IR U6

Spoken Language Audio Retrieval

Spoken Language Audio Retrieval refers to the process of retrieving relevant audio documents that contain human speech, using information retrieval techniques. In this type of MIR (Multimedia Information Retrieval), the audio is first converted into a searchable text representation or acoustic features so that users can search using keywords or spoken queries.

Steps in Spoken Audio Retrieval

(a) Speech Recognition (ASR)

- Automatic Speech Recognition (ASR) converts speech audio into text.
- ASR output is used for indexing.

(b) Indexing

- The transcribed text is indexed using
  - **Inverted indexes**,
  - **TF–IDF**,
  - **Keyword-based indexing**.
- This allows fast matching with user queries.

(c) Phonetic Indexing (Alternative Approach)

- Instead of full words, the audio is converted into **phoneme sequences**.
- Helps handle accents, noise, and mispronunciation more effectively.

(d) Audio Segmentation

- Long audios are divided into smaller meaningful units:
  - Sentences
  - Conversational turns
  - Speaker segments
- Improves retrieval accuracy.

(e) Feature Extraction

- MFCC (Mel Frequency Cepstral Coefficients) and other speech features help identify speech patterns.
- Used during speech recognition and matching.

(f) Matching and Ranking

- Query terms are matched with indexed transcripts.
- Results ranked using:
  - Term frequency
  - ASR confidence score
  - Acoustic similarity

4. Applications

- Call center conversation monitoring
- Lecture and meeting search
- Podcast indexing
- Digital libraries
- Voice-based assistants (e.g., Siri, Alexa)

5. Challenges

- Background noise
- Speaker variability (age, gender, accent)
- Multilingual speech
- ASR errors reduce retrieval accuracy

Non-Speech Audio Retrieval

Non-Speech Audio Retrieval refers to retrieving audio files that do **not** contain human speech. These sounds include **music, animal sounds, environmental noises, alarms, machine sounds**, and many other acoustic signals. Unlike spoken language retrieval, this system focuses purely on **acoustic features** because text-based IR techniques cannot be directly applied.

2. Types of Non-Speech Audio

- **Music** (songs, instrument tones)
- **Environmental sounds** (rain, thunder, wind)
- **Biological sounds** (bird calls, animal noises)
- **Mechanical/Industrial sounds** (engine noise, alarms)
- **Sound effects** (footsteps, explosions)

3. Retrieval Process

(a) Feature Extraction (Core Step)

Non-speech audio is represented using numerical features that describe sound characteristics:

- **Pitch & Frequency**
- **Timbre**
- **Rhythm & tempo**
- **Energy & loudness**
- **MFCC (Mel-Frequency Cepstral Coefficients)**
- **Zero Crossing Rate**
- **Spectral Centroid, Spectral Flux**

These features allow the system to uniquely represent each sound.

(b) Feature Vector Representation

Extracted features are stored as **feature vectors**, which help in comparing and matching sounds.

(c) Query Types

1. **Keyword Search** – e.g., “rain sound”, “dog barking”
2. **Query by Example** – user uploads a sample audio
3. **Humming-based Search** – used in music information retrieval

(d) Similarity Matching

The system compares the query’s feature vector with database vectors using:

- **Euclidean Distance**
- **Cosine Similarity**
- **Dynamic Time Warping (DTW) for music**

Higher similarity → higher ranking in results.

(e) Classification & Clustering

- **Classification** (SVM, Neural Networks, k-NN) → identifies sound type
- **Clustering** (k-means, hierarchical) → groups similar sounds
    - Example: nature sounds cluster, machine sounds cluster

4. Applications

- **Music Information Retrieval (MIR)**
- **Environmental sound monitoring** (forests, traffic, weather)
- **Smart surveillance systems** (detect alarms, gunshots)
- **IoT-based devices** (sound-triggered automation)
- **Digital libraries of sound effects** (movies, games)
- **Healthcare** (cough sound analysis, heart sound detection)

5. Challenges

- **Overlapping sounds** (traffic + people + music)
- **High background noise**
- **Difficulty describing sounds using text**
- **Large intra-class variation** (dog bark varies across dogs)
- **Ambiguity in user queries**

Graph Retrieval – 9 Marks (Structured Answer)

Graph Retrieval refers to retrieving information that is stored and represented in the form of **graph structures**. A graph contains:

- **Nodes (vertices)** → represent entities
- **Edges** → represent relationships between entities

In many modern IR applications, data naturally forms complex networks such as **social networks, knowledge graphs, biological networks, and web graphs**. Graph retrieval focuses on locating relevant nodes, edges, or subgraph patterns based on a user query.

2. Graph Data Representation

Graph data is typically stored using the following structures:

(a) Adjacency List

Stores each node along with a list of connected nodes. Efficient for sparse graphs.

(b) Adjacency Matrix

Represents graph connections in matrix form. Suitable for dense graphs.

(c) Graph Databases

Specialized systems like **Neo4j, OrientDB, Amazon Neptune** provide fast graph queries using languages like Cypher and Gremlin.

3. Types of Graph Queries

Users can express queries in several forms:

- **Node queries** – find nodes with certain properties
- **Edge queries** – retrieve relationships
- **Path queries** – find shortest or specific paths
- **Subgraph queries** – match a smaller graph pattern inside a larger graph
- **Neighborhood queries** – e.g., “find friends-of-friends within 2 hops”

Example: Searching for all nodes connected to “A” within two levels.

4. Techniques Used in Graph Retrieval

(a) Graph Traversal Algorithms

- **BFS (Breadth First Search)** – level-wise exploration
- **DFS (Depth First Search)** – depth-based exploration

(b) Shortest Path Algorithms

- **Dijkstra’s Algorithm**
- **Bellman–Ford**

These are used for path-based queries.

(c) Subgraph Matching / Graph Isomorphism

- Used to check if a query graph structure matches part of the stored graph.
- Computationally expensive (often NP-Complete).

(d) Graph Indexing

Indexes like graph fingerprints, node labels, or structural signatures are stored to speed up retrieval.

(e) Ranking Techniques

Ranking of graphs or nodes is done using:

- **PageRank** (importance of nodes)
- **Betweenness Centrality** (nodes connecting communities)
- **Closeness Centrality**
- **Semantic similarity** (in knowledge graphs)

5. Applications of Graph Retrieval

Graph retrieval is widely used in modern IR systems:

1. **Social Network Analysis** – finding influencers, friend recommendations
2. **Fraud Detection** – detecting suspicious transaction patterns
3. **Recommendation Systems** – item-to-item or user-to-user linking
4. **Biological Networks** – retrieving protein–protein interactions, gene pathways
5. **Knowledge Graph Search** – used in Google Knowledge Graph, Semantic Web
6. **Web Structure Mining** – analyzing hyperlinks and page relationships

6. Challenges in Graph Retrieval

- **NP-completeness** of subgraph matching
- **Large graph size** (millions of nodes)
- **Dynamic updates** (social networks change frequently)
- **Noise and missing data** in real-world networks
- **High computational cost** for complex queries

Imagery Retrieval (Image Retrieval)

Imagery Retrieval, also known as **Image Retrieval**, refers to the process of retrieving relevant images from a large image database using different types of user queries. It is a key component of **Multimedia Information Retrieval (MIR)**.

2. Types of Image Queries

(a) Text-Based Image Retrieval (TBIR)

- User provides keywords such as “sunset”, “mountain”, “car”.
- System uses **captions, metadata, tags, or annotations**.
- Modern systems use automatic **image captioning models** to generate metadata.

(b) Content-Based Image Retrieval (CBIR)

- Users provide an example image or sketch.
- Retrieval is based on **visual features**, not text.

(c) Feature-Based Queries

- Search is performed using extracted image features stored as numeric values.

3. Content-Based Image Retrieval (CBIR) Process

(a) Feature Extraction

Images are converted into feature vectors using:

- **Color Features** → histograms, color moments
- **Texture Features** → Gabor filters, LBP
- **Shape Features** → edges, contours
- **Deep Learning Features** → CNN embeddings (ResNet, VGG)

These features capture the visual characteristics of the image.

(b) Indexing

Extracted features are stored in feature databases for fast retrieval.

(c) Similarity Matching

Similarity between query image and stored images is measured using:

- **Euclidean Distance**
- **Cosine Similarity**
- **Neural network–based similarity**

Higher similarity = higher ranking.

(d) Ranking

Images are ranked based on:

- Feature similarity

- Relevance feedback
- Semantic classification using deep learning

4. Applications of Imagery Retrieval

1. **Medical Image Search** (CT, MRI comparison)
2. **Biometric Identification** (face, fingerprint retrieval)
3. **E-commerce Visual Search** (search by image in shopping apps)
4. **Satellite Image Analysis** (land use, weather detection)
5. **Security and Surveillance** (object or person retrieval)
6. **Digital Art and Multimedia Archives**

5. Challenges

- **High image variation** in scale, rotation, and background
- **Lighting and viewpoint changes** affect feature extraction
- **Semantic gap** between low-level features and human interpretation
- **Large database size** requiring efficient indexing

Video Retrieval

1. Introduction

Video Retrieval refers to the process of retrieving relevant videos or specific video segments based on user queries. A video is a complex multimedia object because it contains **three modalities**:

- **Visual frames**
- **Audio tracks**
- **Textual metadata**

Effective retrieval requires analyzing all these components together.

2. Components Used in Video Retrieval

(a) Visual Component

- Frames contain color, texture, shape, and object information.
- Keyframes are used to represent scenes efficiently.

(b) Audio Component

- Speech transcripts (ASR)
- Background sounds and music
- Helps identify events like cheering, alarms, or conversations.

(c) Text Metadata

- Titles
- Descriptions
- Tags
- Subtitles and closed captions

These support **text-based video search**.

3. Video Retrieval Process

(a) Video Segmentation

- Video is divided into **shots** and **scenes**.
- Makes retrieval faster and more meaningful.

(b) Keyframe Extraction

- Representative frames selected from each scene.
- Used for indexing and comparison.

(c) Feature Extraction

Visual Features:

- Color histogram
- Texture patterns
- Edge/shape features
- Deep learning features (CNN embeddings)

Audio Features:

- MFCC
- Speech transcripts
- Sound patterns

Text Features:

- Metadata
- Captions/subtitles

(d) Content-Based Video Retrieval (CBVR)

Users search using:

- Example images
- Short video clips
- Sketches

(e) Semantic Video Retrieval

Identifies high-level concepts such as:

- “Goal moment”
- “Car chase”

- “News anchor speaking” Deep learning models enable semantic understanding.

(f) Ranking of Results

Ranking is done using **multimodal fusion**, which combines:

- Visual similarity
- Audio similarity
- Text similarity

4. Applications

- YouTube and OTT platform video search
- Surveillance and CCTV video analysis
- Sports highlights detection
- Movie and advertisement retrieval
- Medical and surgical video archives
- Educational video and lecture indexing

5. Challenges

- Very large storage requirements
- High computational cost
- Frequent scene transitions
- Noisy or unclear audio
- Object occlusion and fast motion
- “Semantic gap” between raw features and meaning

Recommender System

A **Recommender System** is an information filtering technique used to suggest items (movies, products, songs, courses, etc.) to users based on their preferences and behavior. It helps reduce information overload by predicting what a user is likely to prefer.

Recommender systems mainly use three approaches:

1. **Collaborative Filtering**
2. **Content-Based Recommendation**
3. **Knowledge-Based Recommendation**

1. Collaborative Filtering

Collaborative Filtering (CF) is one of the most widely used recommendation techniques. It makes predictions about a user’s preferences based on the **preferences of similar users** or **similar items**.
The core assumption is:
“Users who behaved similarly in the past will have similar preferences in the future.”

2. Types of Collaborative Filtering

(a) User-Based Collaborative Filtering

- Identifies users who have similar tastes to the target user.
- Builds a neighborhood of “similar users.”
- Recommends items liked or rated highly by those similar users.
  Example:
  If User A and User B rate similar movies, movies liked by B can be recommended to A.

(b) Item-Based Collaborative Filtering

- Focuses on similarity between items.
- If a user likes a particular item, the system finds other items that are similar.
  Example:
  If a user likes Item X, then Item Y and Z (similar to X) are recommended.

3. Techniques Used

- **Cosine Similarity:** Measures similarity between user or item vectors.
- **Pearson Correlation:** Captures the correlation between rating patterns.
- **Matrix Factorization (SVD):** Reduces dimensionality and handles sparse data efficiently.

4. Advantages

- Does not require item metadata or descriptions.
- Captures real community behavior.
- Can recommend diverse items beyond the user’s history.

5. Limitations

- **Cold Start Problem:** New users/items lack enough ratings.
- **Sparsity:** Most users rate very few items, causing sparse rating matrices.
- **Scalability:** Heavy computation for large datasets.

6. Applications

- Netflix movie recommendation
- Amazon “Customers who bought this also bought…”
- Spotify music recommendations

2. Content-Based Recommendation

Content-Based Recommendation recommends items that are **similar to items the user has liked in the past**. It uses item features such as genre, category, keywords, or product descriptions.

2. How Content-Based Systems Work

(a) Item Feature Extraction

Features are extracted depending on the domain:

- Movies → genre, actors, director
- Products → specifications, brand, description
- Text → TF-IDF, keywords, embeddings

(b) User Profile Creation

- A model of the user's preferences is built.
- Example: The system concludes that the user prefers *action* + *sci-fi* movies.

(c) Matching & Recommendation

- Compare item features with the user profile using similarity measures.
- Items with the highest similarity scores are recommended.
- Common measure: **Cosine similarity**

3. Advantages

- Works well even when there are no ratings from other users.
- Provides personalized, user-specific recommendations.
- No cold start for items (only user needs history).

4. Limitations

- Limited to the user's past behavior (no exploration).
- Cannot introduce new or unexpected items.
- Item feature extraction may be complex (especially for images/videos).

5. Applications

- YouTube video recommendations based on watch history
- Job recommendations on LinkedIn
- News article recommendations

3. Knowledge-Based Recommendation

Knowledge-Based Recommendation suggests items using **explicit knowledge** about user requirements and item characteristics.
Unlike collaborative or content-based systems, it does **not** depend on past ratings or user history.

2. How Knowledge-Based Systems Work

(a) Rules and Constraints

The system applies domain-specific rules such as:

- "Show laptops under ₹50,000 with 8GB RAM."
- "Recommend hotels near the beach with free Wi-Fi."

(b) Requirement Matching

The system matches the user's requirements with product attributes.

(c) Case-Based Reasoning

- Past successful recommendations are stored as cases.
- New cases are matched with similar old cases to recommend items.

3. Types of Knowledge-Based Systems

(i) Constraint-Based Recommendation

- Works by defining clear user constraints.
- Example: Real estate, cars, electronics.

(ii) Case-Based Recommendation

- Compares current user needs with past user cases.
- Example: Suggesting laptops based on similar previous purchases.

4. Advantages

- Works well for **rare or expensive items** (houses, cars, travel packages).
- No cold-start problem.
- Highly accurate when user requirements are clear.

5. Limitations

- Requires a large amount of domain knowledge.
- Difficult and costly to build.
- Complex to maintain rules and cases.

6. Applications

- Travel recommendation systems
- Automobile selection tools
- Real estate property recommendation
- Medical decision-support systems

1. Extracting Data from Text

Extracting data from text refers to converting **unstructured text** into **structured, machine-readable information**. Since most web content, documents, and reports exist in natural language, text extraction is a crucial step in information retrieval and data mining.

2. Need for Text Extraction

- Most digital data is in textual form.
- Organizations need structured information for analytics.
- Helps in automation, decision-making, and search optimization.

3. Text Extraction Process

(a) Text Preprocessing

- Tokenization
- Stop-word removal
- Stemming/Lemmatization
- Normalization

Preprocessing prepares raw text for analysis.

(b) Feature Extraction

- **Bag of Words (BoW)**
- **TF-IDF**
- **Word Embeddings** (Word2Vec, GloVe)
- **Sentence Embeddings**

These convert text into numeric features.

(c) Named Entity Recognition (NER)

Extracts entities such as people, places, organizations, dates, currencies.

(d) Information Extraction (IE)

- Identifies relations, events, facts
- Example: From "Amazon acquired Whole Foods," extract relation: *Acquisition*

(e) Text Classification

- Categorizes documents (spam/ham, sports/politics)

(f) Text Summarization

- Extractive or abstractive summaries
- Converts large content into concise information

4. Techniques Used

- Natural Language Processing (NLP)
- Regular expressions
- Machine Learning (SVM, Naive Bayes)
- Deep Learning (LSTM, Transformers)

5. Applications

- Search engines
- Chatbots
- Resume parsing
- Medical report extraction
- Social media analytics

6. Challenges

- Ambiguity in natural language
- Synonyms and context understanding
- Spelling variations
- Noisy or informal text (slang, social media)

2. Semantic Web

The Semantic Web is an extension of the existing web where information is given **well-defined meaning** so that both humans and machines can understand it. It allows computers to interpret data logically, enabling intelligent search and automation.

2. Goals of the Semantic Web

- Make web content machine-understandable
- Improve accuracy of search engines
- Enable data integration across websites
- Support intelligent applications and agents

3. Key Technologies in Semantic Web

(a) RDF (Resource Description Framework)

- Represents data as **subject–predicate–object** triples
- Forms the foundation of machine-readable data

(b) OWL (Web Ontology Language)

- Provides vocabulary to define classes, properties, relationships
- Allows reasoning and inference

(c) SPARQL

- Query language for RDF data
- Similar to SQL but used for semantic web databases

(d) Ontologies

- Structured knowledge models that define concepts and relationships
- Example: Medical ontology describing diseases, symptoms, treatments.

4. How Semantic Web Works

- Websites publish data with semantic markup (RDF/OWL).
- Intelligent agents crawl the web and interpret meaning.
- SPARQL engines retrieve relevant structured data.

Example: Searching for "movies directed by Nolan" retrieves precise results because data is semantically connected.

5. Applications

- Intelligent search engines
- Personal assistants (Siri, Alexa use semantic knowledge graphs)
- Healthcare decision systems
- E-commerce product linking
- Academic knowledge graphs (Google Scholar)

6. Challenges

- High effort required to create ontologies
- Integration issues with existing unstructured web
- Scalability and reasoning cost
- Lack of universal standards for all domains

3. Collecting and Integrating Specialized Information on the Web

The web contains vast and diverse data spread across multiple sites, formats, and structures. Collecting and integrating specialized information refers to gathering **domain-specific data** (medical, financial, scientific, educational, etc.) and combining it into a unified, usable format.

2. Need for Specialized Information Integration

- Data is scattered across multiple sources
- Formats vary: HTML, XML, JSON, PDFs
- Essential for analytics, decision-making, and research
- Enables domain applications (e.g., medicine, agriculture, finance)

3. Process of Collecting & Integrating Information

(a) Web Crawling & Scraping

- Automated agents collect data from web pages
- Extract text, tables, product details, reviews, etc.

(b) Data Cleaning

- Remove noise, ads, duplicates, inconsistencies
- Standardize units, formats, and naming

(c) Data Transformation

- Convert various formats into a unified structure
- Use schemas, ontologies, APIs

(d) Data Integration

- Combine information from multiple sources
- Resolve conflicts, duplicates, and data mismatches
- Use integration techniques:
  - Schema matching
  - Entity resolution
  - Data fusion

(e) Storage in Structured Repositories

- Databases
- Data warehouses
- Knowledge graphs

4. Tools & Technologies

- APIs (REST, GraphQL)
- Web scrapers (BeautifulSoup, Scrapy)
- Semantic web technologies (RDF, OWL)
- ETL tools (Extract-Transform-Load)

5. Applications

- **Healthcare:** Integrating patient data, research papers
- **Finance:** Stock data aggregation
- **E-commerce:** Price comparison engines
- **Education:** Course and research database integration
- **Travel:** Flight and hotel aggregator websites

6. Challenges

- Inconsistent data formats
- Dynamic web pages and frequent updates
- Legal issues (copyright, scraping policies)
- Ambiguity in merging information
- Large computational costs