School of CIT –
Social Computing
Research Group

Technical
University
of Munich

Advanced Natural Language Processing
CIT4230002

Prof. Dr. Georg Groh
Jakob Sturm, M.Sc.
Alexander Fichtl, M.Sc.

Lecture 11
Knowledge Enhancement

Outline

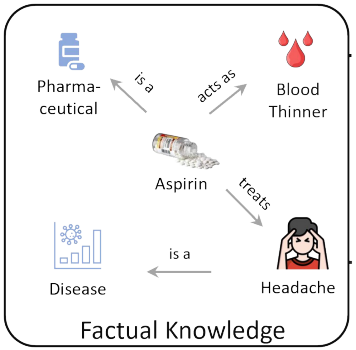
- Overview of Knowledge Enhancement
- Retrieval Augmented Generation (RAG)
  - RAG-phases: IR and GEN
  - Evaluation
  - Painpoints, Advantages, Disadvantages
  - LLM usage
- Adapter-based Knowledge Enhancement
  - Adapters
  - Applications
- Summary

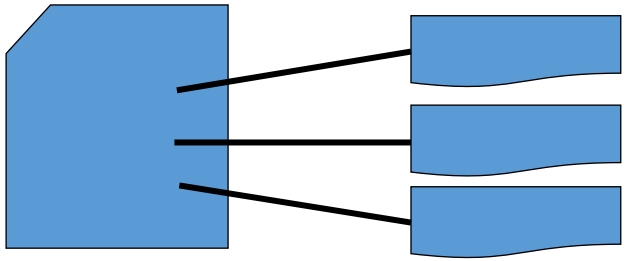
Outline

- **Overview of Knowledge Enhancement**
- Retrieval Augmented Generation (RAG)
  - RAG-phases: IR and GEN
  - Evaluation
  - Painpoints, Advantages, Disadvantages
  - LLM usage
- Adapter-based Knowledge Enhancement
  - Adapters
  - Applications
- Summary

Overview of Knowledge Enhancement

- Probability based working principle of LLMs and static nature of the knowledge encoded in PLMs makes hallucinations unavoidable
- A recent emerging trend in NLP studies has been to explicitly incorporate knowledge into PLMs
- -> integrating dynamic, external sources of knowledge into LLMs, for example, from knowledge graphs (KGs) or external documents





[1]

Overview of Knowledge Enhancement

Domain Adaptation?

Knowledge Enhancement?

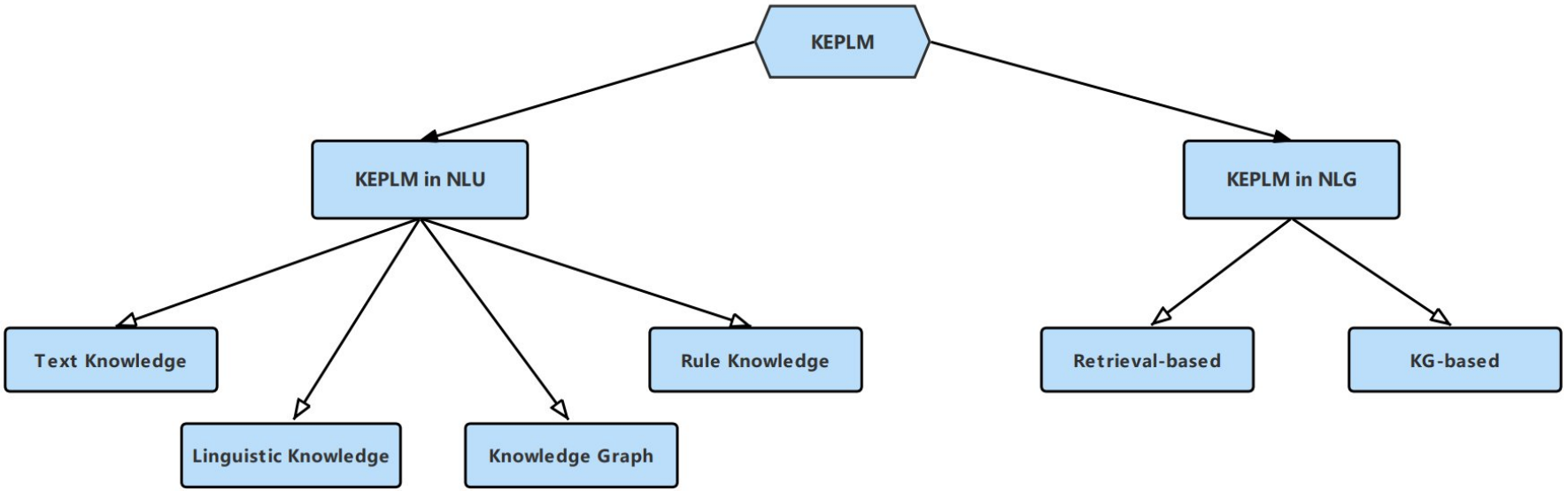
Retrieval Augmentation?

Fine Tuning?

Knowledge Injection?

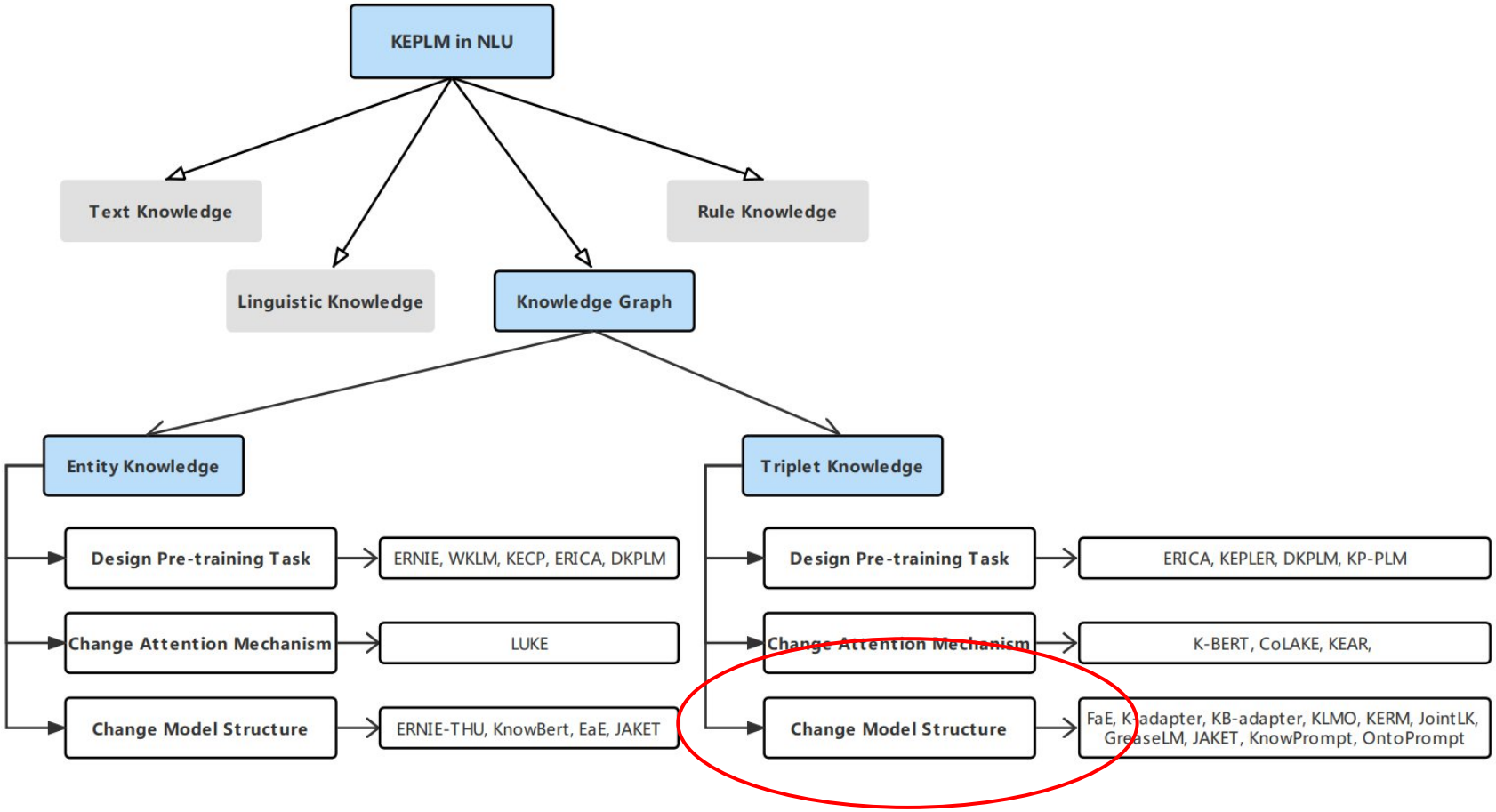
Knowledge Grounding?

Overview of Knowledge Enhancement



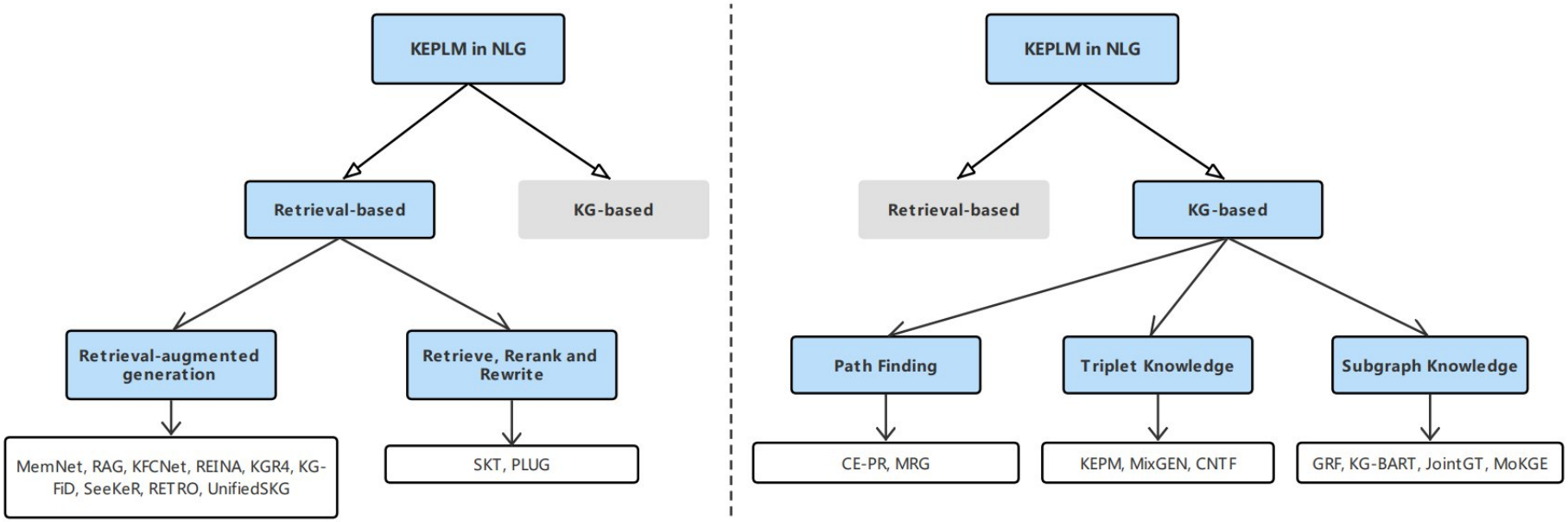
[1]

Overview of Knowledge Enhancement



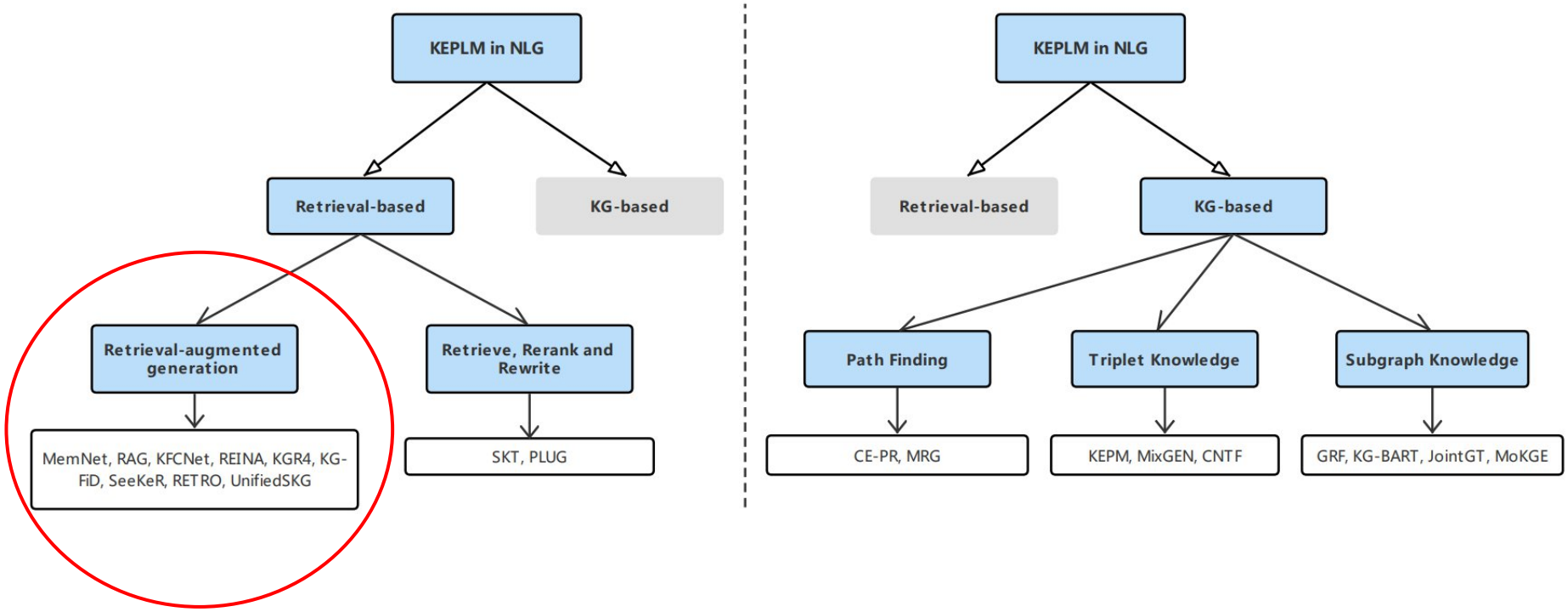
[1]

Overview of Knowledge Enhancement



[1]

Overview of Knowledge Enhancement

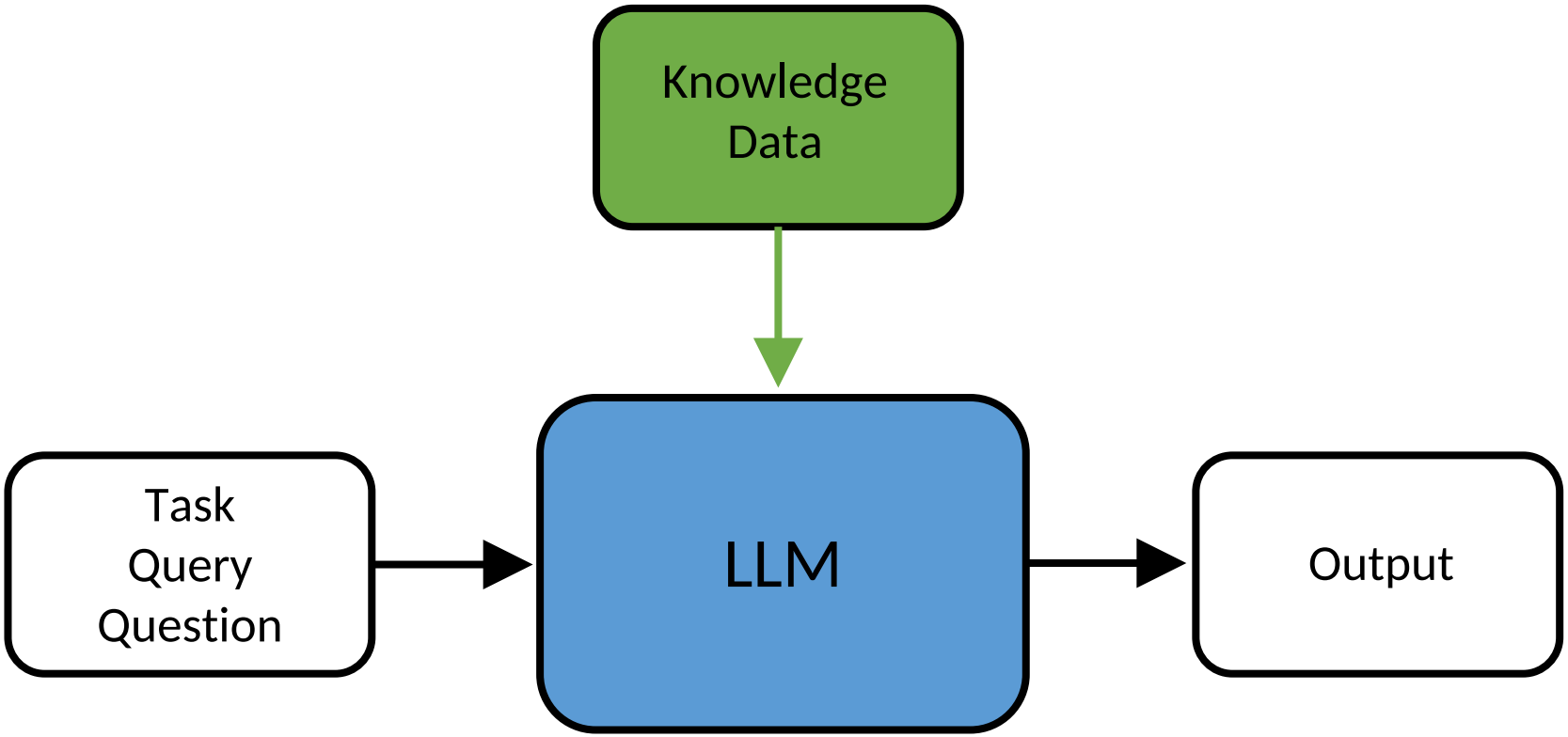


[1]

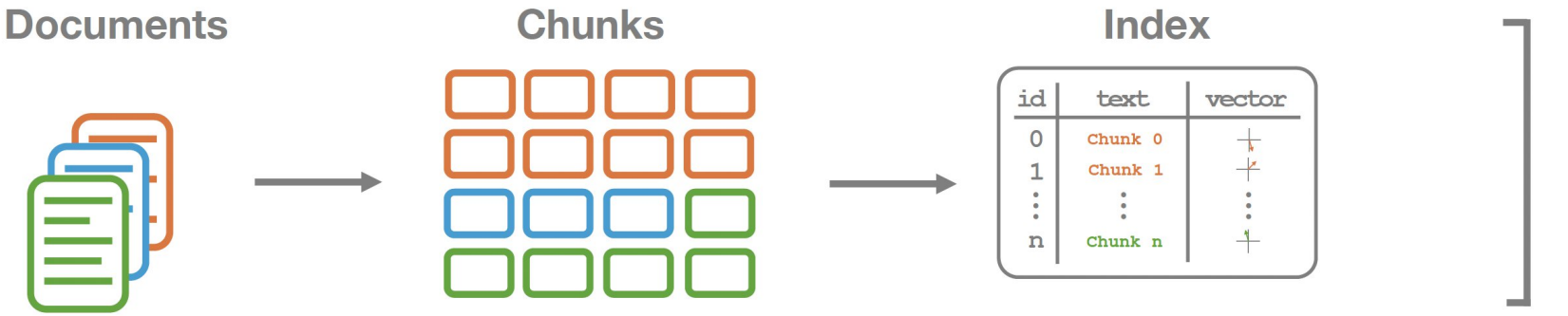
Outline

- Overview of Knowledge Enhancement
- **Retrieval Augmented Generation (RAG)**
  - **RAG-phases: IR and GEN**
  - **Evaluation**
  - **Painpoints, Advantages, Disadvantages**
  - **LLM usage**
- Adapter-based Knowledge Enhancement
- Summary

Basic RAG

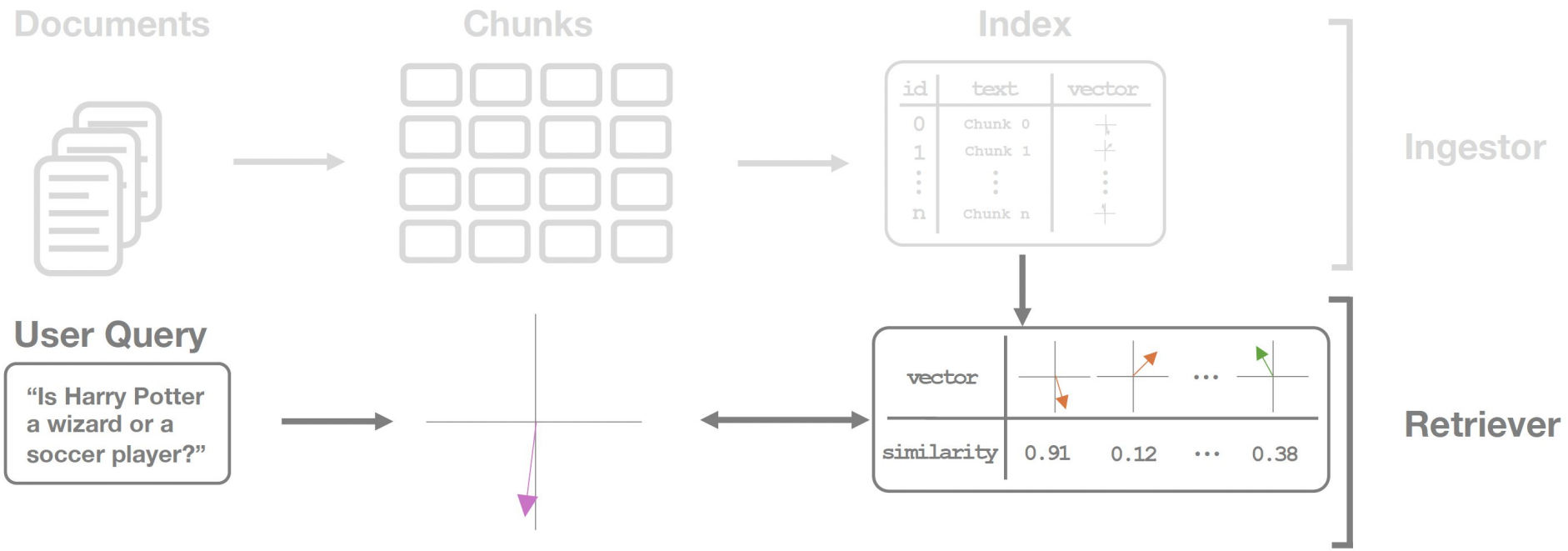


Basic RAG


Documents
Chunks
Index
id
text
vector
0
Chunk 0
1
Chunk 1
n
Chunk n
Ingestor

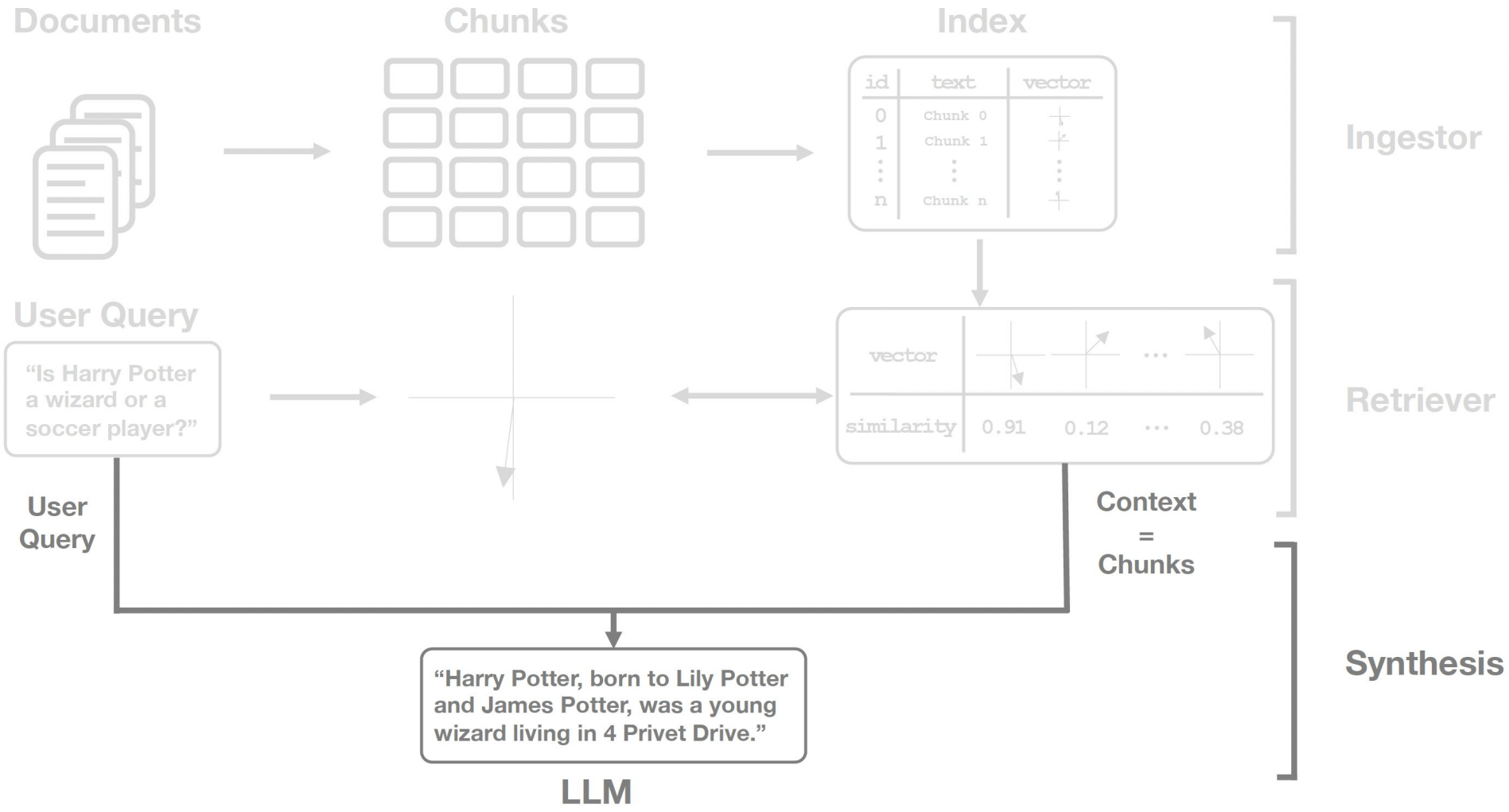
[2]

Basic RAG


Documents
Chunks
Index
id
text
vector
0
Chunk 0
1
Chunk 1
n
Chunk n
Ingestor
User Query
"Is Harry Potter a wizard or a soccer player?"
vector
similarity
0.91
0.12
0.38
Retriever

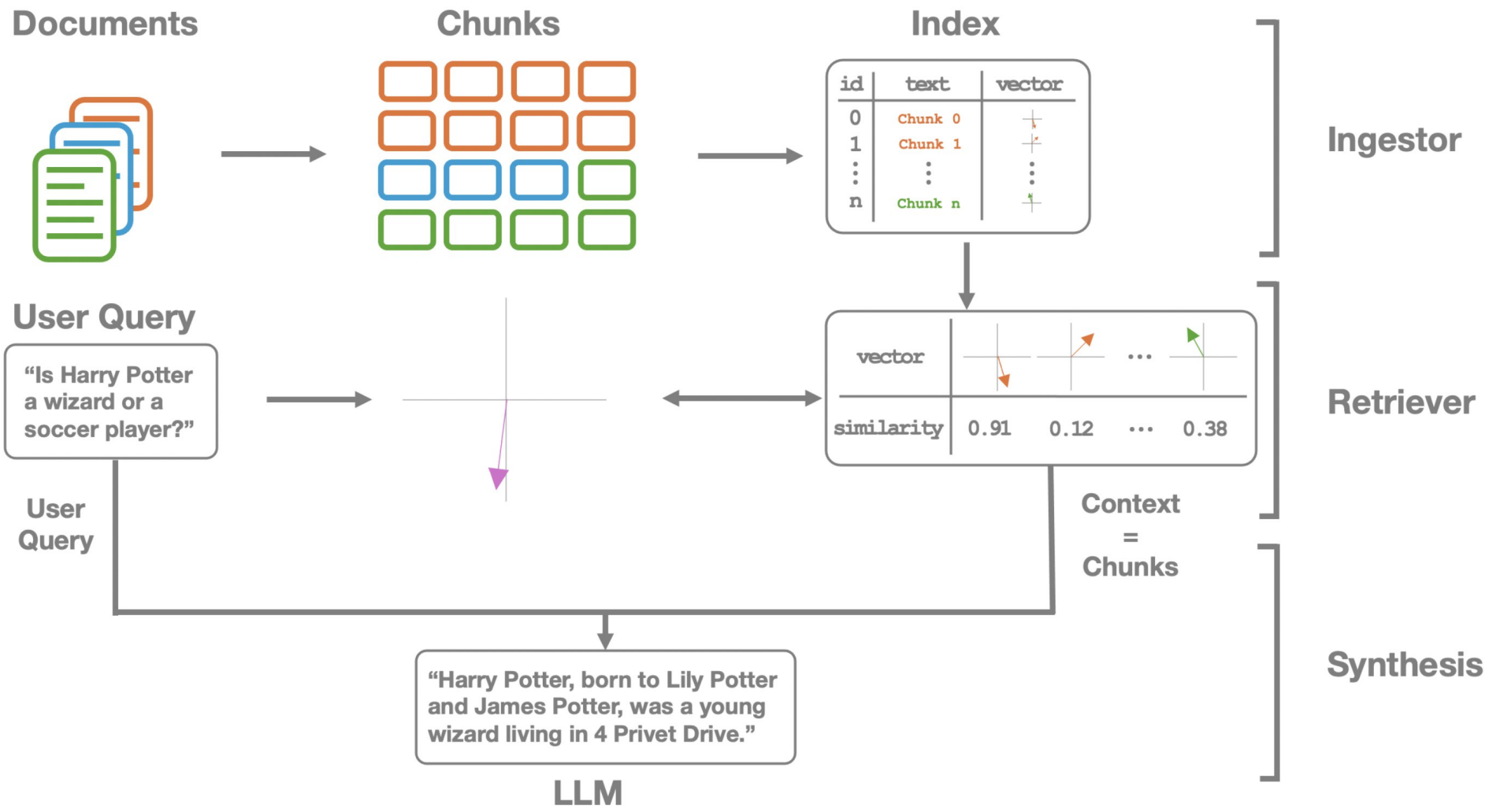
[2]

Basic RAG


Documents
Chunks
Index
id
text
vector
0
Chunk 0
1
Chunk 1
n
Chunk n
Ingestor
User Query
"Is Harry Potter a wizard or a soccer player?"
vector
similarity
0.91
0.12
0.38
Retriever
User Query
Context = Chunks
"Harry Potter, born to Lily Potter and James Potter, was a young wizard living in 4 Privet Drive."
LLM
Synthesis

[2]

Basic RAG



- Ingestor + Retriever = Information Retrieval (IR)
- IR + Synthesis = RAG

[2]

Example

First I play soccer, then I eat an apple.	<- Data
What am I eating?	<- Query
then I eat an apple.	<- Retrieved Information
You are eating an apple.	<- Answer

Overview RAG using semantic similarity

Input	Process		LLMs
Data			
	Decomposition:	Chunking	?
		Meta Data	
	Indexing:	Embedding	<-
Query			
	Chunk selection:	Query enhancement	<-
		Query embedding	<-
		Filtering (meta-data)	?
		Distance calculation	
		Chunk selection	
	Chunks to Prompt		?
	Answer Generation		<-

IR

Generation

Chunking

= Splitting long texts into concise information pieces

Purpose:

- Strip unrelated information
- Respect input bounds of LLM

Good:
First I play soccer,
then I eat an apple.

Bad:
First I play soccer,
then I eat an apple.

Splitting based on:

- Data Structure: documents, pages, tables …
- Text structure: chapters, paragraphs, sentences …
- Content length: tokens, words
- A learned heuristic = Deep Chunking

Chunk overlapping:

- Overlapping chunks reduce the risk of information being lost / destroyed due to wrong chunk boundaries
- But: Introduces redundancy

Chunk: Meta Data

Examples for Meta Data:

- Information from the parent document:
  - Document-, Chapter-Title …
  - Author, Publication Date …
  - Access Rights
- Information extracted from the chunk:
  - Topic, Sentiment

Usage: Hard filter based on Meta Data

- Pre-filter
- E.g. Access Control
- Filter creation based on query by LLM

Usage Alternative: Include meta data as text into the chunk

- Utilize Meta Data during relevance/semantic search

Chunk: Components

Chunk content, Meta Data

- Chunk value, chunk key/index, Meta Data

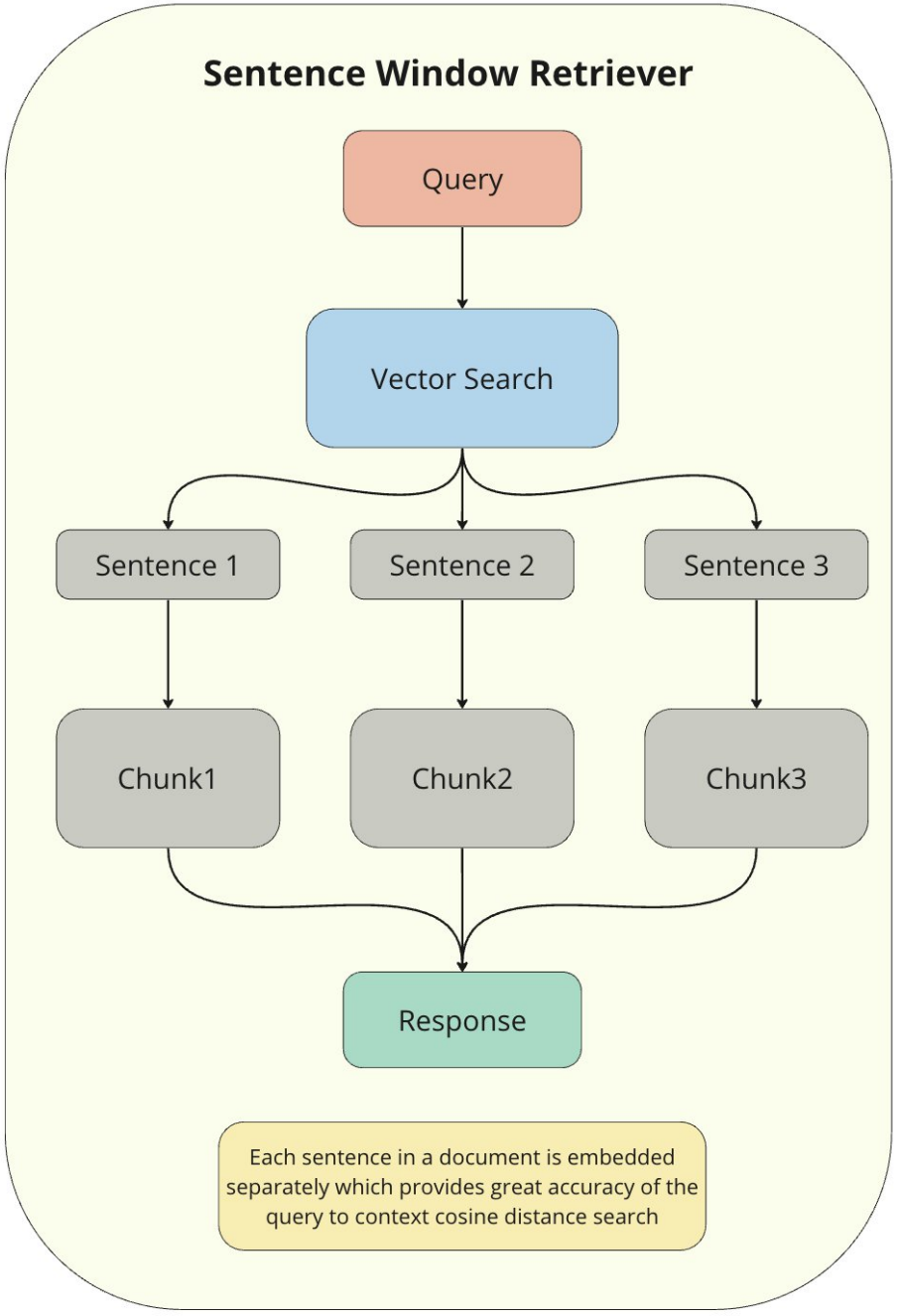
Value and Key/index are independently composed of:

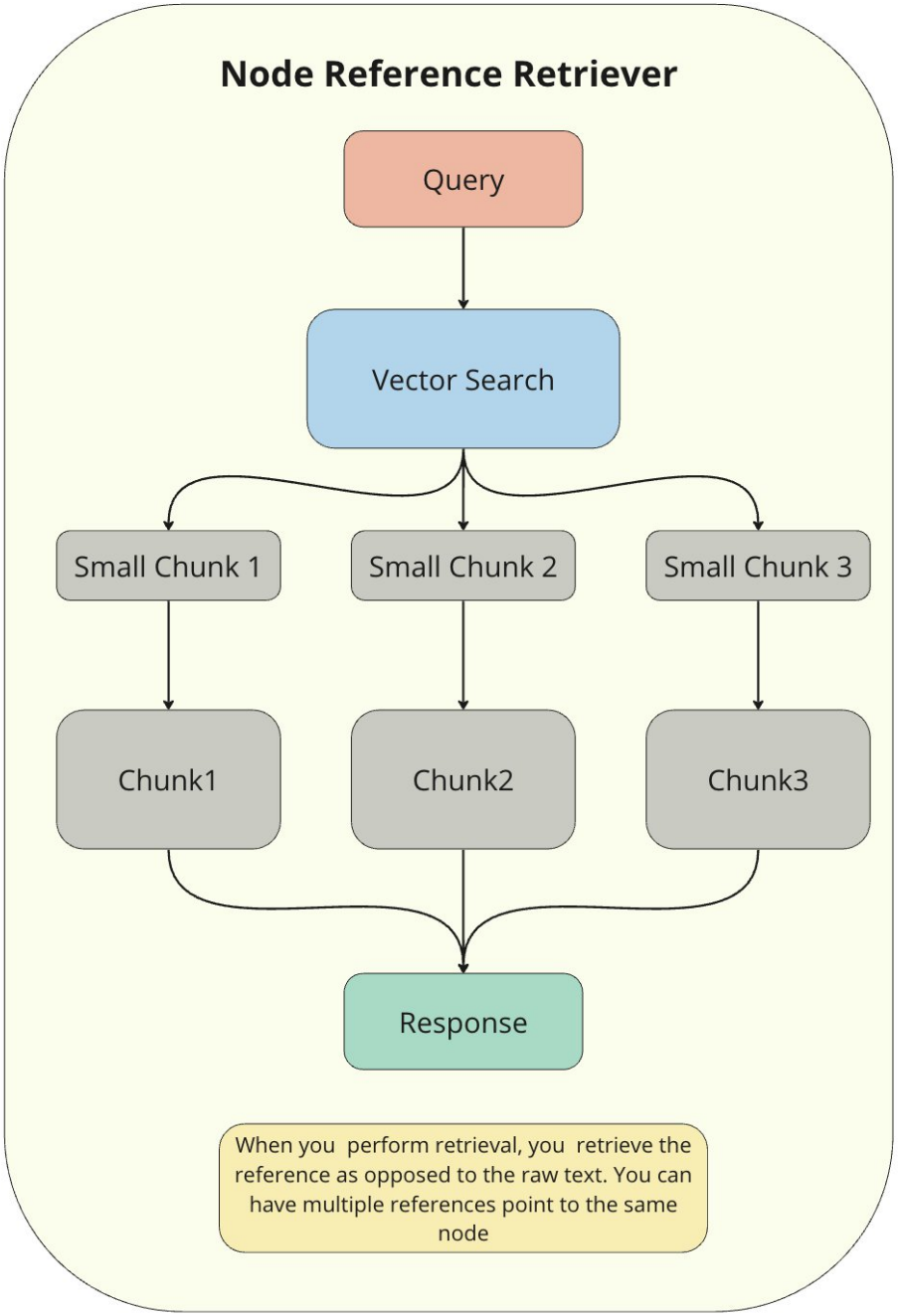
- Text Content
- Meta Data
- Surrounding context

Transformation:

- Value
  - Summarization (“compression” [4])
- Key / index:
  - Summarization
  - stop word removal
  - embedding

One chunk may also have multiple keys. This may make the search more flexible. Then deduplication is needed to prevent inclusion based on two keys.


Sentence Window Retriever
Query
Vector Search
Sentence 1
Sentence 2
Sentence 3
Chunk1
Chunk2
Chunk3
Response
Each sentence in a document is embedded separately which provides great accuracy of the query to context cosine distance search


Node Reference Retriever
Query
Vector Search
Small Chunk 1
Small Chunk 2
Small Chunk 3
Chunk1
Chunk2
Chunk3
Response
When you perform retrieval, you retrieve the reference as opposed to the raw text. You can have multiple references point to the same node

[3,15]

Embedding creation

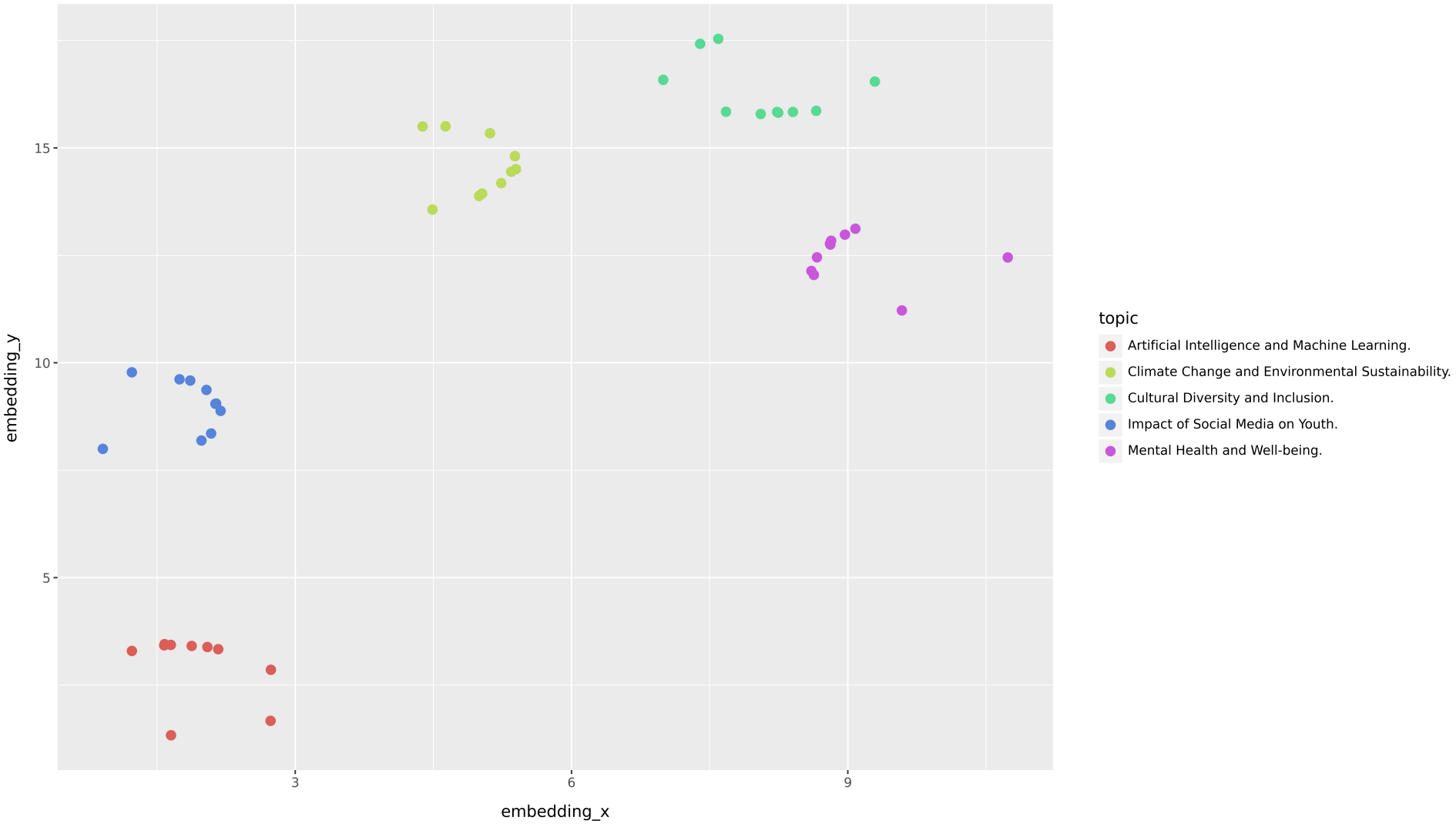
- Same LLM for chunks and query

- Single data type (text, audio, images, tables, graphs)
- Multimodal model
- Conversion to one data type (image captioning, speech-to-text …)

Overview

Input	Process		LLMs
Data			
	Decomposition:	Chunking	?
		Meta Data	
	Indexing:	Embedding	<-
Query			
	Chunk selection:	Query enhancement	<-
		Query embedding	<-
		Filtering (meta-data)	?
		Distance calculation	
		Chunk selection	
	Chunks to Prompt		<-
	Answer Generation		

Distance in embedding space -> Similarity / Relevancy



Chunk Selection

- Distance calculation: Cosine similarity between embedding of query and chunks

- Selection:
  - Top k
  - Maximum marginal relevance (MMR)[4]: From top k select the n most diverse (with respect to pairwise similarity)

Text	Rank	Similarity
A	1	0.91
A	2	0.91
B	3	0.8
C	4	0.78
D	5	0.72
E	6	0.7
F	7	0.65

Method	Result RAW	Result Set
Top 2	A A	A
Top 3	A A B	A B
MMR(3,2)	A B	A B

MMR: 2 from 3

Text	Rank	Similarity
A	1	0.91
A	2	0.91
B	3	0.8
C	4	0.78
D	5	0.72
E	6	0.7
F	7	0.65

RAG prompt example

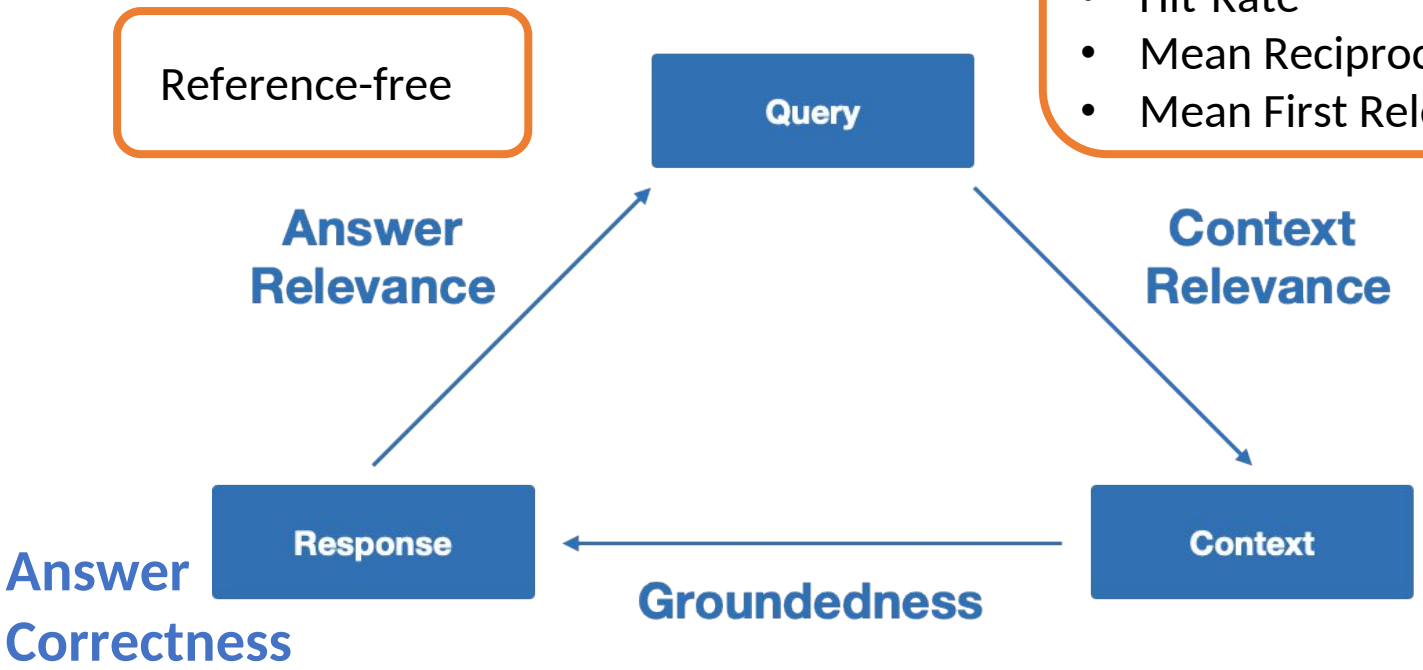
You are an assistant for question-answering tasks. Use the following pieces of retrieved context to answer the question. If you don't know the answer, just say that you don't know. Use three sentences maximum and keep the answer concise.
Question: {question}
Context: {context}
Answer:

[14]

Overview

Input	Process		LLMs
Data			
	Decomposition:	Chunking	?
		Meta Data	
	Indexing:	Embedding	<-
Query			
	Chunk selection:	Query enhancement	<-
		Query embedding	<-
		Filtering (meta-data)	?
		Distance calculation	
		Chunk selection	
	Chunks to Prompt		<-
	Answer Generation		

Eval: RAG Triad



Reference-free

Reference-based:
- Recall
- Precision
- Hit-Rate
- Mean Reciprocal Rank (MRR)
- Mean First Relevant (MFR)

Reference-based:
= text-to-text compare
- BertScore
- Rouge

Measure hallucination,
Reference-free

[2,15]

Context relevance / IR eval

a: relevant documents that are retrieved
b: not relevant documents that are retrieved
c: relevant documents that are not retrieved
d: and not relevant documents that are not retrieved

Hit-Rate: Number of test queries that retrieved at least one relevant item

K = 1

Text	Rank	Relevant?
A	1	+
B	2	-
C	3	+
D	4	-
E	5	-
F	6	+
G	7	+

Text	Rank	Relevant?
B	1	-
A	2	+
C	3	+
F	4	+
D	5	-
F	6	-
G	7	+

Recall	a/(a+b)	1/(1+0) = 1	0/(0+1) = 0
Precision	a/(a+c)	1/(1+3) = 0.25	0/(0+4) = 0
Hit-Rate		1	0

[5,6]

Context relevance / IR eval

a: relevant documents that are retrieved
b: not relevant documents that are retrieved
c: relevant documents that are not retrieved
d: and not relevant documents that are not retrieved

Hit-Rate: Number of test queries that retrieved at least one relevant item

K = 2

Text	Rank	Relevant?
A	1	+
B	2	-
C	3	+
D	4	-
E	5	-
F	6	+
G	7	+

Text	Rank	Relevant?
B	1	-
A	2	+
C	3	+
F	4	+
D	5	-
F	6	-
G	7	+

Recall	a/(a+b)	1/(1+1) = 0.5	1/(1+1) = 0.5
Precision	a/(a+c)	1/(1+3) = 0.25	1/(1+3) = 0.25
Hit-Rate		1	1

[5,6]

Context relevance / IR eval

r = the rank of the first relevant item

Mean Reciprocal Rank (MMR): RR(r)= 1/r; Mean over all queries
Mean First Relevant(MFR): r; Mean over all queries

Not dependent on selection of K

Text	Rank	Relevant?
A	1	+
B	2	-
C	3	+
D	4	-
E	5	-
F	6	+
G	7	+

Text	Rank	Relevant?
B	1	-
A	2	+
C	3	+
F	4	+
D	5	-
F	6	-
G	7	+

MRR	1/1 = 1	1/2 = 0.5
MFR	1	2

[7]

Painpoints of RAG

- Duplicates / Quasi-Duplicates
- Find questions instead of answers
- Information corruption due to suboptimal chunks
- No relevant information provided in topK and the synthesis model can’t reliably deal with it
- Lack of exact match capabilities
- Lost in the middle

‘Find questions instead of answers’; ‘To specific wording in the question’ -> Query expansion

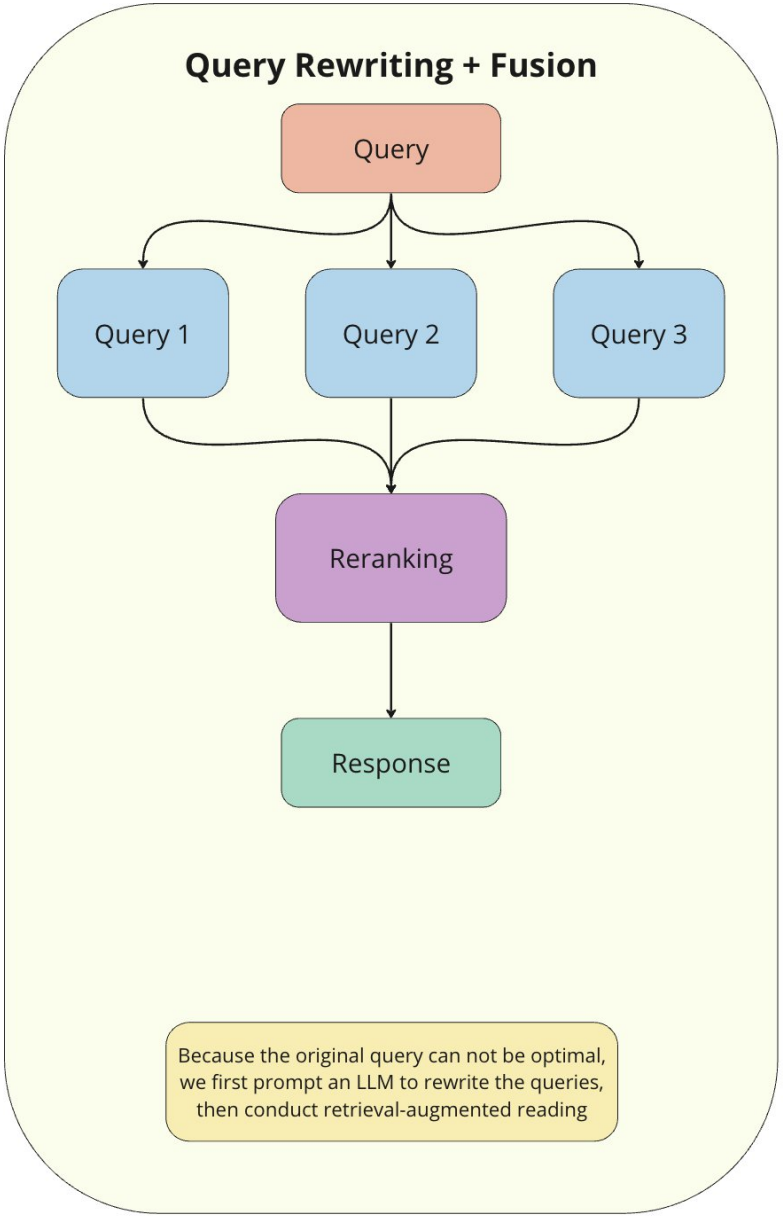
Expansion with generated hypothetical answers:

- For a give query:
- Guess potential answers using a LLM
- IR for this answers and original query
- Mix the retrieved items and rerank them
- -> Find answer instead of question

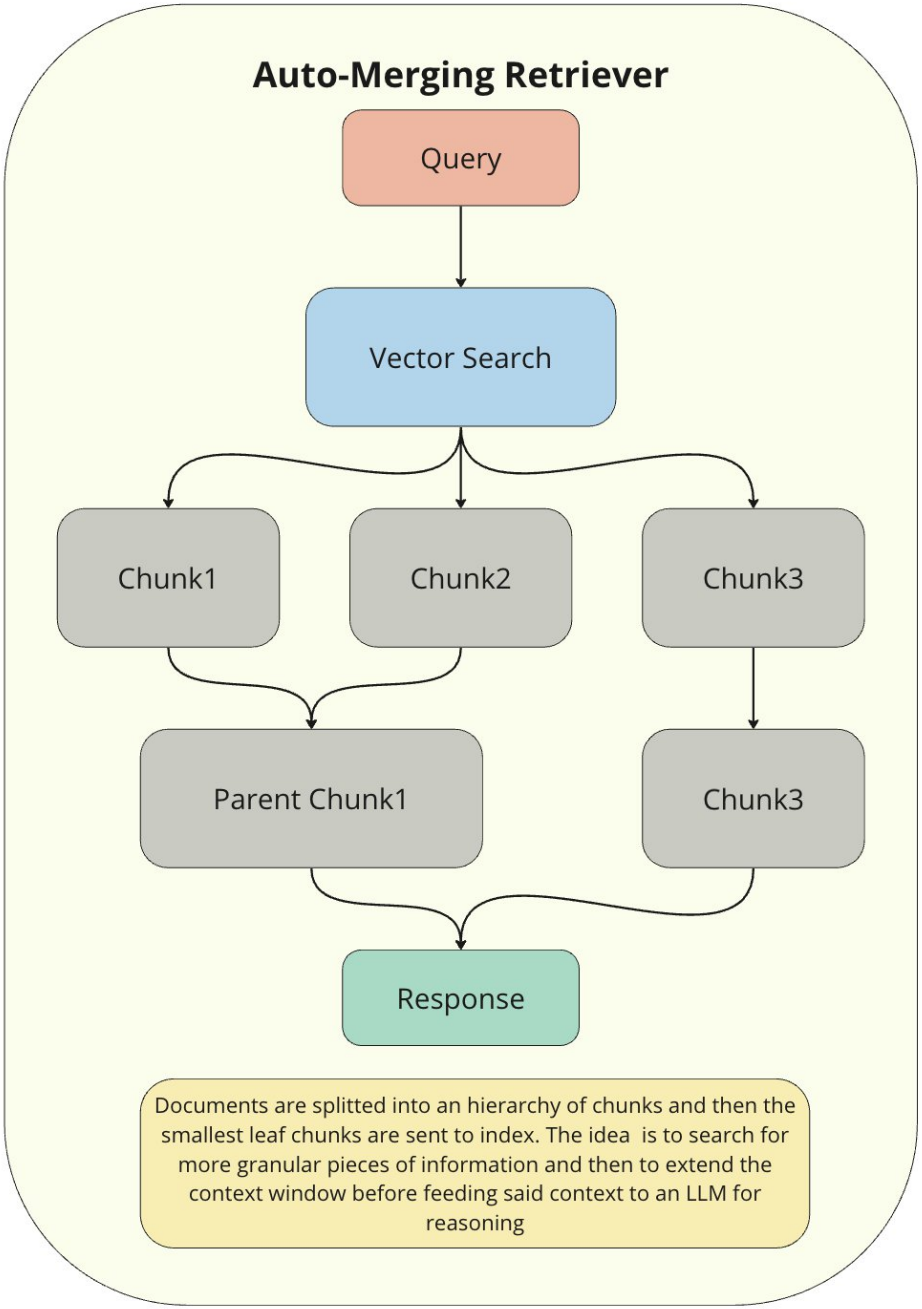
Expansion with multiple queries:[Troynikov2024]

- Generate related queries using LLM, e.g. sub-questions, paraphrasing …

[13,3]



‘Information corruption due to suboptimal chunks’ ->



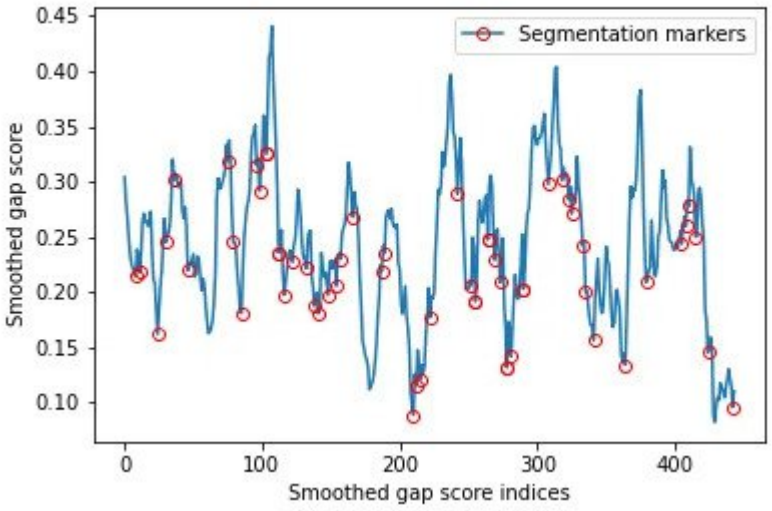
- Finegrained selection of information pieces
- large context
- no redundancy
- respects text relation (order, neighborhood etc.)

[3,15]

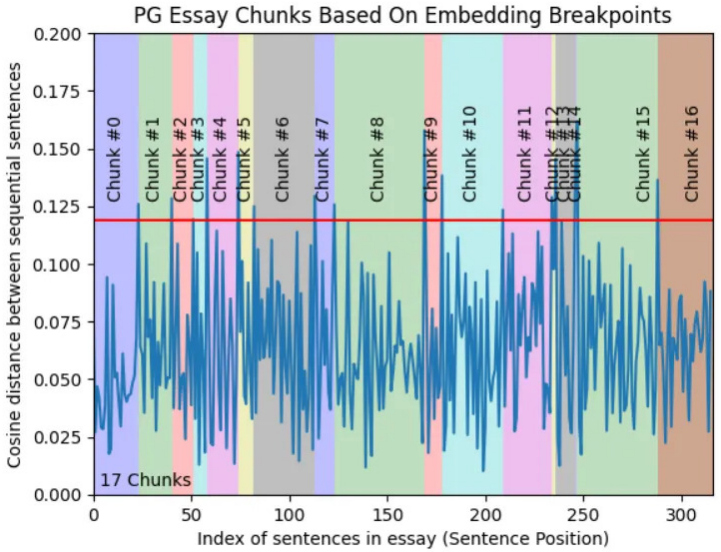
‘Information corruption due to suboptimal chunks’ -> Semantic Chunking

- Idea: Split text when the topic shifts from one sentence to another.

- For each pair of consecutive sentences:
  - split if the cosine similarity of the sentence embeddings is bellow threshold



- cosine similarity
- smoothed
- local minima



- cosine distance
- global maxima

- Threshold selection: e.g. based on standard deviation or percentiles

[8,9,10,16]

‘Information corruption due to suboptimal chunks’ -> Recursive / Hirarchical Chunking

- Motivation:
  - Combine the power of multiple strategies
  - Apply each on the level it is the strongest

- Algorithm:
  - Define a ordered list of chunking strategies
  - Divide the text into smaller chunks iteratively
  - In each step: If a chunk is too big, utilize the next strategy to split it again.

- E.G
  1. splitting based on document structure (headlines etc.)
  2. semantic chunking
  3. splitting based on text length

‘No relevant information provided in topK and the synthesis model can’t reliably deal with it’ -> Retrieval score threshold

Two examples for top3 retrieval:

Text	Rank	Similarity	Relevant
A	1	0.95	+
B	2	0.94	+
C	3	0.92	-

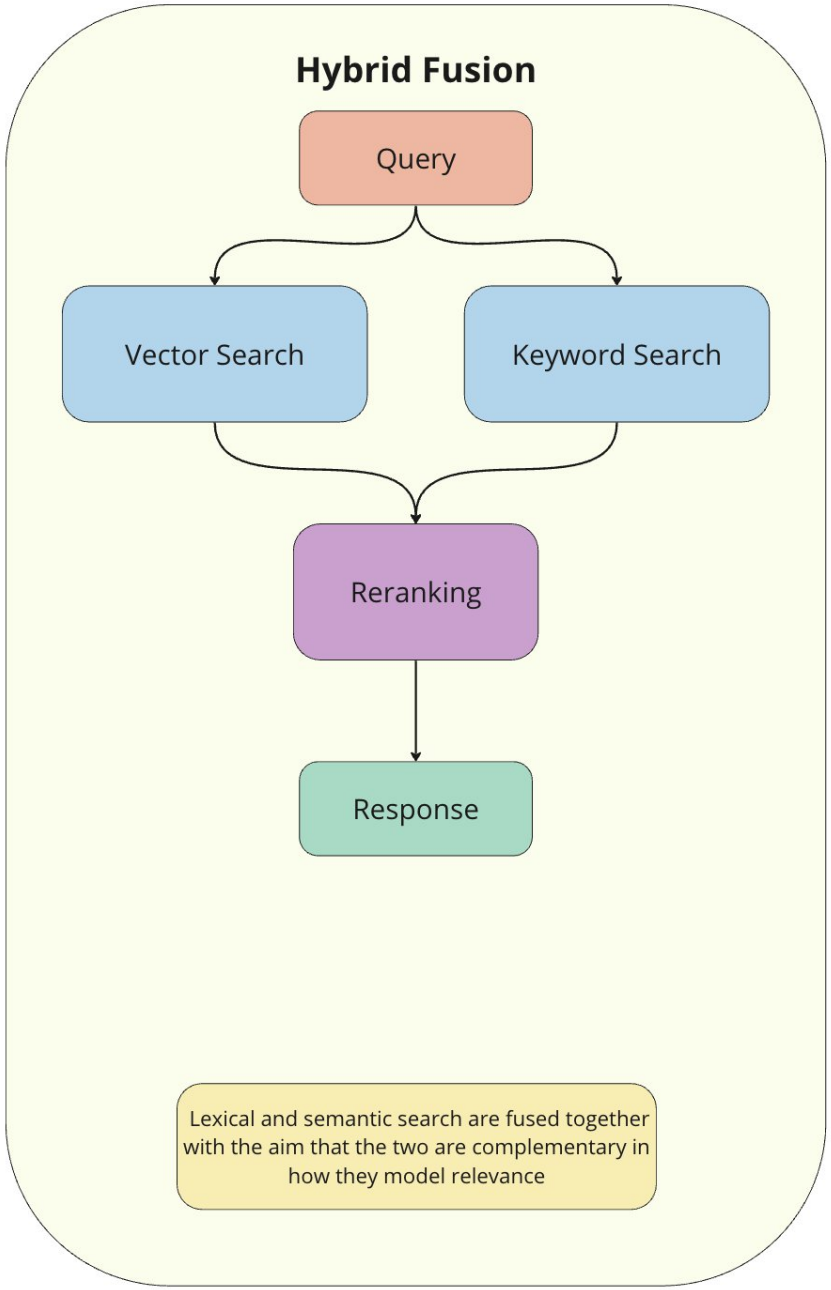
- Relevant items included
- The synthesis model needs to deal only with a small amount of noise

Text	Rank	Similarity	Relevant
C	1	0.92	-
D	2	0.90	-
E	3	0.89	-

- No relevant items included
- The synthesis model needs to deal with noise only

- Distinguishing these situations is a challenge for the synthesis model.
- To assist, items below a similarity threshold (here 0.92) could be removed from the retrieval. An empty retrieval set could be replaced with a special string in the prompt (“no information found”).
- Be careful, there might not be a clear threshold valid for all cases.

‘Lack of exact match capabilities’ -> Hybrid Search / Fusion



[3]

‘Lost in the middle’ -> LongContextReorder

“A study [Liu2023b] observed that the best performance typically arises when crucial data is positioned at the start or conclusion of the input context.

LongContextReorder was designed [(see LlamaIndex)] to address this ‘lost in the middle’ problem by re-ordering the retrieved nodes, which can be helpful in cases where a large top-k is needed.” [12]

Used LLM components

- Embedding Model: Chunks and Queries
- Query expansion Model
- Generation Model
- Chunking Model

- Usage of pretrained all-purpose LLMs possible

- Benefits of finetuning
  - Generation LLM: adapted to handle the RAG prompt well
  - All LLMs: Adapt to special domain language (e.g. technical abbreviations)

RAG Advantages + Disadvantages

Advantages:

- Some degree of explainability as the IR intermediate result is observable and human readable.
- Reuse of pretrained components
- Separation of knowledge and language -> updating information without training possible

Disadvantages:

- Complicated training:
  - Train components independently
  - Or train the whole pipeline: multi-component pipeline is non-differentiable -> no backprop -> need for reinforcement learning
- Complicated debugging: multiple components -> multiple possible error sources

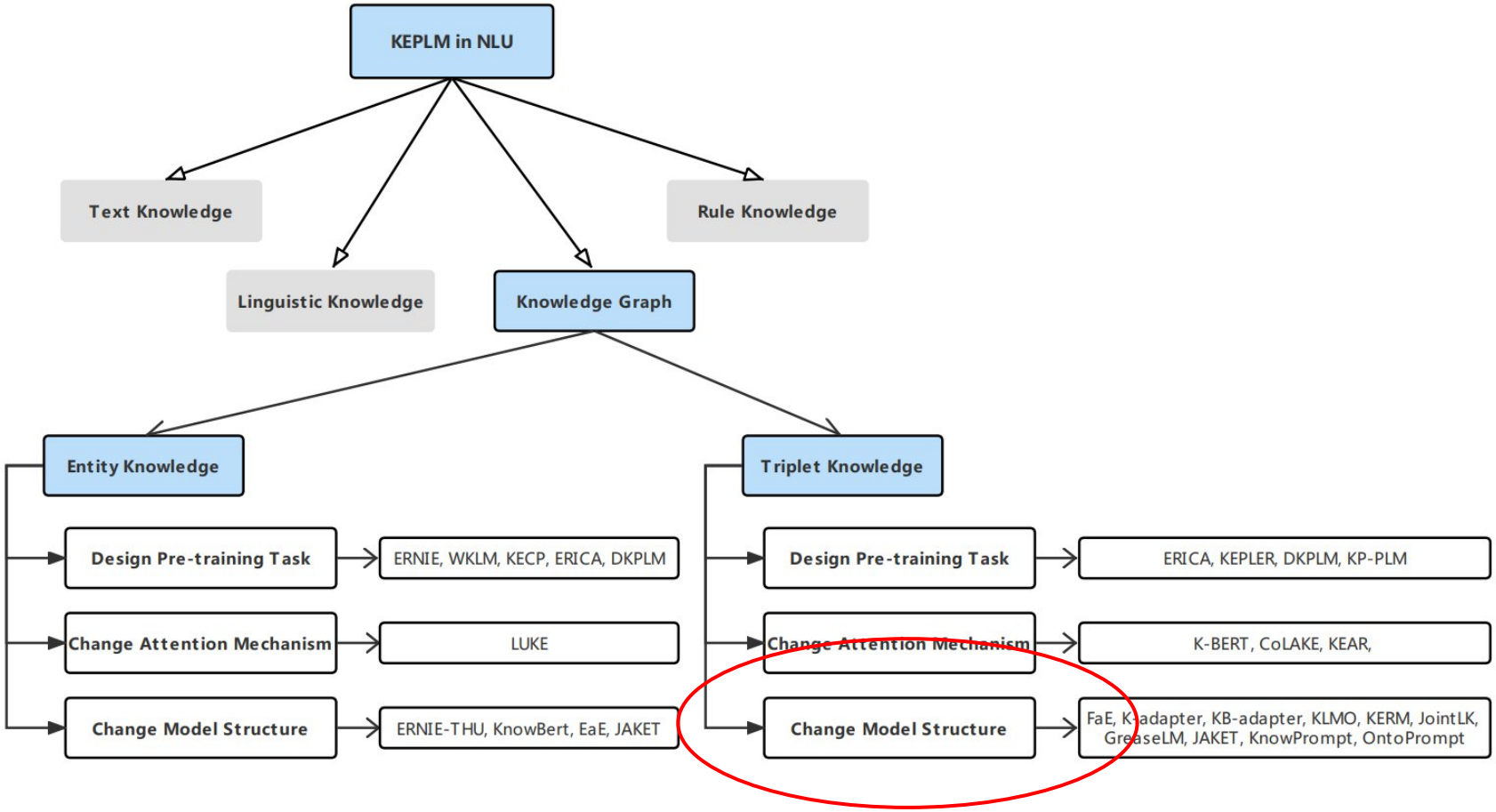
Backup: (buzzwords, which you might use for googling)

Recursive/iterative Retrieval
Hierarchical retrieval
Agent-based RAG
Attacks on RAG: adversarial data injection.
Reranker
Approximate nearest neighbor search (ANN)
Distractor (in the context of IR): A retrieved result which is not relevant to the query. [13]

Outline

- Overview of Knowledge Enhancement
- Retrieval Augmented Generation (RAG)
- **Adapter-based Knowledge Enhancement**
  - **Overview & Motivation**
  - **Adapters**
  - **Applications**
- Summary

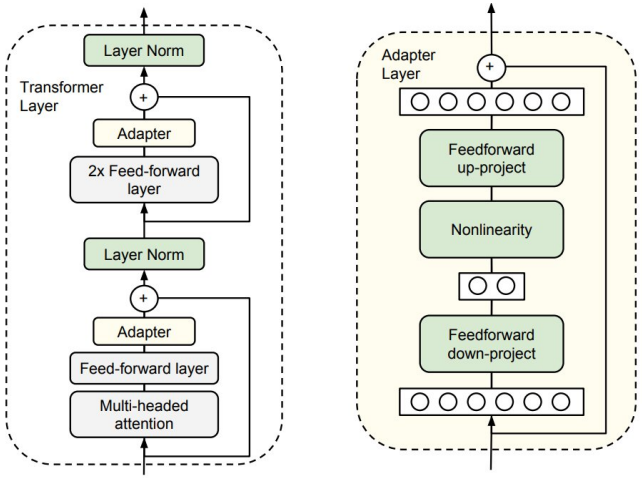
Adapter-based Knowledge Enhancement



[1]

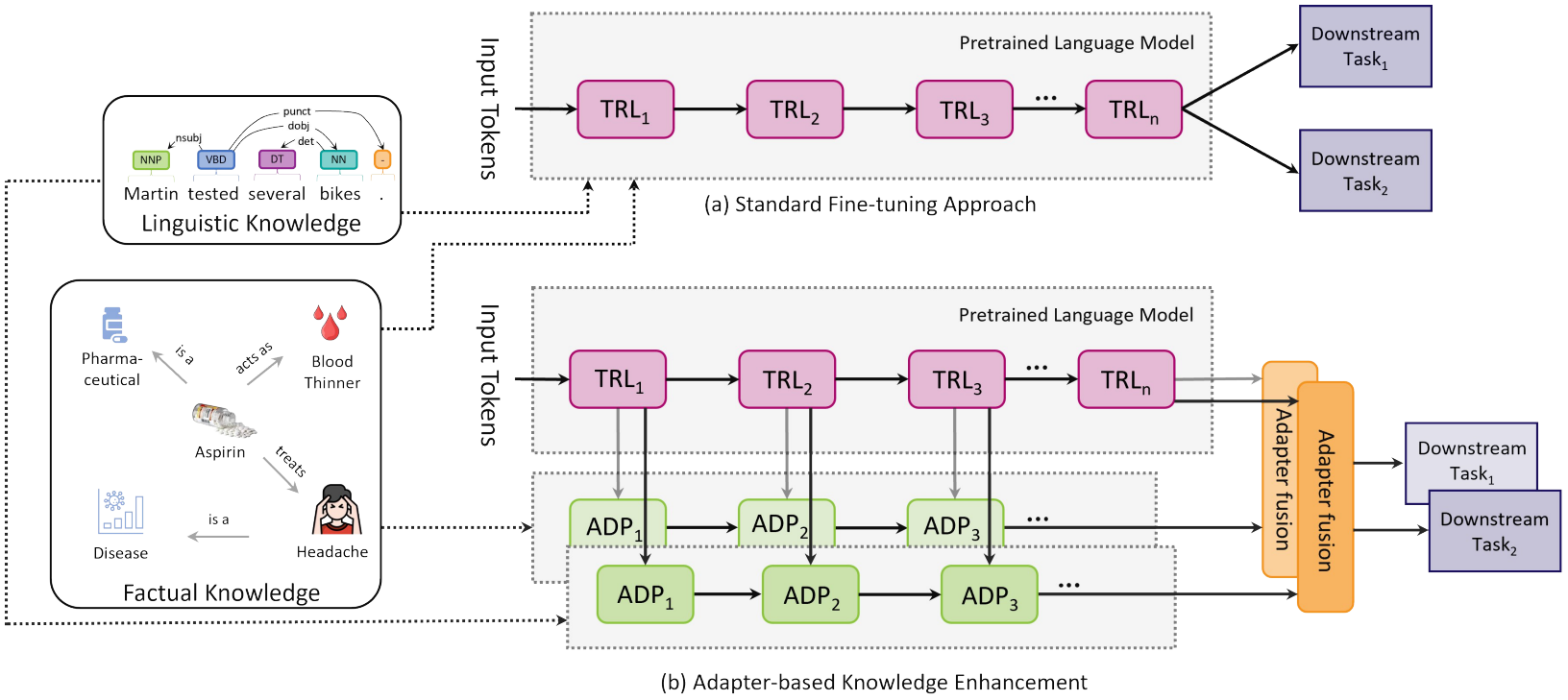
Overview & Motivation

- Most knowledge enhancement methods require the training of additional LMs and costly long training periods
- Augmenting pre-trained LMs can cause catastrophic forgetting
- -> Adapters as a simple, robust, and parameter efficient method to inject structured knowledge



[16]

Overview & Motivation

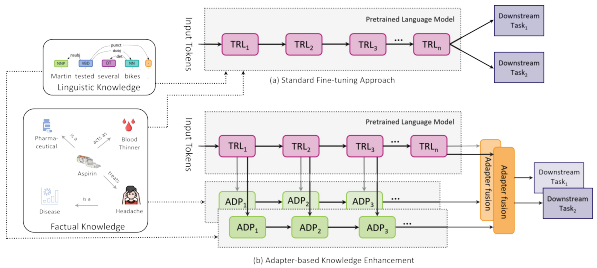


Overview & Motivation

Medical natural language inference task (NLI):

- **Patient Premise**:
  No history of blood clots or DVTs, has never had chest pain prior to one week ago

- **Hypothesis**:
  Patient has angina



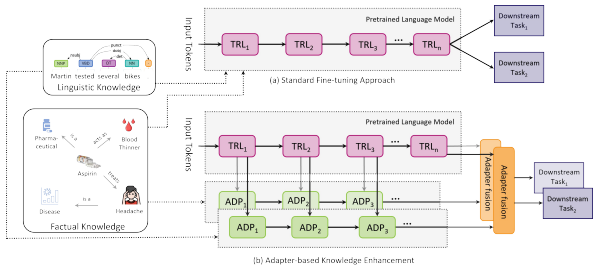
Overview & Motivation

Medical natural language inference task (NLI):

- **Patient Premise**:
  No history of blood clots or DVTs, has never had chest pain prior to one week ago

- **Hypothesis**:
  Patient has angina

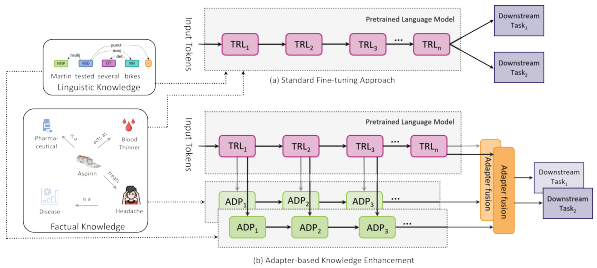
- **Correct Classification**:
  Entailment



Overview & Motivation

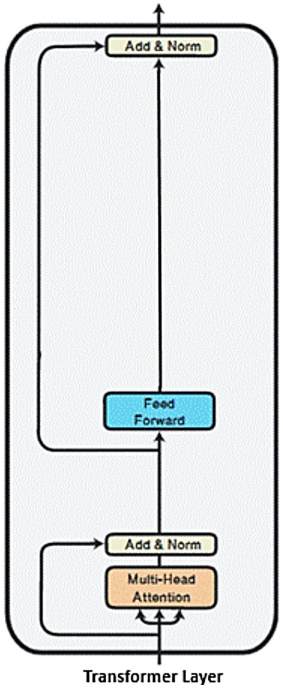
Medical natural language inference task (NLI):

- **Patient Premise**:
  No history of blood clots or DVTs, has never had chest pain prior to one week ago
- **Hypothesis**:
  Patient has angina
- **Correct Classification**:
  Entailment



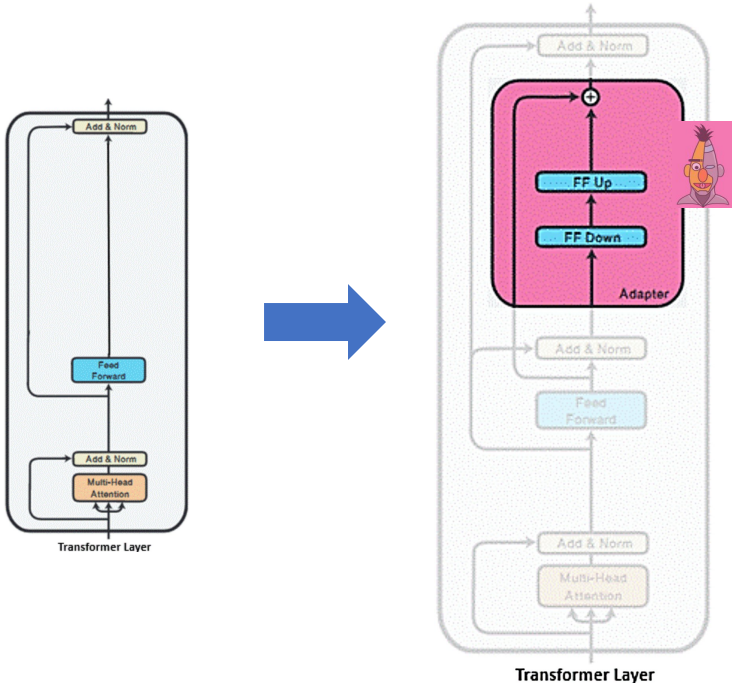
Correct inference more likely if model specifically learned relations and synonyms

Adapters


Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Transformer Layer

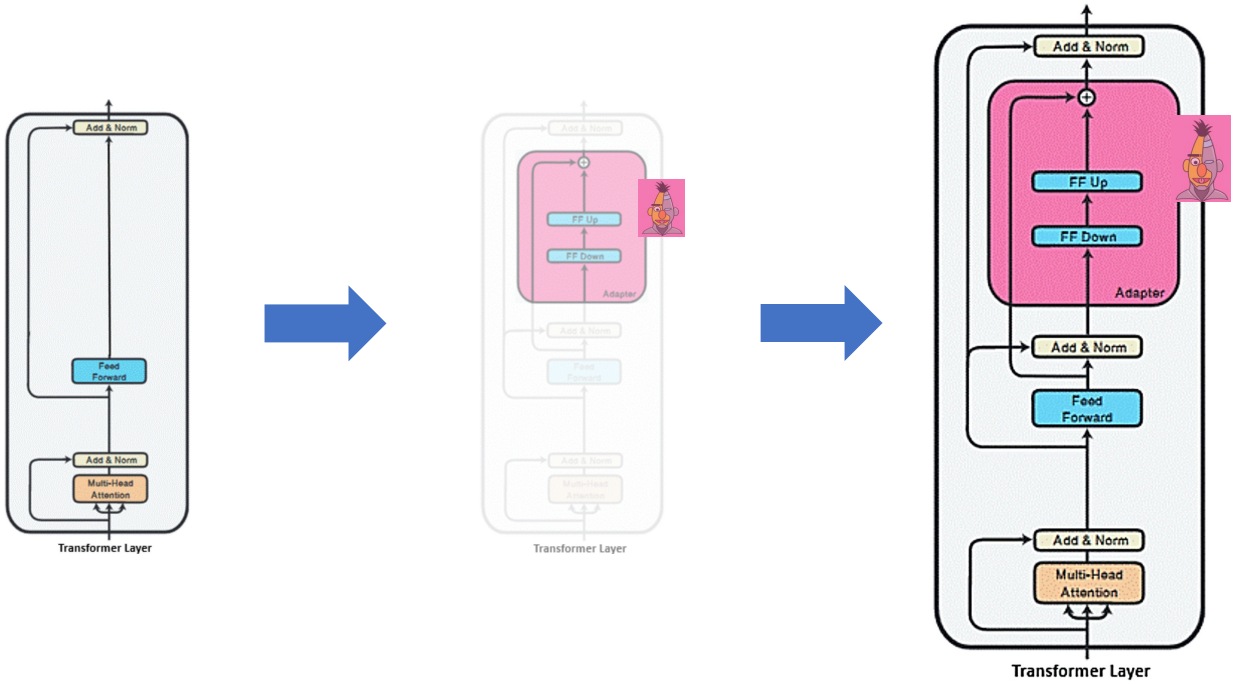
[17]

Adapters


Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Transformer Layer
Add & Norm
FF Up
FF Down
Adapter
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Transformer Layer

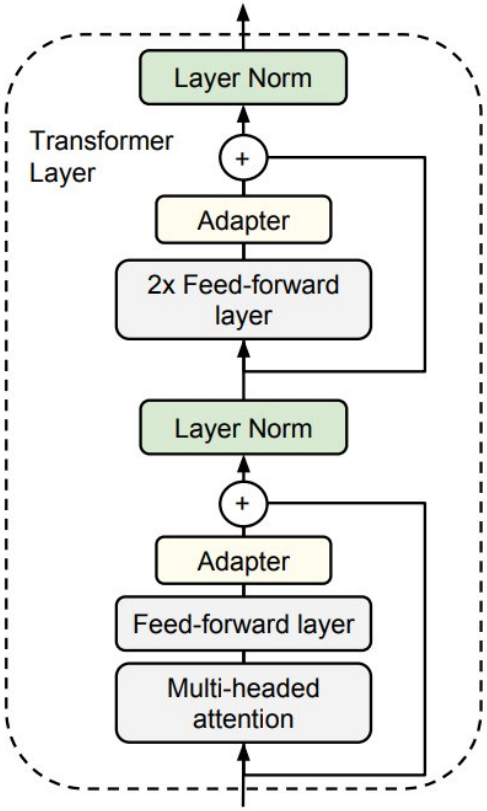
[17]

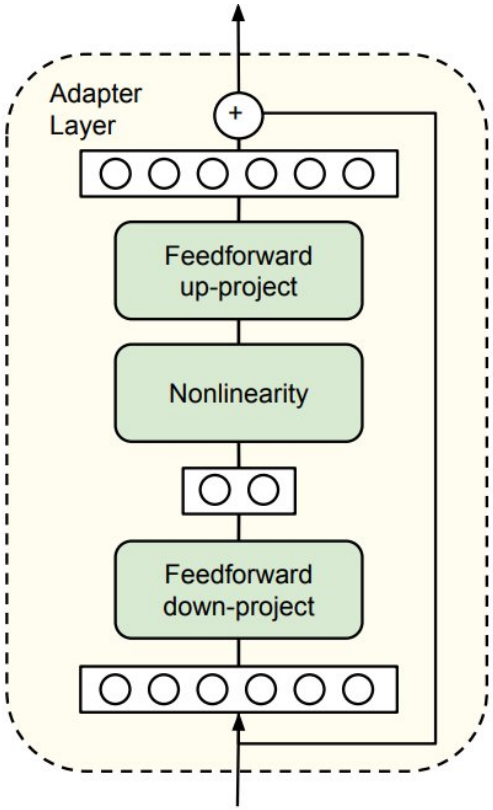
Adapters


Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Transformer Layer
FF Up
FF Down
Adapter
Transformer Layer
Add & Norm
FF Up
FF Down
Adapter
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Transformer Layer

[17]

Adapters


Transformer Layer
Layer Norm
Adapter
2x Feed-forward layer
Layer Norm
Adapter
Feed-forward layer
Multi-headed attention


Adapter Layer
Feedforward up-project
Nonlinearity
Feedforward down-project

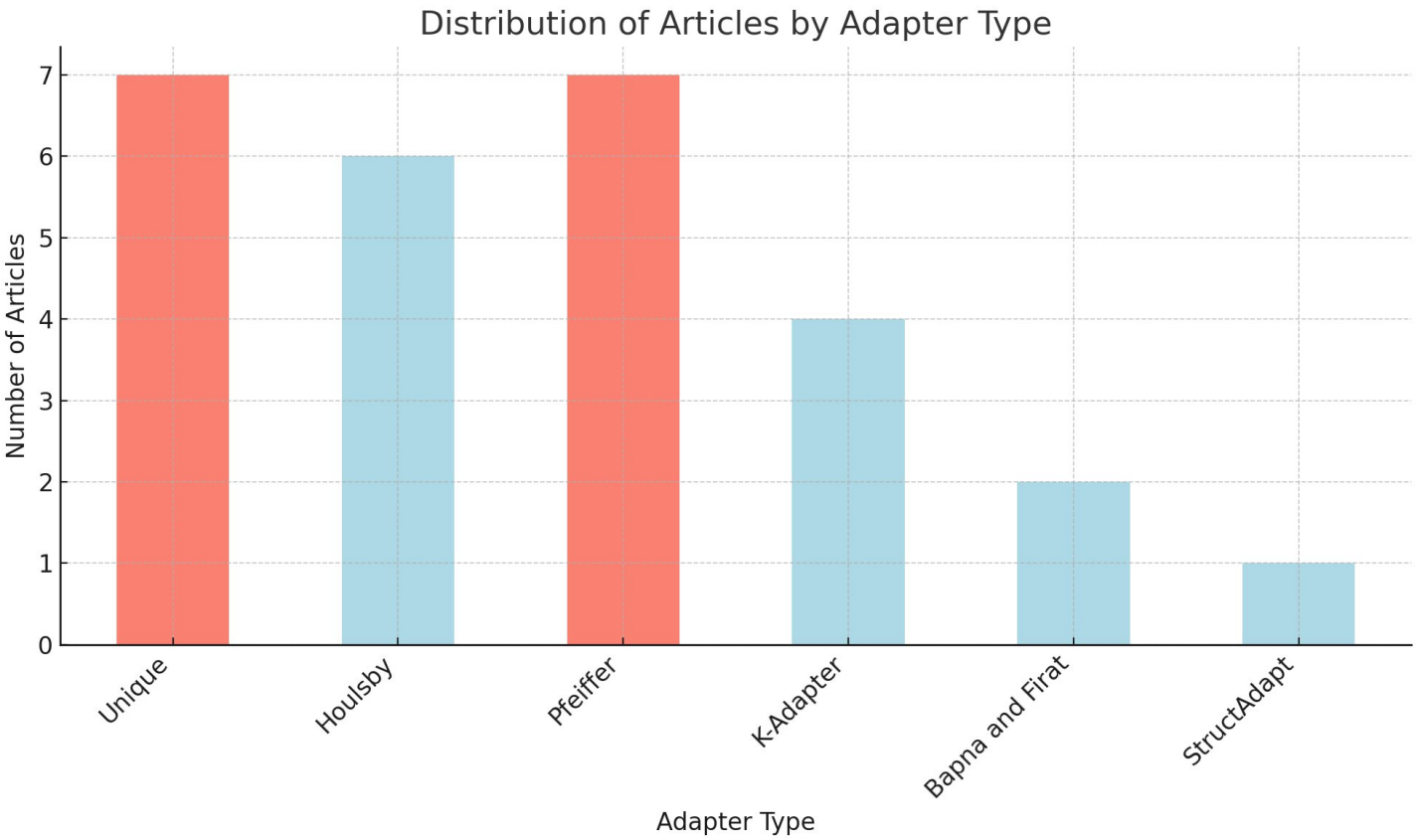
[16]

Adapters

- Only 1% - 8% of base model parameters need to be updated during training
- Prevents catastrophic forgetting due to freezing of pre-trained model parameters
- Can lead to a delay in inference time (in contrast to LoRA)

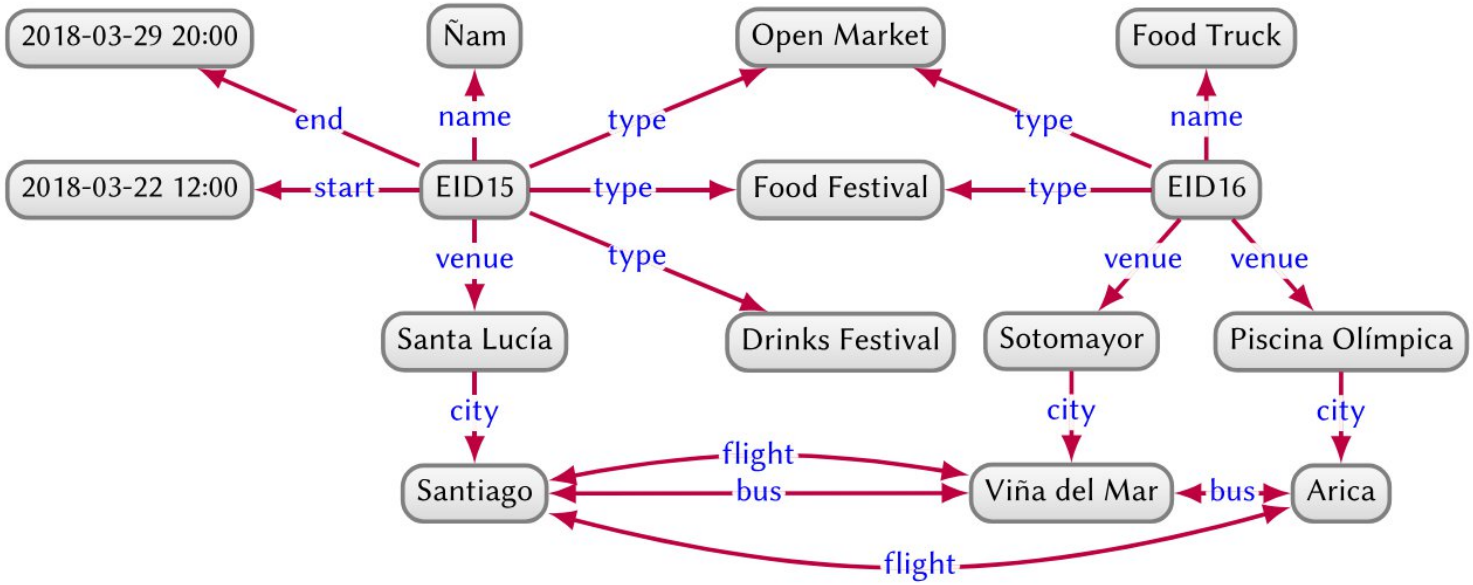
[16]

Adapters


Distribution of Articles by Adapter Type
Number of Articles
0
1
2
3
4
5
6
7
Unique
Houlsby
Pfeiffer
K-Adapter
Bapna and Firat
StructAdapt
Adapter Type

Knowledge Graphs

- Hogan et al. (2021) define a KG as *"a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between these entities"*



Directed-edge labelled graph describing events and their venues as one example of a KG

[18]

Knowledge Graphs

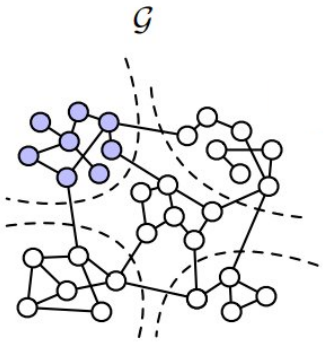
- KGs be extremely large (e.g. UMLS with 1.7GB - 27.1GB)
- Adapters can only capture a limited amount of data

➢ Sub-graph partitioning

  - Optimize for minimal data-loss ect.
  - ➢ Solved NP complete problem (METIS algorithm)
  - Leads to specialised adapters for every sub-graph

[19]

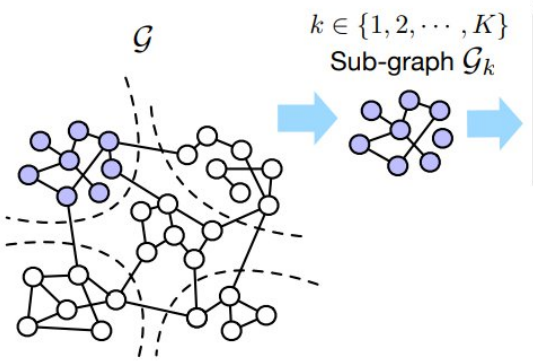
Adapter-based Knowledge Enhancement



(a) Graph Partitioning

[19]

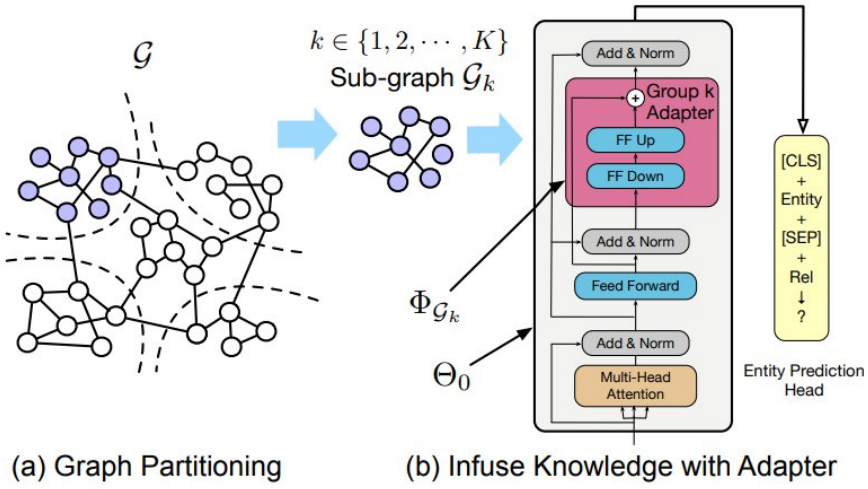
Adapter-based Knowledge Enhancement



(a) Graph Partitioning

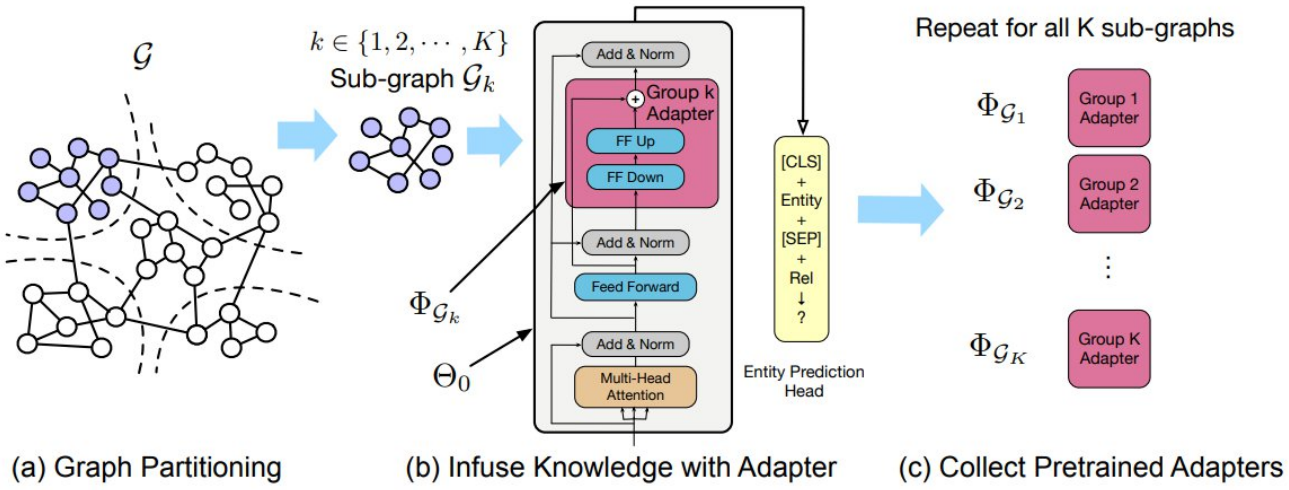
[19]

Adapter-based Knowledge Enhancement



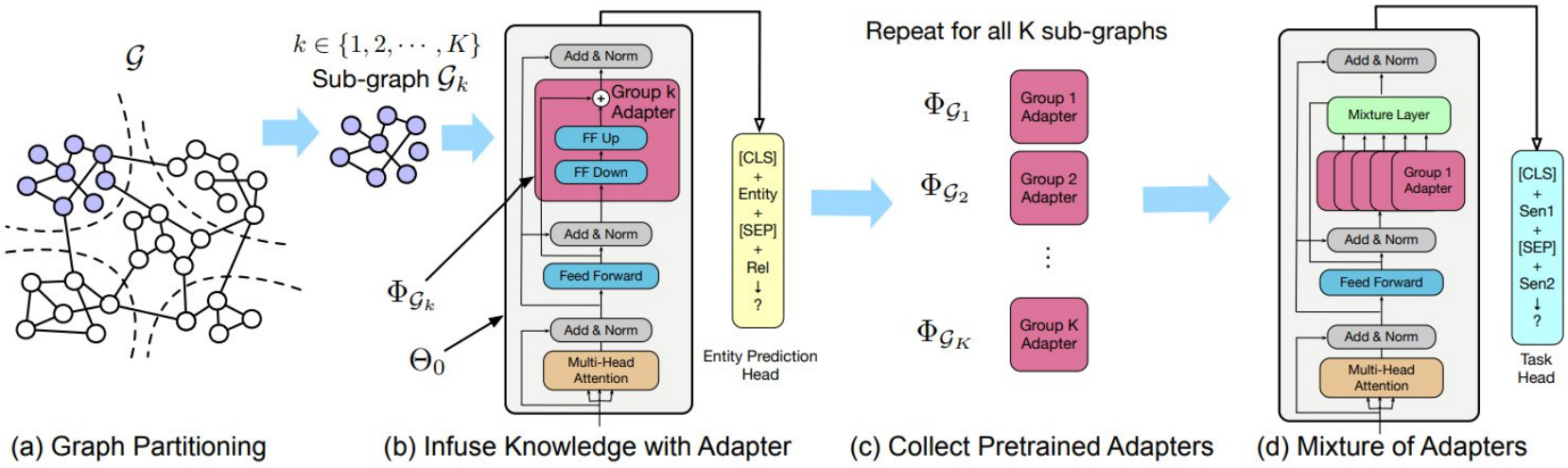
[19]

Adapter-based Knowledge Enhancement



[19]

Adapter-based Knowledge Enhancement



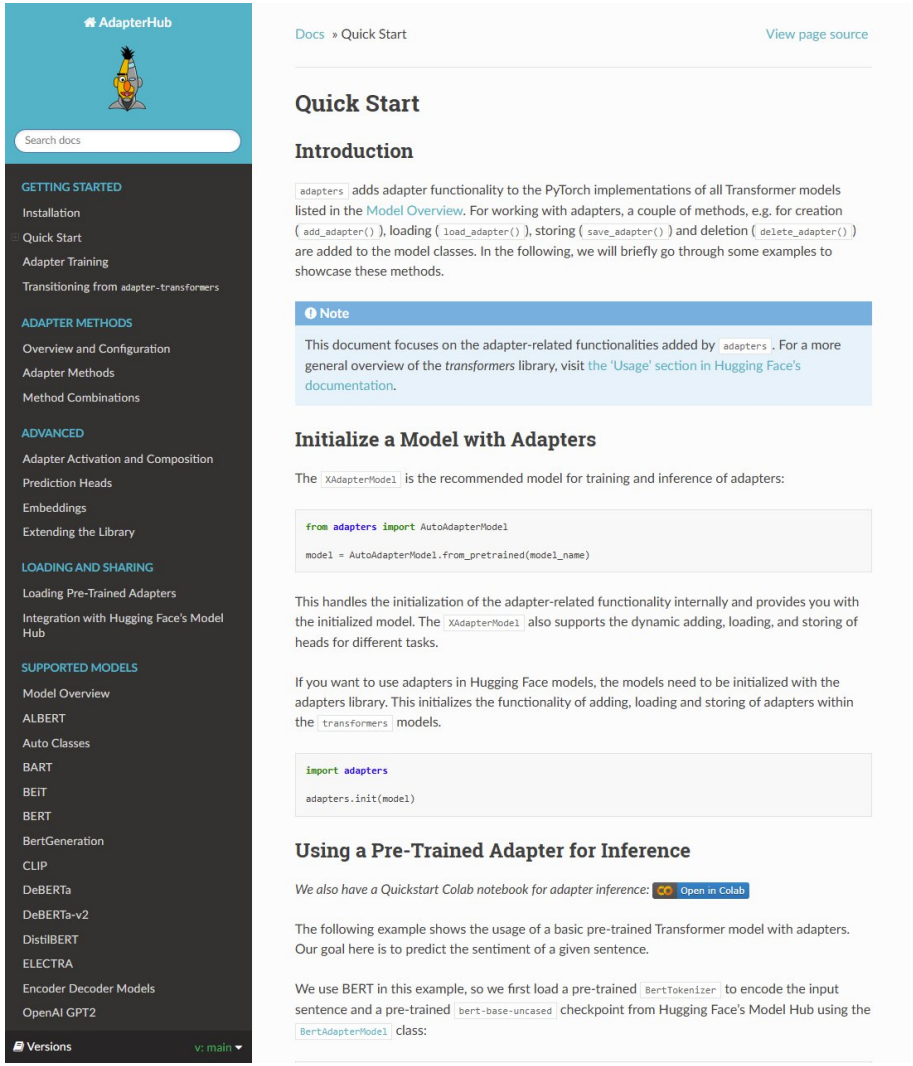
[19]

Adapter-based Knowledge Enhancement

- Different injection objectives, e.g. entity prediction
- The same relation “treats” might appear 20.000 times, but always with different entities
- Adapter should not memorize triplets, but learn the concept of triplets with same structure (e.g. “drug”- “treats” - “disease” for medical domain)

Practical Advice

- Find the best (PEFT) method for your project
  - ➢ RAG (or LoRA) might often work better
- AdapterHub platform for easy access and guides
- Try to re-use adapters before training them yourself



[20]

Outline

- Overview of Knowledge Enhancement
- Retrieval Augmented Generation (RAG)
- Adapter-based Knowledge Enhancement
  - Adapters
  - Applications
- **Summary**

Summary

- Knowledge enhancement encompasses a multitude of methodologies to inject dynamic and/or structured knowledge into an LLM pipeline

- RAG separates knowledge and language
- RAG utilizes Information Retrieval based on semantic search
- A RAG pipeline consists of several components (embedding and synthesis model, chunking and retrieval strategy etc.) which need to be selected with care

- Adapters can alter the model structure for efficient and robust injection
- KGs as a structured knowledge source, graph partitioning for better scalability
- Different methodologies can be combined, even RAG and adapters

Study Approach

Minimal

- Work with the slides

Standard

- Minimal approach + tutorial

In-Depth

- = standard approach + skim through references

See you next time!

Bibliography

(1) Hu, L., Liu, Z., Zhao, Z., Hou, L. (2023). A Survey of Knowledge Enhanced Pre-trained Language Models}

(2) Quan, Guangyao / Göçmen, Berke / Wiehe, Luca Mattes (2024-06-10). RAG for Source Text Summarization (TUM, Midterm Presentation, Supervisor: Miriam Anschütz)

(3) Ordax, Eduardo (2024-02-22). Introducing the ultimate RAG Cheatsheet! https://miro.com/app/board/uXjVNvklNmc=/?trk=public_post_comment-text

(4) Chase, Harrison (2023). LangChain: Chat with Your Data. https://www.deeplearning.ai/short-courses/langchain-chat-with-your-data/

(5) https://en.wikipedia.org/wiki/Universal_IR_Evaluation

(6) Theja, Ravi (2023-11-03). Boosting RAG: Picking the Best Embedding & Reranker models. https://blog.llamaindex.ai/boosting-rag-picking-the-best-embedding-reranker-models-42d079022e83

(7) Fuhr, Norbert (2018-02). Some Common Mistakes In IR Evaluation, And How They Can Be Avoided. https://dl.acm.org/doi/10.1145/3190580.3190586 https://sigir.org/wp-content/uploads/2018/01/p032.pdf

(8) Stöckl, Andreas / Ibrovic, Ernes (2024-02). Document Segmentation for Topic Modelling with Sentence Embeddings. https://ieeexplore.ieee.org/document/10467643

Bibliography

(8) Mishra, Anurag (2024-01-14). Five Levels of Chunking Strategies in RAG| Notes from Greg's Video. https://medium.com/@anuragmishra_27746/five-levels-of-chunking-strategies-in-rag-notes-from-gregs-video-7b735895694d

(9) https://python.langchain.com/docs/modules/data_connection/document_transformers/semantic-chunker

(10) Alroumi, Abdulmajeed (2023-09-19). Enhancing Semantic Search with LangChain, Vector Databases, and Llama2–70B-Chat. https://medium.com/@alroumi.abdulmajeed/enhancing-semantic-search-with-langchain-vector-databases-and-llama2-70b-chat-94d8dd56a450

(11) Glantz, Wenqi (2024-01-30). 12 RAG Pain Points and Proposed Solutions. https://towardsdatascience.com/12-rag-pain-points-and-proposed-solutions-43709939a28c

(12) Troynikov, Anton (2024-01). Advanced Retrieval for AI with Chroma. https://www.deeplearning.ai/short-courses/advanced-retrieval-for-ai/

(13) https://smith.langchain.com/hub/rlm/rag-prompt

(14) Liu, Jerry / Datta, Anupam. Building and Evaluating Advanced RAG Applications. https://www.deeplearning.ai/short-courses/building-evaluating-advanced-rag/

(15) Stöckl, Andreas (2024-06-02). New Chunking Method for RAG-Systems. https://medium.datadriveninvestor.com/new-chunking-method-for-rag-systems-2eb3523d0420

Bibliography

(16) Houlsby,N., Giurgiu, A., Jastrzebski, S., et al. (2019). Parameter-efficient transfer learning for nlp. In International Conference on Machine Learning

(17) Pfeiffer, J., Rücklé, A., Poth, C., et al. (2020). AdapterHub: A Framework for Adapting Transformers. https://arxiv.org/abs/2007.07779

(18) Hogan, A., Blomqvist, E., Cochez, M., et al. (2021). Knowledge Graphs. ACM Computing Surveys. https://doi.org/10.1145/3447772

(19) Meng, Z., Liu, F., Clark, et al. (2021). Mixture-of-Partitions: Infusing Large Biomedical Knowledge Graphs into BERT. Conference on Empirical Methods in Natural Language Processing. https://arxiv.org/abs/2109.04810

(20) Poth, C.A., Sterz, H., Paul, I., et al. (2023). Adapters: A Unified Library for Parameter-Efficient and Modular Transfer Learning. Conference on Empirical Methods in Natural Language Processing. https://arxiv.org/abs/2311.11077

(21) Vladika, J.; Fichtl, A. and Matthes, F. (2024). Diversifying Knowledge Enhancement of Biomedical Language Models Using Adapter Modules and Knowledge Graphs. In Proceedings of the 16th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART