PTN GLOBAL

Introduction to LLM and RAG

AI Training Program 2026

About Me

- Graduate from Can Tho University in 2020
- Master in Computer Science, Kyoto Institute of Technology, 2022
- Doctor in Engineering, Kyoto Institute of Technology, 2025

Language Model

- A language model is a machine learning model that aims to **predict** and **generate** plausible language.
  - Example: Autocomplete
- Language models work by estimating the probability of a token or sequence of tokens occurring within a longer sequence of tokens
  - Example: (If token is word-level)

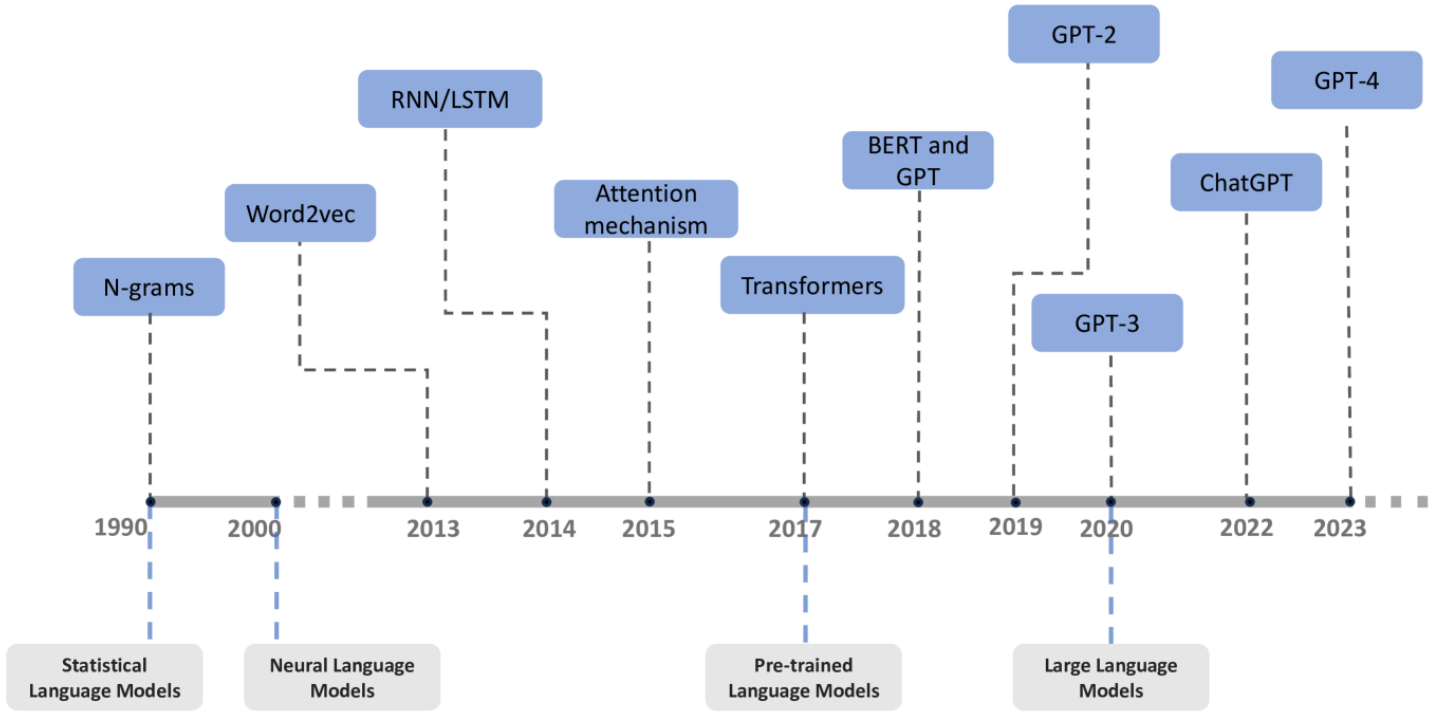
    ```
    When I hear rain on my roof, I _______ in my kitchen.
    ```

    Language models determine probabilities:

    ```
    cook soup 9.4%
    warm up a kettle 5.2%
    cower 3.6%
    nap 2.5%
    relax 2.2%
    ...
    ```
- Estimating the probability of what comes next in a sequence: generating text, translating languages, and answering questions, etc.

Large Language Model (LLM)

- Modeling human language at scale is a highly complex and resource-intensive endeavor.
- The size and capability of language models has exploded.



BERT (110M parameters)
PaLM 2 (up to 340B parameters)

https://arxiv.org/abs/2402.06853

Tokenization

A cute teddy bear is reading.

arbitrary	A	cute	teddy bear			is	reading	.	
word	A	cute	teddy	bear		is	reading	.	
character	A	c	u	.....	d	i	n	g	.
sub-word	A	cute	ted	##dy	bear	is	read	##ing	.

Tokenization Summary

PTN GLOBAL

Method	Pros	Con
Word-level	• Simple • Interpretable	• Risk of OOV • Does not leverage knowledge of root
Sub-word level	• Leverages common prefixed and suffixes • Learned from the data	• Risk of OOV, though less than word-level
Character-level	• Small chance of OOV • Robust to casing and misspelling	• Computation is slower • Embeddings not interpretable

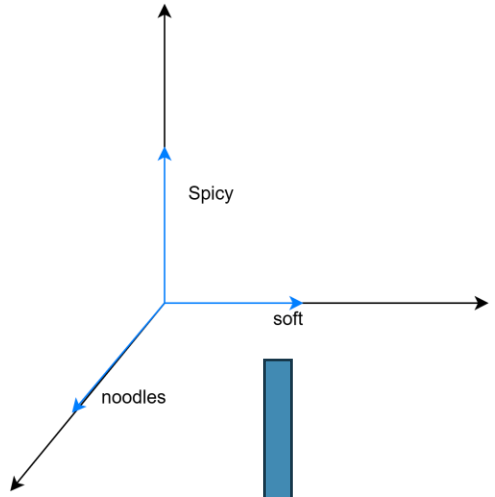
Token representation

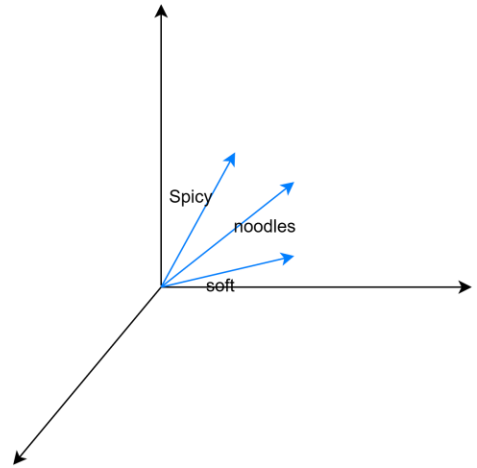
- Naïve (one-hot) encoding
- How can we compare between the 2 tokens?

Hello, I'm Ironman. I have Friday AI

One Hot Encoding

AI	1	0	0	0	0	0	0
Ironman	0	1	0	0	0	0	0
Friday	0	0	1	0	0	0	0
have	0	0	0	1	0	0	0
Hello	0	0	0	0	1	0	0
I	0	0	0	0	0	1	0
I'm	0	0	0	0	0	0	1





Word2vec

- Neural network with a proxy task over billions words worth of text
- Learns an embedding layer
- Proxy tasks:
  - CBOW (continuous bag of words)

    ...**A cute** teddy bear **is reading**...

  - Skip-gram

    ...A cute **teddy bear** is reading...

Word2vec

PTN GLOBAL

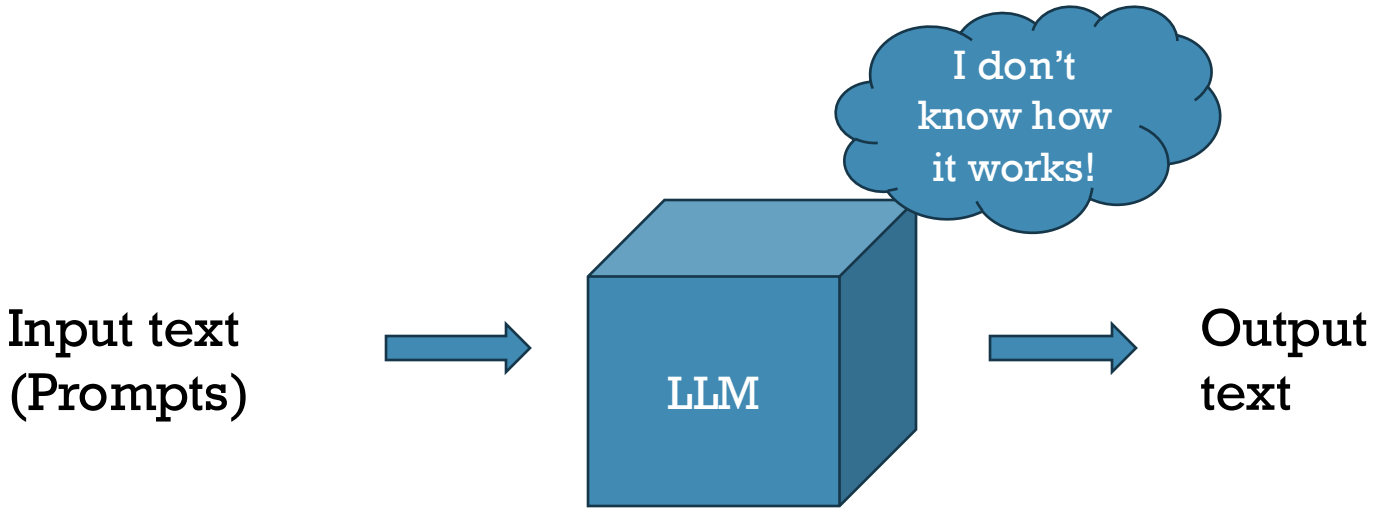
- Neural network with a proxy task over billions words worth of text
- Learns an embedding layer
- Proxy tasks:
  - CBOW (continuous bag of words)

    ...**A cute** teddy bear **is reading**...

  - Skip-gram

    ...A cute **teddy bear** is reading...

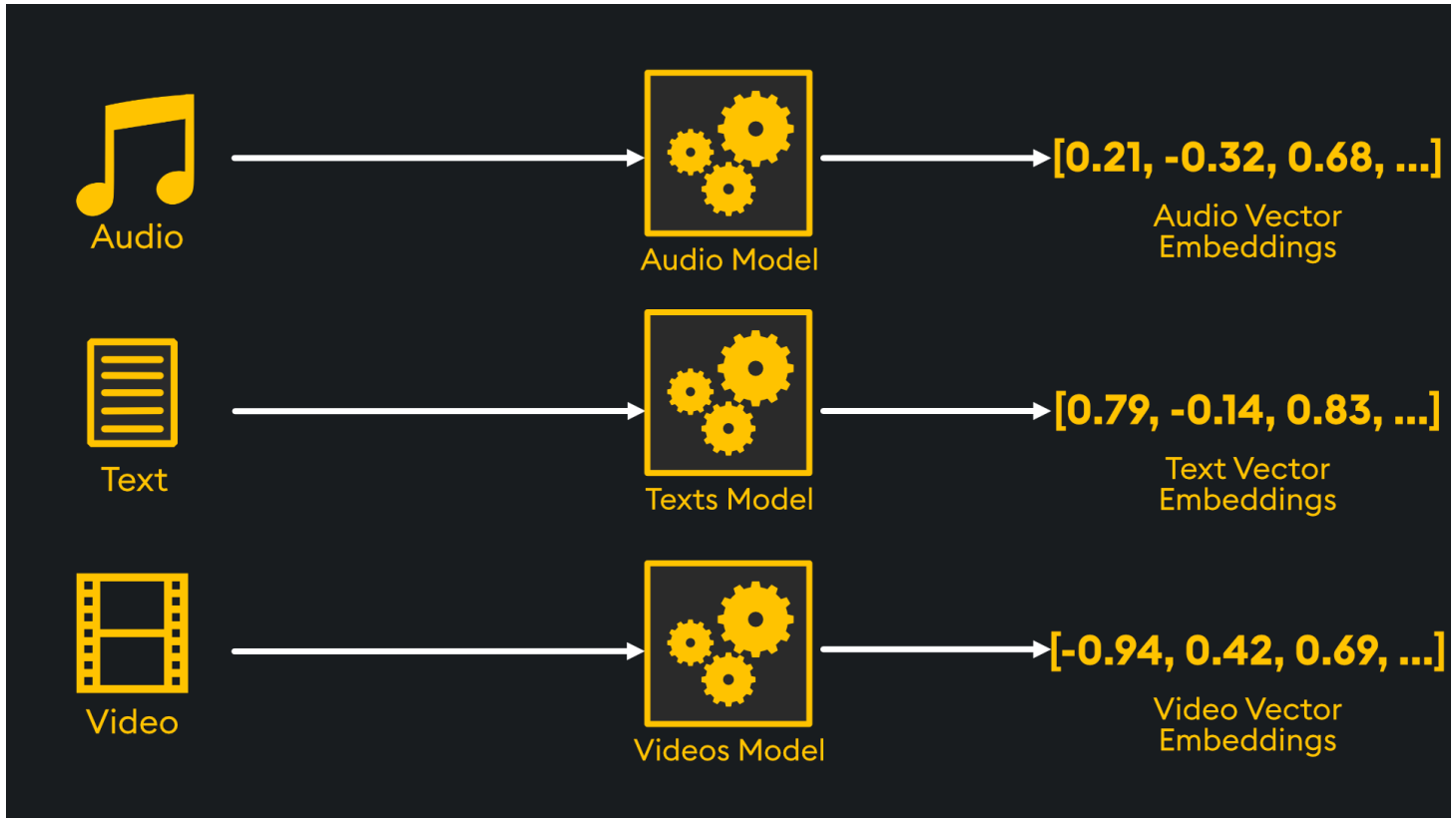
LLM in this lecture



Vector Search and RAG

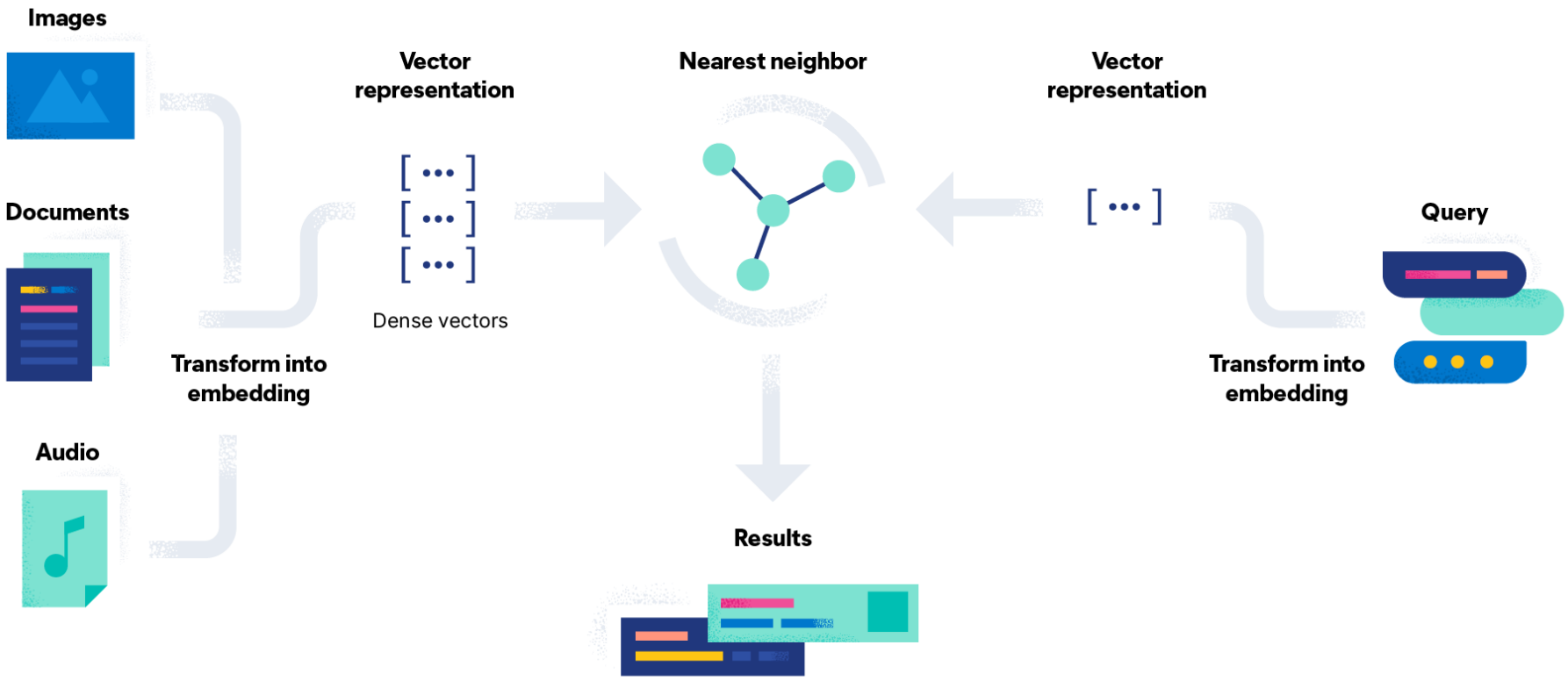
What is Vector Embedding?

- A way to describe data
- Turing data into vector
- We can use math to understand data

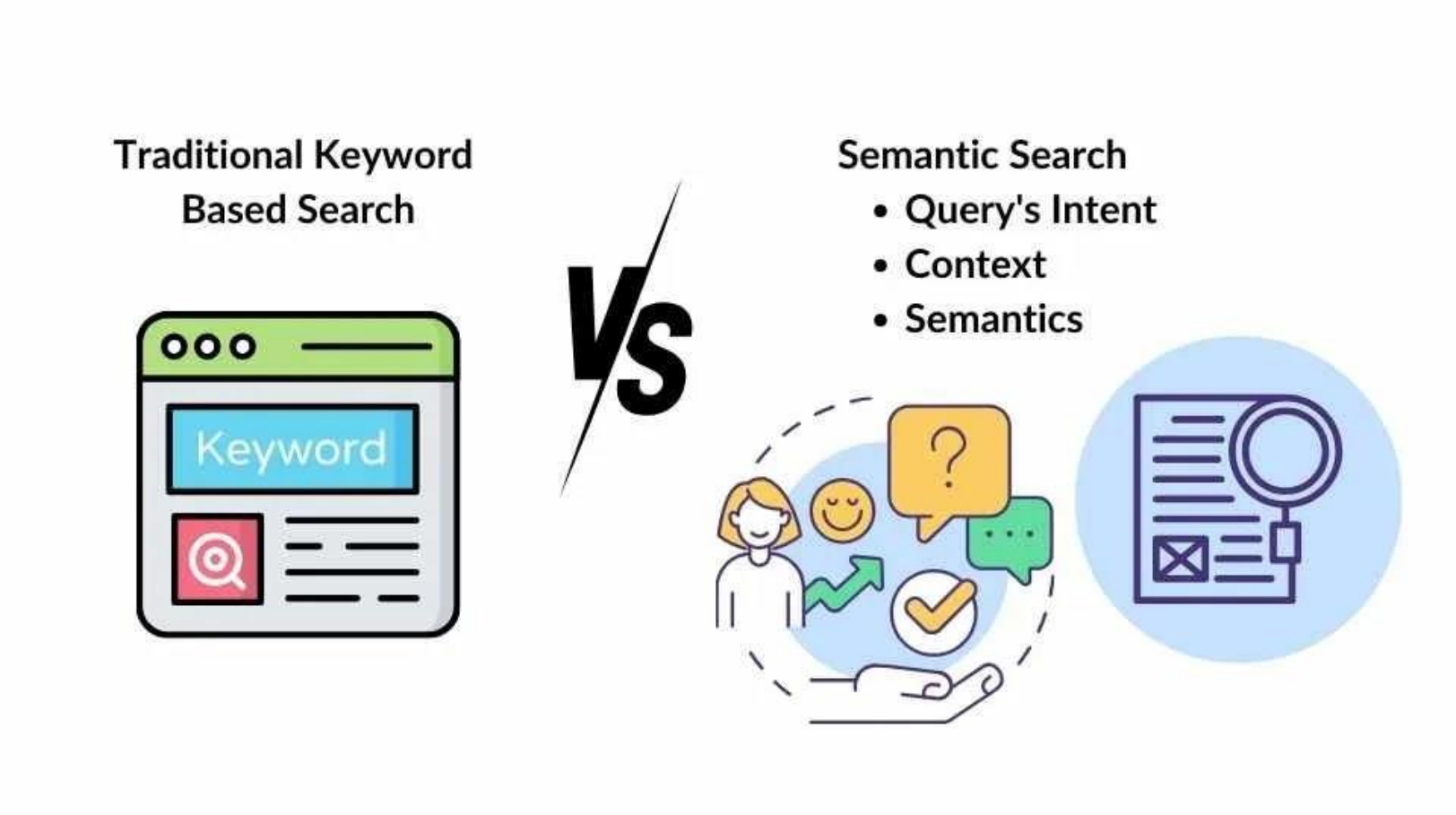


Vector Search

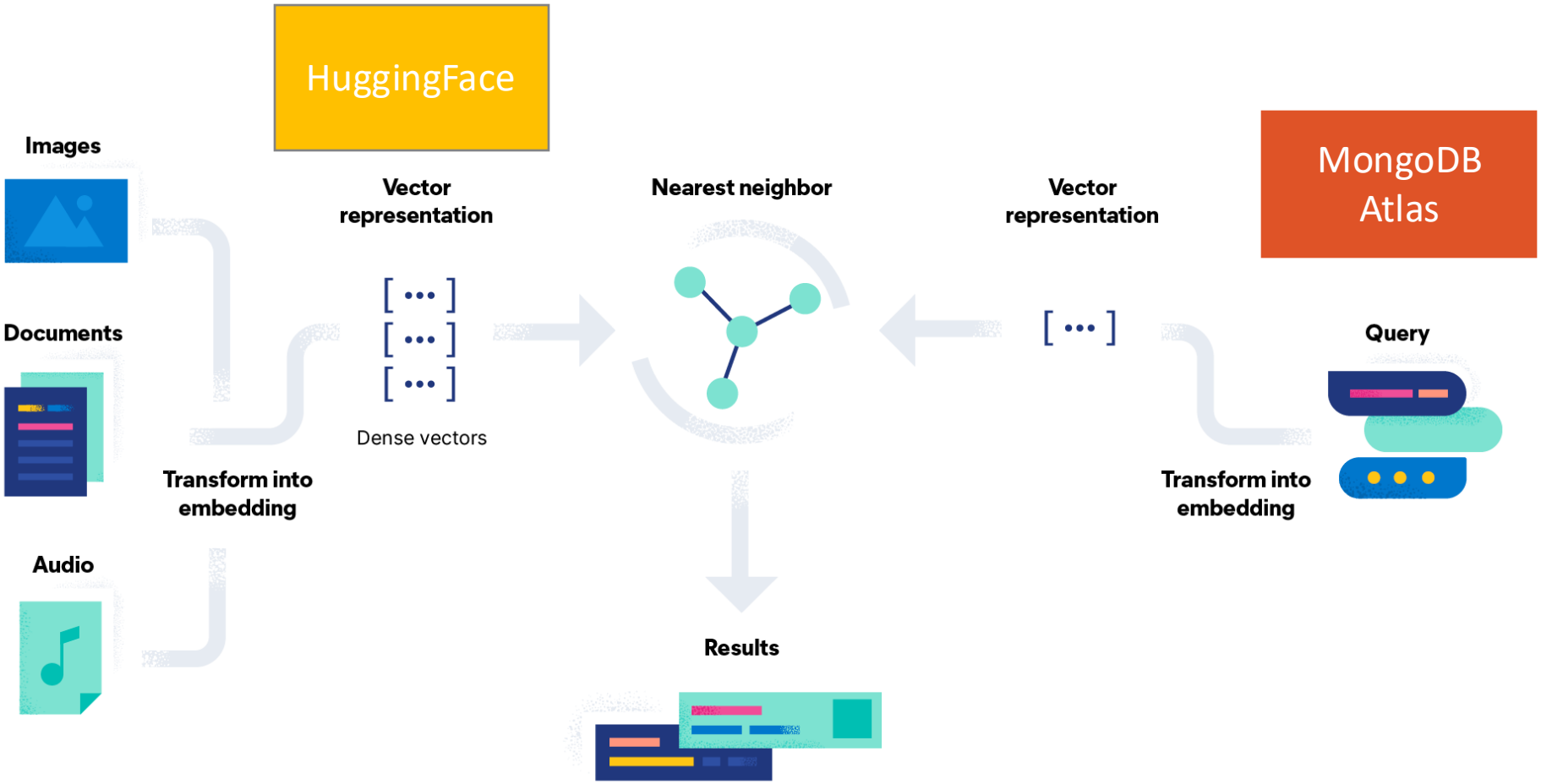
- A way to implement semantic search
- Retrieve information represented under vector
- Understanding the meaning or context of the query



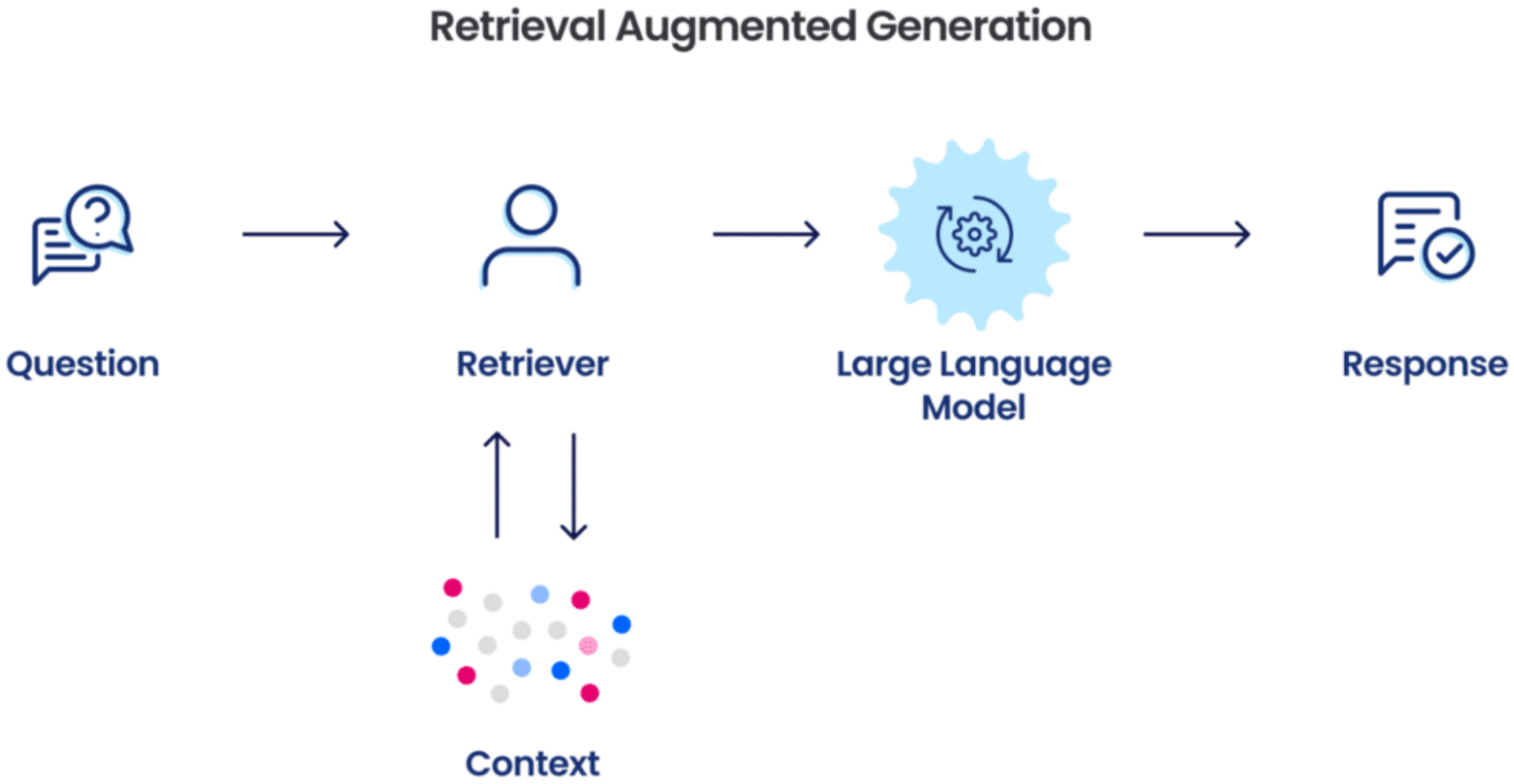
Query vs Semantic Search

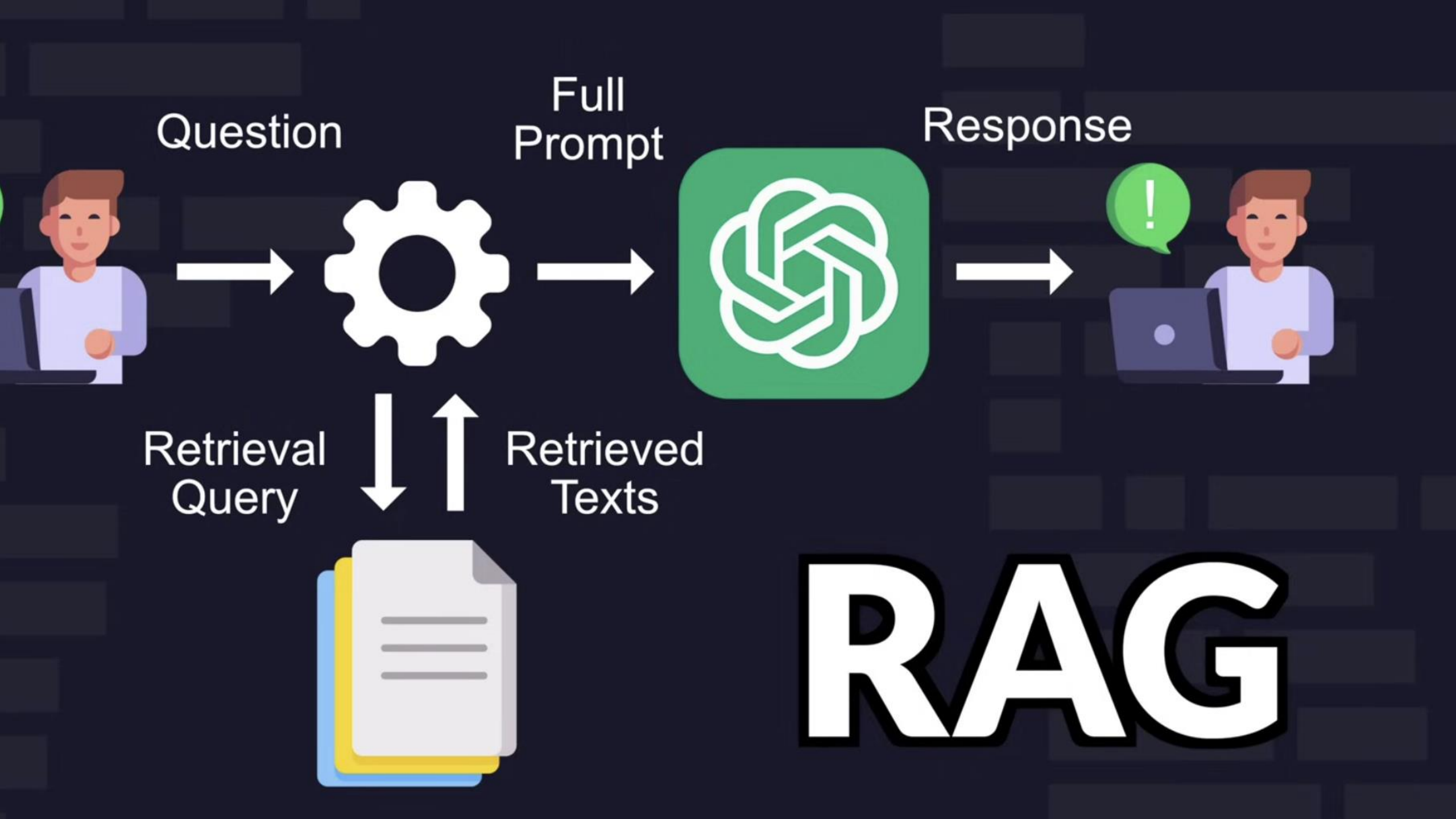


Solution for Movies semantic search



RAG Model

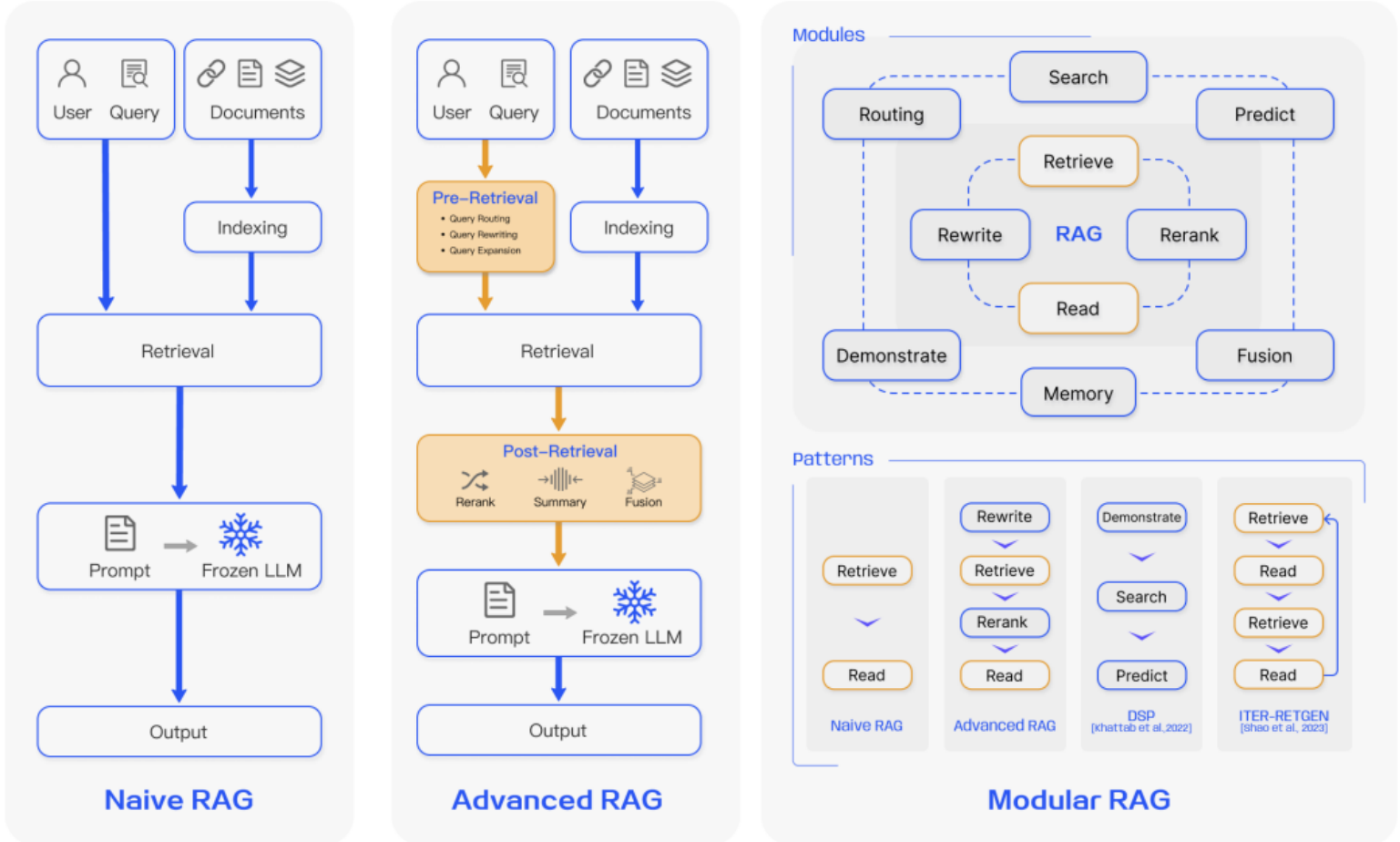

Retrieval Augmented Generation
Question
Retriever
Large Language Model
Response
Context


Question
Full Prompt
Response
Retrieval Query
Retrieved Texts
RAG

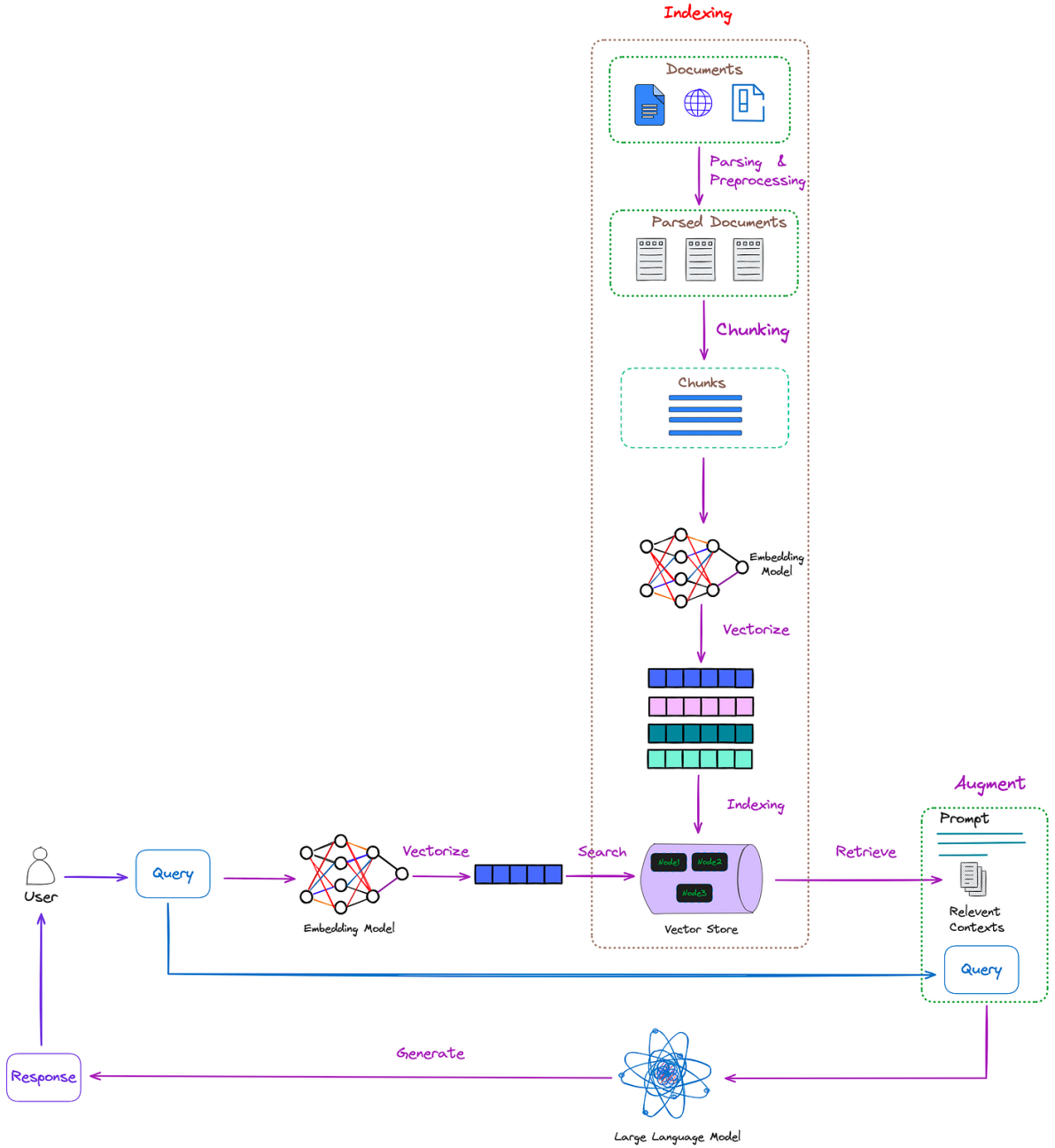
PTN GLOBAL

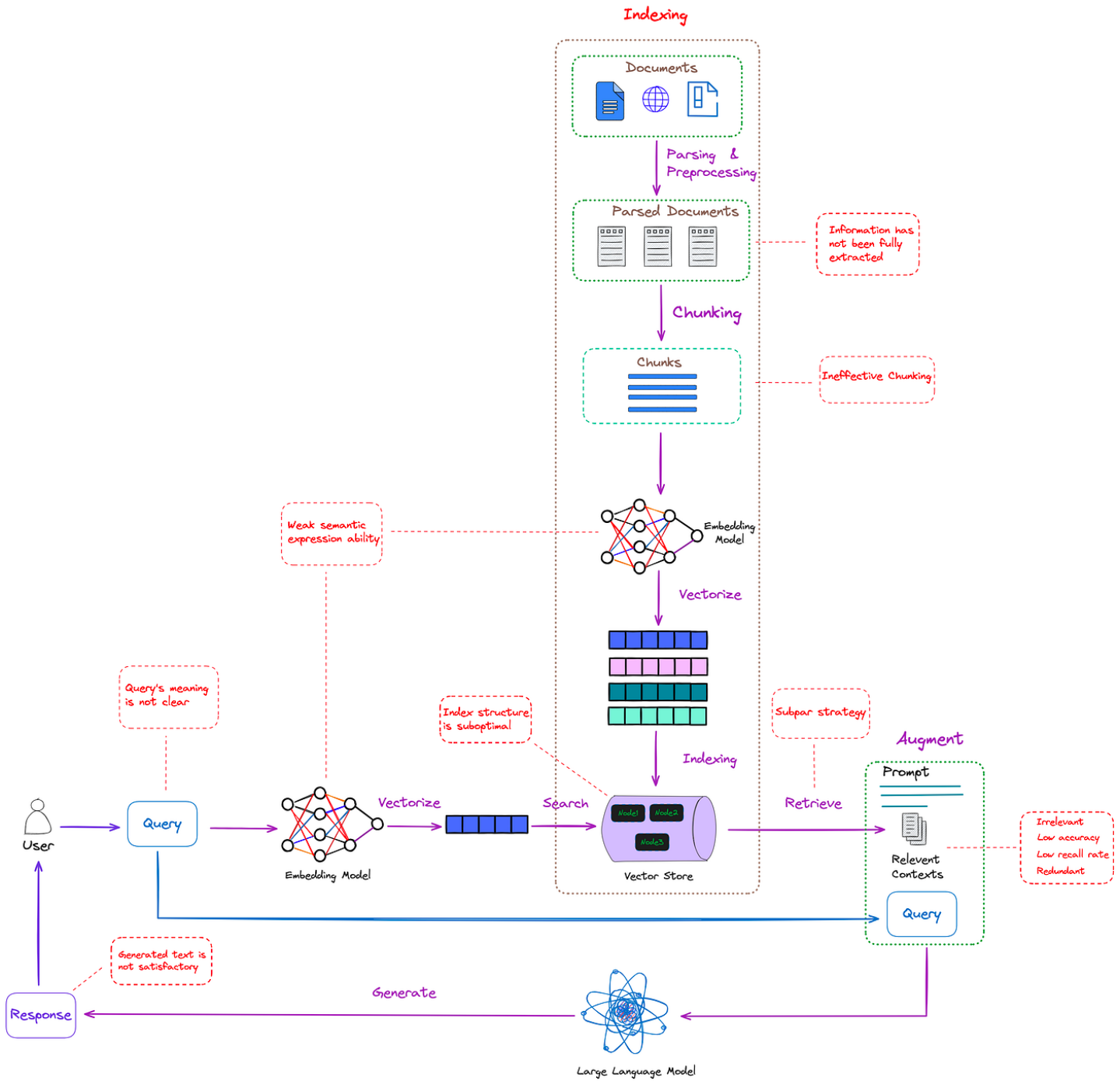
Problems of Naive RAG

AI Team Technical Sharing


User
Query
Documents
Indexing
Retrieval
Prompt
Frozen LLM
Output
Naive RAG
Pre-Retrieval
Query Routing
Query Rewriting
Query Expansion
Post-Retrieval
Rerank
Summary
Fusion
Advanced RAG
Modules
Search
Routing
Predict
Retrieve
Rewrite
RAG
Rerank
Read
Demonstrate
Fusion
Memory
Patterns
Naive RAG
Advanced RAG
DSP
[Khattab et al.,2022]
ITER-RETGEN
[Shao et al., 2023]
Modular RAG

Naive RAG


Indexing
Documents
Parsing & Preprocessing
Parsed Documents
Chunking
Chunks
Embedding Model
Vectorize
Indexing
Augment
Prompt
Relevent Contexts
Query
User
Query
Vectorize
Embedding Model
Search
Node1
Node2
Node3
Vector Store
Retrieve
Response
Generate
Large Language Model


Indexing
Documents
Parsing & Preprocessing
Parsed Documents
Information has not been fully extracted
Chunking
Chunks
Ineffective Chunking
Weak semantic expression ability
Embedding Model
Vectorize
Query's meaning is not clear
Index structure is suboptimal
Subpar strategy
Augment
Indexing
Prompt
User
Query
Vectorize
Search
Retrieve
Node1
Node2
Node3
Embedding Model
Vector Store
Relevent Contexts
Irrelevant
Low accuracy
Low recall rate
Redundant
Query
Generated text is not satisfactory
Generate
Response
Large Language Model

Problem 01: Indexing

- Information extraction is incomplete, as it does not effectively process useful information in images and tables within unstructured files such as PDF.
- The chunking process uses a “one-size-fits-all” strategy instead of selecting optimal strategies based on the characteristics of different file types. This has led to each chunk containing incomplete semantic information. Furthermore, it fails to consider important details, such as existing headings in the text.
- The indexing structure is not sufficiently optimized, leading to inefficient retrieval functionality.
- The embedding model’s semantic representation capability is weak.

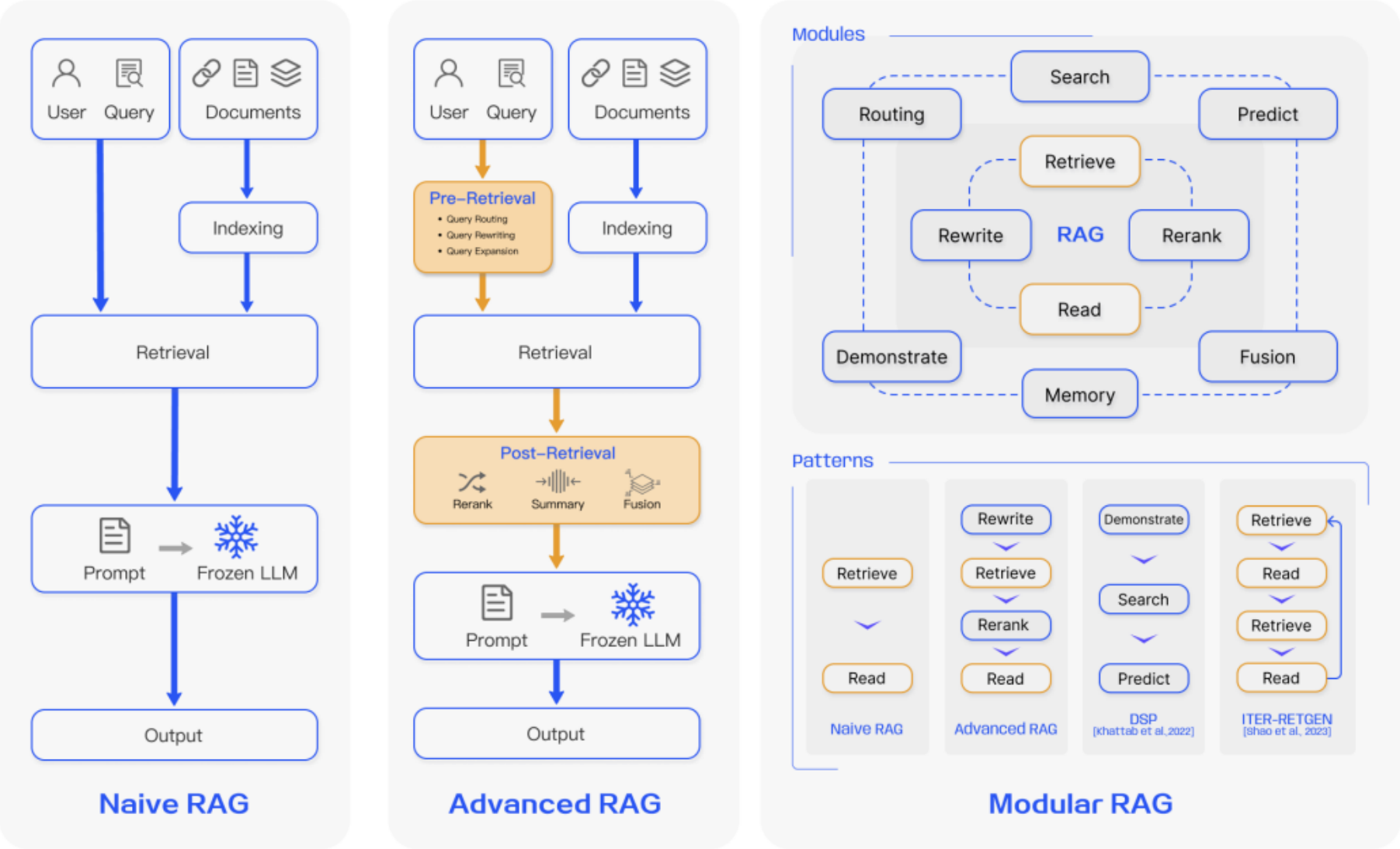
Problem 02: Retrieval

- The relevance of the recalled contexts is inadequate and the accuracy is low.
- The low recall rate prevents the retrieval of all relevant passages, thereby hindering the ability of LLMs to generate comprehensive answers.
- The query may be inaccurate or the semantic representation capability of the embedding model may be weak, resulting in the inability to retrieve valuable information.
- The retrieval algorithm is limited because it does not incorporate different types of retrieval methods or algorithms, such as combining keyword, semantic, and vector retrieval.
- Information redundancy occurs when multiple retrieved contexts contain similar information, leading to repetitive content in the generated answers.

Problem 03: Generation

- Effectively integrating the retrieved context with the current generation task may not be possible, resulting in inconsistent outputs.
- Over-reliance on the enhanced information during the generation process carries a high risk. This can lead to outputs that simply repeat the retrieved content without providing valuable information.
- The LLM may generate incorrect, irrelevant, harmful, or biased responses.

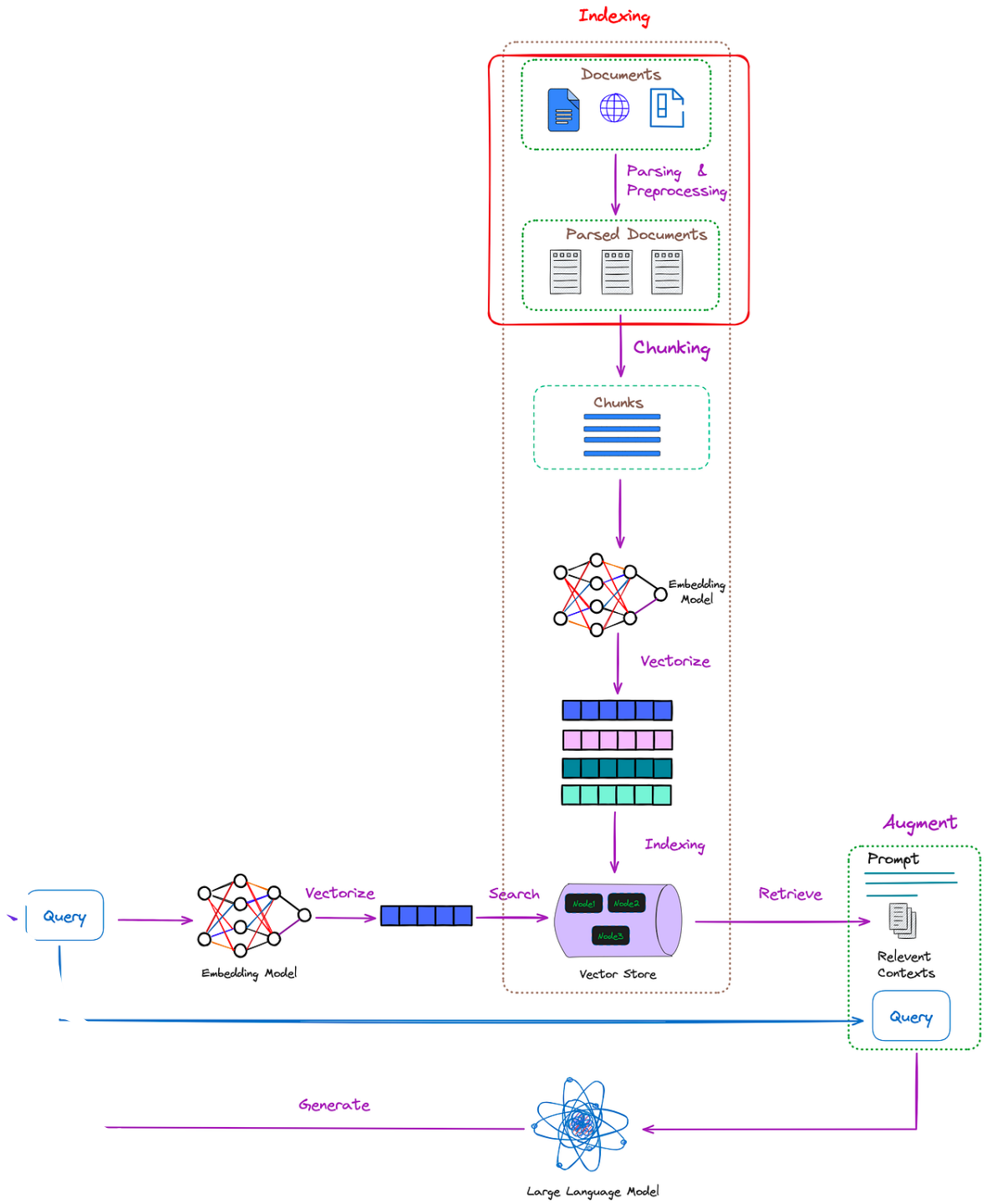
Different type of RAG



PTN GLOBAL

Advanced RAG 02: Unveiling PDF Parsing

Parsing in Naive RAG



The Challenges of Parsing PDF

- Instead of being a data format, it is more accurate to describe PDF as a collection of printing instructions.
- The challenge in parsing PDF documents lies *in accurately extracting the layout of the entire page and translating the content, including tables, titles, paragraphs, and images, into a textual representation of the document.*

Semi-structured document: html

```
<!DOCTYPE html>
<html>
<body>
<p> paragraph </p>
<table border="1">
    <tr>
        <td>Row 1, cell 1</td>
        <td>Row 1, cell 2</td>
    </tr>
</table>
</body>
</html>
```

Unstructured document: PDF

```
4 0 obj
<< >>
stream
1. 0. 0. 1. 50. 300. cm
BT
 /F0 26. Tf
 (Hello, World!) Tj
ET
endstream
endobj
```

How to parse PDF documents

In general, there are three approaches to parsing PDFs:

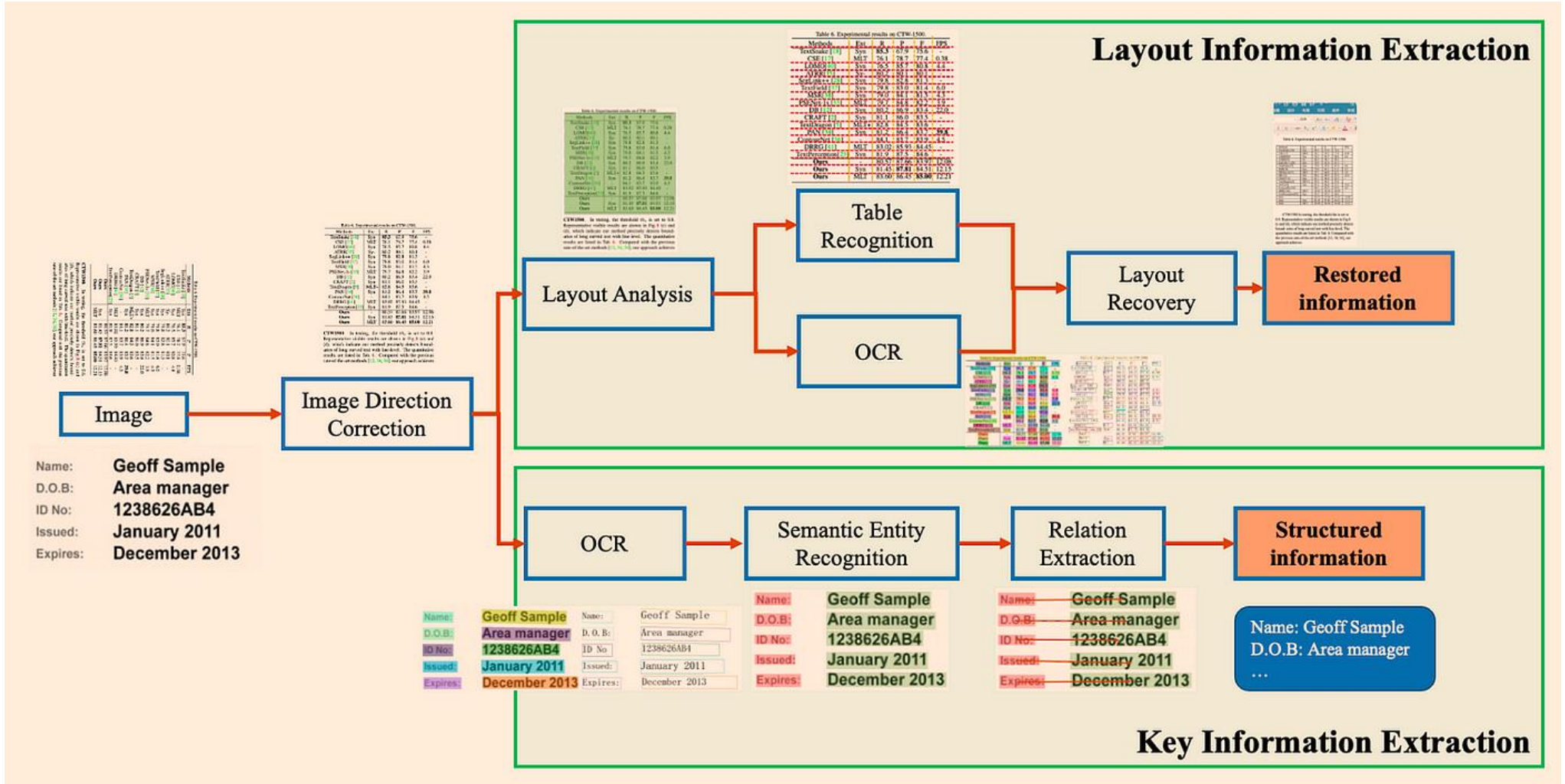
- Rule-based approach: not very generalizable as there are numerous types and layouts of PDFs, making it impossible to cover them all with predefined rules.
- Approach based on deep learning models: such as the popular solution that combines object detection and OCR models.
- Pasing complex structures or extracting key information in PDFs based on multimodal large models.

Rule-based approach

- https://github.com/py-pdf/pypdf - It is a standard method in LangChain and LlamaIndex for parsing PDF files.
- Treating each line of the document as a sequence separated by newline characters “\n”, which prevents accurate identification of paragraphs or tables => **Major limitations**

Methods based on deep learning models

- **Advantages**: its ability to accurately identify the layout of the entire document, including tables and paragraphs
- **Limitations**: The object detection and OCR stages can be time-consuming and requiring computing capacity
- **Suggestions:**
  - Unstructured: It has been integrated into langchain. The table recognition effect of the `hi_res strategy` with `infer_table_structure=True` is good
  - Layout-parser: recognize complex structured PDFs, the largest model for higher accuracy, may be slightly slower.
  - PP-StructureV2: Various model combinations are used for document analysis, with performance above average.

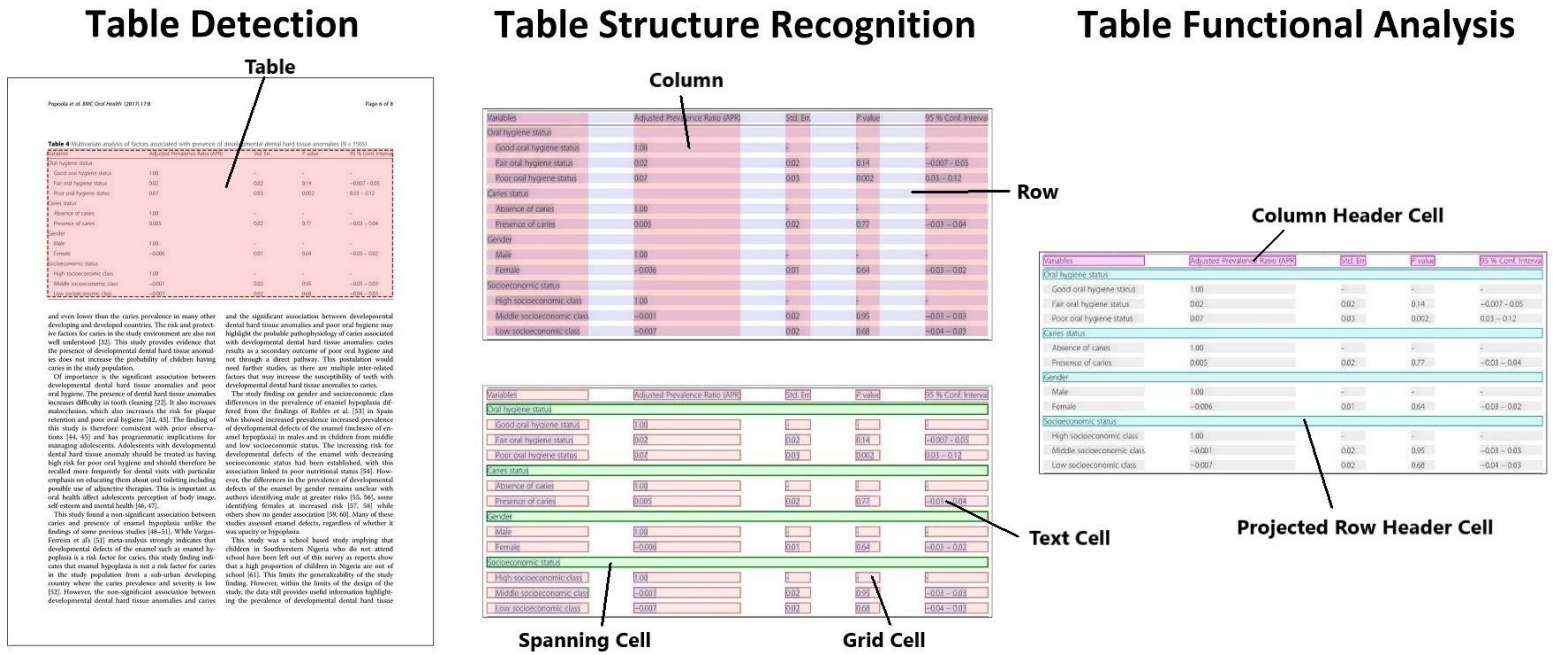

Layout Information Extraction
Image
Image Direction Correction
Layout Analysis
Table Recognition
OCR
Layout Recovery
Restored information
Name: Geoff Sample
D.O.B: Area manager
ID No: 1238626AB4
Issued: January 2011
Expires: December 2013
OCR
Semantic Entity Recognition
Relation Extraction
Structured information
Name: Geoff Sample
D.O.B: Area manager
…
Key Information Extraction

Pasing complex structures in PDFs based on multimodal large model

- Retrieving relevant images (PDF pages) and sending them to GPT4-V to respond to queries.
- Regarding every PDF page as an image, let GPT4-V do the image reasoning for each page. Build Text Vector Store index for the image reasonings. Query the answer against the Image Reasoning Vector Store.
- Using Table Transformer to crop the table information from the retrieved images and then sending these cropped images to GPT4-V for query responses.
- Applying OCR on cropped table images and send the data to GPT4/ GPT-3.5 to answer the query.

Table Transformer (TATR)

- A deep learning model based on object detection for extracting tables from PDFs and images.
- TATR is an object detection model that recognizes tables from image input. The inference code built on TATR needs text extraction (from OCR or directly from PDF) as a separate input in order to include text in its HTML or CSV output.



Challenge: Parse PDFs using the open-source unstructured framework, addressing two key challenges.

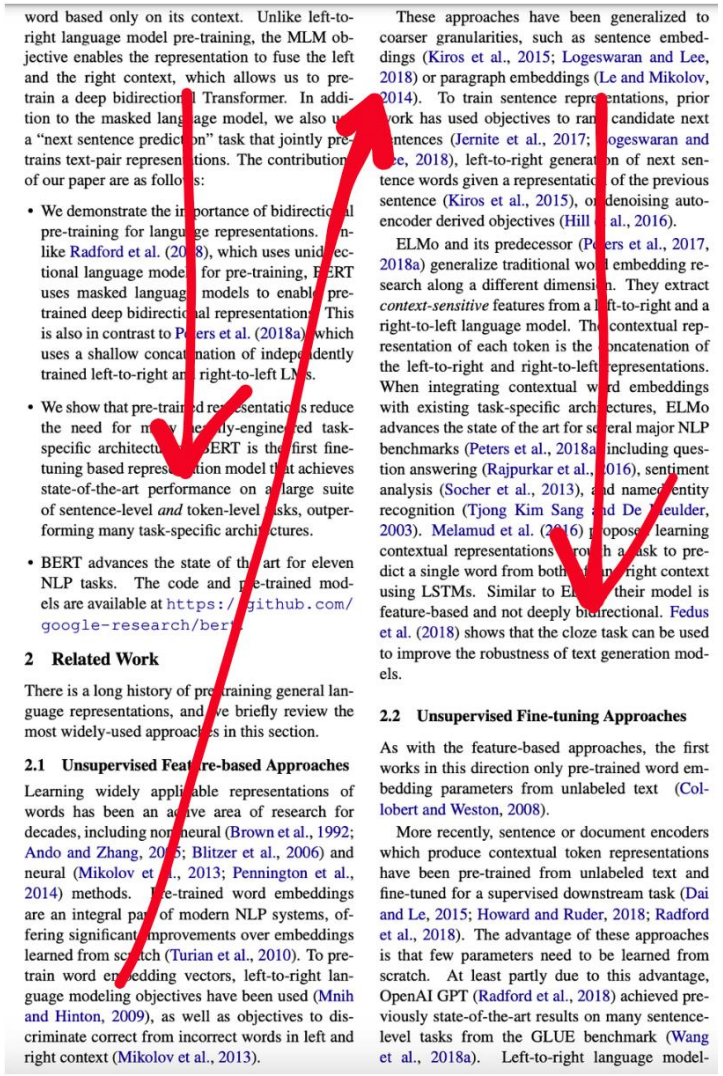
Challenge 1: How to extract data from tables and images

```
from unstructured.partition.pdf import partition_pdf

filename = "/Users/Florian/Downloads/Attention_Is_All_You_Need.pdf"

# infer_table_structure=True automatically selects hi_res strategy
elements = partition_pdf(filename=filename, infer_table_structure=True)
tables = [el for el in elements if el.category == "Table"]

print(tables[0].text)
print('-----------------------------------------------')
print(tables[0].metadata.text_as_html)
```

The flow of partition_pdf

- https://miro.medium.com/v2/resize:fit:1400/format:webp/1*HGr4lT9Sg5d3P63TPu_Acw.png

Challenge 2: How to rearrange the detected blocks? Especially for double-column PDFs.



word based only on its context. Unlike left-to-right language model pre-training, the MLM objective enables the representation to fuse the left and the right context, which allows us to pre-train a deep bidirectional Transformer. In addition to the masked language model, we also use a "next sentence prediction" task that jointly pre-trains text-pair representations. The contributions of our paper are as follows:

- We demonstrate the importance of bidirectional pre-training for language representations. Unlike Radford et al. (2018), which uses unidirectional language models for pre-training, BERT uses masked language models to enable pre-trained deep bidirectional representations. This is also in contrast to Peters et al. (2018a), which uses a shallow concatenation of independently trained left-to-right and right-to-left LMs.
- We show that pre-trained representations reduce the need for many heavily-engineered task-specific architectures. BERT is the first fine-tuning based representation model that achieves state-of-the-art performance on a large suite of sentence-level *and* token-level tasks, outperforming many task-specific architectures.
- BERT advances the state of the art for eleven NLP tasks. The code and pre-trained models are available at https://github.com/google-research/bert.

2 Related Work

There is a long history of pre-training general language representations, and we briefly review the most widely-used approaches in this section.

2.1 Unsupervised Feature-based Approaches

Learning widely applicable representations of words has been an active area of research for decades, including non-neural (Brown et al., 1992; Ando and Zhang, 2005; Blitzer et al., 2006) and neural (Mikolov et al., 2013; Pennington et al., 2014) methods. Pre-trained word embeddings are an integral part of modern NLP systems, offering significant improvements over embeddings learned from scratch (Turian et al., 2010). To pre-train word embedding vectors, left-to-right language modeling objectives have been used (Mnih and Hinton, 2009), as well as objectives to discriminate correct from incorrect words in left and right context (Mikolov et al., 2013).

These approaches have been generalized to coarser granularities, such as sentence embeddings (Kiros et al., 2015; Logeswaran and Lee, 2018) or paragraph embeddings (Le and Mikolov, 2014). To train sentence representations, prior work has used objectives to rank candidate next sentences (Jernite et al., 2017; Logeswaran and Lee, 2018), left-to-right generation of next sentence words given a representation of the previous sentence (Kiros et al., 2015), or denoising auto-encoder derived objectives (Hill et al., 2016).

ELMo and its predecessor (Peters et al., 2017, 2018a) generalize traditional word embedding research along a different dimension. They extract *context-sensitive* features from a left-to-right and a right-to-left language model. The contextual representation of each token is the concatenation of the left-to-right and right-to-left representations. When integrating contextual word embeddings with existing task-specific architectures, ELMo advances the state of the art for several major NLP benchmarks (Peters et al., 2018a) including question answering (Rajpurkar et al., 2016), sentiment analysis (Socher et al., 2013), and named entity recognition (Tjong Kim Sang and De Meulder, 2003). Melamud et al. (2016) proposed learning contextual representations through a task to predict a single word from both left and right context using LSTMs. Similar to ELMo, their model is feature-based and not deeply bidirectional. Fedus et al. (2018) shows that the cloze task can be used to improve the robustness of text generation models.

2.2 Unsupervised Fine-tuning Approaches

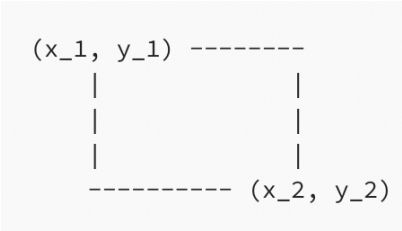
As with the feature-based approaches, the first works in this direction only pre-trained word embedding parameters from unlabeled text (Collobert and Weston, 2008).

More recently, sentence or document encoders which produce contextual token representations have been pre-trained from unlabeled text and fine-tuned for a supervised downstream task (Dai and Le, 2015; Howard and Ruder, 2018; Radford et al., 2018). The advantage of these approaches is that few parameters need to be learned from scratch. At least partly due to this advantage, OpenAI GPT (Radford et al., 2018) achieved previously state-of-the-art results on many sentence-level tasks from the GLUE benchmark (Wang et al., 2018a). Left-to-right language model-

Challenge 2: How to rearrange the detected blocks? Especially for double-column PDFs

- Output in format:

- [LayoutElement(bbox=Rectangle(x1=851.1539916992188, y1=181.15073777777613, x2=1467.844970703125, y2=587.8204599999975), text='These approaches have been generalized to coarser granularities, such as sentence embed- dings (Kiros et al., 2015; Logeswaran and Lee, 2018) or paragraph embeddings (Le and Mikolov, 2014). To train sentence representations, prior work has used objectives to rank candidate next sentences (Jernite et al., 2017; Logeswaran and Lee, 2018), left-to-right generation of next sen- tence words given a representation of the previous sentence (Kiros et al., 2015), or denoising auto- encoder derived objectives (Hill et al., 2016). ', source=<Source.YOLOX: 'yolox'>, type='Text', prob=0.9519357085227966, image_path=None, parent=None),……]
- where (x1, y1) is the coordinate of the top-left vertex, and (x2, y2) is the coordinate of the bottom-right vertex



- Sorting: layout.sort(key=lambda z: (z.bbox.x1, z.bbox.y1, z.bbox.x2, z.bbox.y2))

Reading

- https://arxiv.org/pdf/2312.10997.pdf
- Evaluate RAG

THANK YOU!

For more information – Contact

Nguyen Tran

Mobile: +842274245
Email: NGUYEN.TRAN@PTNGLOBALCORP.COM
MS Teams: