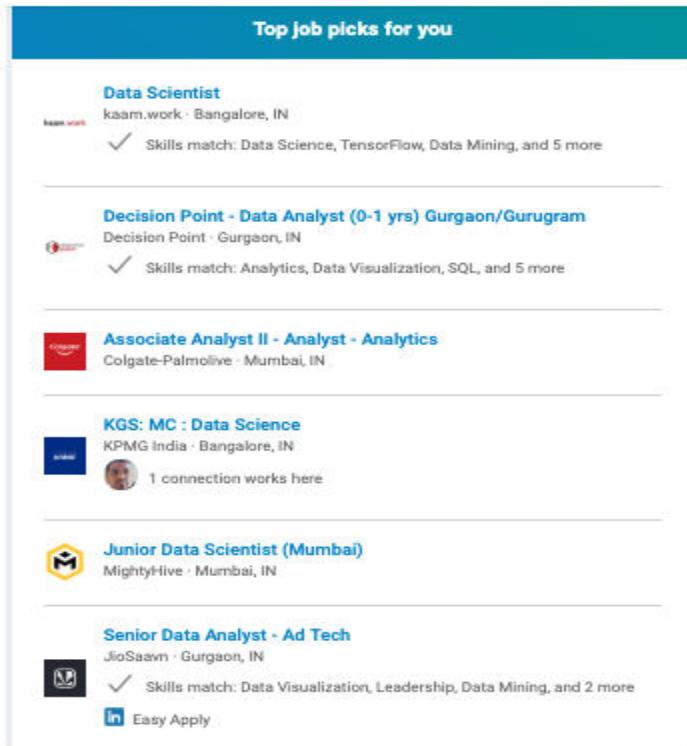
Cornell bioengineers print human ears
Cornell University | Written by Lance Pugmire
2 hours ago

This handout photo provided by Cornell University, taken Feb. 13, 2013, shows Cornell University biomedical engineer Lawrence Bonassar holding the scaffolding for an ear his laboratory is creating using a 3-D printer and cartilage-producing cells.

DOJ move a bad sign for Armstrong
ESPN - 14 hours ago

Lance Armstrong's decision to tell the truth about his doping exposed him to two major problems -- the risk of a \$60 million debt and the renewed possibility of a criminal charge and time in the penitentiary.

Linkedin



Top job picks for you

Data Scientist
kaam.work · Bangalore, IN
✓ Skills match: Data Science, TensorFlow, Data Mining, and 5 more

Decision Point - Data Analyst (0-1 yrs) Gurgaon/Gurugram
Decision Point · Gurgaon, IN
✓ Skills match: Analytics, Data Visualization, SQL, and 5 more

Associate Analyst II - Analyst - Analytics
Colgate-Palmolive · Mumbai, IN

KGS: MC : Data Science
KPMG India · Bangalore, IN
1 connection works here

Junior Data Scientist (Mumbai)
MightyHive · Mumbai, IN

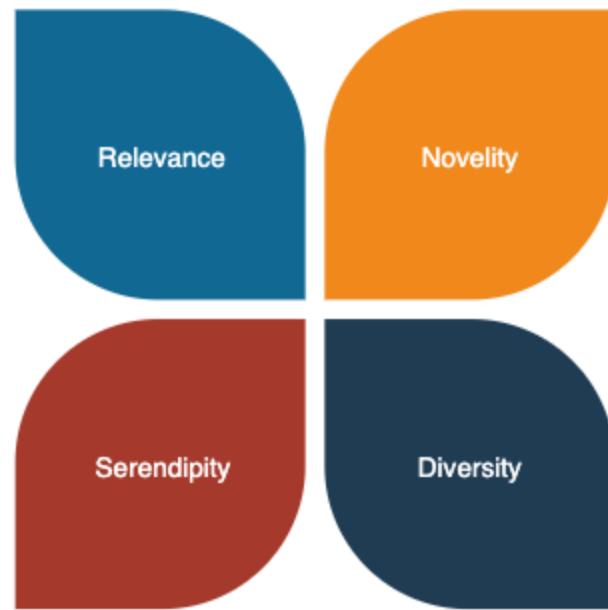
Senior Data Analyst - Ad Tech
JioSaavn · Gurgaon, IN
✓ Skills match: Data Visualization, Leadership, Data Mining, and 2 more
Easy Apply



Modelling Recommender Systems

- $U = \{\text{USERS}\}$
- $I = \{\text{ITEMS}\}$
- F is a utility function, measures the usefulness of items I to user U .
 $F: U \times I \rightarrow R$ where R is the rating

Characteristics of recommendation system



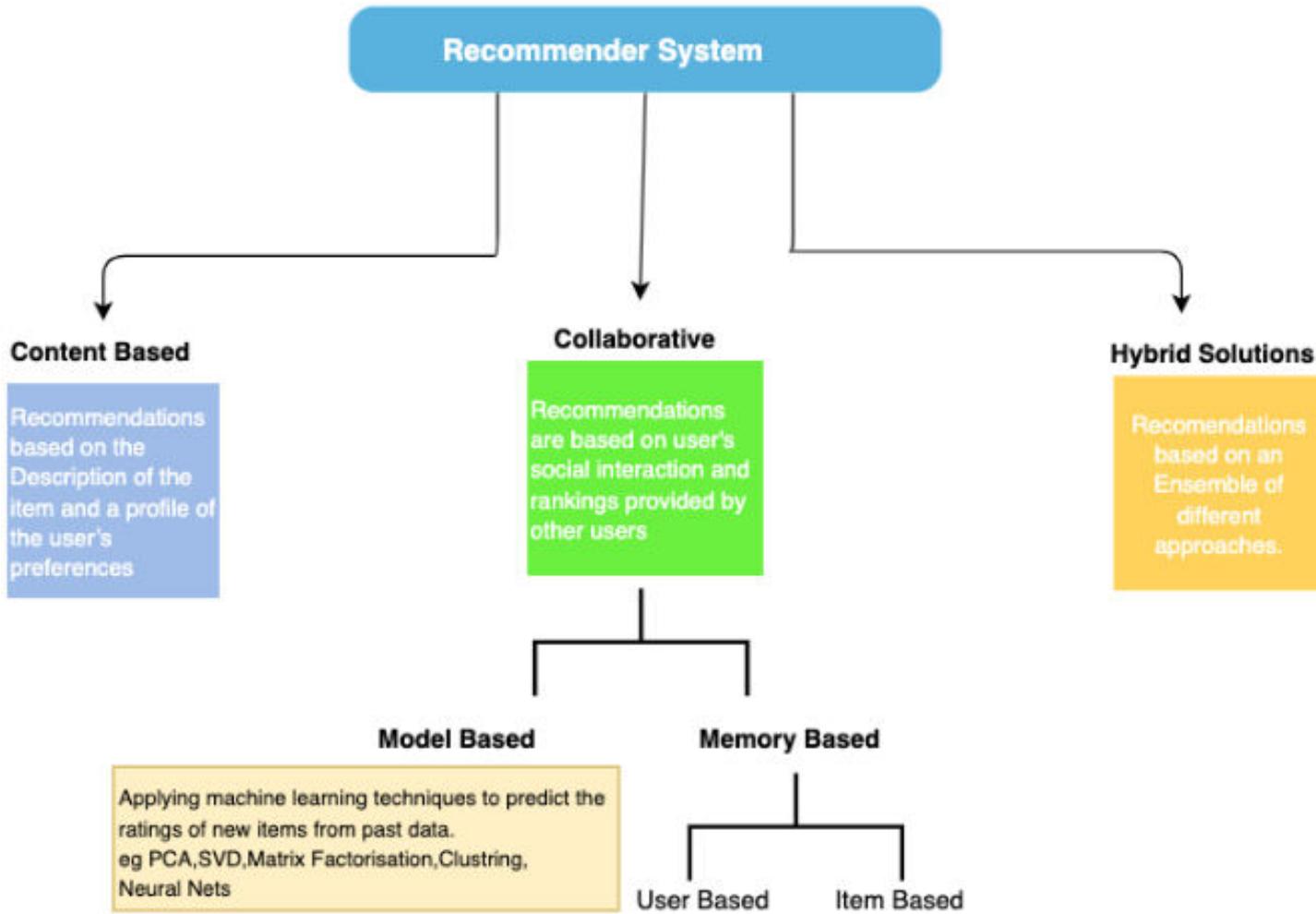


Problems faced while building recommender systems

- Gathering the rating from users.
 - Developing right models to learn function from known ratings
 - Evaluating the models for unknown rating
-



Types of Recommender systems



What is collaborative filtering

For each user, recommender systems recommend items based on how similar users liked the item

Two approaches

- Memory based :
 - **User-based:** measure the similarity between target users and other users
 - **Item-based:** measure the similarity between the items that target users rates/ interacts with and other items
 - Model based
 - SVD,PCA.. etc
-

Baseline approach for rating prediction

Predicted Rating(E,Equilibrium) = $\mu + b_E + b_{\text{Equilibrium}}$

Where μ is the global average

b_E is the deviation of the user E

$b_{\text{Equilibrium}}$ is the deviation of the Movie Equilibrium

Predicted Rating = $3.78+0.22+0.22 = 4.22$

User/Movie	Batman	Alice in Wonderland	Dumb and Dumber	Equilibrium
User A	4		3	5
User B		5	4	
User C	5	4	2	
User D	2	4		3
User E	3	4	5	?



User Based Collaborative Filtering

- Identify the set of items rated by the target user.
- Identify which other users have rated the same items as target user.
- Compute similarity of each user to the target user.
- Select top K similar users

User/Movie	Batman	Alice in Wonderland	Dumb and Dumber	Equilibrium
User A	4		3	5
User B		5	4	
User C	5	4	2	
User D	2	4		3
User E	3	4	5	?

Finding similar users

- Let r_x and r_y be the vector of users x and y 's ratings
- Jaccard similarity measure between x and y
 - Problem: Ignores the value of the rating
- Cosine similarity measure
- $\text{sim}(x, y) = \cos(r_x, r_y) = \frac{r_x \cdot r_y}{\|r_x\| \cdot \|r_y\|}$
- Problem: Treats missing ratings as “negative”
- Pearson correlation coefficient

➤ S_{xy} = items rated by both users x and y

$$\text{Sim}(x, y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \bar{r}_x)(r_{ys} - \bar{r}_y)}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \bar{r}_x)^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \bar{r}_y)^2}}$$

--

Jaccard similarity

Let r_x and r_y be the vector of users x and y 's ratings

User/Movie	Batma n	Alice in Wonderland	Dumb and Dumber	Equilibriu m
User A	4		3	5
User B		5	4	
User C	5	4	2	
User D	2	4		3
User E	3	4	5	?

- $\text{sim}(A,B) = |r_A \cap r_B| / |r_A \cup r_B|$

$$\text{Sim}(A,B)=1/4 = 0.25; \text{ Sim}(A,C)=2/4 = 0.5$$

$$\text{Sim}(A,B) < \text{Sim}(A,C)$$

➤ **Problem:** Ignores the value of the rating

Cosine similarity

- Let r_x and r_y be the vector of users x and y 's ratings

User/Movie	Batma n	Alice in Wonderland	Dumb and Dumber	Equilibriu m
User A	4		3	5
User B		5	4	
User C	5	4	2	
User D	2	4		3
User E	3	4	5	?

$$\cos \theta = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

Sim(A,B)=0.265

Sim(A,C)=0.54

Sim(A,B) < Sim(A,C)

Problem: Treats missing ratings as “negative”

Centered cosine

- Normalize ratings by subtracting row mean

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	2/3			5/3	-7/3		
B	1/3	1/3	-2/3				
C				-5/3	1/3	4/3	
D		0					0

$$\text{sim}(A, B) = \cos(r_A, r_B) = 0.09; \text{sim}(A, C) = -0.56$$

- $\text{sim}(A, B) > \text{sim}(A, C)$



Estimating the Predicted Rating

- Let N be the set of k users most similar to x who have rated item i
- **Prediction for item s of user x :**

$$r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$$

$$r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$$

Item-item based collaborative filtering

- For item i , find other similar items
- Estimate rating for item i based on ratings for similar items
- Can use same similarity metrics and prediction functions as in user-user model

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

s_{ij} ... similarity of items i and j

r_{xj} ... rating of user u on item j

$N(i;x)$... set items rated by x similar to i

Item Based Collaborative Filtering

	users												
	1	2	3	4	5	6	7	8	9	10	11	12	$\text{sim}(1,m)$
movies	1	1		3		?	5			5		4	1.00
	2			5	4			4			2	1	3
	3	2	4		1	2		3		4	3	5	0.41
	4		2	4		5			4			2	-0.10
	5			4	3	4	2					2	-0.31
	6	1		3		3			2			4	0.59

Neighbor selection:

Identify movies similar to movie 1, rated by user 5

- 1) Subtract mean rating m_i from each movie i

$$m_1 = (1+3+5+5+4)/5 = 3.6$$

row 1: [-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0]

- 2) Compute cosine similarities between rows

	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

$\text{sim}(1,m)$

1.00

-0.18

0.41

-0.10

-0.31

0.59

Compute similarity weights:

$s_{1,3}=0.41$, $s_{1,6}=0.59$

	1	2	3	4	5	6	7	8	9	1	1	1
1	1		3		2.6	5			5	0	1	2
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2				2	5	
6	1		3		3			2			4	

Predict by taking weighted average:

$$r_{1.5} = (0.41 \cdot 2 + 0.59 \cdot 3) / (0.41 + 0.59) = 2.6$$

$$r_{ix} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{jx}}{\sum s_{ij}}$$

User Based Vs Item Based

- User based Similarity is more **dynamic** and can be used if
 - Item base is smaller than user base
 - Item base changes rapidly
- Item based Similarity is **static** and recommends new items that were also liked by the same users
 - Good if the use base is small



Issues in implementing Collaborative Filtering

- Many Items to choose from
 - No data for new users
 - Very large datasets
-



Matrix factorization

SVD Theorem

$$\mathbf{A}_{[m \times n]} = \mathbf{U}_{[m \times r]} \Sigma_{[r \times r]} (\mathbf{V}_{[n \times r]})^T$$

A: Input data matrix

- $m \times n$ matrix (e.g., m users, n movies)

U: Left singular vectors

- $m \times r$ matrix (m users, r concepts)

Σ : Singular values

- $r \times r$ diagonal matrix (strength of each ‘concept’)
(r : rank of the matrix \mathbf{A})

V: Right singular vectors

- $n \times r$ matrix (n movies, r concepts)



Singular Value Decomposition

- The key issue in an SVD decomposition is to **find a lower dimensional feature space** where the **new features represent “concepts”** and the **strength of each concept** in the context of the collection is computable.
- The core of the SVD algorithm lies in the following theorem
- It is always possible to decompose a given matrix A into $A = U \Sigma V^T$.

Example

$$\begin{bmatrix} 4 & 1 & 1 & 4 \\ 1 & 4 & 2 & 0 \\ 2 & 1 & 4 & 5 \end{bmatrix} =
 \begin{bmatrix} -0.61 & 0.28 & -0.74 \\ -0.29 & -0.95 & -0.12 \\ -0.74 & 0.14 & 0.66 \end{bmatrix} \times
 \begin{bmatrix} 8.87 & 0.00 & 0.00 & 0.00 \\ 0.00 & 4.01 & 0.00 & 0.00 \\ 0.00 & 0.00 & 2.51 & 0.00 \end{bmatrix} \times
 \begin{bmatrix} -0.47 & -0.28 & -0.47 & -0.69 \\ 0.11 & -0.85 & -0.27 & 0.45 \\ -0.71 & -0.23 & 0.66 & 0.13 \\ -0.52 & 0.39 & -0.53 & 0.55 \end{bmatrix}$$

User to Movie Utility Matrix User to Concept Strength of each concept Movie to Concept

$$\begin{bmatrix} 2.69 & 0.57 & 2.22 & 4.25 \\ 0.78 & 3.93 & 2.21 & 0.04 \\ 3.17 & 1.38 & 2.92 & 4.78 \end{bmatrix} =
 \begin{bmatrix} -0.61 & 0.28 \\ -0.29 & -0.95 \\ -0.74 & 0.14 \end{bmatrix} \times
 \begin{bmatrix} 8.87 & 0.00 \\ 0.00 & 4.01 \end{bmatrix} \times
 \begin{bmatrix} -0.47 & -0.28 & -0.47 & -0.69 \\ 0.11 & -0.85 & -0.27 & 0.45 \end{bmatrix}$$

Reconstructed Matrix User to Concept Strength of each concept Movie to Concept

Recommendations for new User

- How to use SVD for recommendations?

$$\mathbf{u}_{new} = \mathbf{u} \times \mathbf{V}_{m \times r} \times \mathbf{S}^{-1}_{r \times r}$$

$$\begin{bmatrix} 8.87 & 0.00 \\ 0.00 & 4.01 \\ 0.00 & 0.00 \end{bmatrix}$$

Strength of each concept

$$\begin{bmatrix} -0.23 & -0.89 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} -0.47 & 0.11 \\ -0.28 & -0.85 \\ -0.47 & -0.27 \\ -0.69 & 0.45 \end{bmatrix} \times \begin{bmatrix} 0.11 & 0.00 \\ 0.00 & 0.25 \end{bmatrix}$$

$$\begin{bmatrix} -0.47 & -0.28 & -0.47 & -0.69 \\ 0.11 & -0.85 & -0.27 & 0.45 \end{bmatrix}$$

Movie to Concept

Drawbacks of SVD

- Conventional SVD is undefined for incomplete matrices!
 - Imputation to fill in missing values
 - Increases the amount of data
 - We need an approach that can simply ignore missing ratings
-



User – Movie interactions in real world

User/ Movie	M1	M2	M3	M4	M5
User1	4	4	4	4	4
User2	4	4	4	4	4
User3	4	4	4	4	4
User4	4	4	4	4	4

User/ Movie	M1	M2	M3	M4	M5
User1	1	3	2	5	4
User2	2	1	1	1	5
User3	3	2	3	1	5
User4	2	4	1	5	2

User/ Movie	M1	M2	M3	M4	M5
User1	3	1	1	3	1
User2	1	2	4	1	3
User3	3	1	1	3	1
User4	4	3	5	4	4

Movie / User Features

➤ Movie Features



Comedy



Action



Horror



Drama

➤ User Preferences

- Likeliness of genre of movies
(0/1)

Movie / User Features

Movie/User	comedy	action
M1	3	1
M2	1	2
M3	1	4
M4	3	1
M5	1	3

User/Movie	Comed y	Action
User1	1	0
User2	0	1
User3	1	0
User4	1	1



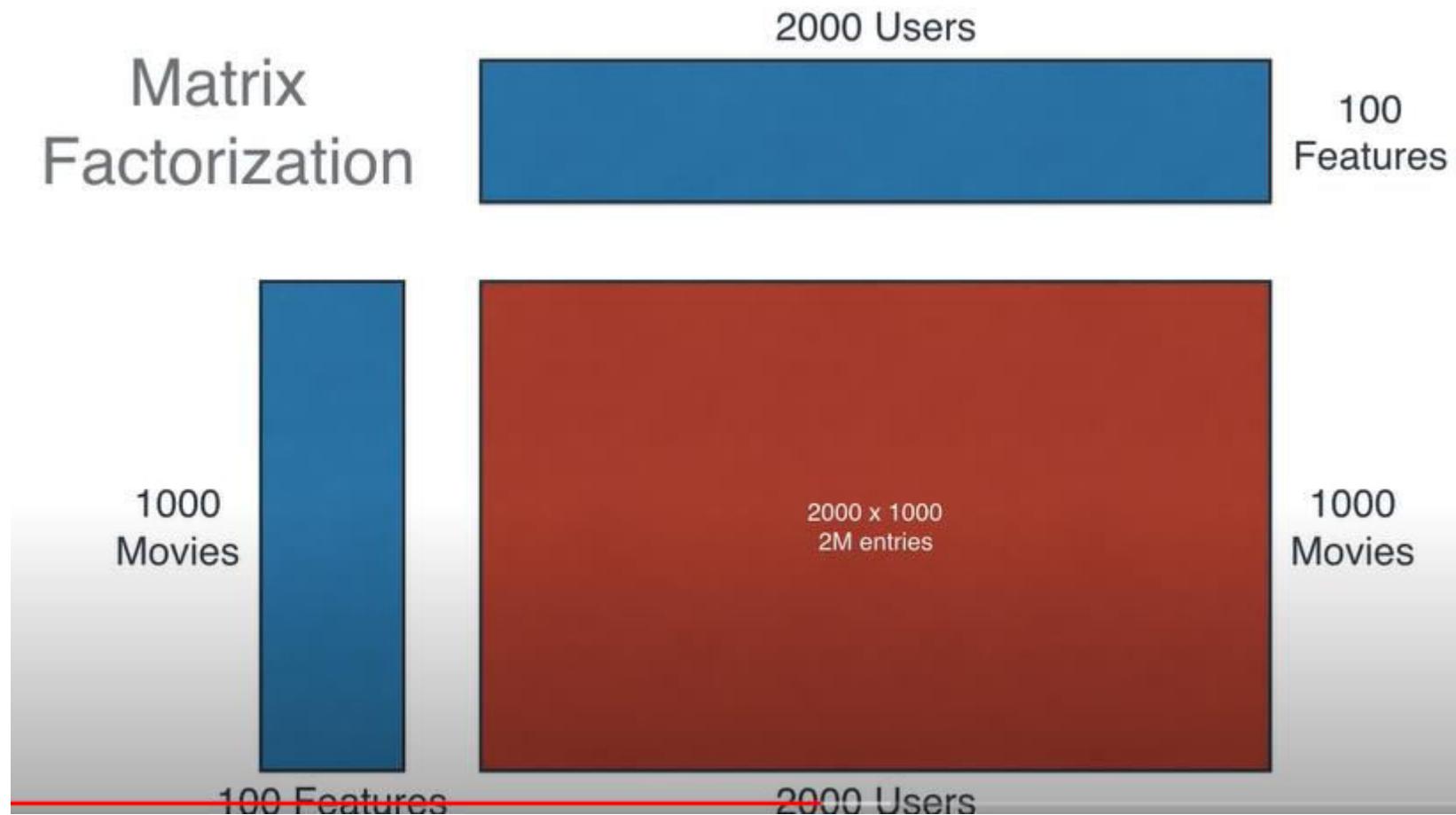
Latent Factors using Matrix Factorization

Movie/ Features	M1	M2	M3	M4	M5
Comedy	3	1	1	3	1
Action	1	2	4	1	3

User/Feature s	Comed y	Actio n
User1	1	0
User2	0	1
User3	1	0
User4	1	1

User/ Movie	M1	M2	M3	M4	M5
User1	3	1	1	3	1
User2	1	2	4	1	3
User3	3	1	1	3	1
User4	4	3	5	4	4

Benefit of matrix factorization-Storage



Stochastic Gradient Descent (SGD)

Movie/ Features	M1	M2	M3	M 4	M 5
Comedy	1.2	3.1	0.3	2.5	0.2
Action	2.4	1.5	4.4	0.4	1.1

User/Fe atures	F1	F2
User1	0.2	0.5
User2	0.3	0.4
User3	0.7	0.8
User4	0.4	0.5

User/Movie	M1	M2	M3	M4	M5
User1	1.44	1.37	2.26	0.7	0.59
User2	1.32	1.53	1.85	0.91	0.5
User3	2.76	3.37	3.73	2.07	1.02
User4	1.68	1.99	2.32	1.2	0.63



User/ Movie	M1	M2	M3	M4	M5
User1	3	1	1	3	1
User2	1	2	4	1	3
User3	3	1	1	3	1
User4	4	3	5	4	4

Rating predictions

Movie/ Features	M1	M2	M3	M4	M5
Comedy	3	1	1	3	1
Action	1	2	4	1	3

User/Feature s	Comedy	Action
User1	1	0
User2	0	1
User3	1	0
User4	1	1

User/ Movie	M1	M2	M3	M4	M5
User1	3		1		1
User2	1		4	1	
User3	3	1		3	1
User4		3		4	4

Accuracy of estimated rating / Error Based

- Mean Absolute Error (MAE)
- Mean square error (MSE)
- Root Mean Squared Error(RMSE)

User/Movie	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie6
User1	2			4	4	
User2	5		4			1
User3			5		2	
User4		1		5		4
User5			4			2
User6	4	Training Set	5	1	Test Set	

Accuracy of estimated rating / Error Based (Contd...)

User Id	Movie Id	Actual	Predicted	MAE	MSE	RMSE
User1	4	4	2	2	4	4
User1	5	4	1	3	9	9
User2	6	1	1.5	0.5	0.25	0.25
User3	5	2	2	0	0	0
User4	4	5	4.5	0.5	0.25	0.25
User4	6	4	3	1	1	1
User5	6	2	2	0	0	0
User6	4	1	3	2	4	4
				9/8	18.5/8	Sqrt(18.5/8)

$$MAE = \frac{1}{n} \sum_{j=1}^n |y_j - \hat{y}_j|$$

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{n}}$$

Precision at rank K

Movie	User 1	Actual	Predicted
1	1	4	2.3
2	1	2	3.6
3	1	3	3.4
4	1	?	4.4
5	1	5	4.5
6	1	?	2.3
7	1	2	4.9
8	1	?	4.3
9	1	?	3.3
10	1	4	4.3

Ignore the values whose actual is not known and sort the items in Descending order of the predicted rating.

Item7 2/4.9
 item5 5/4.5
 item10 4/4.3
 item2 2/3.6
 item3 3/3.4
 item1 4/2.3

Let's assume that all rating ≥ 3.5 are relevant then the recommender system will suggest the following

Item7 2/4.9
 item5 5/4.5
 item10 4/4.3



Advantages

-
1. They help users deal with the information overload by giving them recommendations of products, etc.
 2. They help businesses make more profits, i.e., selling more products.
-



Challenges

- Scalability.
- Cold Start
- Imbalanced Dataset.



Thank you



BITS Pilani
Pilani Campus

Information Retrieval

Dr. Vijayalakshmi Anand

BITS pilani

Introduction to neural IR

Outline

- *What is neural IR
- *Types of neural IR
- *Application of Neural IR

What is neural IR

Neural Information Retrieval refers to the use of neural networks for the purpose of retrieving information

Types of Neural IR

- Shallow neural network
- Deep neural network

Application of neural IR

- ad-hoc retrieval
- recommender systems
- multi media search
- conversational systems

Challenges of Traditional IR

- short queries
- documents that may also contain large sections of irrelevant text.
- learn patterns in query and document text

Requirements for the good IR model

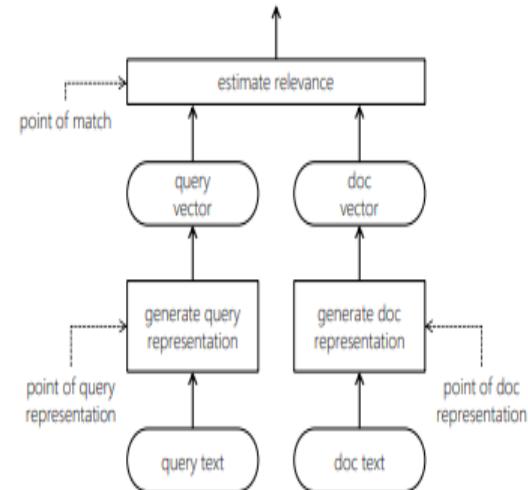
- Semantic understanding
- Robustness to rare inputs
- Robustness to corpus variance
- Robustness to variable length inputs
- Robustness to errors in input
- Sensitivity to context

Traditional IR models

- BM25
- Language modelling (LM)
- Translation models
- Dependence model

Neural approaches to IR

- query representation
- the document represer
- estimating relevance



Unsupervised learning of term representation

Vector representation

- Local representation
- Distributed representation

Vector representation By embedding

- Why do we need embedding
- What is embedding
- Different algorithms for word embedding
 - word 2vec
 - Glove
 - BERT
 - GPT
 - ELMO

Example –Notion of similarity



Bedrooms: 3
Area: 1850 **sqr ft**
Bathrooms: 2.5

Bedrooms: 3
Area: 1700 **sqr ft**
Bathrooms: 2

Bedrooms: 10
Area: 7500 **sqr ft**
Bathrooms: 2



$$\begin{bmatrix} 1 \\ 0.9 \\ 0 \\ 1 \\ 0 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0.87 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.7 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

Cummins and Cummins and Australia are

	ashes	Australia	Bat	Cummins	cover	Dhoni	World cup	..	Zimbabwe
Person	0	0.02	0.1	0.95	0.03	0.96	0.67	...	0.04
Country	0	0.97	0	0	0	0	0	...	1
Healthy & Fit	0	0	0.3	0.87	0	0.9	0	...	0
Event	1	0.1	0	0	0.4	0	1	...	0
gear	0	0	1	0	0	0	0	...	0



BERT-Introduction

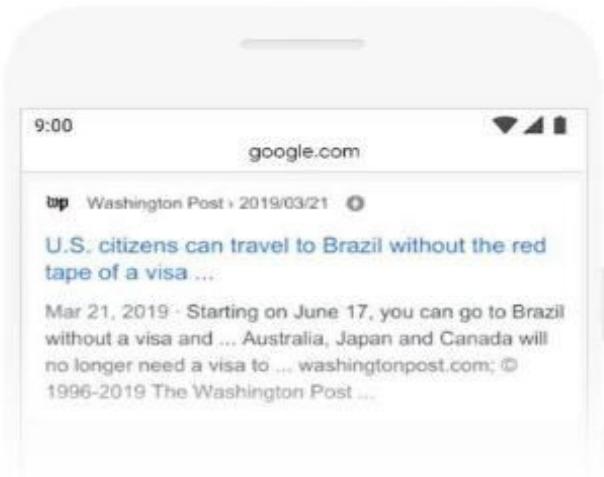


BERT (Bidirectional Encoder Representations from Transformers)

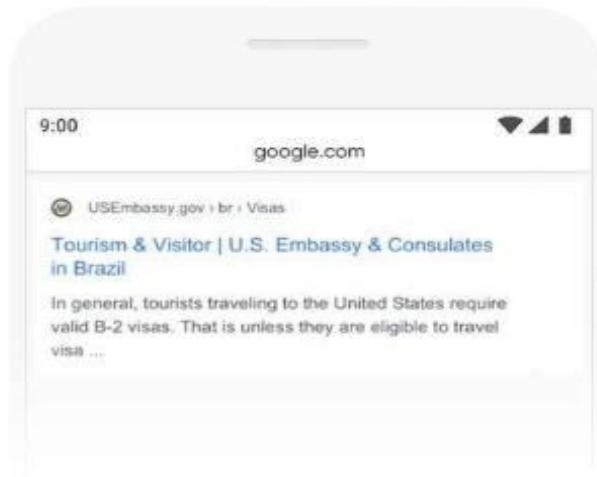
- BERT can generate embeddings for entire sentence
- Vector size is 768 length
- Bert can generate contextualized embeddings

2019 brazil traveler to usa need a visa

BEFORE



AFTER



Important steps in BERT

pre-training

fine-tuning.

-
- Masked language model
 - Next word prediction

Deep neural network

- Input text rep
- One hot representation
- Pre trained embeddings

Standard architecture

- Shift-invariant neural operations

Convolutional and recurrent neural networks

- Autoencoders
- Siamese networks



Thank You!

