Applied Machine Learning for Business Analytics

Lecture 5: Improving LLM’s performance - RAG

Lecturer: Zhao Rui

Agenda

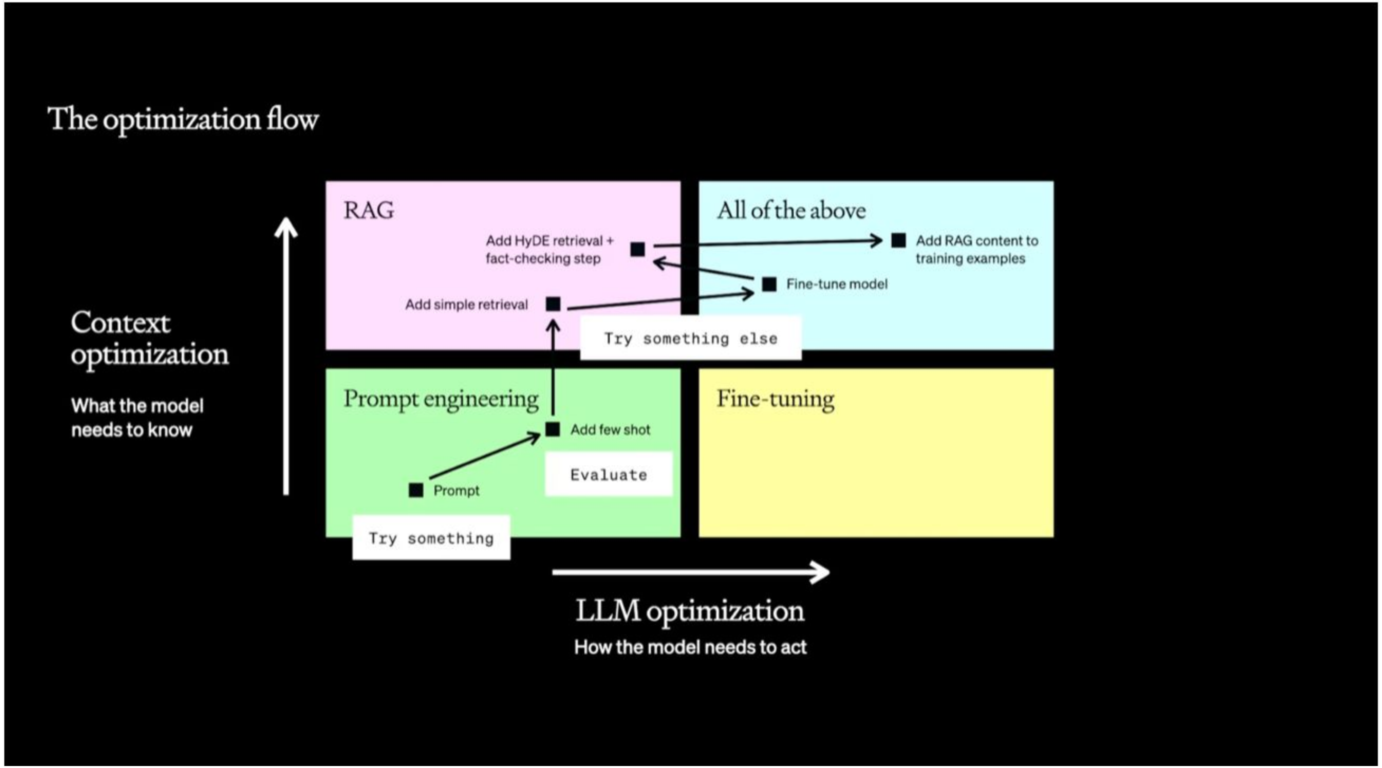
1. Improving LLMs' performance
2. Evaluating LLMs' performance
3. Prompt engineering
4. RAG

1. Improving LLM's performance

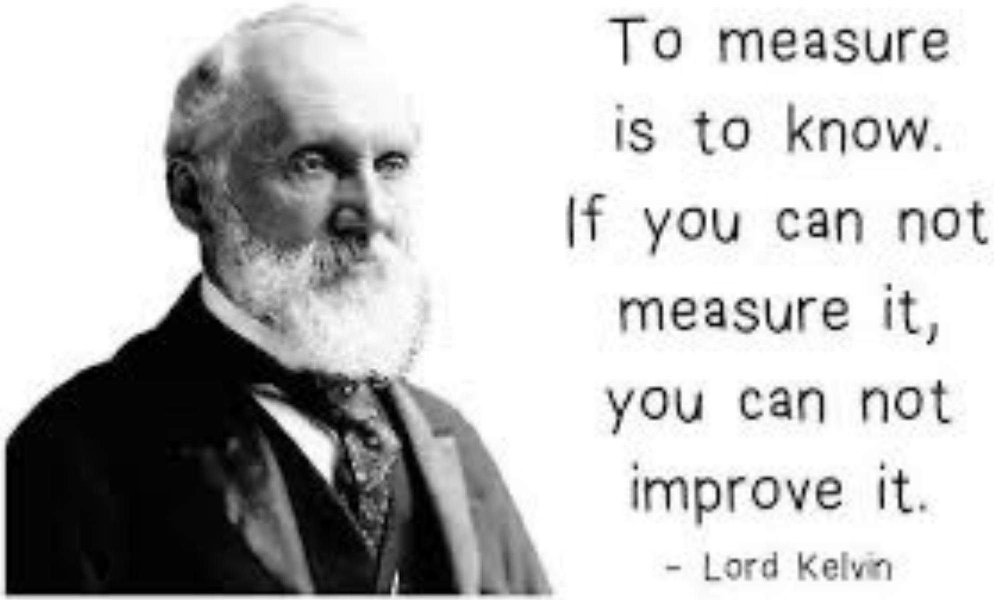
Optimization options

- **Prompt engineering**
  - Modify the prompt to guide the LLM's outputs
- **Retrieval-augmented generation**
  - Use the retriever to get external knowledge to enrich the context for LLM
- Fine tuning
  - Tune the LLM (its parameters) to better suit downstream applications

Iterative process



source


To measure
is to know.
If you can not
measure it,
you can not
improve it.
- Lord Kelvin

2. Evaluating LLM

Evaluating LLM

- Model-based evaluation
  - Use another LLM to evaluate the system's performances
- Rule-based evaluation
  - Implement heuristic rule to assess specific aspects of the LLM's output
- Accuracy metrics
  - If the task has clear labels, metrics can be used such as F1, precision and recall
- End-user feedback and A/B testing

“Accuracy” Metrics

- Evaluation on standardized benchmarks across all of NLP tasks
- Model Eval Benchmarks
  - GLUE: General language understanding evaluation benchmark provides a standardized set of diverse NLP tasks to evaluate the effectiveness of different language models
  - HellaSwag: evaluates how well an LLM can complete a sentence
  - TruthfulQA: measure truthfulness of model responses

End-user Feedback

- It is the process involving automatic evaluators triggered by new log entries derived from live production
- List of metrics as examples:
  - User engagement & utility metrics: number of views
  - User interaction: average number of LLM conversion per user
  - Quality of responses: average length of prompts and responses
  - User feedback and retention: user return rate
  - Performance metrics : latency
  - Cost metrics : infra cost (storage, networks, and computing resources, etc)
  - More metrics could be found here

3. Prompt engineering

Prompt engineering (Do it first)

- It refers to methods for how to communicate with LLM to steer its behavior for desired outcomes <u>without updating the model</u>
  - More empirical, less scientific
- Advantages
  - Testing and learning early
  - When paired with evaluation it provides your baseline and sets up further optimization
- Disadvantages
  - Introducing new information
  - Reliably replicating a complex style or method
  - Minimizing token usage

How to prepare a good prompt

- Start with:
  - Write clear instructions
  - Split complex tasks into simpler subtasks
  - Give GPTs time to “think”
  - Test changes systematically
- Extend to:
  - Provide reference text
    - Few-shot prompt -> RAG
  - Use external tools

Intuition behind Prompt Engineering

- LLMs understand better when you use familiar language and constructs
- LLMs can not know everything. If information is neither in training or in the prompt, they do not know it
- If you look at Prompt and you do not understand it, the prompt can not work

A “bad” prompt

You

You will be presented with a news article. Extract useful information in a structured format.

Unclear instructions

The article starts as below:
"Match-winner Gabriel Martinelli told Sky Sports: "I am not the man. I think everyone has a part in this victory. I scored the goal but it was the whole team and the staff and everyone. I am happy and the team is too.
Arsenal forward Gabriel Jesus added: "For sure it is a big win. Everyone knows how hard it is to play against them. But it is not impossible to beat them.

ChatGPT

Sure, I can extract useful information from the provided news article:

Event: Arsenal's victory in a match.

Date of the Event: Not mentioned in the provided text.

Quotes from the Players:

1. Gabriel Martinelli said, "I am not the man. I think everyone has a part in this victory. I scored the goal but it was the whole team and the staff and everyone. I am happy, and the team is too."
2. Gabriel Jesus (Arsenal forward) added, "For sure it is a big win. Everyone knows how hard it is to play against them. But it is not impossible to beat them."

Key Points:

- Gabriel Martinelli emphasized the collective effort of the team, staff, and everyone in the victory.
- Gabriel Jesus acknowledged the difficulty of playing against the opposing team but expressed confidence in their ability to win.

Random output

A “good” prompt

You

You will be presented with a news article about a scorer match. Your task is to identify any opinions expressed about the match, and their sentiment.

Clear instructions

Approach this task step-by-step, take your time and don not skip steps:

Give time to think

1. Read a line of news article
2. Determine whether an opinion is expressed in this paragraph. If not, continue to the next line.
3. If there is an opinion, extract a JSON with keys as follows:
   * opinion: allowable values are “positive”, “negative” or “neutral”
   * evidence: contain a list of strings evidencing the opinion.
   * speaker: the person who expressed the opinion.
4. Include as much evidence as possible and make sure the output is in JSON format

Break down complex tasks

The article starts as below:
"Match-winner Gabriel Martinelli told Sky Sports: "I am not the man. I think everyone has a part in this victory. I scored the goal but it was the whole team and the staff and everyone. I am happy and the team is too.
Arsenal forward Gabriel Jesus added: "For sure it is a big win. Everyone knows how hard it is to play against them. But it is not impossible to beat them.

A “good” prompt

ChatGPT

Let's go through the article step by step to identify any opinions expressed and their sentiment:

1. Read a line of the news article:
   "Match-winner Gabriel Martinelli told Sky Sports: 'I am not the man. I think everyone has a part in this victory. I scored the goal but it was the whole team and the staff and everyone. I am happy and the team is too.'"
2. Determine whether an opinion is expressed in this paragraph:
   Yes, an opinion is expressed in this paragraph. The opinion is about the match victory being a collective effort and not solely attributed to Gabriel Martinelli.
3. Extract a JSON with keys as follows:
   - opinion: "positive" (the sentiment is positive as Martinelli expresses happiness about the team's victory)
   - evidence: ["I am not the man.", "everyone has a part in this victory.", "I scored the goal but it was the whole team and the staff and everyone.", "I am happy and the team is too."]
   - speaker: "Gabriel Martinelli"

Next, let's continue to the next line of the article:

1. Read a line of the news article:
   "Arsenal forward Gabriel Jesus added: 'For sure it is a big win. Everyone knows how hard it is to play against them. But it is not impossible to beat them.'"
2. Determine whether an opinion is expressed in this paragraph:
   Yes, an opinion is expressed in this paragraph. Gabriel Jesus expresses the opinion that it is a significant win and that it is not impossible to beat the opposing team.
3. Extract a JSON with keys as follows:
   - opinion: "positive" (the sentiment is positive as Gabriel Jesus considers the win significant)
   - evidence: ["For sure it is a big win.", "Everyone knows how hard it is to play against them.", "But it is not impossible to beat them."]
   - speaker: "Gabriel Jesus"

So, here are the JSON representations of the opinions expressed in the article:

Opinion 1:

json Copy code

```json
{
  "opinion": "positive",
  "evidence": [
    "I am not the man.",
    "everyone has a part in this victory.",
    "I scored the goal but it was the whole team and the s
    "I am happy and the team is too."
  ],
  "speaker": "Gabriel Martinelli"
}
```

Opinion 2:

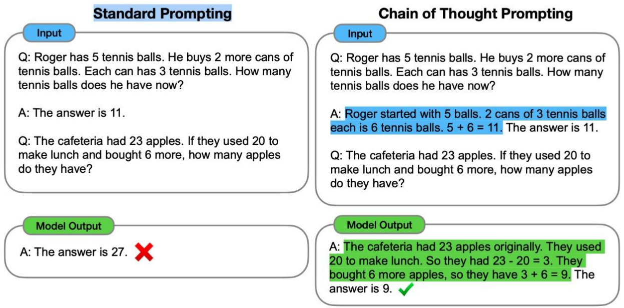
json Copy code

```json
{
  "opinion": "positive",
  "evidence": [
    "For sure it is a big win.",
    "Everyone knows how hard it is to play against them.",
    "But it is not impossible to beat them."
  ],
  "speaker": "Gabriel Jesus"
}
```

These JSONs represent the opinions and sentiments expressed by the mentioned individuals in the article.

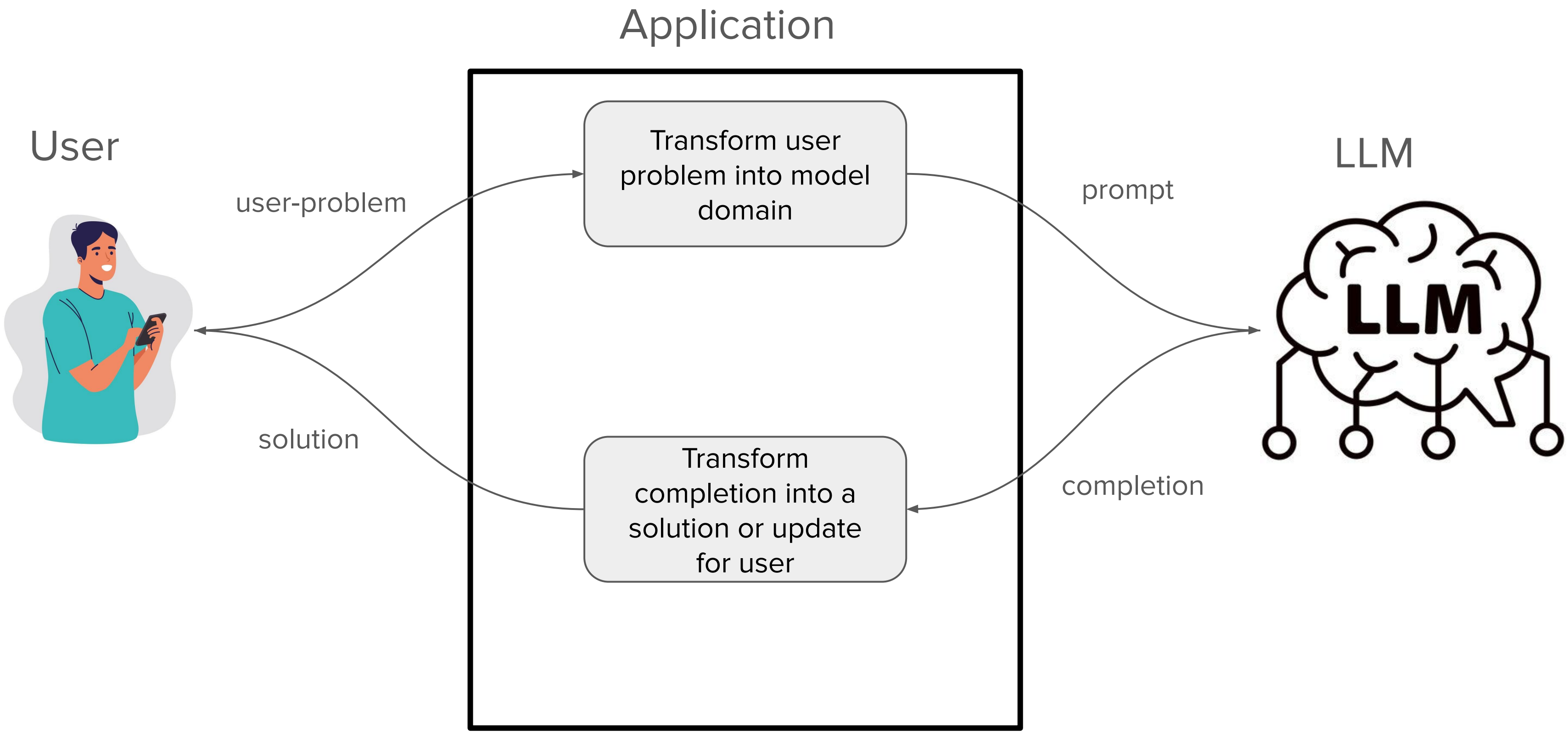
Prompting techniques

- Zero-shot prompting
  - No examples are given in prompt
- Few-shot prompting
  - A few shot examples of tasks are provided
- Chain of Thoughts prompting
  - Examples with the reasoning processes are improved.
- More prompting techniques could be found in this good survey and this openAI blog.



Source: https://arxiv.org/abs/2201.11903

Building LLM Applications



Creating the Prompt

- Create context
- Rank context
- Trim context
- Assembling context

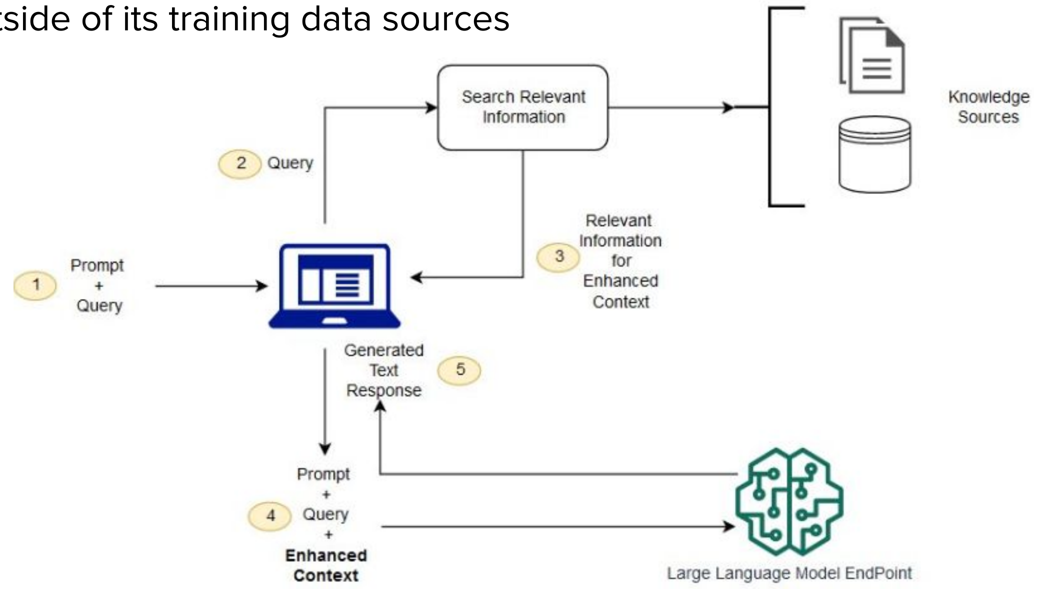
Creating the Prompt - Copilot Example

- Create context
  - Current document, open tabs, symbols, file path
- Rank context
  - Filter path >> current documents >> open tabs
- Trim context
  - Drop open tab snippets, truncate current documents
- Assembling context

4. RAG

What is RAG

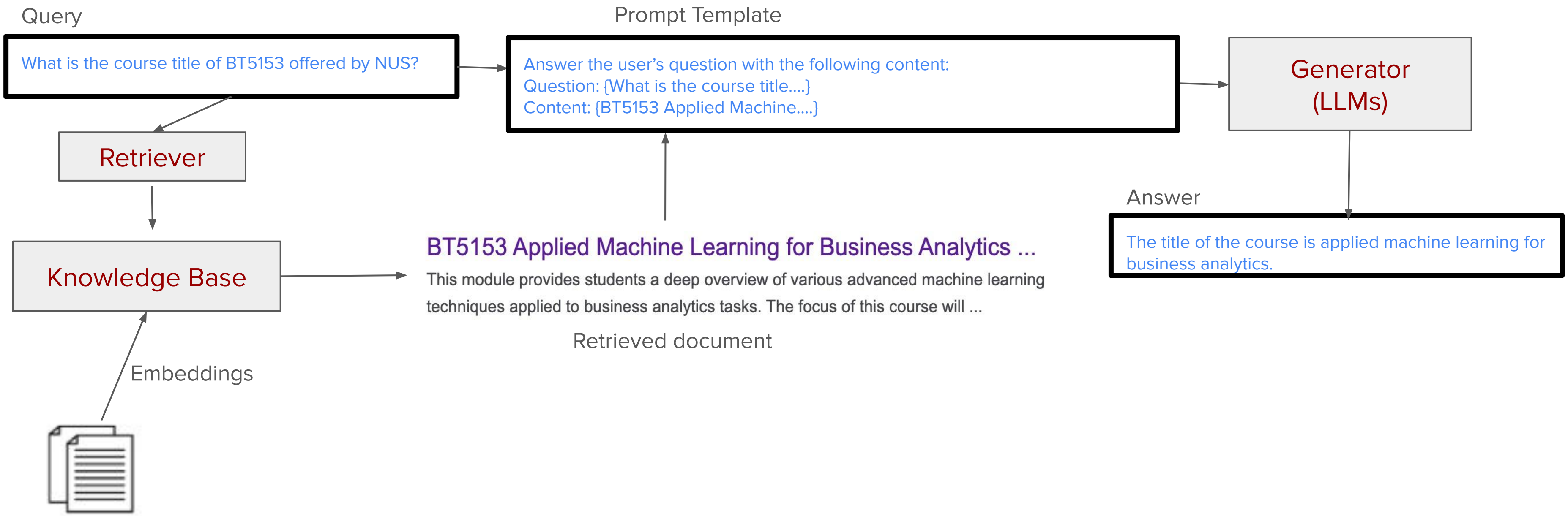
- Retrieval-Augmented Generation
  - A process to optimize the output of a LLM, so it references an authoritative knowledge base outside of its training data sources before responding.
- Benefits of RAG
  - Cost-effective implementation
  - Current information
  - Enhanced user trust
  - More develop control



source: https://aws.amazon.com/what-is/retrieval-augmented-generation/

One trivial example

- LLM is trained before the course website was built



It can be in webpage, pdf files, or even images/videos.

Old wine in new bottles

- RAG is not:
  - A new idea
  - A framework
- RAG is just combining Retrieval and Generation
  - Retrieval comes from Information Retrieval (IR):
    - The process of obtaining relevant information based on a user' information need expressed as a query/question
    - It is a research topic in computer science for decades
  - Generation is handled by LLMs

“Good” RAG

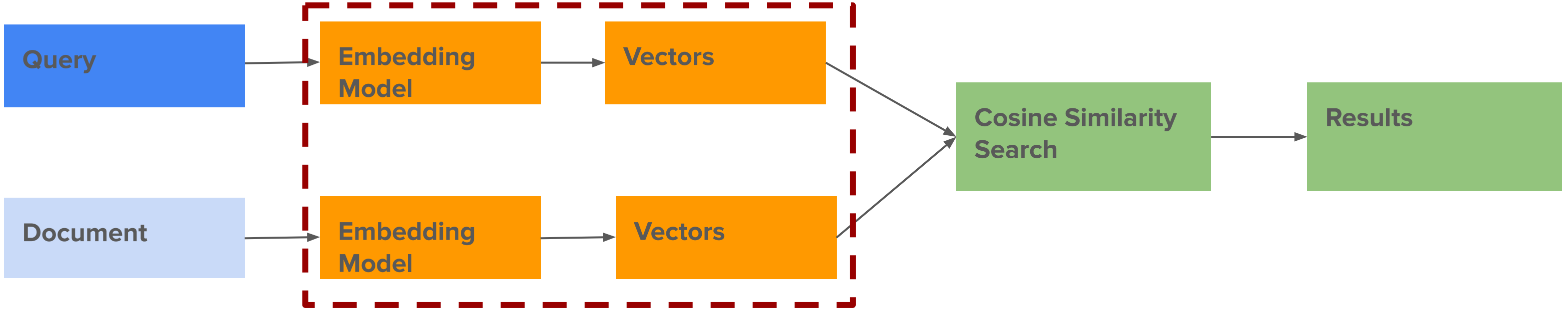
- **Good retrieval pipeline**
- Good generative model
- Good way of linking them up

Performance issues with Naive RAG

- Bad Retrieval
  - Low precision: not all retrieved chunks are relevant
    - Hallucination + Wrong replies
  - Low recall: not all relevant chunks are retrieved
    - Lacks enough context for LLM to synthesize the answer
  - Outdated information: the knowledge base is out of date
- Bad Response Generation (Native LLM Issues)
  - Hallucination: Model makes up an answer that isn’t in the context
  - Irrelevance: Model makes up an answer that does not answer the question
  - Toxicity/Bias: Model makes up an answer that is offensive

Start from Basic - Retrieval Pipeline

This is called a Bi-encoder approach



Start from Basic - Retrieval Pipeline

```
[ ] from sentence_transformers import SentenceTransformer
    model = SentenceTransformer('paraphrase-MiniLM-L6-v2')
```

Load Bi-encoder

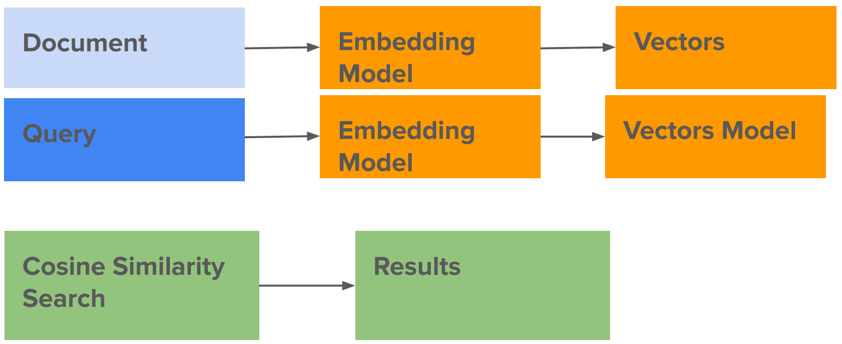
```
[ ] from wikipediaapi import Wikipedia
    wiki = Wikipedia('RAGBot/0.0', 'en')
    doc = wiki.page('Rock_Lee').text
    paragraphs = doc.split('\n\n')
```

```
[ ] docs_embed= model.encode(paragraphs,normalize_embeddings=True)
```

```
[ ] # Embedt h equery
    query = "How did Rock Lee open his chakra gates?"
    query_embed= model.encode(query,normalize_embeddings=True)
```

```
[ ] import numpy as np
    import torch

    similarities= np.dot(docs_embed,query_embed.T)
    # Convert the NumPy array to a PyTorch tensor
    similarities_tensor = torch.tensor(similarities)
    # Now apply topk to the tensor
    top_3_idx= similarities_tensor.topk(3).indices.tolist()
    most_similar_documents= [paragraphs[idx] for idx in top_3_idx]
```



Vector DB

- As the crucial part of RAG pipeline, search index supports the retrieval of content based on query
    - Indexing
        - Naive implementation: flat index
            - A brute force distance calculation between the query vector and all the chunks' vectors
    - Retrieving (nearest neighbours searching)
        - Efficient framework:
            - Open-source libraries: Faiss, Annoy
            - Managed solutions: Pinecone

Numpy is the vector DB

- The vector DB (or an index)
    - Allow Approximate search to avoid computing too many distance scores
    - It is only applicable for large scales retrieval
- Any modern CPU can search through hundreds of vectors in milliseconds

Dense Representation

- Vectorisation
  - The embedding model should be selected.
  - Search optimized models:
    - The leaderboard for massive text embedding benchmark

Rank	Model	Model Size (GB)	Embedding Dimensions	Sequence Length	Average (56 datasets)	Classification Average (12 datasets)	Clustering Average (11 datasets)	Pair Classification Average (3 datasets)	Reranking Average (4 datasets)	Ret Ave (15 dat
1	UAE-Large-V1	1.34	1024	512	64.64	75.58	46.73	87.25	59.88	54.
2	voyage-lite-01-instruct		1024	4096	64.49	74.79	47.4	86.57	59.74	55.
3	Cohere-embed-english-v3.0		1024	512	64.47	76.49	47.43	85.84	58.01	55

Dense Representation

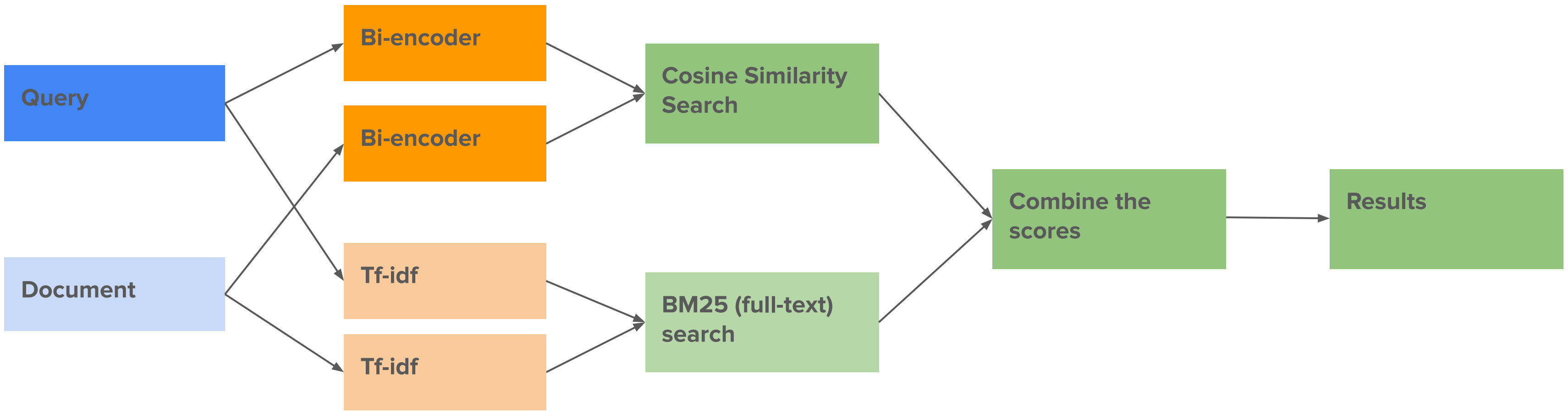
- Compressing information from tokens to a single vector is bound to lose information
- Embeddings learned is limited to the training data of the embeddings models
- Fixed Vocabulary
  - Llama 3 2024 => [‘ll’, ‘##ama’, ‘3’, ‘202’, ‘##4’]

Sparse Representation - BM25

- Keyword search, is built on an old technology: BM25
  - Based on TF-IDF (sparse)
  - Pretty powerful on longer documents and documents containing a lot of domain-specific jargon
  - It’s inference-time compute overhead is quite tiny
- Free lunch for any pipeline

Hybrid: dense + sparse

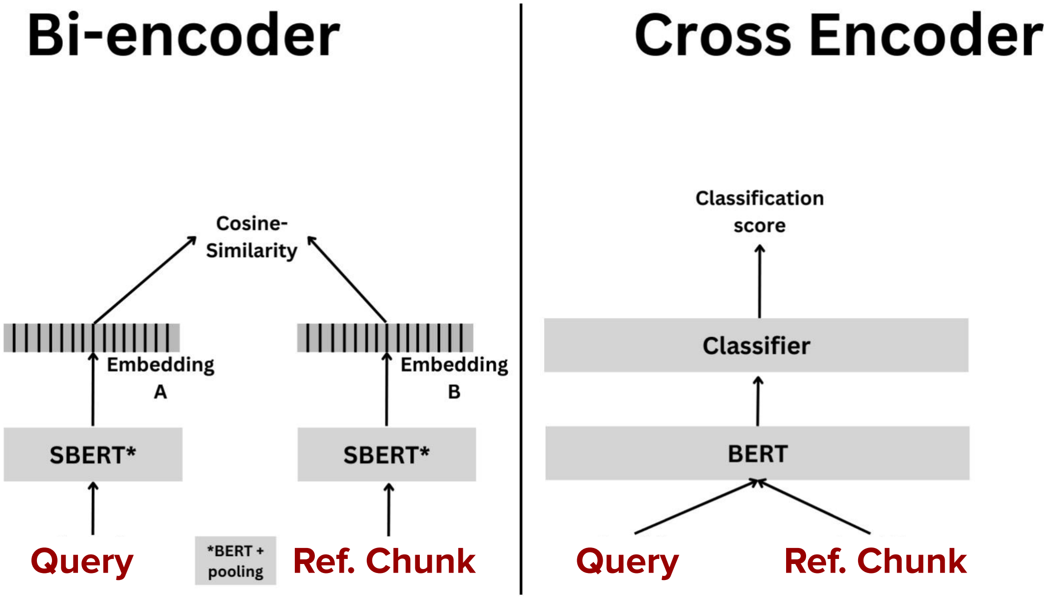
- Combine spread and dense representations



Bi-encoder

- Bi-encoders are (generally) used to create single-vector representations. They pre-compute document representations.
- Documents and query representations are computed entirely separately, they are not aware of each other
- Thus, document vectors can be pre-computed and at inference, encode your query and search for similar vectors
  - Very computationally efficient
  - It comes with retrieval performance tradeoffs.

Bi-encoder vs Cross Encoder



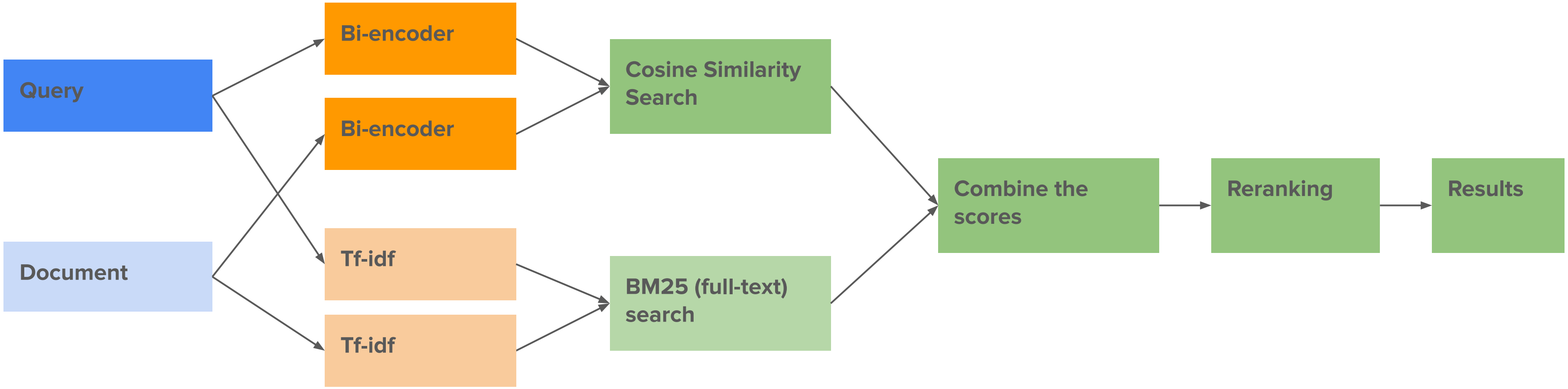
Source: https://osanseviero.github.io/hackerllama/blog/posts/sentence_embeddings2/

Given 10 queries and 1000 chunks in the reference:

- For Bi-encoder, how many times that the transformer would be called?
- For Cross-Encoder, how many times that the transformer would be called?

Reranker: The power of Cross-Encoders

- Leverage a powerful but computationally expensive model to score only a subset of your documents, previously retrieved by a more efficient model.
- One approach: cross-encoder
    - It is effectively a binary classifier
        - The input is a pair of query and document
        - The output is the probability of being the positive class as the similarity score



Metadata filter

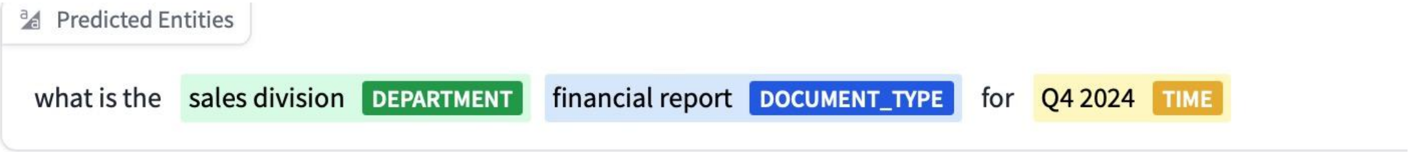
- Documents do not exist in a vacuum. There is a lot of metadata around them, some of which can be very informative
- Metadata is the context you can add with each text chunk
    - Examples
        - Page number
        - Document title
        - Summary of adjunct chunks
        - Hypothesis questions (reverse HyDE)
            - Ask the LLM to generate a question for each chunk
    - Benefits
        - Can help retrieval
        - Can augment response quality
        - Integrates with Vector DB metadata filters

Metadata filter

- Query: what is the sales division financial report for Q4 2024?
- Raw top-k retrieval via embeddings/keyword would have low precision
  - The model should accurately capture all of the key meaning: financial report, sales division, Q4 and 2024.
  - And the k also can not be set too high. Otherwise irrelevant financial reports would be passed to LLM

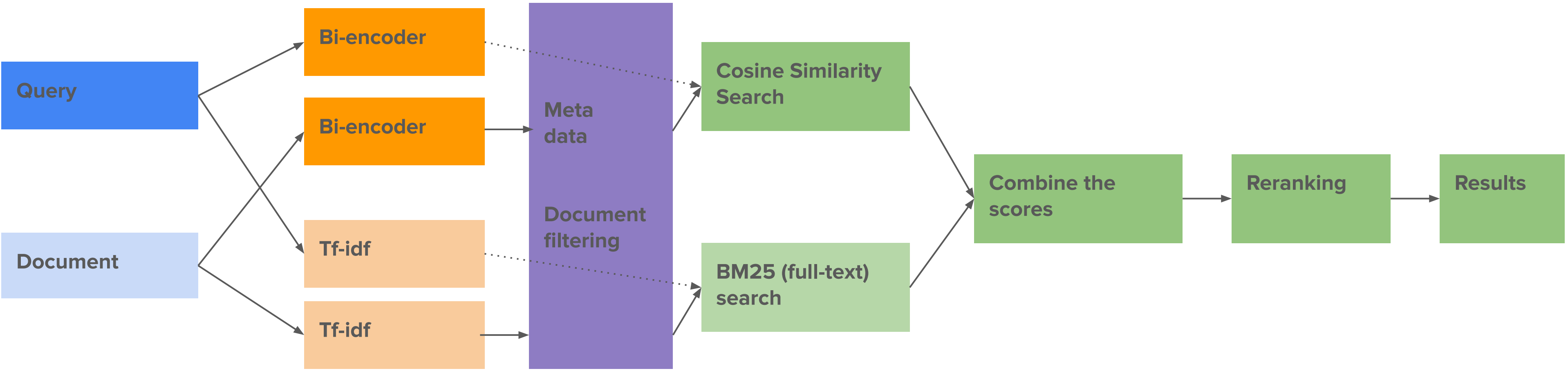
Metadata filter

- Apply entity detection models on query such as GliNER
- Ensure the business/query-relevant information is stored alongside their associated documents
- Than, the extracted entities could be used to pre-filter the documents, ensuring we only perform search on documents whose attributes are related to the query.



source: https://huggingface.co/spaces/urchade/gliner_mediumv2.1

The Final MVP



The Final MVP

```
[ ] from wikipediaapi import Wikipedia
    wiki = Wikipedia('RAGBot/0.0', 'en')
    docs = [{'text': x, 'category': "anime"} for x in wiki.page("Rock_Lee").text.split('\n\n')]
    docs += [{'text': x, 'category': "sports"} for x in wiki.page("Rock_Lee_(basketball)").text.split('\n\n')]
```

```
[ ] import lancedb
    from lancedb.pydantic import LanceModel, Vector
    from lancedb.embeddings import get_registry
    from lancedb.rerankers import CrossEncoderReranker
```

```
[ ] model = get_registry().get("sentence-transformers").create(name="paraphrase-MiniLM-L6-v2")
```

Bi-encoder

```
[ ] class Document(LanceModel):
      text: str = model.SourceField()
      vector: Vector(384) = model.VectorField()
      category: str
    db = lancedb.connect('.my_db')
    tbl = db.create_table("my_docv1", schema=Document)
```

Meta data

```
[ ] tbl.add(docs)
    tbl.create_fts_index("text")
```

BM25 (full-text) search

```
[ ] reranker = CrossEncoderReranker()
    query = "How did Rock Lee open his chakra gates?"

    # Apply filter within the search query
    results = (tbl.search(query, query_type='hybrid').where("category='anime'").limit(3).rerank(reranker=reranker))
```

Document Filtering

Demo

```
all_data = results.to_list()[:3]
[ele['text'] for ele in all_data[:3]] # get top3 results
```

```
['Appearances\nIn Naruto\nRock Lee is a ninja from Konohagakure part of Team Guy, a four-man cell of ninja led by Might Guy.
Inspired by Lee\'s determination to become stronger despite his inability to perform basic ninja techniques, Guy takes a
personal interest in him, deciding to help him achieve his dream of becoming a powerful ninja by using only taijutsu that is
primary hand-to-hand combat. This relationship with Guy causes Lee to acquire many of Guy\'s traits. Lee believes he can
surpass the natural talents of others through hard work and passion; throughout the series, he attempts to surpass Neji Hyuga,
who is labeled a "genius". Lee first appears in the series as a participant in the Chunin Exams, twice a year exams for ninja
who wish to increase their rank. During the Chunin Exams, Lee battles Gaara, a ninja from the village of Sunagakure. In the
fight, Lee opens the five of the eight chakra gates, limits on the body\'s ability to use chakra, using a forbidden technique
known as the Hidden Lotus, increasing his natural abilities at the cost of his health. Despite his effort, Gaara cripples Lee
by crushing his left arm and leg, injuring Lee to the point that he must abandon being a ninja.\nWhen Tsunade, a Konohagakure
medical ninja, returns to lead the village as the Fifth Hokage, she offers to operate on him. Despite the procedure\'s fifty
percent chance of failure, Guy encourages Lee to have the operation. Ultimately, Lee undergoes the surgery, which succeeds in
healing his arm and leg. Immediately after the operation, Lee follows a team of ninja led by Shikamaru Nara who attempt to stop
Sasuke Uchiha from defecting from Konohagakure to the village of Otogakure. Lee battles the Otogakure ninja Kimimaro through
using the Potion Punch (酔拳, Suiken, literally "Drunken Fist", English TV: "Loopy Fist") fighting style in which he becomes
inebriated with unpredictable attacks. When Kimimaro is on the verge of defeating Lee, Gaara intervenes, continuing the
battle.\nIn Part II, Lee obtains the rank of Chunin, and is dispatched with his team to help save Gaara following his abduction
by the criminal organization Akatsuki. During the events of the Fourth Shinobi War Lee is assigned to the Third Division, Lee
helps in fighting the Kabuto Yakushi\'s reanimated army and later aids Naruto Uzumaki in the fight against Obito Uchiha and
Madara Uchiha. Years after the war, Lee marries an unknown woman and has a son named Metal Lee. In the epilogue, Lee is last
seen many years later, training with his son. In Boruto: Naruto the Movie, Lee hosts the third stage of the Chunin Exam.',
 'Reception\nLee has ranked highly in the Shōnen Jump popularity polls for the series, initially continuously placing in the
top ten and reaching fifth place once. In later polls, Lee lost his top ten status. Also, several pieces of merchandise based
on Lee have also been released, including action figures of his Part I and Part II appearances, plush dolls, and keychains. My
Hero Academia author Kōhei Horikoshi said that Rock Lee\'s inhability to use ninjutsu and instead use hand-to-hand combat made
him highly stand out in the manga. He also praised how heroic is Rock Lee\'s portrayal in the Chunnin Exams when facing
Orochimaru\'s underlings.\nSeveral publications for manga, anime, video games, and other media have provided commentary on
Lee\'s character. IGN\'s A.E. Sparrow called Lee one of his favorite characters in the series and compared his personality to
that of Bruce Lee and Noel Gallagher. Fellow editor Ramsey Isler ranked him as the eight best character on the series and said
he "was the true underdog of the series." Isler added, "Perhaps a little too intense, but always fiercely devoted to his cause,
Rock Lee added all sorts of flavor to the series." However, Rock Lee\'s profile on IGN describes him as "kind of stiff" because
of his very polite demeanor. Active Anime celebrated Lee\'s introduction in the series as a comedic relief to the growing
tension of the story at that point. Anime Insider listed him in their top five list for "pure-hearted heroes" from anime and
manga publications, ranking at number five. Insider praised him for "never [giving] up, even in the face of people with actual
ninja powers."\nAnime News Network referred to Lee as the "star of [the Chunin Exam arc]", and claimed that he "almost single-
handedly rescues this arc from being tossed into the \'entertaining but disposable\' bin". His fight against Gaara in the exams
was listed as second best one in anime by AnimeCentral. Anime News Network also called Lee the "goofiest looking character" in
the series and praised Kishimoto\'s "ninja-punk visual sensibilities" that allowed him to make Lee "damn cool when the action
starts". In the NEO Awards 2007 from Neo, Rock Lee won in the category "Best Anime Character". He was also listed as one of the
three "Honorable Mentions" from Naruto by Wizard Entertainment\'s Danica Davidson with comments from the article being focused
on Lee\'s determination. When the Naruto manga ended, writer Yūto Kubota expressed that Lee was his favorite character.',
 'Rock Lee (Japanese: ロック・リー, Hepburn: Rokku Rī) is a fictional character in the anime and manga series Naruto and Naruto
Shippuden created by Masashi Kishimoto. At first Masashi designed Lee to symbolize human strength. In the anime and manga, Lee
is a ninja affiliated with the village of Konohagakure, and is a member of Team Guy, which consists of himself, Neji Hyuga,
Tenten, and Might Guy—the team\'s leader. Unable to use most ninja techniques, Lee dedicates himself  to using solely taijutsu,
ninja techniques similar to martial arts. Lee dreams of becoming a "splendid ninja" despite his inabilities. Lee has appeared
in several pieces of Naruto media, including the third and fourth featured films in the series, the third original video
animation, and multiple video games.\nNumerous anime and manga publications have commented on Lee\'s character. IGN compared
Lee to Bruce Lee and Noel Gallagher, and Anime News Network called Lee the "goofiest looking character" in the series. Among
the Naruto reader base, Lee has been popular, placing high in several popularity polls. Numerous pieces of merchandise have
been released in Lee\'s likeness, including figurines and plush dolls.']
```

Code:
https://colab.research.google.com/drive/1L0VWAgIiywqjNE1TFmdfusAV86yDuR94?usp=sharing

How to Improve RAG

- Data
  - Store additional information beyond raw text chunks
- Embeddings
  - Optimize embedding representations
- Retrieval
  - Advanced retrieval instead of top-k embedding lookup
- Synthesis
  - LLMs can play a big role in the process

How do we properly evaluate a RAG

- After those changes, we need to evaluate the impacts
  - Evaluate various components
    - Is the retrieval good?
    - Is the LLM good?
  - Evaluate end-to-end

“Advanced” Topics for RAG

- Chunking Size
  - Divide large documents into small chunks
- Query Transformation
- [Parametric Retrieval Augmented Generation]

See more in the appendix

Parametric Retrieval Augmented Generation

Weihang Su
swh22@mails.tsinghua.edu.cn
DCST, Tsinghua University
Beijing 100084, China

Yichen Tang*
DCST, Tsinghua University
Beijing 100084, China

Qingyao Ai†
aiqy@tsinghua.edu.cn
DCST, Tsinghua University
Beijing 100084, China

Junxi Yan
DCST, Tsinghua University
Beijing 100084, China

Changyue Wang
DCST, Tsinghua University
Beijing 100084, China

Hongning Wang
DCST, Tsinghua University
Beijing 100084, China

Ziyi Ye
DCST, Tsinghua University
Beijing 100084, China

Yujia Zhou
DCST, Tsinghua University
Beijing 100084, China

Yiqun Liu
DCST, Tsinghua University
Beijing 100084, China

5. Use LangChain to Build RAG

Build RAG quickly

```
print("Creating RAG chain...")
# Initialize LLM (OpenAI)
llm = ChatOpenAI(temperature=0)
# Create prompt template
template = """Use the following pieces of context to answer the question. If you don't know the answer, just say that you don't know.

Context: {context}
Question: {question}

Answer:"""
prompt = PromptTemplate.from_template(template)
# Format documents function
def format_docs(docs):
    return "\n\n".join(
        f"{doc.page_content}\n(Source: {doc.metadata['source']}, Page: {doc.metadata['page']})"
        for doc in docs
    )
# Create the RAG chain
# RunnablePassthrough() is used to pass the question through the chain unchanged. It means that the question firstly pass through the retriever
# then format the context. And than the question would be passed with the retrieved context to the prompt
rag_chain = (
    {"context": retriever | format_docs, "question": RunnablePassthrough()}
    | prompt
    | llm
    | StrOutputParser()
)
# StrOutputParser() is used to parse the output of the llm to a string.
# Here, it is the ChatModel. So it extracts the .content attribute of the message, ensuring that the final output is in string format.
```

Test the flow

- Given the question: How is the dataset prepared to detect the AI-generated essays
- Knowledge base: Project paper from last year
- Two LLMs are checked:
  - GPT-40
  - DeepSeek-R1-Distill-Qwen-1.5B

Detection of AI-Generated Text for Essay Competitions

Abstract

The widespread adoption of large language models like GPT (Generative Pre-trained Transformer) has facilitated the generation of text that closely mimics human writing, raising concerns about plagiarism and academic integrity. This project aims to develop a specialized system capable of detecting AI-generated text, particularly in the context of essay submissions to academic competitions. By leveraging advanced natural language processing techniques and pre-trained machine learning models, including XLNet, GPT-2.0, and DistilBERT, we conduct a comprehensive exploration encompassing two phases of fine-tuning. The initial phrase optimizes each model's performance with the most appropriate datasets and features whereas the subsequent phase refines the models within the specific domain of academic essays. Through meticulous evaluation and comparison, we select the most suitable model for integration into our detection system. By combining cutting-edge techniques with rigorous experimentation, we seek to enhance the credibility and fairness of academic assessments while promoting a culture of originality and integrity in scholarly work.

task by employing traditional classification algorithms, such as TF-IDF combined with a logistic regression model, and other traditional algorithms like support vector machine and random forest models. This approach was advantageous for its interpretability. Subsequently, the researchers explored deep learning approaches, specifically leveraging language models like pretrained BERT models. These models were fine-tuned using a meticulously curated dataset consisting of generated-text pairs. Notably, Rodriguez et al. demonstrated that even in situations with limited available resources, robust performance could be achieved by utilizing a few hundred labeled authentic and synthetic texts specific to the domain, without the need for complete information about the LLM text generation pipeline. Therefore, our proposed project adopted a similar methodology, aligning with the flow of Rodriguez et al.'s research.

Moreover, while studies have explored LLM-generated text detection, there is a need for specialized detectors tailored to the essay competition domain, incorporating domain-specific features and writing patterns. Efforts will

Build RAG quickly

```
Answer:  The dataset is prepared by removing duplicates, extracting relevant features, converting to lowercase, removing punctuation, and eliminating stop words.
</think>

The dataset preparation process involves several key steps:

1. **Duplicate Removal**: To ensure data integrity and eliminate redundancy, duplicate entries are removed.
2. **Feature Extraction**: Four main features are extracted:
   - **Sentence Length Variation**: Standard deviation of sentence complexity.
   - **Mean Word Length**: Indicates the complexity of word usage.
   - **Stop Words Removal**: Common words like "the" and "is" are removed to enhance clarity.
   - **Adjective Usage**: Percentage of adjectives reflects descriptive depth.
3. **Lowercasing**: All text is converted to lowercase for uniformity.
4. **Punctuation Removal**: Punctuation marks are stripped to focus solely on text content.

These steps collectively prepare the dataset for effective AI text detection.
```

```
The dataset is prepared by categorizing it into general text and real competition essay submissions, with each co
ntributing uniquely to the study. The general text category includes a dataset from Kaggle and GPT(Curie)-generat
ed essays on car-free cities, while the Wiki Introduction dataset consists of pairs of Wikipedia introductions an
d their AI-generated versions. The competition essay category includes past winning essays from global competitio
ns matched with GPT-3.5/4.0 generated content. These datasets are utilized for preliminary finetuning of candidat
e models to choose the best combination of modeling features for subsequent finetuning. Additionally, features su
ch as the percentage of stop words and adjectives per text are incorporated to capture meaningful linguistic char
acteristics and enhance the model's capability to detect AI-generated text.
```

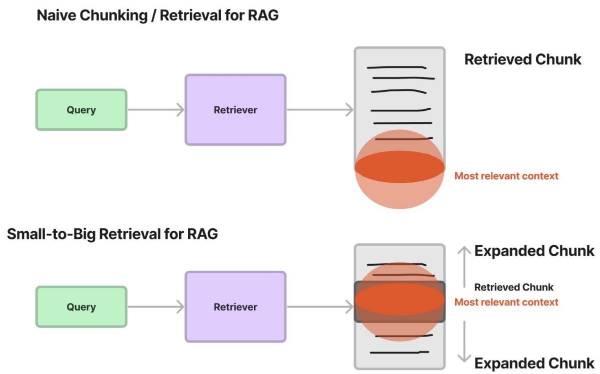
Appendix

Chunk sizes

- Chunking
  - Split texts into chunks of some size without losing their meaning
  - The size of the chunk is also a hyper-parameter to be tuned
    - Embeddings models’ capacity
    - Conflicts between enough context for LLM to reason (wide) and specific enough text embedding (narrow) in order to efficiently execute search upon
  - A few chunking techniques:
    - Fixed-size chunking
    - Content-aware chunking
      - Sentence splitting: sentence-level chunking
      - Specialized chunking: preserve the original structure of the content if it is in markdown, latex or other formats
    - A good survey here.

Small-to-Big

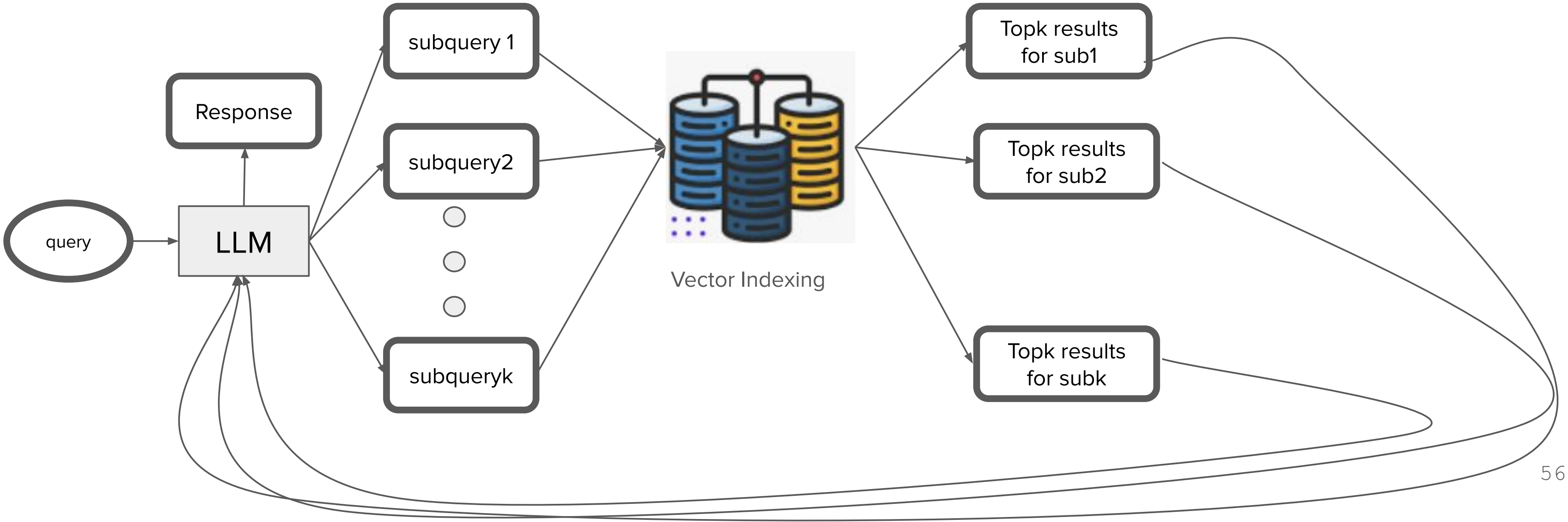
- Intuition:
  - Embedding a big text chunk is not optimal
  - Defining a chunking boundaries is completely arbitrary and independent of the relevant context
- Solution:
  - Embed text at the sentence-level
  - Expand that window for the context in the query to LLM



Source: https://twitter.com/jerryjliu0/status/1732503009891127676/photo/1

Query transformation

Query transformation is referred to those techniques using LLM as a reasoning engine to modify user inputs in order to improve retrieval quality



Query transformation

- Core idea behind transformation is:
  - Decompose the complex query into several sub queries
- Open-source implementation
  - Multi query retriever in Langchain
  - Sub question query engine in Llamaindex

Run queries

```
response = query_engine.query(
    "How was Paul Grahams life different before, during, and after YC?"
)
```

```
Generated 3 sub questions.
[pg_essay] Q: What did Paul Graham do before YC?
[pg_essay] Q: What did Paul Graham do during YC?
[pg_essay] Q: What did Paul Graham do after YC?
```

Chat Template

Chat Template

- What if you found this scrap of paper on the ground?
- What do you think the rest of the paper would say?

My cable is out! And I'm going to miss the Superbowl!

Chat Template

```
# IT Support Assistant
The following is a transcript between an
award winning IT support rep and a
customer

## Customer
My cable is out! And I'm going to miss the
Superbowl!

## Support Assistant:
```

Chat Markup Language (ChatML)

- A tool that helps manage multi-turn conversion and can also be used for non-chat scenarios.
  - Make model understand where each piece of text comes from
  - The conversion is segregated into three layers:
    - System
    - Assistant
    - User
    - Tool (for some LLMs)
  - <im_start> and <im_end> are special tokens to set boundaries of information.

Copy

```
<|im_start|>system
Provide some context and/or instructions to the model.
<|im_end|>
<|im_start|>user
The user’s message goes here
<|im_end|>
<|im_start|>assistant
```

Chat Markup Language (ChatML)

- System message (optional but suggested for better result)
  - It is included at the beginning of the prompt
  - It can have the following info:
    - A brief description of the assistant
    - Personality traits of the assistant
    - Instructions of rules you would like the assistant to follow
    - Data or information needed for the model
- Message
  - It includes a series of message between the user and the assistant
  - Each message should begin with the <im_start> token followed by the role and end with the <im_end> token.
  - To trigger a response from the model, the prompt should end with <im_start>assistant

More info: https://learn.microsoft.com/en-us/azure/ai-services/openai/how-to/chat-markup-language

Apply Chat Template

- Really easy for users to build assistant
  - Rather than continuing a single string of text, the model will continue a conversion
  - System messages make controlling behavior easy
- Safety is baked in:
  - Assistant will (almost) never respond with harmful info
  - Prompt injection is (almost) possible

Apply Chat Template

API

```
messages =
[{
  "role": "system"
  "content": "You are
an award winning
support staff
representative that
helps customers."
 },

 {"role": "user",
  "content":"My cable
is out! And I'm going
to miss the
Superbowl!"
 }
]
```

`tokenizer.apply_chat_template`

Chat Markup Language (ChatML)

<|im_start|> system
You are an award winning IT support rep. Help the user with their request.<|im_stop|>

<|im_start|> user
My cable is out! And I'm going to miss the Superbowl!<|im_stop|>

<|im_start|> assistant

More details could be found here.