Advanced Topics in IR

Jan, 2025

Outline

9.1. Cross-lingual and multilingual IR

9.2. Image and Multimedia Retrieval

9.3. Generative Retrieval: Retrieval Augmented Generation (RAG) with LLM

Cross-Lingual and Multilingual Information Retrieval

- advanced topics such as **Cross-Lingual & Multilingual IR**, **Image & Multimedia Retrieval**, and **Generative Retrieval (RAG with LLMs)** are becoming essential for modern search systems.

...

Cross-Lingual Information Retrieval (CLIR) and Multilingual Information Retrieval (MLIR) focus on retrieving relevant information across different languages.

- **Cross-Lingual IR (CLIR)**: Users submit queries in one language, but results are retrieved from documents in different languages.

- **Multilingual IR (MLIR)**: Users can query in multiple languages, and the system searches across multilingual document collections.

How it works

- CLIR is often a combination of translation and traditional IR.
- Queries are usually translated from the user's language to the language of the documents.
- CLIR can involve dealing with different vocabularies, grammars, and semantics across languages.

Why it's useful

- CLIR can help users access information that is not available in their own language.
- CLIR can promote cultural and linguistic diversity.
- CLIR can be useful for reporters who want to search foreign language news.
- CLIR can be useful for inventors who want to explore patents in other countries.

Architecture if a CLIR

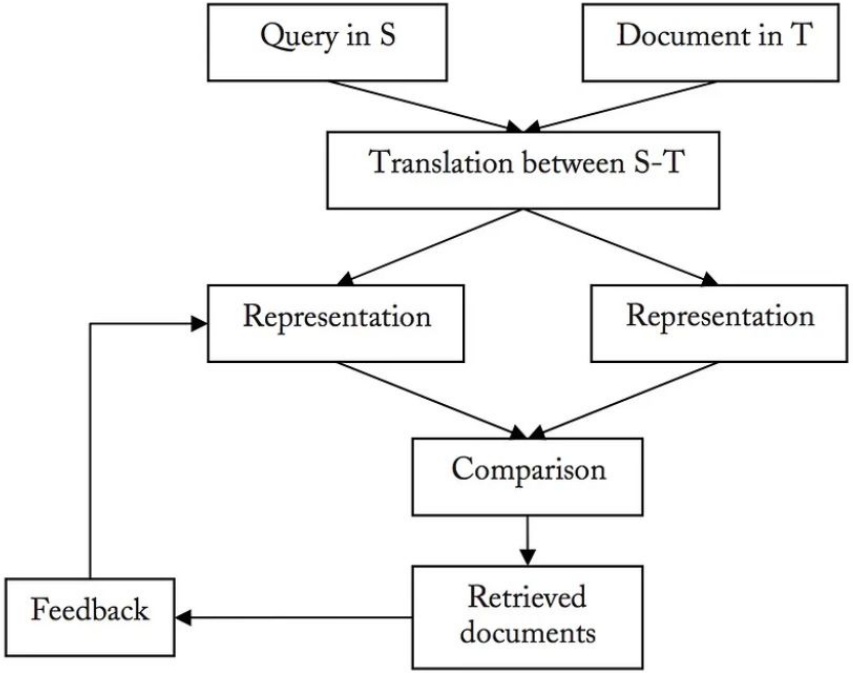


Figure 1. Typical architecture of a CLIR system [1].

Translation Approaches

- CLIR requires the ability to represent and match information in the same representation space even if the query and the document collection are in different languages.

  - The fundamental problem in CLIR is to match terms in different languages that describe the same or a similar meaning. The strategy of mapping between different language representations is usually machine translation.

- In CLIR, this translation process can be in several ways.
    - **Document translation** is to map the document representation into the query representation space, as illustrated in Figure 2.
    - **Query translation** is to map the query representation into the document representation space, as illustrated in Figure 3.
    - **Pivot language or Interlingua** is to map both document and query representations to a third space.

...

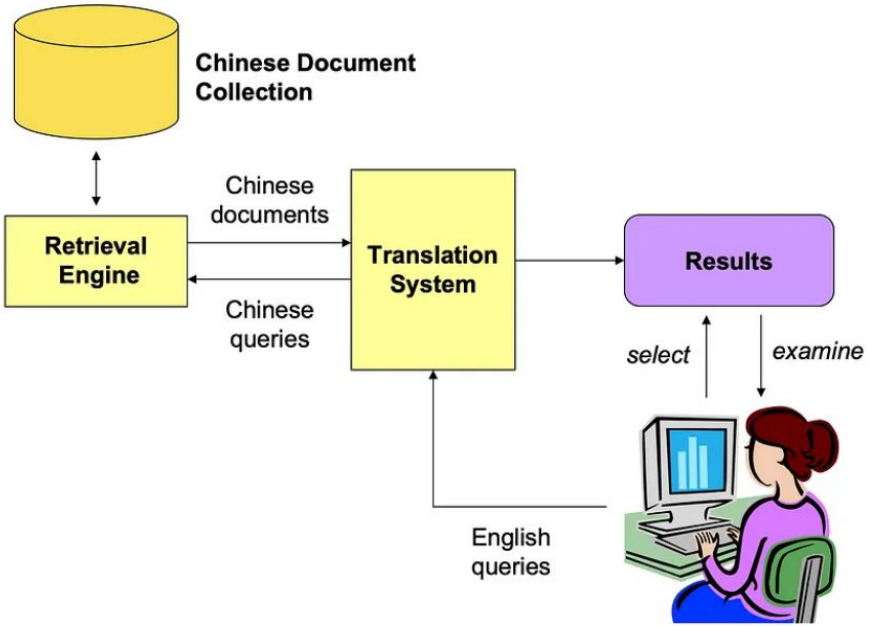


Figure 3. Query Translation for CLIR [8].

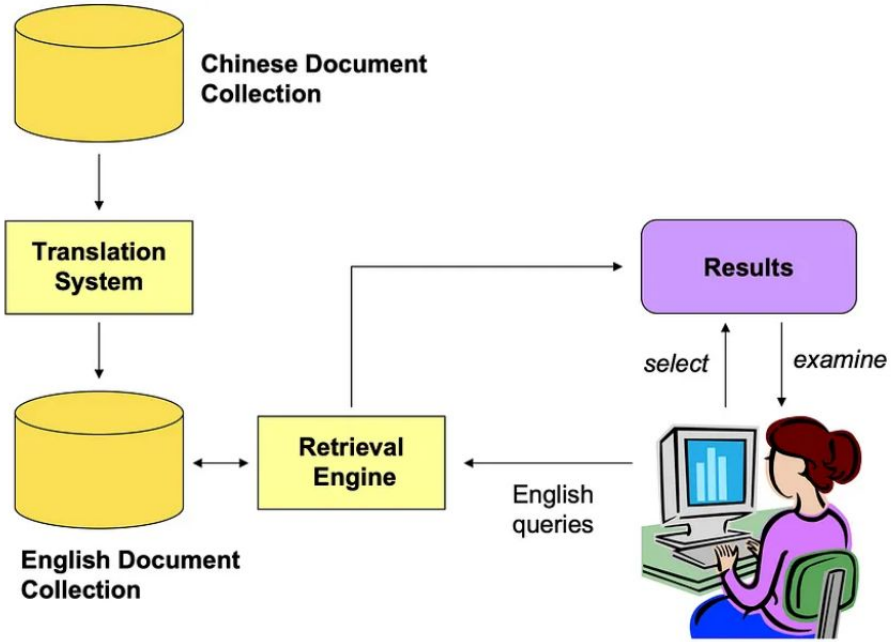


Figure 2. Document Translation for CLIR [8].

...

Query translation is generally considered the most appropriate approach.

The query is short and thus fast to translate than the document, and it is more flexible and allows more interaction with users if the user understands the translation.

- However, query translation can suffer from translation ambiguity, and this problem is even more obvious for the short query text due to the limited context.
- By contrast, document translation can provide more accurate translation thanks to richer contexts.
- Document translation also has the advantage that once the translated document is retrieved, the user can directly understand it, while the query translation still needs a post-retrieval translation.
- However, several experiments show that there is no clear evidence of one approach or the other using the same machine translation system, and the effectiveness is more dependent on the translation direction.

Recent Progress : https://github.com/ryanzhumich/awesome-clir

DUET

This is the paper *Learning to Match using Local and Distributed Representations of Text for Web Search, WWW 2017*

by Bhaskar Mitra, Fernando Diaz, and Nick Craswell.

MUSE

The second paper is *Word translation without parallel data, ICLR 2018* by Alexis Conneau, Guillaume Lample,

Marc'Aurelio Ranzato, Ludovic Denoyer, Hervé Jégou.

Unsupervised CLIR

The third paper is *Unsupervised Cross-Lingual Information Retrieval using Monolingual Data Only, SIGIR 2018* by

Robert Litschko, Goran Glavaš, Simone Paolo Ponzetto, Ivan Vulić.

Multilingual Information Retrieval (MLIR) is the ability to process queries in multiple languages and retrieve relevant objects (text, images, sound files, etc.), translating them if necessary into the user's language.

- MLIR is crucial in a world where information is available in multiple languages across different media formats, especially on the World Wide Web.

Unlike traditional **Information Retrieval (IR)**, which primarily deals with monolingual searches, MLIR expands into:

- **Image, video, speech, and multimedia retrieval**
- **Structured data access** (such as MARC library records)
- **Access across multiple languages**

Key Considerations:

- **Machine Translation (MT)** and **Multimedia Processing** are related but typically not included under MLIR.
- **Descriptive catalogue data** and **unstructured data** are both part of MLIR.
- **Linguistic and cultural differences** play a role in search behavior.

Categories of MLIR

Hull and Grefenstette (1996) define **five categories** of MLIR:

1. **Monolingual IR in languages other than English**
   - Conducting IR in a **single non-English** language.
   - Example: Searching for Spanish documents in a Spanish-only database.
2. **Multilingual IR with a monolingual query (Cross-Lingual IR - CLIR)**
   - The query is entered in one language, but documents exist in **multiple languages**.
   - The system **translates the query** and treats each language separately.
   - Example: A user searches in **English**, but retrieves documents in **French, German, and Hindi**.
3. **Monolingual document collection with multilingual queries**
   - The document collection is in **one language**, but queries are entered in **multiple languages**.
   - Queries are **translated into the document's language**.
   - Example: A search engine indexing only **Chinese** documents but allowing queries in **English, Nepali, and Hindi**.

..

4. Fully Multilingual IR

- Queries can be in **any language**, and retrieved documents may be in **multiple languages**.
- Example: A user submits a **French query** and retrieves documents in **French, English, and Arabic**.

5. IR on Multilingual Documents

- Some documents themselves contain **multiple languages**.
- Example: A legal document in **English** with embedded **Nepali and Hindi** quotes.

Different MLIR Use Cases

- Monolingual IR in non-English languages.
- Multilingual document collections queried in one or multiple languages.
- Retrieving multilingual or hybrid-language documents.

MLIR vs. IR

- Both share common IR techniques (vector space indexing, LSI, etc.).
- MLIR introduces challenges such as translation tracking and cross-lingual similarity matching.

MLIR vs. MT

- MT requires deep linguistic analysis and word sense disambiguation.
- MLIR often does not need full document translation; keyword-based translation can suffice.
- Queries are often incomplete and lack syntactic structure, making direct MT less effective.

...

Key Challenges in MLIR

- **Indexing**: Should documents be indexed separately by language, or all together? Should the indexed material be aligned, cross-indexed, or independent across languages?
- **Query treatment**: Should disambiguation proceed monolingually or multilingually? Should query term expansion be performed monolingually, multilingually, or both?
- **Cross-language document ranking**: How must documents retrieved in different languages by different retrieval processes be compared? If they contain the same information, how should they be merged?
- **Feedback processing**: How should the user's selection of relevant documents and/or passages be propagated to other languages?

MLIR **cannot simply be achieved by coupling IR and MT**—instead, they must be **tightly integrated**, allowing each system to leverage intermediate processing results of the other.

- Future MLIR systems will likely involve **deep fusion** between IR and MT components for better efficiency and accuracy.

Image and Multimedia Retrieval

Traditional IR systems focus on text-based retrieval, but **Image and Multimedia Retrieval** extends this to images, videos, and audio content.

Challenges

- **Feature Representation**: How to represent images and videos for retrieval.
- **Semantic Gap**: The difference between how humans perceive images and how machines process them.
- **Metadata Dependency**: Some systems rely heavily on text-based metadata instead of analyzing multimedia content directly.

Techniques for Image & Multimedia Retrieval

1. **Text-Based Image Retrieval (TBIR)**
   - Uses metadata (captions, tags, file names) for retrieval.
   - Simple but **not scalable** for large datasets.
2. **Content-Based Image Retrieval (CBIR)**
   - Extracts **visual features** like color, shape, texture.
   - Uses models like **ResNet, VGG, ViT (Vision Transformers)** for feature extraction.
3. **Deep Learning-Based Retrieval**
   - Uses deep neural networks (CNNs, Vision Transformers) to encode images into vector embeddings.
   - Enables **semantic search** (e.g., "Find all images of cats").
4. **Multimodal Retrieval**
   - Combines text, image, and audio features for better search.
   - Uses **CLIP (Contrastive Language–Image Pretraining)** to align images with textual descriptions.

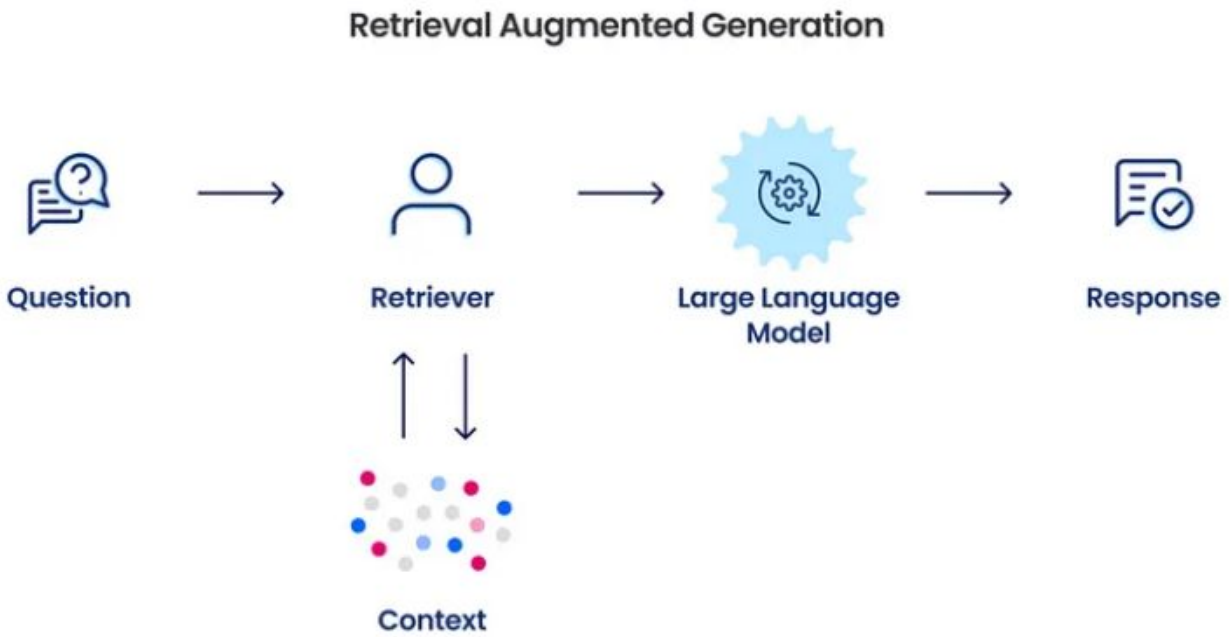
Generative Retrieval: Retrieval-Augmented Generation (RAG) with LLMs

- **Retrieval-Augmented Generation (RAG)** combines **retrieval** and **generation** by enhancing LLMs with relevant external knowledge.
- Unlike traditional IR, which **retrieves** documents, and traditional LLMs, which **generate** text from their training data, **RAG dynamically retrieves relevant context** before generating responses.

Why Use RAG?

- **Reduces Hallucinations**: Ensures LLMs rely on real, retrieved data instead of making up answers.
- **Improves Answer Quality**: Provides **up-to-date** and **factual** responses.
- **Scalability**: Works well for large knowledge bases and enterprise search.

RAG


Retrieval Augmented Generation
Question
Retriever
Large Language Model
Response
Context

Key Components of RAG

Retriever

- The retriever’s job is to find relevant documents or pieces of information that can help answer a query.
- It takes the input query and searches a database to retrieve information that might be useful for generating a response.

Types of Retrievers:

- Dense Retrievers: These use neural network-based methods to create dense vector embeddings of the text.
  - They tend to perform better when the meaning of the text is more important than the exact wording since the embeddings capture semantic similarities.

...

- **Sparse Retrievers:**
  - These rely on term-matching techniques like TF-IDF or BM25.
  - They excel at finding documents with exact keyword matches, which can be particularly useful when the query contains unique or rare terms.
    i. BM25 (Sparse Retrieval)
    ii. DPR (Dense Passage Retrieval)
    iii. FAISS (Fast Approximate Nearest Neighbor Search)

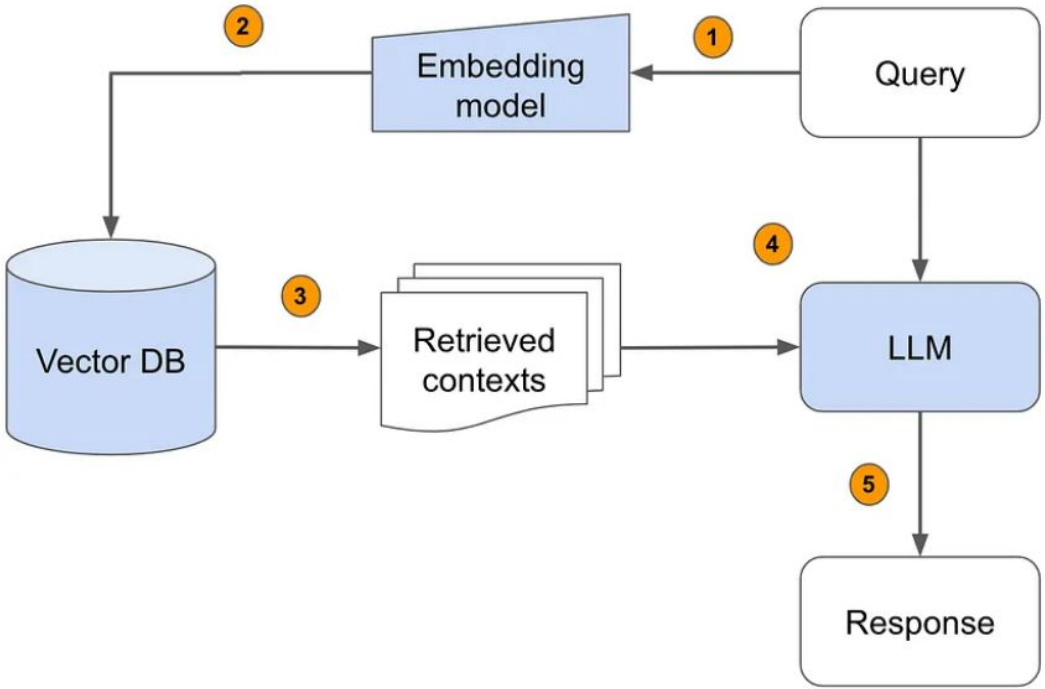
...

Generator Component:

- **Function:** The generator is a language model that produces the final text output.
  - It takes the input query and the contexts retrieved by the retriever to generate a coherent and relevant response.
- **Interaction with Retriever:** The generator doesn't work in isolation;
  - it uses the context provided by the retriever to inform its response, ensuring that the output is not just plausible, but also rich in detail and accuracy.
    - Large Language Models (GPT, T5, LLama)
    i. Uses retrieved documents as input

2. **Hybrid RAG (Dense + Sparse)**
   - Combines **BM25 + Dense Retrieval** for better results.

Workflow of a RAG system



...

- **Query Processing:** It all starts with a query.
  - This could be a question, a prompt, or any input that you want the language model to respond to.
- **Embedding Model:** The query is then passed to an embedding model.
  - This model converts the query into a vector, which is a numerical representation that can be understood and processed by the system.
- **Vector Database (DB) Retrieval:** The query vector is used to search through a vector database.
  - This database contains precomputed vectors of potential contexts that the model can use to generate a response. The system retrieves the most relevant contexts based on how closely their vectors match the query vector.

...

- **Retrieved Contexts:** The contexts that have been retrieved are then passed along to the Large Language Model (LLM).
  - These contexts contain the information that the LLM uses to generate a knowledgeable and accurate response.
- **LLM Response Generation:** The LLM takes into account both the original query and the retrieved contexts to generate a comprehensive and relevant response.
  - It synthesizes the information from the contexts to ensure that the response is not only based on its pre-existing knowledge but is also augmented with specific details from the retrieved data.
- **Final Response:** Finally, the LLM outputs the response, which is now informed by the external data retrieved in the process, making it more accurate and detailed.

In short,

1. **User Query → Retrieve Relevant Documents**
   - The query is first passed through a **retrieval model** (e.g., BM25, Dense Retriever).
2. **Retrieve Top-K Passages**
   - The retrieval model fetches the **top K** most relevant documents.
3. **Pass Retrieved Context to LLM**
   - These documents are passed as additional context to an LLM like **GPT, Llama, or Falcon**.
4. **Generate Answer Using LLM**
   - The LLM generates a response using the retrieved documents.

Challenges in Implementing RAG:

Complexity: Combining retrieval and generation processes adds complexity to the model architecture, making it more challenging to develop and maintain.

Scalability: Managing and searching through large databases efficiently is difficult, especially as the size and number of documents grow.

Latency: Retrieval processes can introduce latency, impacting the response time of the system which is critical for applications requiring real-time interactions, like conversational agents.

Synchronization: Keeping the retrieval database up-to-date with the latest information requires a synchronization mechanism that can handle constant updates without degrading performance.

Limitations of Current RAG Models:

Context Limitation: RAG models may struggle when the context required to generate a response exceeds the size limitations of the model's input window.

Retrieval Errors: The quality of the generated response is heavily dependent on the quality of the retrieval step; if irrelevant information is retrieved, the generation will suffer.

Bias: RAG models can inadvertently propagate and even amplify biases present in the data sources they retrieve information from.

Applications of Retrieval-Augmented Generation (RAG)

1. Chatbots & Assistants

- **Customer Support:** Retrieves FAQs and documents for accurate responses.
- **Personal Assistants:** Fetches real-time data (weather, news) for relevant interactions.

2. Content Generation

- **Journalism & Copywriting:** Enhances articles and marketing copy with factual accuracy.

3. Question-Answering Systems

- **Education:** Provides detailed explanations using academic databases.
- **Research:** Summarizes scientific papers for quick insights.

4. Industry Applications

- **Healthcare:** Assists in diagnosis by retrieving medical research.
- **Customer Service:** Pulls company policies for personalized support.
- **Legal Aid:** Fetches case law for legal research.
- **Translation:** Enhances accuracy with bilingual corpora.

. . .

Some popular RAG Implementations

- **LlamaIndex** (Previously GPT Index)
- **Haystack (deepset.ai)**
- **LangChain** (Integrates retrieval with LLMs)

Any queries???