ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Multimedia and Multimodal Information Retrieval

Surabhi Narayan

Department of Computer Science Engineering

Slides collated by :

Gaurav Mahajan, TA, VIII sem

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

PES UNIVERSITY

Multimedia and Multimodal Information Retrieval - Techniques for Content Processing, and Examples

Surabhi Narayan
Department of Computer Science Engineering

Techniques for Content Processing

PES UNIVERSITY

- **Transformation**: This kind of operation converts the format of media items, for making the subsequent analysis steps more efficient or effective. For instance, a video transformer can modify an MPEG2 movie file to a format more suitable for the adopted analysis technologies (e.g., MPEG); likewise, an audio converter can transform music tracks encoded in MP3 to WAV, to eliminate compression and make content analysis simpler and more accurate.

- **Feature Extraction:** calculates low-level representations of media contents, i.e. feature vectors, in order to derive a compact, yet descriptive, representation of a pattern of interest. Such representation can be used to enable content based search, or as input for classification tasks. Examples of visual features for images are colour, texture, shape, etc.; examples of aural features for music contents are loudness, pitch, tone (brightness and bandwidth).

Techniques for Content Processing

- **Classification**: assigns conceptual labels to content elements by analyzing their raw features; the techniques required to perform this operations are commonly known as machine learning. For instance, an image classifier can assign to image files annotations expressing the subject of the pictures (e.g., mountains, city, sky, sea, people, etc.), while an audio file can be analyzed in order to discriminate segments containing speech from the ones containing music.

Techniques for Content Processing

PES UNIVERSITY

Techniques for Audio Analysis

Techniques	Purpose
Audio Segmentation	split audio track according to the nature of its content For example: a file can be segment according to the presence of noise, music, speech, etc.
Audio event identification	identify the presence of events like gunshots and scream in an audio track.
Music genre identification	to identify the genre (e.g., rock, pop, jazz, etc.) or the mood of a song
Speech recognition	to convert words spoken in an audio file into text. Speech recognition is often associated with Speaker identification, that is to assign an input speech signal to one person of a known group

Techniques for Content Processing

PES UNIVERSITY

Techniques for Image Analysis

Techniques	Purpose
Semantic Concept Extraction	the process of associating high-level concepts (like sky, ground, water, buildings, etc.) to pictures.
Optical character recognition (OCR)	to translate images of handwritten, typewritten or printed text into an editable text.
Face recognition and identification	to recognize the presence of a human face in an image, possibly identifying its owner.
Object detection and identification	to detect and possibly identify the presence of a known object in the picture.

Techniques for Content Processing

PES UNIVERSITY

Techniques for Video Analysis

Techniques	Purpose
Scene detection	detection of scenes in a video clip; a scene is one of the subdivisions of a play in which the setting is fixed, or that presents continuous action in one place
Video text detection and segmentation	to detect and segment text in videos in order to apply image OCR techniques.
Video summarization	to create a shorter version of a video by picking important segments from the original.
Shot detection	detection of transitions between shots. Performed by means of Keyframe segmentation algorithms that segment a video track according to the key frames produced by the compression algorithm.

Techniques for Content Processing

Having come across a variety of these methods, how do we leverage them for an effective multimodal analysis?

The answer lies in arbitrary combinations of transformation, feature extraction techniques and classification operation that result in several analysis algorithms.

The techniques shown in the previous tables can be used in isolation, to extract different features from an item. Since the corresponding algorithms are probabilistic, each extracted feature is associated with a confidence value that denotes the probability that item X contain feature Y. To increase the confidence in the detection, different analysis techniques can be used jointly to reinforce each other.

Examples of MIR Query Languages

- The last two decades have witnessed to a lot of efforts in the definition of more expressive and structured query languages, designed specifically for multimedia retrieval.
- **POQL** : is a general purpose query language for object oriented multimedia databases exposing arbitrary data schema.
- **MuSQL**: is a music structured query language, composed of a schema definition sub-language and a data manipulation sub-language.
- One of the latest attempts in providing a unified language for MIR is represented by the MPEG Query Format (MPQF).

Examples of Commercial MIR System

- **PHAROS** – Proposed by Italy and implemented collaboratively with other European countries.
- **VITALAS** – European commission information society
- **THESEUS** – Germany
- **Quaero, Voxalead** – France
- **Midomi (SoundHound), Google Images and Microsoft Bing, Tiltomo** - USA
- **Shazam** – UK (now owned by Apple)
- **SAPIR** – France, Italy, Israel
- **Blinkx** – UK (now known as RhythmOne and based in USA)

THANK YOU

Surabhi Narayan
Department of Computer Science & Engineering

surabhinarayan@pes.edu

PES UNIVERSITY

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Content Based Visual Information Retrieval

Surabhi Narayan
Department of Computer Science Engineering
Slides collated by:
Gaurav Mahajan (TA VIII sem)
Anshula Aithal (TA VIII sem)

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Metadata Searching

Surabhi Narayan

Department of Computer Science Engineering

Metadata Searching

- Text-based image retrieval (TBIR) systems are focused on text connected with the images or in relation to the images. For text-based image retrieval, one typically uses metadata to describe images.

- Metadata can be defined as data about data.

- Image metadata can be of different kinds, e.g., tags, keywords, and descriptors of relevance for the image.

- This includes data added by the capturing device, e.g., time/date and GPS coordinates, keywords manually added by individual users to describe the image (tags) or automatic image annotations added by the image retrieval system to simplify search and indexing. The latter is usually referred to as Auto- annotations or linguistic indexing.

Metadata Searching

- Images are first annotated with text and then searched using a text-based approach from traditional database management systems.

- However, since automatically generating descriptive texts for a wide spectrum of images are not feasible, most text-based image retrieval systems require manual annotation of images.

- Obviously, annotating images manually is a cumbersome and expensive task for large image databases and is often subjective, context-sensitive, and incomplete. As a result, it is difficult for the traditional text-based methods to support a variety of task-dependent queries.

Metadata Searching

- The text-based systems are fast as the string matching is computationally less time-consuming process.
- However, it is sometimes difficult to express the whole visual content of images in words and TBIR may end up in producing irrelevant results.
- Manually added tags are keywords added to the image by individual users. In theory, they represent the individuals' natural perception of the image.
- Manually added tags can be very helpful for the retrieval system if available.

Metadata Searching

- One concern that remains in dealing with manually added tags is the subjective nature of human tagging introduces variability, as individual users may possess different perceptions and employ diverse tags to describe the same images.

- For example, an image consisting of grass and flowers might be labeled as either “grass” or “flower” or “nature” by different people.

- There is a high probability of error occurrence during the image-tagging process when the database is large.

Metadata Searching

- As a result, text-based image retrieval cannot achieve high level of efficiency and effectiveness.

- For finding the alternative way of searching and overcoming the limitations imposed by TBIR systems, more intuitive and user-friendly

Metadata Searching

Advantages:

- Precise Retrieval
- Improved Search Relevance
- Facilitates multimodal search

Disadvantages:

- Manual Annotations are time consuming
- Annotations may be subjective
- Not Scalable

References

- “Content-Based Image Retrieval : Ideas, Influences, and Current Trends” - Vipin Tyagi, Springer Nature Singapore Pte Ltd., 2017.

PES UNIVERSITY

THANK YOU

Surabhi Narayan

Department of Computer Science Engineering

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Content Based Visual Information Retrieval

Surabhi Narayan
Department of Computer Science Engineering

Slides collated by:
Gaurav Mahajan (TA VIII sem)
Anshula Aithal (TA VIII sem)

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Semantic Retrieval

Surabhi Narayan
Department of Computer Science Engineering

Semantic Retrieval

- So far we have discussed about low-level features of images and the queries that a user can enter to retrieve relevant images.
- The queries are always of the high-level features such as keywords, text, images, etc.
- However, in general there is no direct link between high-level concepts and low-level features.

Semantic Retrieval

- Extensive experiments on CBIR systems show that low-level contents often fail to describe the high-level semantic concepts in user's mind.

- Therefore, the performance of CBIR is still far from user's expectations. Scientists have mentioned three levels of queries in CBIR.

Semantic Retrieval

- Level 1:
  - Retrieval by primitive features such as color, texture, shape, or the spatial location of image elements.
  - Typical query is a query by example, “find pictures like this.”

- Level 2:
  - Retrieval of objects of given type identified by derived features, with some degree of logical inference.
  - For example, “find a picture of a flower.”

- Level 3:
  - Retrieval by abstract attributes, involving a significant amount of high-level reasoning about the purpose of the objects or scenes depicted. This includes retrieval of named events, of pictures with emotional or religious significance, etc.
  - For example, “find pictures of a joyful crowd.”

Semantic Retrieval

- Levels 2 and 3 together are referred to as semantic image retrieval and the gap between Levels 1 and 2 as the semantic gap.

- Users in Level 1 retrieval are usually required to submit an example image or sketch as query.

- Semantic image retrieval is more convenient for users as it supports query by keywords or by texture.

- Therefore, to support query by high-level concepts, a CBIR system should provide full support in bridging the “semantic gap” between numerical image features and the richness of human semantics

Semantic Retrieval

- The human perception of visual contents is strongly associated to high-level, semantic information about the scene.

- Current Computer Vision techniques work at a lower level (as low as individual pixels).

- CBVIR systems that rely on low-level features only can answer queries such as:
  - Find all images that have 30% of red, 10% of orange and 60% of white pixels, where orange is defined as having a mean value of red = 255, green = 130, and blue = 0.
  - Find all images that have a blue sky above a green grass.
  - Find all images that are rotated versions of this particular image.

Semantic Retrieval

- In general case, the user is looking for higher-level semantic features of the desired image, such as "a beautiful rose garden", "a batter hitting a baseball", or "an expensive sports car".

- There is no easy or direct mapping between the low-level features and the high-level concepts.

- The distance between these two worlds is normally known as “semantic gap.”

Semantic Retrieval

- Currently there are two ways of minimizing the semantic gap:
  - The first consists of adding as much metadata as possible to the images, which was already discussed and shown to be impractical.
  - The second suggests the use of rich user interaction with relevance feedback combined with learning algorithms to make the system understand and learn the semantic context of a query operation

References

- “Content-Based Image Retrieval : Ideas, Influences, and Current Trends” - Vipin Tyagi, Springer Nature Singapore Pte Ltd., 2017. - Chapter 1

PES UNIVERSITY

THANK YOU

Surabhi Narayan
Department of Computer Science and Engineering

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Content Based Visual Information Retrieval

Surabhi Narayan
Department of Computer Science Engineering

Slides collated by:
Gaurav Mahajan (TA VIII sem)
Anshula Aithal (TA VIII sem)

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Content based Visual Information Retrieval - Introduction

Surabhi Narayan

Department of Computer Science Engineering

Content-based Visual Information Retrieval (CBVIR)

- Content-based image retrieval, also known as Query By Image Content (QBIC), is an automated technique that takes an image as a query and returns a set of images similar to the query.

- Content-based image retrieval techniques use visual contents of the images described in the form of low-level features like color, texture, shape, and spatial locations to represent and search the images in the databases.

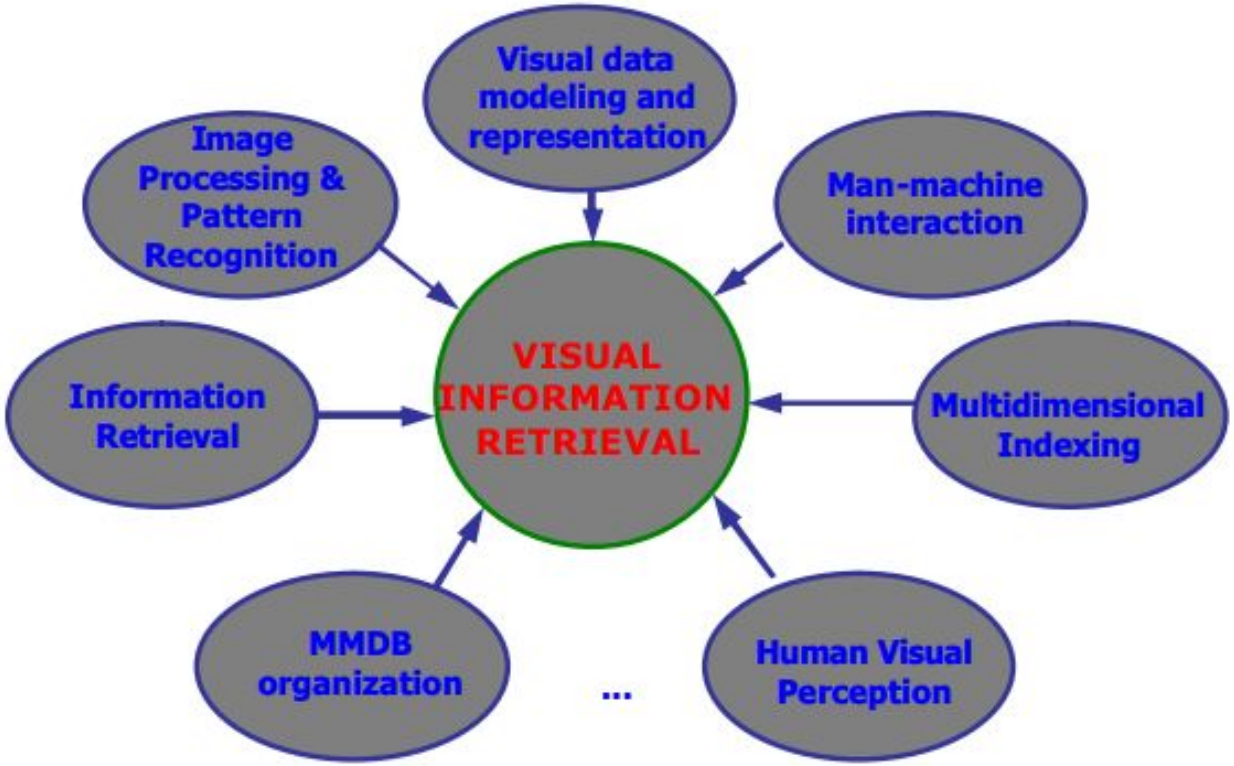
Evolution of CBVIR

- Research in the field of Content-Based Visual Information Retrieval (CBVIR) started in the early 1990's and is likely to continue during the first decade of the 21st century.

- Many research groups in leading universities and companies are actively working in the area and a fairly large number of prototypes and commercial products are already available.

Evolution of CBVIR

- Progress in visual information retrieval has been fostered by many research fields particularly: (text -based) information retrieval, image processing and computer vision, pattern recognition, multimedia database organization, multidimensional indexing, psychological modeling of user behavior, man-machine interaction, among many others.



Evolution of CBVIR

VIR systems can be classified in two main generations, according to the attributes used to search and retrieve a desired image or video file:

- **First-generation VIR systems**: use query by text, allowing queries such as “all pictures of red Ferraris” or “all images of Van Gogh’s paintings”. They rely strongly on metadata, which can be represented either by alphanumeric strings, keywords, or full scripts.

- **Second-generation (CB)VIR systems**: support query by content, where the notion of content, for still images, includes, in increasing level of complexity: perceptual properties (e.g., color, shape, texture), semantic primitives (abstractions such as objects, roles, and scenes), and subjective attributes such as impressions, emotions and meaning associated to the perceptual properties. Many second-generation systems use content-based techniques as a complementary component, rather than a replacement, of text-based tools.

Overview of how CBVIR works

- The system retrieves similar images when an example image or sketch is presented as input to the system.
- The query image is converted into the internal representation of the feature vector using the same feature extraction routine that was used for building the feature database.
- The similarity measure is employed to calculate the distance between the feature vectors of the query image and those of the target images in the feature database.

Overview of how CBVIR works

- Finally, the retrieval is performed using an indexing scheme which facilitates the efficient searching of the image database.

- Therefore, when similarity measurement is performed based on image features, the output set achieves a high retrieval performance.

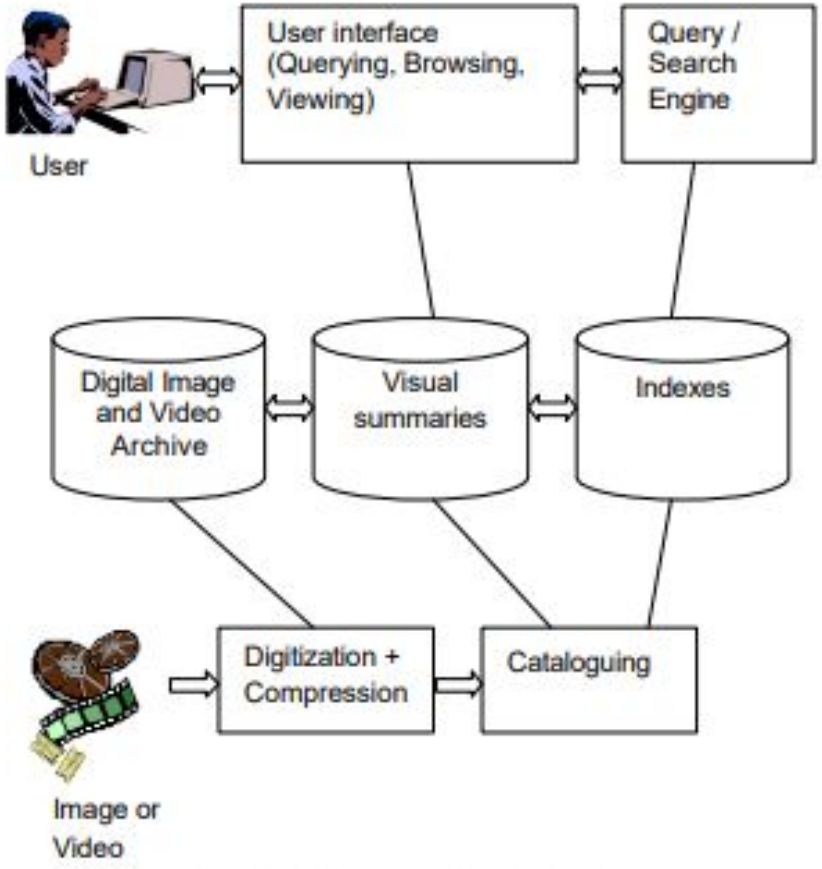


Figure 2. Block diagram of a CBVIR system.

CBVIR : Advantages

CBVIR has several advantages over traditional text-based image retrieval.

- Using the visual contents of the query image in CBVIR, it is a more efficient and effective way of finding relevant images than searching based on text annotations.
- It is more close to human perception of visual data. Also, CBIR does not consume the time in the manual annotation process as in the text-based approach.
- User-relevant feedback can also be incorporated to further improve the retrieval process to produce perceptually and semantically more meaningful retrieval results

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Typical Architecture of a CBVIR system

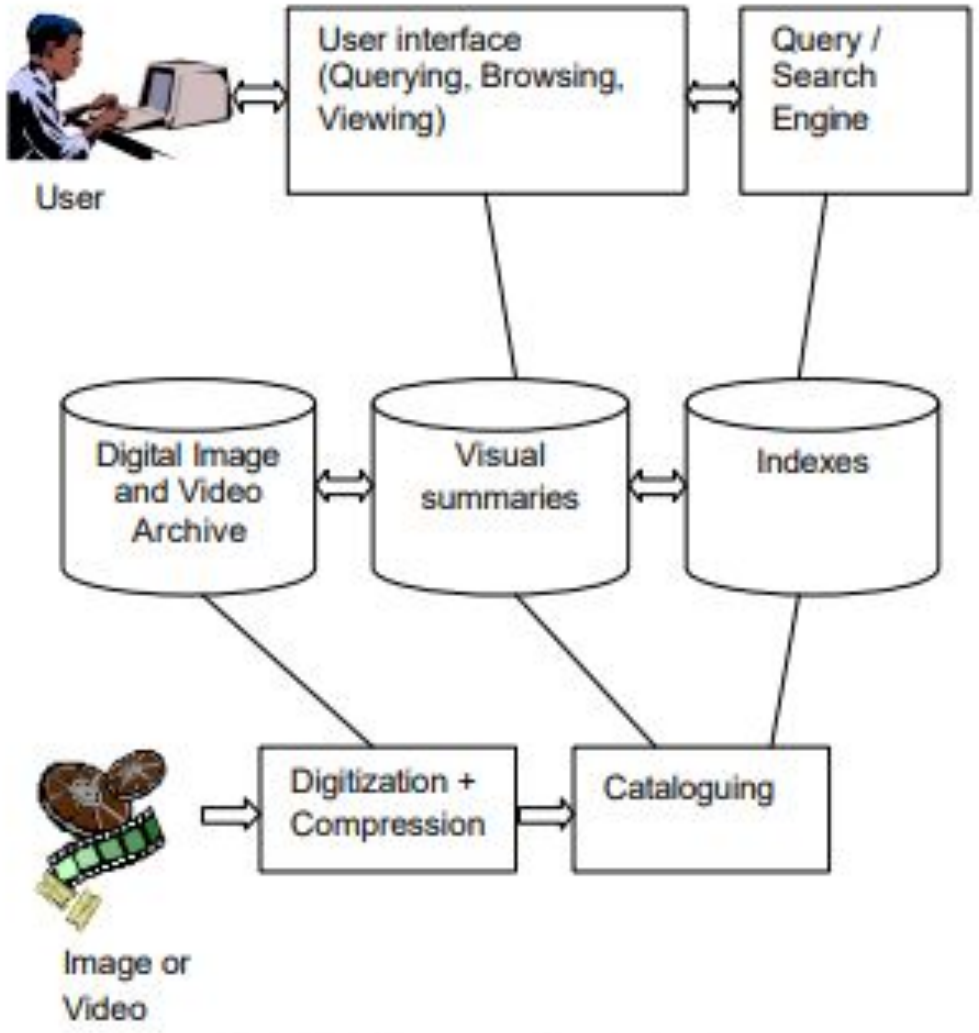


Figure 2. Block diagram of a CBVIR system.

User interface: friendly GUI that allows the user to interactively query the database, browse the results, and view the selected images / video clips.

Query / search engine: responsible for searching the database according to the parameters provided by the user

Digital image and video archive : repository of digitized, compressed images and video clips.

Visual summaries: representation of image and video contents in a concise way, such as thumbnails for images or keyframes for video sequences.

Typical Architecture of a CBVIR system

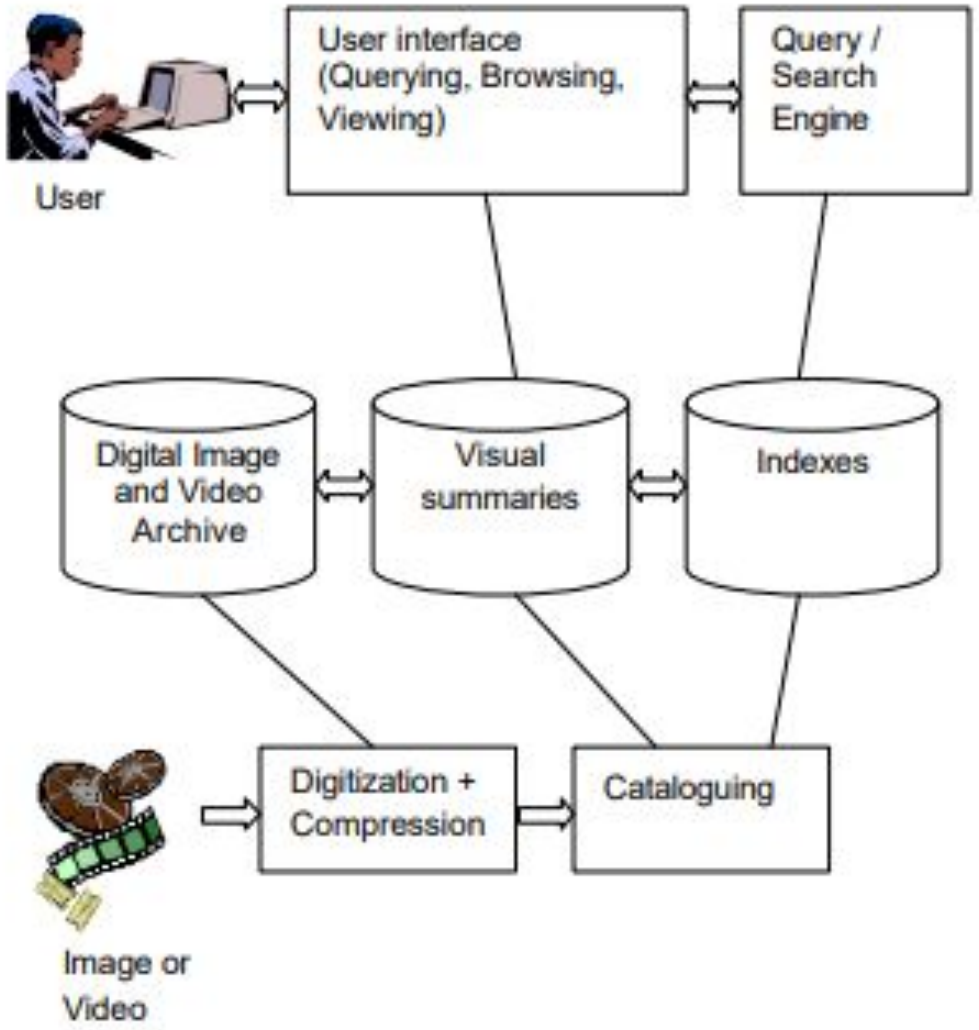


Figure 2. Block diagram of a CBVIR system.

Indexes: pointers to images or video segments.

Digitization and compression: hardware and software necessary to convert images and videos into digital compressed format.

Cataloguing: process of extracting features from the raw images and videos and building the corresponding indexes.

References

- “Content-Based Visual Information Retrieval” - Oge Marques and Borko Furht

PES UNIVERSITY

THANK YOU

Surabhi Narayan

Department of Computer Science Engineering

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Multimedia and Multimodal Information Retrieval

Surabhi Narayan

Department of Computer Science Engineering

Slides collated by :

Gaurav Mahajan, TA, VIII sem

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

PES UNIVERSITY

Multimedia and Multimodal Information Retrieval - Introduction, Requirements, Applications, and Challenges

Surabhi Narayan
Department of Computer Science Engineering

Introduction

- With emergence of Web 2.0 and lots of content available today, Web, mobile access, and digital television has boosted the production and consumption of audio-visual materials, making the Web a truly multimedia platform.
- This Unit will give us a concise overview of Multimedia Information Retrieval (MIR), the long-standing discipline at the base of audio-visual search engines, and connects the research challenges in this area to the objectives and research goals of Search Computing.
- The grand challenge of MIR is bridging the gap between queries and content: the former are either expressed by keywords, like in text search engines, or, by extension, with non-textual samples (e.g., an image or a piece of music). Unlike in text search engines, where the query has the same format of content and can be matched almost directly to it, query processing in MIR must fill an enormous gap

Introduction

- An MIR system can be seen as an infrastructure for governing two main processes:
  - the content process deals with metadata
  - the query process deals with user's query
- The link between the content process and the query process is represented by metadata, which encode the knowledge that the MIR system is able to extract from the media assets, index and use for answering queries.

Requirements

The requirements of an MIR system go beyond the problems faced in classical text based IR.

- **Opacity of Content**
- **Query Formulation Paradigm**
- **Relevance Computation**

Requirements

Opacity of Content

- When we consider a classic case of text IR, the query and content use the same medium. (Transparency of Content).
- But, MIR content is opaque.
- The knowledge necessary to verify if an item is relevant to a user's query is deeply embedded in it and must be extracted by means of a complex pre-processing.
- Example : Extracting speech transcriptions from a video.

Requirements

Query Formulation Paradigm

- In traditional engines, not only keywords but queries can be in the form of an analogy.
- Thus, content sample is similar to whatever is user searching for.
- But in case of MIR, content samples could be used as queries and those could be images, a piece of audio or a video fragment.

Requirements

Relevance Computation

- In text IR, relevance of documents to the user's query is computed as similarity scores. Which involves comparison between vectors of words in the query and document.
- In MIR, comparison has to be done on a wider variety of features – not only the medium of query and the content but also their

Applications

MIR applications requirements have been extensively addressed in the last three decades, both in the industrial and academic fields :

- **Architecture, Real estate and Design**
- **Broadcast media selection**
- **Cultural service**
- **Digital libraries**
- **Education**
- **Entertainment**
- **Journalism**
- **Investigation**
- **Multimedia directory**
- **Multimedia editing**
- **Remote sensing**
- **Social**
- **Surveillance**

Challenges

MIR systems operate on a very heterogeneous spectrum of content, ranging from home-made content created by users (like vlogs, reviews, etc) to high value premium productions (like movies, trailers, webseries).

The challenges in the design of an MIR solution, by following the lifecycle of multimedia content is as follows:

- Entrance into the system (**Acquisition)**
- Preparation of analysis (**Normalization)**
- Extraction of metadata for building a search engine
- Constructing indexes **(Indexing)**
- Processing of a query (**Querying**)
- Presentation of results **(Browsing)**

Challenges

Content Acquisition

- Multimedia content can be acquired in a way similar to document acquisition:
  - By crawling the Web or local media repositories.
  - By user's contribution or syndicated contribution from content aggregators.
    - Directly from production devices directly connected to the system

- Just like different sources, the size of media files also varies make the content ingestion task more complicated, e.g., because the probability of download failures increases, the cost of storing duplicates or near duplicates becomes less affordable, and the presence of DRM issues on the downloaded content is more frequent.

Challenges

- The ability of the content ingestion subsystem to maintain or even improve the intrinsic quality of the downloaded digital assets—for example, by obtaining them at the best resolution feasible given the bandwidth constraints and preserving all the available metadata associated with them—is crucial.

- Building scalable and intelligent content acquisition systems, which could ingest content exploiting different communication protocols and acquisition devices, decide the optimal resolution in case alternative representations are available, detect and discard duplicates as early as possible, respect DRM issues, and enrich the raw media asset with the maximum amount of metadata that could be found inside or around it.

Challenges

PES UNIVERSITY

Content Normalization

- Multimedia content needs a more sophisticated preprocessing phase, because the elements to be indexed (called “features” or “annotations”) are numerical and textual metadata that need to be extracted from raw content by means of complex algorithms.
- Due to the variety of multimedia encoding formats, prior to processing content for metadata extraction, it is necessary to submit it to a normalization step, with a twofold purpose:
    - translating the source media items represented in different native formats into a common representation format. (e.g. MPEG4 for video files)
    - producing alternative variants of native content items, e.g., to provide freebies (free sample copies) of copyrighted elements or low resolution copies for distribution on mobile or low-bandwidth delivery channels. (e.g. making a 3GP version of video files)

Challenges

Content Indexing and Analysis

- After the normalization step, a multimedia collection has to be processed in order to make the knowledge embedded in it available for querying, which requires building the internal indexes of the search engine.
- Indexes are a concise representation of the content of an object collection, constructed out of the features extracted from it; the features used to build the indexes must be both sufficiently representative of the content and compact to optimize storage and retrieval.
- Features are traditionally grouped into two categories:
  - **Low level features**: concisely describe physical or perceptual properties of a media element (e.g., the colour or edge histogram of an image).
  - **High level features**: domain concepts characterizing the content (e.g., extracted objects and their properties, geographical references, etc.).

Challenges

- As in text, where the retrieved keywords can be highlighted in the source document, also in MIR there is the need of locating the occurrences of matches between the user's query and the content. Such requirement implies that features must be extracted from a time continuous medium, and that the coordinates in space and time of their occurrence must be extracted as well
- Content analysis and indexing are the prominent research problem of MIR, as the quality of the search engine depends on the precision at which the extracted metadata describe the content of a media asset.

Challenges

PES UNIVERSITY

Content Querying

- The expression of the user's information need allows for alternative query representation formats and matching semantics.
- Textual: one or more keywords, to be matched against textual metadata extracted from multimedia content.
- Mono-media: a content sample in a single media (e.g., an image, a piece of audio) to be matched against an item of the same kind (e.g., query by music or image similarity, query by humming) or of a different medium (e.g., finding the movies whose soundtrack is similar to an input audio file).
- Multi-media: a content sample in a composite medium, e.g., a video file to be matched using audio similarity, image similarity, or a combination of both.

Challenges

- Another implication of non-textual queries is the need for the MIR architecture to coordinate query processing across multiple dedicated search engines.

- The grand challenge of MIR query processing is in part the same as for textual IR: retrieving the media objects more relevant to the user's query with high precision and recall. MIR adds the specific problem of content-based queries, which demand suitable architectures for analysing a query content sample on the fly and matching its features to those stored in the indexes

Challenges

PES UNIVERSITY

Content Browsing

- Unlike data retrieval queries (such as SQL or XPATH queries), IR queries are approximate and thus results are presented in order of relevance, and often in a number that exceeds the user's possibility of selection.
- The interface must also permit users to quickly inspect continuous media and locate the exact point where a match has occurred. This can be done in many ways, e.g., by means of annotated time bars that permit one to jump into a video where a match occurs, with VCR-like commands, and so on.
- The challenge of MIR interfaces is devising effective renditions (visual, but also aural) that could convey both the global characteristics of the result set (e.g., the similarity distribution across a result collection) and the local features of an individual result item that justify the query match.

THANK YOU

Surabhi Narayan

Department of Computer Science Engineering

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Content Based Visual Information Retrieval

Surabhi Narayan
Department of Computer Science Engineering

Slides collated by:
Gaurav Mahajan (TA VIII sem)
Anshula Aithal (TA VIII sem)

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

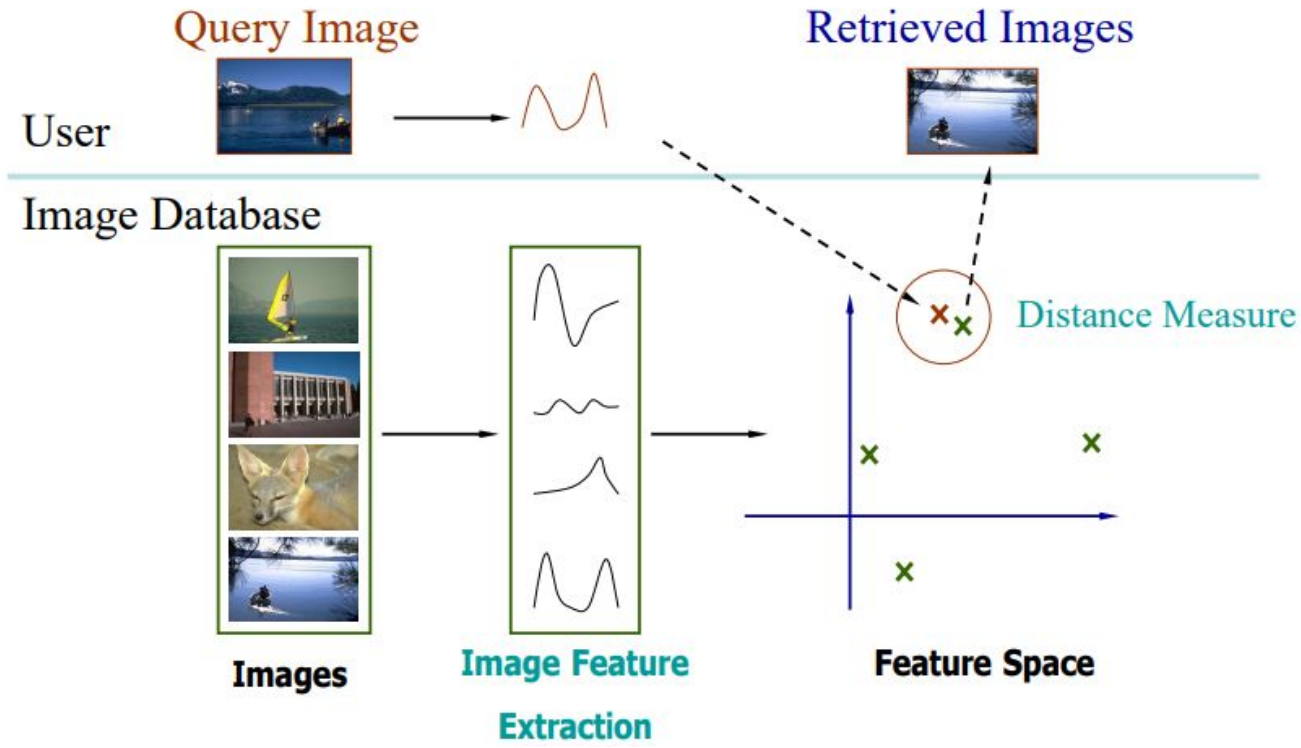
Content comparison using image distance measures

Surabhi Narayan
Department of Computer Science Engineering

Content Comparison using Image Distance Measures

Instead of exact matching, content-based image retrieval calculates visual similarities between a query image and images in a database. Accordingly, the retrieval result is not a single image but a list of images ranked by their similarities with the query image.

Many similarity measures have been developed for image retrieval based on empirical estimates of the distribution of features in recent years. Different similarity/distance measures will affect retrieval performances of an image retrieval system significantly.



Minkowski-form Distance

Minkowski-form is the most widely used distance measure due to its computational simplicity. It is also known as L_P norm; where P represents the order of distance. Consider two vectors M and N in $\mathbb{R}^k$; where k is the dimensionality of Euclidean space, then Minkowski-form distance L_P is:

$$L_P(M,N) = \left(\sum_{i=1}^{k} |m_i - n_i|^P\right)^{1/P}$$

Minkowski-form distance of order $P \geq 1$ is a metric. On the contrary, it is a non-metric for $P < 1$ due to the violation of triangle inequality. The most commonly used distances in this family are L_1 distance and L_2 distance, also known as Manhattan distance and Euclidean distance, respectively.

Euclidean Distance

Euclidean distance is a straight-line distance between two corresponding elements of a feature vector. It is a bin-by-bin distance, also called as Pythagoras distance.

$$L_2(M,N) = \sqrt{\left(\sum_{i=1}^{k} |m_i - n_i|^2\right)}$$

Additionally, it is the only Lp distance that is invariant to orthogonal transformation. Euclidean distance is applicable in various areas of image processing such as clustering, classification, retrieval, and also in distance transformation or mapping.

Distance transform is a process of converting a binary digital image into another image using some distance function. Each pixel in converted image has a value corresponding to the distance to the nearest pixel in image.

Euclidean Distance

Applications that use Euclidean distance:

MARS system used Euclidean distance to compute the similarity between **texture** features

Netra used Euclidean distance for **color** and **shape** feature, and

Blobworld used Euclidean distance for **texture** and **shape** feature

Manhattan Distance

Apart from Euclidean distance, another popular distance metric of L_P family is Manhattan distance, also known as taxi-cab distance or city block distance.

$$L_1(M,N) = \sum_{i=1}^{k} |m_i - n_i|$$

It is widely used in many image retrieval and color indexing-based approaches especially for texture features.

Chebyshev Distance

Lastly, another useful member of L_P distance family is L_∞ distance, also known as Chebyshev distance or chessboard distance. Chebyshev metric computes the distance between two feature vectors by taking the maximum of their element differences along any coordinate dimension. In two-dimensional space, it is equivalent to the minimum number of moves a king needed to travel between any two squares on the chessboard.

$$L_\infty(M,N) = \max_{i=1}^{k}(|m_i - n_i|) = \lim_{P\to\infty}\left(\sum_{i=1}^{k}|m_i - n_i|^P\right)^{1/P}$$

L∞ distance to compute the similarity between texture images

Chi-square Statistic

Chi-square statistic is a weighted Euclidean distance which is used to measure the corresponding elements of two feature vectors. This distance measure derived its name from chi-square test statistics which is used to test fitness between a distribution and observed frequencies. It is defined as:

$$\chi^2 = \frac{1}{2}\sum_{i=1}^{k}\frac{(m_i - n_i)^2}{m_i + n_i}$$

It is widely used to compute the difference between two histograms where the difference between the large bins is less important than the difference between the small bins

Histogram Intersection Distance

PES UNIVERSITY

Histogram intersection distance was originally proposed by for comparison of color histograms. This approach is robust to occlusion, image resolution, variation, varying viewpoint, and distraction in the background of the object. It is defined as:

$$\text{Histogram intersection distance } D_{\cap}(M,S) = \sum_{i=1}^{n} \min(M_i, S_i)$$

where M and S are two histograms with n bins. The outcome of histogram intersection is the number of pixels from the model image that has corresponding pixels of the same color in the sample image. The intersection result can be normalized by dividing the distance value by number of pixel in model histogram. Then, the distance value is:

$$D_{\cap}(M,S) = 1 - \frac{\sum_{i=1}^{n} \min(M_i, S_i)}{\sum_{i=1}^{n} M_i}$$

Mahalanobis Distance

Mahalanobis distance is used to compute the distance between a given feature vector and a distribution.

$$D_M(v) = \sqrt{(v-m)^T C^{-1}(v-m)}$$

where v is the feature vector of form $v = (v_1, v_2, v_3, \ldots v_k)$, m represents the mean row vector, C is the covariance matrix and T indicates the transpose operation. It is unit-less and scale-invariant distance metric and takes into account the correlations of the dataset.

However, for certain high-dimensional data, the computation of Mahalanobis distance is quite expensive due to the covariance matrix calculation.

Quadratic Form Distance

Quadratic form (QF) distance is used in color-based image retrieval.

two N-dimensional distributions $x, y \in \mathbb{R}^n$, quadratic form distance is defined as:

$$\mathrm{QF}(\mathrm{x}, \mathrm{y}) = \sqrt{(x-y)^{\mathrm{T}} A (x-y)}$$

where A is bin similarity matrix that stores the cross-bin information in form of matrix elements a_{ij}. Each element in similarity matrix tries to capture the perceptual similarity between the features represented by bins i and j. Computation of a_{ij} usually depends on the ground distance (d_{ij}). One such interpretation of a_{ij} is:

$$a_{ij} = 1 - \frac{d_{ij}}{d_{\max}}$$

d_{ij} could be the Euclidean distance between bin i and j, and $d_{\max} = \max_{ij}(d_{ij})$. Generally, the QF distance is not a metric, but for certain choice of similarity matrix A, it is indeed a metric.

Quadratic Form Distance

Quadratic form distance has been used in many retrieval systems for color histogram-based image retrieval. It has been shown that quadratic form distance can lead to perceptually more desirable results than Euclidean distance and histogram intersection method as it considers the cross-similarity between colors.

CBVIR - Examples

PES UNIVERSITY

QBIC

QBIC (Query By Image Content) was developed by IBM Almaden Research Center. Its framework and techniques have influenced many later systems. QBIC supports queries based on example images, user-constructed sketches, and selected colors and texture patterns. In its most recent version, it allows text-based keyword search to be combined with content based similarity search. The online QBIC demo can be found at: http://wwwqbic.almaden.ibm.com

CBVIR - Examples

PES UNIVERSITY

Photobook

Photobook is a set of interactive tools for browsing and searching images developed at MIT Media Lab. Photobook consists of three sub-books, from which shape, texture, and face features are extracted respectively. Users can query the system based on features from each of the three sub-blocks. Additional information about Photobook can be found at: http://www.white.media.mit.edu/vismod/demos/photobook/index.html.

CBVIR - Examples

PES UNIVERSITY

Netra

Netra is a prototype CBVIR system developed in the UCSB Alexandria Digital Library (ADL) project. It uses color, shape, texture, and spatial location information in the segmented image regions to search and retrieve similar images from the database. An online demo is available at http://vivaldi.ece.ucsb.edu/Netra/.

A new version of Netra, Netra 2, which emphasizes the group's latest work on color image segmentation and local color feature, is available at http://maya.ece.ucsb.edu/Netra/index2.html.

CBVIR - Examples

MARS

MARS (Multimedia Analysis and Retrieval System) was originally developed at University of Illinois at Urbana-Champaign. The main focus of MARS is not on finding a single “best” feature representation, but rather on how to organize the various visual features into a meaningful retrieval architecture, which can dynamically adapt to different applications and different users. MARS formally proposes a relevance feedback architecture in Image Retrieval and integrates such technique at various levels during retrieval, including query vector refinement, automatic matching tool selection, and automatic feature adaptation. More information about MARS can be obtained at: http://www-db.ics.uci.edu/pages/ research/mars.shtml.

CBVIR - Examples

Blobworld

Blobworld is a CBVIR system developed at U.C. Berkeley. The program automatically segments an image into regions, which roughly correspond to object or parts of objects allowing users to query for photographs or images based on the objects they contain. Their approach is useful in finding specific objects and not, as they put it, “stuff” as most systems which concentrate only on “low level” features with little regard for the spatial organization of those features. It allows for both textual and content-based searching. This system is also useful in its feedback to the user, in that it shows the internal representation of the submitted image and the query results. Thus, unlike some of the other systems, which allow for color histogram similarity metrics, which can be adjusted, this can help the user understand why they are getting certain results.

References

- “Content-Based Image Retrieval : Ideas, Influences, and Current Trends” - Vipin Tyagi, Springer Nature Singapore Pte Ltd., 2017. - Chapter 1 & Chapter 4

THANK YOU

Surabhi Narayan

Department of Computer Science Engineering

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Lucene

Dr Chandrashekhar P Chavan

Department of Computer Science Engineering

Lucene

- Apache Lucene is a full-text search engine which can be used from various programming languages.
- Lucene is an open source Java based search library.
- It is used in Java based applications to add document search capability to any kind of application in a very simple and efficient way.
- Scalable and high-performance library

Lucene

- Widely used academic systems
  - Terrier (Java, U. Glasgow) http://terrier.org
  - Indri/Galago/Lemur (C++ (& Java), U. Mass & CMU)
  - Tail of others (Zettair, …)
- Widely used non-academic open source systems
  - **Lucene**
    - Things built on it: Solr, ElasticSearch
  - A few others (Xapian, …)

Lucene

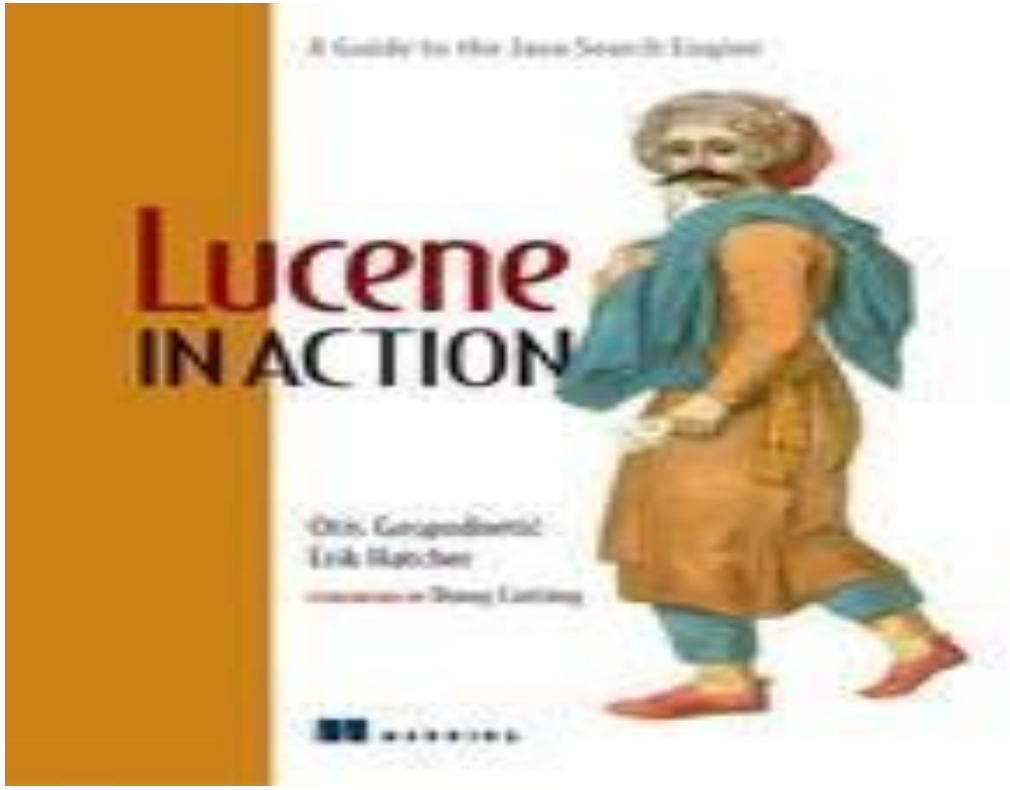
- Open source Java library for indexing and searching
  - Lets you add search to your application
  - Not a complete search system by itself
  - Written by Doug Cutting
- Used by: Twitter, LinkedIn, Zappos, CiteSeer, Eclipse, …
  - … and many more (see http://wiki.apache.org/lucene-java/PoweredBy)
- Ports/integrations to other languages
  - C/C++, C#, Ruby, Perl, Python, PHP, …

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Reference - Based on “Lucene in Action”

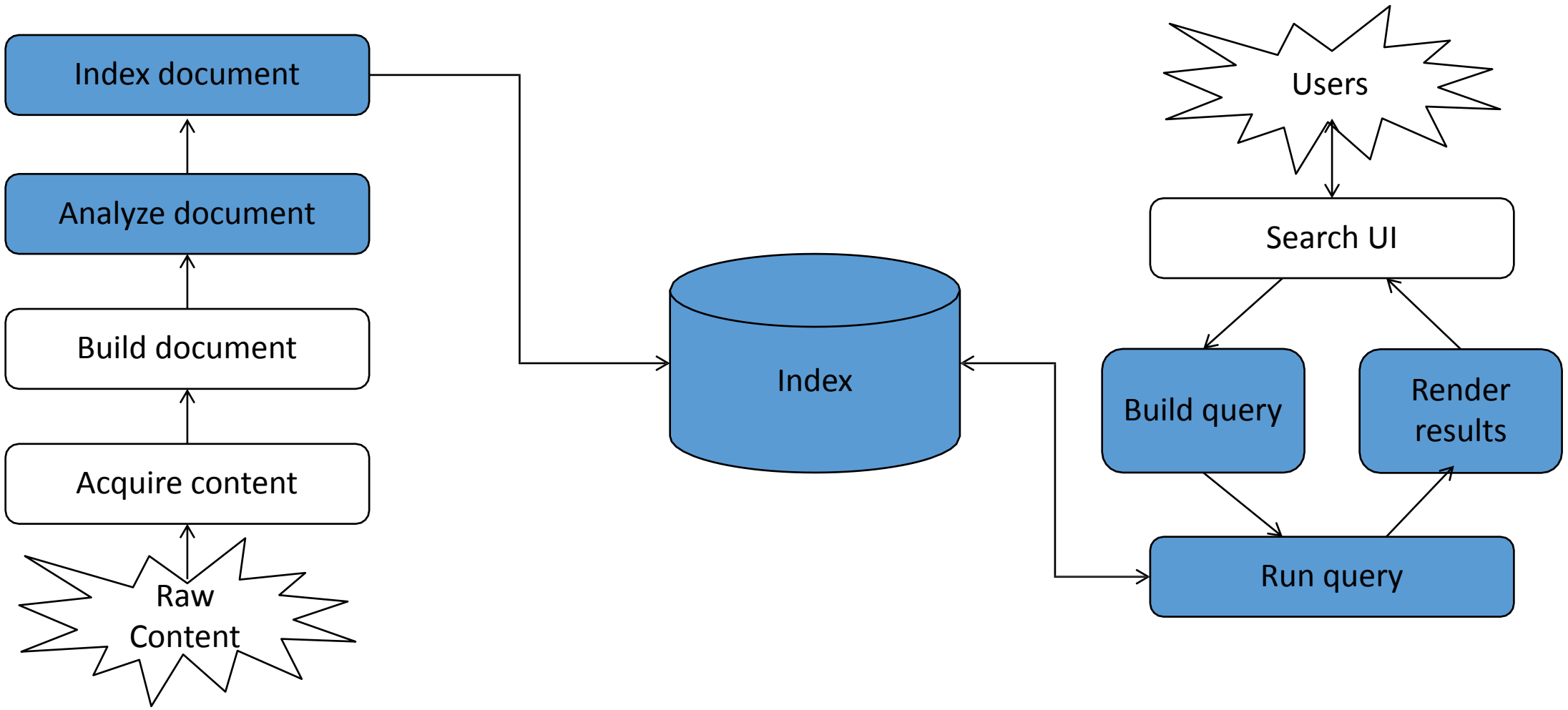
PES UNIVERSITY

By Michael McCandless, Erik Hatcher, Otis Gospodnetic



Covers Lucene 3.0.1. It’s now up to 5.1.0

Lucene in a search system



Lucene demos

- **Acquire Raw Content:** The first step of any search application is to collect the target contents on which search application is to be conducted.
- **Build the document**:build the document(s) from the raw content, which the search application can understand and interpret easily.
- **Analyze the document** :Before the indexing process starts, the document is to be analyzed as to which part of the text is a candidate to be indexed. This process is where the document is analyzed.
- **Indexing the document**: Once documents are built and analyzed, the next step is to index them so that this document can be retrieved based on certain keys instead of the entire content of the document.

Lucene demos

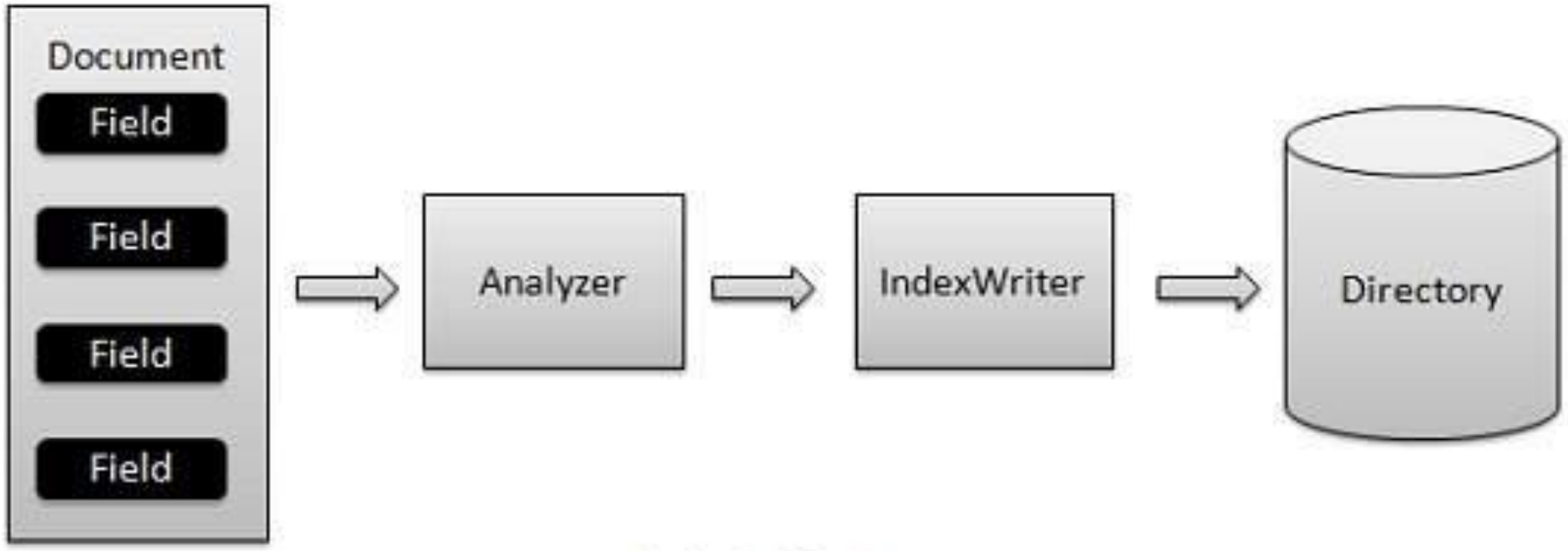
- **User Interface for Search:** Once a database of indexes is ready then the application can make any search. To facilitate a user to make a search, the application must provide a user **a mean** or **a user interface** where a user can enter text and start the search process.
- **Build Query**: Once a user makes a request to search a text, the application should prepare a Query object using that text which can be used to inquire index database to get the relevant details.
- **Search Query** : Using a query object, the index database is then checked to get the relevant details and the content documents.
- **Render Results :** Once the result is received, the application should decide on how to show the results to the user using User Interface. How much information is to be shown at first look and so on.

Lucene demos

- Source files in lia2e/src/lia/meetlucene/
  - Actual sources use Lucene 3.6.0
  - Code in these slides upgraded to Lucene 5.1.0
- Command line **Indexer**
  - lia.meetlucene.Indexer
- Command line **Searcher**
  - lia.meetlucene.Searcher

Core Indexing Classes

- Indexing process is one of the core functionalities provided by Lucene. The following diagram illustrates the indexing process and the use of classes. **IndexWriter** is the most important and the core component of the indexing process.



Indexing Process

Core Indexing Classes

- We add **Document(s)** containing **Field(s)** to IndexWriter which analyzes the **Document(s)** using the **Analyzer** and then creates/open/edit indexes as required and store/update them in a **Directory**. IndexWriter is used to update or create indexes. It is not used to read indexes.
- **IndexWriter :** This class acts as a core component which creates/updates indexes during the indexing process.

- **Directory :** This class represents the storage location of the indexes.
- **Analyzer :** This class is responsible to analyze a document and get the tokens/words from the text which is to be indexed. Without analysis done, IndexWriter cannot create index.

Core Indexing Classes

- **Document :** This class represents a virtual document with Fields where the Field is an object which can contain the physical document's contents, its meta data and so on. The Analyzer can understand a Document only.
- **Field :** This is the lowest unit or the starting point of the indexing process. It represents the key value pair relationship where a key is used to identify the value to be indexed. Let us assume a field used to represent contents of a document will have key as "contents" and the value may contain the part or all of the text or numeric content of the document. Lucene can index only text or numeric content only.

Core Indexing Classes

- IndexWriter
  - Central component that allows you to create a new index, open an existing one, and add, remove, or update documents in an index
  - Built on an IndexWriterConfig and a Directory
- Directory
  - Abstract class that represents the location of an index
- Analyzer
  - Extracts tokens from a text stream

Creating an IndexWriter

```
Import org.apache.lucene.analysis.Analyzer;
import org.apache.lucene.index.IndexWriter;
import org.apache.lucene.index.IndexWriterConfig;
import org.apache.lucene.store.Directory;
...

private IndexWriter writer;

public Indexer(String dir) throws IOException {
    Directory indexDir = FSDirectory.open(new File(dir));
    Analyzer analyzer = new StandardAnalyzer();
    IndexWriterConfig cfg = new IndexWriterConfig(analyzer);
    cfg.setOpenMode(OpenMode.CREATE);
    writer = new IndexWriter(indexDir, cfg)
}
```

Core indexing classes (contd.)

- Document
  - Represents a collection of named Fields. Text in these Fields are indexed.
- Field
  - Note: Lucene Fields can represent both “fields” and “zones” as described in the textbook
  - Or even other things like numbers.
  - StringFields are indexed but not tokenized
  - TextFields are indexed and tokenized

A Document contain Fields

```
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field;

...
protected Document getDocument(File f) throws Exception {
        Document doc = new Document();
        doc.add(new TextField("contents", new FileReader(f)))
  doc.add(new StringField("filename",
                f.getName(),
                Field.Store.YES));
  doc.add(new StringField("fullpath", f.getCanonicalPath(),Field.Store.YES));
  return doc;
}
```

Index a Document with IndexWriter

PES UNIVERSITY

```
private IndexWriter writer;
...
private void indexFile(File f) throws
                    Exception {
          Document doc = getDocument(f);
          writer.addDocument(doc);
}
```

Indexing a directory

```
private IndexWriter writer;
...
public int index(String dataDir, FileFilter filter)
                    throws Exception {
        File[] files = new File(dataDir).listFiles();
        for (File f: files) {
                if (... &&
                        (filter == null || filter.accept(f))) {
                    indexFile(f);
                }
        }
        return writer.numDocs();
}
```

Closing the IndexWriter


```
private IndexWriter writer;
...
public void close() throws IOException {
        writer.close();
}
```

The Index

- The Index is the kind of inverted index we know and love
- The default Lucene50 codec is:
  - variable-byte and fixed-width encoding of delta values
  - multi-level skip lists
  - natural ordering of docIDs
  - encodes both term frequencies and positional information
- APIs to customize the codec

Core searching classes

- The process of Searching is again one of the core functionalities provided by Lucene. Its flow is similar to that of the indexing process. Basic search of Lucene can be made using the following classes which can also be termed as foundation classes for all search related operations.
- **IndexSearcher :** This class act as a core component which reads/searches indexes created after the indexing process. It takes directory instance pointing to the location containing the indexes.
- **Term :** This class is the lowest unit of searching. It is similar to Field in indexing process.
- **Query :** Query is an abstract class and contains various utility methods and is the parent of all types of queries that Lucene uses during search process.

Core searching classes

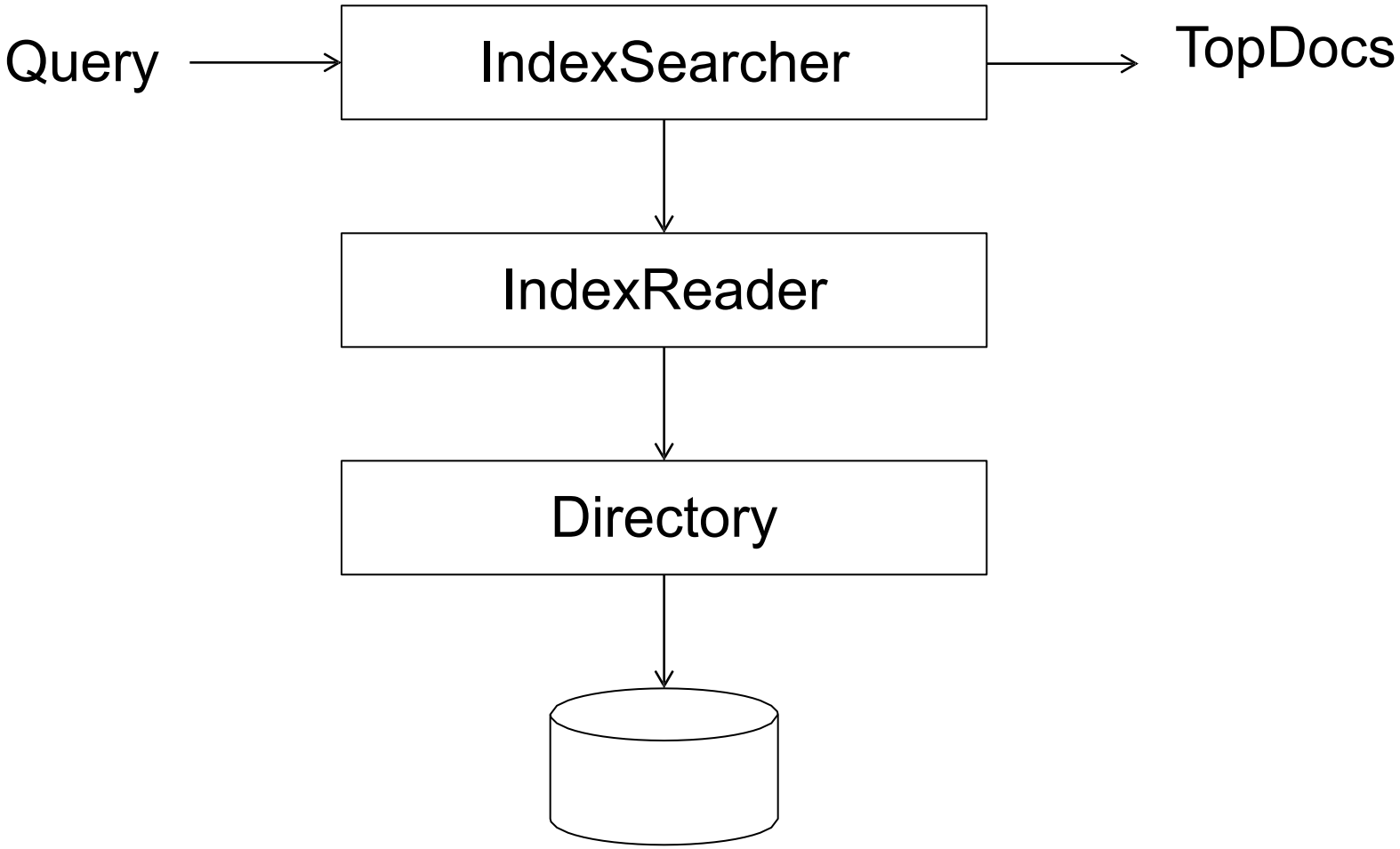
- **TermQuery :** TermQuery is the most commonly-used query object and is the foundation of many complex queries that Lucene can make use of.
- **TopDocs :** TopDocs points to the top N search results which matches the search criteria. It is a simple container of pointers to point to documents which are the output of a search result.

Core searching classes

PES UNIVERSITY

- IndexSearcher
  - Central class that exposes several search methods on an index
  - Accessed via an IndexReader
- Query
  - Abstract query class. Concrete subclasses represent specific types of queries, e.g., matching terms in fields, boolean queries, phrase queries, …
- QueryParser
  - Parses a textual representation of a query into a Query instance

IndexSearcher



Creating an IndexSearcher

PES UNIVERSITY

```
import org.apache.lucene.search.IndexSearcher;
...
public static void search(String indexDir, String q)  throws
                    IOException, ParseException {
        IndexReader rdr = DirectoryReader.open(FSDirectory.open(new File(indexDir)));
        IndexSearcher is = new IndexSearcher(rdr);
        ...
}
```

Query and QueryParser


```
import
org.apache.lucene.queryParser.QueryParser;
import org.apache.lucene.search.Query;

...
public static void search(String indexDir, String q)
                    throws IOException, ParseException
          ...
          QueryParser parser =
                    new QueryParser("contents”, new StandardAnalyzer());
          Query query = parser.parse(q);

          ...
}
```

Core searching classes (contd.)

PES UNIVERSITY

- TopDocs
  - Contains references to the top documents returned by a search
- ScoreDoc
  - Represents a single search result

search() returns TopDocs

PES UNIVERSITY

```
import org.apache.lucene.search.TopDocs;
...
public static void search(String indexDir,String q)
                          throws IOException, ParseException
        ...
        IndexSearcher is = ...;
        ...
        Query query = ...;
        ...
        TopDocs hits = is.search(query, 10);
}
```

TopDocs contain ScoreDocs

```
import org.apache.lucene.search.ScoreDoc;
...
public static void search(String indexDir, String q)
                throws IOException, ParseException
        ...
        IndexSearcher is = ...;
        ...
        TopDocs hits = ...;
        ...
        for(ScoreDoc scoreDoc : hits.scoreDocs) {
                Document doc = is.doc(scoreDoc.doc);
                System.out.println(doc.get("fullpath"));
        }
}
```

Closing IndexSearcher

```
public static void search(String indexDir, String q)
                    throws IOException, ParseException

        ...
        IndexSearcher is = ...;
        ...
        is.close();
}
```

How Lucene models content

- A Document is the atomic unit of indexing and searching
  - A Document contains Fields
- Fields have a name and a value
  - You have to translate raw content into Fields
  - Examples: Title, author, date, abstract, body, URL, keywords, ...
  - Different documents can have different fields
  - Search a field using name:term, e.g., title:lucene

Fields

- Fields may
  - Be indexed or not
    - Indexed fields may or may not be analyzed (i.e., tokenized with an Analyzer)
      - Non-analyzed fields view the entire value as a single token (useful for URLs, paths, dates, social security numbers, ...)
  - Be stored or not
    - Useful for fields that you'd like to display to users
  - Optionally store term vectors
    - Like a positional index on the Field's terms
    - Useful for highlighting, finding similar documents, categorization

Field construction - Lots of different constructors


```
import org.apache.lucene.document.Field
import
org.apache.lucene.document.FieldType

Field(String name,
                    String value,
                    FieldType type);
```

value can also be specified with a Reader, a TokenStream, or a byte[]. FieldType specifies field properties.

Can also directly use sub-classes like TextField, StringField, …

Using Field properties

PES UNIVERSITY

Index	Store	TermVector	Example usage
NOT_ANALYZED	YES	NO	Identifiers, telephone/SSNs, URLs, dates, ...
ANALYZED	YES	WITH_POSITIONS_OFFSETS	Title, abstract
ANALYZED	NO	WITH_POSITIONS_OFFSETS	Body
NO	YES	NO	Document type, DB keys (if not used for searching)
NOT_ANALYZED	NO	NO	Hidden keywords

Multi-valued fields

- You can add multiple Fields with the same name
  - Lucene simply concatenates the different values for that named Field

```
Document doc = new Document();
doc.add(new TextField("author","chris manning"));
doc.add(new TextField("author", "prabhakar raghavan"));
...
```

Analyzer

- Tokenizes the input text
- Common Analyzers
  - WhitespaceAnalyzer

    Splits tokens on whitespace
  - SimpleAnalyzer

    Splits tokens on non-letters, and then lowercases
  - StopAnalyzer

    Same as SimpleAnalyzer, but also removes stop words
  - StandardAnalyzer

    Most sophisticated analyzer that knows about certain token types, lowercases, removes stop words, ...

Analysis example

PES UNIVERSITY

- "The quick brown fox jumped over the lazy dog"
- WhitespaceAnalyzer
  - [The] [quick] [brown] [fox] [jumped] [over] [the] [lazy] [dog]
- SimpleAnalyzer
  - [the] [quick] [brown] [fox] [jumped] [over] [the] [lazy] [dog]
- StopAnalyzer
  - [quick] [brown] [fox] [jumped] [over] [lazy] [dog]
- StandardAnalyzer
  - [quick] [brown] [fox] [jumped] [over] [lazy] [dog]

Another analysis example

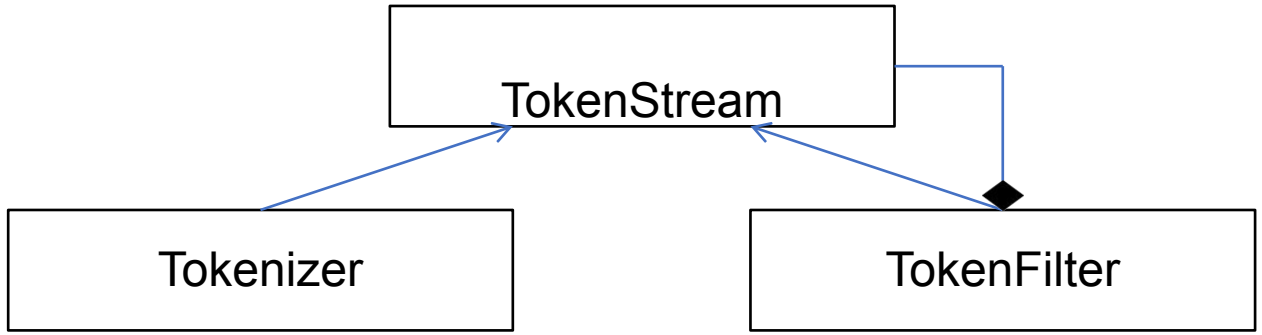
PES UNIVERSITY

- "XY&Z Corporation – xyz@example.com"
- WhitespaceAnalyzer
  - [XY&Z] [Corporation] [-] [xyz@example.com]
- SimpleAnalyzer
  - [xy] [z] [corporation] [xyz] [example] [com]
- StopAnalyzer
  - [xy] [z] [corporation] [xyz] [example] [com]
- StandardAnalyzer
  - [xy&z] [corporation] [xyz@example.com]

What's inside an Analyzer?

- Analyzers need to return a TokenStream

  public TokenStream tokenStream(String fieldName,Reader reader)



Tokenizers and TokenFilters

PES UNIVERSITY

- Tokenizer
  - WhitespaceTokenizer
  - KeywordTokenizer
  - LetterTokenizer
  - StandardTokenizer
  - ...

- TokenFilter
  - LowerCaseFilter
  - StopFilter
  - PorterStemFilter
  - ASCIIFoldingFilter
  - StandardFilter
  - ...

Adding/deleting Documents to/from an IndexWriter


```
void addDocument(Iterable<IndexableField> d);
```

IndexWriter's Analyzer is used to analyze document.

Important: Need to ensure that Analyzers used at indexing time are consistent with Analyzers used at searching time

```
// deletes docs containing terms or matching
// queries. The term version is useful for
// deleting one document.
void deleteDocuments(Term... terms);
void deleteDocuments(Query... queries);
```

Index format

- Each Lucene index consists of one or more segments
  - A segment is a standalone index for a subset of documents
  - All segments are searched
  - A segment is created whenever IndexWriter flushes adds/deletes
- Periodically, IndexWriter will merge a set of segments into a single segment
  - Policy specified by a MergePolicy
- You can explicitly invoke forceMerge() to merge segments

Basic merge policy

- Segments are grouped into levels
- Segments within a group are roughly equal size (in log space)
- Once a level has enough segments, they are merged into a segment at the next level up

Searching a changing index

```
Directory dir = FSDirectory.open(...);  DirectoryReader reader =
DirectoryReader.open(dir);
IndexSearcher searcher = new IndexSearcher(reader);
```

Above reader does not reflect changes to the index unless you reopen it. Reopening is more resource efficient than opening a brand new reader.

```
DirectoryReader newReader =
    DirectoryReader.openIfChanged(reader);
If (newReader != null) {
        reader.close();  reader = newReader;
        searcher = new IndexSearcher(reader);
}
```

Near-real-time search

```
IndexWriter writer = ...;
DirectoryReader reader =
    DirectoryReader.open(writer, true);
IndexSearcher searcher = new IndexSearcher(reader);

// Now let us say there's a change to the index using writer
writer.addDocument(newDoc);

DirectoryReader newReader =
    DirectoryReader.openIfChanged(reader, writer, true);
if (newReader != null) {
        reader.close();
        reader = newReader;
        searcher = new IndexSearcher(reader);
}
```

QueryParser

- Constructor
  - QueryParser(String defaultField, Analyzer analyzer);

- Parsing methods
- Query parse(String query) throws ParseException;
  - ... and many more

QueryParser syntax examples

Query expression	Document matches if…
java	Contains the term *java* in the default field
java junit java OR junit	Contains the term *java* or *junit* or both in the default field (*the default operator can be changed to* AND)
+java +junit java AND junit	Contains both *java* and *junit* in the default field
title:ant	Contains the term *ant* in the title field
title:extreme –subject:sports	Contains *extreme* in the title and not *sports* in subject
(agile OR extreme) AND java	Boolean expression matches
title:”junit in action”	Phrase matches in title
title:”junit action”~5	Proximity matches (within 5) in title
java*	Wildcard matches
java~	Fuzzy matches
lastmodified:[1/1/09 TO 12/31/09]	Range matches

Construct Queries programmatically

PES UNIVERSITY

- TermQuery
  - Constructed from a Term
- TermRangeQuery
- NumericRangeQuery
- PrefixQuery
- BooleanQuery
- PhraseQuery
- WildcardQuery
- FuzzyQuery
- MatchAllDocsQuery

IndexSearcher

PES UNIVERSITY

- Methods
  - TopDocs search(Query q, int n);
  - Document doc(int docID);

TopDocs and ScoreDoc

PES UNIVERSITY

- TopDocs methods
  - Number of documents that matched the search
    totalHits
  - Array of ScoreDoc instances containing results
    scoreDocs
  - Returns best score of all matches
    getMaxScore()
- ScoreDoc methods
  - Document id
    doc
  - Document score
    score

Scoring

- Original scoring function uses basic tf-idf scoring with
  - Programmable boost values for certain fields in documents
  - Length normalization
  - Boosts for documents containing more of the query terms
- IndexSearcher provides an explain() method that explains the scoring of a document

Lucene 5.0 Scoring

- As well as traditional tf.idf vector space model, Lucene 5.0 has:
  - BM25
  - drf (divergence from randomness)
  - ib (information (theory)-based similarity)

```
indexSearcher.setSimilarity(  new
        BM25Similarity());

BM25Similarity custom = new BM25Similarity(1.2, 0.75); // k1, b
indexSearcher.setSimilarity(custom);
```

Resources

- Lucene: http://lucene.apache.org
- Lucene in Action: http://www.manning.com/hatcher3/
  - Code samples available for download
- Ant: http://ant.apache.org/
  - Java build system used by “Lucene in Action” code

Implementation example - Importing packages and dependencies


```
package com.luceneapp;

import java.io.IOException;
import java.io.InputStream;
import java.nio.file.FileVisitResult;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.SimpleFileVisitor;
import java.nio.file.attribute.BasicFileAttributes;

import org.apache.lucene.analysis.Analyzer;
import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field;
import org.apache.lucene.document.Field.Store;
import org.apache.lucene.document.LongPoint;
import org.apache.lucene.document.StringField;
import org.apache.lucene.document.TextField;
import org.apache.lucene.index.IndexWriter;
import org.apache.lucene.index.IndexWriterConfig;
import org.apache.lucene.index.IndexWriterConfig.OpenMode;
import org.apache.lucene.index.Term;
import org.apache.lucene.store.Directory;
import org.apache.lucene.store.FSDirectory;
```

Implementation example - Creating Inverted Index

PES UNIVERSITY

```
public static void main(String[] args)
{
    //Input folder
    String docsPath = "inputFiles";

    //Output folder
    String indexPath = "indexedFiles";

    //Input Path Variable
    final Path docDir = Paths.get(docsPath);

    try
    {
        //org.apache.lucene.store.Directory instance
        Directory dir = FSDirectory.open( Paths.get(indexPath) );

        //analyzer with the default stop words
        Analyzer analyzer = new StandardAnalyzer();

        //IndexWriter Configuration
        IndexWriterConfig iwc = new IndexWriterConfig(analyzer);
        iwc.setOpenMode(OpenMode.CREATE_OR_APPEND);

        //IndexWriter writes new index files to the directory
        IndexWriter writer = new IndexWriter(dir, iwc);

        //Its recursive method to iterate all files and directories
        indexDocs(writer, docDir);

        writer.close();
    }
    catch (IOException e)
    {
        e.printStackTrace();
    }
}
```

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

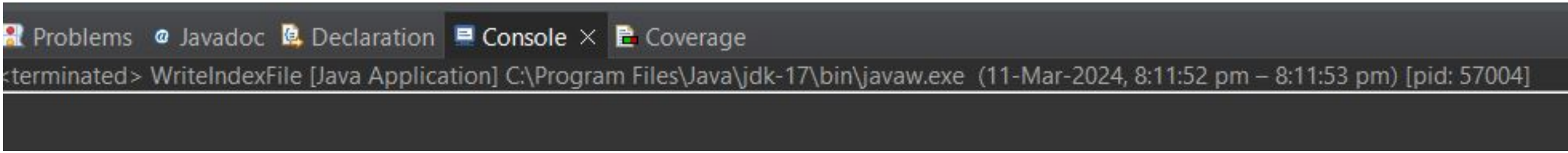
Implementation example - Creating Inverted Index

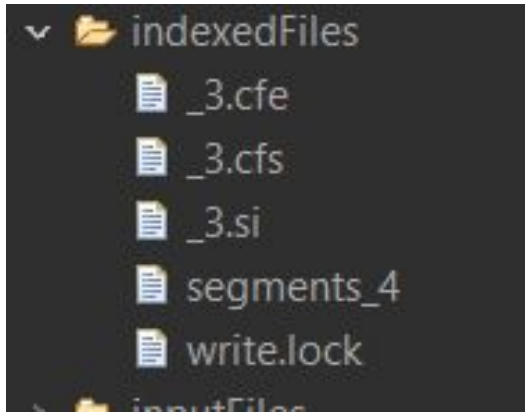
PES UNIVERSITY

```
static void indexDocs(final IndexWriter writer, Path path) throws IOException
{
    //Directory?
    if (Files.isDirectory(path))
    {
        //Iterate directory
        Files.walkFileTree(path, new SimpleFileVisitor<Path>()
        {
            @Override
            public FileVisitResult visitFile(Path file, BasicFileAttributes attrs) throws IOExcepti
            {
                try
                {
                    //Index this file
                    indexDoc(writer, file, attrs.lastModifiedTime().toMillis());
                }
                catch (IOException ioe)
                {
                    ioe.printStackTrace();
                }
                return FileVisitResult.CONTINUE;
            }
        });
    }
    else
    {
        //Index this file
        indexDoc(writer, path, Files.getLastModifiedTime(path).toMillis());
    }
}
```

```
    static void indexDoc(IndexWriter writer, Path file, long lastModified) throws IOException
    {
        try (InputStream stream = Files.newInputStream(file))
        {
            //Create lucene Document
            Document doc = new Document();

            doc.add(new StringField("path", file.toString(), Field.Store.YES));
            doc.add(new LongPoint("modified", lastModified));
            doc.add(new TextField("contents", new String(Files.readAllBytes(file)), Store.YES));

            //Updates a document by first deleting the document(s)
            //containing <code>term</code> and then adding the new
            //document.  The delete and then add are atomic as seen
            //by a reader on the same index
            writer.updateDocument(new Term("path", file.toString()), doc);
        }
    }
}
```

Implementation example - Creating Inverted Index





.cfe = Compound file entry table
.cfs = Compound file
.si = Segment info

Implementation example - Searching

PES UNIVERSITY

```java
package com.luceneapp;

import java.io.IOException;
import java.nio.file.Paths;

import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.index.DirectoryReader;
import org.apache.lucene.index.IndexReader;
import org.apache.lucene.queryparser.classic.QueryParser;
import org.apache.lucene.search.IndexSearcher;
import org.apache.lucene.search.Query;
import org.apache.lucene.search.ScoreDoc;
import org.apache.lucene.search.TopDocs;
import org.apache.lucene.store.Directory;
import org.apache.lucene.store.FSDirectory;
```

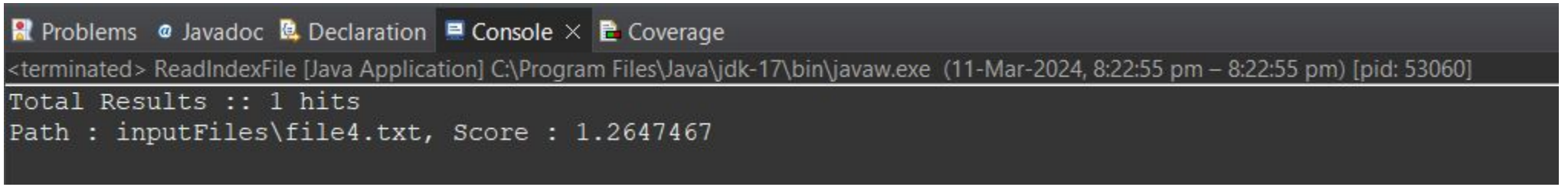
```java
public class ReadIndexFile
{
        //directory contains the lucene indexes
    private static final String INDEX_DIR = "indexedFiles";

    public static void main(String[] args) throws Exception
    {
        //Create lucene searcher. It search over a single IndexReader.
        IndexSearcher searcher = createSearcher();

        //Search indexed contents using search term
        TopDocs foundDocs = searchInContent("India", searcher);

        //Total found documents
        System.out.println("Total Results :: " + foundDocs.totalHits);

        //Let's print out the path of files which have searched term
        for (ScoreDoc sd : foundDocs.scoreDocs)
        {
            Document d = searcher.doc(sd.doc);
            System.out.println("Path : "+ d.get("path") + ", Score : " + sd.score);
        }
     }
```

Implementation example - Searching

```
private static TopDocs searchInContent(String textToFind, IndexSearcher searcher) throws Exception
{
    //Create search query
    QueryParser qp = new QueryParser("contents", new StandardAnalyzer());
    Query query = qp.parse(textToFind);

    //search the index
    TopDocs hits = searcher.search(query, 10);
    return hits;
}
```

```
    private static IndexSearcher createSearcher() throws IOException
    {
        Directory dir = FSDirectory.open(Paths.get(INDEX_DIR));

        //It is an interface for accessing a point-in-time view of a lucene index
        IndexReader reader = DirectoryReader.open(dir);

        //Index searcher
        IndexSearcher searcher = new IndexSearcher(reader);
        return searcher;
    }
}
```

Implementation example - Results



THANK YOU

Dr. Chandrashekhar P Chavan
Department of Computer Science and Engineering
cpchavan@pes.edu
+91 9844556230, 9110291866

PES UNIVERSITY

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Content Based Visual Information Retrieval

Surabhi Narayan
Department of Computer Science Engineering

Slides collated by:
Gaurav Mahajan (TA VIII sem)
Anshula Aithal (TA VIII sem)

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Query by Example

Surabhi Narayan

Department of Computer Science Engineering

User Perspective

- The user interface is a crucial component of a CBVIR system. Ideally such interface should be simple, easy, friendly, functional, and customizable.
- It should provide integrated browsing, viewing, searching, and querying capabilities in a clear and intuitive way.
- This integration is extremely important, since it is very likely that the user will not always stick to the best match found by the query/search engine.
- More often than not users will want to check the first few best matches, browse through them, preview their contents, refine their query, and eventually retrieve the desired image or video segment.

User Perspective

- Specifying what kind of images a user wishes to retrieve from the database can be done in many ways.
- Several querying mechanisms have been created to help users define their information need.
- Commonly used query formations are as follows:
  - Category Browsing
  - Query by Concept
  - Query by Sketch
  - **Query by Example**

Category Browsing, Query by Concept and Query by Sketch

Category browsing: Category browsing is to browse through the database according to the category of the image.

Query by Concept: Query by concept is to retrieve images according to the conceptual description associated with each image in the database.

Query by Sketch: Query by sketch allows user to draw a sketch of an image with a graphic editing tool provided either by the retrieval system or by some other software. Queries may be formed by drawing several objects with certain properties like color, texture, shape, sizes, and locations. In most cases, a coarse sketch is sufficient, as the query can be refined based on retrieval results.

Query by Example

- Query by example allows the user to formulate a query by providing an example image.
- The system converts the example image into an internal representation of features.
- Images stored in the database with similar features are then searched.

Query by Example

- Query by example can be further classified into:
  Query by external image example, if the query image is not in the database, and
  Query by internal image example, if otherwise.

- For query by internal image, all relationships between images can be pre-computed.

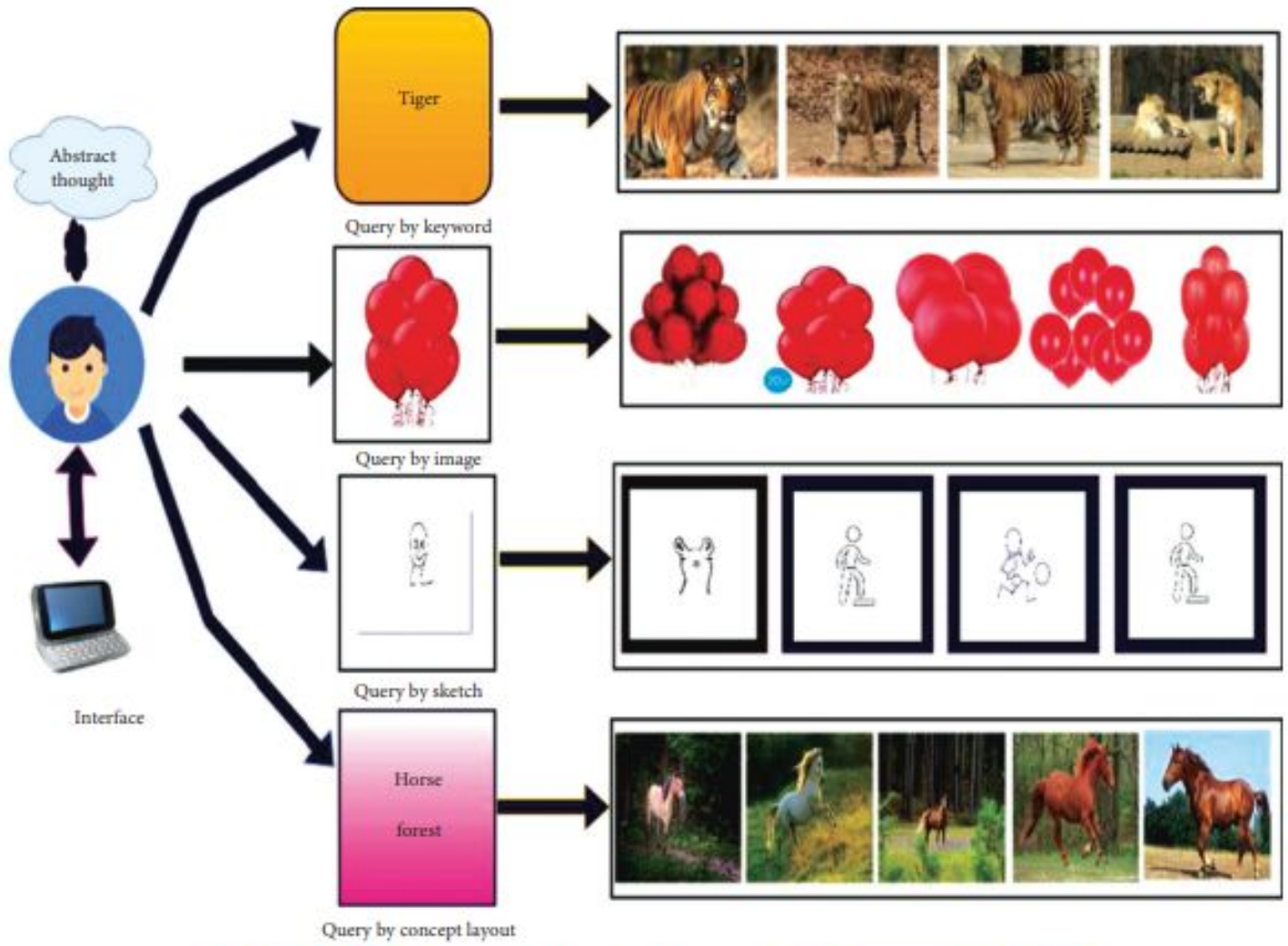
Query by Example

- The main advantage of query by example is that the user is not required to provide an explicit description of the target, which is instead computed by the system.
- It is suitable for applications where the target is an image of the same object or set of objects under different viewing conditions.
- Most of the current systems provide this form of querying.

Query by Example

- Another modification that can be made is by taking a group of examples – **Query by Group Example**
- Query by group example allows user to select multiple images.
- The system will then find the images that best match the common characteristics of the group of examples.
- In this way, a target can be defined more precisely by specifying the relevant feature variations and removing irrelevant variations in the query.
- In addition, group properties can be refined by adding negative examples. Many recently developed systems provide both query by positive and negative examples

Query by Example



References

- “Content-Based Image Retrieval and Feature Extraction: A Comprehensive Review” - Afshan Latif

THANK YOU

Surabhi Narayan

Department of Computer Science Engineering

PES UNIVERSITY

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Content Based Visual Information Retrieval

Surabhi Narayan
Department of Computer Science Engineering

Slides collated by:
Gaurav Mahajan (TA VIII sem)
Anshula Aithal (TA VIII sem)

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Content based Visual Information Retrieval – Image Feature Basics

Surabhi Narayan

Department of Computer Science Engineering

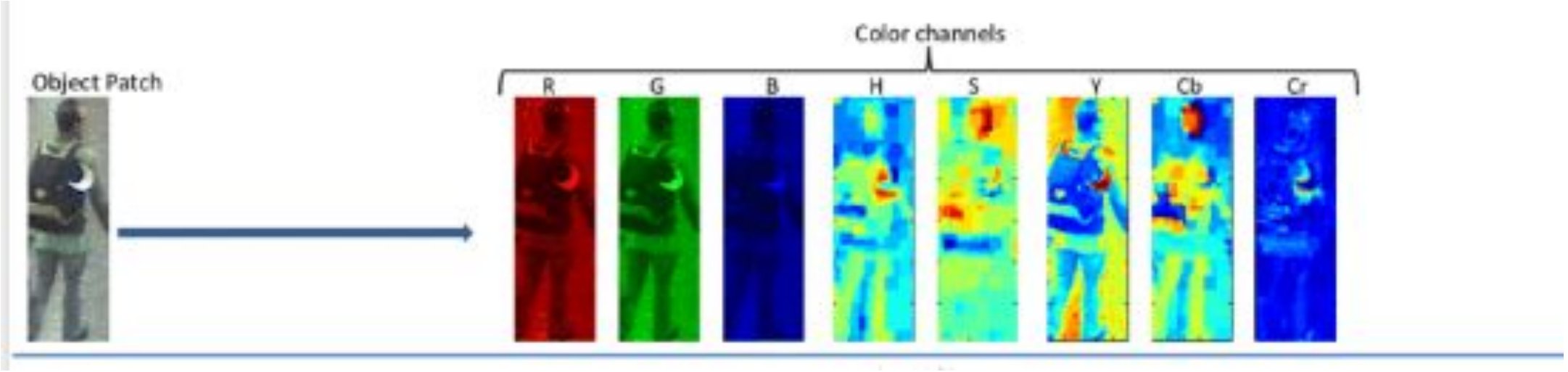
Image Features

CBVIR systems should be able to automatically extract visual features that are used to describe the contents of an image or video clip. Examples of such features include color, texture, size, shape, and motion information.

We will keep our discussion limited to features such as Color, Texture, Shape and Spatial Information.

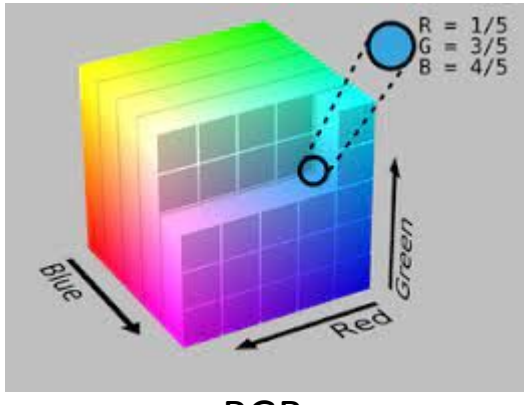
Color Feature

- Color is the most extensively used visual content for image retrieval.
- Its three-dimensional values make its discrimination potentiality superior to the single-dimensional gray values of images.
- Before selecting an appropriate color description, color space must be determined first

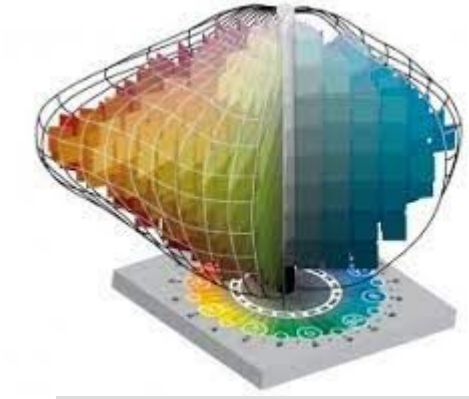


Color Feature

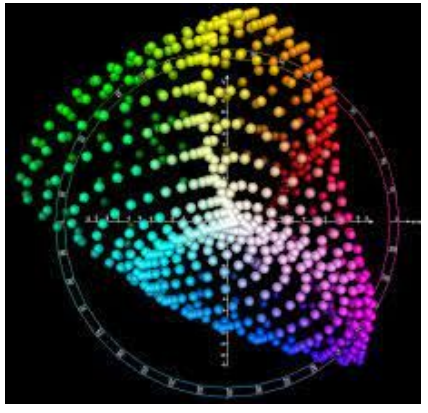
- **Color space**:
  - Each pixel of the image can be represented as a point in a 3D color space.
  - Commonly used color space for image retrieval includes RGB, Munsell, CIE L*a*b*, CIE L*u*v*, HSV (or HSL, HSB), and opponent color space.



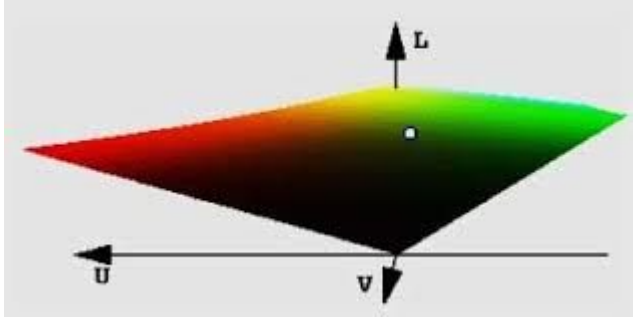
RGB



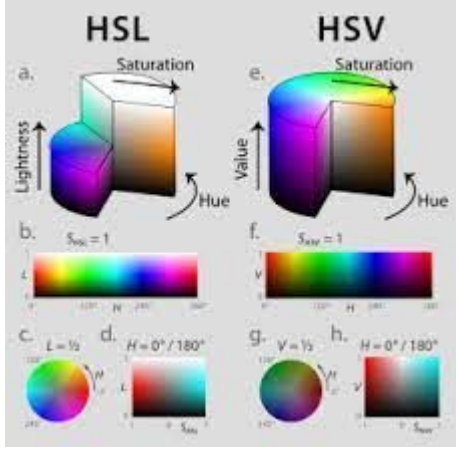
Munsell

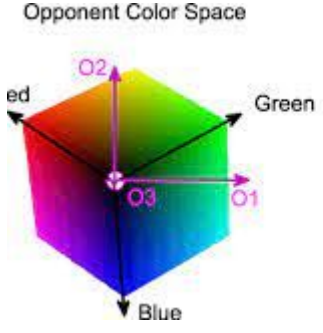


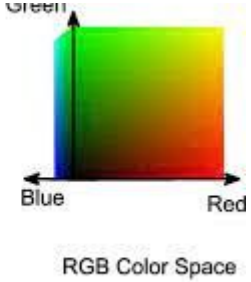
CIE L*a*b*



CIE L*u*v*







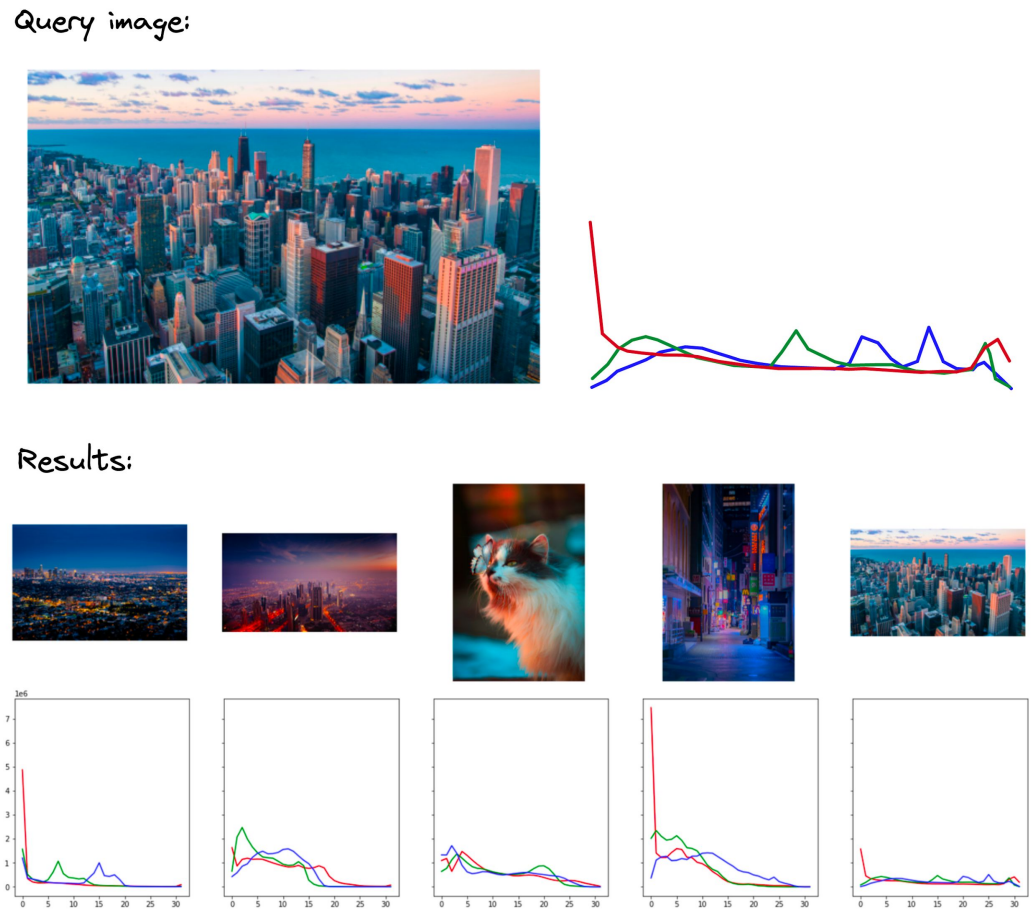
opponent color space.

Color Feature

- **Color Descriptors:**
  - A color descriptor is a generalization of a color histogram that captures some spatial characteristics of the color distribution in an image. Color descriptors are essential for accessing reliable visual information as illumination variation is inevitable in many practical cases.
  - Some commonly used color descriptors are as follows: the color moments, color histogram, color coherence vector, and color correlogram.

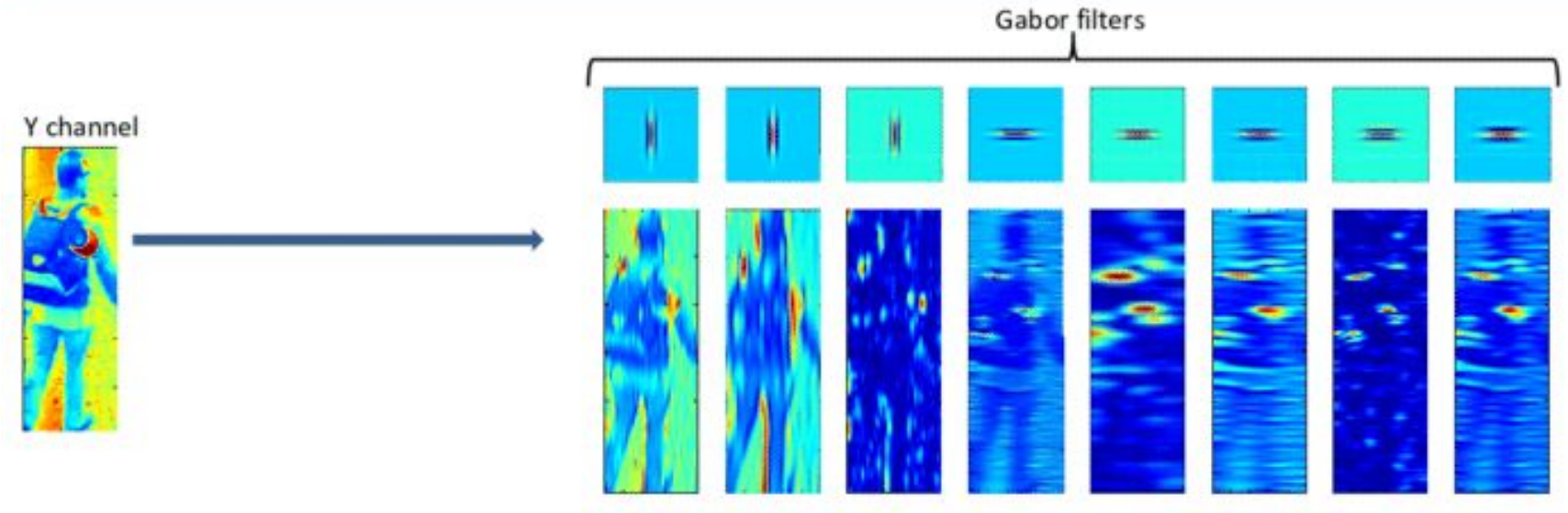
Color Feature

- **Color Descriptors:**
  - **Color Moments:** Color moments have been successfully used in many retrieval systems like QBIC, especially when the image contains only objects.
  - **Color Histogram:** The color histogram serves as an effective representation of the color content of an image if the color pattern is unique compared with the rest of the dataset. The color histogram is easy to compute and effective in characterizing both the global and local distributions of colors in an image



Texture Feature

- Texture is another important property of images.
- Various texture representations have been investigated in pattern recognition and computer vision.

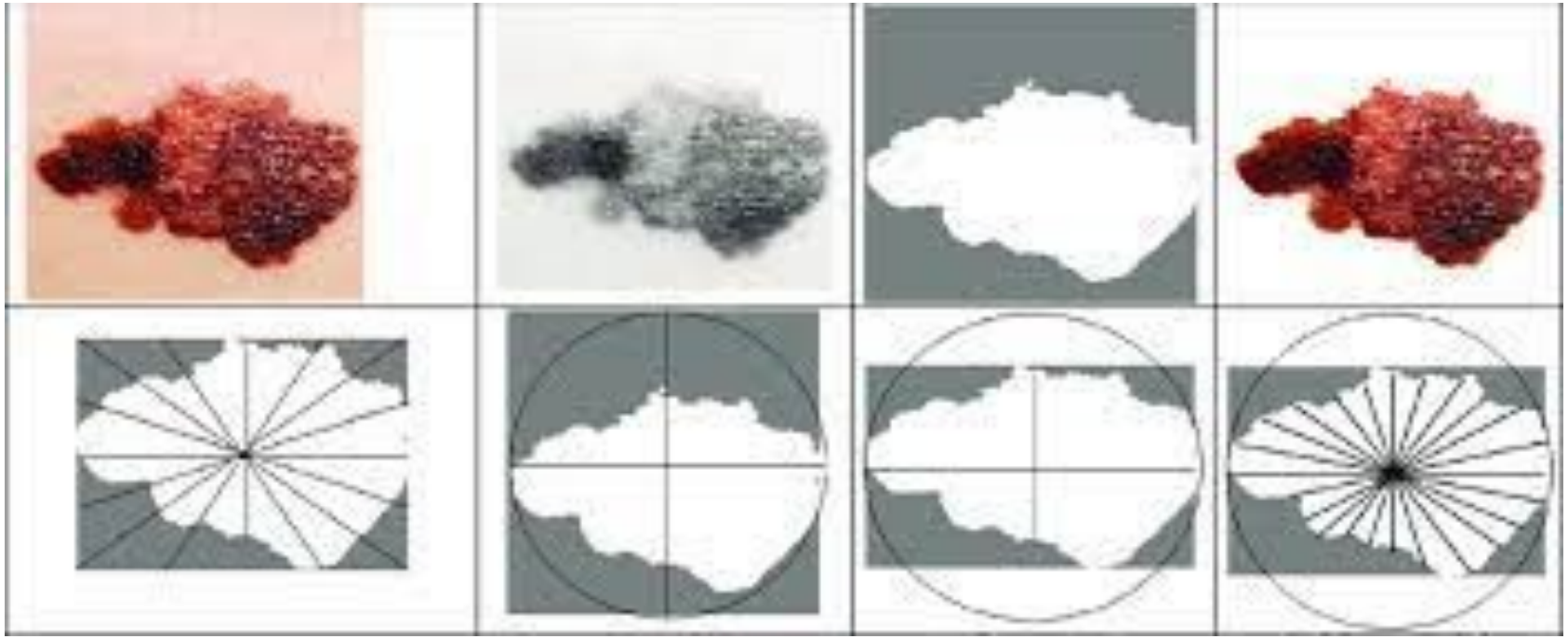


Texture Feature

- Basically, texture representation methods can be classified into two categories: structural and statistical.
  - Structural methods, including morphological operator and adjacency graph, describe texture by identifying structural primitives and their placement rules. They tend to be most effective when applied to textures that are very regular.
  - Statistical methods, including Fourier power spectra, cooccurrence matrices, shift-invariant principal component analysis (SPCA), Tamura feature, Wold decomposition, Markov random field, fractal model, and multiresolution filtering techniques such as Gabor and wavelet transform, characterize texture by the statistical distribution of the image intensity

Shape Feature

- Shape features of objects or regions have been used in many content-based image retrieval systems.
- Compared with color and texture features, shape features are usually described after images have been segmented into regions or objects.



Shape Feature

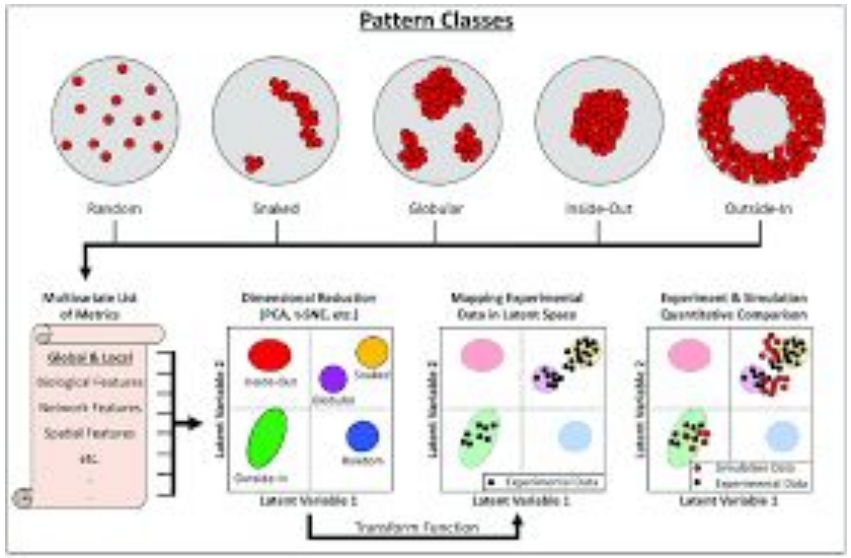
- Since robust and accurate image segmentation is difficult to achieve, the use of shape features for image retrieval has been limited to special applications where objects or regions are readily available.

- The state-of-the-art methods for shape description can be categorized into either boundary-based (e.g., rectilinear shapes, polygonal approximation, finite element models, and Fourier-based shape descriptors) or region-based methods (e.g., statistical moments).

Spatial Information Feature

- Regions or objects with similar color and texture properties can be easily distinguished by imposing spatial constraints.

- For instance, regions of blue sky and ocean may have similar color histograms, but their spatial locations in images are different. Therefore, the spatial location of regions (or objects) or the spatial relationship between multiple regions (or objects) in an image is very useful for searching images.



References

- “Content-Based Image Retrieval : Ideas, Influences, and Current Trends” - Vipin Tyagi, Springer Nature Singapore Pte Ltd., 2017.

PES UNIVERSITY

THANK YOU

Surabhi Narayan

Department of Computer Science Engineering

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Crawler Overview

Bhaskarjyoti Das
Department of Computer Science Engineering

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Crawler Basics

Bhaskarjyoti Das
Department of Computer Science Engineering

Basic Crawler operation

- Initialize queue with URLs of known seed pages
- Repeat
  - Take URL from queue
  - Fetch and parse page
  - Extract URLs from page
  - Add URLs to queue
- Fundamental assumption: The web is well linked.
- If this is implemented as Queue, what sort of traversal of the web graph does it represent ?
  - Roughly (since it is initiated from several seeds) it is equivalent to Breadth First Search since it is FIFO based (deleting from front and inserting at the back)

What's wrong with this crawler

```
urlqueue := (some carefully selected set of seed urls)  while
urlqueue is not empty:
    myurl := urlqueue.getlastanddelete()  mypage :=
    myurl.fetch()  fetchedurls.add(myurl)
    newurls := mypage.extracturls()  for myurl in
    newurls:
        if myurl not in fetchedurls and not in urlqueue:
            urlqueue.add(myurl)  addtoinvertedindex(mypage)
```

What is wrong with the simple crawler

- Scale: we need to distribute.
- We can't index everything: we need to subselect. How?
- Duplicates: need to integrate duplicate detection
- Spam and spider traps: need to integrate spam detection
- Politeness: we need to be "nice" and space out all requests for a site over a longer period (hours, days)
- Freshness: we need to recrawl periodically.
  - Because of the size of the web, we can do frequent recrawls only for a small subset.
  - Again, subselection problem or prioritization

Magnitude of the crawling problem

- To fetch 20,000,000,000 pages in one month . . .
- . . . we need to fetch almost 8000 pages per second!
- Actually: many more since many of the pages we attempt to crawl will be duplicates, unfetchable, spam etc.

ALGORITHMS FOR INFORMATION RETRIEVAL

Crawler Operations

Bhaskarjyoti Das
Department of Computer Science Engineering

Basic Crawler operation

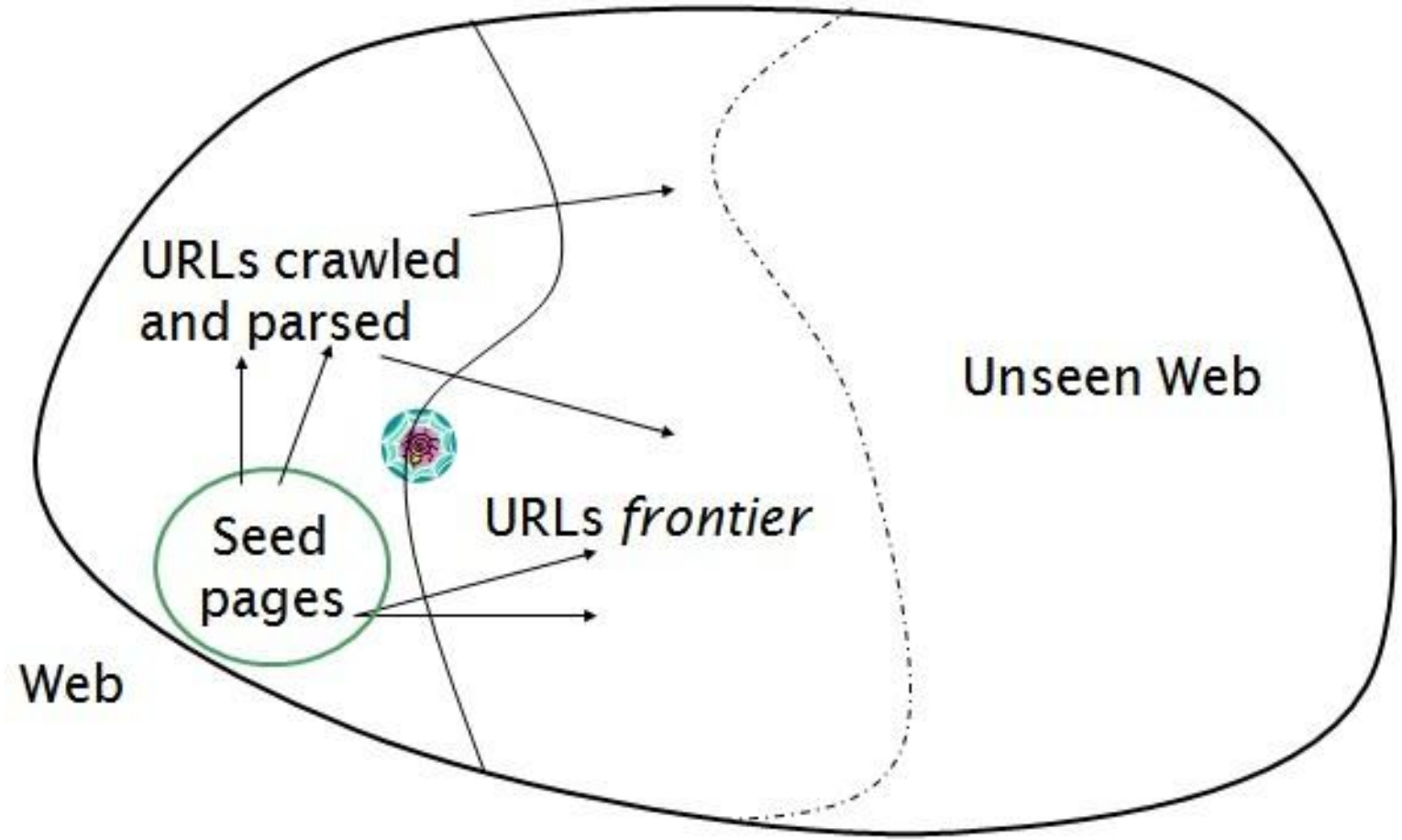
PES UNIVERSITY

- Crawler also called Robot and Spider

- Job of the Crawler is to get the web documents that the indexer indexes.

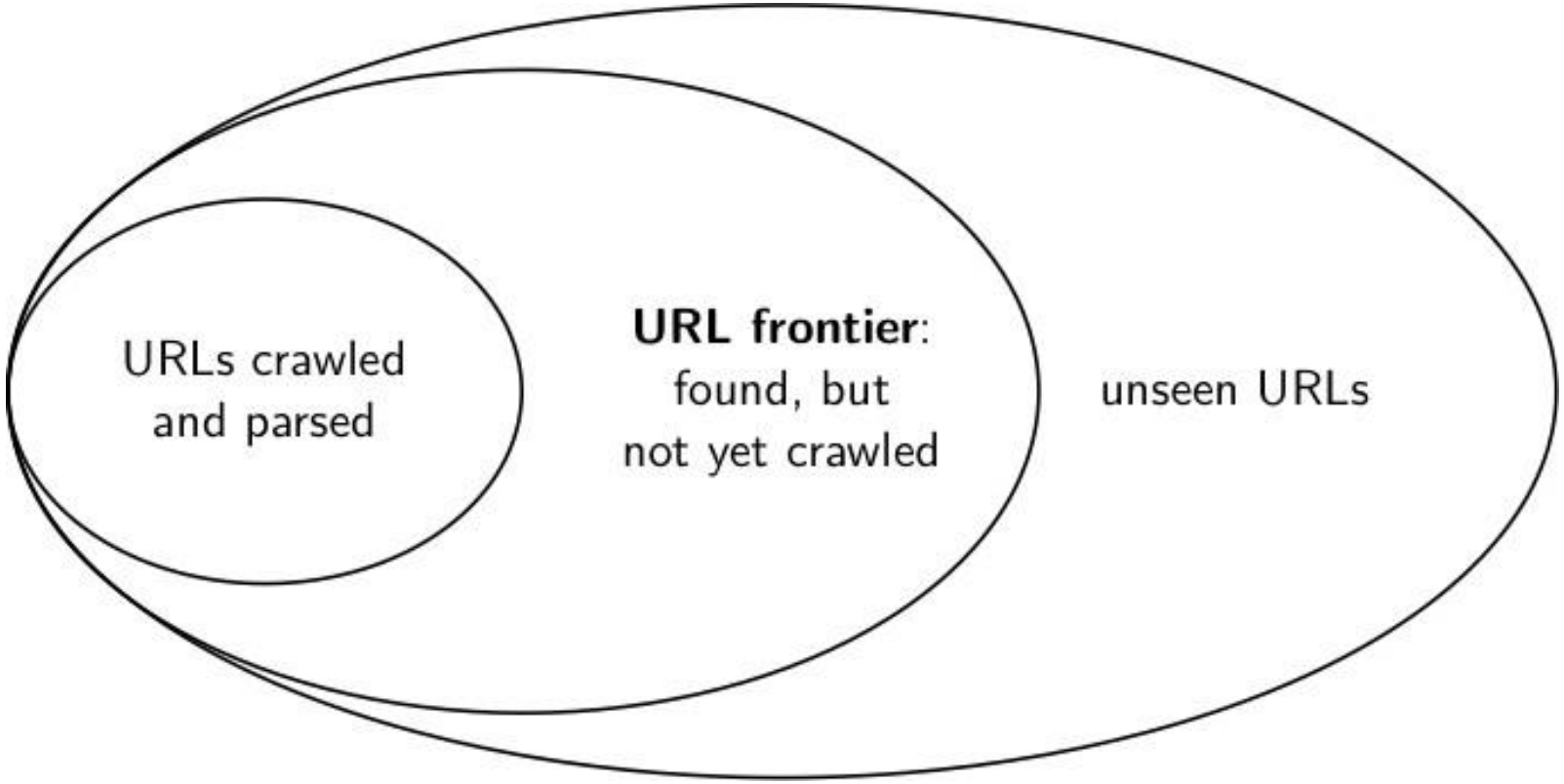
- Crawler Workflow :
  - Begin with known “seed” URLs
  - Fetch and parse them
    - Extract URLs they point to
    - Place the extracted URLs on a queue
  - Fetch each URL on the queue and repeat

Crawling Picture



URL frontier consists of the set of URLs
identified for crawling but not yet crawled

URL Frontier



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 - 1

- Be Polite: Respect implicit and explicit politeness considerations
  - Only crawl allowed pages
  - Respect *robots.txt* (more on this shortly)

- Be Robust: Be immune to spider traps and other malicious behavior from web servers

What any crawler should do - 2

- Be capable of distributed operation: designed to run on multiple distributed machines (may not be able to fetch all pages into a single machine)
- Be scalable: designed to increase the crawl rate by adding more machines to your distributed systems
- Performance/efficiency: permit full use of available processing and network resources

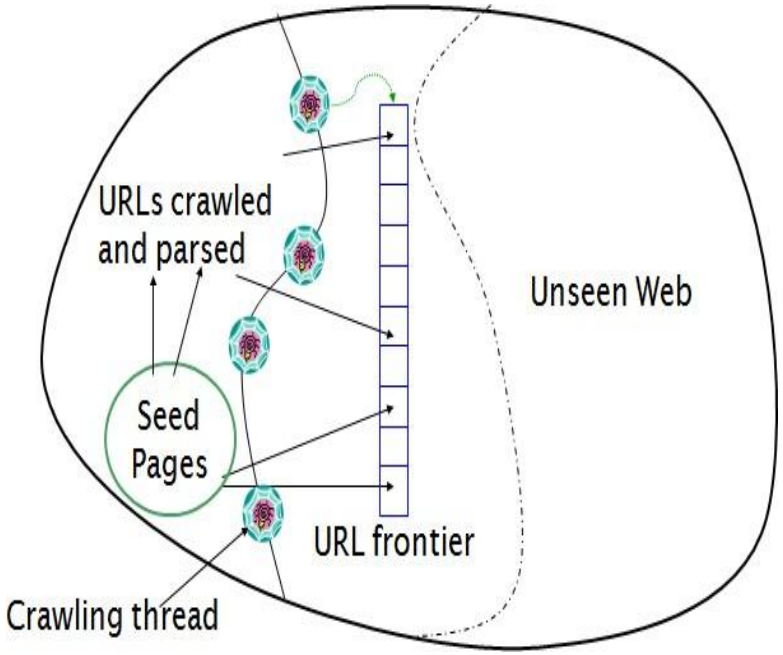
What any crawler should do - 3

- Fetch pages of "higher quality" first

- Continuous operation: Continue fetching fresh copies of a previously fetched page (crawling is never over)

- Extensible: Adapt to new data formats, protocols

Updated Crawler picture



- We have multiple spiders (distributed or multiple threads)
- URL frontier depicted as queue (structure later)
- Each crawler thread requests the next URL from frontier
- Parses the text at that URL
- Text sent to Indexing pipeline
- New found URLs are pushed back to URL frontier
- Region shown by Seed pages will grow into “crawled and parsed pages”

URL frontier Specifications - 1

1. Can include multiple pages from the same host
2. Must avoid trying to fetch them all at the same time
3. Must try to keep all crawling threads busy
4. There has to a be a trade-off between 2 and 3

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 example

PES UNIVERSITY

- Protocol for giving spiders ("robots") limited access to a website, originally from 1994
  - www.robotstxt.org/wc/norobots.html
- Website announces its request on what can(not) be crawled
  - For a server, create a file /robots.txt
  - This file specifies access restrictions
- No robot should visit any URL starting with "/yoursite/temp/", except the robot called "searchengine":

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

DNS server

- DNS : lookup service on the internet
  - Given a URL, retrieve its IP address
  - Service provided by a distributedset 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

- 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

Filters and Robots.txt

PES UNIVERSITY

- Filters – regular expressions for URL's 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

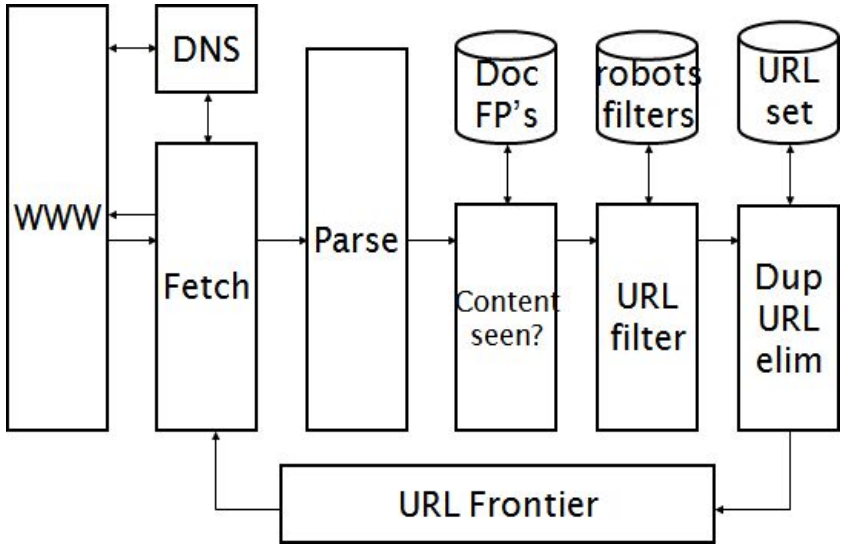
- Cache robots.txt files

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 – see details of frontier implementation

Processing steps in crawling

- Pick a URL from the frontier
- Fetch the document at the URL
- Parse the URL
  - Extract links from it to other docs (URLs)
- Check if URL has content already seen
  - If not, add to indexes
- For each extracted URL
  - Ensure it passes certain URL filter tests
  - Check if it is already in the frontier (duplicate URL elimination)



E.g., only crawl .edu, obey robots.txt, etc.

THANK YOU

Bhaskarjyoti Das
Department of Computer Science Engineering

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Crawler Overview

Bhaskarjyoti Das
Department of Computer Science Engineering

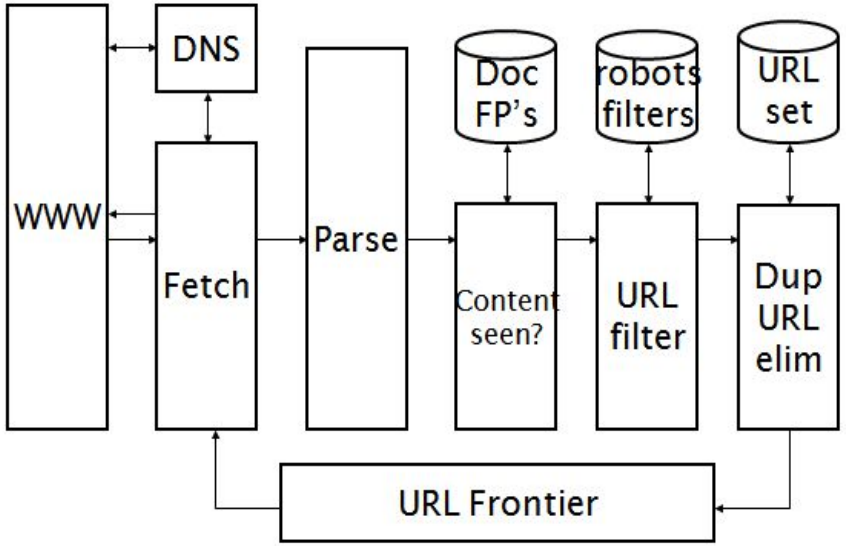
ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

URL Frontier

Bhaskarjyoti Das
Department of Computer Science Engineering

Recap : Processing steps in crawling

- Pick a URL from the frontier
- Fetch the document at the URL
- Parse the URL
  - Extract links from it to other docs (U
- Check if URL has content already seen
  - If not, add to indexes
- For each extracted URL
  - Ensure it passes certain URL filter tests
  - Check if it is already in the frontier (duplicate URL elimination)



E.g., only crawl .edu, obey robots.txt, etc.

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Distributing the Crawlers

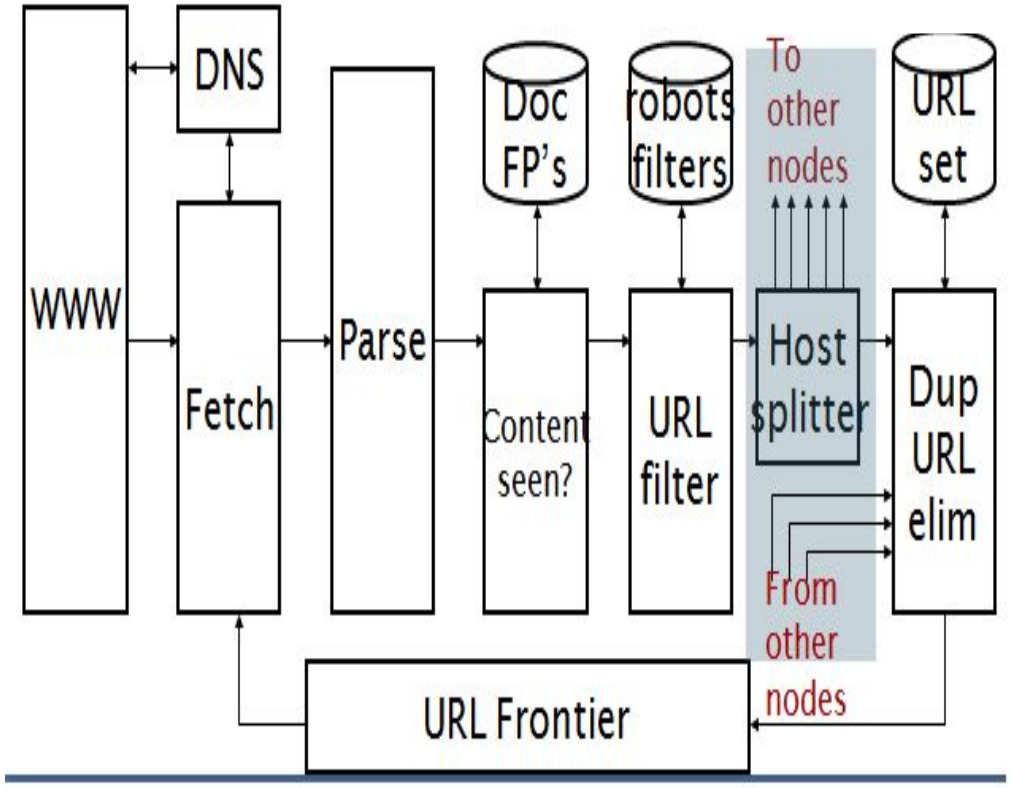
Bhaskarjyoti Das
Department of Computer Science Engineering

Distributing the Crawler

- Run multiple crawl threads, under processes – different 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?

Communication between Nodes in a Setup with Multiple Hosts

- Had it not been distributed, processed URL would have been pushed to URL frontier
- In the distributed setup, output of the URL filter at each node is sent to the Dup URL Eliminator of the appropriate node (hash into a host id)
- Like this particular node is sending a URL, it is also receiving processed URL
- Since already processed, previous processing steps can be skipped



URL Frontier: two main considerations

PES UNIVERSITY

- Politeness: do not hit a web server too frequently

- Freshness: crawl some pages more often than others (=priority)
  - E.g., pages (such as News sites) whose content changes often

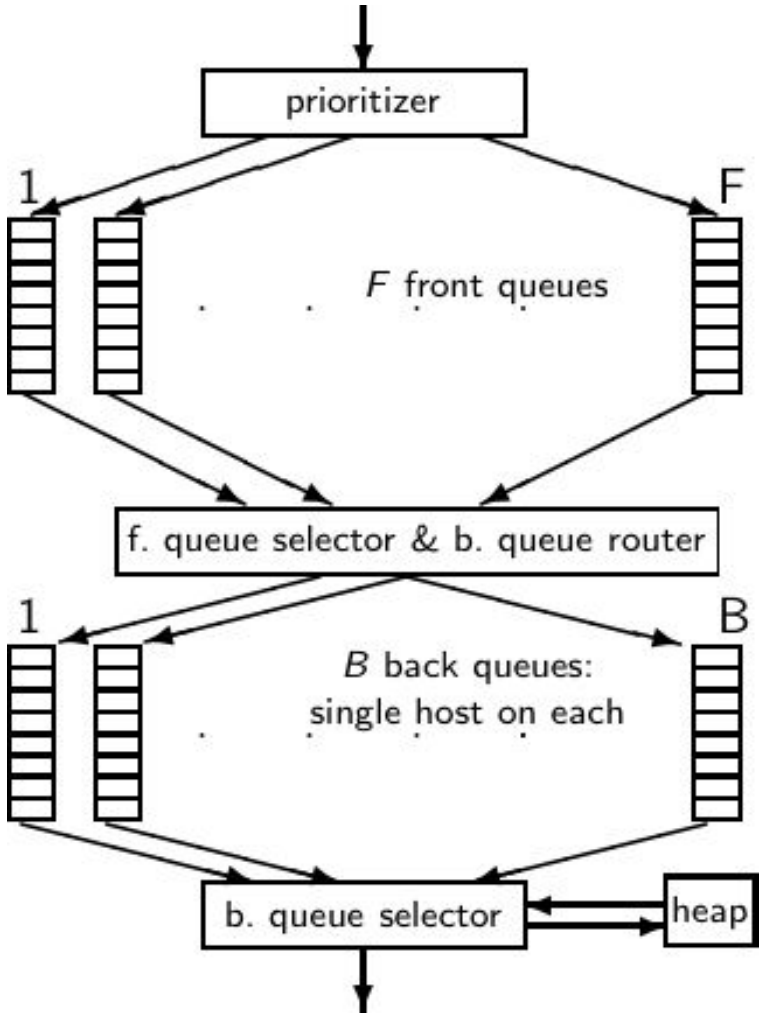
Had the URL frontier been implemented as a simple FIFO queue, these goals may conflict each other.

- A popular site may have many links out of a page go to its own site and these pages need to be refreshed for indexing
- They would have been inserted together into the FIFO queue
- They would be taken out together from the FIFO queue for the crawler threads creating a burst of accesses to that site
- This would have violated “politeness”

Freshness

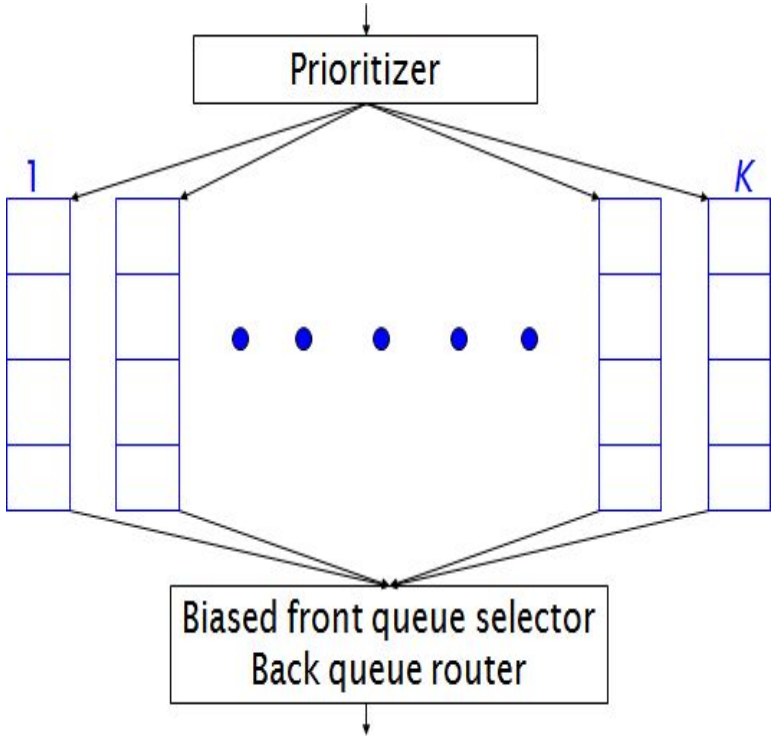
- Prioritizer assigns to URL an integer priority between 1 and *K*
  - Appends URL to corresponding queue
- Heuristics for assigning priority
  - Refresh rate sampled from previous crawls
  - Application-specific (e.g., “crawl news sites more often”)

Mercator URL frontier



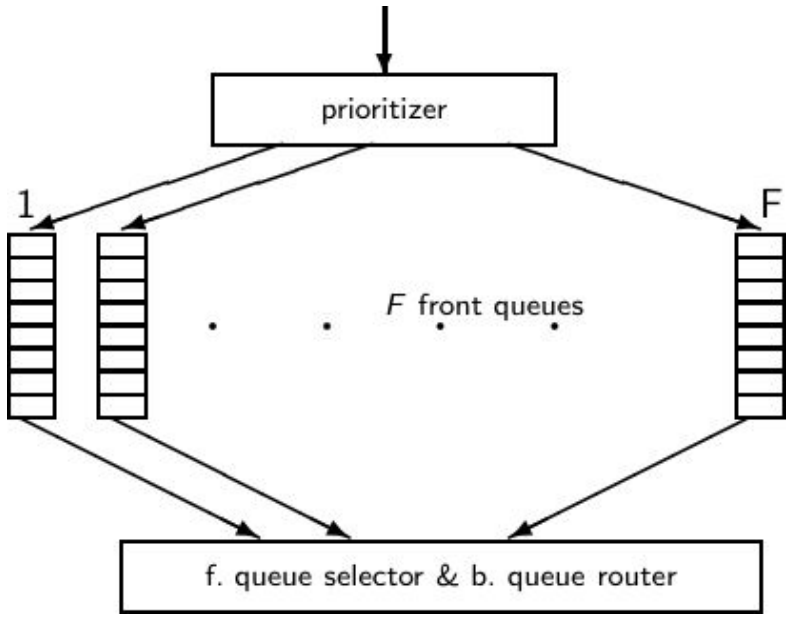
- URLs flow in from the top into the frontier.
- Front queues manage prioritization.
- Back queues enforce politeness. Heap helps in doing that
  - Each queue is FIFO.

Front Queues : Freshness



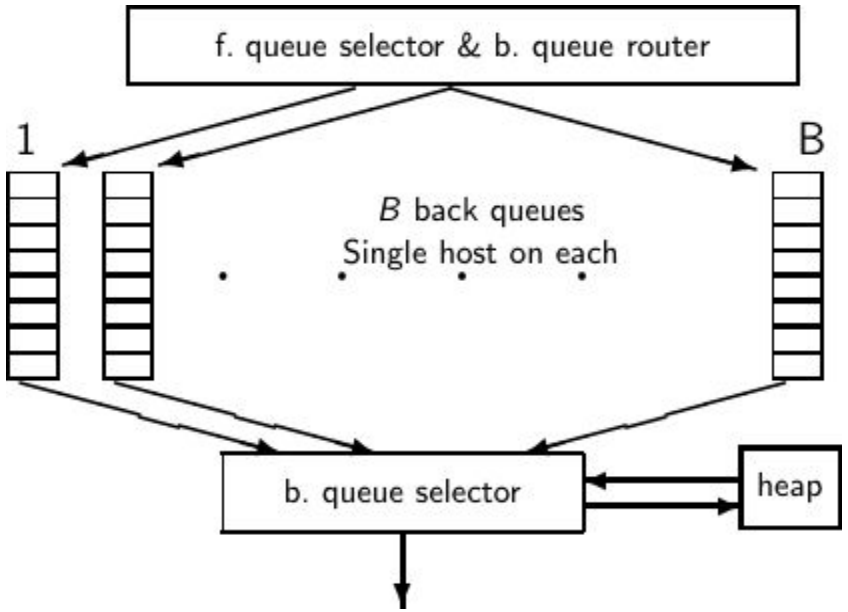
- Prioritizer assigns to URL an integer priority between and *F*. 1
- Priority 1 Queue for webserver that needs to be crawled very frequently (by tracking how frequently web pages are changing)
- Heuristics for assigning priority: refresh rate, PageRank etc
- Then appends URL to corresponding queue

Mercator URL frontier : Freshness by Biased Front Queue Selector



- Selection from front queues is initiated by back queues
- Pick a front queue from which to select next URL: Round robin (Example, 1, 1- 2, 1-2-3, etc), randomly (random number generated but a bigger range meant for high priority queue), or more sophisticated variant
- With a bias in favor of high-priority front queues.

Back Queue : Politeness



Host name	Back queue
...	3
	1
	B

- Invariant 1. Each back queue only contains URLs from a single host server.
- Invariant 2. Each back queue is kept non-empty while the crawl is in progress
- Maintain a table from hosts to back queues.

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

- The min-heap keeps the "earliest hitting time" for all the servers and root of the min-heap is selected

Back Queue Processing : How Fetcher Interacts with back Queue

PES UNIVERSITY

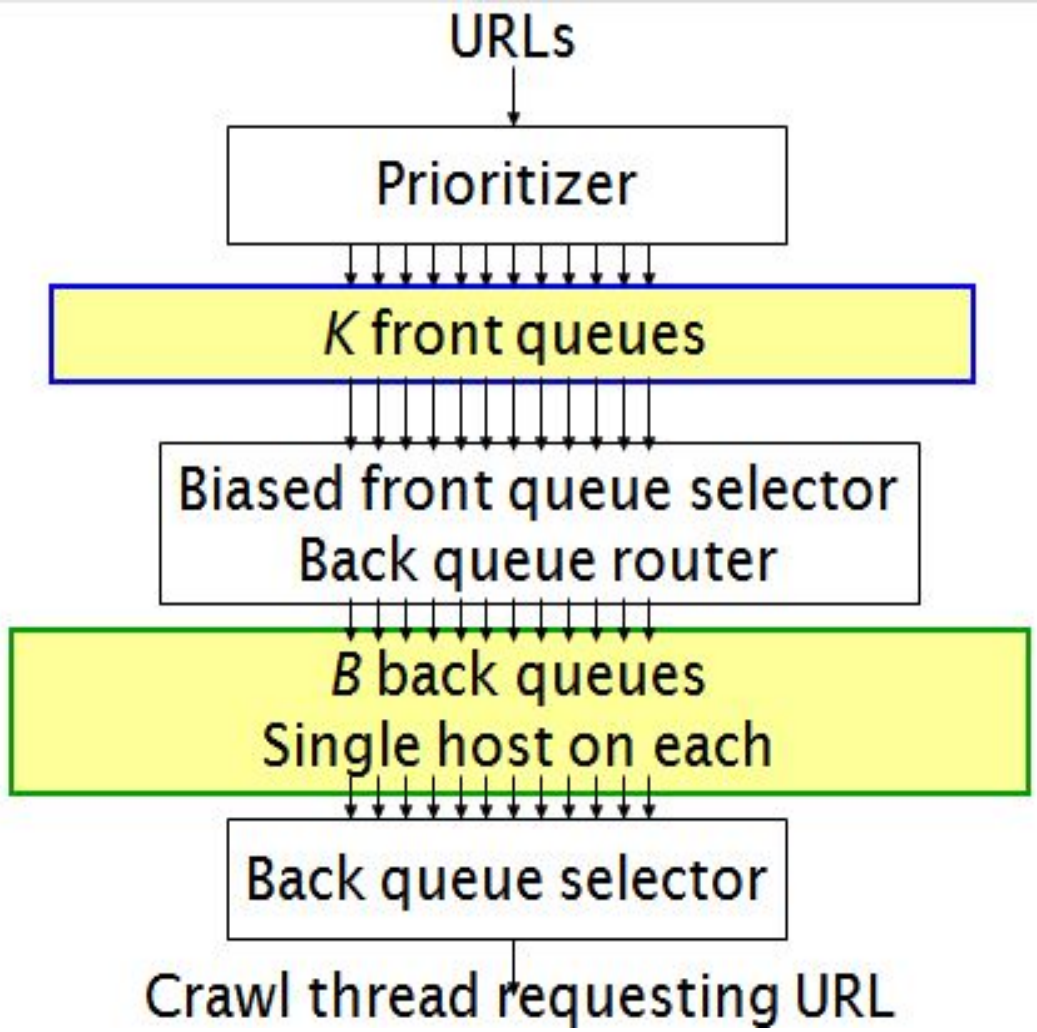
- A crawler thread seeking a URL to crawl:
- Extracts the root of the heap
- Fetches URL at head of corresponding back queue *q* (look up from table)
- Checks if queue *q* is now empty – if so, pulls a URL *v* from front queues
  - If there's already a back queue q for *v*'s host, append *v* to *q* and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

Number of back queue B

- Should we have as many back queue as there are web servers that we are crawling ?
  - No ! Number of back queues are limited i.e. a queue allocated for host A can get reallocated to host B
  - These back queuesare filled in a prioritydriven manner by biased front queue selector
  - Mercator recommendation: three times as many back queues as crawler threads

- Keep all threads busy while respecting politeness

URL Frontier : Mercator Scheme



THANK YOU

Bhaskarjyoti Das
Department of Computer Science Engineering

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Content Based Visual Information Retrieval

Surabhi Narayan
Department of Computer Science Engineering

Slides collated by:
Gaurav Mahajan (TA VIII sem)
Anshula Aithal (TA VIII sem)

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Machine Learning Approaches - 1

Surabhi Narayan
Department of Computer Science Engineering

Machine Learning Approaches

- The main aspects in designing a CBVIR system are :
    - feature extraction and representation,
    - dimension reduction and multidimensional indexing,
    - extraction of image semantics, and
    - design of user relevance feedback mechanisms.

- A variety of machine learning and deep learning techniques are being employed and developed to improve these systems.

Feature Extraction and Representation

- CBVIR systems should be able to automatically extract visual features that are used to describe the contents of an image or video clip.
- Examples of such features include color, texture, size, shape, and motion information.
- In specific contexts the process of feature extraction can be enhanced and/or adapted to detect other, specialized attributes, such as human faces or objects.
- Because of perception subjectivity, there is no best representation for a given feature.
- The color information, for instance, can be represented using different color models (e.g., RGB, HSV, YCbCr) and mathematical constructs, such as color histograms, color moments, color sets, color coherence vectors, or color correlograms. In a similar way, texture can be represented using co-occurrence matrix , Tamura texture features or Wavelets, to name just a few.

Dimension Reduction and Multidimensional Indexing

- The extracted features are grouped into some suitable data structure or mathematical construct (e.g., a normalized feature vector), and suitable metrics (e.g., Euclidean distance) are used to measure the similarity between an image and any other image.

- One of the techniques commonly used dimension reduction is principal component analysis (PCA).

- It is an optimal technique that linearly maps input data to a coordinate space such that the axes are aligned to maximally reflect the variations in the data.

- The QBIC system uses PCA to reduce a 20-dimensional shape feature vector to two or three dimensions.

Dimension Reduction and Multidimensional Indexing

- In addition to PCA, many researchers have used Karhunen–Loeve (KL) transform to reduce the dimensions of the feature space.

- Although the KL transform has some useful properties such as the ability to locate the most important subspace, the feature properties that are important for identifying the pattern similarity may be destroyed during blind dimensionality reduction.

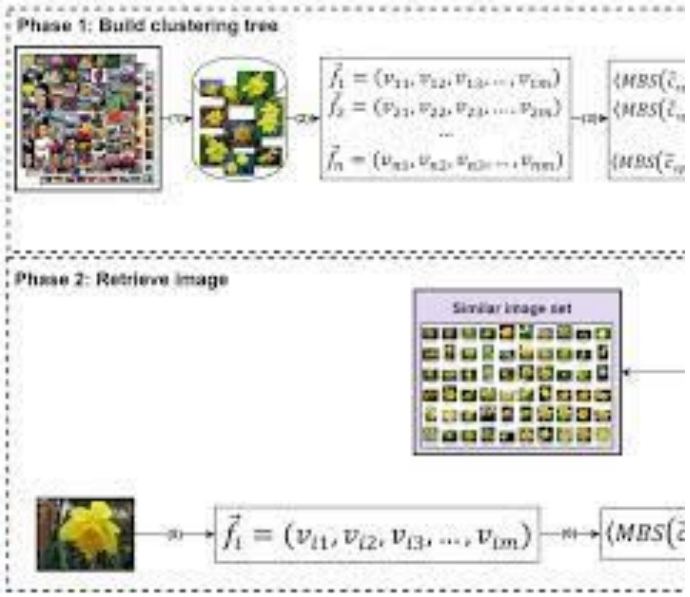
Apart from PCA and KL transformation, neural network has also been demonstrated to be a useful tool for dimension reduction of features

Dimension Reduction and Multidimensional Indexing

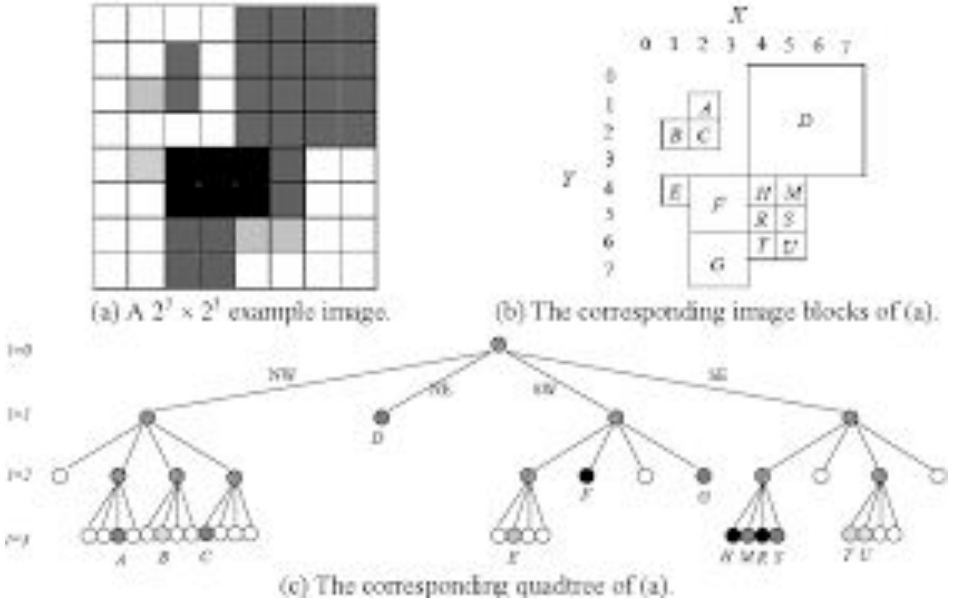
- After dimension reduction, the multidimensional data is indexed.
- A number of approaches have been proposed for this purpose, including R-tree (particularly, R*- tree), linear quad-trees, K-d-B tree, and grid files.
- Most of these multidimensional indexing methods have reasonable performance for a small number of dimensions (up to 20), but explore exponentially with the increasing of the dimensionality and eventually reduce to sequential searching.

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

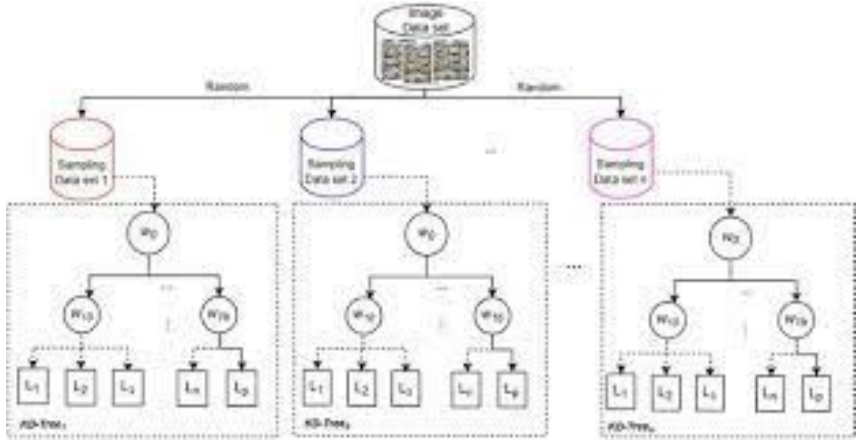
Dimension Reduction and Multidimensional Indexing



R-tree



linear quad-trees



K-d-B tree

Common techniques

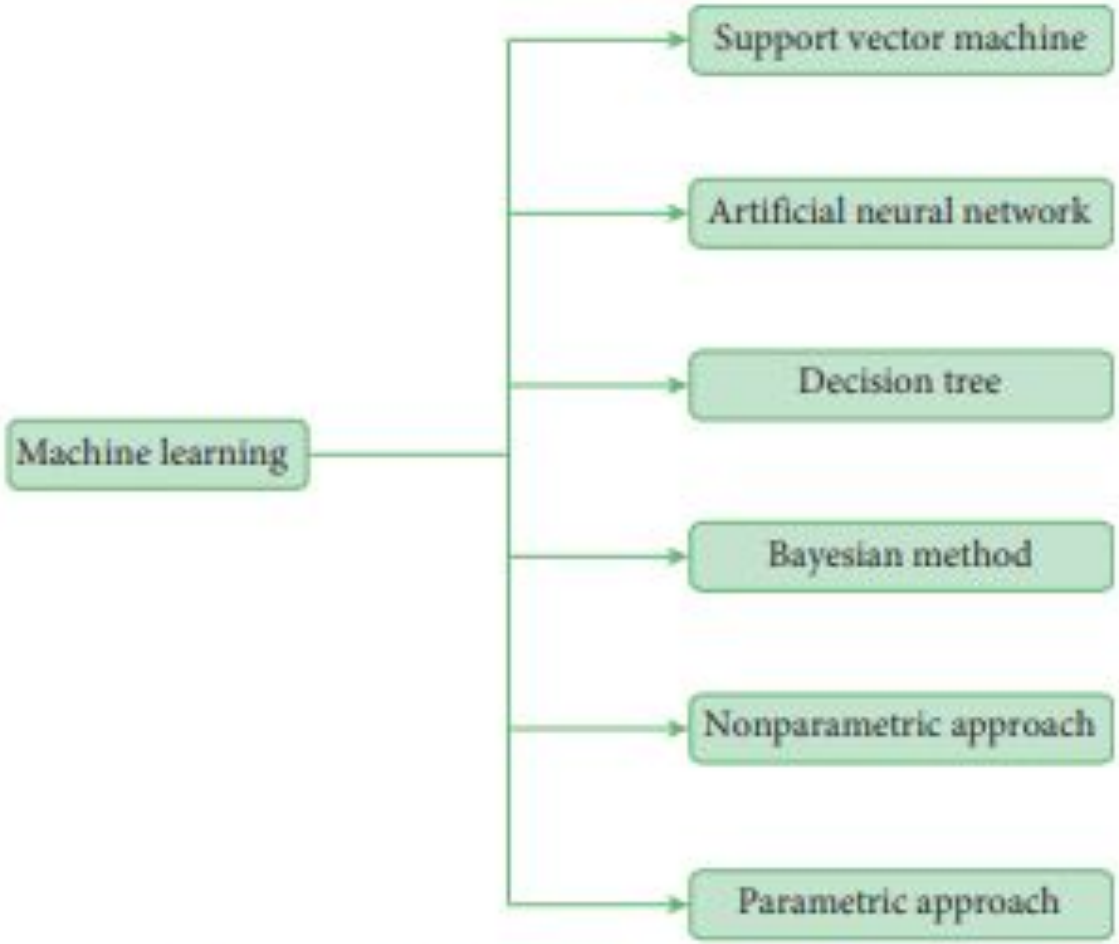


FIGURE 8: An overview of basic machine learning techniques for CBIR [1, 76].

Approach 1: TF-IDF CNN

PES UNIVERSITY

- Due to most prominent results and with a great performance of the deep convolutional neural network (CNN), a novel term frequency-inverse document frequency (TF-IDF) using as description vector the weighted convolutional word frequencies based on CNN is proposed for CBIR.

- The tf-idf weighting scheme for document retrieval is described as follows: Considering a vocabulary of ***n*** terms, the tf-idf is a **n-dimensional vector** that is calculated as the element-wise product of the tf and idf vectors. That is, a document is represented by a n-dimensional vector, $\boldsymbol{x}_d$, where each of its ***n*** dimensions is given by:

$$[\mathbf{x}_d]_t = F_{tf}(t, d) \times F_{idf}(d, t, D),$$

d is the index of a document,
t is the index of a certain term of the vocabulary of n terms,
D is the corpus,
$\boldsymbol{F_{tf}(t, d)}$ is the function which computes the frequency of term t in the document d, and
$\boldsymbol{F_{idf}(d, t, D)}$ is the function which computes the inverse document frequency

Approach 1: TF-IDF CNN

- The term frequency (***tf***) part produces a ***n***-dimensional vector where each dimension is given by the frequency of occurrence of each term ***t*** in the document ***d***, that is how many times the term ***t*** appears in the document ***d***.
- The inverse document frequency (***idf***) part provides information about the importance of each term, in the sense that terms that appear in many different documents are less informative, and hence of less importance, as compared to those that appear rarely.
- Thus, the idf part produces a ***n***-dimensional vector, ***r***, where each dimension is given by

$$[\mathbf{r}]_t = \log \frac{|\, D \,|}{|\, d \in D : F_{tf}(t, d) \neq 0 \,|},$$

where $|\cdot|$ is the cardinality of a set.

Approach 1: TF-IDF CNN

The filters of the convolutional layers of a pretrained CNN model have been trained to recognize specific visual features in the input image

consider these features as the convolutional words, and the learned filters as the convolutional word detectors

For an input image each filter is activated when it sees a certain visual feature, e.g. edges, blobs, shapes, etc., in the first layers or a visual concept, e.g, face, car, etc., in the last layers, and thus the activations of the filters reveal the degree of presence of the convolutional word that learned during the training procedure

Approach 1: TF-IDF CNN

To translate this to the tf-idf technique, the activations of each filter correspond to the tf part.

Since the activations of each filter output a 2-dimensional activation map of the responses of the filter at every position of the input image, they take the maximum activation value, in order to assign a unique activation value to every filter, considering that the degree of presence of the specific visual pattern is equal to the maximum degree of presence.

This leads to loss of the spatial information that capture the convolutional layers. Thus, the vocabulary size is equal to the number of the learned filters (or the convolutional neurons) of the utilized convolutional layer or the number of the neurons if we obtain the responses of the fully connected layers.

Approach 1: TF-IDF CNN

Since the range of the activation value of a neuron varies through the network layers, we apply a normalization step in order to ensure that the activation value of each neuron of the network lies in the range of [0, 1]. That is, for each image of the dataset, they divide each activation value of the neurons of a certain layer by the maximum activation value of the neurons of this layer over the entire dataset.

Approach 1: TF-IDF CNN

An issue arising on materializing the idf part is that the utilized normalized activations take values between 0 and 1, while in the standard tf-idf technique in text domain a word either exists or not. Thus, we need to investigate when a neuron is considered as activated or not.

The first approach defines a threshold T with value between 0 and 1. If the activations are over this threshold value, the neurons are considered as activated. That is:

$$x_n^i = \begin{cases} 1, & \text{if } a_n^i > T \\ 0, & otherwise \end{cases}$$

Approach 1: TF-IDF CNN

The second approach defines a percentage threshold value for the entire network activation and in each layer we consider as activated the neurons with the maximum activations in a cumulative way until we reach the predefined activation percentage threshold.

$$x_n^i = \begin{cases} 1, & \text{if } a_n^i \text{ in top K\% of activations} \\ 0, & otherwise \end{cases}$$

Approach 1: TF-IDF CNN

Finally, the third approach estimates the importance weight of each neuron based on the standard deviation of the normalized activation values of the specific neuron to the entire dataset. More specifically, the weight w_n of each neuron n is computed as follows:

$$w_n = \sqrt{\sum_i (a_n^i - \mu_n)^2},$$

where µn = i ai n is the mean value of the activations of the neuron n in the entire dataset I , and ai n is the normalized activation of the neuron n, for an image i ∈ I. Thus, the weighted description is given by the following equation:

$$x_n^i = w_n \times a_n^i$$

Approach 1: TF-IDF CNN

- The proposed weighting scheme tries to use the statistical behavior of the neurons in order to estimate its importance.
- That is, a neuron that is consistently activated with similar values throughout the entire dataset, and thus it has a very small standard deviation in the activation value, has limited importance in the retrieval task.
- The standard deviation of the activation resamples the idf value of the neuron.
- On the contrary, a neuron that has a large standard deviation is more informative since it has different behavior across the dataset and thus it can be used for discriminating the images. As previously, the tf term is estimated by the normalized activation of the specific neuron n to the specific image i.

Approach 1: TF-IDF CNN

PES UNIVERSITY

Pyramid-based approach

- In order to partially preserve spatial information from the activations of the convolutional layers they propose the following approach. They divide the activation map into S sections.

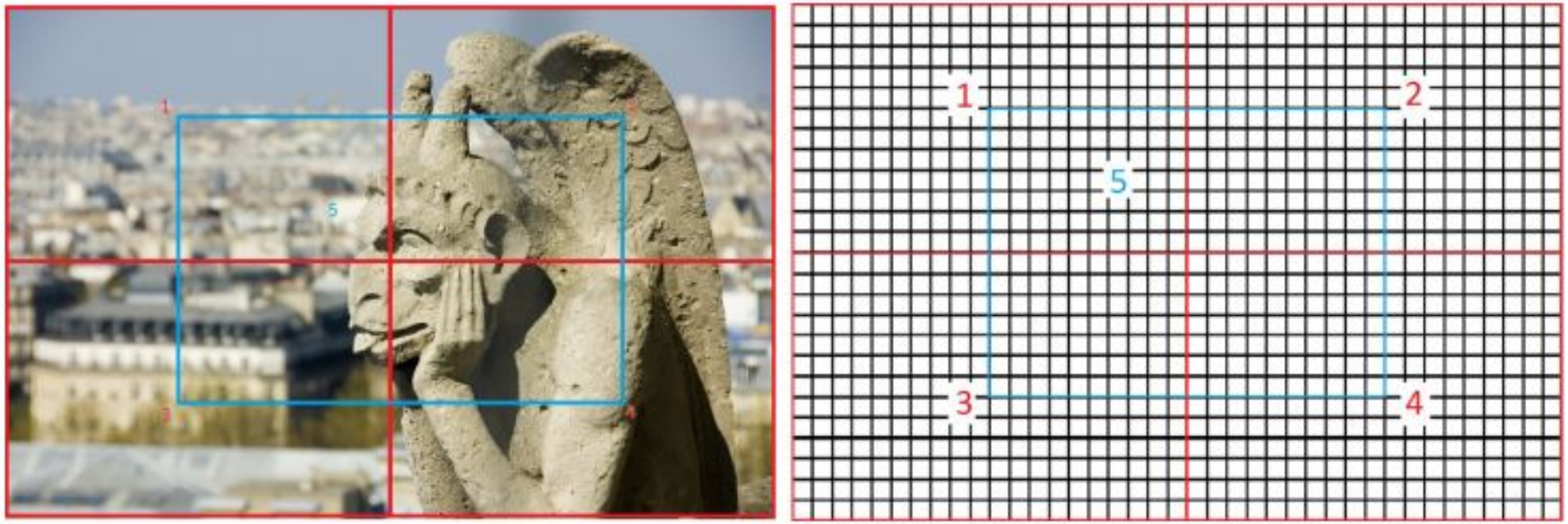
- Then they treat each of the resulting sections as a separate image. That is, we apply the proposed tf-idf approach to each of them. Thus, instead of considering the maximum activation value over the whole image, we consider the maximum activation value over each section.

- This value informs us about the presence of the convolutional words on the corresponding section of the input image.

Approach 1: TF-IDF CNN

- Regarding the idf part, which provides information about the importance of the certain convolutional word, they maintain the same weights calculated for the whole image.
- That is, they extract more than one value for a convolutional neuron for an input image in order to partially preserve the position of the detected convolutional word.
- Hence, they produce S times bigger description for the whole image. This allows us to decide not only whether two images contain the same convolutional words, but also if these words are approximately in the same position on the image.
- It is clear that the more sections lead to more detailed detections on the image, however this also renders the comparison between two images more difficult.

Approach 1: TF-IDF CNN



(a) Five sections in an input image (b) Five sections in the activation map

Fig. 1 Division of the activation maps into sections

Approach 1: TF-IDF CNN

- More specifically, the initial image is divided into five sections (four non overlapping sections whose width and height are equal to the half-width and half-height of the full image, respectively, while the fifth section positioned in the center of the image is of equal dimensions, and has 25% overlap with each of the other four sections), and correspondingly the activation map for a certain neuron is divided into the same sections.

- Then, the maximum activation values for each of the five sections is derived, and the tf-idf scheme is applied. The final representation is derived by concatenating the five representations.

Approach 1: TF-IDF CNN

- Kondylidis *et al* utilize a commonly used CNN model, that is the VGG-16, to apply the proposed tf-idf scheme, and subsequently utilize their recently published fully convolutional model, optimized toward image retrieval in a fully unsupervised fashion.
- They first utilize the VGG 16-layer model, trained on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2014 to classify 1,000 ImageNet classes, since it is a commonly used baseline model. The model consists of sixteen trained neural layers; the first thirteen are convolutional and the remaining three are fully connected.

- Max-pooling layers follow the second, forth, seventh, tenth, and thirteenth convolutional layers, while the ReLU non-linearity (f (x) = max(0, x)) is applied to every convolutional and fully connected layer, except the last fully connected layer. The output of the last fully connected layer is a distribution over 1000 ImageNet classes. The softmax loss is used during the training. They use the VGG 16 layer model to directly extract the representations from certain layers in order to apply the proposed tf-idf approach.

Approach 1: TF-IDF CNN

- The CNN features have a hierarchical nature. That is, convolutional neurons at the first levels are meant to detect low level local concepts like edges and corners.
- The next levels are able to detect more complicated patterns like basic shapes, which are based on the detections of previous convolutional levels, which can be considered as combination of convolutional words; the mid level local concepts.
- The last convolutional layers are able to detect even more complicated patterns, close to ones that humans detect, like hands, faces or even cars; the high level local concepts.
- Finally, the activations of neurons of the fully connected layers are based on the detection of combinations of patterns and the reason they activate cannot be pointed directly on some pattern on the input image.
- Neurons of fully connected layers produce only one output for the whole image, and they are able to detect high level global concepts.

Approach 1: TF-IDF CNN

PES UNIVERSITY

Fully Unsupervised Model

- The Fully Unsupervised (FU) model, is a recent state-of-the-art fully convolutional model, which is retrained on the modified CaffeNet model, n order to produce more efficient and compact image representations for the retrieval task.

- More specifically, in the Fully Unsupervised (FU) approach, the goal is to amplify the primary retrieval presumption that the relevant image representations are closer to the certain query representation in the feature space. The rationale behind this approach is rooted to the cluster hypothesis which states that documents in the same cluster are likely to satisfy the same information need.

- That is, the pre-trained CNN model is retrained on the given dataset, aiming at maximizing the cosine similarity between each image representation and its n nearest representations, in terms of cosine distance.

Approach 1: TF-IDF CNN

The set of N images to be searched is denoted by $\mathcal{I} = \{\mathbf{I}_i, i = 1, \ldots, N\}$, their corresponding feature representations emerged in the L layer by $\mathcal{X} = \{\mathbf{x}_i, i = 1, \ldots, N\}$, and by μ^i the mean vector of the $n \in \{1, \ldots, N-1\}$ nearest representations to $\mathbf{x}_i$, denoted as $\mathcal{X}^i = \{\mathbf{x}_l^i, l = 1, \ldots, N-1\}$. That is,

$$\mu^i = \frac{1}{n} \sum_{l=1}^{n} \mathbf{x}_l^i$$

The optimization problem to be solved is:

$$\max_{\mathbf{x}_i \in \mathcal{X}} \mathcal{J} = \max_{\mathbf{x}_i \in \mathcal{X}} \sum_{i=1}^{N} \frac{\mathbf{x}_i^\mathsf{T} \mu^i}{\|\mathbf{x}_i\| \, \|\mu^i\|}$$

Approach 1: TF-IDF CNN

The above optimization problem is solved using gradient descent. The first-order gradient of the objective function $\mathcal{J}$ is given by:

$$\frac{\partial \mathcal{J}}{\partial \mathbf{x}_i} = \frac{\partial}{\partial \mathbf{x}_i}\left(\sum_{i=1}^{N} \frac{\mathbf{x}_i^{\mathsf{T}} \mu^i}{\|\mathbf{x}_i\| \, \|\mu^i\|}\right) = \frac{\mu^i}{\|\mathbf{x}_i\| \, \|\mu^i\|} - \frac{\mathbf{x}_i^{\mathsf{T}} \mu^i}{\|\mathbf{x}_i\|^3 \, \|\mu^i\|}\mathbf{x}_i$$

The update rule for the v-th iteration for each image can be formulated as:

$$\mathbf{x}_i^{(v+1)} = \mathbf{x}_i^{(v)} + \eta\left(\frac{\mu^i}{\|\mathbf{x}_i^{(v)}\| \, \|\mu^i\|} - \frac{\mathbf{x}_i^{(v)\mathsf{T}} \mu^i}{\|\mathbf{x}_i^{(v)}\|^3 \, \|\mu^i\|}\mathbf{x}_i^{(v)}\right), \quad \mathbf{x}_i \in \mathcal{X}$$

A normalization step has been introduced as:

$$\mathbf{x}_i^{(v+1)} = \mathbf{x}_i^{(v)} + \eta\|\mathbf{x}_i^{(v)}\| \, \|\mu^i\|\left(\frac{\mu^i}{\|\mathbf{x}_i^{(v)}\| \, \|\mu^i\|} - \frac{\mathbf{x}_i^{(v)\mathsf{T}} \mu^i}{\|\mathbf{x}_i^{(v)}\|^3 \, \|\mu^i\|}\mathbf{x}_i^{(v)}\right), \quad \mathbf{x}_i \in \mathcal{X}$$

Approach 1: TF-IDF CNN

- Hence, using the above representations as targets in the layer of interest, a regression task is formulated for the neural network, which is initialized on the CaffeNet's weights and is trained on the utilized dataset, using back-propagation.

- The Euclidean loss is used during training for the regression task. Thus, the procedure is integrated by feeding the entire dataset into the input layer of the retrained adapted model and obtaining the new representations.

Approach 1: TF-IDF CNN

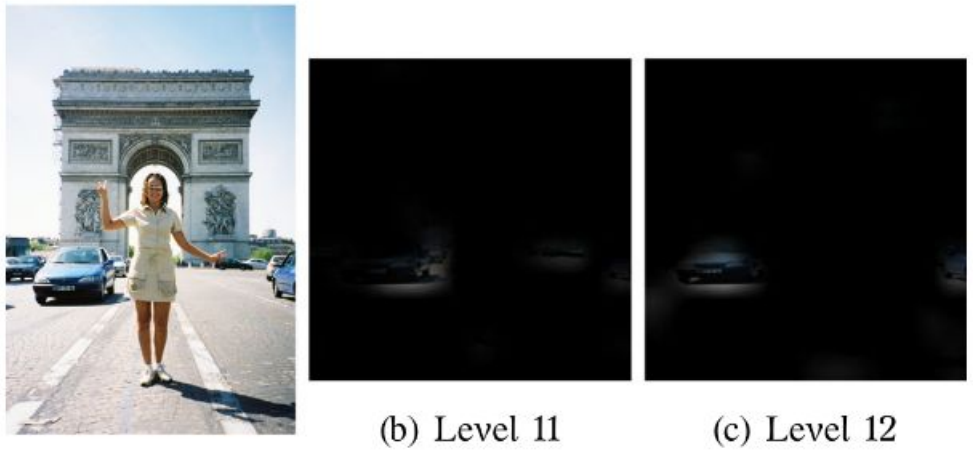
- More specifically, they study the usefulness of the patterns that are detected by the aforementioned layers. Toward this aim, they also propose a novel visualization method, which reveals in which parts of the input image, a convolutional filter was activated, and thus, what patterns has been trained to recognize. The technique of filter visualization is described below.

- Every convolutional neuron takes as input many regions of the input image and outputs an activation value for each of them, forming the activation map.

- The values of the activation maps are normalized as mentioned above in order to belong to the interval [0, 1]. We multiply the RGB values of all pixels of one region with the produced activation value.

- This produces a new image where every region that did not activate the neuron is shaded. We use bicubic interpolation to resize the activation map so that the visual result appears more uniform.

Approach 1: TF-IDF CNN



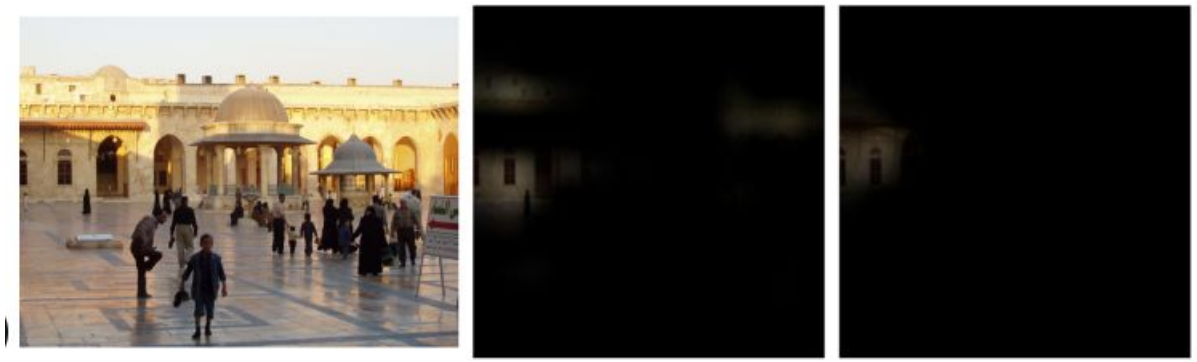
(a) Input Image (b) Level 11 (c) Level 12

Fig. 3 Example of tracing cars

(a) Level 9 (b) Level 10 (c) Level 11 (d) Level 13

Fig. 4 Example of tracing human parts for the input image of Fig. 3

Approach 1: TF-IDF CNN



(a) Input Image (b) Level 12 (c) Level 13

Fig. 5 Example of tracing windows

(a) Level 9 (b) Level 10 (c) Level 11 (d) Level 12 (e) Level 13

Approach 1: TF-IDF CNN

PES UNIVERSITY

Query Expansion using pseudo relevance feedback

- Query Expansion is a standard, in most cases of negligible cost, technique for accomplishing better retrieval results.
- Concisely, the idea is to re-formulate the initial query, utilizing information deriving from the evaluation of the initial query.
- The majority of CBIR methods include a query expansion step that boosts the retrieval performance.
- On the top of the proposed descriptors they also introduce a simple query expansion method using Pseudo Relevance Feedback.

Approach 1: TF-IDF CNN

That is, they propose to re-issue the top n ranked results from the Initial query as a new query, in order to enhance the original query representation with additional relevant representations, following either the average query expansion scheme or the max one.

The average scheme can be described as follows:

> Let $\mathbf{Q}$ be a certain query image, and $\mathbf{q}$ the corresponding CNN representation using the proposed method. We consider the top k retrieved images and their corresponding CNN representations $\mathbf{x}_j$, $j = 1, \ldots, k$. Then, the new query representation $\mathbf{q}_{new}$ is as follows:

$$\mathbf{q}_{new} = \frac{1}{k+1}\left(\mathbf{q} + \sum_{j=1}^{k} \mathbf{x_j}\right)$$

The max scheme considers as the new query representation the max values of the top k retrieved images in each dimension.

Approach 1: TF-IDF CNN

PES UNIVERSITY

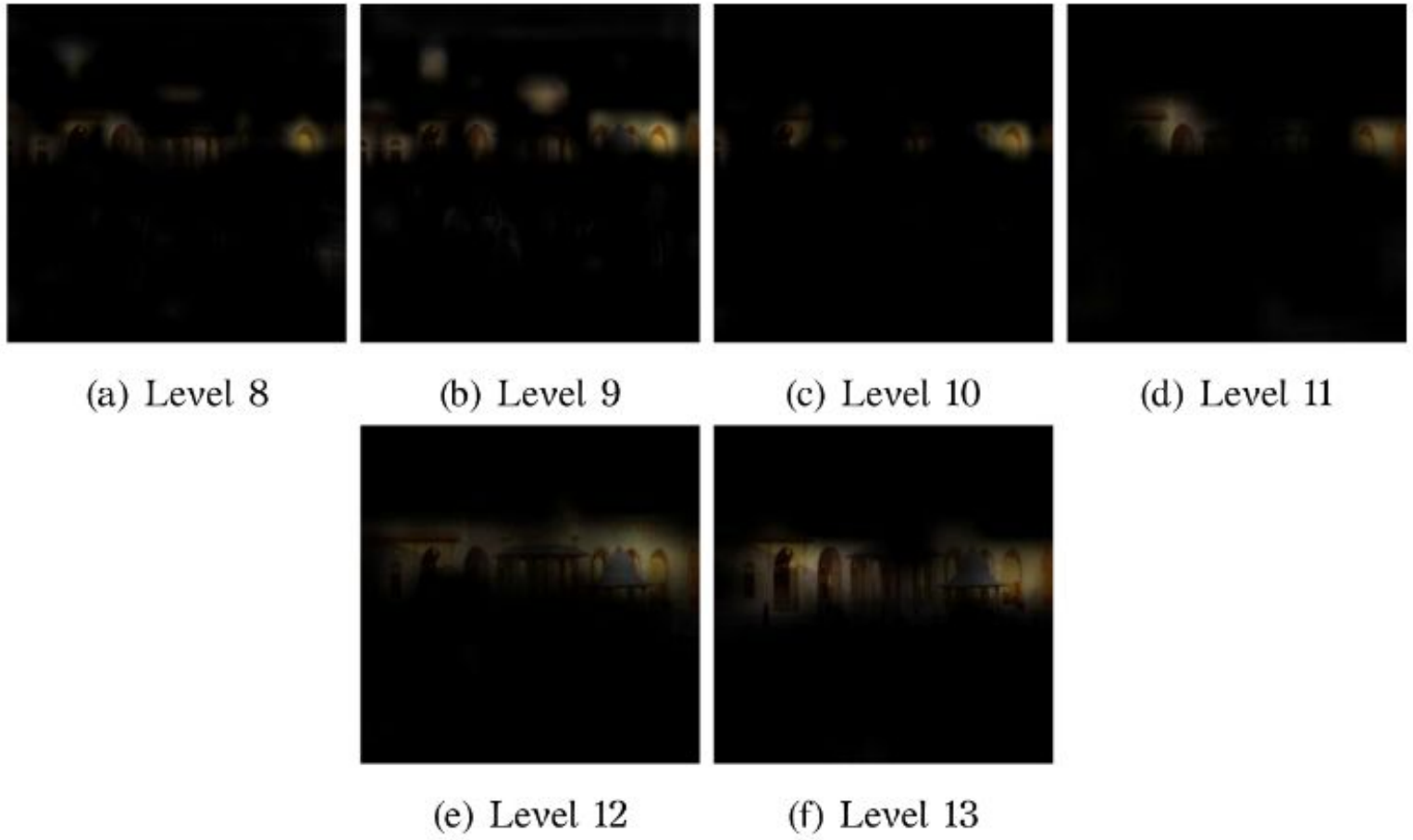
Throughout this work they use **mean Average Precision (mAP)**, and top-N score in order
to evaluate the performance of the proposed method. The definitions of the above metrics
follow below:

Mean average precision is the mean value of the Average Precision (AP) of all the queries. The definition of AP for the i-th query is formulated as follows:

$$AP_i = \frac{1}{Q_i} \sum_{n=1}^{N} \frac{R_i^n}{n} t_n^i,$$

where Q_i is the total number of relevant images for the i-th query, N is the total number of images of the search set, R_i^n is the number of relevant retrieved images within the n top results; t_n^i is an indicator function with $t_n^i = 1$ if the n-th retrieved image is relevant to the i-the query, and $t_n^i = 0$ otherwise.

Approach 1: TF-IDF CNN



(a) Level 8 (b) Level 9 (c) Level 10 (d) Level 11

(e) Level 12 (f) Level 13

Fig. 7 Example of tracing arch like patterns for the input image of Fig. 5

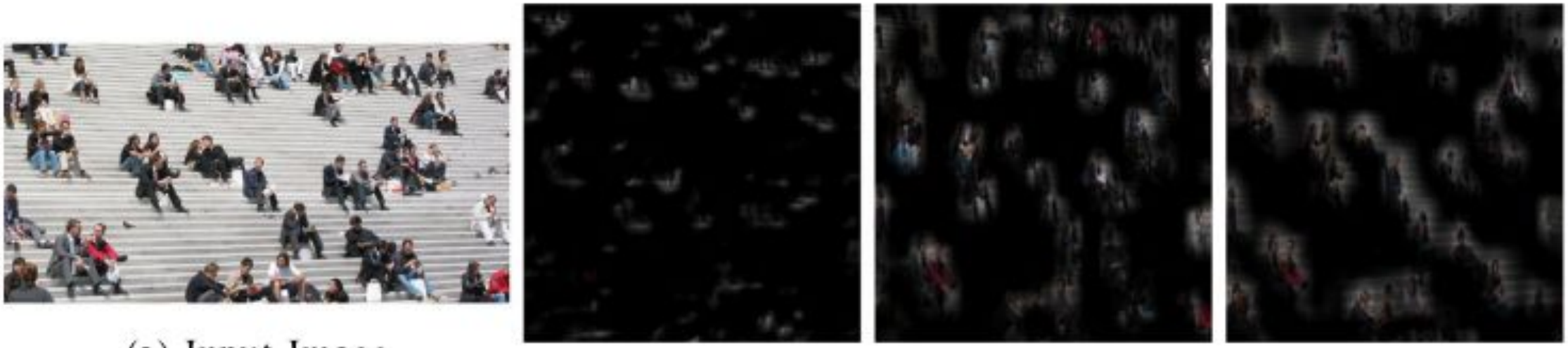
Approach 1: TF-IDF CNN

(a) Level 9 (b) Level 10 (c) Level 11 (d) Level 12 (e) Level 13

Fig. 8 Example of tracing domes for the input image of Fig. 5

Top-N score refers to the average number of same-object images, within the top-N ranked images.

Approach 1: TF-IDF CNN



(a) Input Image (b) Level 7 (legs) (c) Level 9 (bodies) (d) Level 9 (heads)

Fig. 9 Example of tracing Human parts

Approach 1: TF-IDF CNN

PES UNIVERSITY

Datasets

Inria Holidays : consists of 991 images divided into 500 classes, and 500 discrete queries. Each class in the search set consists of between 1 and 12 images. Some images of the dataset are not in a natural orientation. We measure the retrieval performance in terms of mAP.

Oxford 5k : consists of 5062 images collected from Flickr by searching for particular Oxford landmarks. The collection has been manually annotated to generate a comprehensive ground truth for 11 different landmarks, each represented by 5 possible queries. Images are assigned one of four possible queries: Good, Ok, Junk and Absent. Good and ok images are considered as positive examples, absent as negative examples while junk images as null examples. Following the standard evaluation protocol they measure the retrieval performance in mAP.

Approach 1: TF-IDF CNN

Paris 6k : similar to Oxford Buildings dataset, consists of 6392 images (20 of the 6412 provided images are corrupted) collected from Flickr by searching for particular Paris landmarks and provides 55 queries. The retrieval performance is also measured in terms of mAP, using both the full queries and cropped queries versions of the dataset.

UKBench: contains 10200 images of objects divided into 2550 classes. Each class consists of 4 images. All 10200 images are used as queries. The performance is reported as Top-4 Score, which is a number between 0 and 4.

Approach 1: TF-IDF CNN

- They apply the proposed tf-idf approach in the VGG model, on certain layers and they also experiment with combinations of different layers, in order to show that the proposed scheme can improve the baseline results of directly extracted feature representations using a commonly used CNN pre-trained model.

- Additionally, in order to partially preserve spatial information from the convolutional layers, in the VGG case, they apply the aforementioned method of division into five sections in the last convolutional layer.

Approach 1: TF-IDF CNN

- Subsequently, they apply the proposed tf-idf based representation scheme on the FU retrained model, in order to show that the proposed method is applicable to different network architectures, and also can be combined with other state-of-the-art CNN-based works, improving their performance even more.
- In this case, they perform experiments utilizing the last two optimized convolutional layers, where the max pooling operator outputs 384-dimensional, and 256-dimensional feature representations respectively. All results are obtained using the **cosine distance**.

- In the following they abbreviate the proposed tf-idf scheme utilizing the VGG model as TF-IDF(VGG), and correspondingly TF-IDF(FU) the proposed tf-idf scheme on the FU optimized model.

Approach 1: TF-IDF CNN

Feature representation	Oxford 5k cropped	Oxford 5k full	Paris 6k cropped	Paris 6k full
VGG baseline	0.411	0.475	0.358	0.676
TF-IDF(VGG)	0.473	0.543	0.699	0.705
TF-IDF(VGG) & QE	0.52	0.563	0.757	0.746

Feature representation	Inria holidays	UKBench
VGG baseline	0.772	3.184
TF-IDF(VGG)	0.8	3.668
TF-IDF(VGG) & QE	–	3.779

Tables 1 and 2 illustrate the experimental results utilizing the VGG-16 model, for the Oxford 5k and the Paris 6k datasets, and for the Inria Holidays and the UKBench, respectively.

They compare the proposed method against the baseline VGG utilizing the same layers, and extracting directly the feature representations from them, using the common max-pooling operation in the case of the convolutional layers

Approach 1: TF-IDF CNN

- As they can observe the proposed method can achieve notably improved results against the baseline, in all the used datasets, validating our claim that the tf-idf weighting method can successfully applied in the deep CNNs enhancing the information captured from the CNN layers. Furthermore they observed that the query expansion gives another boost in the performance.

- We should also note that the query expansion cannot be applied to the Inria Holidays dataset, since in many cases there is only one relevant to the query sample in the dataset.

Approach 1: TF-IDF CNN

Feature representation	Oxford 5k	Oxford 105k	Paris 6k	Paris 106k	UKBench
FU baseline	0.5509	0.5302	0.8107	0.7468	3.8094
TF-IDF(FU)	0.5742	0.5476	0.8292	0.7481	3.8421

Subsequently, in Table 3, they provide the experimental results for the UKBench, Paris 6k, and Oxford 5k datasets, for the tf-idf scheme on the FU retrained model.

Approach 1: TF-IDF CNN

Table 4 mAP and Top-4 Score for different idf computation approaches (FU retrained model)

Idf approach	UKBench	Oxford 5k	Oxford 105k	Paris 6k	Paris 106k
No idf	3.8094	0.5509	0.5302	0.8107	0.7468
Threshold value	3.8336	0.5658	**0.5476**	0.8270	0.7461
Percentage threshold	3.8294	**0.5742**	0.5475	**0.8292**	**0.7481**
Statistical idf	**3.8421**	0.5636	0.5415	0.8287	0.73

- In Table 4 they present the Top-4 Score and mAP for the different idf computation approaches in the UKBench, Oxford and Paris datasets, utilizing the FU retrained model.
- The best results are printed in bold. It can be observed that the proposed tf-idf schemes outperform the baseline without tf-idf in all the utilized datasets.

Table 5 Comparison against other unsupervised methods

Method	Dim	Oxford 5k	Oxford 105k	Paris 6k	Paris 106k	UKBench
CVLAD* [47]	64k	0.514	–	–	–	3.62
VLAD* [1]	128	0.448	–	–	–	–
T-embedding* [15]	512	0.528	0.461	–	–	–
Fine-residual VLAD [24]	256	–	–	–	–	3.43
BOW* [17]	200k	0.364	–	0.46	–	2.81
SPoC [2]	256	**0.589**	**0.578**	–	–	3.65
Multi-layer [45]	4k	0.567	–	–	–	3.243
MAC* [39]	256	0.522	–	–	–	–
Small Memory Footprint Regimes [33]	256	0.533	0.489	0.67	–	3.368
TF-IDF(FU)	640	0.5742	0.5476	**0.8292**	**0.7481**	**3.8425**

- Table 5 provides a comparison of the proposed tf-idf scheme on the FU optimized model, against other CNN-based as well as hand-crafted techniques for CBIR. The best results are printed in bold.
- Since the proposed method does not utilize supervised learning, they only compare it with unsupervised methods.

PES UNIVERSITY

Conclusion

- Experimental results on four challenging image retrieval datasets demonstrated the improved performance of the proposed approach.
- It should also be noted that the proposed approach can be easily combined with more sophisticated approaches that have been recently proposed to give a new perspective on treating convolutional image retrieval.
- To this aim, they also conducted experiments utilized our fully unsupervised model toward image retrieval enhancing even more the retrieval performance, leading also to state-of-the-art results against other unsupervised approaches.

References

- “Content-Based Visual Information Retrieval”, Oge Marques and Borko Furht
- “Exploiting tf-idf in deep Convolutional Neural Networks for Content Based Image Retrieval” - Nikolaos Kondylidis, Maria Tzelepi, Anastasios Tefas.
- “Content-Based Image Retrieval and Feature Extraction: A Comprehensive Review” - Afshan Latif

THANK YOU

Surabhi Narayan

Department of Computer Science Engineering

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Content Based Visual Information Retrieval

Surabhi Narayan
Department of Computer Science Engineering

Slides collated by:
Gaurav Mahajan (TA VIII sem)
Anshula Aithal (TA VIII sem)

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Machine Learning Approaches - 2

Surabhi Narayan
Department of Computer Science Engineering

Approach 2: Hashing Based Retrieval

- Hashing-based retrieval methods refer to techniques that use hash functions to map data into a fixed-size hash code or hash value. These methods are commonly used in information retrieval systems to efficiently store and retrieve data, especially in scenarios where the dataset is large.
- The purpose of hashing is to convert high-dimensional images to low-dimensional binary codes while preserving the similarity relation of the images in the original space.
- Hashing allows for rapid data lookup and retrieval by mapping data items to hash codes and organising them in hash tables or other data structures.

Approach 2: Hashing Based Retrieval

- To handle large-scale image data, hashing-based retrieval methods have become attractive because hashing can encode the high-dimensional data into compact binary codes while maintaining similarity among neighbors, leading to significant gains in both computation and storage.

Approach 2: Hashing Based Retrieval

Based on whether employing semantic information, hashing methods can be classified into two groups:

Unsupervised Hashing:

- The primary goal of unsupervised hashing is to explore the intrinsic structure of the data, maintain the similarity among neighbours without any semantic information and represent it as compact binary codes without relying on any external semantic information.
- Unsupervised hashing methods typically leverage unsupervised learning algorithms to map data points into binary codes. These methods often focus on preserving the pairwise similarities or distances between data points during the hashing process.

Approach 2: Hashing Based Retrieval

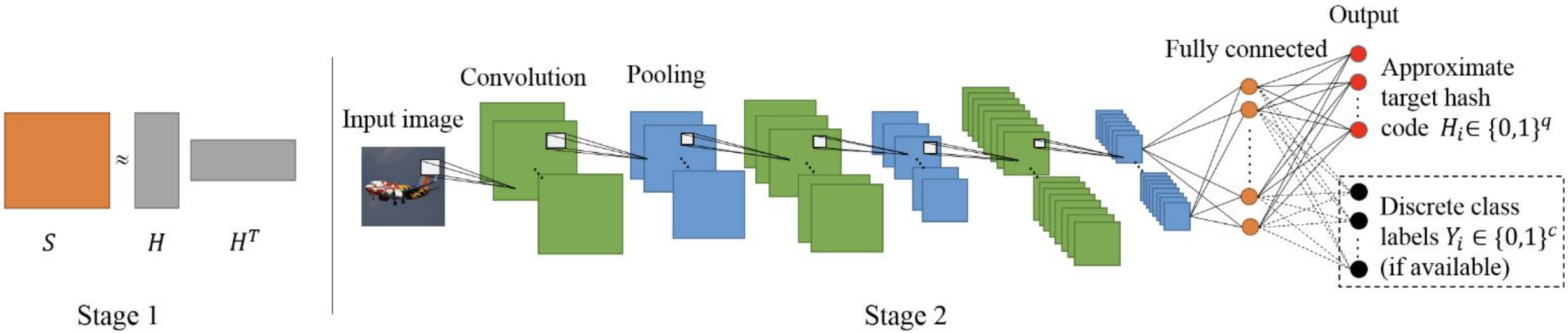
- Unsupervised hashing is useful when the data lacks explicit semantic labels or when semantic information is not readily available.

Supervised Hashing:

- Supervised hashing incorporates semantic information (Eg: class labels , user-defined similarities) during the learning process to generate binary codes.
- Supervised hashing methods utilize labeled data to learn a mapping that preserves the intrinsic structure but also respects the semantic relationships among data points. This often involves using techniques like deep neural networks or other machine learning models trained with supervision.
- Supervised hashing is particularly useful when semantic information is available and can enhance the quality of binary codes.

Approach 2: Hashing Based Retrieval

Supervised Hashing example:



Reference: [Supervised Hashing for Image Retrieval via Image Representation Learning - R Xia et al]

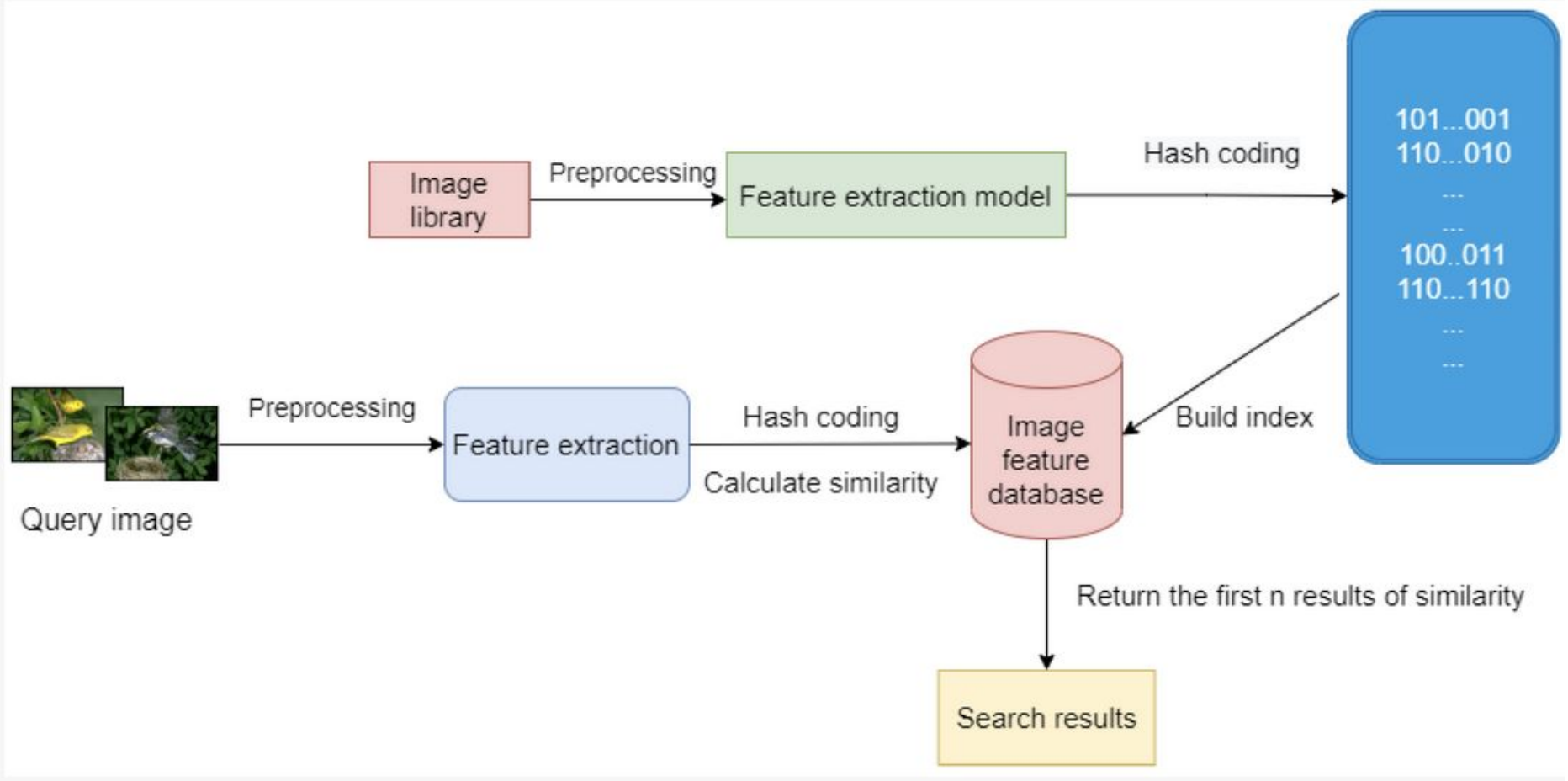
Approach 2: Hashing Based Retrieval

- In stage 1, the pairwise similarity matrix S is decomposed into a product HH^T ,where H is a matrix of approximate target hash codes.
- In stage 2, we use a convolutional network to learn the feature representation for the images as well as a set of hash functions.
- The network consists of three convolution-pooling layers, a fully connected layer and an output layer.
- The output layer can be simply constructed with the learned hash codes in H (the red nodes). If the image tags are available in training, one can add them in the output layer (the black nodes) so as to help to learn a better shared representation of the images.
- By inputting an test image to the trained network, one can obtain the desired hash code from the values of the red nodes in the output layer.

Approach 2: Hashing Based Retrieval

PES UNIVERSITY

Architecture



Approach 2: Hashing Based Retrieval

1. **Feature Extraction:**
    - Identify the features or characteristics in the image that are relevant to the detection task. These could be keypoints, descriptors, or any other distinctive features.
2. **Hashing Function:**
    - Design a hashing function that can encode the extracted features into a fixed-size hash code. The goal is to represent the essential information about the features in a compact form.
    - Traditional hash functions, like random projections or cryptographic hashes, are often manually designed, but ML allows for the automatic learning of hash functions from data.

Approach 2: Hashing Based Retrieval

- ML models, particularly deep learning architectures, can be trained to directly generate hash codes that capture intricate patterns and semantic similarities within the data.
- For instance, deep hashing models employ neural networks to map input data into a hash space where similar items yield similar hash codes.
- Supervised hashing leverages labeled data to guide the learning process, while unsupervised hashing methods, such as autoencoders or GANs, focus on learning hash codes without explicit labels.

3. **Database Construction:**
   - Create a database of hash codes for known objects or features. This involves processing a set of training images containing the target objects or features, extracting their features, and generating hash codes.

Approach 2: Hashing Based Retrieval

4. **Encoding the Query Image:**
   - Apply the same feature extraction process to the query image. Use the hashing function to generate a hash code for the features in the query image.
5. **Hash Code Comparison:**
   - Compare the hash code of the query image with the hash codes stored in the database. The goal is to find the closest matches based on hash code similarity.
   - One of the most common methods to do so is using Hamming distance. The Hamming distance measures the number of positions at which the corresponding bits are different.

Approach 2: Hashing Based Retrieval

The Hamming distance between two integers is the number of positions at which the corresponding bits are different. For example 20 (10100 in binary) and 17 (10001 in binary), the Hamming distance is 2 as there are two differing bits.

1	0	**1**	0	**0**

1	0	**0**	0	**1**

Approach 2: Hashing Based Retrieval

PES UNIVERSITY

Hashing Methods:

A. Data-Independent Methods

- The hash function design of the data-independent hashing method is independent of the data, and the hash function is generally generated by means of random mapping.
- The most typical representative method is the Locality-Sensitive Hashing (LSH) . The principle of locality-sensitive hashing is to map samples with high similarity in the original space to the same hash bucket with higher probability, which ensures that the hash codes of the neighbor samples in the original space can be as close as possible after hash mapping.

Approach 2: Hashing Based Retrieval

- The random mapping matrix is independent of the data and is determined by the probability distribution.
- However, in large-scale data retrieval, because of the hash collision problem, a longer hash code is needed to ensure the precision of the retrieval, which brings additional time and computational overhead, and leads to a decrease in the recall rate.

B. **Spectral Hashing (SH)**

- Spectral hashing is a type of single-modal hashing technique designed to maintain the neighborhood relationships of data points in the original feature space even after they have been hashed which ensures that the hash code obtained by the mapping is compact and rich in information.

Approach 2: Hashing Based Retrieval

The algorithm works as follows:

1. **Principal Component Analysis (PCA)**: Find the principal components of the data using PCA.
2. **Calculate eigenfunctions**: Calculate the k smallest single-dimensional analytical eigenfunctions of Lp using a rectangular approximation along every PCA direction. This is done by evaluating the eigenvalue equation to obtain an eigenvector solution
3. **Fit a multidimensional rectangle**: The aspect ratio of this multidimensional rectangle determines the code using a simple formula

Reference:
https://people.csail.mit.edu/torralba/publications/spectralhashing.pdf

Approach 2: Hashing Based Retrieval

PES UNIVERSITY

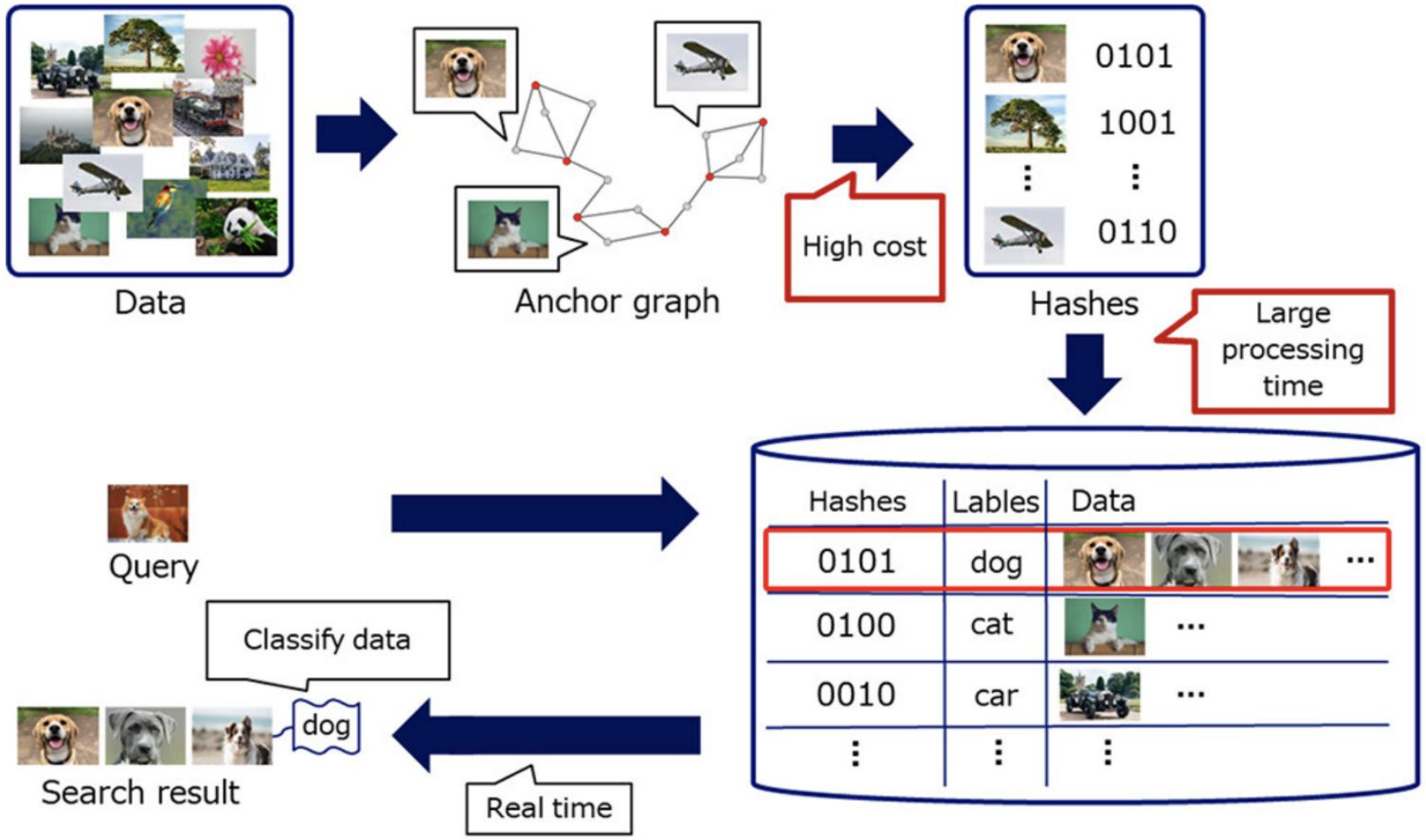
C. Anchor Graph Hashing (AGH)

- It is a single-modal graph-based hashing method whose principle is similar to the spectral hashing, but it does not require the precondition of uniform data distribution.
- The method first selects the points in the dataset (generally using the K-means clustering) as an anchor point to approximate the similarity matrix, and converts the similarity between any two data sample points into a sample-anchor relationship.
- The anchor graph hashing method is based on graph analysis and has strong scalability. It also uses a double hash function to generate a multidimensional hash code to solve the problem of uneven information content in the feature vector generated by the original data.

Approach 2: Hashing Based Retrieval

PES UNIVERSITY

AGH Architecture



Approach 2: Hashing Based Retrieval

D. Cross-View Hashing (CVH)

- The Cross-View Hashing is a multi-modal extension of the spectral hashing.
- The basic idea is to learn the hash function by minimizing the weighted average Hamming distance of different modalities and use the generalized eigenvalue solution method to obtain the minimum value.
- CVH can be applied to the data retrieval of multiple modalities, but the differences between modalities are not fully considered, hence the retrieval performance is limited.
- CVH is an unsupervised method. It uses graphs to describe the similarity between modalities. It can also use the label information to solve the similarity matrix and convert it into a supervised method.

Approach 2: Hashing Based Retrieval

PES UNIVERSITY

Advantages:

- Fast retrieval
- Space efficient
- Scalable
- Suitable for parallelism

Disadvantages:

- Collision Risks
- Sensitivity to small changes: Even a slight modification in an item can result in a significantly different hash code, impacting the effectiveness of similarity search.
- Some hash functions lack semantic relationships and can limit their ability to capture complex relationships.

References

- https://cdn.aaai.org/ojs/8952/8952-13-12480-1-2-20201228.pdf
- W. Cao, W. Feng, Q. Lin, G. Cao and Z. He, "A Review of Hashing Methods for Multimodal Retrieval," in IEEE Access, vol. 8, pp. 15377-15391, 2020

PES UNIVERSITY

THANK YOU

Surabhi Narayan

Department of Computer Science Engineering