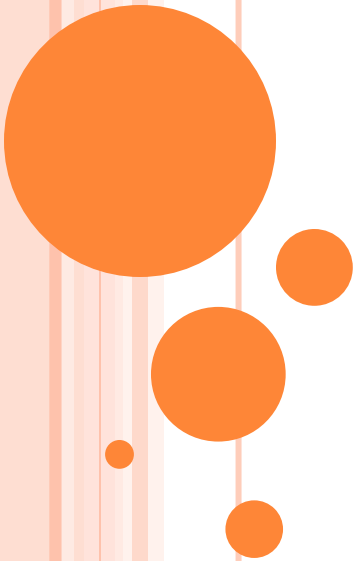
Visual Information Retrieval

Gang Hua

Department of Computer Science
Stevens Institute of Technology

01/22/2014

Acknowledgement:
Thanks Cees Snoek for sharing his course slides!
A portion of the slides in this course is adapted from his slides.



CS598 Visual information Retrieval

Lecture I: Part I: Introduction to the course

Who am I?

- Prof. Gang Hua 华刚
  - Associate Professor of Computer Science
  - Stevens Institute of Technology
  - Research Staff Member (07/2010—08/2011)
  - IBM T J. Watson Research Center
  - Senior Researcher (08/2009—07/2010)
  - Nokia Research Center Hollywood
  - Scientist (07/2006—08/2009)
  - Microsoft Live Labs Research
  - Ph.D. in ECE, Northwestern University, 06/2006

Now, who are you?

- Why do you choose this class?
- What do you expect to learn?
- Previous experience?

Course promise

- You will learn the theory and practice of visual information retrieval.
- You will be able to recall the major scientific problems in visual information retrieval.
- You will be able to understand and explain how state-of-the-art visual search systems work.

Course Information (1)

- **CS598** Visual Information Retrieval
- **Term**: Spring 2014
- **Instructor**: Prof. Gang Hua
- **Class time**: Monday 6:15pm—8:40pm
- **Location**: McLean Chemical Sciences Building 414
- **Office Hour**: Wednesday 4:00pm—5:00pm by appointment
- **Office**: Lieb/Room305
- **Course Assistant**: Haoxiang Li
- **Course Website**: http://www.cs.stevens.edu/~ghua/ghweb/Teaching/CS598Spring2014.htm

Course Information (2)

- **Text Book**:
  - No required text book
- Recommended Reading
  - See course website for recommended
- **Grading**:
  - Class participation: 10%
  - One written homework: 10%
  - 4 course project:
    - Project #1(10%), Project #2(10%), Project #3 (15%), Project #4 (15%)
  - Final Project & Presentation: (15% System demo and presentation, 15% final report)
  - Final grade: A-- 90% to 100%; B--80% to 89%; C-- 65% to 79%; F -- < 65% .

Course information (3)

- Projects are all group project
  - You will be grouped into a team with 5-6 members
- Your final project will be integrating your first four project into an end-to-end image search system
- So it is important that you don't lag behind!!
- Program language is not specified, pick up your favorite one with the team!

Rules

- **Need to be absent from class?**
  - 1 point per class: please send notification and justification at least 2 days before the class
- **Late submission of homework?**
  - The maximum grade you can get from your late homework decreases 50% per day
- **Zero tolerance on plagiarism!!**
  - The first time you receive zero grade for the assignment
  - The second time you get “F” in your final grade
  - Refer to Stevens honor system for your behavior

Prerequisites

- No definite pre-requisite
- However, it would be helpful if you possess
  - Basic knowledge of computer vision, pattern recognition and information retrieval
  - e.g.,
    - CS558 Computer Vision
    - CS559 Machine Learning
- We will learning something about information retrieval today

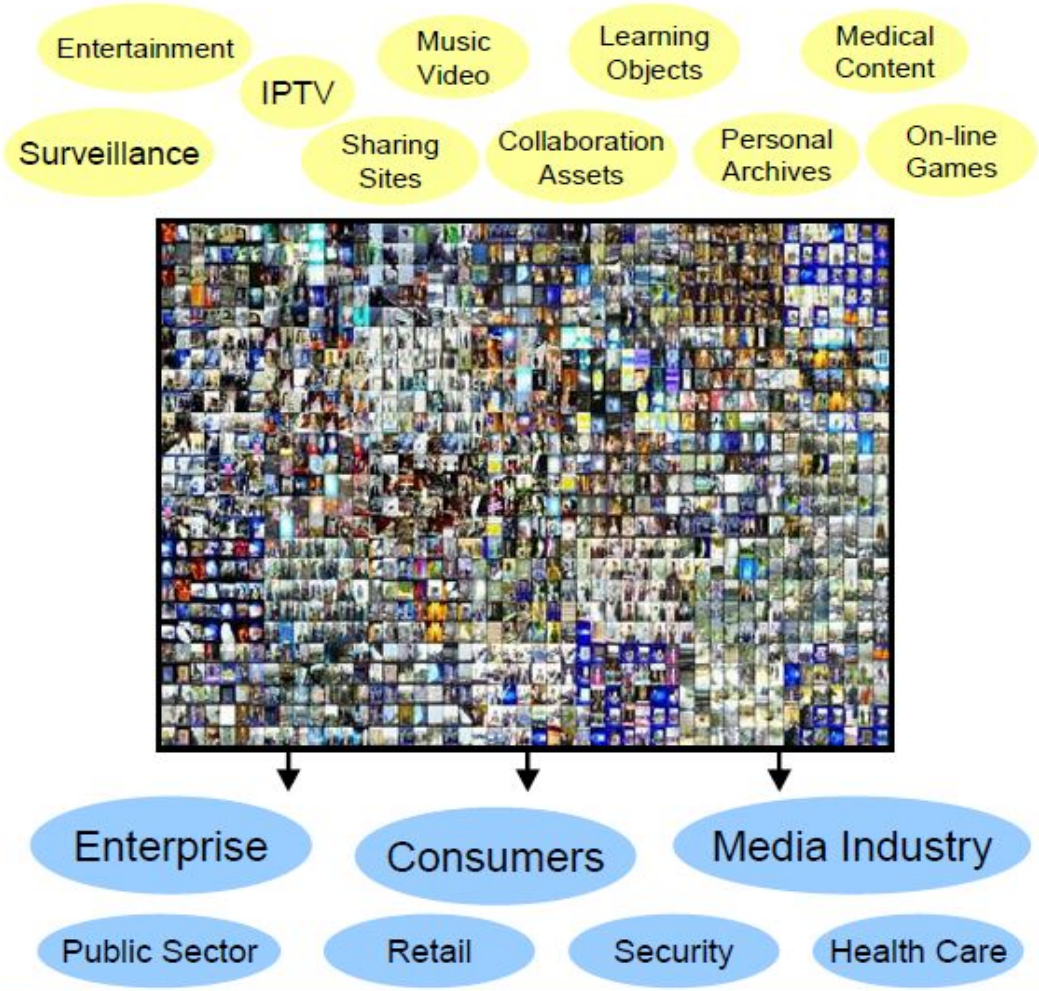
What is a visual Information retrieval?

- Definition
  - Visual information retrieval is the activity of obtaining (**i.e., search and retrieve**) visual information resources relevant to an information need (**i.e., a query**) from a collection of visual information resources (**e.g., image and video database**).

Slide credit: JR Smith, IBM

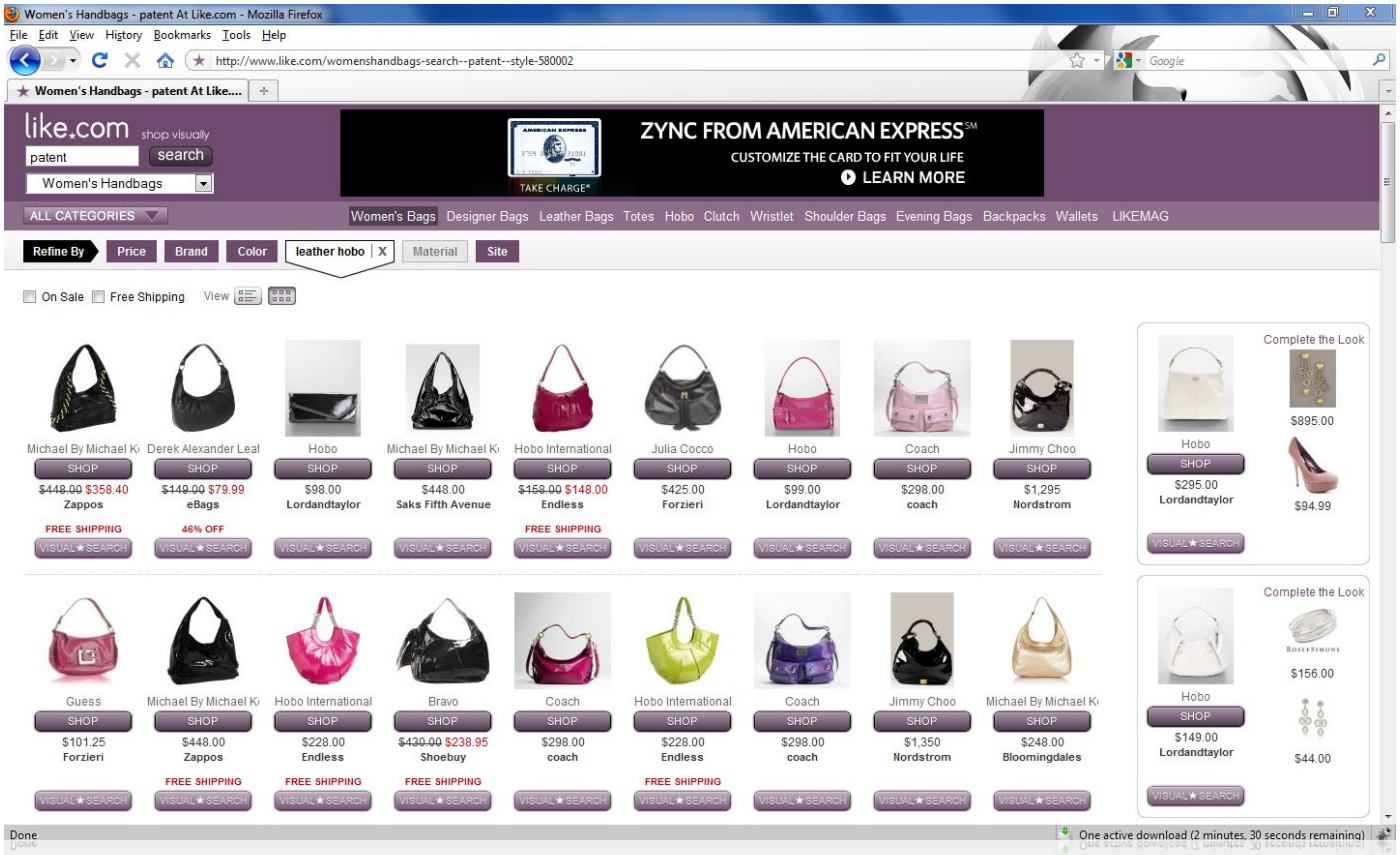
Why needed?

- Growing deluge requires more effective solutions for organizing, managing & searching video content
- Manual indexing is costly, time-consuming and inadequate
- New technologies are needed to automate processing and unlock value of large repositories
- Metadata standards are needed to support interoperable search



Why bother?

> $100 million



Slide credit: C. Snoek (Univ. of Amsterdam)

$1.65 billion

Still not convinced?

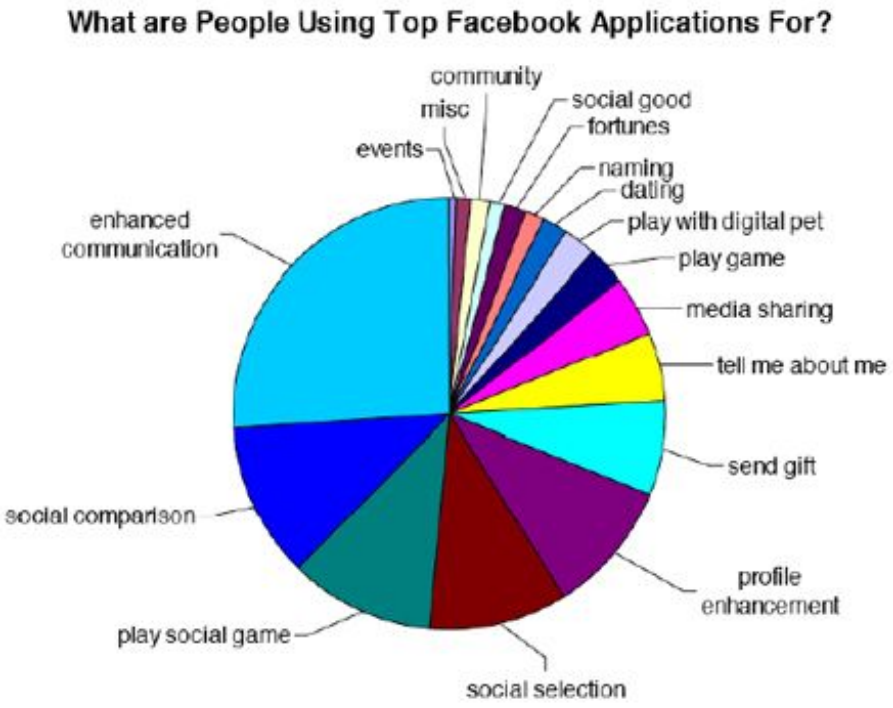
Slide credit: C. Snoek (Univ. of Amsterdam)

Quiz: What is the size of this problem?

- How many images uploads to Facebook?
- How many uploads on YouTube per day?
- What is the size of a typical national broadcast video archive?

Statistics from Facebook

- 350 millions of photos uploaded to Facebook daily
- Photo application was NO.1 utilized application for a long time since the inception of Facebook



Statistics from YouTube

- More than 1 billion unique users visit YouTube each month
- Over 6 billion hours of video are watched each month on
- 100 hours of video are uploaded to YouTube every minute
- 80% of YouTube traffic comes from outside the US
- YouTube is localized in 61 countries and across 61 languages
- According to Nielsen, YouTube reaches more US adults ages 18-34 than any cable network
- Millions of subscriptions happen each day. The number of people subscribing daily is up more than 3x since last year, and the number of daily subscriptions is up more than 4x since last year

Answer from the Netherlands

- Europe's largest digitization project

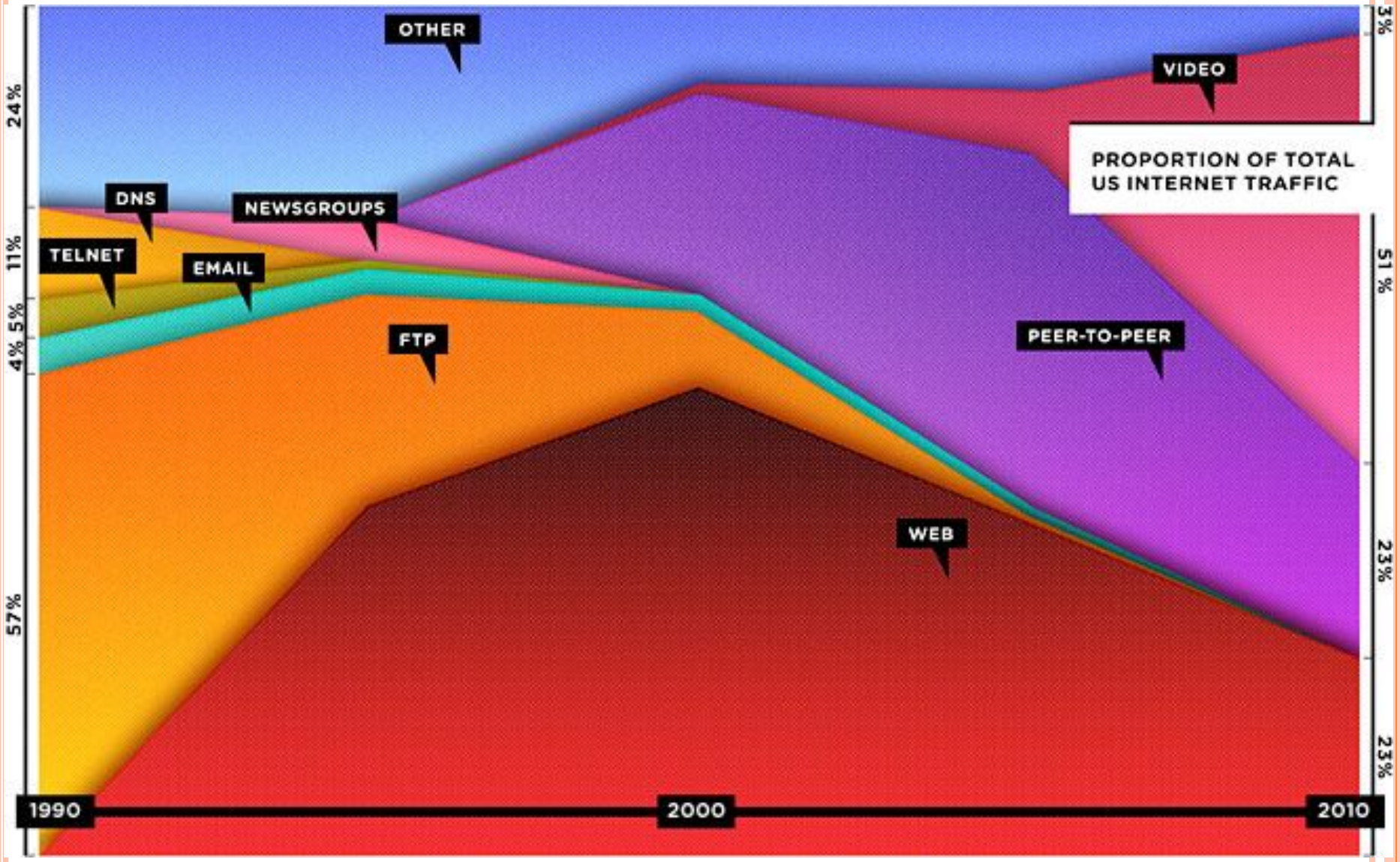
BEELDEN VOOR DE TOEKOMST

- Yearly ingest
  - 15,000 hours of video
  - 40,000 hours of radio
  - >1 peta-byte per year
- Next 6 years
  - 137,200 hours of video
  - 22,510 hours of film
  - 123,900 hours of audio
  - 2,900,000 photo's

Lack of metadata

Slide credit: C. Snoek (Univ. of Amsterdam)

Latest update from Wired (the web is dying!)



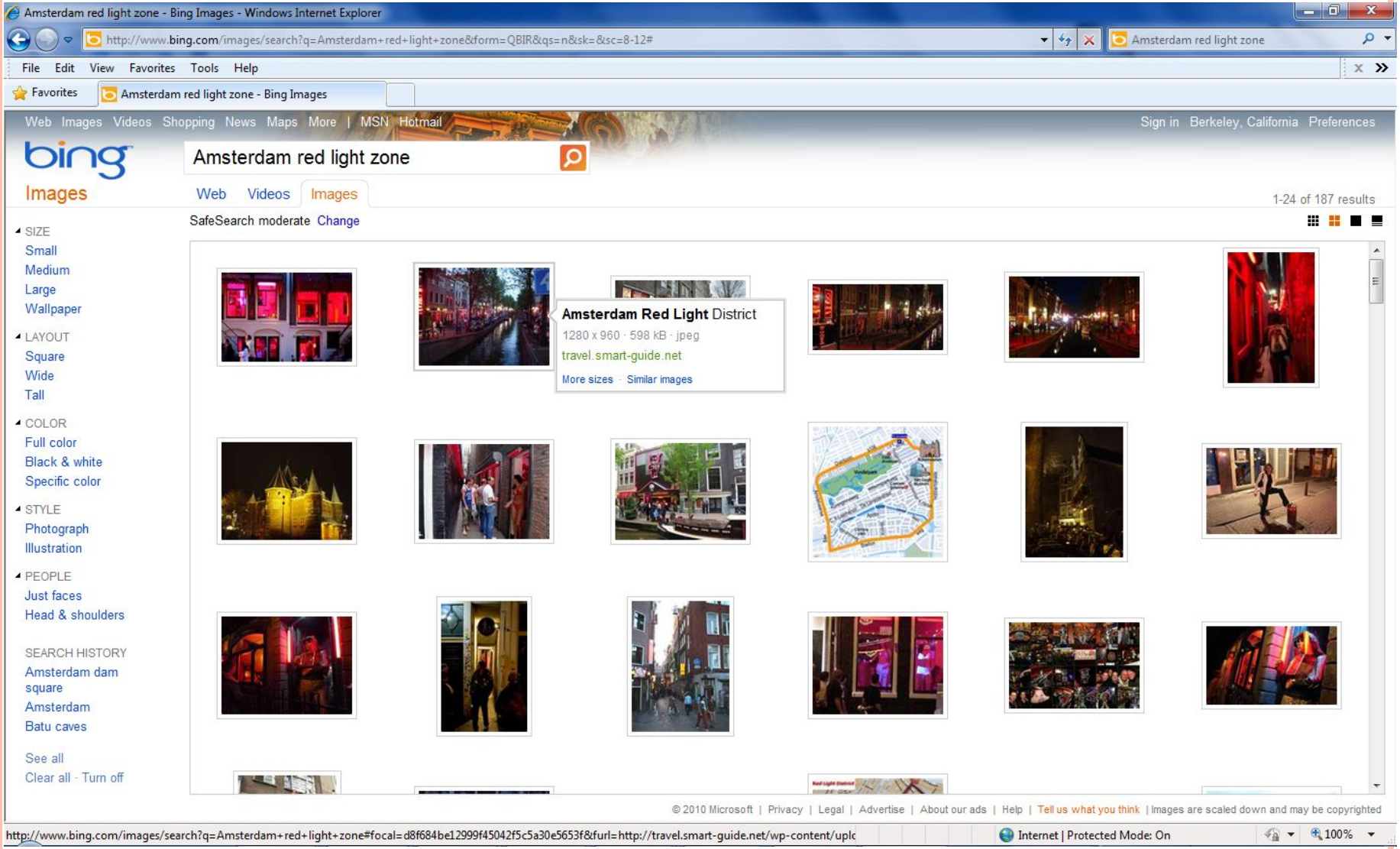
Visual information retrieval: Three cues

- Expert-driven search
  - Exploit professional annotations
- Crowd-driven search
  - Exploit what others are saying about the image
- Content-driven search
  - Exploit the content of the images and video

http://e-culture.multimedian.nl

Expert driven search

Crowd-driven search



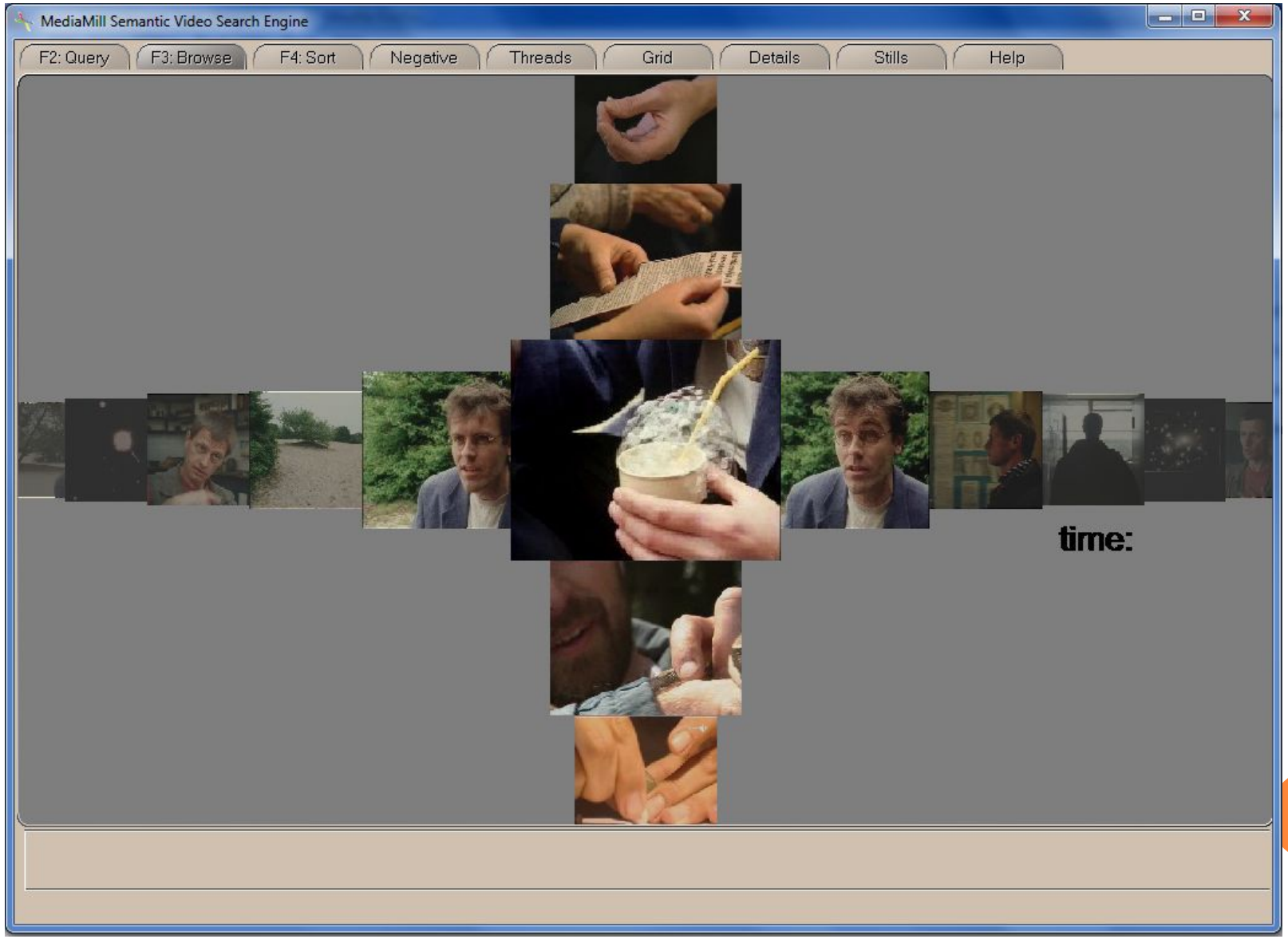
Quiz: What's the limitations of meta data based search?

Issue	What's wrong
Too sparse	Few video objects have any metadata
Inadequate	Mainly tags or few keywords, program-guide info for broadcast video, speech available in few cases
Coarse-grain	At level of digital objects only
Not visual	Does not describe what is visually depicted
Ambiguous	Taxonomies not widely used; folksonomies creating new problems
Inconsistent	Vocabularies and taxonomies not standardized
Subjective	Limited verification across users
Not trustworthy	Professional metadata mixed-in with noise

Slide credit: JR Smith, IBM

http://www.mediamill.nl

Content-driven search



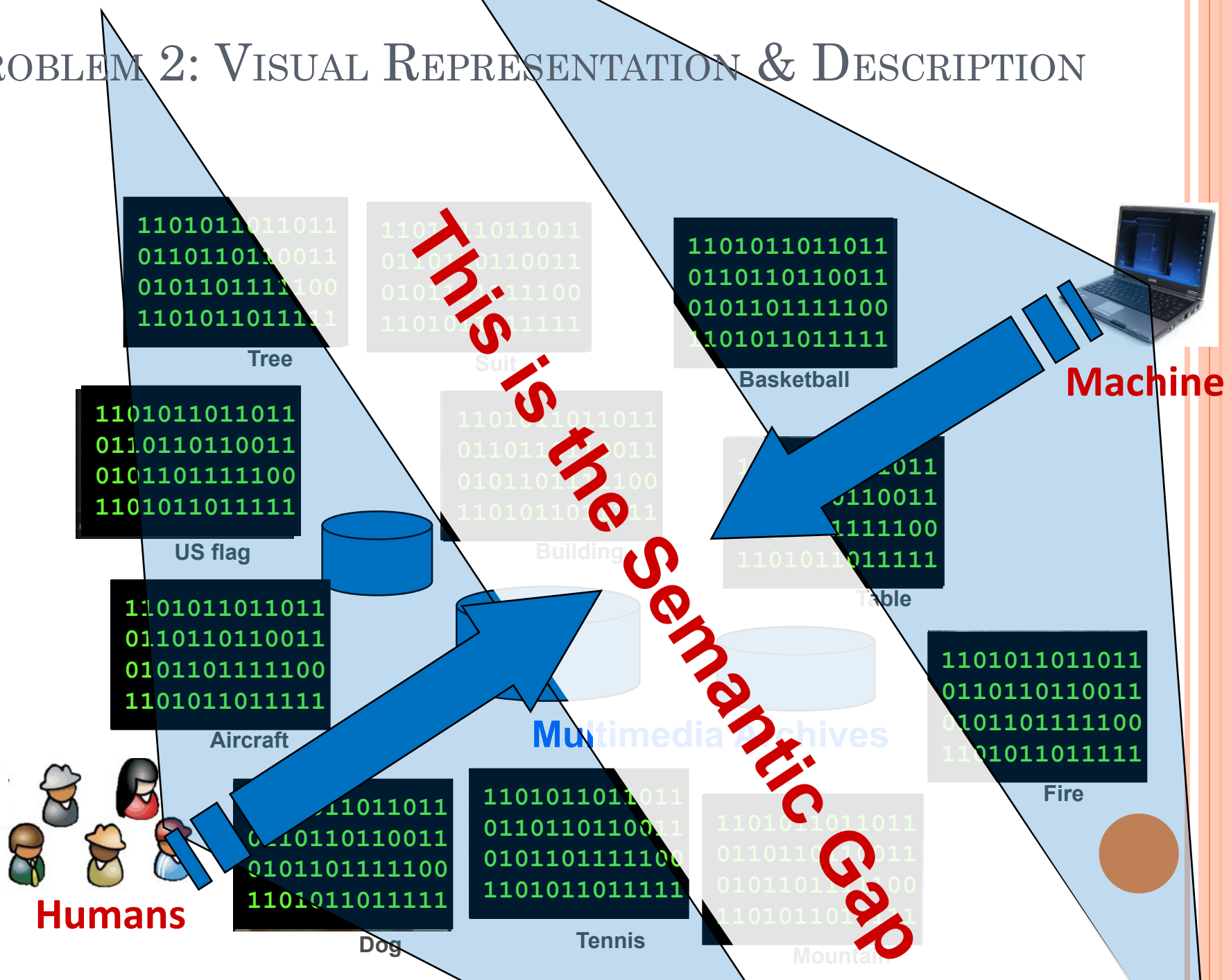
What are the challenges?

Problem 1: Visual Variation

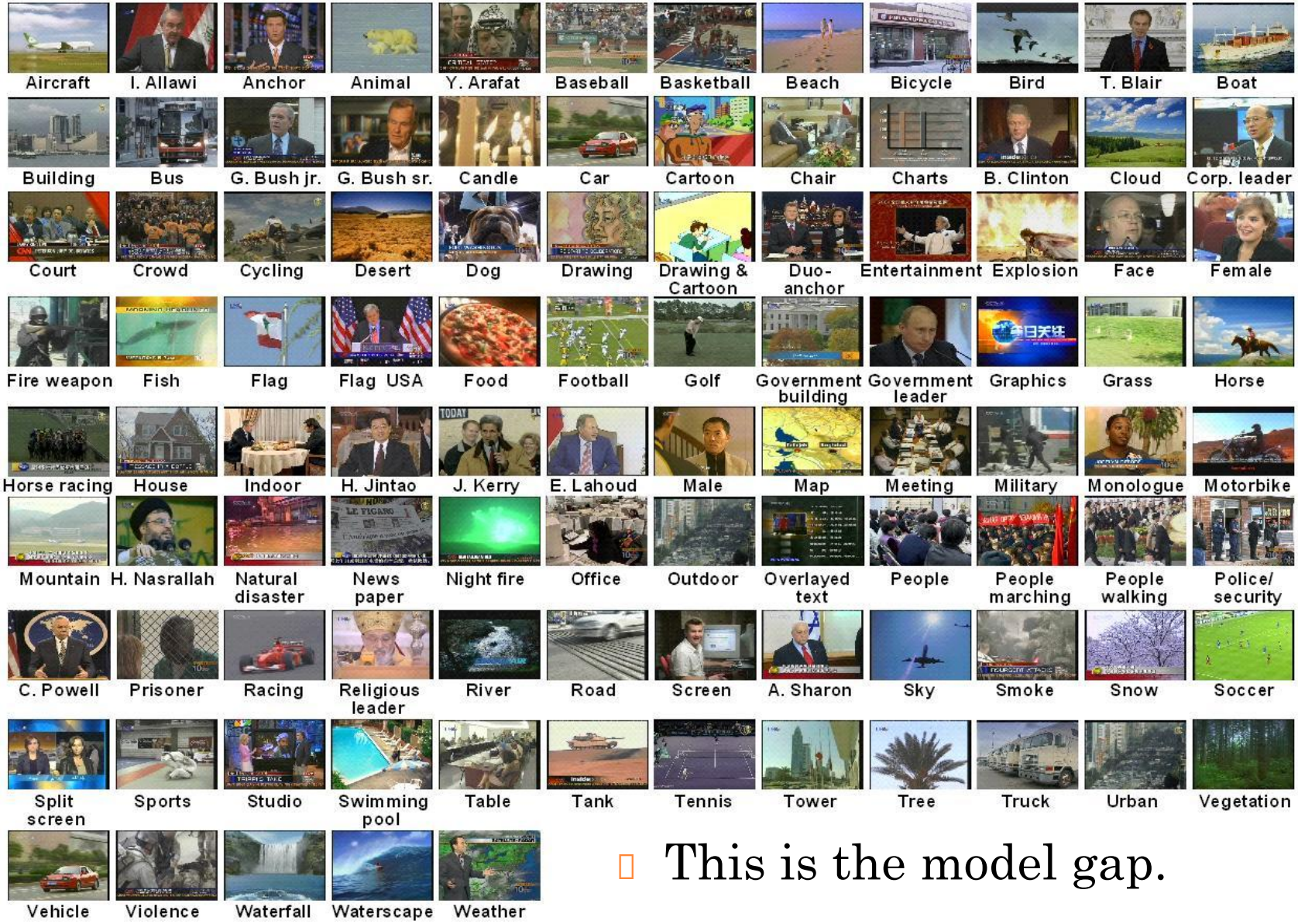
- Many images of one thing, due to minor differences in
  - Illumination, Background, Occlusion, Viewpoint, …

This is also called the “sensory gap.”

Problem 2: Visual Representation & Description



Problem 3: Many things in the world



- This is the model gap.

Problem 4: Query variation

Find shots of people shaking hands

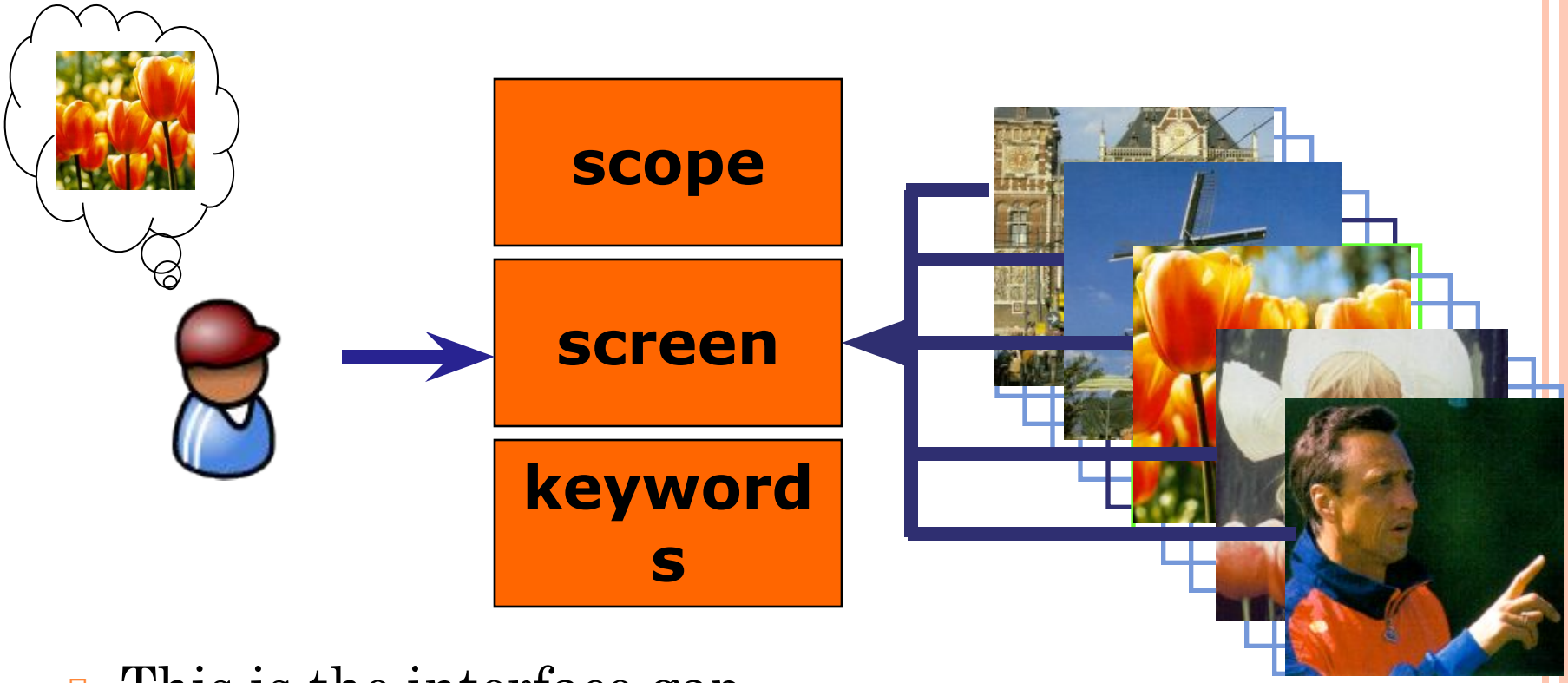
Query Prediction

- **Query-by-keyword**
- **Query-by-concept**
- **Query-by-example**
- **Query-by-humming**
- **Query-by-gesture**

This is the query-context gap

Any combination, any sequence?

Problem 5: Use is open-ended

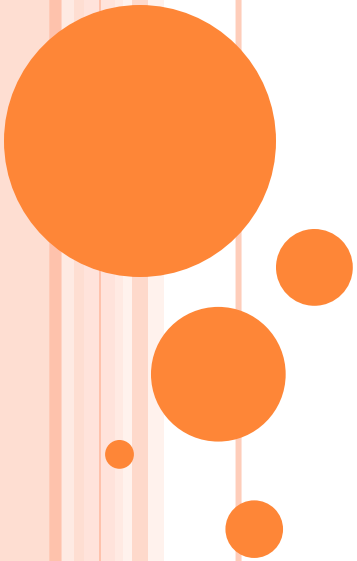


- This is the interface gap

Summary on problems

- Visual search is a diverse and challenging research topic
  - Sensory gap
  - Semanic gap
  - Model gap
  - Query-context gap
  - Interface gap


Q&A



CS598 Visual Information Retrieval

Lecture I: Part II: Basics of Information Retrieval

Information Retrieval (IR)

- Definition
  - Information retrieval is the activity of obtaining (**i.e., search and retrieve**) information resources relevant to an information need (**i.e., a query**) from a collection of information resources (**e.g., a document database**).

Basic assumptions of IR

- Collection: Fixed set of documents
- Goal: Retrieve documents with information that is relevant to the user's information need and helps the user complete a task

How good are the retrieved docs?

- *Precision* : The fraction of retrieved docs, i.e., those are claimed by the retrieval algorithm to be relevant to the query, that are ***actually relevant*** to user's information need
- *Recall* : The fraction of relevant docs in the collection that are retrieved (i.e., the collection of document that are claimed by the retrieval algorithm to be relevant to the query).
- More measurements to follow in later lectures

Information retrieval in 1680

- Which plays of Shakespeare contain the words ***Brutus*** *AND* ***Caesar*** but *NOT* ***Calpurnia***?
- One could grep all of Shakespeare's plays for ***Brutus*** and ***Caesar,*** then strip out lines containing ***Calpurnia***, but…
  - Slow (for large corpora)
  - *NOT* ***Calpurnia*** is non-trivial
  - Other operations (e.g., find the word ***Romans*** near ***countrymen***) not feasible

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

1 if play contains word, 0 otherwise

Boolean query solving for dummies

- Computing the answer to the Boolean query

Brutus *AND* ***Caesar*** *BUT NOT* ***Calpurnia***

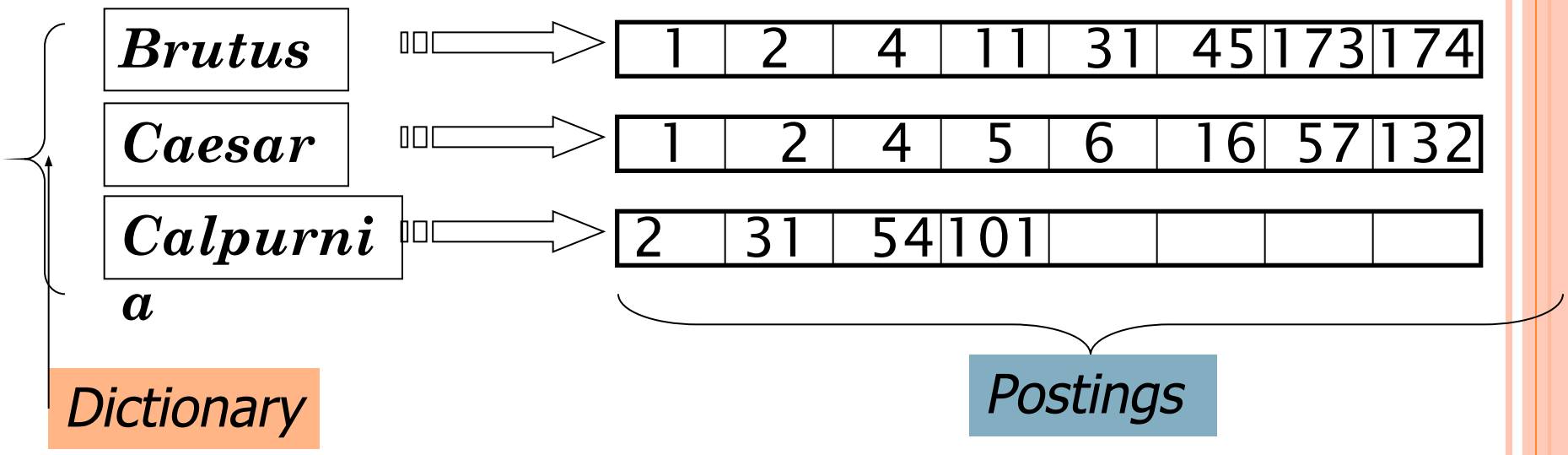
- Take the vectors for ***Brutus, Caesar*** and ***Calpurnia*** (complemented) □ bitwise *AND*.
  - 110100 *AND* 110111 *AND* 101111 = 100100.

Quiz

- Bitwise AND is slow and memory intensive for large text collections, how to speed this up?

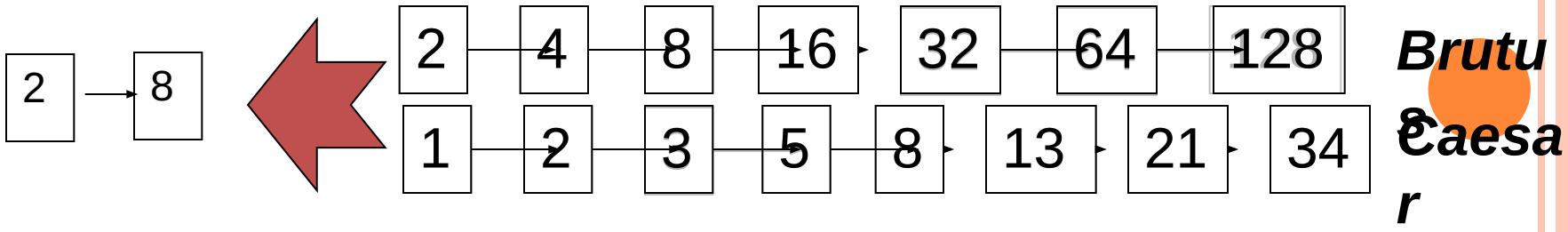
Inverted index

- For each term *t*, we must store a list of all documents that contain *t*.
  - Identify each by a **docID**, a document serial number



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:



Boolean retrieval model

- 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 the collection
- Not good for the majority of users
  - Most users incapable or too lazy to write Boolean queries
  - Most users don't want to wade through 1000s of results

Ranked retrieval models

- The system returns an ordering over the (top) documents in the collection with respect to a query

- Free text queries
  - The user's query is just one or more words in a human language

Scoring as the basis of ranked retrieval

- We wish to return an ordered set of documents most likely to be useful to the searcher

- How can we rank-order the documents in the collection with respect to a query?
  - Assign a score – say in [0, 1] – to each document
  - This score measures how well document and query "match".

Term-document frequency matrices

- Consider the number of occurrences of a term in a document:
  - Each document is a count vector in $\mathbb{N}^{|V|}$: a column

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

Bag of words model

- Vector representation doesn't consider the ordering of words in a document
  - *"John is quicker than Mary"* and "*Mary is quicker than John"* have the same vectors

- This is called the 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.

- Raw term frequency is not what we want:
  - A document with 10 occurrences of the term is more relevant than a document with 1 occurrence of the term.
  - But not 10 times more relevant.

- Relevance does not increase proportionally with term frequency

Log frequency weighting

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

- Score for a document t in both q and d:

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

  - The score is 0 if none of the query terms is present in the document

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

Document frequency (df)

- 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 to the query *arachnocentric*

    → We want a high weight for rare terms like *arachnocentric*

Document frequency (cont'd)

- Frequent terms are less informative than rare terms
- Consider a query term that is frequent in the collection (e.g., *high, increase, line*)
  - A document containing such a term is more likely to be relevant than a document that doesn't
  - But it's not a sure indicator of relevance.
- For frequent terms, we want high positive weights but lower weights than for rare terms.
  - We will use document frequency (df) to capture this.

IDF WEIGHT

- df_t is the document frequency of t: the number of documents that contain t
  - df_t is an inverse measure of the informativeness of t
  - $df_t \leq N$
- We define the idf (inverse document frequency) of t by
  - We use log (N/df_t) instead of N/df_t to "dampen" the effect of idf.

$$idf_t = \log_{10}(N/df_t)$$

IDF example (N=1million)

term	df_t	idf_t
calpurnia	1	
animal	100	
sunday	1,000	
fly	10,000	
under	100,000	
the	1,000,000	

$$\text{idf}_t = \log_{10}(N/\text{df}_t)$$

There is one idf value for each term t in a collection

TF.IDF WEIGHTING

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

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

- Best known weighting scheme in information retrieval
- Increases with
  - the number of occurrences within a document
  - the rarity of the term in the collection

Final ranking of documents for a query

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

Binary → count → weight matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

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

Documents as vectors

- So we have a |V|-dimensional 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 a web search engine
  - These are very sparse vectors - most entries are zero

Queries as vectors

- **Key idea 1:** Do the same for queries: represent them as vectors in the space

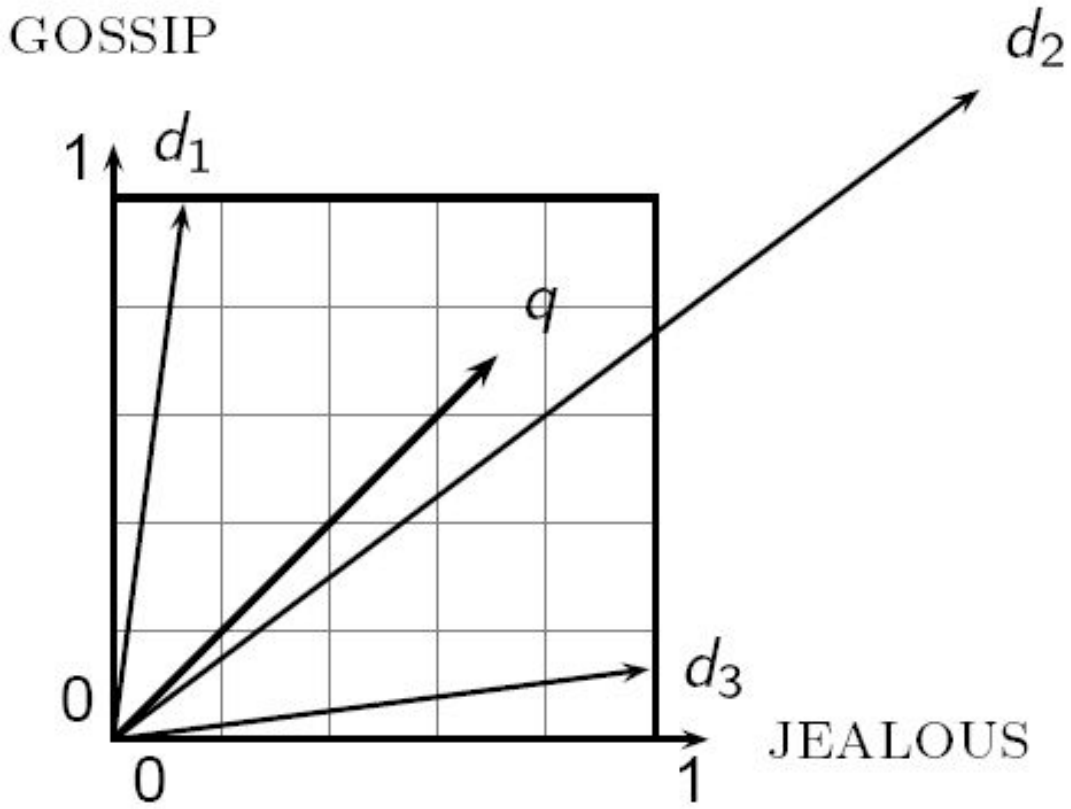
- **Key idea 2**: Rank documents according to their proximity to the query in this space
  - proximity = similarity of vectors

Quiz: why is Euclidean distance a bad estimator for proximity?

- . . . because Euclidean distance is large for vectors of different lengths.

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.

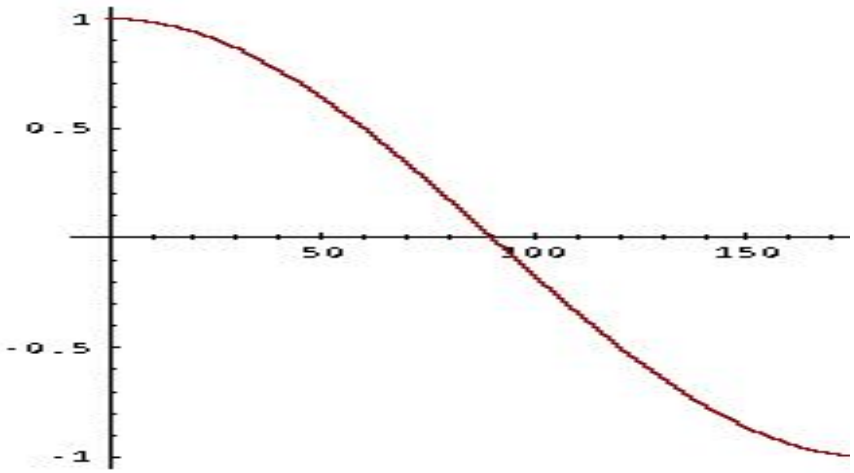


Use angle instead of distance

- Thought experiment: take a document *d* and append it to itself. Call this document *d*".
  - "Semantically" d and d′ have the same content
- The Euclidean distance between the two documents can be quite large
  - The angle between the two documents is 0, corresponding to maximal similarity.
- **Key idea**: Rank documents according to angle with query

From angles to cosines

- The following two notions are equivalent.
  - Rank documents in <u>decreasing</u> order of the angle between query and document
  - Rank documents in <u>increasing</u> order of cosine(query,document)
- Cosine is a monotonically decreasing function for the interval [0°, 180°]



Length normalization

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

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

- Dividing a vector by its L_2 norm makes it a unit (length) vector
  - on surface of unit hypersphere

Length normalization (cont'd)

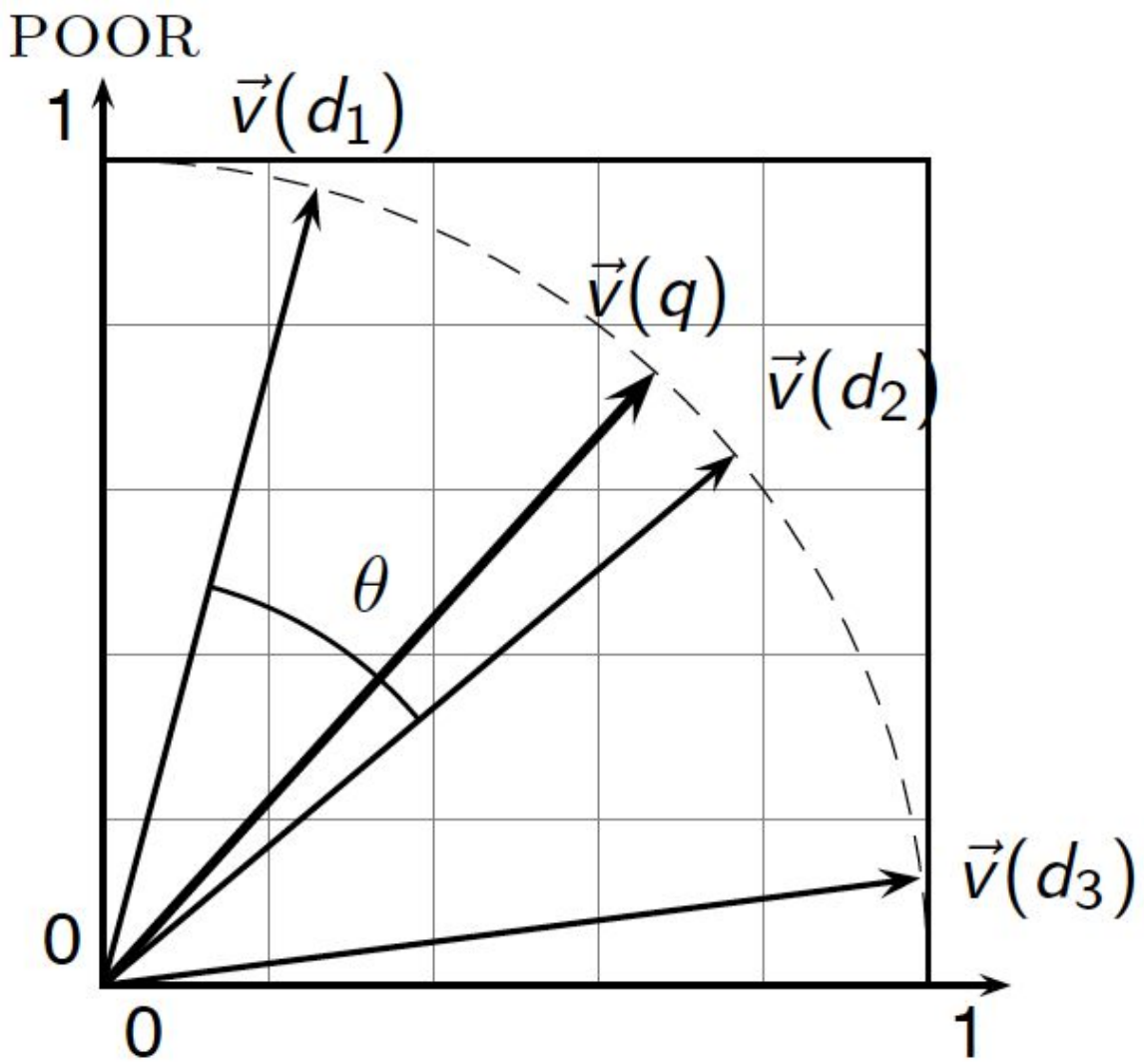
- Effect on the two documents d and d′ (d appended to itself) from earlier slide:
  - they have identical vectors after length-normalization
- Long and short documents now have comparable weights

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
- This the cosine similarlity of q and d

Cosine similarity illustrated



TF.IDF WEIGHTING HAS MANY VARIANTS

Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1 + \log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$
a (augmented)	$0.5 + \frac{0.5 \times \mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$
b (boolean)	$\begin{cases} 1 & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$			b (byte size)	$1/CharLength^{\alpha}$, $\alpha < 1$
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t \in d}(\mathrm{tf}_{t,d}))}$				

Summary – vector space model and ranking

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

Slide credit from here: M. de Rijke, UvA

An alternative: Okapi/BM25

- Not the full theory
- From idf weighting to BM25 in a few steps
- Simplest score for doc *d* is idf weighting of the query terms present:

$$RSV_d = \sum_{t \in q} \log \frac{N}{df_t}$$

  - (RSV: retrieval status value, the "score")

- Next step, factoring in frequency of each term and document length:

Okapi/BM25

$$RSV_d = \sum_{t \in q} \log \left[\frac{N}{\mathrm{df}_t} \right] \cdot \frac{(k_1 + 1)\mathrm{tf}_{td}}{k_1((1-b) + b \times (L_d / L_{ave})) + \mathrm{tf}_{td}}$$

- Here, tf_{td} is the tf of t in d and L_d and L_{ave} are the length of d and the average doc length for the whole collection
- Parameters
  - k_1 is a positive tuning parameter that calibrates the doc term frequency scaling
  - b calibrates the doc length scaling

Okapi/BM25

$$RSV_d = \sum_{t \in q} \left[\log \frac{N}{\mathrm{df}_t} \right] \cdot \frac{(k_1 + 1)\mathrm{tf}_{td}}{k_1((1-b) + b \times (L_d / L_{ave})) + \mathrm{tf}_{td}} \cdot \frac{(k_3 + 1)\mathrm{tf}_{tq}}{k_3 + \mathrm{tf}_{tq}}$$

- If the query is long use weighting for query terms
  - tf_{tq}: frequency of t in q
  - k_3: tuning frequency scaling of query

Quiz: three key notions?

- Many "document retrieval models" share three key ingredients:

- **Term frequency**
  - A term is a better indicator of the contents of a document if it occurs more frequently in that document

- **Document frequency**
  - Terms that occur in many documents are not good discriminators

- **Document length normalization**
  - Longer documents are more likely to contain a term (without necessarily being "about" the term)

Query independent ranking

- PageRank (named after Larry Page)
  - http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf
- Citation analysis: analysis of citations in the scientific literature
  - Example citation: “Miller (2001) has shown that physical activity alters the metabolism of estrogens.”
  - “Miller (2001)” is a hyperlink linking two scientific articles.

Query independent ranking

- One application of these “hyperlinks” in the scientific literature:
  - Measure the similarity of two articles by the overlap of other articles citing them
  - This is called cocitation similarity

- **Key idea:** Cocitation similarity on the web?

Link-based ranking for web search

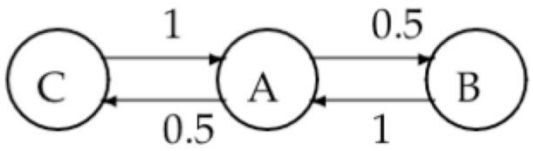
- Simple version of using links for ranking on the web
  - First retrieve all pages satisfying the query (say *venture capital*)
  - Order these by the number of in-links
- Simple link popularity (= number of in-links) is easy to spam. Why?

PageRank basis: 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 = steady state probability = long-term visit rate
- Concept of long-term visit rate clear?

Markov chains

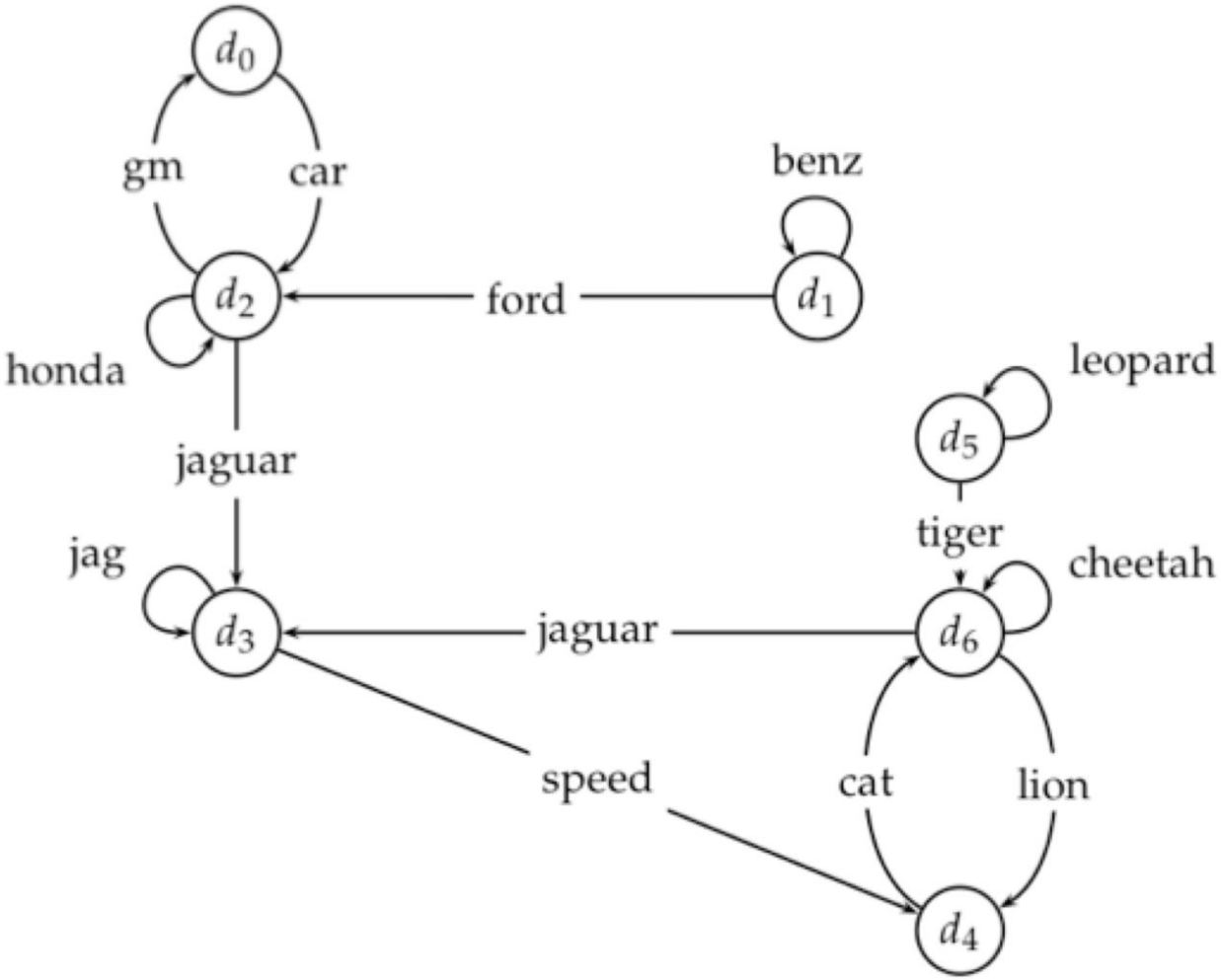
- A Markov chain consists of N states, plus an $N \times N$ transition probability matrix P.
- state = page
- At each step, we are on exactly one of the pages.
- For $1 \leq i, j \leq N$, the matrix entry P_{ij} tells us the probability of j being the next page, given we are currently on page i.



Markov chains

- Clearly, for all i, $\sum_{j=1}^{N} \boldsymbol{P}_{ij} = \mathbf{1}$
- Markov chains are abstractions of random walks.

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 matric for example

	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

- Recall: 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.
- Next: what properties must hold of the web graph for the long-term visit rate to be well defined?
- 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 (or non-sensical).

Quiz: How to cope with dead ends?

Teleporting

- At a dead end, jump to a random web page
- At any non-dead end, with probability 10%, jump to a random web page
- With remaining probability (90%), go out on a random hyperlink (e.g., randomly choose with probability (1-0.1)/4=0.225 one of the four hyperlinks of the page)
- 10% is a parameter.

Result of teleporting

- With teleporting, we cannot get stuck in a dead end.
- Concept of teleporting clear?
- 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.

Ergodic Markov chains

- 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.

Formalization of "visit"

- A probability (row) vector $\vec{x} = (x_1, \ldots, x_N)$ tells us where the random walk is at any point.
- Example:

(	0	0	0	…	1	…	0	0	0	)
	1	2	3	…	***i***	…	N-2	N-1	N	

- More generally: the random walk is on page i with probability x_i.
- Example:

(	0.05	0.01	0.0	…	0.2	…	0.01	0.05	0.03	)
	1	2	3	…	***i***	…	N-2	N-1	N	

- $\sum x_i = 1$

Change in probability vector

- If the probability vector is $\vec{x} = (x_1, \ldots, x_N)$ at this step, what is it at the next step?
- Recall that row i of the transition probability matrix P tells us where we go next from state i.
- Equivalently: column j of P tells us "where we came from" (and with which probability).
- So from $\vec{x}$, our next state is distributed as $\vec{x}P$.

Steady state in vector notation

- The steady state in vector notation is simply a vector $\vec{\pi} = (\pi_1, \pi_2, \ldots, \pi_N)$ of probabilities.
- (We use $\vec{\pi}$ to distinguish it from the generic notation $\vec{x}$ for a probability vector.)
- π_i is the long-term visit rate (or PageRank) of page i.
- So we can think of PageRank as a very long vector – one entry per page.

How do we compute the steady state vector?

- In other words: how do we compute PageRank?
- Recall: $\vec{\pi} = (\pi_1, \pi_2, \ldots, \pi_N)$ is the PageRank vector, the vector of steady-state probabilities ...
- ...and if the distribution in this step is described by $\vec{x}$, then the distribution in the next step is distributed as $\vec{x}P$.
- But $\vec{\pi}$ is the steady state! So: $\vec{\pi} = \vec{\pi}P$
- Solving this matrix equation gives us $\vec{\pi}$.
- $\vec{\pi}$ is the principal left eigenvector for P ...
- ...that is, $\vec{\pi}$ is the left eigenvector with the largest eigenvalue.
- Transition probability matrices always have largest eigenvalue 1.

One way of computing the PageRank

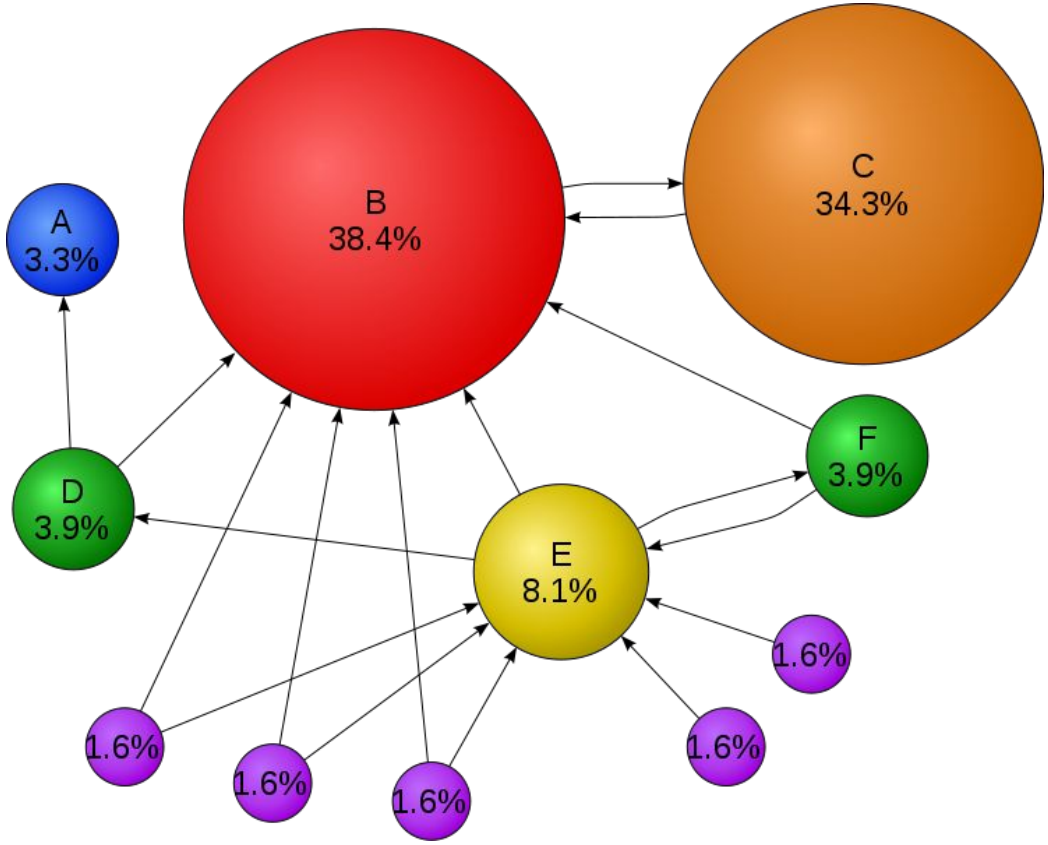
- Recall: regardless of where we start (except for pathological cases), we eventually reach the steady state $\vec{\pi}$.
- Start with (almost) any distribution $\vec{x}$, e.g., uniform distribution
- After one step, we're at $\vec{x}P$.
- After two steps, we're at $\vec{x}P^2$.
- After k steps, we're at $\vec{x}P^k$.
- Algorithm: multiply $\vec{x}$ by increasing powers of P until the product looks stable.
- This is called the power method.

Power method: Example

- Two-node example: $\vec{x} = (0.5, 0.5), P = \begin{pmatrix} 0.25 & 0.75 \\ 0.25 & 0.75 \end{pmatrix}$
- $\vec{x}P = (0.25, 0.75)$
- $\vec{x}P^2 = (0.25, 0.75)$
- Convergence in one iteration!

Quiz: page rank

- Why is website C's page rank higher than E's?



PageRank summary

- Preprocessing
  - Given graph of links, build matrix $\boldsymbol{P}$
  - Apply teleportation
  - From modified matrix, compute $\vec{\pi}$
  - $\vec{\pi}_i$ is the PageRank of page $\boldsymbol{i}$.
- Query processing
  - Retrieve pages satisfying the query
  - Rank them by their PageRank
  - Return reranked list to the user

PageRank issues

- Real surfers are not random surfers – Markov model is not a good model of surfing.
  - Issues: back button, short vs. long paths, bookmarks, directories – and search!
- Simple PageRank ranking (as described on previous slide) produces bad results for many pages.
  - Consider the query *video service*
  - The Yahoo home page (i) has a very high PageRank and (ii) contains both words.
  - If we rank all Boolean hits according to PageRank, then the Yahoo home page would be top-ranked.
  - Clearly not desirable
- In practice: rank according to weighted combination of many factors, including raw text match, anchor text match, PageRank and many other factors

How important is PageRank?

- Frequent claim: PageRank is the most important component of web ranking.
- The reality:
  - There are several components that are at least as important: e.g., anchor text, phrases, proximity, tiered indexes ...
  - Rumor has it that PageRank in its original form (as presented here) has a negligible impact on ranking!
  - However, variants of a page's PageRank are still an essential part of ranking.
  - Adressing link spam is difficult and crucial.

Next week

- We will start from simple visual representations of an image

Acknowledgements

- Information Retrieval slide credit:
  - C. Manning and P. Raghavan, Stanford University
  - M. de Rijke, University of Amsterdam