Presented by John Snow LABS

NLP SUMMIT

Building Healthcare NLP & LLM Applications

Veysel Kocaman

Head of Data Science

John Snow Labs

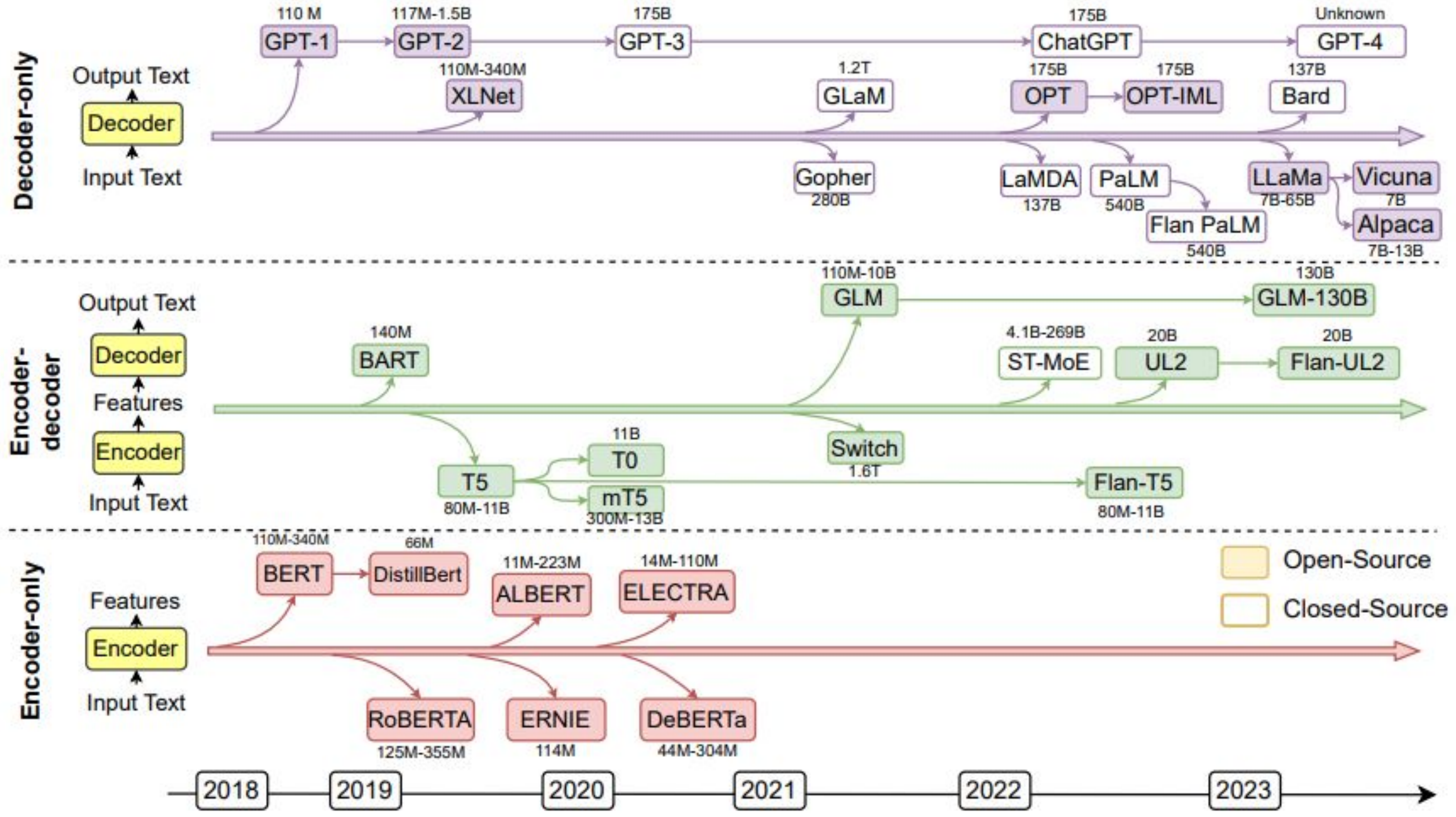
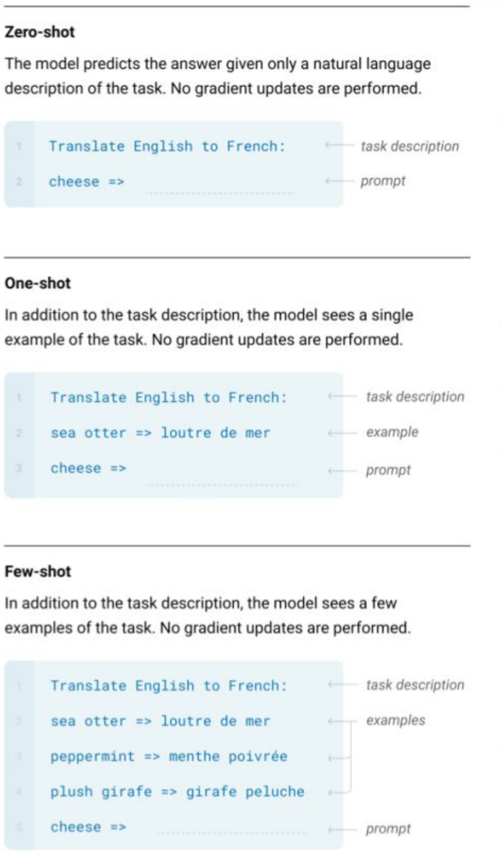


Fig. 2. Representative large language models (LLMs) in recent years. Open-source models are represented by solid squares, while closed source models are represented by hollow squares.

https://arxiv.org/pdf/2306.08302.pdf

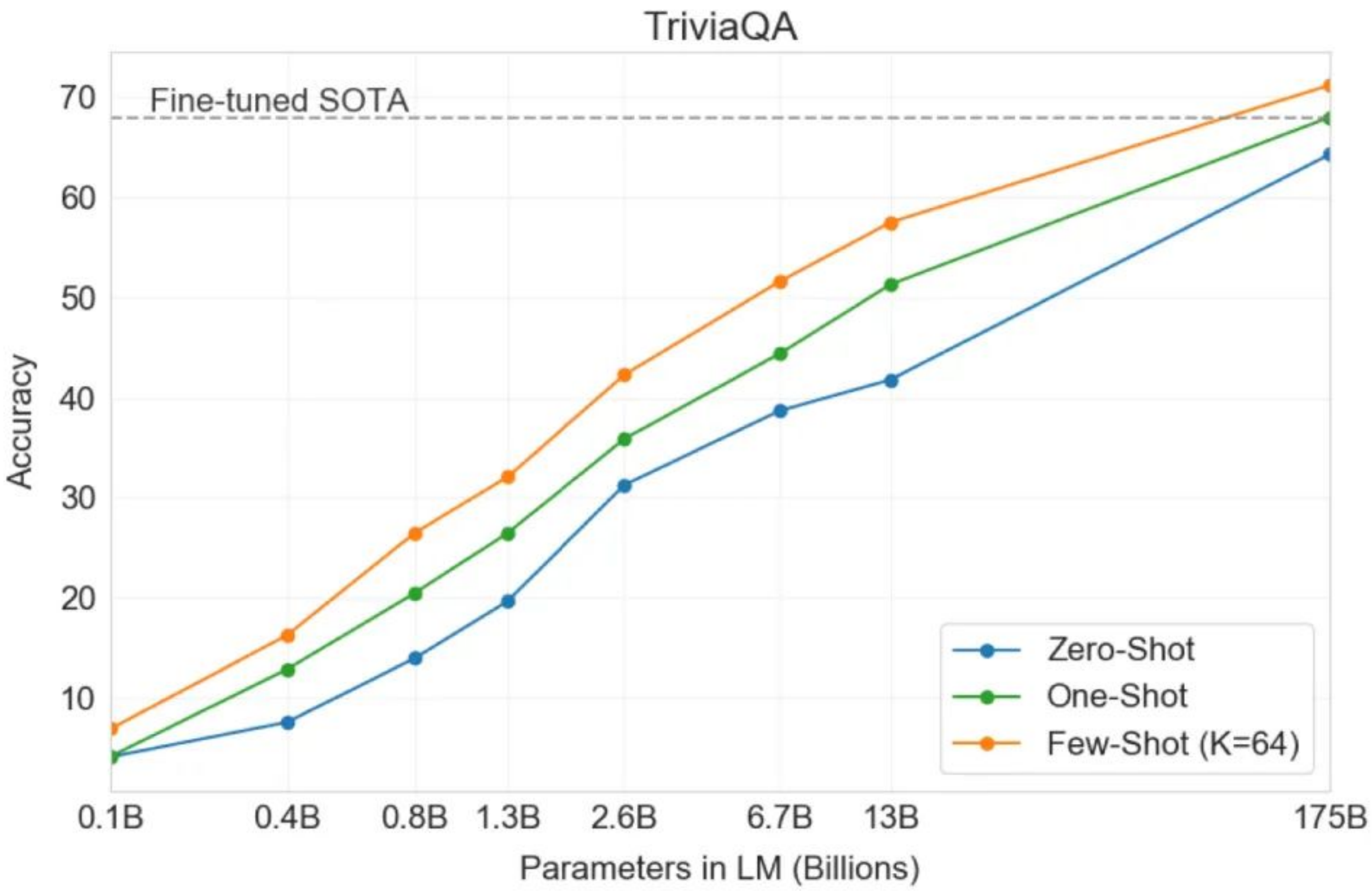
Capabilities of GPT-4 on Medical Challenge Problems

Dataset	Component	**GPT-4 (5 shot)**	GPT-4 (zero shot)	GPT-3.5 (5 shot)
MedQA	Mainland China	**75.31**	71.07	44.89
	Taiwan	**84.57**	82.17	53.72
	United States (5-option)	**78.63**	74.71	47.05
	United States (4-option)	**81.38**	78.87	53.57
PubMedQA	Reasoning Required	74.40	75.20	60.20
MedMCQA	Dev	**72.36**	69.52	51.02
MMLU	Clinical Knowledge	**86.42**	86.04	68.68
	Medical Genetics	**92.00**	91.00	68.00
	Anatomy	**80.00**	**80.00**	60.74
	Professional Medicine	**93.75**	93.01	69.85
	College Biology	93.75	**95.14**	72.92
	College Medicine	76.30	**76.88**	63.58

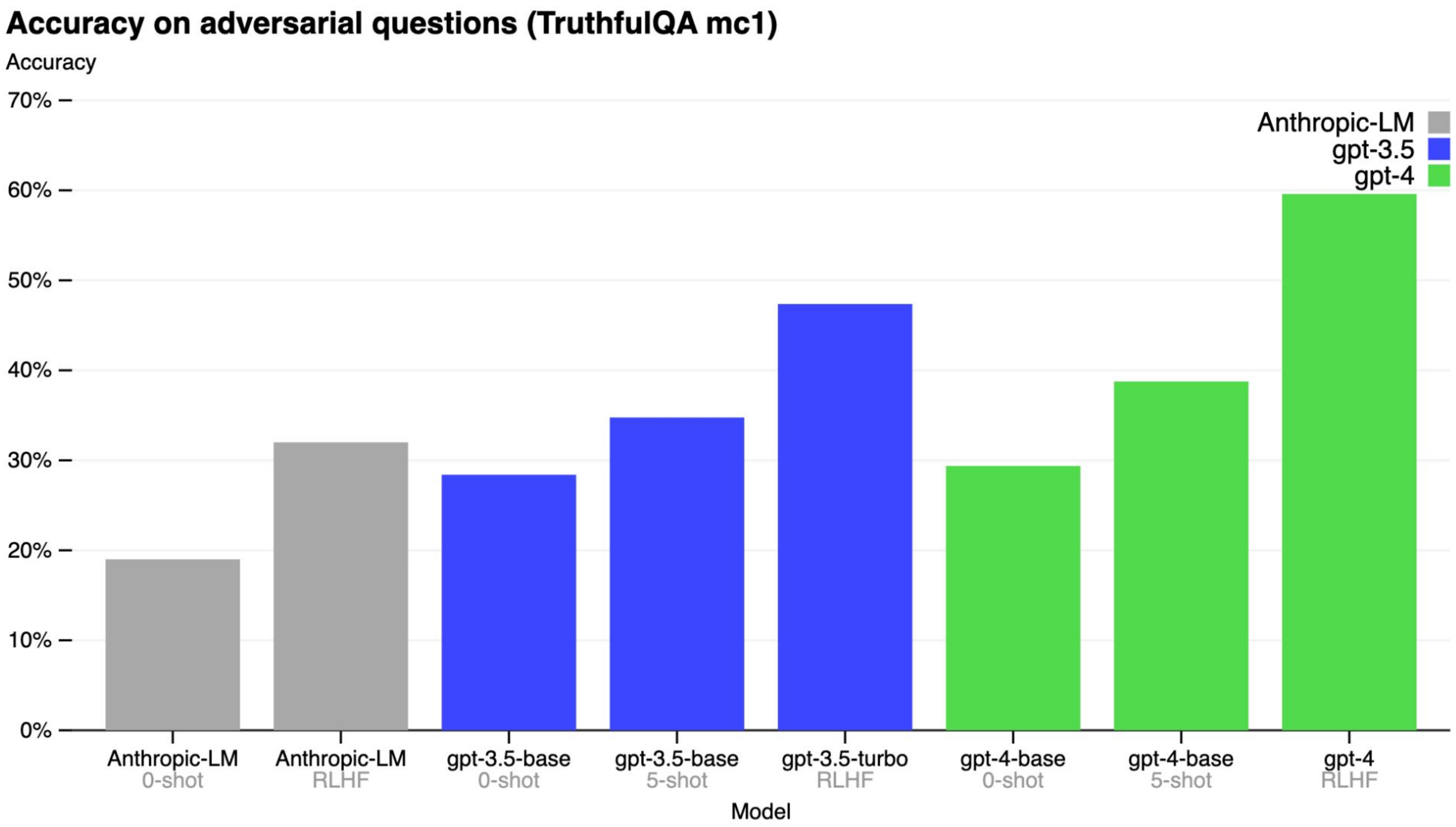


Nori et al. "Capabilities of GPT-4 on Medical Challenge Problems." *arXiv preprint 2303.13375* (2023).

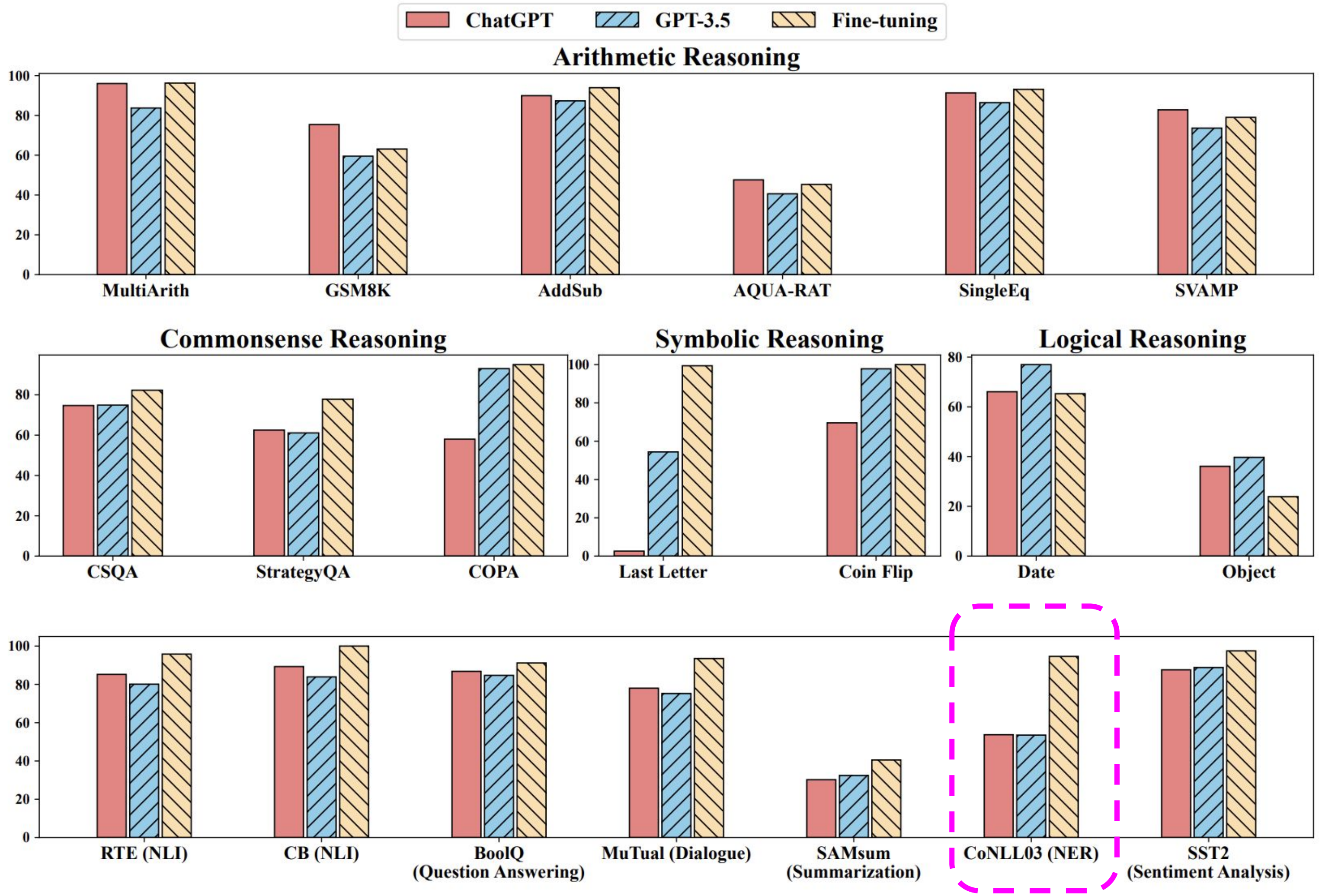
Model size matters (so far) !



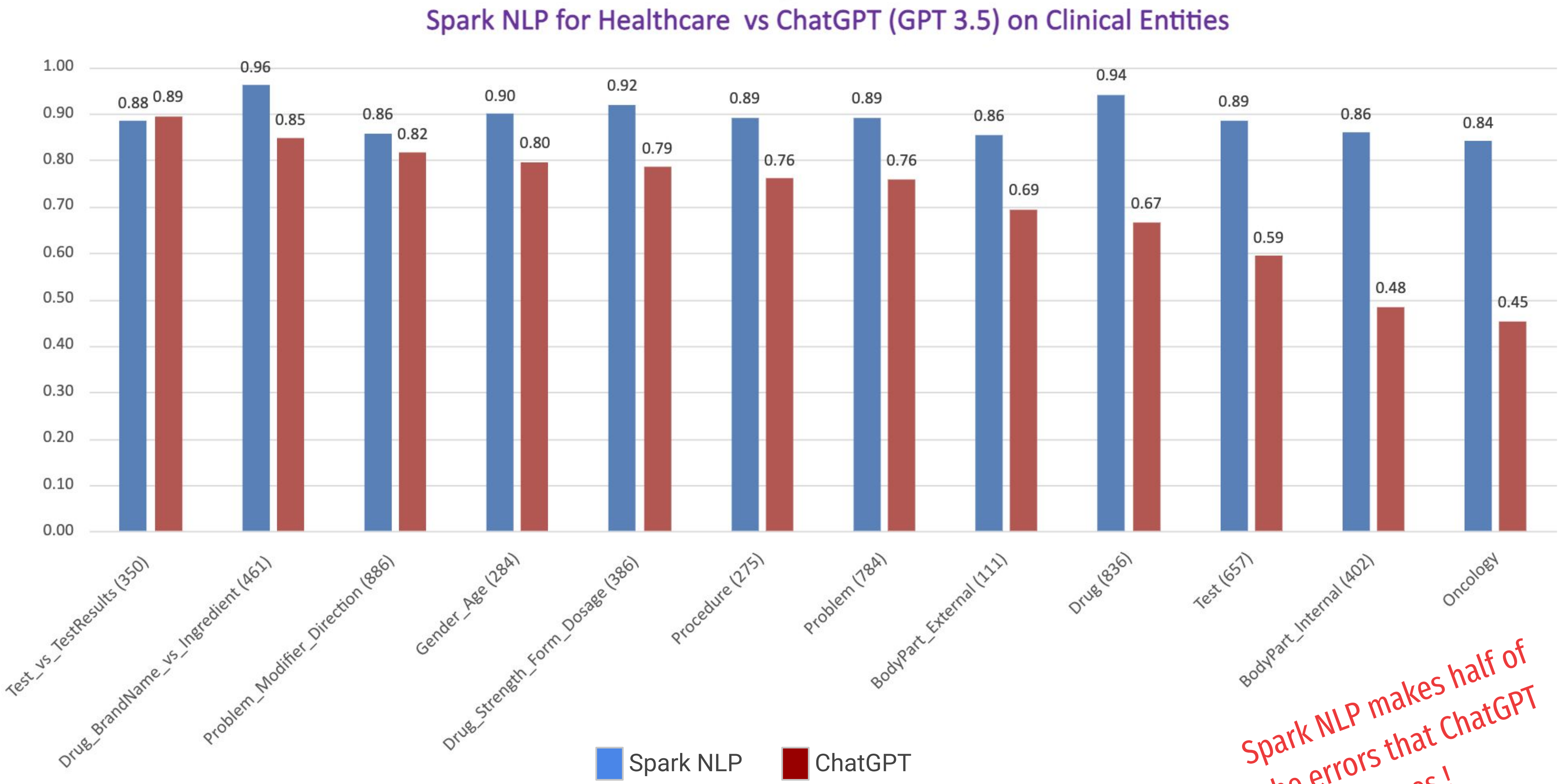
ChatGPT (GPT-4) still answers > %40 of the questions incorrectly



ChatGPT vs GPT-3.5 vs Fine-tuned Models



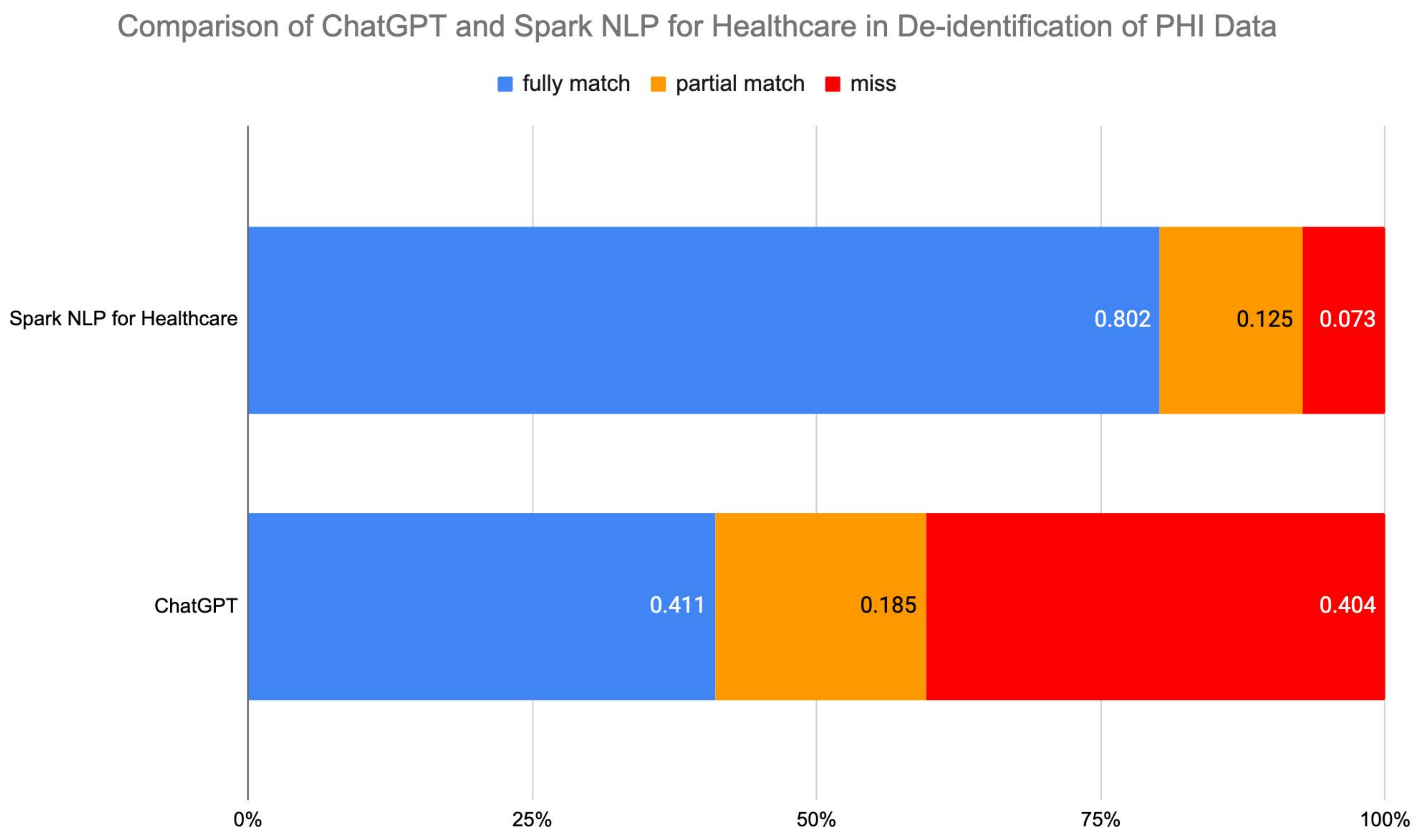
Qin, Chengwei, et al. "Is chatgpt a general-purpose natural language processing task solver?." *arXiv preprint arXiv:2302.06476* (2023).



Spark NLP makes half of the errors that ChatGPT does !

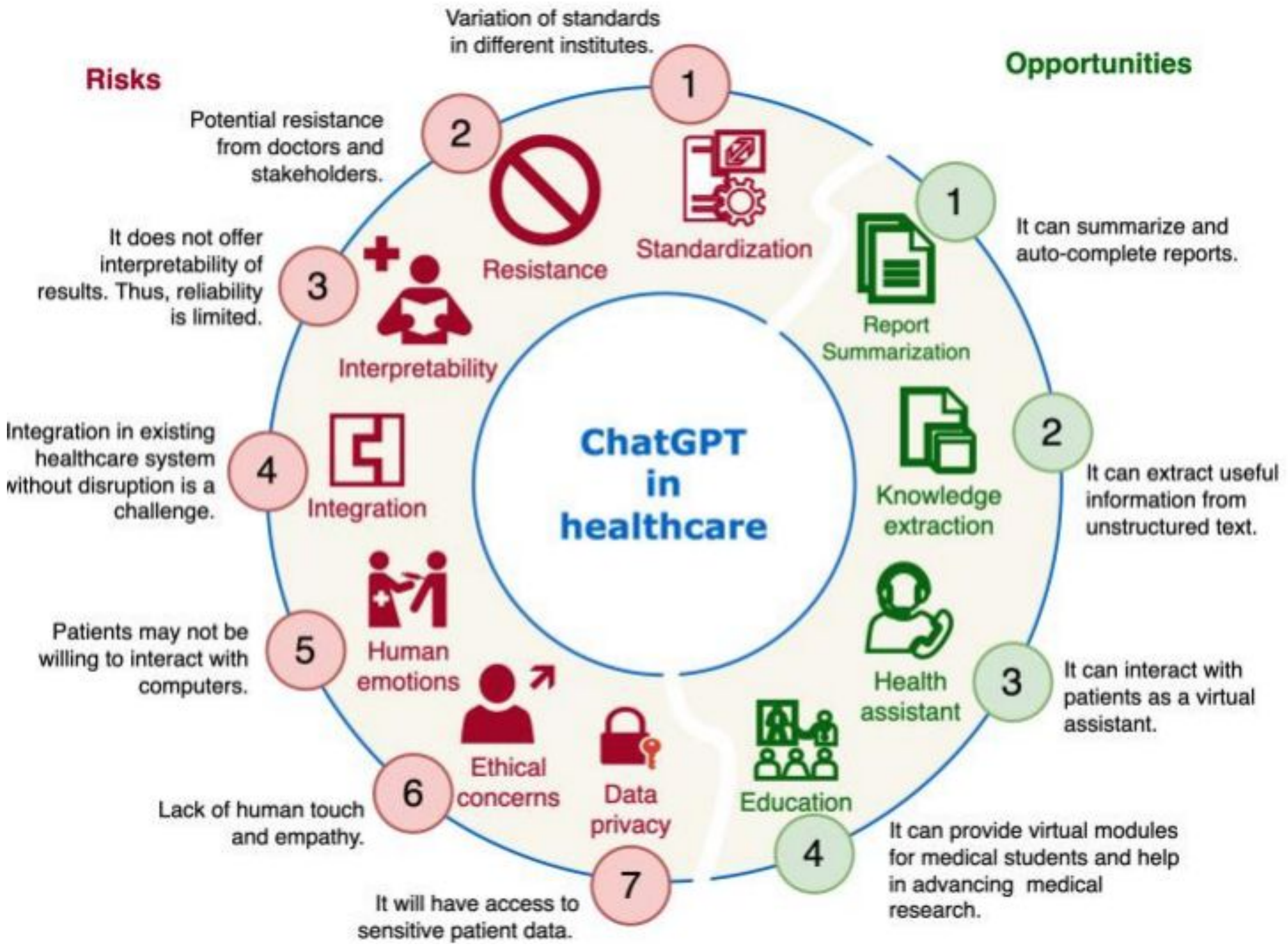
https://github.com/JohnSnowLabs/spark-nlp-workshop/tree/master/tutorials/academic/LLMs_in_Healthcare

https://medium.com/john-snow-labs/in-depth-comparison-of-spark-nlp-for-healthcare-and-chatgpt-on-clinical-named-entity-recognition-76b39477686



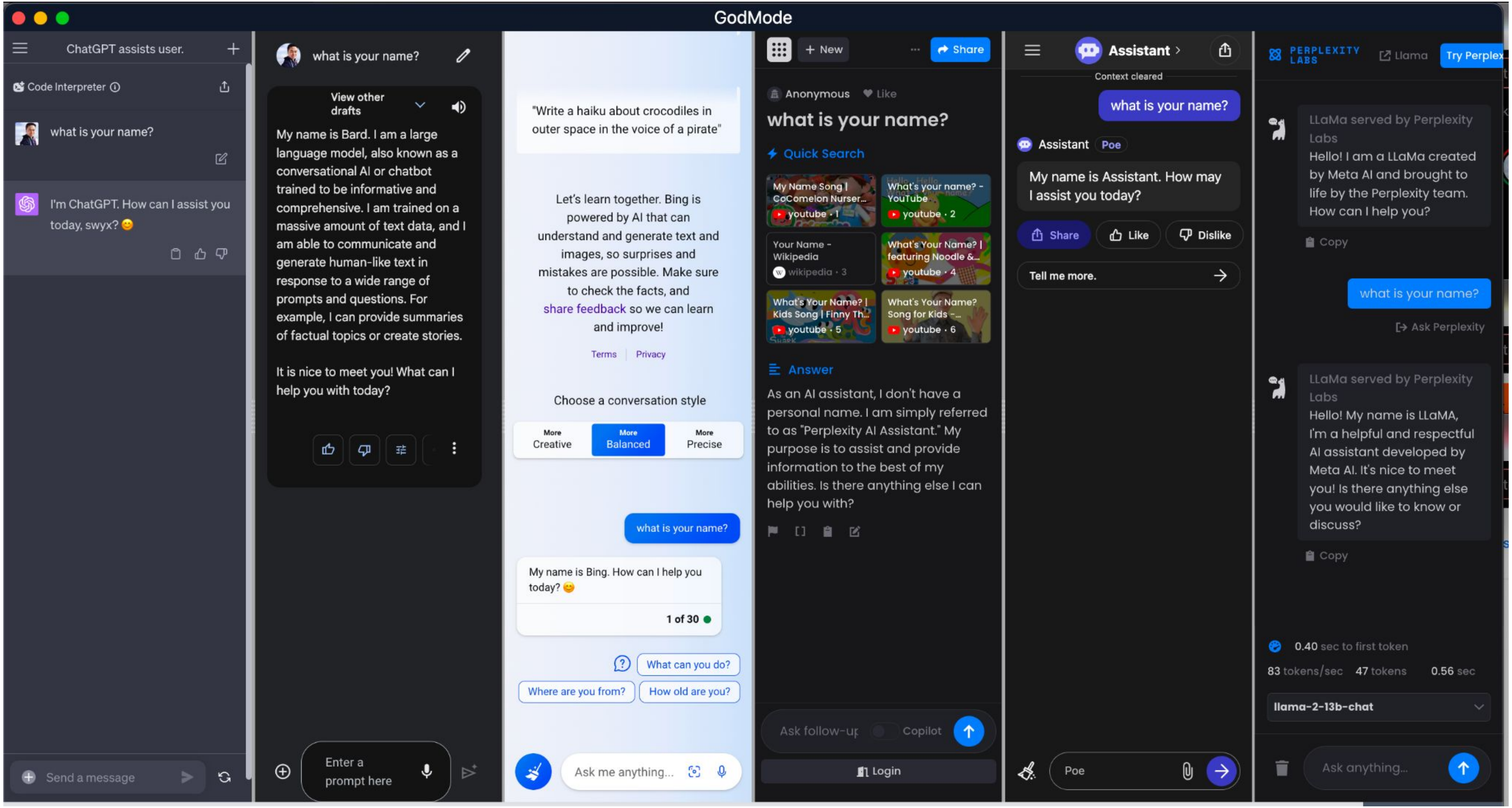
https://medium.com/john-snow-labs/a-comprehensive-comparison-of-chatgpt-and-spark-nlp-for-healthcare-in-de-identification-of-2cc2cf3dac6

Key opportunities and risks for ChatGPT in healthcare



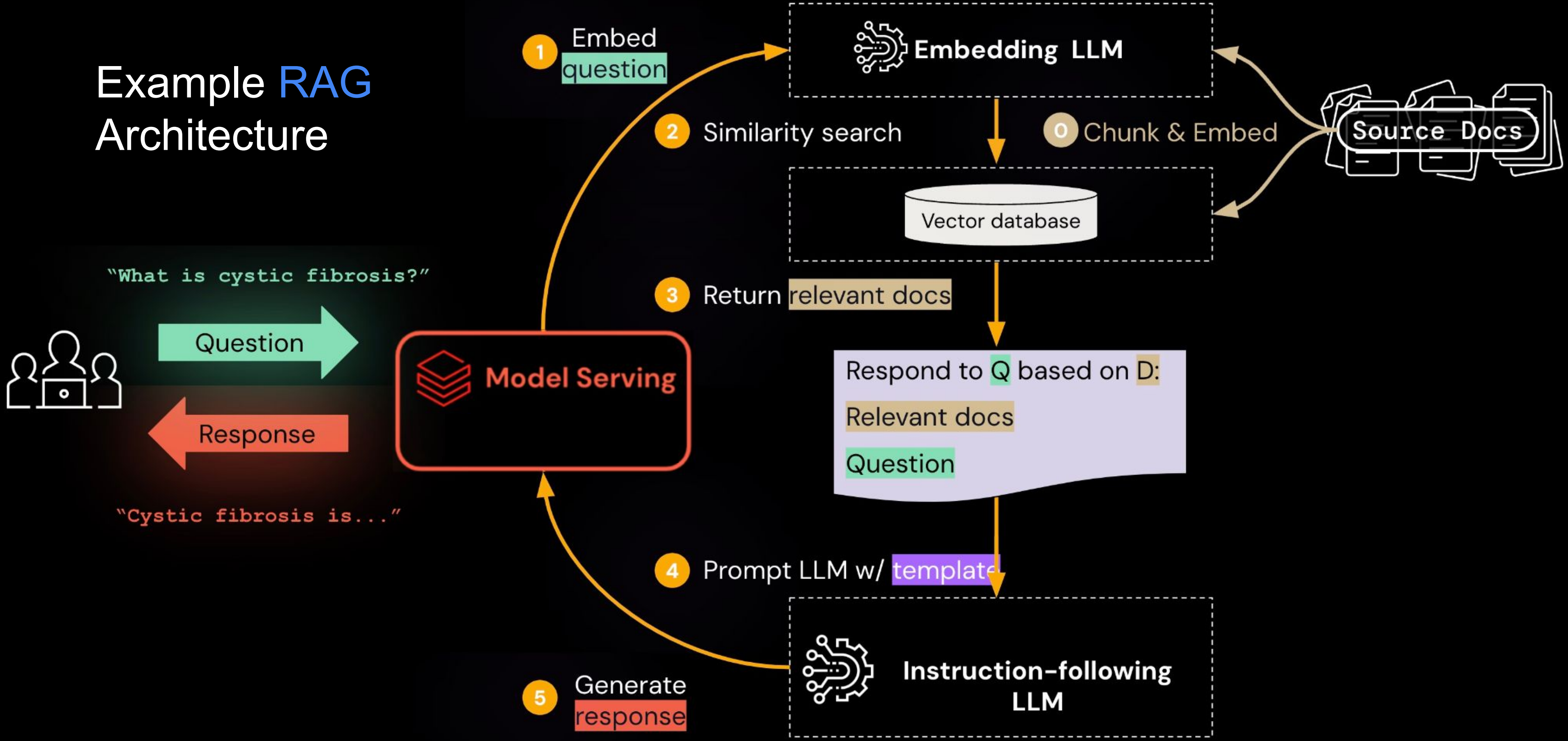
[ChatGPT and Large Language Models (LLMs) in Healthcare]

Popular Trends of LLM Applications in Enterprise



Retrieval-augmented Generation (RAG)

Example RAG Architecture



Retrieval-augmented Generation (RAG)

… the stages one can make a difference in a RAG application

1 Source Documents

- Preprocessing (OCR, basic cleaning, formatting, …)
- Metadata extraction (keywords, entities, author, title, …)
- Feature engineering (table understanding, chart2text, summarization, …)

2 Document Splitting/ Chunking

- Splitting strategy (content-aware, section-wise, task-based, char-based, tables, figures, items…)
- Max chunk size, overlap area, …

3 Split Embeddings

- Embeddings models (e5, allmpnet, gte, bge, openai-text-ada, … > MTEB)
- Model size and speed (384, 512, 768, 1024, …)
- Scalability (embeddings collection at scale)

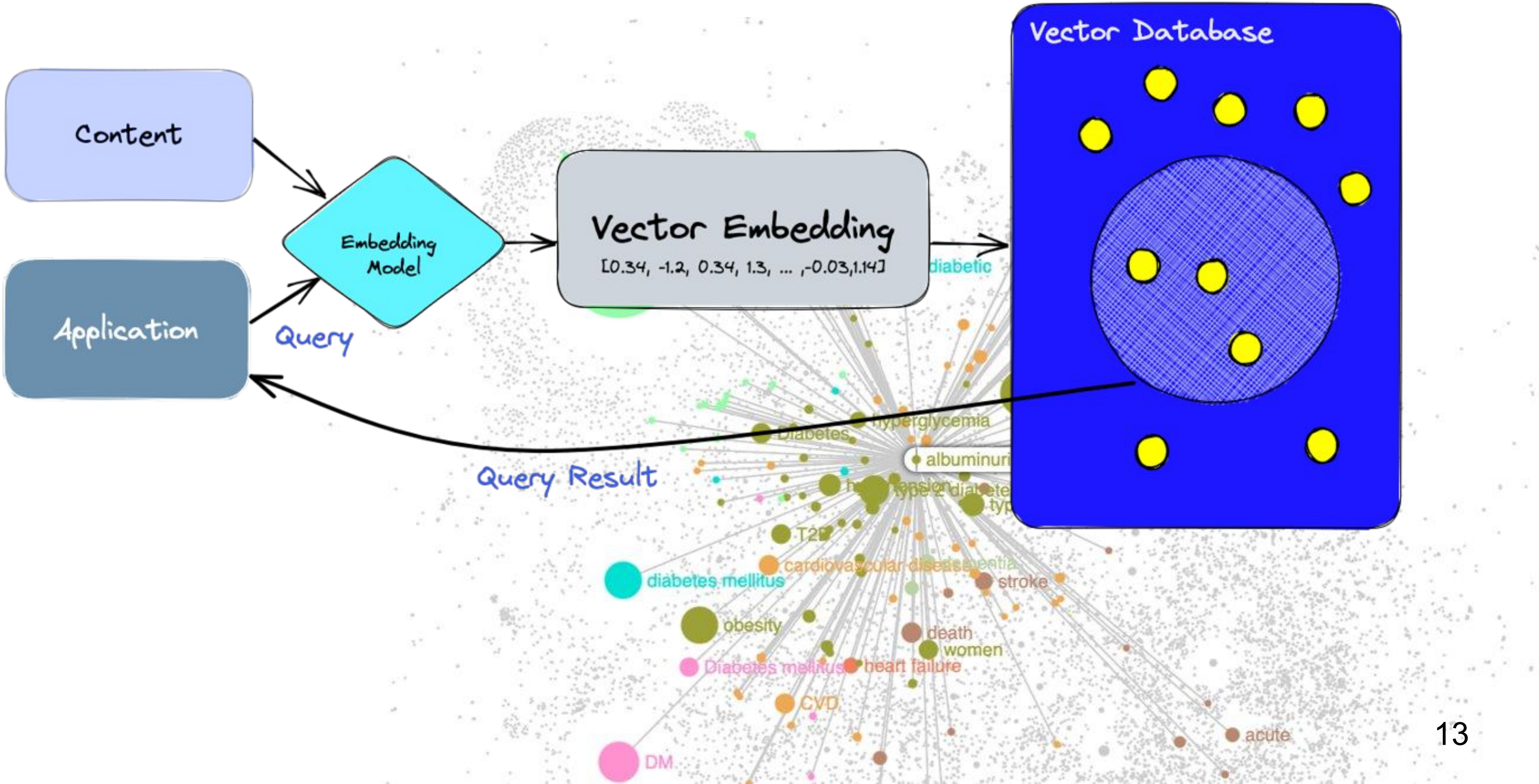
4 VectorDB

- Retrieval strategy (recursive, knn, BM25, span/ query expansion, …)
- Postprocessing (reranking, filtering, diversity, …)
- Speed and scalability

5 LLM

- Model performance, instruction following, guardrails, size, deployment)
- Context size (16K, 100K, … > chat memory)
- Prompt template (*given the context splits, answer the question*)

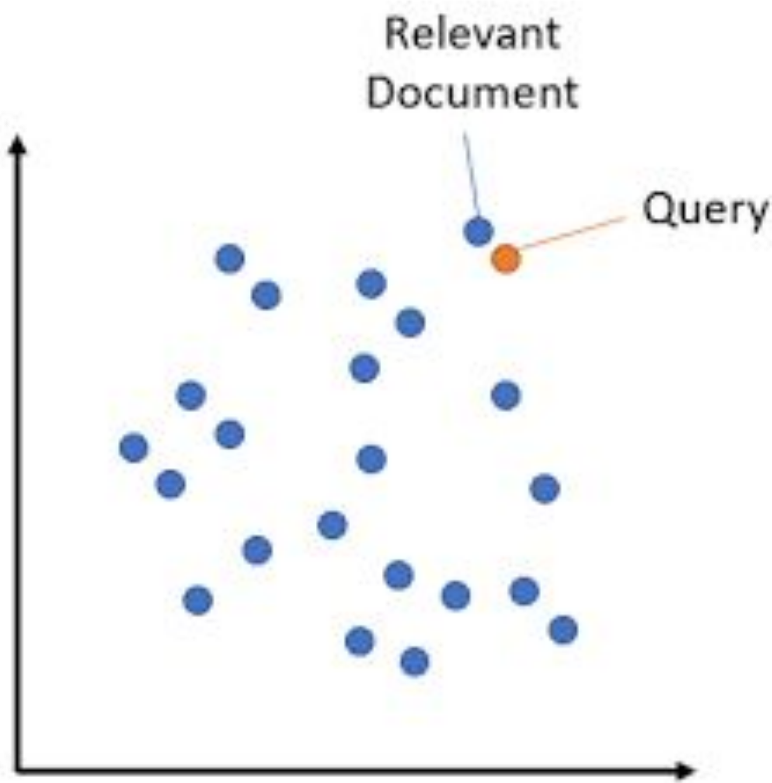
Picking the top similar document splits in RAG



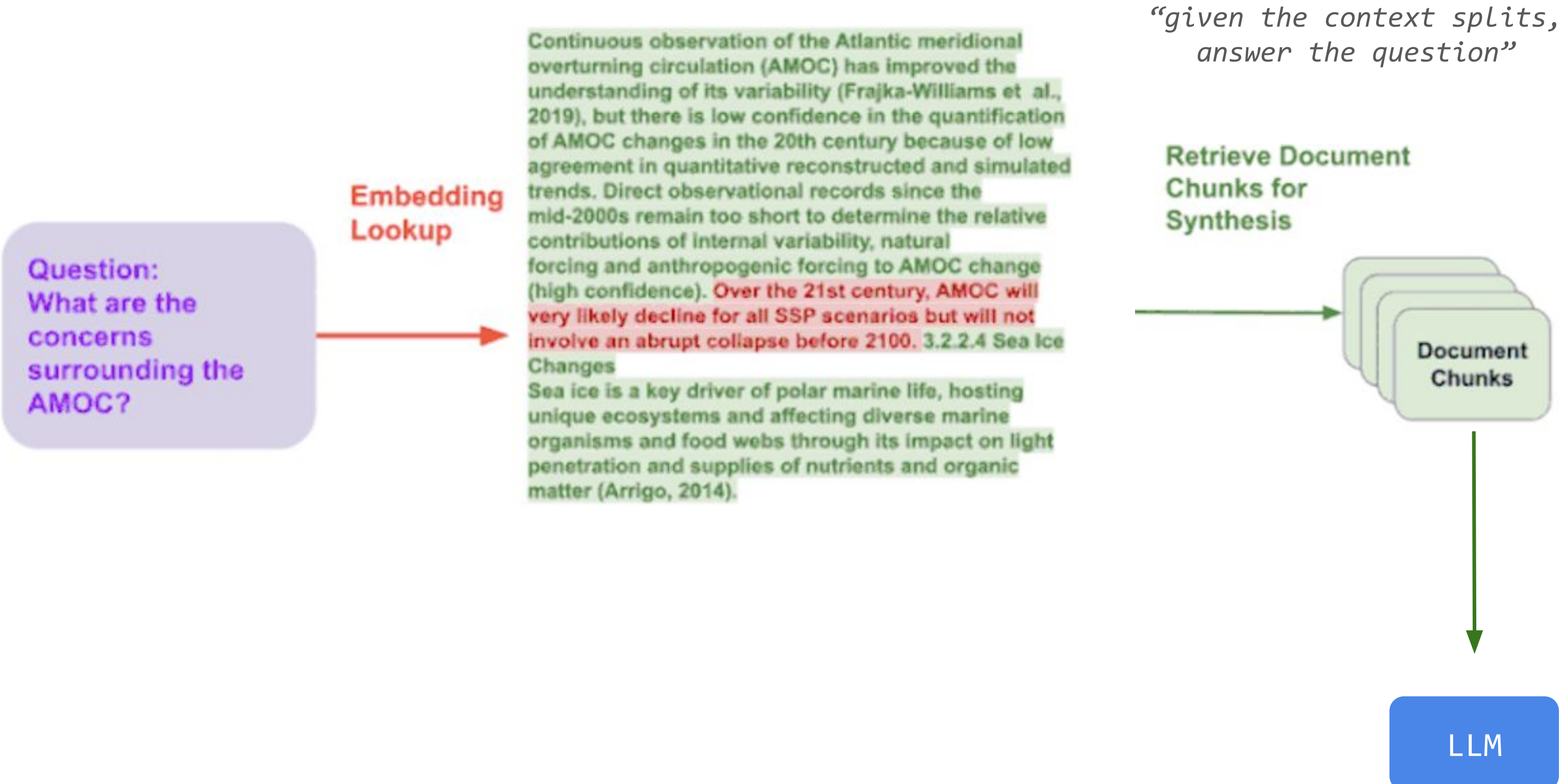
Pitfalls in Semantic Search

```
Query:  I like apples
Statement: I like all fruits but apples
Similarity: 0.8455890417098999

Statement: I dont like apples
Similarity: 0.8211406469345093

Statement: I love fruits
Similarity: 0.7780510187149048
```

```
Query:  I enjoy watching action movies.
Statement: I don't like action movies.
Similarity: 0.7076171636581421

Statement: I prefer documentaries.
Similarity: 0.4851611852645874

Statement: I really like to be kept on the edge of my seat.
Similarity: 0.3027774691581726
```



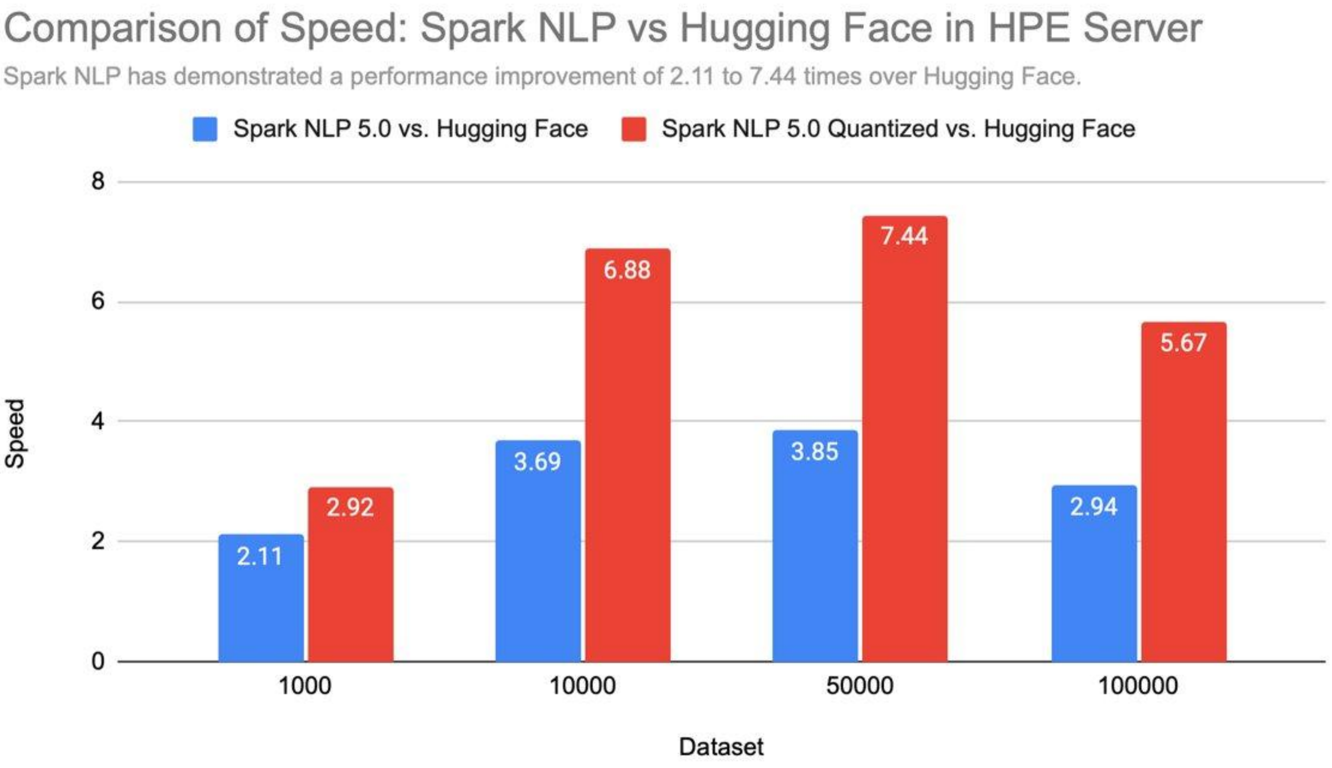
Finding similar splits via embeddings in RAG



Embeddings at Scale in RAG

Overall MTEB English leaderboard

Rank	Model	Model Size (GB)	Embedding Dimensions
1	sionic-ai-v2		
2	sionic-ai-v1		
3	bge-large-en-v1.5	1.34	1024
4	bge-large-en	1.34	1024
5	bge-base-en-v1.5	0.44	768
6	gte-large	0.67	1024
7	gte-base	0.22	768
8	e5-large-v2	1.34	1024
9	bge-small-en-v1.5	0.13	384
10	instructor-xl	4.96	768
11	instructor-large	1.34	768
12	e5-base-v2	0.44	768
13	multilingual-e5-large	2.24	1024
14	e5-large	1.34	1024
15	gte-small	0.07	384
16	gte-small	0.07	384
17	text-embedding-ada-002		1536
18	e5-base	0.44	768
19	e5-small-v2	0.13	384
20	instructor-base	0.44	768

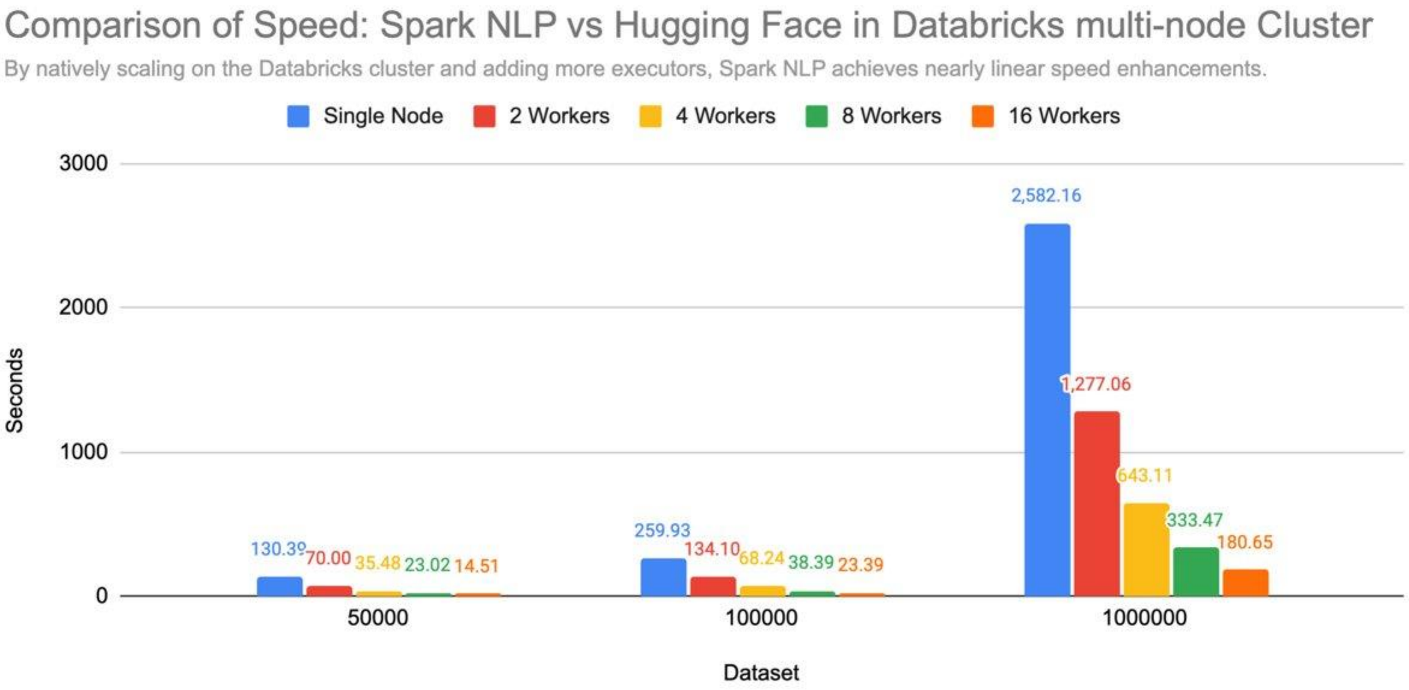


Spark NLP based on ONNX Runtime vs.
Hugging Face based on PyTorch,
single machine, 32-core, 80-GB memory

Embeddings at Scale in RAG

Overall MTEB English leaderboard

Rank	Model	Model Size (GB)	Embedding Dimensions
1	sionic-ai-v2		
2	sionic-ai-v1		
3	bge-large-en-v1.5	1.34	1024
4	bge-large-en	1.34	1024
5	bge-base-en-v1.5	0.44	768
6	gte-large	0.67	1024
7	gte-base	0.22	768
8	e5-large-v2	1.34	1024
9	bge-small-en-v1.5	0.13	384
10	instructor-xl	4.96	768
11	instructor-large	1.34	768
12	e5-base-v2	0.44	768
13	multilingual-e5-large	2.24	1024
14	e5-large	1.34	1024
15	gte-small	0.07	384
16	gte-small	0.07	384
17	text-embedding-ada-002		1536
18	e5-base	0.44	768
19	e5-small-v2	0.13	384
20	instructor-base	0.44	768



By **natively scaling** on the Databricks cluster and adding more executors, **Spark NLP 5.0** achieves nearly **linear speed enhancements**.

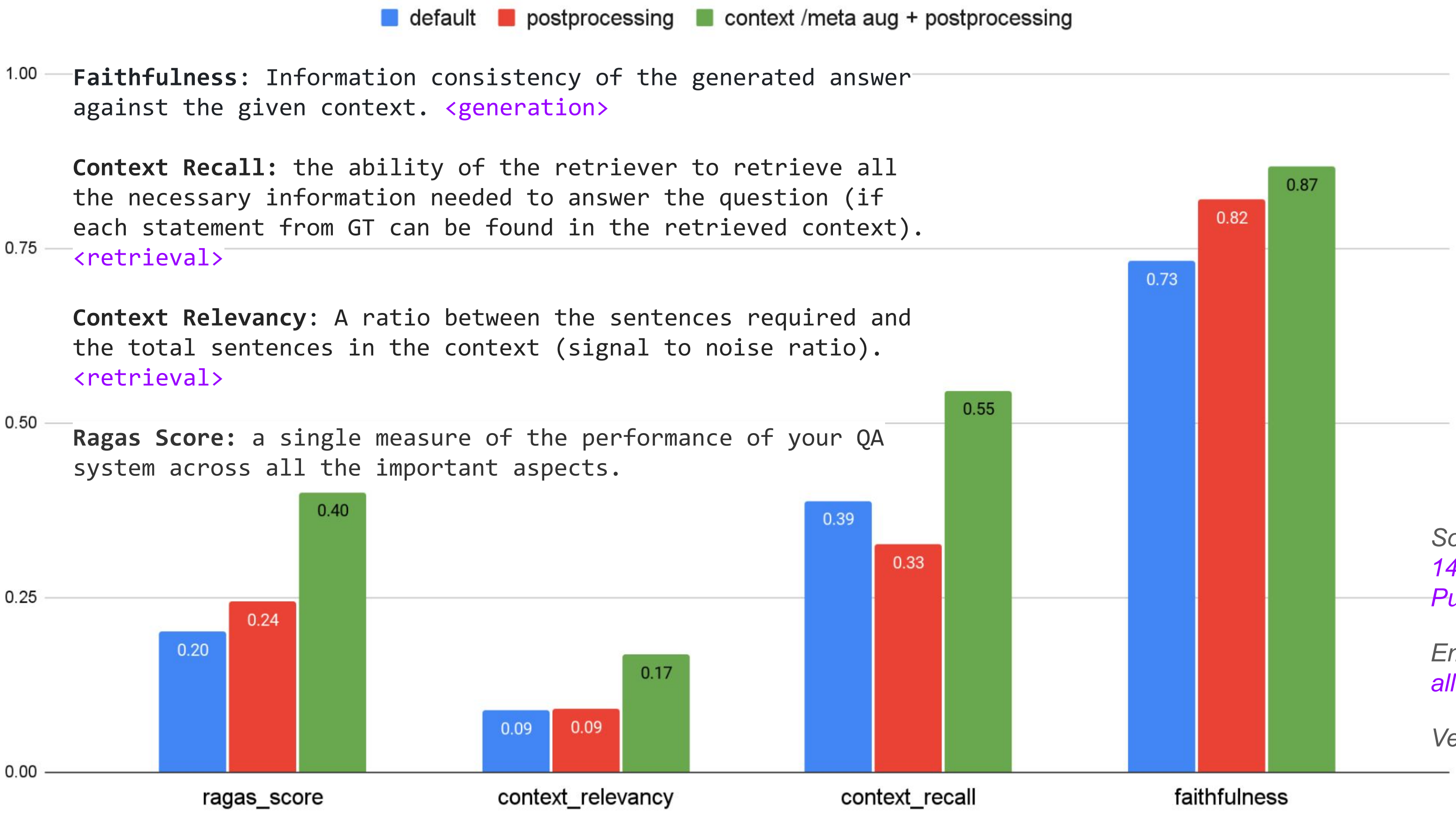
John Snow Labs - RAG Benchmarks

Faithfulness: Information consistency of the generated answer against the given context. <generation>

Context Recall: the ability of the retriever to retrieve all the necessary information needed to answer the question (if each statement from GT can be found in the retrieved context). <retrieval>

Context Relevancy: A ratio between the sentences required and the total sentences in the context (signal to noise ratio). <retrieval>

Ragas Score: a single measure of the performance of your QA system across all the important aspects.



Source KB:
1400 articles from Pubmed diabetes

Embeddings:
allmpnet-base-v2

Vector DB: Faiss

https://github.com/explodinggradients/ragas

Evaluation framework for your Retrieval Augmented Generation (RAG) pipelines

Foundational LLMs vs Smaller Domain-specific Language Models

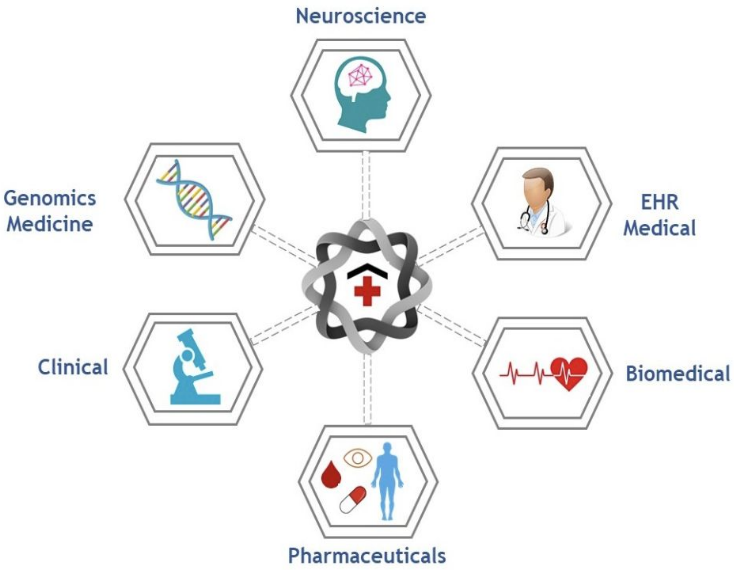
> Given that LLMs already encode clinical knowledge, do we still need to train or fine-tune our own use in clinical settings ?

- **Small Specialized Models Outperform:** Latest researches demonstrate that small, specialized clinical models outperform even fine-tuned LLMs in clinical settings.
- **Efficiency with Pre-Training:** Models that are pre-trained on clinical tokens can be smaller and more parameter-efficient.
- Surprisingly, even models trained on scientific domains, like **PubmedGPT, do not outperform smaller clinical models.**
- **USMLE vs. Clinical Tasks:** Despite performing well on medical exam questions like those in the USMLE, scientific-domain models struggle with tasks in a clinical setting, indicating a significant difference in requirements.
- **Need for Real-World Data:** To be truly effective, LLMs must be trained on real-world clinical data. Privacy and confidentiality must be navigated carefully.
- **Benchmarks Aligned with Real-World Scenarios:** We need more benchmarks that reflect actual clinical situations, not just exam datasets.
- **Nuanced Metrics Required:** Current tasks and metrics don't fully cover the diverse range of activities clinicians engage in. Human evaluation and more nuanced metrics are necessary.
- **Further Research Required:** Additional studies are needed to understand the impact of instruction tuning and RLHF on the performance of both LLMs and domain-specific language models.

RAG vs Fine-tuning ?

- TL:DR > **Most Cases Favor RAG**
- **Task-Specific Needs:** LLMs excel in text generation, QA, summaries, and content creation. For complex, domain-specific classification or regression tasks, fine-tuning is better.
- **Desired Modifications:** Use RAG to teach new facts and improve answer accuracy. Use fine-tuning to change style or tone.
- **Data Update Frequency:** RAG is better for frequently changing data as it updates automatically.
- **Privacy Concerns:** Fine-tuning can expose sensitive data and requires trust in the LLM provider. RAG allows granular access control.
- **Explainability:** RAG enables citations for verification, while fine-tuning does not allow easy investigation into the correctness of answers.
- **Costs:** Fine-tuning is generally more expensive, especially in ongoing operational costs.
- **Customer Preference:** Most of the customer cases are better suited for RAG.
- **Fine-Tuning Retriever:** When fine-tuning is employed, it's generally applied to the retriever in a RAG application, not the LLM itself.
- **Combination Approach:** In some cases, a combination of RAG and fine-tuning might be the best solution.

No LLM or RAG application can answer this question alone !

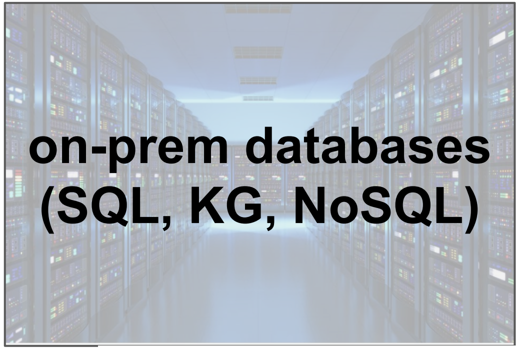


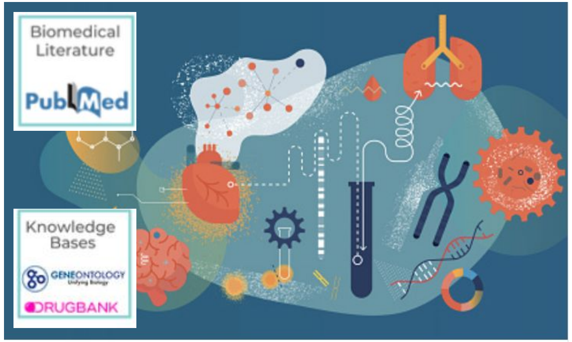
>> Give me all the patients who have type 2 diabetes, using metformin for the last 3 years, and also recently diagnosed stage-IV lung cancer?

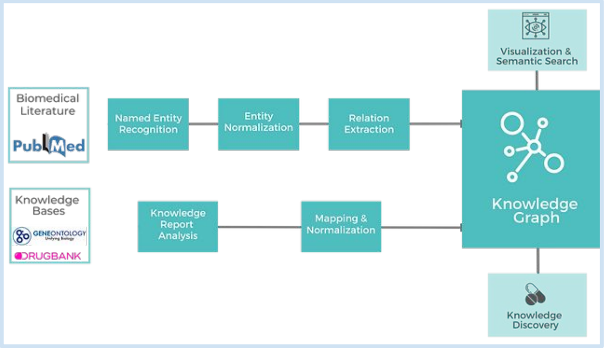
Unstructured EHR data

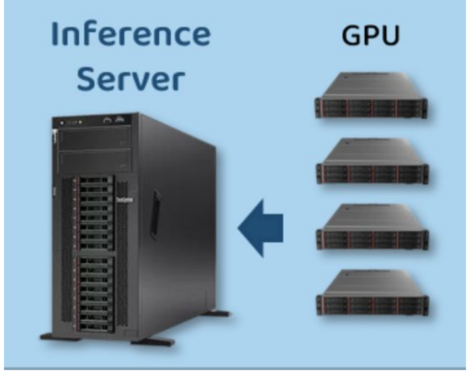
John Snow Labs - Medical Chatbot

-> Using LLMs as smart agents rather than information retrieval bots.









John Snow Labs Medical Chatbot

- RAG via KBs & Text2SQL via DBs

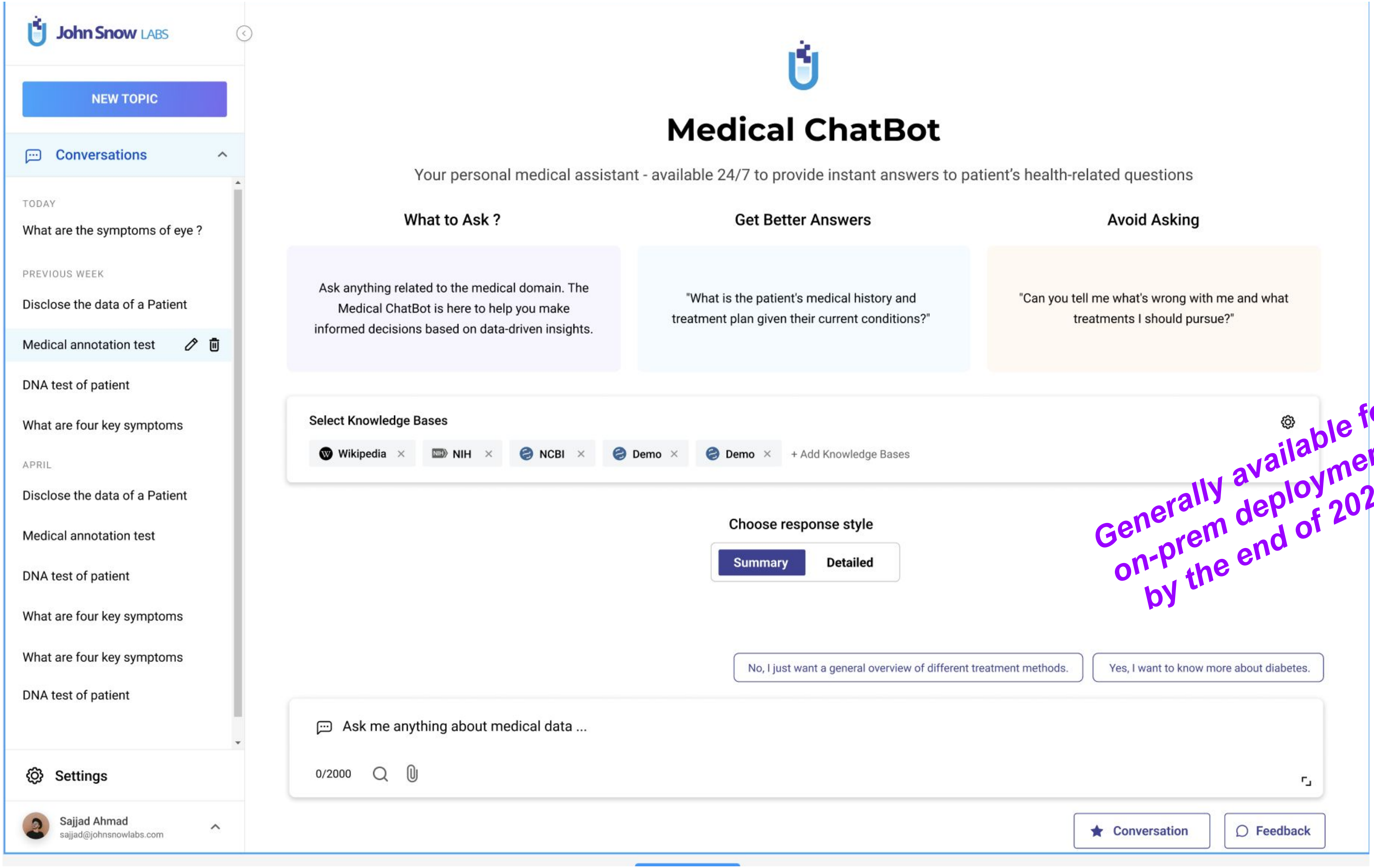
- KBs from Pubmed, MedArxiv, Clinical Trials, etc.

- KBs from your in-house documents

- Chat mode for swift interaction

- Citing the resources that the answer is generated from.

- No hallucinations.



Generally available for on-prem deployments by the end of 2023 !

1.Clinical Named Entity Recognition (NER)

- Blogposts and videos:
- Clinical NER Pipeline (with pretrained models)
- Clinical NER Models
  - with LightPipeline
  - NER Visualizer
- Clinical NER Chunk Merger
- Clinical NER Training
  - NERDL Graph
  - Evaluating the Model
  - Saving the model and using it in different pipeline
- BertForTokenClassification NER models
- Zero-Shot NER models
- Pretrained NER Profiling Pipelines
- NER Model Finder Pretrained Pipeline
- NER Model Playground:

2.ContextualParser (Rule Based NER)

- Date of Brith Contexrual Parser Model

3.Clinical Assertion Status

- Pretrained Assertion Status Models
- Oncology Assertion Models
  - Assertion Filterer Results
- Assertion Visualizer
- Train a Custom Assertion Model
  - Assertion Graph
  - Evaluating the Model

4.Clinical Deidentification

- Masking
- Reidentification
- Enriching with Regex and Override NER
- Obfuscation
- Shifting Days
  - Shifting days according to the ID column
  - Shifting days according to specified values
- Age Groups Obfuscation

5.Clinical Relation Extraction

- Pretrained Relation Extraction Models
  - Posology Relation Extraction
  - ReDL - ADE
- Merging Multiple RE Model Results
- Zero-shot Clinical Relation Extraction Model
- Train a Custom Relation Extraction Model
  - RE Graph

6.Clinical Entity Resolvers

- Sentence Entity Resolver Models
  - RxNorm Resolver
  - RxNorm with DrugNormalizer
  - Drug Spell Checker
  - ICD-10-CM Resolver
  - Entity Resolver Visualizer
  - CPT Resolver
  - BertSentenceChunkEmbeddings
  - Router - Using Resolver Models Together
  - Sentence Entity Resolvers with EntityChunkEmbedding

7.Chunk Mapping

- Pretrained ChunkMapper Models
- Chunk Mapping with Fuzzy Distance Calculation
- Creating a Mapper Model
- ResolverMerger - Using Sentence Entity Resolver and ChunkMapperModel Together

8.Pretrained Clinical Pipelines

9.Clinical Text Classification

- Classifiers
  - Load & Prepare ADE Classification Dataset
  - DocumentMLClassifier with Logistic Regression
  - GenericClassifier
  - FewShotClassifier
  - Pretrained Clinical Text Classification Models
    - genericclassifier_sdoh_alcohol_usage_sbiobert_ca
    - bert_sequence_classifier_sdoh_community_preser
    - classifierdl_ade_biobert
    - classifierdl_gender_biobert

10.Medical LLM

- Medical Text Summarization
  - summarizer_clinical_jsl
  - Text Summarization with Extractive Approach
- Medical Question Answering
  - clinical_notes_qa_base
- Medical Text Generation
  - text_generator_biomedical_biogpt_base
  - BioGPT - Chat JSL - Closed Book Question Answ
    - biogpt_chat_jsl
- Text2SQL Generation
  - Text2SQL_MIMICSQL
  - Text2SQL_With_Schema_Single_Table

11.Serving Spark NLP with API: Fast API with LightPipelines

- Using Fast API and LightPipeline
  - Dockerfile
  - Other files of the project
  - Example to serve 2 pipelines
  - Keys file
  - Building and running Docker
  - Consuming the API from a Python Script

12.Serving Spark NLP with API: Synapse ML

- Preparing a pipeline with Entity Resolution
- Creating a JSON file with the clinical note
- Running a Synapse server
- Checking Results

https://bit.ly/healthcare_nlp_workshop_2023

```python
from glob import glob

from langchain.chains import ConversationalRetrievalChain
from langchain.memory import ConversationBufferMemory
from langchain.document_loaders import UnstructuredMarkdownLoader
from langchain.text_splitter import CharacterTextSplitter
from langchain.embeddings import OpenAIEmbeddings
from langchain.vectorstores import FAISS
from langchain.llms import OpenAI


documents = []
for markdown_path in glob(f"{data_path}/*.md"):
    loader = UnstructuredMarkdownLoader(markdown_path)
    documents.append(loader.load()[0])

llm = OpenAI(temperature=0)

text_splitter = CharacterTextSplitter(chunk_size=1000, chunk_overlap=0)
texts = text_splitter.split_documents(documents)

embeddings = OpenAIEmbeddings()
db = FAISS.from_documents(texts, embeddings)

retriever = db.as_retriever()
memory = ConversationBufferMemory(memory_key="chat_history", return_messages=True)

qa = ConversationalRetrievalChain.from_llm(llm, retriever, memory=memory)
answer = qa.run(question)
print(answer)
```

```python
retriever = JSLEmbeddingRetriever(
    document_store=document_store,
    scale_score=False,
    embedding_model="all-mpnet-base-v2"
)
```

Presented by John Snow LABS

NLP SUMMIT

Thank you !

Veysel Kocaman

Head of Data Science

John Snow Labs