Text, XML, and Multimedia Information Retrieval

Arjen P. de Vries

arjen@acm.org

EDBT Summer School 2007

Centrum voor Wiskunde en Informatica

Outline

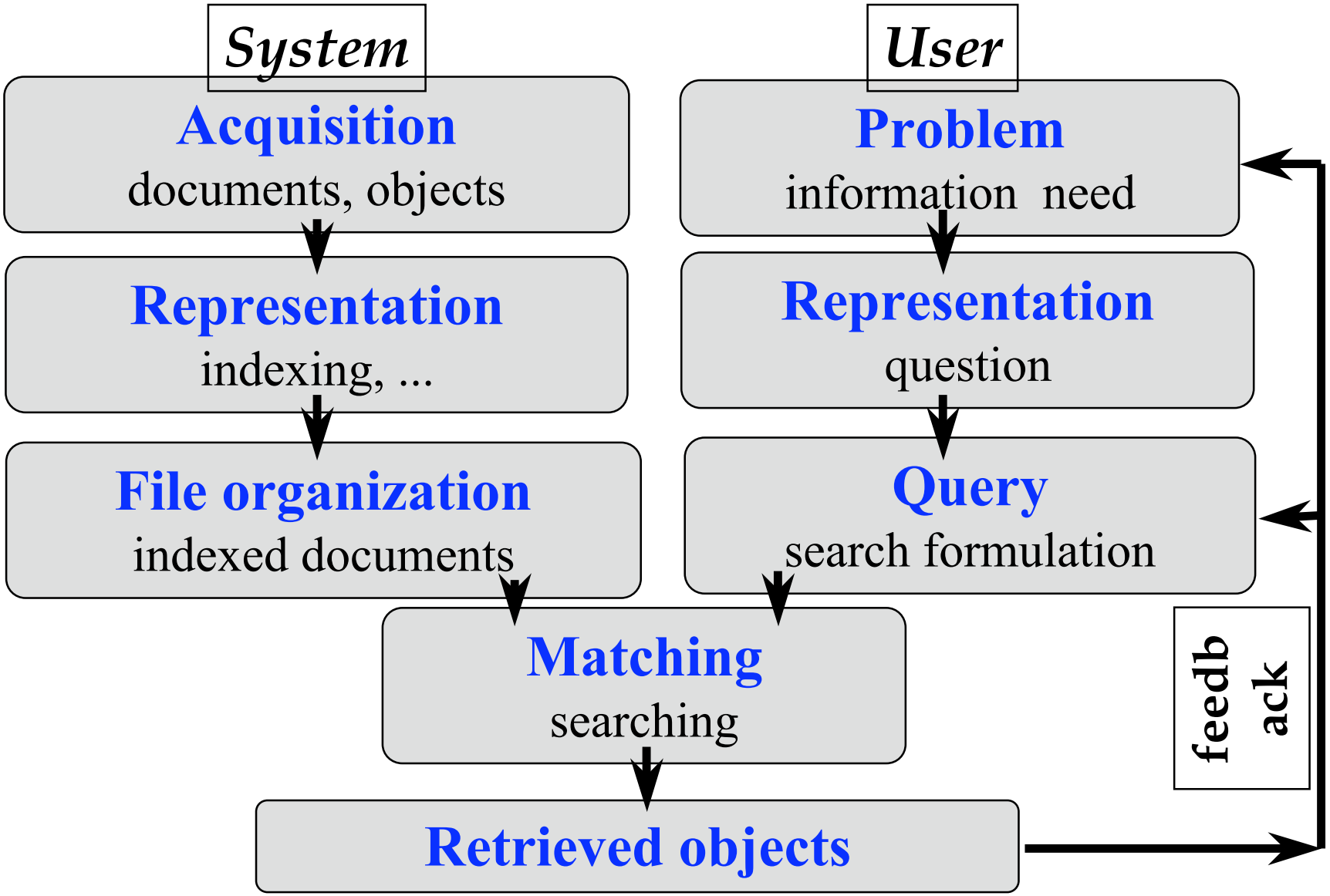
- Information Retrieval (IR)
- The Language Modeling Approach
- Text Retrieval
- XML Retrieval
- Multimedia Retrieval
- Integration of IR and DB

Information Retrieval

IR is about satisfying vague information needs provided by users, (imprecisely specified in ambiguous natural language) by satisfying them approximately against information provided by authors (specified in the same ambiguous natural language)

(Alan Smeaton)

Information Retrieval



Data vs. Information Retrieval
(Keith van Rijsbergen)

- Data Retrieval
  - Deductive
  - Attributes are necessary and sufficient for class membership

- Information Retrieval
  - Inductive
  - No attribute is necessary nor sufficient to judge relevance

Retrieval Model

- Document representation
  - E.g., bag-of-words
- Query formulation
  - E.g., reformulation after relevance feedback
- Ranking function
  - E.g., vector space model, $tf \cdot idf$

Evaluation

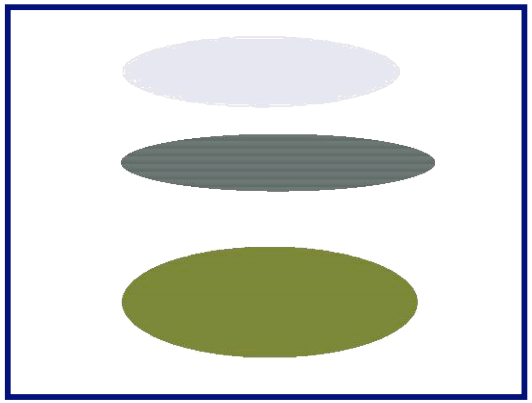
- As opposed to DB research, measuring success in IR can only be done experimentally, not analytically!
- 'Laboratory experiments' are done with Test Collections (e.g., TREC)
  - Triples of topic, document collection and relevance assessments

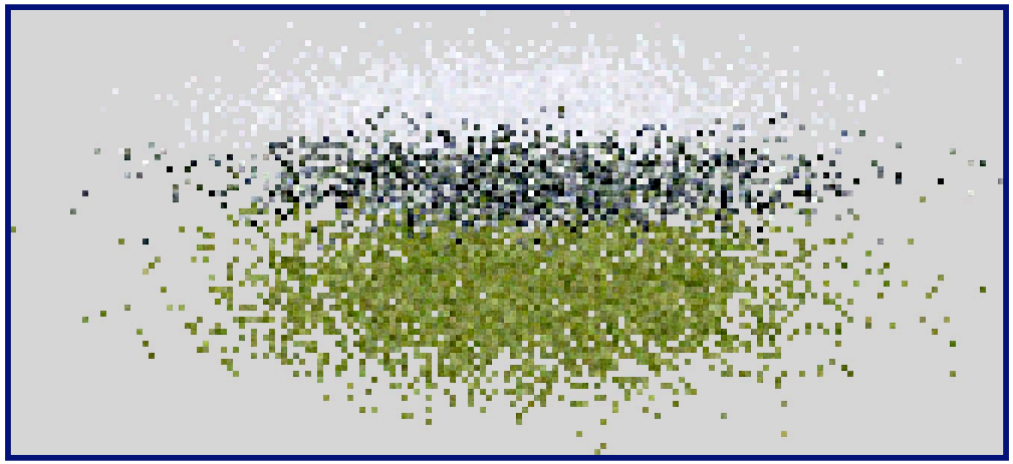
Measures

- Precision
  - fraction of retrieved documents that is relevant
- Recall
  - fraction of relevant documents that is retrieved
- Average Precision
  - precision averaged over different levels of recall
- Mean Average Precision (MAP)
  - mean of average precision over all queries

The Language Modeling Approach to IR

Generative Models





video of Bayesian model to that present the disclosure can a on for retrieval in have is probabilistic still of for of using this In that is to only queries queries visual combines visual information look search video the retrieval based search. Both get decision (a visual generic results (a difficult We visual we still needs, search. talk what that to do this for with retrieval still specific retrieval information a as model still

Generative Models…

- A statistical model for generating data
  - Probability distribution over samples in a

aka

‘Language Modelling’

P (● | M, ●)

P (● | M, ● ○)

P (● | M, ● ○ ●)

… in Information Retrieval

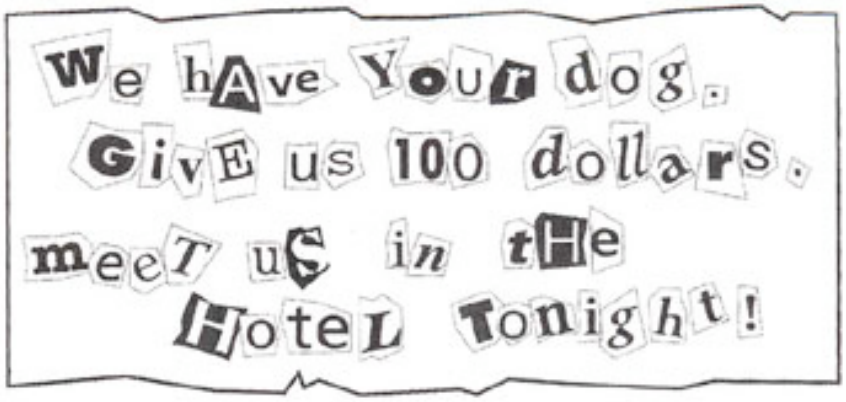
- Basic question:
  - What is the likelihood that this document is relevant to this query?
- P(rel|I,Q) = P(I,Q|rel)P(rel) / P(I,Q)

- P(I,Q|rel) = P(Q|I,rel)P(I|rel)

‘Language Modelling’

- Not just ‘English’
- But also, the *language* of
  - author
  - newspaper
  - text document
  - image

- Guardian or Times?



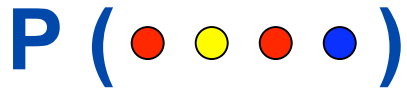
‘Language Modelling’

- Not just English!
- But also, the *language* of
  - author
  - newspaper
  - text document
  - image

- 

or  ?

Unigram and higher-order models



- Unigram Models

- N-gram Models

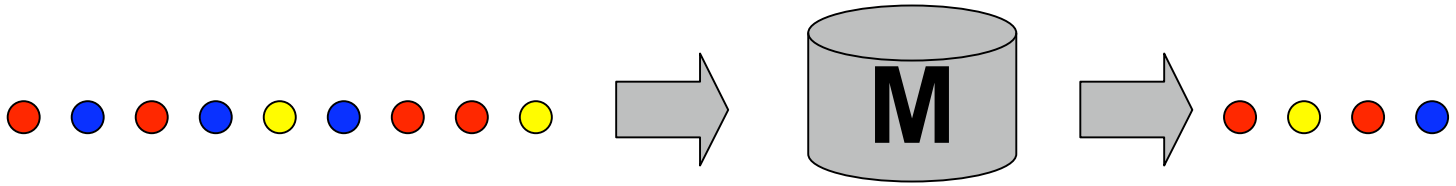
- Other Models
  - Grammar-based models, etc.
  - Mixture models

The fundamental problem

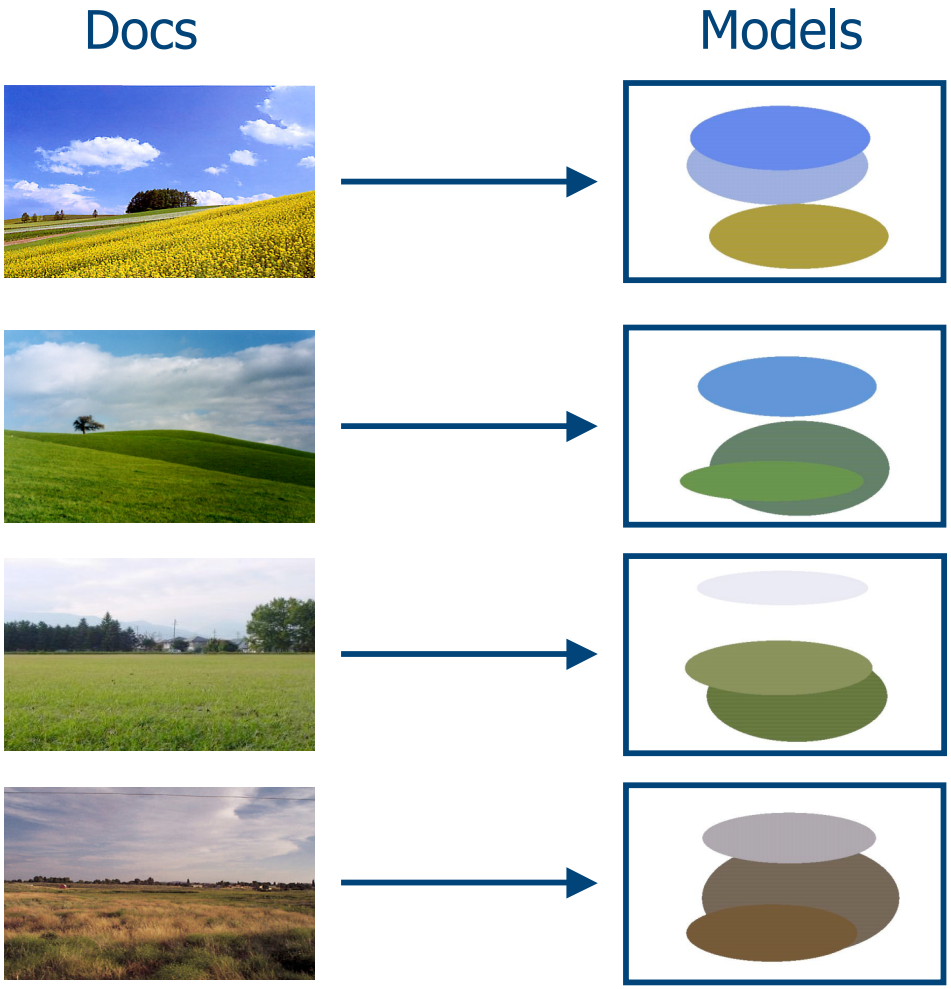
- Usually we don't know the model **M**
  - But have a sample representative of that model

P (● ● ● ● | M (● ● ● ● ● ● ● ● ●))

- First estimate a model from a sample
- Then compute the observation probability



Indexing: determine models



- Indexing
  - Estimate Gaussian Mixture Models from images using EM
  - Based on feature vector with colour, texture and position information from pixel blocks
  - Fixed number of components

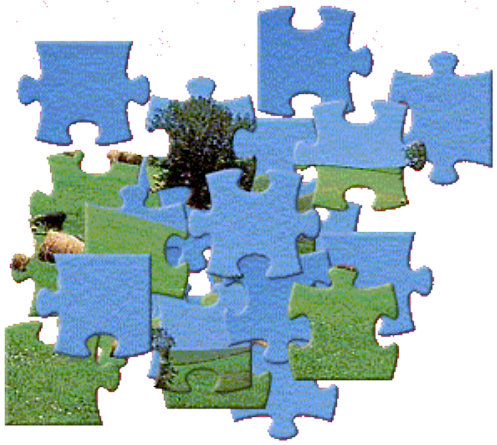
Retrieval: use query likelihood

- Query:

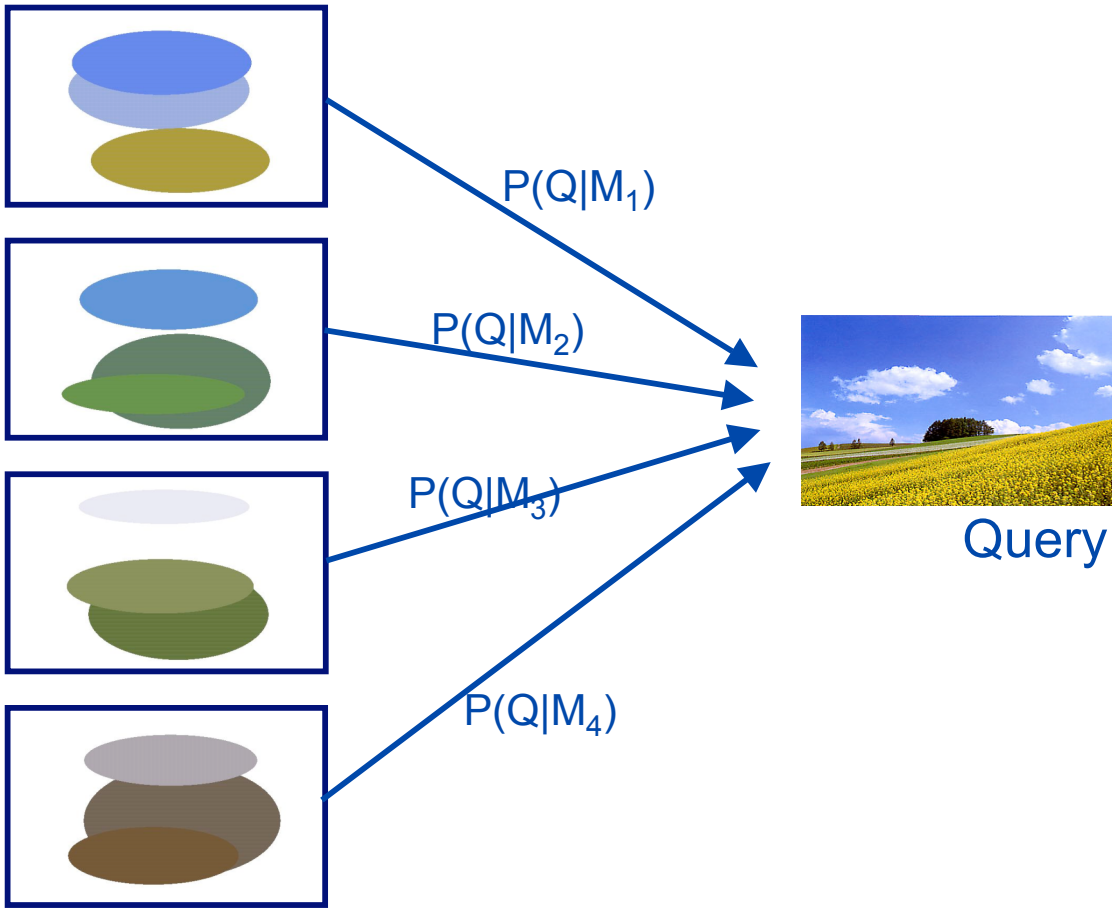
- Which of the models is most likely to generate these 24 samples?

Probabilistic Image Retrieval

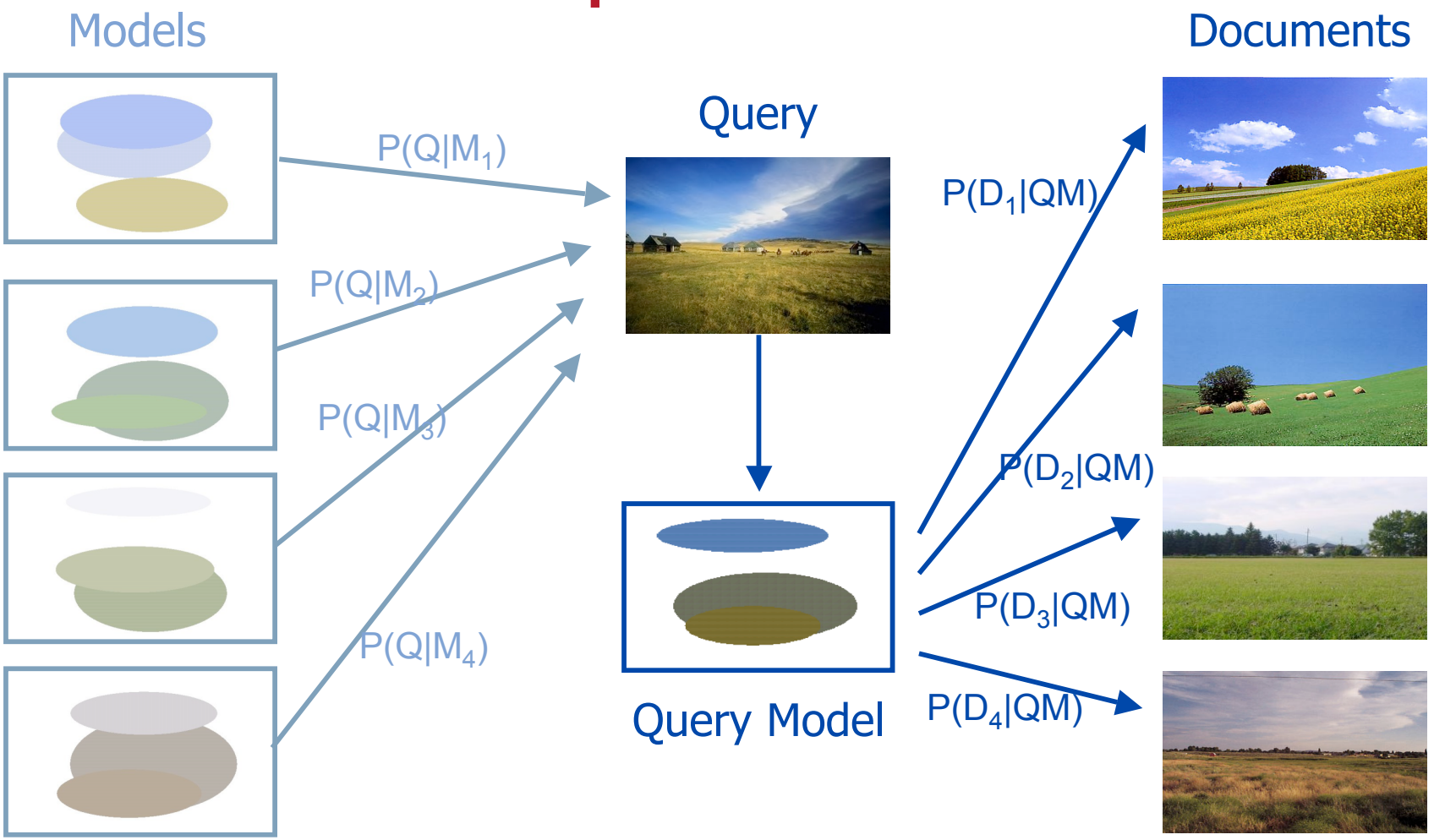
?



Rank by P(Q|M)



Topic Models



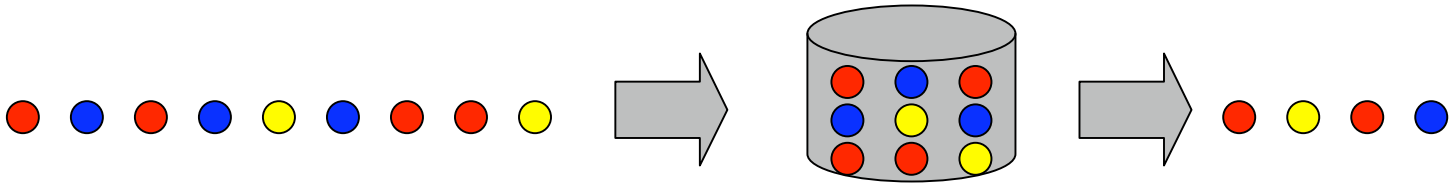
Probabilistic Retrieval Model

- Text
  - Rank using probability of drawing query terms from document models
- Images
  - Rank using probability of drawing query blocks from document models
- Multi-modal
  - Rank using joint probability of drawing query *samples* from document models

Text Retrieval

Text Models

- Unigram Language Models (LM)
  - Urn metaphor

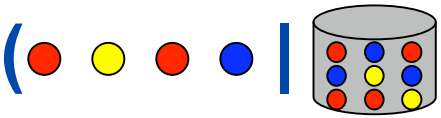
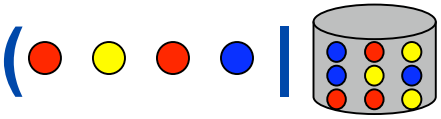
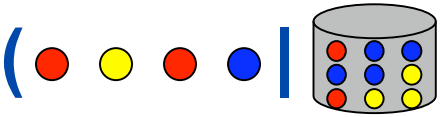
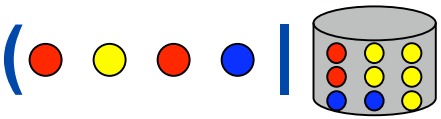


- **P(● ○ ● ●) ~ P (●) P (○) P (●) P (●)**

  = 4/9 * 2/9 * 4/9 * 3/9

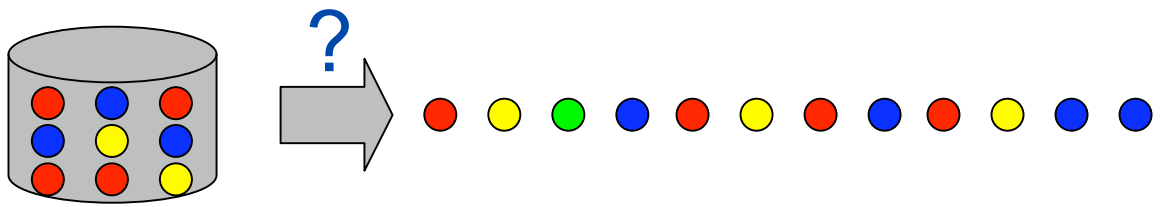
Generative Models and IR

- Rank models (documents) by probability of generating the query
- Q: 

- **P(**  **)** = 4/9 * 2/9 * 4/9 * 3/9 = $96/9^4$
- **P(**  **)** = 3/9 * 3/9 * 3/9 * 3/9 = $81/9^4$
- **P(**  **)** = 2/9 * 3/9 * 2/9 * 4/9 = $48/9^4$
- **P(**  **)** = 2/9 * 5/9 * 2/9 * 2/9 = $40/9^4$

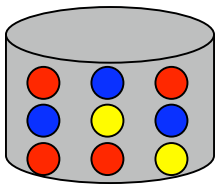
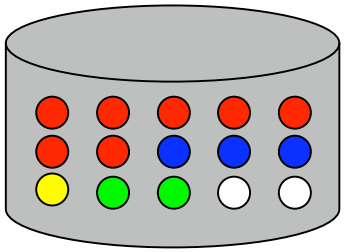
The Zero-frequency Problem

- Suppose some event not in our example – very common when sampling from natural language data
- Inferring zero probabilities is not correct,
  - Especially when dealing with incomplete samples



Smoothing

- Idea: shift part of probability mass to unseen events
- Interpolation with background ('general English')
  - Reflects expected frequency of events
  - Behaves like inverse document frequency

– λ  +(1-λ) 

XML Retrieval

Document-Centric XML-IR

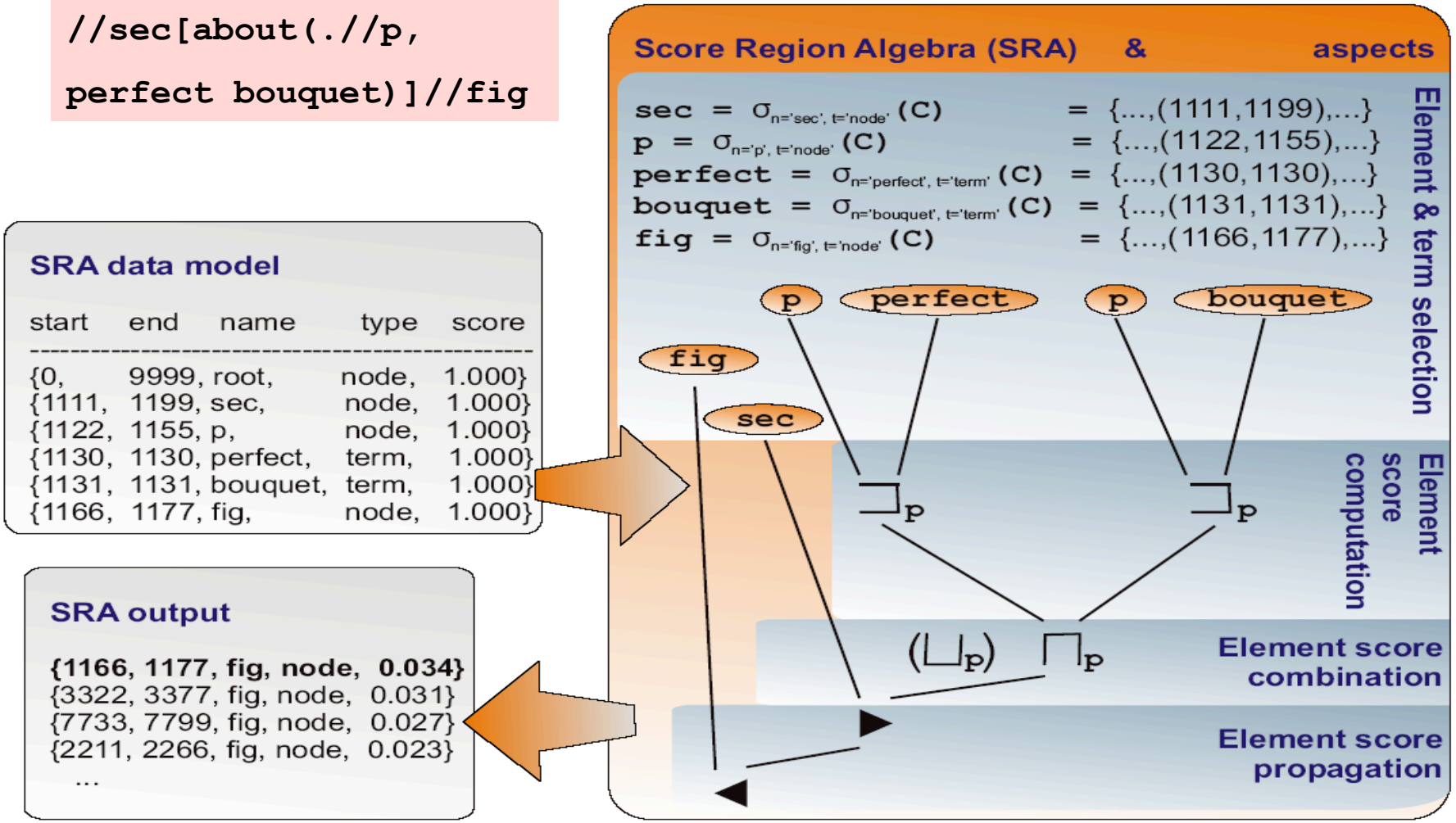
- Content-only (CO) queries
  - Standard IR queries retrieving document components instead of documents
    - ‘Find me elements about…’
- Content-and-structure (CAS) queries
  - Query expresses constraints on which types of components are to be retrieved
  - E.g., NEXI queries: ‘XPath + about’
    - `//article[about(.//sec, “databases information retrieval")]`

Common Approach at INEX

- About clauses are processed independently, e.g., using the language modeling approach to IR for text as discussed before
- In CAS queries, the XPath constraints are applied to the ranked elements – as (usually weighted) filters

Example: TIJAH

```
//sec[about(.//p,
perfect bouquet)]//fig
```



XML-IR Retrieval Models

- Really a ‘combination of evidence’ problem, where the bits and pieces of evidence come from different parts of the XML tree
- Still not well understood

XML-IR Techniques

- Length priors
  - Reward smaller elements for their relevancy
- Article weighting
  - Reward elements in relevant articles
- Score propagation
  - More general propagation of scores of related elements than ‘just’ the article context

IR and DB

Still on the Research Agenda...

- Integration of data and information retrieval
  - DB: Data model, query language
  - IR: Retrieval model
- Follow the database approach to the management of data *and of content*
  - Reduce implementation effort for search applications!

Candidate Architectures

- IR retrieval model expressed as queries on top of (relational) database system
- IR supported via user-defined functions
- Middleware layer on top of DB & IR systems

Experimental Setup

- GOV2 Test Collection
  - TREC Terabyte Track
  - 25M web pages (.gov domain, 426GB)
  - 50,000 queries (50 with relevance assessments)
- Platform
  - A ~USD 4000 Linux PC
  - 3GHz P4, 4GB RAM, 1TB 12-disk RAID

IR on a DBMS in four steps

- Parsing
  - Extract term occurrences within collection
- Index Creation
  - Map IR data-structures to relational tables
- Document Ranking
  - Map keyword search to relational queries
- IR-specific Optimizations

Parsing

- Remove markup and stop-words, and apply standard Porter stemming
- Represent resulting tokens as 64-bit integers
- Result:
  - DT [docid, term] (12.4 Gtuples, ~140GB)
    - Contains entry for each term occurrence!
  - D [docid, name, length] (25 Mtuples)

Index Creation

- Create sorted index (*inverted file*):
  - TD [term, docid, tf] (3.5 Gtuples, ~56 GB)

```
# TD computation using DT
Aggr(
  Sort(
     Scan(DT, [docid, term]),
     [ term, docid ] )
  [ term, docid ],
  [ tf = count() ] )
```

Example

term	docid	tf
…	…	…
information	1	3
information	2	10
…	…	…
retrieval	1	7
storage	3	2
storage	4	1

Example: Boolean Retrieval

- “information AND (storage OR retrieval)”

```
Join(
  ScanSelect(TD1=TD, TD1.term='information'),
  OuterJoin(
     ScanSelect(TD2=TD, TD2.term='storage'),
     ScanSelect(TD3=TD, TD3.term='retrieval'),
     TD2.docid = TD3.docid)
  ),
  TD1.docid = OuterJoinResult.docid
)
```

Example: Probabilistic IR

- E.g., Okapi BM25
  - frequency of term within document (tf)
  - #documents containing term (df)
  - length of document (doclen)

$$S_{BM25}^{(D)} = \sum_{i=1}^{|Q|} \omega_{D,T_i} \tag{1}$$

$$\omega_{D,T} = \log\left(\frac{f_D}{f_{T,D}}\right) \cdot \frac{(k_1+1)\cdot f_{D,T}}{f_{D,T} + k_1 \cdot ((1-b) + b \cdot \frac{|D|}{avgdl})} \tag{2}$$

Additional Tables

- Document table:
  `D[docid, docname, doclen]`
  - We already had this from parsing
- Term table (vocabulary):
  `T[term, df]`
  - For each unique term, store the number of documents containing that term (df)

BM25 Query

```
TopN(
  Project(
    Join(
      OuterJoin(
        ScanSelect( TD1=TD, TD1.term=t1_term ),
        ScanSelect( TD2=TD, TD2.term=t2_term )
        TD1.docid=TD2.docid),
      Scan( D ),
      D.docid=(TD1.docid OR TD2.docid))
    [D.docname,score=BM25(TD1.tf,D.doclen,t1_df)
     +BM25(TD2.tf,D.doclen,t2_df) ]
  ),
  [ score DESC ],
  20
)
```

Reality Check

- Search engine companies as well as TREC Terabyte Track participants use customized IR systems
  - Highly optimized *Inverted indices* for document retrieval
- Traditional DBMSs are not suited for their data management requirements!
  - Inefficient and/or resource consuming

X100 DBMS Architecture

- Our prototype system (called X100) deviates from traditional database architecture in two ways:
  - **Vectorized processing**
  - **Light-weight compression**
- Additional factors
  - Algebra-level interface
  - Extremely cache-conscious
  - Scripting, for iterative algorithms

Volcano Iterator Model

- Each relational operator has its own class, with open(), next(), and close() interface
- This results in tuple-at-a-time processing

➔*DBMS bad at filling pipelines*

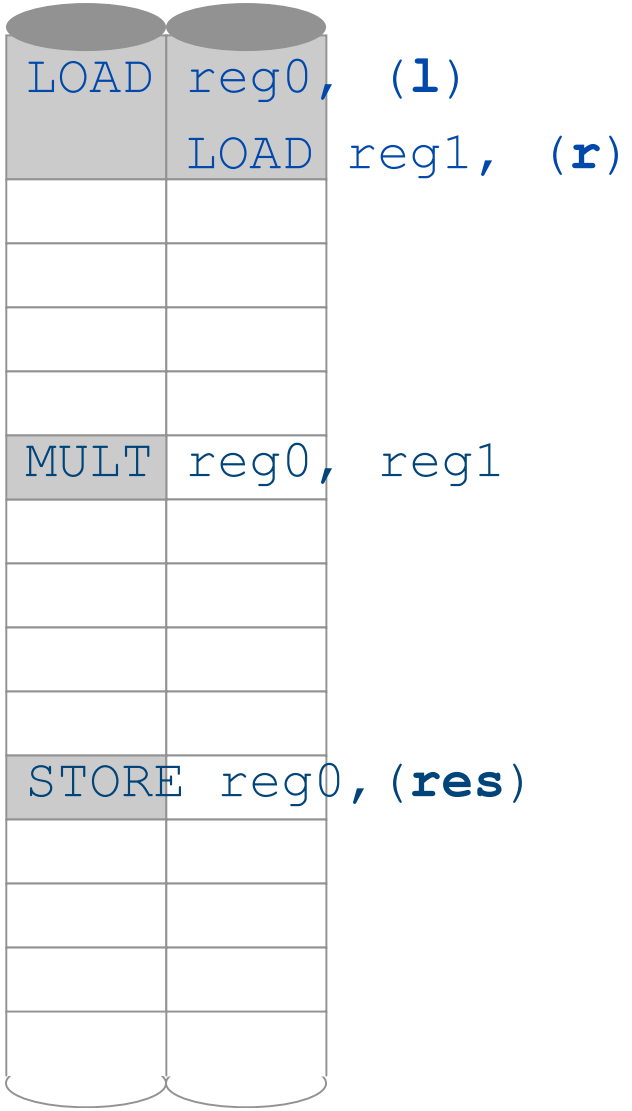
Tuple-at-a-time Primitives

*(int,int): int

```
void
mult_int_val_int_val(
  int *res, int l, int r)
{
  *res = l * r;
}
```

15 cycles-per-tuple
+ function call cost (~20cycles)

Total: ~35 cycles per tuple



X100: Vectorized Processing

- Instead of single tuples, entire vectors (1-dimensional arrays) are passed through the Volcano pipeline
- Each operator boils down to a sequential loop over aligned input arrays, which will benefit from *loop pipelining*

X100: Vectorized Primitives

*(int,int): int ➔ *(int[],int[]) : int[]

Pipelinable loop

```
void
map_mult_int_col_int_col(
            int _restrict_*res,
            int _restrict_*l,
            int _restrict_*r,
            int n)
{
  for(int i=0; i<n; i++)
      res[i] = l[i] * r[i];
}
```

X100: Vectorized Primitives

Pipelined loop, by C compiler

```
LOAD reg0, (l+0)
LOAD reg1, (r+0)
 LOAD reg2, (l+1)
 LOAD reg3, (r+1)
  LOAD reg4, (l+2)
  LOAD reg5, (r+2)
MULT reg0, reg1
 MULT reg2, reg3
  MULT reg4, reg5
STORE reg0, (res+0)
 STORE reg2, (res+1)
  STORE reg4, (res+2)
```

```
void
map_mult_int_col_int_col(
            int _restrict_*res,
            int _restrict_*l,
            int _restrict_*r,
            int n)
{
  for(int i=0; i<n; i++)
      res[i] = l[i] * r[i];
}
```

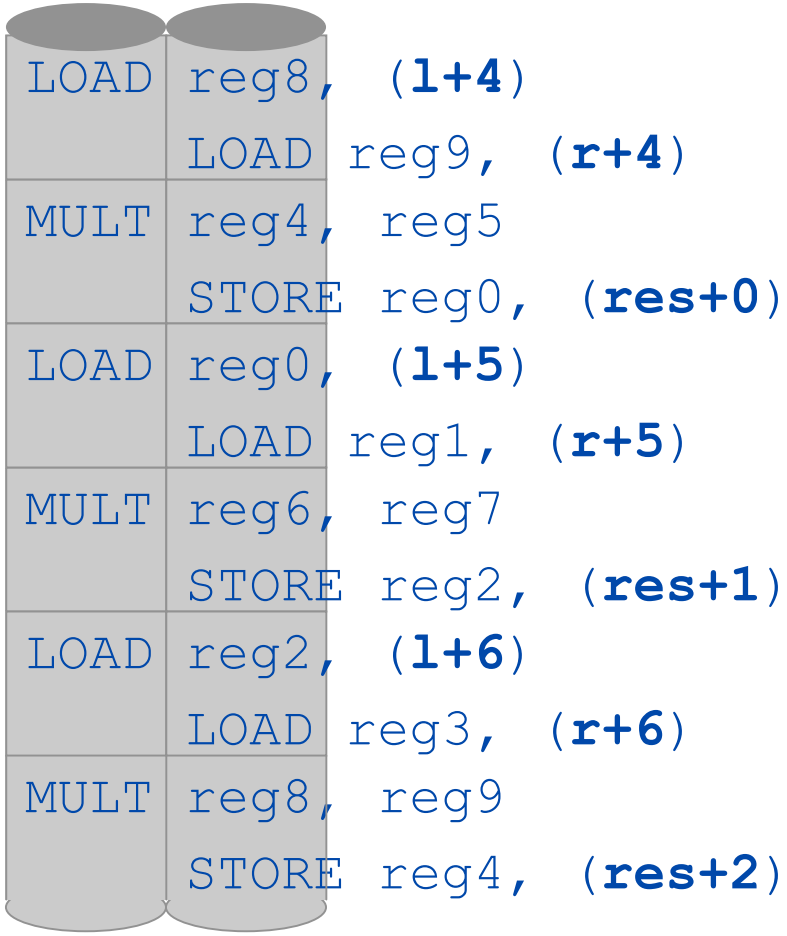
X100: Vectorized Primitives

Estimated throughput

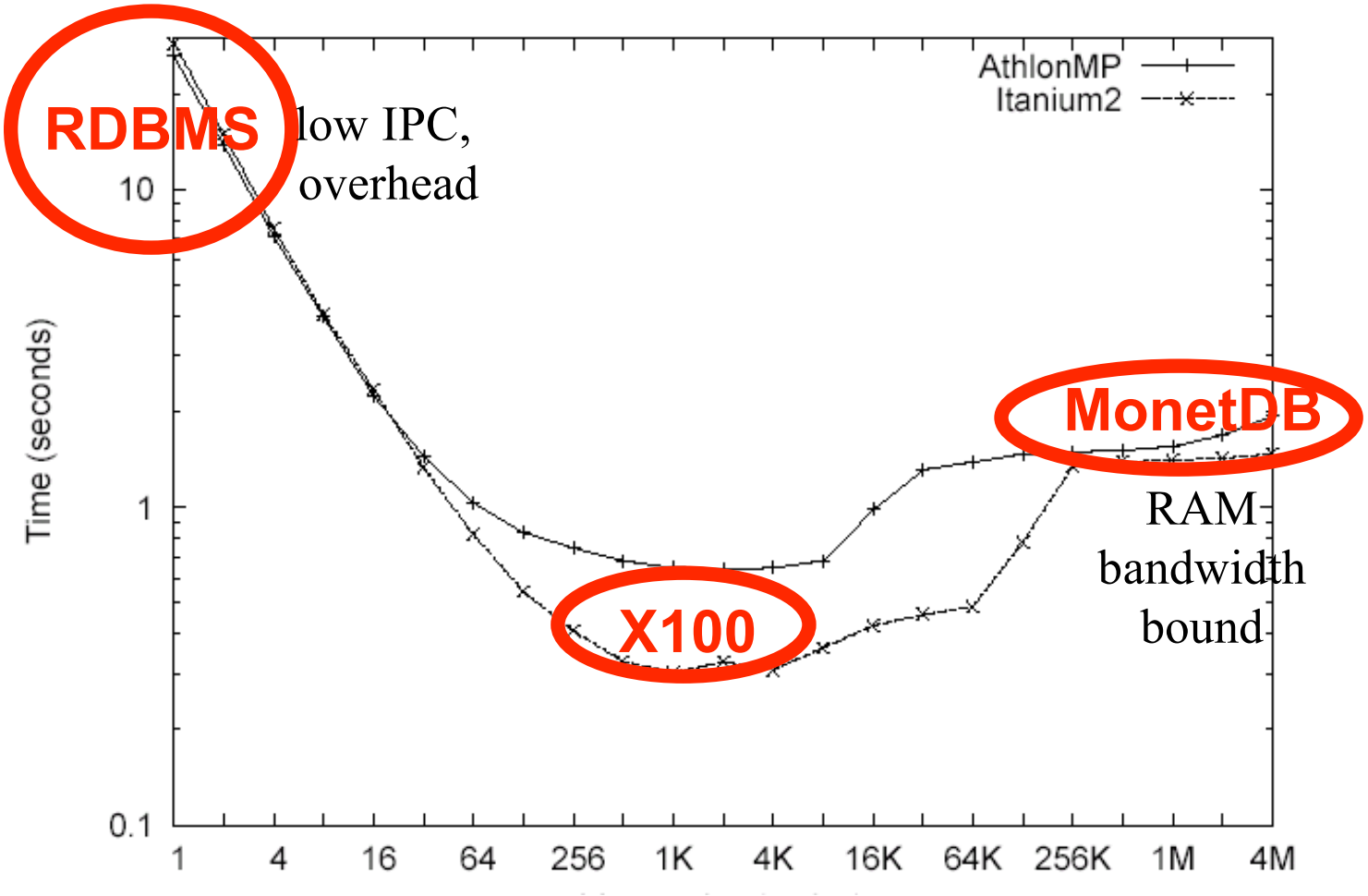
2 cycles per tuple

1 function call (~20 cycles)
per vector (i.e. 20/100)

Total: 2.2 cycles per tuple



Varying vector size (TPC-H Q1)



Results: Boolean Retrieval

Table 3: XXX Experiments On TREC-TB

Run name (+ added feature)	p@20	Avg.query time (ms), cold data	Avg.query time (ms), hot data
BoolAND	0.0130	76	12
BoolOR	0.0000	133	80

- Fast but highly ineffective…

Results: Probabilistic IR

Table 3: XXX Experiments On TREC-TB

Run name (+ added feature)	p@20	Avg.query time (ms), cold data	Avg.query time (ms), hot data
BoolAND	0.0130	76	12
BoolOR	0.0000	133	80
BM25	0.5460	440	342

- Effective, but relatively slow…
- `OuterJoin` generates large results, causing high processing overhead

IR Optimization: Two-Pass

- [Broder et al., 2003]:
  - Documents containing many of the query terms are likely to score high
- First pass use `Join` instead of `OuterJoin`
  - Conjunctive, so more restrictive
- Second pass using `OuterJoin`, only if first pass did not return enough results
  - Disjunctive, so many results
  - Needed for ~15 percent of TREC queries

Other (IR) Optimizations

- Term-document-score materialization and quantization
- Light-weight compression
- Dynamic index pruning [Turtle & Flood, 1995]
  - Query terms are processed *one at a time*, in order of increasing frequency (df)
  - Prune docs with

    score < topScores[r] – availScore

With All Optimizations

Table 3: XXX Experiments On TREC-TB

Run name (+ added feature)	p@20	Avg.query time (ms), cold data	Avg.query time (ms), hot data
BoolAND	0.0130	76	12
BoolOR	0.0000	133	80
BM25	0.5460	440	342
BM25T (+Two-pass)	0.5470	198	72
BM25TC (+Compression)	0.5470	158	73
BM25TCM (+Materialization)	0.5470	155	29
BM25TCMQ8 (+Quant.8-bit)	0.5490	118	28
BM25TCMQ8S (+max-Score)	0.5490	115	28

IR & DB conclusion

- Competitive IR performance can be achieved on a DBMS…
- *… but only if the DBMS takes modern hardware considerations into account*
- Also: IR-specific optimizations must be applied

Multimedia Retrieval

Indexing Multimedia

- Manually added descriptions
  - ‘Metadata’
- Analysis of associated data
  - Speech, captions, OCR, auto-cues, …
- Content-based retrieval
  - Approximate retrieval
  - Domain-specific techniques

Limitations of Metadata

- Vocabulary problem
  - Dark vs. somber
- Different people describe different aspects
  - Dark vs. evening

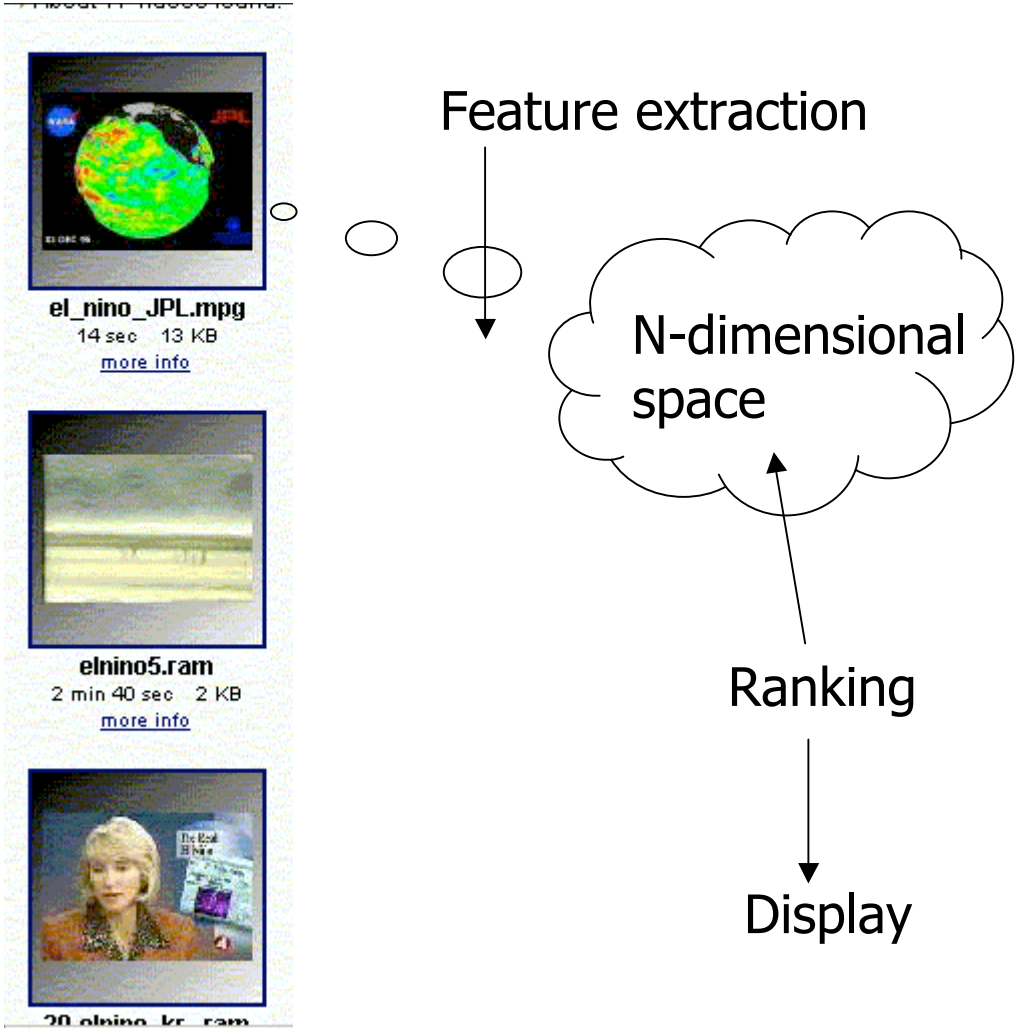
Limitations of Metadata

- Encoding Specificity Problem
  - A single person describes different aspects in different situations
- Many aspects of multimedia simply cannot be expressed unambiguously
  - Processes in left (analytic, verbal) vs. right brain (aesthetics, synthetic, nonverbal)

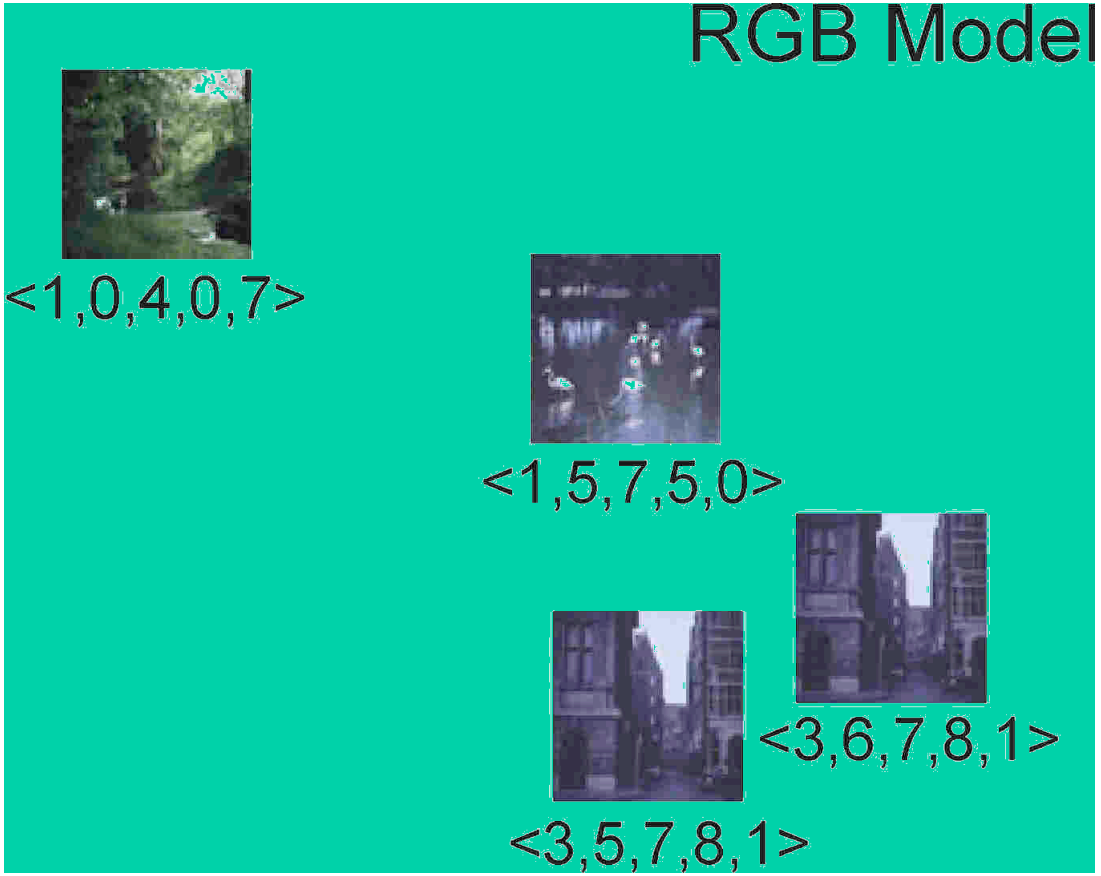
Approximate Retrieval

- Based on **similarity**
  - Find all objects that are similar to this one
  - Distance function
  - Representations capture some (syntactic) meaning of the object

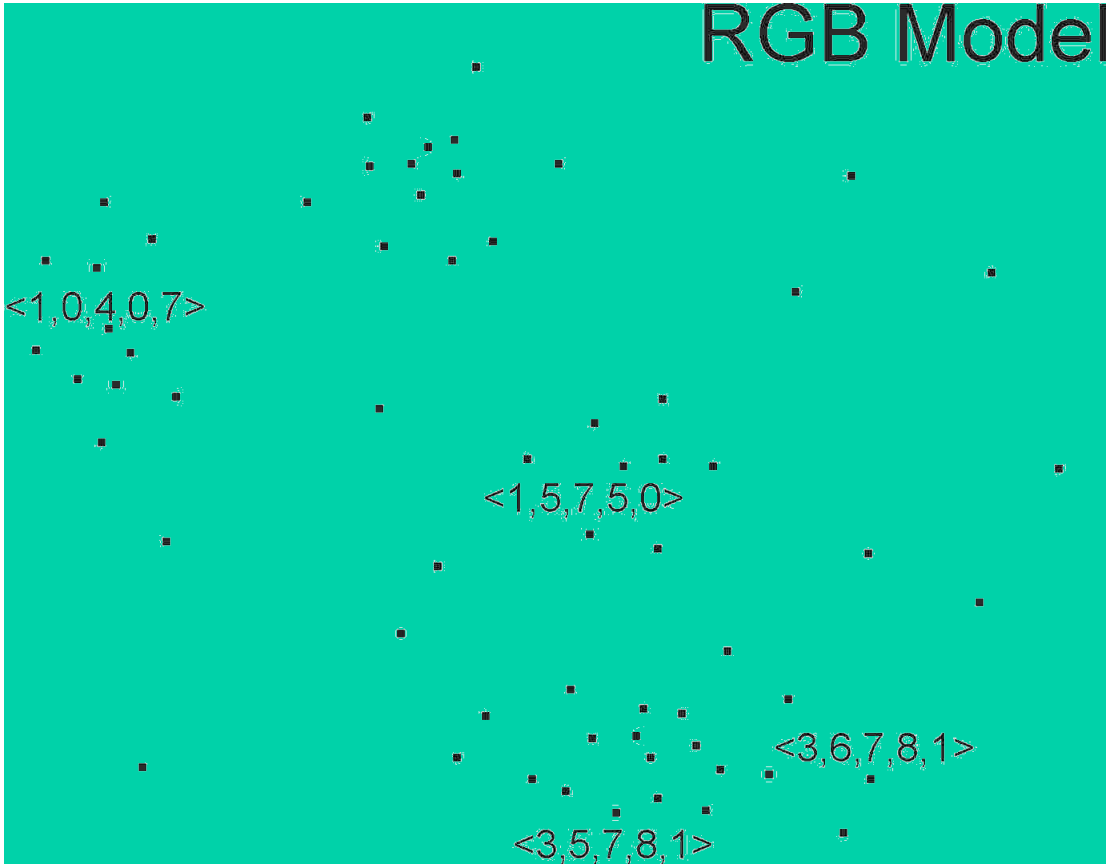
- 'Query by Example' paradigm


el_nino_JPL.mpg
14 sec 13 KB
more info
elnino5.ram
2 min 40 sec 2 KB
more info
Feature extraction
N-dimensional space
Ranking
Display

Low-level Features

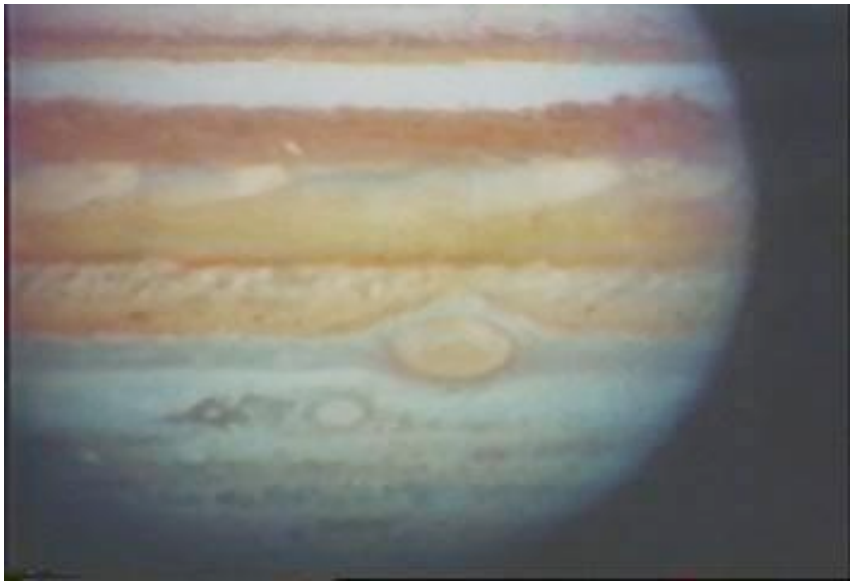


Low-level Features

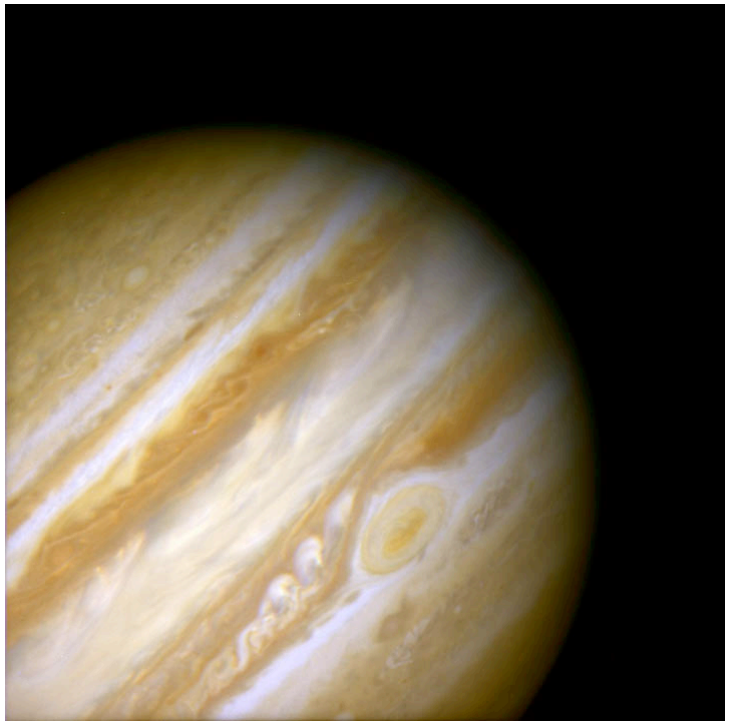


Query image

Known Item



Query



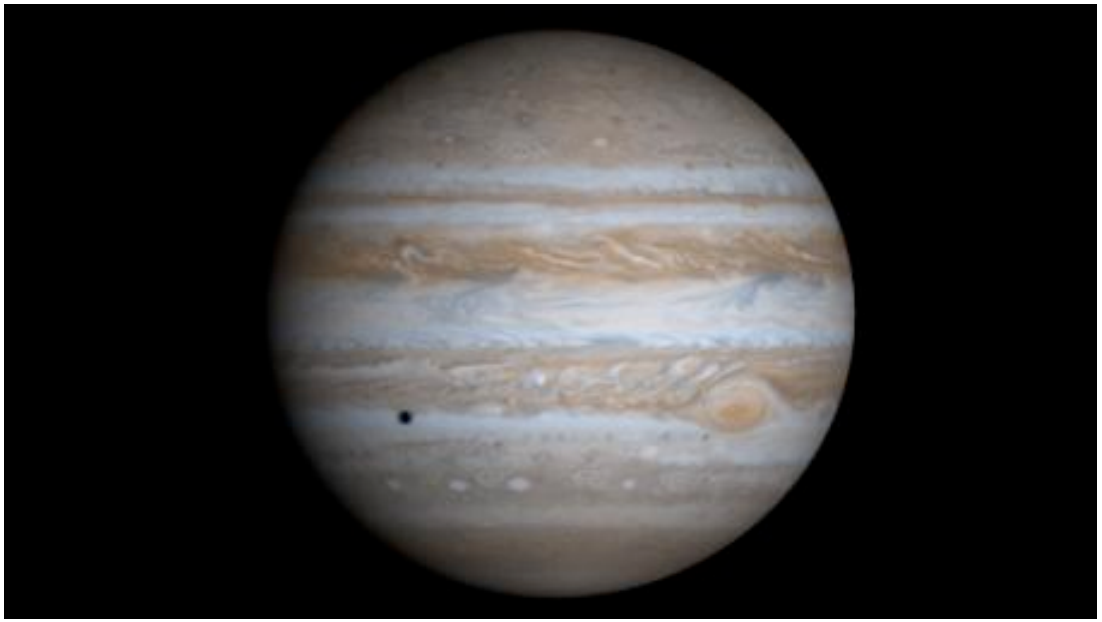
Results



...



Query



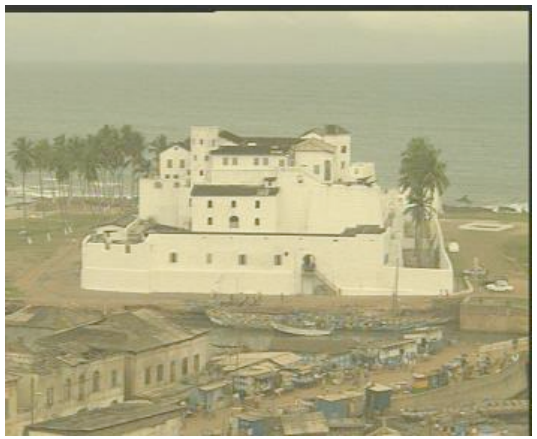
Results



...

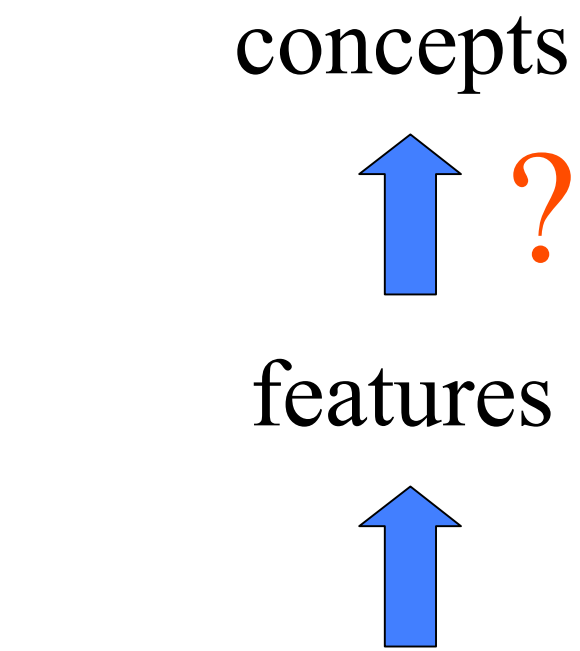


Observation



- Automatic approaches are successful under two conditions:
  - the query example is derived from the same source as the target objects
  - a domain-specific detector is at hand

Semantic gap…

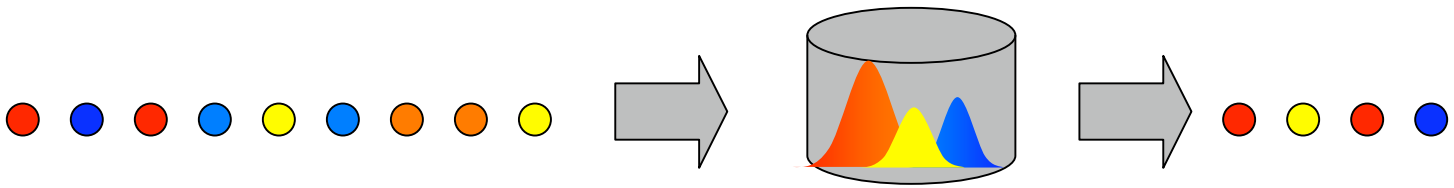


Complicating Factors

- What are Good Feature Models?
- What are Good Ranking Functions?
- Queries are Subjective!

Image Models

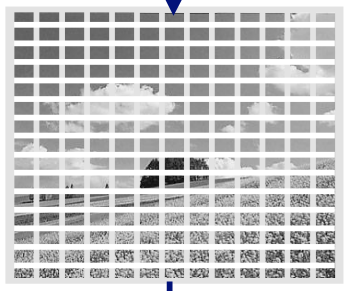
- Urn metaphor not useful
  - Drawing pixels useless
    - Pixels carry no semantics
  - Drawing pixel blocks not effective
    - chances of drawing exact query blocks from document slim
- Use Gaussian Mixture Models (GMM)
  - Fixed number of Gaussian components/clusters/concepts

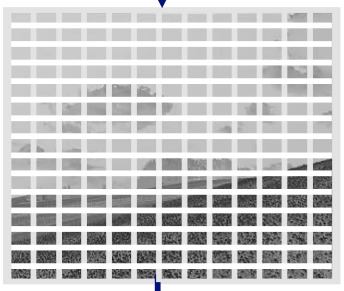


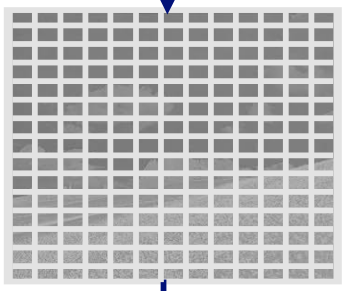
Key-frame representation

split colour channels

Take samples







DCT coefficients

position

675	9	12	11	1	9	4	1517	-9	-3	0	0	0	1	850	15	4	0	1	4	-2	1	1
661	7	13	5	-5	11	3	1536	2	-4	0	1	1	0	844	5	4	-2	0	1	-2	1	2
668	-7	13	3	-3	0	-1	1534	0	-5	0	0	0	0	837	3	3	-3	0	-2	1	1	3
665	10	11	2	4	5	2	1534	0	-5	0	0	0	0	829	0	3	-1	0	0	0	1	4
669	-5	18	7	-3	1	-5	1534	0	-5	0	0	0	0	833	-5	4	-1	0	3	-1	1	5

EM algorithm

Query model

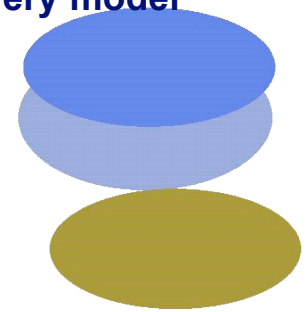
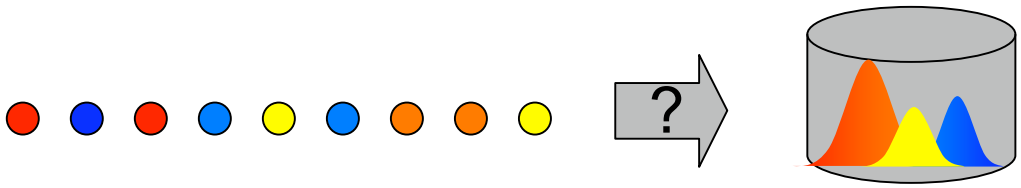


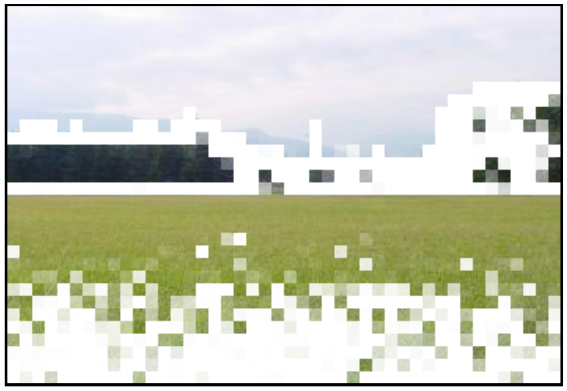
Image Models

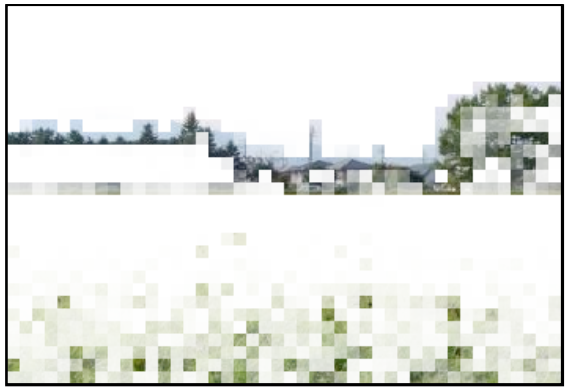


- Expectation-Maximisation (EM) algorithm
  - iteratively
    - estimate component assignments
    - re-estimate component parameters

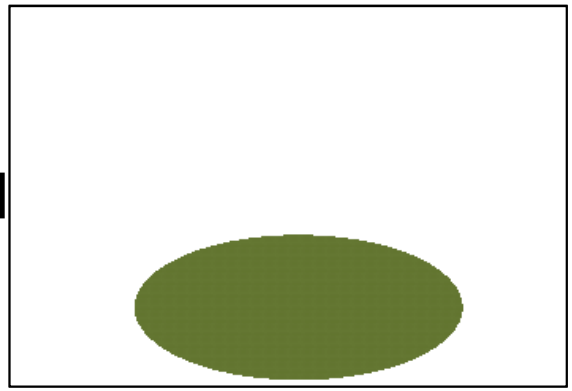
Expectation Maximization

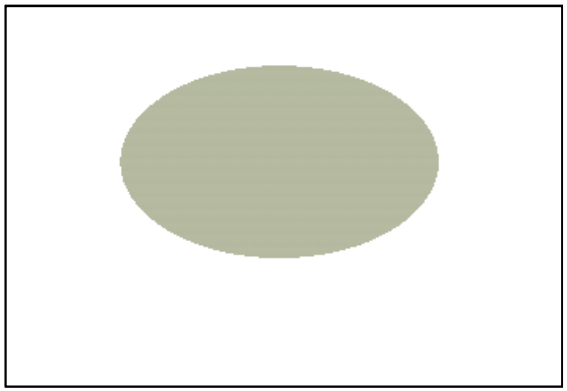
E

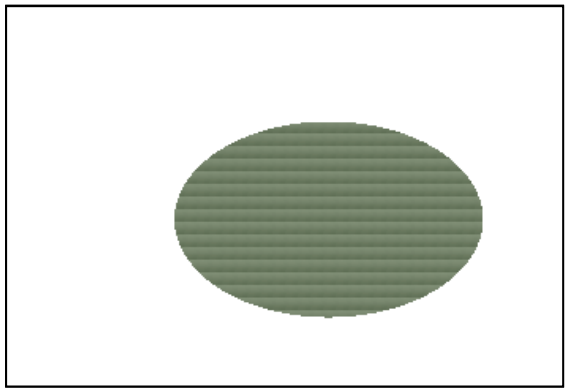




M



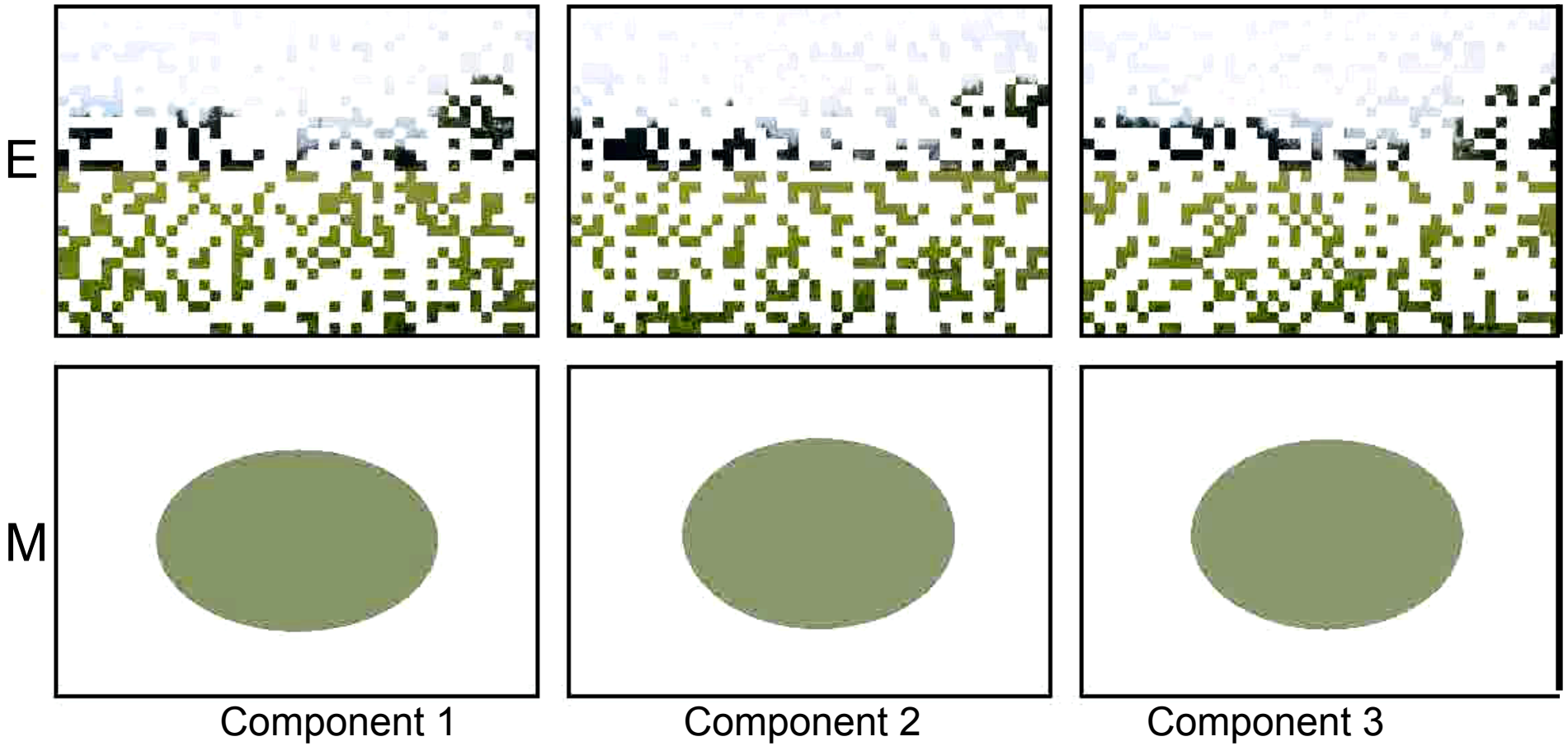




Component 1 Component 2 Component 3

Expectation Maximization

animation



Testing the Model on Corel

- 39 classes, ~100 images each
- Build models from all images
- Use each image as query
  - Rank full collection
  - Compute MAP (mean average precision)
    - AP=average of precision values after each relevant image is retrieved
    - MAP is mean of AP over multiple queries
  - Relevant ⇔ from query class

Example results

Query:

Top 5:



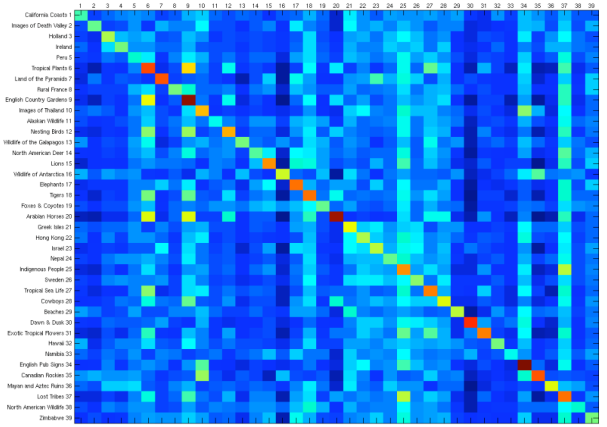


MAP per Class (mean: .12)

- English Pub Signs .36
- English Country Gardens .33
- Arabian Horses .31
- Dawn & Dusk .21
- Tropical Plants .19
- Land of the Pyramids .19
- Canadian Rockies .18
- Lost Tribes .17
- Elephants .17
- Tigers .16
- Tropical Sea Life .16
- Exotic Tropical Flowers .16
- Lions .15
- Indigenous People .15
- Nesting Birds .13
- …

- …
- Sweden .07
- Ireland .07
- Wildlife of the Galapagos .07
- Hawaii .07
- Rural France .07
- Zimbabwe .07
- Images of Death Valley .07
- Nepal .07
- Foxes & Coyotes .06
- North American Deer .06
- California Coasts .06
- North American Wildlife .06
- Peru .05
- Alaskan Wildlife .05
- Namibia .05

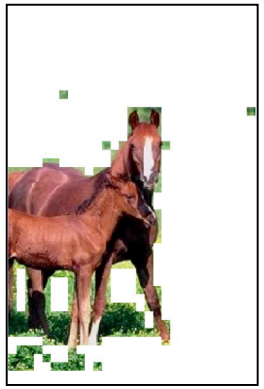
Class confusi



- Query from class A
- Relevant ⇔ from class B
- Queries retrieve images from own class
- Interesting mix-ups
  - Beaches – Greek islands
  - Indigenous people – Lost tribes
  - English country gardens – Tropical plants – Arabian Horses
- Similar backgrounds

Background Matching

Query:



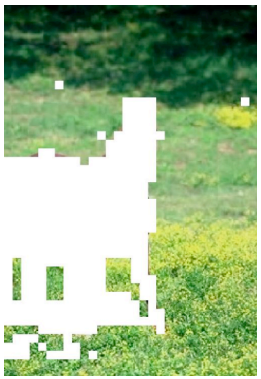
Top 5:





Background Matching

Query:



Top 5:

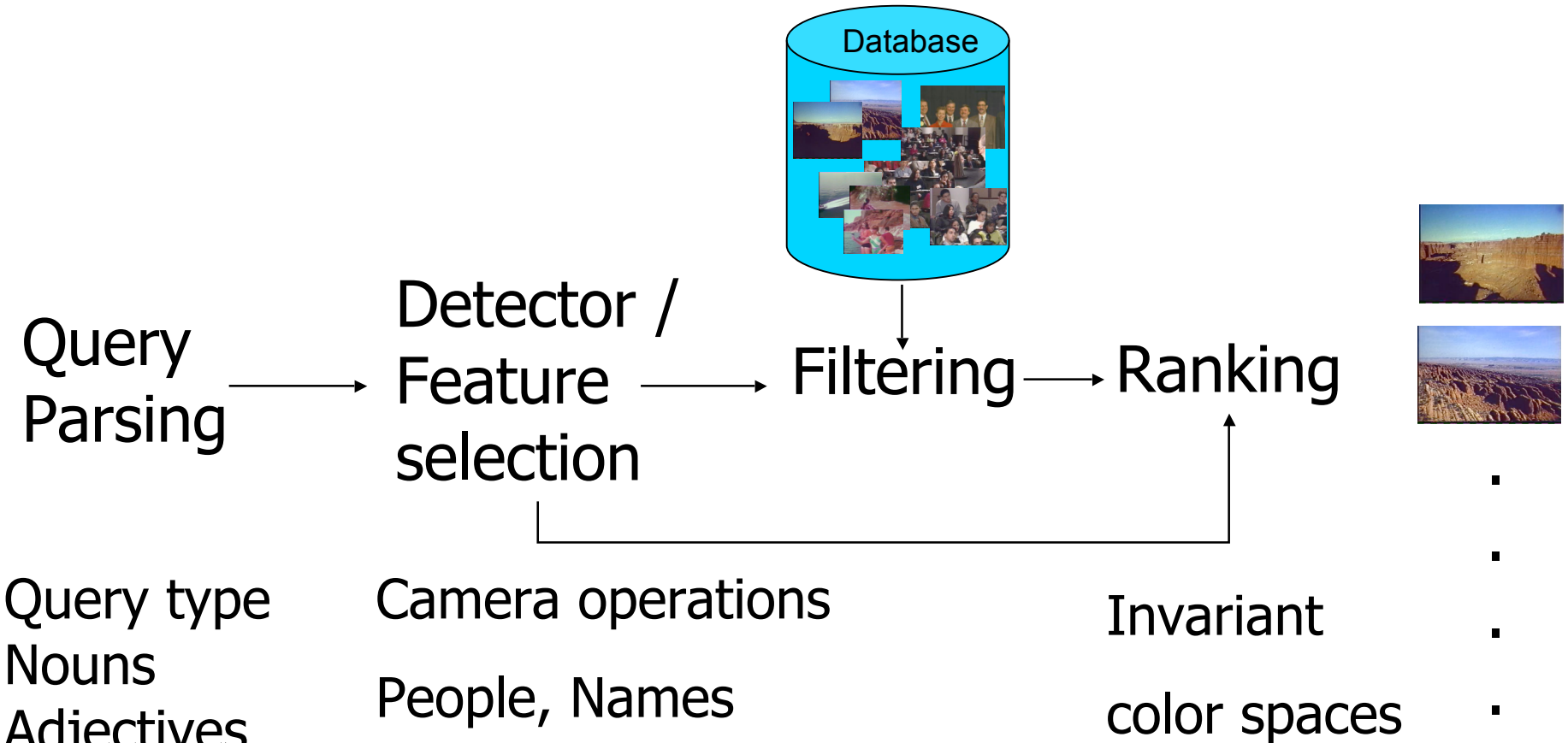




Other Approaches

- Generic Detectors
- Domain-specific Knowledge
- Collaborative Filtering

Retrieval Process



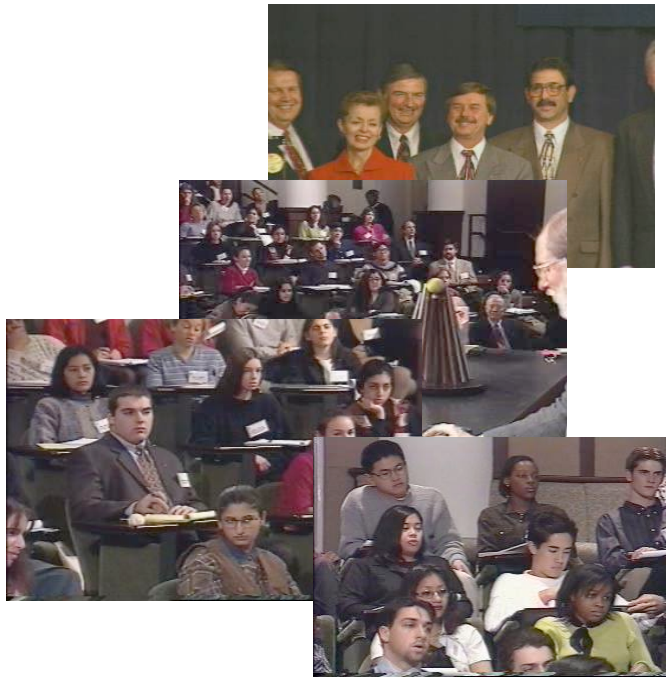
Parameterized detectors

Example

Results

Topic 41 'Shots with at least 8 people'

People detector <1, 2, 3, many>



Detectors

The universe and everything

FOCUS

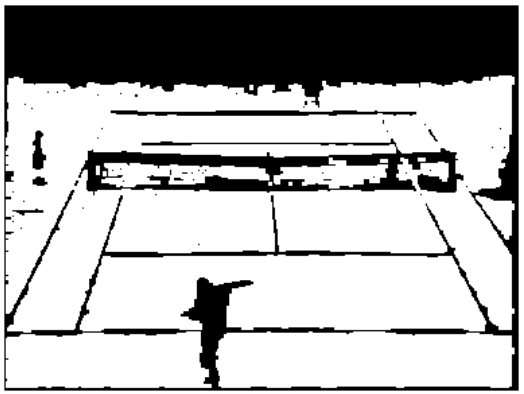
Camera operations (pan, zoom, tilt, …)
People (face based)
Names (VideoOCR)
Natural objects (color space selection)
Physical objects (color space selection)
Monologues (specifically designed)
Press conferences (specifically designed)
Interviews (specifically designed)

Domain specific detectors


Aircraft
I. Allawi
Anchor
Animal
Y. Arafat
Baseball
Basketball
Beach
Bicycle
Bird
T. Blair
Boat
Building
Bus
G. Bush jr.
G. Bush sr.
Candle
Car
Cartoon
Chair
Charts
B. Clinton
Cloud
Corp. Leader
Court
Crowd
Cycling
Desert
Dog
Drawing
Drawing & Cartoon
Duo-anchor
Entertainment
Explosion
Face
Female
Fire weapon
Fish
Flag
Flag USA
Food
Football
Golf
Gov. building
Gov. leader
Graphics
Grass
H. Nasrallah
Horse
Horse racing
House
H. Jintao
Indoor
J. Kerry
E. Lahoud
Male
Map
Meeting
Military
Monologue
Motorbike
Mountain
Natural disaster
News paper
Night fire
Office
Outdoor
Overlayed text
People
People marching
Police/ security
C. Powell
Prisoner
Racing
Religious leader
River
Road
Screen
A. Sharon
Sky
Smoke
Snow
Soccer
Split screen
Sports
Studio
Swimming pool
Table
Tank
Tennis
Tower
Tree
Truck
Urban
Vegetation
Vehicle
Violence
Walking
Water body
Waterfall
Weather

Player Segmentation

Original image



Initial segmentation

Final segmentation

Advanced Queries

Show clips from tennis matches,

starring Sampras,

playing close to the net;

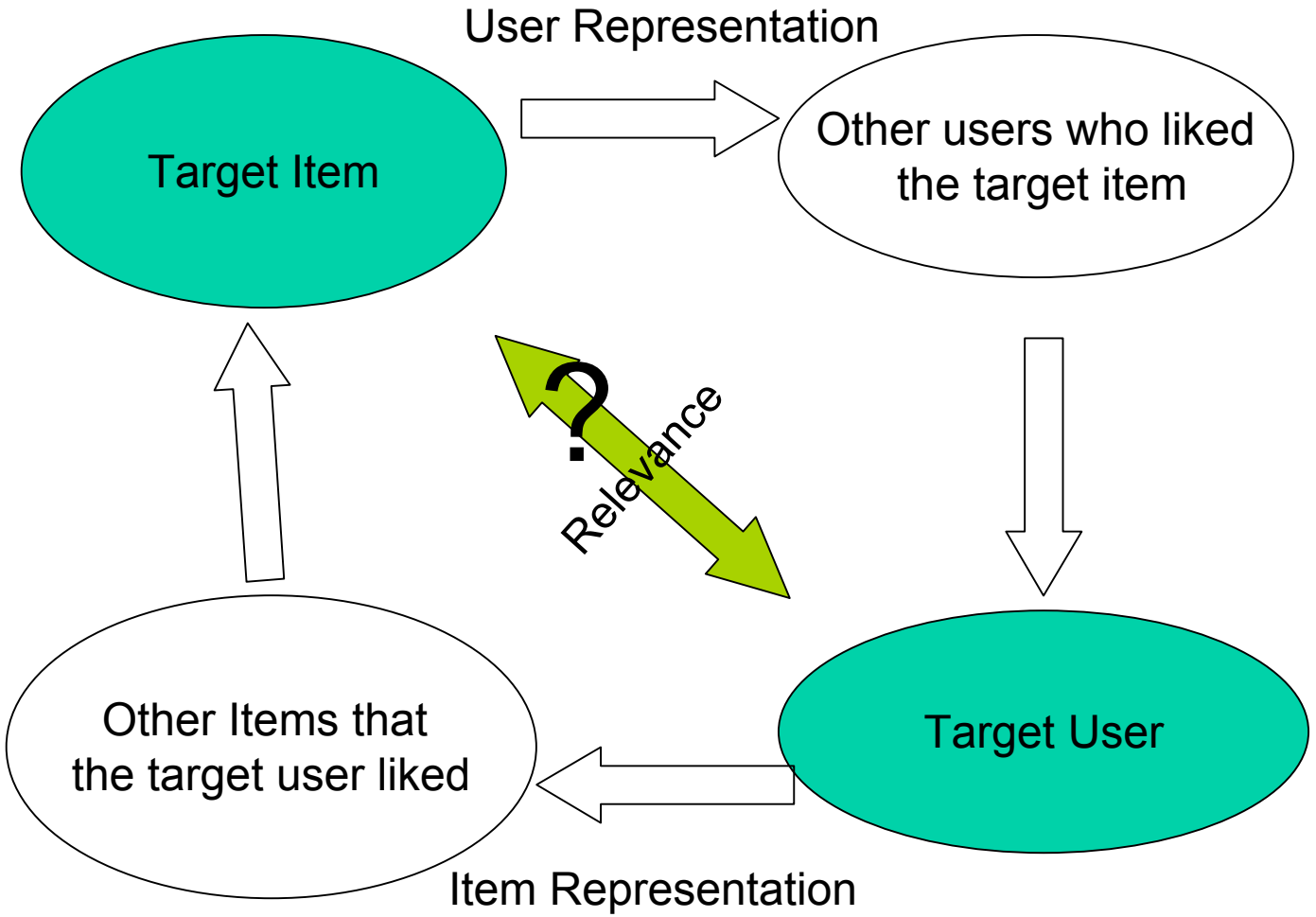
Collaborative Filtering

- Also: **social information filtering**
  - Compare user judgments
  - Recommend differences between similar users
- People's tastes are not randomly distributed
- You are what you buy (Amazon)

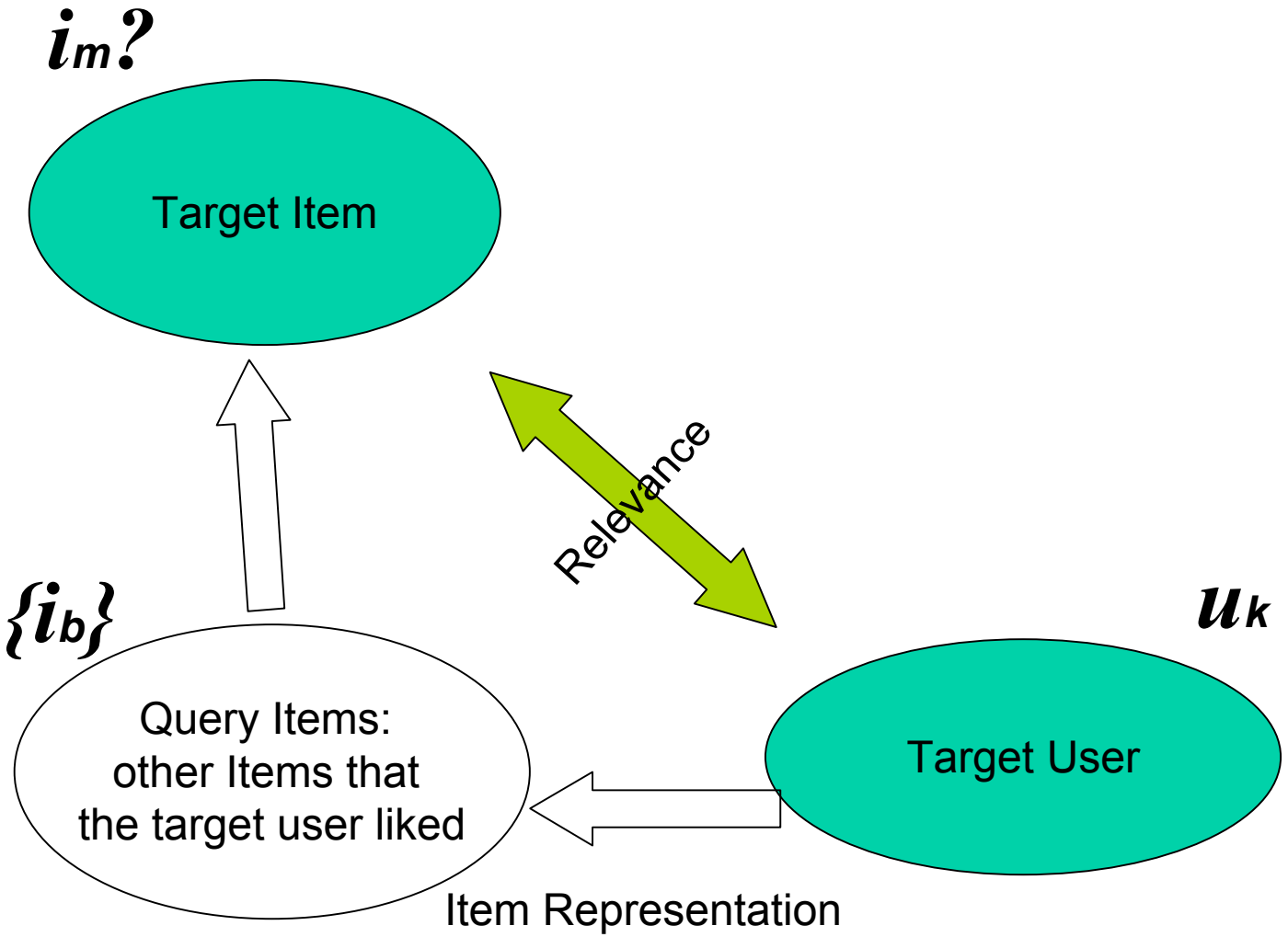
Collaborative Filtering

- Benefits over content-based approach
  - Overcomes problems with finding suitable features to represent e.g. art, music
  - Serendipity
  - Implicit mechanism for qualitative aspects like style
- Problems: large groups, broad domains

User-Item Relevance Models



Item Representation (Notation)

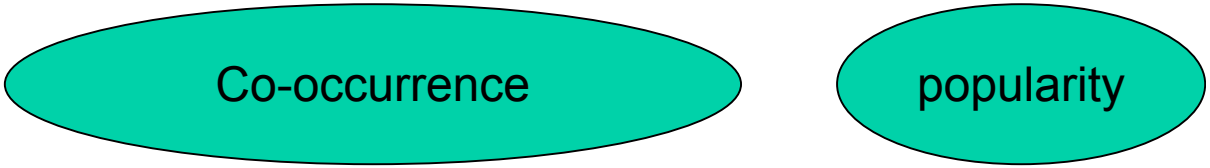


User-Item Relevance Models

- Item representation
  - Use *items that I liked* to represent the target user
  - Assume these item ratings are independent
  - Linear interpolation smoothing addresses sparsity

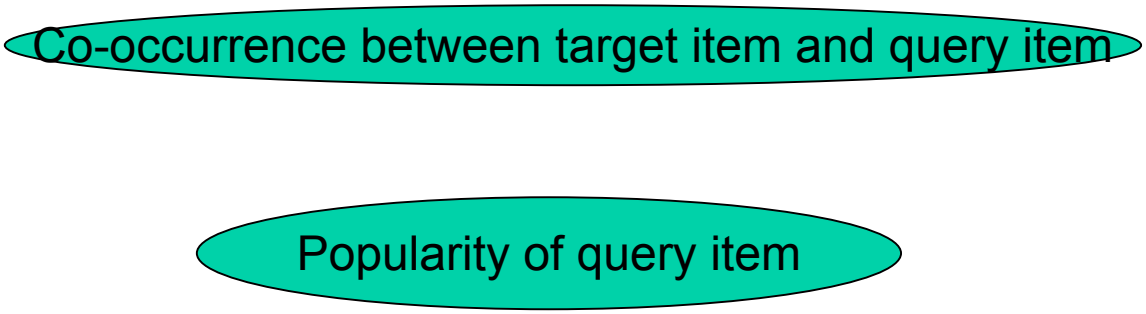
User-Item Relevance Models

- Probabilistic justification of Item-based CF
  - The RSV of a target item is the combination of its **popularity** and its **co-occurrence** with items (*query items*) that the target user liked.

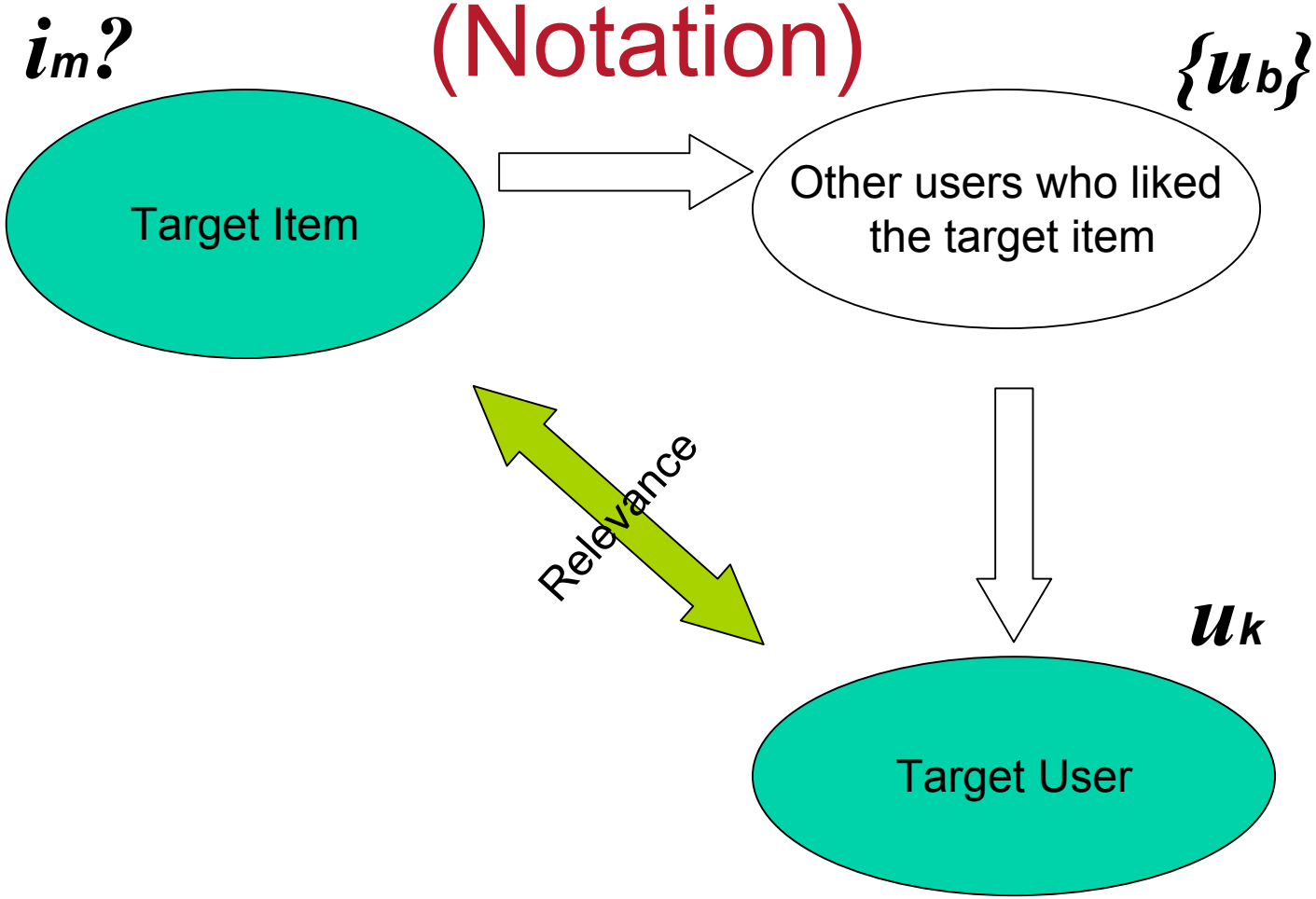


User-Item Relevance Models

- Probabilistic justification of Item-based CF
  - The RSV of a target item is the combination of its **popularity** and its **co-occurrence** with items (*query items*) that the target user liked
  - Item co-occurrence should be emphasized if more users express interest in target item as well as query item
  - However, item co-occurrence should be suppressed when the popularity of the query item is high

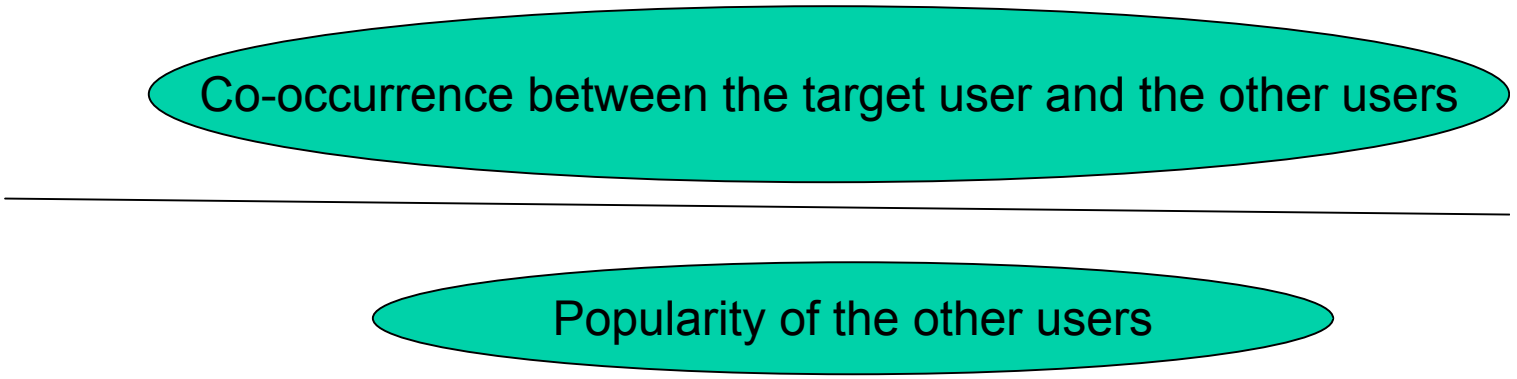


User Representation (Notation)



User-Item Relevance Models

- Probabilistic justification of User-based CF
  - The RSV of a target item towards a target user is calculated by the target user's **co-occurrence** with other users who liked the target item
  - User co-occurrence is emphasized if more items liked by target user are also liked by the other user
  - However, this co-occurrence should be suppressed when this user liked many items



Summarizing

- Language Modeling Approach to IR
  - Powerful formalism that has been applied successfully to many retrieval problems
- Open issue:
  - How do these various generative models interrelate?
    - E.g., how to take content and metadata into account during collaborative filtering?
  - How to handle structural knowledge?

Summarizing

- DB+IR
  - Desirable now that IR models deal with more than just bag-of-words
  - Efficiency concerns may be addressed
- Open issue:
  - Will progress in IR+DB really make IR research easier?

Finally

- Many thanks to these colleagues for their contributions to this tutorial:
  - **Thijs Westerveld**
  - Georgina Ramirez
  - Roberto Cornacchia
  - Marcin Zukowski, Sandor Héman and Peter Boncz
  - Jun Wang
  - Vojkan Mihajlovic and Djoerd Hiemstra
  - Mounia Lalmas