Outline

- Quality of query result
- User-system interaction techniques
- MM applications: SHIATSU case study
Effectiveness

- ! Remind from previous lesson...

- Effectiveness in term of:
  1. quality of query result
  2. user provided tools (i.e., interfaces) for
     - query formulation
     - result interpretation

- Let's focus on point 1..

Effectiveness metrics

- Traditional metrics for evaluating the effectiveness objects are precision (P) and recall (R)
Precision and recall

\[ P = \frac{|\text{Retrieved and Relevant}|}{|\text{Retrieved}|} \]
- measures the effect of false hits

\[ R = \frac{|\text{Retrieved and Relevant}|}{|\text{Relevant}|} \]
- measures the effect of false drops

How can a user effectively search?

- Although with traditional DB’s and a few attributes this might be a reasonable assumption, when we consider MM DBs with many attributes/features it is not clear how a user might guess the right query and the right combination of weights
- E.g., how can you define the 64 weights of a color-based search using the weighted Euclidean distance?
The idea of relevance feedback

- Shift the burden of finding the “right query formulation” from the user to the system [RHO+98]
- For this to be possible, the user has to provide the system with some information about “how well” the system has performed in answering the original query
- This user feedback typically takes the form of relevance judgments expressed over the answer set
- The “feedback loop” can then be iterated multiple times, until the user gets satisfied with the answers

Relevance judgments

- The most common way to evaluate the results is based on a 3-valued assessment:
  - **Relevant**: the object is relevant to the user
  - **Non-relevant**: the object is definitely not relevant (false drop)
  - **Don’t care**: the user does not say anything about the object

- Information provided by the relevant objects constitutes the so-called “positive feedback”, whereas non-relevant objects provide the so-called “negative feedback”
  - It’s common the case of systems that only allow for positive feedback

- “Don’t care” is needed also to avoid the user the task of assessing the relevance of all the results
- Models that allow a finer assessment of results (e.g., relevant, very relevant, etc.) have also been developed
A practical example (1)

Query Image

Euclidean distance

32-D HSV histograms

This is the initial query, for which 2 objects are assessed as relevant by the user

Precision = 0.3 (including the query image)

I. Bartolini

Multimedia Data Management M

A practical example (2)

Query Image

These are the results of the “refined” (new) query, generated using the 1st strategy we will see

Precision = 0.6 (including the query image)

I. Bartolini

Multimedia Data Management M
A practical example (3)

Query Image

These are the results of the “refined” (new) query, generated using the 2nd strategy we will see

Precision = 0.8 (including the query image)

I. Bartolini  Multimedia Data Management M

A practical example (4)

Query Image

And these are the results obtained by combining the 2 strategies…

Precision = 0.9 (including the query image)

I. Bartolini  Multimedia Data Management M
Basic query refinement strategies

- When the feature values are vectors, two basic strategies for obtaining a refined query from the previous one and from the user feedback are:

Query point movement:
the idea is simply to move the query point so as to get closer to relevant objects

Re-weighting:
the idea is to change the weights of the features so as to give more importance to those features that better capture, for the given query at hand, the notion of relevance

The 1st formulation of the query point movement (QPM) strategy dates back to 70's, when it was proposed by J.J. Rocchio in the context of text retrieval systems based on the Vector Space model

Rocchio’s formula is:

$$q_{\text{new}} = q_{\text{old}} + \beta \times \frac{\sum_{p_j \in \text{Rel}} (p_j - q_{\text{old}})}{|\text{Rel}|} - \gamma \times \frac{\sum_{p_j \in \text{NonRel}} (p_j - q_{\text{old}})}{|\text{NonRel}|}$$

where:
- $q_{\text{old}}$ is the previous query point
- $\text{Rel}$ is the set of relevant objects that have been retrieved by $q_{\text{old}}$
- $\text{NonRel}$ is the set of non-relevant objects that have been retrieved by $q_{\text{old}}$
- $\beta$ and $\gamma$ are non-negative parameters that control at which speed the query point moves towards relevant objects and far from non-relevant objects
Re-weighting

- The idea of the re-weighting strategy is to analyze the relevant objects in order to understand if some feature (dimension) is more important than others in determining “what makes an object relevant”

![Diagram showing re-weighting strategy](image)

- The feature F2 allows a better discrimination than F1 of relevant and non-relevant objects

Variance-based re-weighting

- For the relevant case of weighted Euclidean distances, the re-weighting strategy is easily implemented as follows:
  - Let Rel = \{p_1, …, p_{|Rel|}\} be the set of relevant objects retrieved by \(q_{\text{old}}\)
  - Let \(p_{i,j}\) be the feature value of \(p_j\) for the i-th feature (i=1,…,D)

  - The weight \(w_i\) of the i-th feature is estimated as \(w_i \propto 1/\sigma_i^2\), that is, the inverse of the variance of feature values along the i-th coordinate
    - In the figure \(w_2 > w_1\) since the variance on F2 is less than the variance on F1

  - Besides the intuition, this strategy has a theoretical justification, which relies on the minimization of distances from the relevant objects [RH00]
Problems with relevance feedback

- Supporting relevance feedback (RF) is costly for both user and system (numerous iterations, I)
- No feedback information is retained beyond one search
- Querying twice with the same/very similar object requires the user to repeat the entire process! 😁

FeedbackBypass (FB) [BCW00, BCW01, Bar05] represents an efficient solution:

it is able to cut down short I, complementing the role of RF

Motivations

- Each RF loop maps query object $Q$ to “optimal” set of query parameters $P$
- $f_{OPT}(Q) : R^D \rightarrow R^N$ function which determines optimal query parameters
- Cut RF loop short by “predicting”
  $P = f_{OPT}(Q)$
- Mathematically speaking: approximate $f_{OPT}$
FeedbackBypass

- Increases the performances of traditional relevance feedback techniques; it complements the role of relevance feedback engines by storing and maintaining the query parameters determined during the feedback loop over time.

- Alternative technologies for the approximation phase, e.g.,
  - Wavelet or
  - Support Vector Machine (SVM)

---

Learning from feedback

- User sends query & feedback to retrieval engine.
- Retrieval engine receives query and returns results.
- User receives results and sends feedback to feedback module.
- Feedback module adjusts optimal parameters.
- Adjusted parameters are sent back to retrieval engine.
- Retrieval engine uses predicted parameters for better results.
- Predicted parameters are sent to feedback learning module.

---

I. Bartolini
Multimedia Data Management M
Basic idea

- Break $f_{\text{OPT}}(Q) : R^D \rightarrow R^N$ down into components $f_{\text{OPT},i}(Q)$
- Construct approximation/interpolation per dimension using wavelets
- Refine wavelet over time

Visual example
Visual results: default strategy

Visual results: FB prediction
Content-based MM applications

- In addition to all case studies considered till now, we complete our overall view of MM applications by focusing on a *general framework for the smart management of video collection*, namely SHIATSU…

- However, let’s keep in mind that many content-based MM data retrieval systems (both commercial and research prototypes) have been proposed in the last ten years, especially for image and video DBs


I. Bartolini  Multimedia Data Management M  25

Content-based video retrieval systems

- How commercial systems tackle the problem?

- **YouTube** or **Vimeo** allows the user to search videos based on keywords
  - considering the original Web context (e.g., file name, title, surrounding text)
  - like image search extensions of Google and Yahoo

![](image-url)
SHIATSU case study [BPR10, BR10a, BR10b, BPR13]

- SHIATSU: Semantic-Based Hierarchical Automatic Tagging of Videos by Segmentation using Cuts

- Based on
  - Shot boundaries detection
  - Hierarchical annotations

SHIATSU in practice

- Videos are segmented into shots
  - *key frames* sequences sharing the same visual features
  - Each shot is modeled as a set of representative key frames (e.g., first, last, first-middle-last, etc.)
- Shots are processed in order to extract semantic labels from similar pre-annotated key frames (*training* images DB)
  - most recurring shot labels are then propagated at the video level, allowing semantic indexing of both videos and shots for retrieval purposes
SHIATSU overview

- SHIATSU consists of two main components:
  - the Shot Detection Module and
  - the Video/Shot Tagging Module

- They work sequentially to segment a video into coherent frame sequences and to attach concepts to them

- Shot tags are then propagated to the whole video, so as to obtain semantic indices for both the video and its shots

Shot detection module

- Since videos can hold a wide range of subjects, detection of hard cuts and gradual transitions is used in order to segment a video object into sequences sharing the same visual content

- The shot detection algorithm computes color histograms and object edges of every frame

- Features are used to compare consecutive frames by applying two different distance metrics:
  - usually a shot transition produces a change in both the color and the texture structure of the frames

- Shot selection process is done with a double dynamic threshold system which takes into account video content in order to adapt to different video types

- Frames are filtered on their color features and then on their edge features
Video tagging module

- The tagging module of SHIATSU exploits the *Imagination* system and uses a set of pre-annotated images as a knowledge base.

- Semantic concepts in the knowledge base can be organized into:
  - either a tree-shaped taxonomy (terms are linked with a father/child relationship) or
  - as a flat structure (all terms at the same level) and can be easily modified and expanded.

- When provided with a key frame, the module extracts its visual features, exploits an index structure, i.e., the M-Tree index, to efficiently retrieve images having similar features and proposes semantic concepts depending on the similarity of the shot with the images in the knowledge base.

Shot tagging

- Using shot cuts timestamps, SHIATSU extracts a set of key frames of each shot sequence, computes their visual features and compares them with those contained in the knowledge base.

- The module suggests a set of concepts for each key frame
  - only terms recurring in the majority of key frames are selected as suitable concepts to describe the whole shot sequence.

- To avoid producing an overwhelming number of tags for each shot, only the most frequent tags retrieved for each key frame in the sequence are maintained.

- The proposed tags can then be reviewed by the user so as to verify them and are finally stored into the database.
Hierarchical tagging

- Shot tags are useful to browse sequences across different videos, but they could be too specific to index a whole video; system selects video tags from the set of shot tags depending on their frequency and the length of the shot they are associated to.

- Rank $R(t)$ for every shot tag $t$ is defined as

$$R(t) = \frac{1}{N_s} \sum_s W(s)A(t,s)$$

$N_s$ is the total number of shots, $W(s)$ is the shot length and $A(t,s)$ is 1 if shot $s$ contains tag $t$, 0 otherwise.

- Tags are ordered by descending $R(t)$ values and the first $k$ tags become video tags.
  - Concepts extracted from long shots and/or that appear in several ones are more relevant to describe the content of a whole video.
Video retrieval

- Users can search videos (both at key frame, shot and video level) through different modalities:
  
  - **Keyword-based Search (KS):** given a set of keywords as query semantic concepts, videos/shots are selected by applying a co-occurrence search on the Tag DB
  - **Feature-based Search (FS):** given an input query, a Nearest-Neighbors search is performed on the Features DB
  - **Keyword&Feature-based Search (KFS):** it combines KS and FS

- Note that, if present, also *meta-data information* can be integrated in the query process as further filter
  - i.e., traditional DB type filter

---

**Keyword-based search example**

- **Keywords selection panel**
- **Returned shots**
- **Shot rank**
- **Current shot tags**
- **Current video tags**
- **Current shot&video playback**