Information Retrieval

By
Dr Syed Khaldoon Khurshid

Lecture of the week

- **Probabilistic Retrieval Model**
- **Structured Text Retrieval Model**
  - **Overlapped Lists**
  - **Non-Overlapped Lists**
  - **Proximal Nodes**

Probabilistic Retrieval Model

Description 1:

The probabilistic retrieval model is based on the Probability Ranking Principle, which states that "an information retrieval system is supposed to rank the documents based on their probability of relevance to the query, given all the evidence available."

Probabilistic Retrieval Model

Description 2:

The Probabilistic Retrieval Model is a method used in Information Retrieval to estimate the probability that a document is relevant to a user's query based on statistical analysis. It helps rank documents by their likelihood of being useful to the user.

Simple Example

- Imagine you have **a vast collection of computer programs,** and a user searches for "**virus protection software.**" The Probabilistic Retrieval Model would help **rank these programs based on the probability that they are effective virus protection software.** It uses statistics to estimate how likely each program is to be **what the user needs**, helping you find the best antivirus software quickly.

Mathematical Formulation of the Probabilistic Retrieval Model

- **P(Relevance | Q, D) = 1 / (1 + e^(-s))**

Where:

- **P(Relevance | Q, D)** represents the probability that document D is relevant to query Q.
- **e** is the base of the natural logarithm (approximately 2.71828).
- **s** is a scoring function that calculates the similarity between the query and the document. This function typically involves factors like term frequency, document length, and collection statistics.
- The Probabilistic Retrieval Model computes the probability of relevance by modeling the relationship between the query and the document using this mathematical formula. **The document with the highest probability is considered the most relevant to the query.**

Mathematical Formulation of the Probabilistic Retrieval Model

- The Probabilistic Retrieval Model uses a mathematical formulation to estimate the probability of relevance between a document (D) and a user's query (Q). Here's a general mathematical representation:

- **P(Relevance | Q, D) = 1 / (1 + e^(-s))**

- **Sigmoid Function Shape**: The use of e^-s results in a sigmoid-shaped curve, which is a smooth, continuous function. Sigmoid functions are mathematically convenient and often used in probability and logistic models. This curve smoothly transitions from 0 to 1, making it suitable for modeling probabilities.
- **Normalization**: The negative exponent helps normalize the scoring function (s) into a probability distribution. By applying the exponent, the scoring function is transformed into a value between 0 and 1, which represents a probability. This normalization is essential for comparing documents based on their probability of relevance.

Mathematical Formulation of the Probabilistic Retrieval Model

- **Exponential Weights**: Exponentials are used to give greater importance to documents that closely match the query. When the scoring function s is large (indicating a strong match between the query and document), e^-s approaches 0, resulting in a high probability of relevance (close to 1). Conversely, when s is small, e^-s approaches 1, indicating a low probability of relevance (close to 0).
- **Probabilistic Interpretation**: The use of e^-s is aligned with the probabilistic interpretation of the model. It allows the model to estimate the probability of relevance based on the similarity between the query and the document. The sigmoid curve allows for intuitive probability calculations.
- Overall, the use of e^-s is a standard choice in probabilistic models like the Probabilistic Retrieval Model because it offers mathematical convenience, normalization, and a probabilistic interpretation of relevance.

Computed Example

- Let's create a simple numerical example using the Probabilistic Retrieval Model formula in a real-world context:
- **Scenario**: You are searching for a Smartphone online, and you want to know how likely each product is to meet your requirements based on your **query** for **"high-quality camera Smartphone."**
- You have three Smartphone listings, each with a relevance score (s) calculated based on various factors like camera quality, customer reviews, and price.
- 1. Smartphone A: s = 3.5
- 2. Smartphone B: s = 2.0
- 3. Smartphone C: s = 4.8

Computed Example (Continued...)

- Using the Probabilistic Retrieval Model formula:
- P(Relevance | Query, Smartphone) = 1 / (1 + e^(-s))
- Let's calculate the probability of relevance for each Smartphone:

1. For Smartphone A:

- - P(Relevance | Query, Smartphone A) =
  1 / (1 + e^(-3.5)) ≈ 0.9707
- The probability that Smartphone A is relevant to your query is approximately **97.07%.**

2. For Smartphone B:

- - P(Relevance | Query, Smartphone B) =
  1 / (1 + e^(-2.0)) ≈ 0.8808
- The probability that Smartphone B is relevant to your query is approximately **88.08%.**

Computed Example (Continued...)

3. For Smartphone C:

- - P(Relevance | Query, Smartphone C) =
  1 / (1 + e^(-4.8)) ≈ 0.9918
- The probability that Smartphone C is relevant to your query is approximately **99.18%.**
- Based on these calculated probabilities, Smartphone C has the highest probability of relevance to your query for a "high-quality camera Smartphone." Therefore, **Smartphone C** is the most likely to meet your requirements.

Binary Independence Model (BIM)

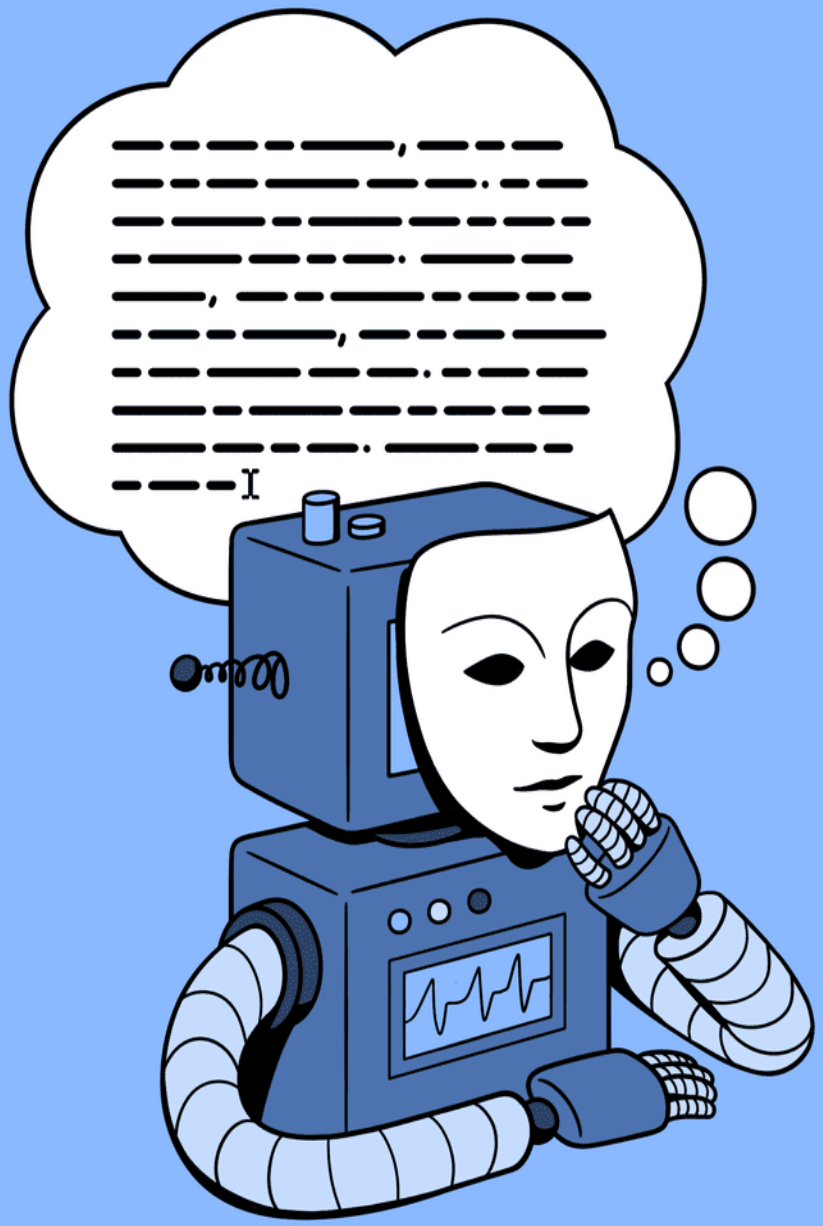
- The Binary Independence Model (BIM) in computing and information science is a **probabilistic information retrieval technique.**
- The Binary Independence Model (BIM) in information retrieval is a straightforward approach:
- **Binary Terms**: BIM treats each word or term in a document or query as either present (1) or absent (0). It doesn't consider how many times a word appears; it just cares if it's there or not.
- **Matching**: **When you search for something, BIM checks which words in your query are present in each document. It marks them as 1 if they are there, and 0 if they are not.**

Binary Independence Model (BIM)

- **Similarity:** BIM calculates how many words in your query match the words in each document. If more words match, the document gets a higher score. This means it's more likely to be what you're looking for.
- **Ranking:** BIM ranks the documents based on their scores. The ones with the most matching words rank higher.
- **Results:** Finally, BIM shows you the documents with the highest scores as your search results. These documents are considered more relevant because they match your query words the most.

More Probabilistic Models:

- Models like the **Okapi BM25** and **language models like Latent Dirichlet Allocation (LDA)** are simpler probabilistic models that can be effective in information retrieval for tasks like document ranking and topic modeling.
- These simpler models are often favored in situations where computational resources are constrained, and the complexity of **LLMs** is not required. They can still provide good performance for many IR tasks, especially when properly tuned and integrated into an IR system. **But what is LLM?**



Large Language Model (LLM)

['lärj 'laŋ-gwij 'mä-dᵊl]

A deep learning algorithm that's equipped to summarize, translate, predict, and generate human-sounding text to convey ideas and concepts.

Investopedia

Large Language Models:

- Large Language Models (LLMs) play a pivotal role in Information Retrieval (IR) by harnessing their exceptional language understanding capabilities. These models, like GPT-4.0, are trained on vast internet text, giving them a profound grasp of human language nuances. LLMs serve as smart connectors between user queries and relevant documents.
- In essence, **LLMs are like supercharged search engines.** They excel at comprehending user queries in natural language, going beyond mere keywords. **They consider the meaning and context of words, making search results more precise. Unlike traditional search engines, LLMs understand not just what words mean individually but also how they relate in a sentence or query.**


1 Data Gathering
2 Data Cleaning
3 Data Splitting
4 Model Set-up
LLM Models Training
5 Model Training
6 Model Checking
7 Model Usability
8 Model Enhancement

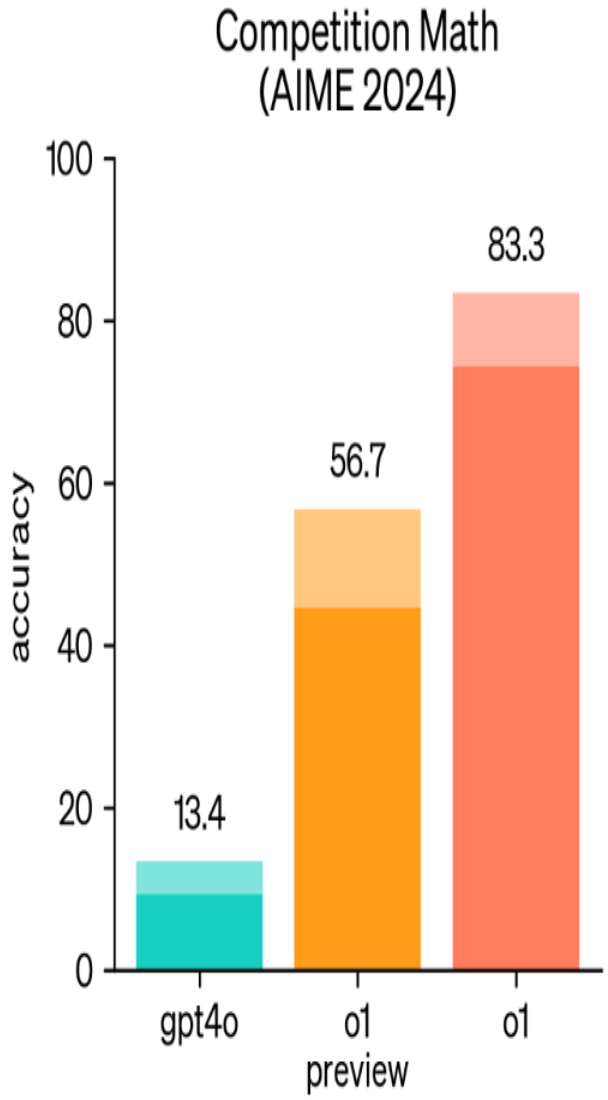
Large Language Models:

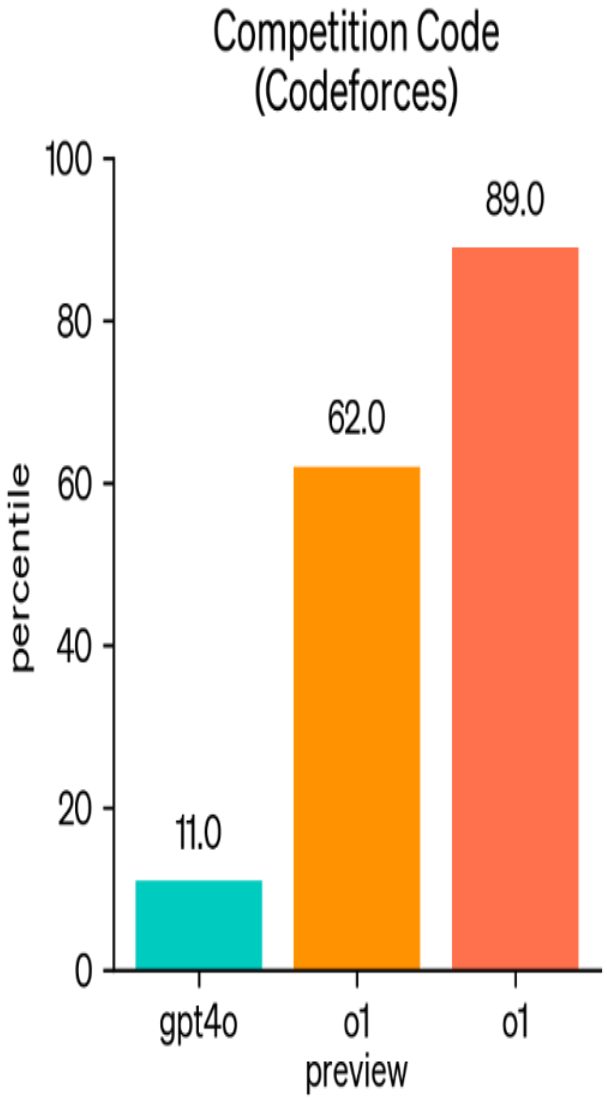
- When you search with LLMs, you're not just matching keywords; you're engaging in contextual search. They analyze the entire query and compare it to their **extensive knowledge base** to find the best-matching documents. This context-based approach greatly enhances the quality of search results.
- Moreover, LLMs don't stop at retrieval; they can generate meaningful answers too. Given a user's question, they can separate through documents, pick out relevant information, and **craft coherent responses.**
- In summary, LLMs in IR are like **super-smart search and answer engines.** They bring context to information retrieval, making it easier for users to find exactly what they need in the vast sea of digital data. This **transformative technology** has revolutionized how we interact with information, making LLMs an integral part of modern IR systems.

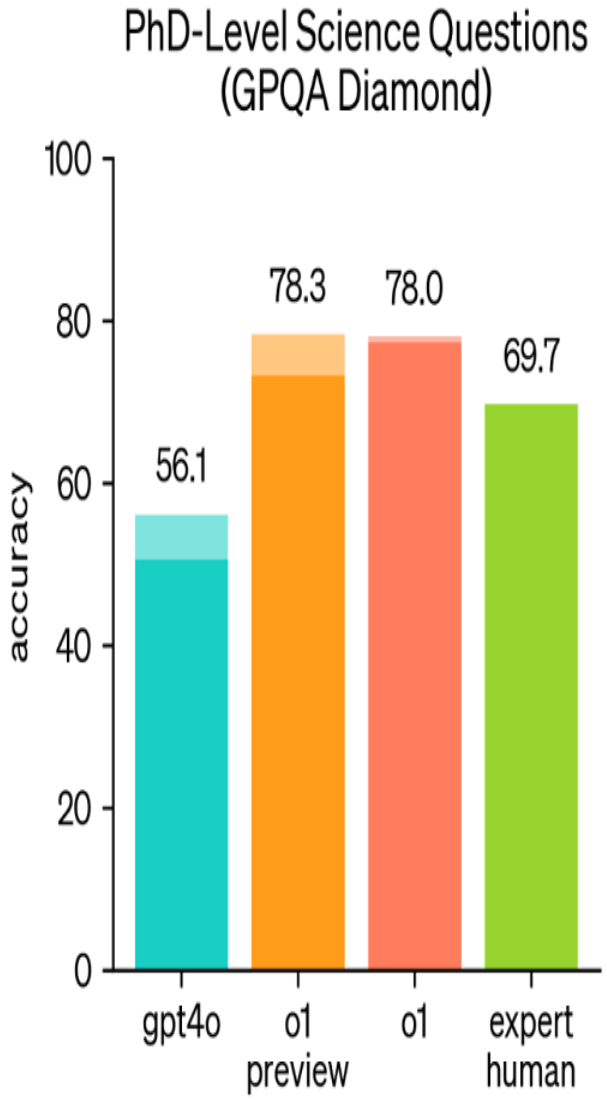
Example:

- Imagine you have a **super-smart robot librarian** in a massive library filled with books. This robot understands exactly what you're looking for when you ask a question. Instead of just matching single words like old libraries, it knows the meaning of your whole sentence.
- So, if you ask, **"What's the best book about space exploration?"** this robot librarian doesn't just look for books with the word "space" or "exploration." It thinks about what you mean and finds the perfect book that talks about space travel and exploration.
- But it doesn't stop there. If you ask, "**Tell me about the first moon landing**," it not only finds a book on the topic but also reads it quickly and gives you a summary, like your own personal book report machine.
- In simple terms, **Large Language Models in Information Retrieval are like super-smart helpers that understand your questions really well and find exactly what you're looking for in a giant library of information**. They make finding stuff and getting answers way easier.

What is next?


Competition Math
(AIME 2024)
accuracy
100
80
60
40
20
0
13.4
56.7
83.3
gpt4o
o1 preview
o1


Competition Code
(Codeforces)
percentile
100
80
60
40
20
0
11.0
62.0
89.0
gpt4o
o1 preview
o1


PhD-Level Science Questions
(GPQA Diamond)
accuracy
100
80
60
40
20
0
56.1
78.3
78.0
69.7
gpt4o
o1 preview
o1
expert human

Are there any LLMs in IR without Machine learning and AI?

No, Large Language Models (LLMs) in Information Retrieval (IR) are inherently based on machine learning and artificial intelligence (AI) techniques. LLMs, such as GPT-4 and its variants, are built using deep learning architectures, specifically **transformer neural networks.** These models are trained on vast amounts of text data from the internet, allowing them to understand and generate human-like text.

- **The fundamental principle behind LLMs is their ability to learn patterns, associations, and context from large datasets, which is a key aspect of machine learning and AI. They use these learned patterns to perform tasks like natural language understanding, text generation, and information retrieval.**
- So, by definition, LLMs in IR are a product of machine learning and AI technologies. They rely on these advanced techniques to provide the sophisticated language understanding and information retrieval capabilities they are known for. **There are no LLMs in IR that operate without machine learning and AI at their core.**

Language models in Information Retrieval

- There are simpler language models used in Information Retrieval. While Large Language Models (LLMs) like GPT-4 are very powerful but complex,. **there are smaller and more specialized language models designed specifically for IR tasks** These simpler models are often used when computational resources are limited or when a less complex approach is sufficient for the task at hand. Here are a couple of examples:

TF-IDF (Term Frequency-Inverse Document Frequency):

- TF-IDF is one of the most basic and widely used models in Information Retrieval. It's not really a language model in the same sense as LLMs but is a technique for measuring the importance of words in documents relative to a collection of documents (corpus). It's simple and effective for tasks like keyword matching and ranking documents based on their relevance to a query.

BM25 (Best Matching 25):

- BM25 is an extension of TF-IDF and is also a simpler model compared to LLMs. It's designed to overcome some of the limitations of TF-IDF and is widely used in search engines for ranking documents based on their relevance to a query.

Structured Text Retrieval Model

- **Structured Text Retrieval Models are designed to retrieve information from structured or semi-structured documents, which contain organized data fields or elements.** Unlike traditional Information Retrieval (IR) models that primarily deal with unstructured text documents, these models focus on documents with well-defined structures. Here are some key features and examples of Structured Text Retrieval Models:

1. **Structured Document Format:** In structured retrieval, documents are typically stored in formats like XML (eXtensible Markup Language), JSON (JavaScript Object Notation), or databases. These formats organize data into fields, tags, or records, making it easier to extract specific information.
2. **Field-Level Retrieval:** Instead of treating the entire document as a unit, structured models allow retrieval at the field level. Users can specify which fields or elements they want to search in and retrieve.
3. **Query Language:** Structured retrieval often involves specialized query languages that allow users to express complex queries involving field-specific conditions and constraints. For example, XQuery for XML documents or SQL for databases.

Structured Text Retrieval Model

4. **Structured Indexing:** To support efficient retrieval, structured models create specialized indexes tailored to the document format. These indexes help locate relevant documents and specific data fields quickly.

Examples:

- **XML Retrieval:** In XML retrieval, users can search for specific elements and attributes within XML documents. **For instance**, finding all books by a particular author from a library database stored in XML format.
- **Database Retrieval:** Structured models are commonly used in database retrieval, where users can search for records that meet specific criteria. **For example**, retrieving all customer orders placed in a particular month from a sales database.

Structured Text Retrieval Model

Applications: Structured Text Retrieval is prevalent in applications involving data retrieval from structured sources like databases, content management systems, scientific datasets, and more. It's often used in enterprise search, e-commerce, scientific research, and data integration tasks.

Structured vs. Unstructured: The main distinction between structured and unstructured retrieval models lies in the document format. Unstructured models deal with plain text documents, while structured models focus on documents with organized structures.

- Overall, Structured Text Retrieval Models cater to scenarios where information is stored in structured or semi-structured formats, enabling users to perform precise and targeted searches within specific data fields or elements.

Structured Text Retrieval Model

Retrieval models which combine information on text content with information on the document structure are called structured text retrieval model.

- **Match point :** refer to the position in the text of a sequence of words which matches the user query.
- **Region :** refer to a contiguous portion of the text.
- **Node :** refer to a structural component of the document such as a chapter, a section, a subsection.

Structured Text Retrieval Models

- Text Retrieval retrieves documents based on **index terms**.
- **Observation:** Documents have implicit structure.
- Regular text retrieval and indexing strategies lose the information available within the structure.
- Text Retrieval desired based on structure. e.g. All documents having “Pakistani Flag” in the caption of a photo.

Models for Structured Text Retrieval

- PAT Expressions
- **Non-Overlapped Lists**
- **Proximal Nodes**
- List of References
- Tree-based
- Query Languages (SFQL,CCL)

Overlapped Lists Model:

- The Overlapped Lists Model is a technique used in Information Retrieval to efficiently find documents that contain multiple specified terms or keywords. Imagine you're looking for documents that mention both "cats" and "dogs." This model helps identify documents where these terms **overlap, meaning they appear together in the same document.**
- **How it Works:**
- Documents are pre-indexed, and for each term (e.g., "cats" and "dogs"), there is a list of documents where that term appears.
- To find documents containing both "cats" and "dogs," the model identifies the lists associated with each term.
- It then looks for documents that appear in both lists, indicating that these documents contain both terms.

Overlapped Lists Model:

Benefits:

- **Efficient:** The model reduces the need to examine all documents. It focuses on documents where the specified terms overlap, saving time and resources.
- **Precision:** It helps find documents that are highly relevant to the query because they contain all the required terms.
- **Example:** Suppose you're searching for articles about "pets" that mention both "cats" and "dogs." The Overlapped Lists Model quickly identifies articles where these terms co-occur, ensuring you find content specifically about pets that include information about both cats and dogs.

Non-Overlapped List Model

- The **Non-Overlapped List Model** is an approach in Information Retrieval used to find documents that contain specific terms or keywords **without requiring those terms to appear together in the same document**. It focuses on locating documents that individually contain each of the specified terms, even if they are in different parts of the document.

Non-Overlapped List Model

How it Works:

1. Documents are pre-indexed, and for each term (e.g., "cats" and "dogs"), there is a list of documents where that term appears.
2. To find documents containing both "cats" and "dogs," the model identifies the lists associated with each term.
3. Instead of looking for documents that appear in both lists (as in the Overlapped Lists Model), this model **retrieves documents from each list separately.**

Non-Overlapped List Model

Benefits:

- **Versatility:** It allows for more flexibility in retrieving documents. They don't need to contain all specified terms together, making it useful for broader queries.
- **Coverage:** The model can find documents that discuss each term individually, even if they are not closely related within the same document.

Example:

- If you're interested in articles about "pets" that may mention "cats" and "dogs" separately, the Non-Overlapped List Model would retrieve articles that discuss each of these pets individually, even if they don't specifically talk about both cats and dogs in the same context.

The Non-Overlapped List Model in simple mathematical notation can be represented as follows:

Let's consider two terms, Term1 and Term2, and the documents associated with each term are represented by lists:

- Documents containing Term1: $D_{\text{Term1}} = \{d_1, d_2, \ldots, d_n\}$
- Documents containing Term2: $D_{\text{Term2}} = \{d_1, d_2, \ldots, d_m\}$

To find documents that contain either Term1 or Term2 (non-overlapping), we can represent this as a union of the two lists:

$$D_{\text{NonOverlap}} = D_{\text{Term1}} \cup D_{\text{Term2}}$$

In this model, we're retrieving documents that are in either list, considering them separately without requiring both terms to appear in the same document.

Simple Example:

- Let's consider a simple example with two terms, Term1 ("cats") and Term2 ("dogs"), and their associated lists of documents:

- Documents containing "cats" (D_{cats}):
  - $D_{\text{cats}} = \{doc1, doc2, doc3\}$
- Documents containing "dogs" (D_{dogs}):
  - $D_{\text{dogs}} = \{doc2, doc4, doc5\}$

Now, to find documents that contain either "cats" or "dogs" (non-overlapping), we take the union of the two lists:

$$D_{\text{NonOverlap}} = D_{\text{cats}} \cup D_{\text{dogs}}$$

$$D_{\text{NonOverlap}} = \{doc1, doc2, doc3, doc4, doc5\}$$

In this example, we're retrieving documents that contain either "cats" or "dogs," without requiring both terms to appear in the same document.

Model based on Non-overlapping lists

- Divide the whole text of each document in non-overlapping text regions which are collected in a list.
- Text regions in the same list have no overlapping, but text regions from distinct lists might overlap.

Drawbacks of the Non-Overlapping Lists Model:

Relevance Challenges: Retrieving documents based on single terms can sometimes bring up irrelevant results and false positives.

Limited Context: Focusing only on individual terms may miss the bigger picture and related information in documents.

Reduced Precision: Trying to capture too much information at once can lead to less accurate results, making it harder to find exactly what you need.

Drawbacks of the Non-Overlapping Lists Model:

Complex Queries: Complicated searches may overwhelm you with too many documents.

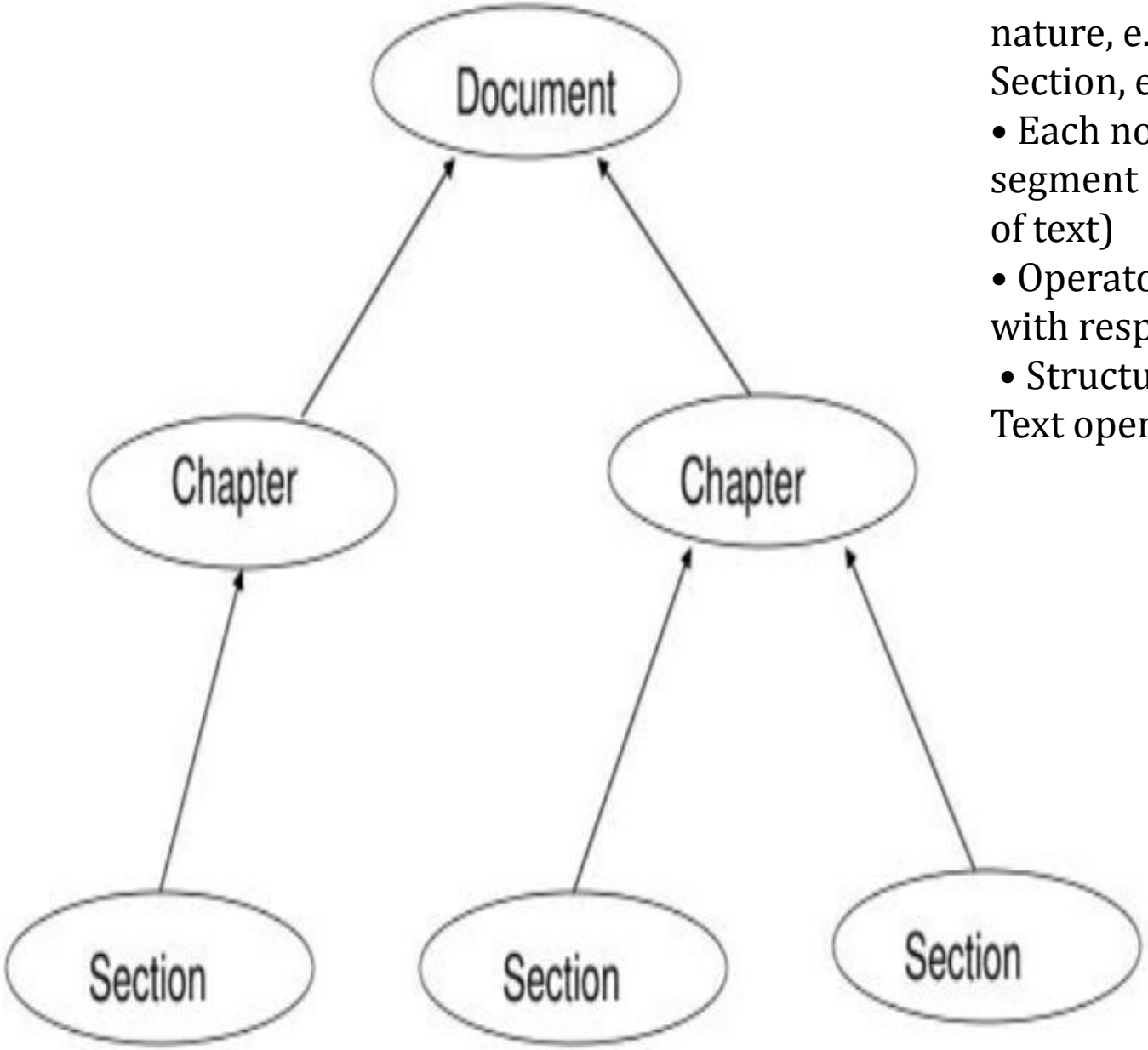
Missed Co-Occurrences: Documents where terms appear together may not show up in results, even if they're relevant.

Resource Intensive: Dealing with a large number of documents individually can be time-consuming and demanding on resources.

Incomplete Information: Looking for connections between terms may not give you the full picture of the topic you're exploring.

Proximal Nodes Model

- **The Proximal Nodes Model is an approach in Information Retrieval used to find relevant documents by considering their relationships within a network or graph structure.** Instead of searching for documents directly, it identifies "**proximal nodes,**" which are intermediary nodes or entities connected to the desired information. This **model helps in exploring and retrieving information within a network context.**



Proximal Nodes

- Nodes are structural in nature, e.g. Chapter, Section, etc.
- Each node has a defined segment (Contiguous part of text)
- Operators are defined with respect to this model.
- Structure operators and Text operators.

Proximal Nodes Model

How it Works:

1. Documents or data points are represented as **nodes in a network or graph**, where connections represent relationships or links between them.
2. **Instead of querying for documents, the user specifies a set of proximal nodes, which are nodes that are likely to be related to the desired information.**
3. **The model explores the network, identifying documents connected to the proximal nodes through relationships.** These connected documents are considered relevant.

The Proximal Nodes Model can be represented in simple mathematical notation as follows:

Let's consider a network or graph of interconnected nodes (representing documents, entities, or data points). Each node is denoted by N_i for i in the range from 1 to n.

To find relevant documents within the network, we specify a set of proximal nodes, denoted as P, which are the intermediary nodes or entities that are likely to be related to the desired information.

The Proximal Nodes Model retrieves documents connected to the proximal nodes through relationships within the network:

$$D_{\text{Proximal}} = \{N_i \mid \text{There exists a relationship between } P \text{ and } N_i\}$$

In this model, D_{Proximal} represents the set of documents that are considered relevant because they are connected to the specified proximal nodes P through network relationships.

Simple Computed Example:

- Let's consider a simple example with a network of interconnected nodes representing articles in an online platform, where nodes are denoted by N_i for simplicity.

Let's say we're interested in articles about "technology" (P), and we identify some proximal nodes related to technology, such as "computer scientists," "tech enthusiasts," and "software developers."

- Proximal nodes (P):
  - $P = \{N_1, N_3, N_5\}$ (where N_1, N_3, and N_5 are nodes related to technology)

Now, using the Proximal Nodes Model, we retrieve documents connected to these proximal nodes:

$$D_{\text{Proximal}} = \{N_i \mid \text{There exists a relationship between } P \text{ and } N_i\}$$

Let's say the documents connected to these proximal nodes are:

- N_1 is connected to articles A, B, C
- N_3 is connected to articles A, D, E
- N_5 is connected to articles B, F, G

$$D_{\text{Proximal}} = \{A, B, C, D, E, F, G\}$$

In this example, we're retrieving articles about "technology" (P) by considering the network relationships with the specified proximal nodes.

Proximal Nodes Model

Benefits:

- **Contextual Retrieval:** Proximal Nodes Model helps in retrieving documents within a broader context of relationships, which can provide more contextually relevant results.

- **Network Exploration:** It facilitates the discovery of information by traversing connections and exploring the relationships between data points.

Proximal Nodes Model

Example:

- Suppose you're interested in scientific research papers related to "**climate change.**" Instead of searching for papers directly, you specify proximal nodes like "**climate scientists**," "**environmental organizations**," and "**climate conferences**." The model then identifies research papers linked to these nodes, providing a broader view of climate change research.
- **The Proximal Nodes Model is valuable for discovering information within interconnected networks, making it useful for applications involving graph databases, social networks, and knowledge graphs**.

Drawbacks Structured text retrieval Models:

- Difficult to specify the structural query
- An advanced user interface is needed

- Structured text retrieval models include no ranking (open research problem!)

Case Study Assignment-III for IR:

Implement to retrieve documents by Probabilistic, Non-Overlapped List and Proximal Nodes Models

REFERENCES:

[1] “INFORMATION RETRIEVAL MODELS,” *Ebrary*, 2013. https://ebrary.net/201793/psychology/retrieval_models (accessed Feb. 22, 2023).

[2] “Ch_2 Information Retrieval Models,” *Rutgers.edu*, 2023. https://aspoerri.comminfo.rutgers.edu/InfoCrystal/Ch_2.html (accessed Feb. 22, 2023).

[3] “Fig. 2. Horizontal taxonomy.,” *ResearchGate*, 2021. https://www.researchgate.net/figure/Horizontal-taxonomy_fig2_47397195 (accessed Feb. 22, 2023).

[4] “4.1. IR MODELS – BASIC CONCEPTS – Wachemo University e-Learning Platform,” *Wachemo-elearning.net*, 2023. https://wachemo-elearning.net/courses/information-storage-and-retirievalitec3081/lessons/chapter-four-ir-model/topic/4-1-ir-models-basic-concepts/ (accessed Feb. 22, 2023).


Any
Questions