**Benchmark:** ViDoRe<https://github.com/illuin-tech/vidore-benchmark?tab=readme-ov-file>, ArXivqa test collection (500 istanze, <https://huggingface.co/datasets/vidore/arxivqa_test_subsampled>)

**Task:** Retrieval-enhanced multiple-choice question answering su documenti scientifici ArXiv

**Obiettivo:** Realizzare un Google Colab notebook per comparare prestazioni di information retrieval tra *(i)* modelli rappresentazionali multimodali (i.e., retrieval di chunk visivi da pagine PDF) con late interaction, *(ii)* text-only (con retrieval di chunk testuali tradizionali) con late interaction e *(iii)* text-only con late chunking. Per ogni retriever, valutare infine l'accuratezza downstream in multiple-choice question answering usando un modello generativo di proprio piacimento (es. Gemma 2B).

**Modelli:**

* Multimodal retriever con late interaction (usare Byaldi, <https://github.com/AnswerDotAI/byaldi>)
  + <https://huggingface.co/vidore/colpali-v1.2> (2.9B, 5.5GB VRAM)
  + <https://huggingface.co/vidore/colqwen2-v0.1> (2.2B, 4.7GB VRAM)
* Text-only retriever con late interaction
  + <https://huggingface.co/colbert-ir/colbertv2.0> (110M)
  + opzionale: <https://huggingface.co/jinaai/jina-colbert-v2> (559M)

**Notebook utili:**

* Multimodal RAG using ColPali (with Byaldi) and Qwen2-VL: <https://github.com/merveenoyan/smol-vision/blob/main/ColPali_%2B_Qwen2_VL.ipynb>
* Esempio codice Late Chunking: <https://www.linkedin.com/posts/victorialslocum_contextual-retrieval-is-a-new-rag-method-activity-7245389587768909824-37b9?utm_source=share&utm_medium=member_desktop>

<https://www.linkedin.com/posts/ownyourai_the-perfect-local-llm-gpu-cluster-doesnt-activity-7257286812367261698-gyqP?utm_source=share&utm_medium=member_android>

Vi è molta confusione sull'approccio metodologico da adottare nel momento in cui si parla di "GraphRAG"

In assenza di ulteriori specifiche, si tende a far riferimento all'omonimo paper di Microsoft

Più in generale però, la pipeline è la seguente

 Come in RAG moderno, parliamo di una procedura puramente prompting-based più o meno articolata

(i) si parte da un corpus di documenti e si estraggono entità e relazioni con un LLM.

Già in questo step vi possono essere delle varianti.

C'è chi chiede al LLM di stimare anche la forza della relazione (informazione da sfruttare nelle successive fasi di prompting). Chi invece chiede, come Microsoft, di generare anche una descrizione per ogni entità individuata nel testo (sempre al fine di ottenere risultati migliori nelle successive fasi di prompting).

questo processo è quello che più ha impatto sulla "economia" metodologica. Qualora non si vogliano investire grandi risorse sul fronte OpenAI Key, GPT4o-mini è la via: vedi costi, <https://techcommunity.microsoft.com/t5/ai-azure-ai-services-blog/graphrag-costs-explained-what-you-need-to-know/ba-p/4207978>

(ii) opzionalmente, le triple sintetiche ottenute possono essere usate per popolare un graphDB, principalmente Neo4j

(iii) le informazioni strutturate precedentemente estratte e opzionalmente indicizzate vengono sfruttate come grounding al fine di generare la risposta al quesito dell'utente, aumentando il contesto della singola domanda e riconducendoci a uno scenario RAG tipico

sul come sfruttare le triple a tal fine, si apre tutto un capitolo

* c'è chi tira fuori le entità dalla query dell'utente con un LLM, e cerca vicini diretti nel grafo o triple note a una certa profondità massima
* c'è chi prima usa algoritmi di clustering gerarchico o tool quali graph analytics per Neo4j allo scopo di individuare community (i.e., cluster con relazioni dense), e a quel punto recupera l'intera community cui l'entità query appartiene
* c'è chi tira giù le triple contesto mediante l'esecuzione di una query Chyper generata da un LLM

Il lavoro originale di GraphRAG Microsoft rientra nella famiglia metodologica del clustering ma persegue una quarta via ancora (difficilmente percorribile sul fronte costi)

Nello specifico, viene utilizzato un LLM per generare un riassunto testuale in merito al contenuto informativo di **ogni** community, istruendo il LLM alla menzione di entità (nodi) con alto degree e relative relazioni

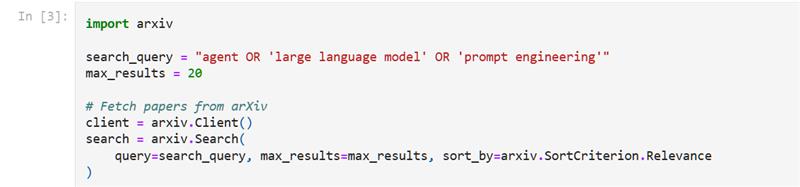
a questo punto, Microsoft usa un LLM per generare una risposta (cosiddetta "locale") usando come contesto il riassunto di una community... N community --> N risposte locali condizionate dalla descrizione testuale del loro contenuto informativo

Microsoft chiede inoltre al LLM di stimare la rilevanza / utilità della community al fine della risposta e, di conseguenza, quanto la risposta sia attendibile. Così facendo, per ogni risposta locale, vi è anche un weight silver.

Con un altro LLM prompting ancora, si genera una risposta globale partendo dalle singole risposte locali e loro peso

Detto ciò, vi è un notebook online che ritengo un'ottima base di partenza: <https://github.com/milvus-io/bootcamp/blob/master/bootcamp/RAG/advanced_rag/langgraph-graphrag-agent-local.ipynb>

All'interno vengono scaricati on-the-fly 20 paper da arXiv e indicizzati su Milvus come VectorDB



Si usa un modello OpenAI per estrarre nodi e archi, anche con constraint sul tipo:



Nota: tra l'altro si usa ollama, rendendo il passaggio da modelli closed --> a modelli open in locale molto agevolato; se sei interessato, trovi parecchie guide online

Viene creato un database Neo4j

Viene utilizzato LangGraph per costruire una elaborata pipeline automatica. Ma a noi basterebbe questo:



Importante: GraphRAG ha senso nella risoluzione di domande richiedenti multi-hop reasoning e/o aggregazioni, e non risposte riscontrabili direttamente nei chunk di testo di partenza. Esempio: *quanti sono i paper che parlano di...?*, similmente a quanto mostrato nel notebook

Di conseguenza, GraphRAG non è utile alla risoluzione del task che avevamo precedentemente individuato per te nell'ambito multimodal, i.e., multiple-choice QA su una specifica figura di un paper arXiv... Ma su domande più repository-level

La proposta quindi è creare un notebook diviso in due parti:

* multimodal, seguendo le nostre precedenti istruzioni
* graphrag, partendo dal notebook condiviso qua sopra

Sulla nuova parte graphrag, per creare coerenza, potresti usare gli stessi paper arXiv (o un subset) del dataset indicato per la prima parte multimodal

con <=10€ di budget, potresti estrarre nodi e archi con GPT4o-mini; in alternativa, potremmo condividerti un esempio di codice da noi fatto sull'uso di modelli open source (con tutte le implicazioni del caso, come ti dicevo)

su questa parte graphrag, come dicevo, useresti alcune delle migliori tecnologie: neo4j, milvus, langgraph...

e non ci sarebbe valutazione automatica delle prestazioni su un benchmark, come invece nel primo modulo multimodale... Ma solamente una piccola demo / parte interattiva con domande d'esempio a fine notebook

chiaramente, in virtù di questo nuovo modulo, puoi ribilanciare il carico di lavoro: es. usare solo ColQwen2 come modello multimodal nel primo modulo, anzichè anche ColPali

Mi fa piacere! Se hai qualunque difficoltà sui tempi e vedi che alcuni task ti portano via troppo, tagliamo alcuni pezzi dal modulo 1: ad esempio consideriamo solo ColQwen multimodal e ColBERT text-only, e buttiamo via anche il late chunking. Rimango disponibile. Buona giornata Roberto