AKADEMIA E FORCAVE
TË ARMATOSURA

# Hyrje në Modele të Mëdha Gjuhësore

## RAG (Gjenerim i përforcuar nga kërkimi)

Dr. Fiorela Ciroku

# Bazimi i MMGj-ve në njohuri të jashtme

Në leksionet e mëparshme kemi mësuar:

- si funksionojnë transformer-at
- si para-trajnohen modelet
- si përfaqësohet kuptimi me embeddings
- si funksionojnë vector databases

Tani vijmë tek arkitektura më e rëndësishme praktike për përdorimin e MMGj-ve në botën reale:

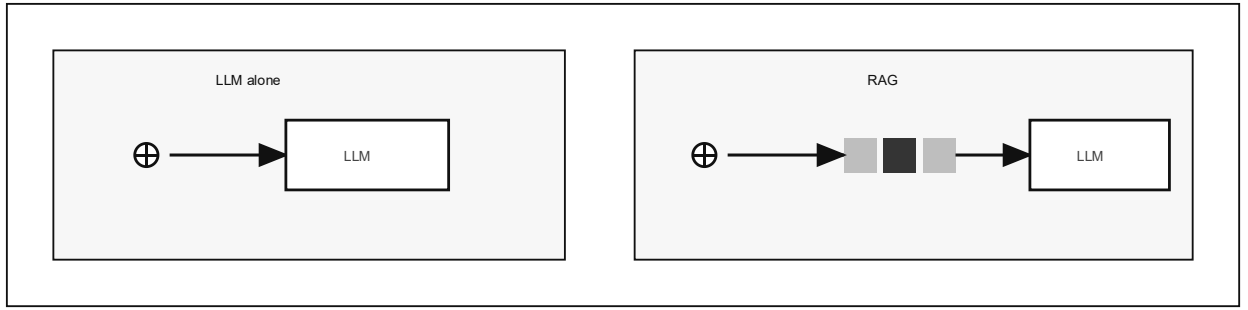
**Retrieval-Augmented Generation (RAG)**

# Kufizimet e modeleve gjuhësore të paratrajnuara

MMGj-të nuk janë databaza faktesh. Ato janë **modele probabilistike të gjuhës**.

Kjo do të thotë:

- Përgjigjja që duken “të sakta” janë shpesh thjesht *të mundshme*.
- Modeli nuk e di nëse diçka është ligjërisht e vlefshme.
- Modeli nuk di cilat dokumente janë zyrtare.
- Modeli nuk di versionin e fundit të një procedure.



# Përkufizimi formal i RAG

**Retrieval-Augmented Generation** përkufizohet si një arkitekturë ku gjenerimi i tekstit kushtëzohet nga dokumente të rikthyera përmes mekanizmave të kërkimit informacionit.

Formalisht:

- le të jetë q pyetja e përdoruesit,
- D korpusi dokumentar,
- R(q, D) operatori i retrieval.

$$\mathrm{LLM}(q, R(q, D))$$

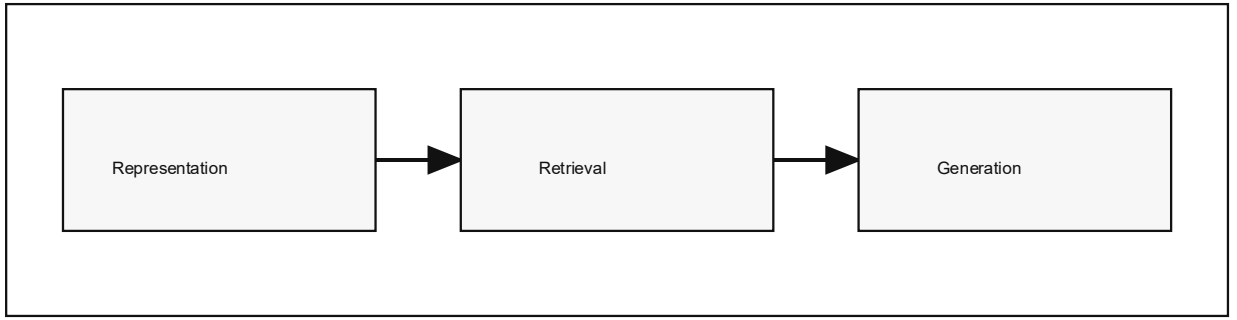
Kjo ndarje lejon kontroll më të madh mbi burimet e njohurisë.

# Komponentët konceptualë të RAG

Një sistem RAG mund të ndahet konceptualisht në:

- Një komponent përfaqësimi (që transformon  tekstin në vektorë)
- Një komponent kërkimi (që identifikon dokumente relevante),
- Një komponent gjenerimi (që prodhon përgjigjen).

Kjo ndarje lejon dizajnim, optimizim dhe vlerësim të pavarur të secilit komponent.



# Pozicionimi i RAG në ekosistemin MMGj

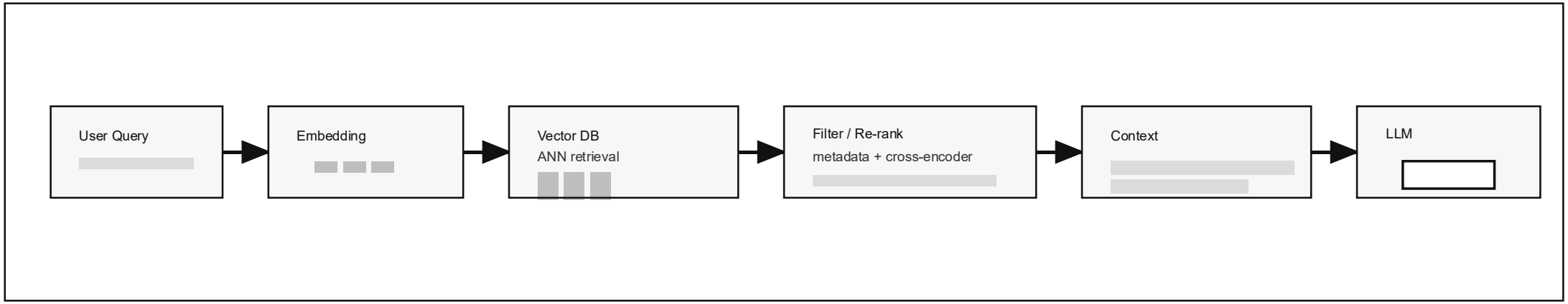
RAG qëndron midis dy ekstremeve:

- para-trajnimit (njohuri e ngrirë)
- fine-tuning (njohuri e koduar në parametra)

Ndryshe nga të dyja, RAG injekton njohuri **në kohë inferencë,** pa ndryshuar modelin, pa ri-trajnim.

Kjo e bën RAG veçanërisht të përshtatshëm për dokumente që ndryshojnë shpesh, si politika, procedura, manuale, etj.

# RAG Pipeline



# RAG si arkitekturë dyfazore

- Një sistem RAG ndahet qartë në dy faza:

**FAZA OFFLINE**

- Dokumentet përgatiten
- Ndahen në chunks
- Gjenerohen embeddings
- Ruhen në vector database

**FAZA ONLINE**

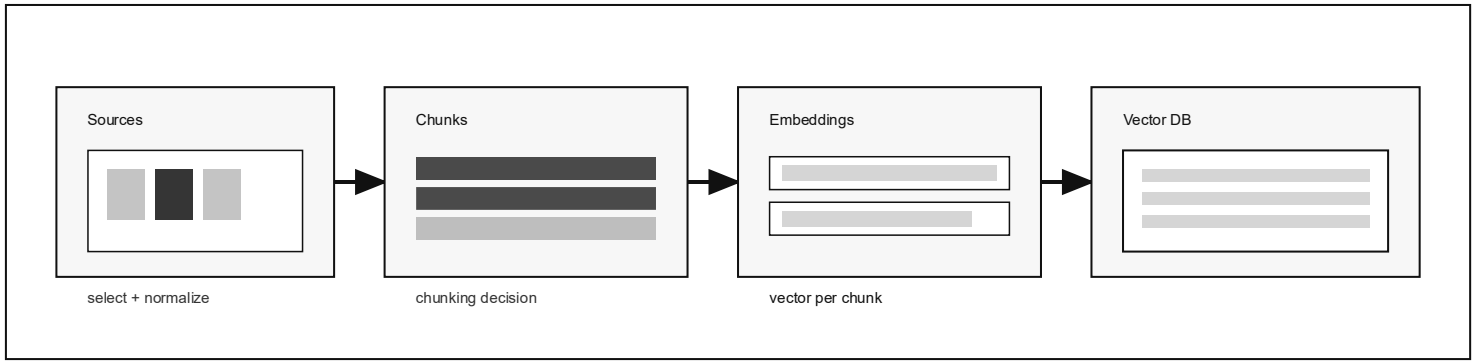
- Pyetja përpunohet
- Kryhet retrieval
- Dokumentet injektohen në kontekst
- Modeli gjeneron përgjigje

- Kjo ndarje është thelbësore për shkallëzim, mirëmbajtje, dhe auditim.

# Faza offline: Përgatitja e dokumenteve

Në fazën offline, korpusi dokumentar përgatitet për kërkim:

- përzgjidhen burime autoritative,
- dokumentet ndahen në segmente (chunks),
- për secilin chunk llogaritet embedding,
- Embedding pasqyrohen në bazën e të dhënave vektoriale.




Vendimet në këtë fazë (p.sh. madhësia e segmentit) ndikojnë drejtpërdrejt cilësinë e retrieval.

# Faza online: Përgatitja e përgjigjes

- Faza online e një sistemi RAG është ajo që ekzekutohet në kohë inferencë, pra për çdo kërkesë individuale të përdoruesit.

- Sistemi kryen një sekuencë të fiksuar hapash:

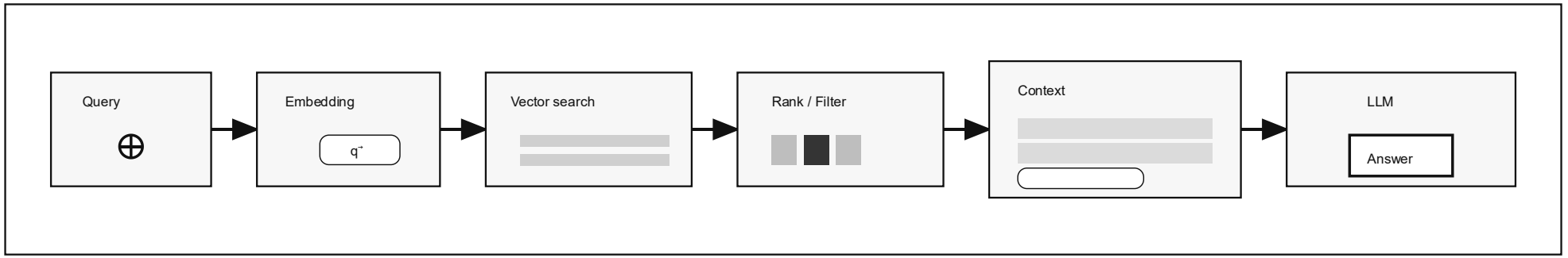
    1. Pyetja e përdoruesit përpunohet dhe përfaqësohet me  embedding.
    2. Kryhet kërkimi në indeksin vektorial për të identifikuar dokumente ose segmente relevante.
    3. Rezultatet e retrieval renditen dhe filtrohen sipas kritereve të paracaktuara (p.sh. relevanca, burimi, data).
    4. **Dokumentet e përzgjedhura bashkohen në një kontekst të vetëm që i paraqitet modelit gjuhësor.**
    5. Modeli gjeneron përgjigjen në mënyrë autoregresive, i kushtëzuar nga konteksti i injektuar.

# Faza online: Përgatitja e përgjigjes

Në fazën online, asnjë parametër i modelit nuk ndryshohet. RAG vepron ekskluzivisht duke manipuluar input-in dhe jo modelin.

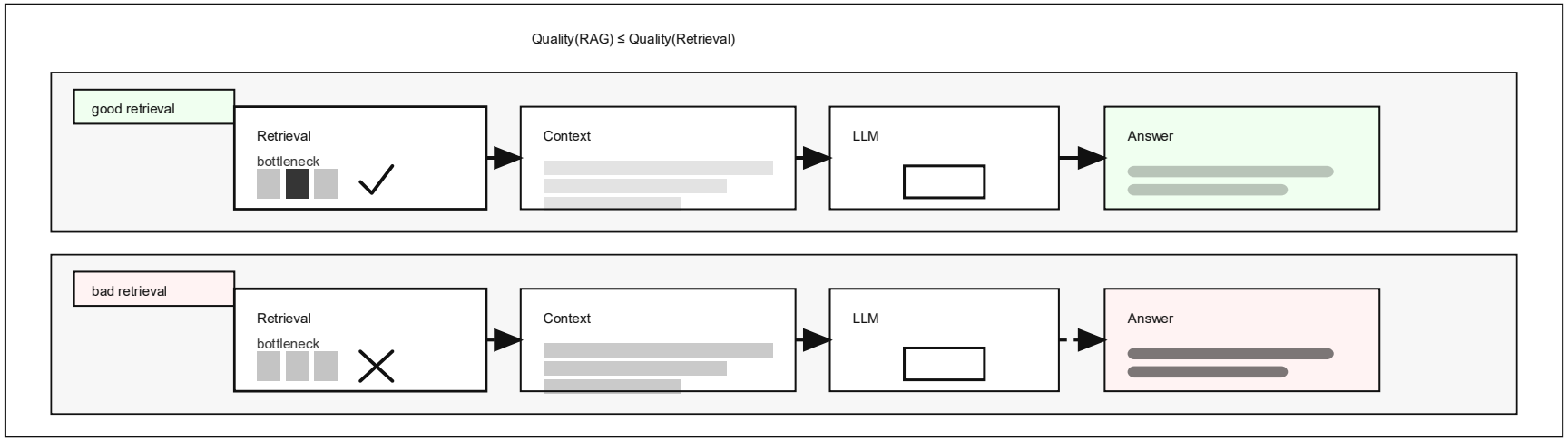
Kjo fazë është shpesh bottleneck-u kryesor, pasi përfshin:

- operacione të shumta në kohë reale,
- ndërveprim me shërbime të jashtme (vector database),
- dhe gjenerim nga modele të mëdha gjuhësore me kosto të lartë llogaritëse.



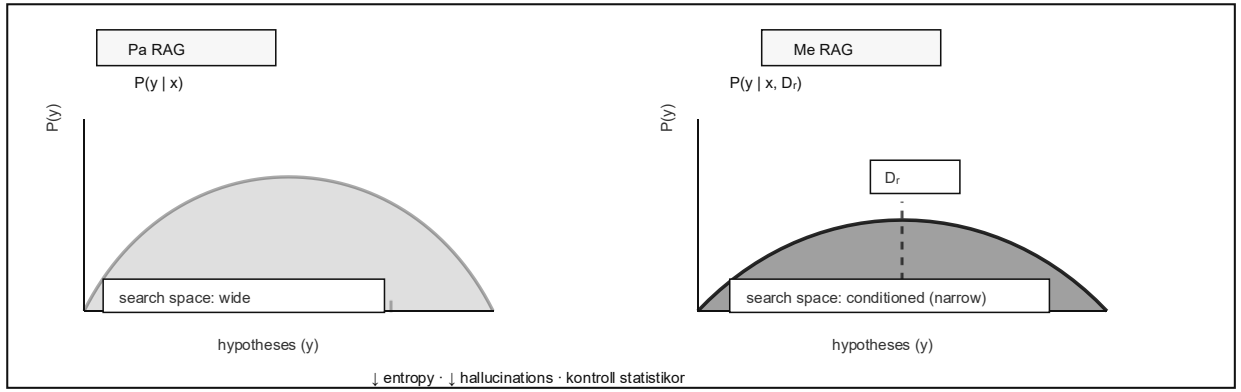
# RAG  dhe  varësia nga retrieval

- Në RAG, gjenerimi është **plotësisht i varur** nga retrieval.
- Nëse retrieval rikthen dokumente të gabuara, jo relevante, ose të paplota, atëherë gjenerimi do të jetë i gabuar, i paqartë, edhe nëse modeli është i fortë.
- Prandaj, në analizë formale: cilësia e RAG ka më pak rëndësi se cilësia e retrieval.



# Kufizimi i hapësirës së hipotezave

- RAG funksionon duke **kufizuar hapësirën e hipotezave** të modelit.
- Pa RAG, modeli zgjedh nga e gjithë shpërndarja e gjuhës.
- Me RAG, modeli zgjedh nga një nën-shpërndarje e kushtëzuar nga dokumentet
- Kjo ul entropinë e gjenerimit dhe probabilitetin e shpikjes.
- Ky është mekanizëm statistik, jo semantik.



# RAG si problem probabilistik

- Në nivel formal, RAG ndryshon shpërndarjen probabilistike të gjenerimit.

- Kjo nuk garanton korrektësi logjike, por:
  - redukton pasigurinë
  - ul entropinë e shpërndarjes (Entropia e një shpërndarjeje probabilistike mat shkallën e pasigurisë ose paqartësisë që ka një variabël e rastësishme mbi rezultatet e saj të mundshme.)
  - zhvendos masën probabilistike drejt përgjigjeve të mbështetura nga evidenca.

- Kjo është një pikë kritike për të kuptuar pse RAG **redukon**, por **nuk eliminon** haluçinacionet.

# RAG dhe entropia e gjenerimit

Në termat e teorisë së informacionit, injektimi i dokumenteve rrit informacionin e kushtëzuar dhe ul entropinë e gjenerimit.

Entropia e gjenerimit është një masë e pasigurisë së shpërndarjes probabilistike që modeli gjuhësor prodhon për token-in e radhës gjatë gjenerimit.

Kjo ka pasoja praktike pasi:

- përgjigjet bëhen më të qëndrueshme
- varianca mes përgjigjeve zvogëlohet
- krijimtaria reduktohet në favor të saktësisë

# Ndarja Knowledge / Reasoning

RAG formalizon një ndarje arkitekturore:

**Knowledge storage** → dokumentet | **Reasoning engine** → MMGj

Kjo ndarje:

- ul varësinë nga parametrat
- rrit transparencën
- lejon përditësim të njohurive pa retraining

# Faithfulness vs Correctness

**Faithfulness**: përgjigja është e bazuar në dokumentet e dhëna

**Correctness**: dokumentet janë faktikisht të sakta

RAG garanton (në masë të madhe) faithfulness, por jo saktësi absolute. Prandaj:

- Cilësia e korpusit është vendimtare
- RAG nuk zëvendëson verifikimin e burimeve

# RAG dhe përgjegjësia (auditability)

Një avantazh kyç i RAG është **auditability**.

***Auditability është aftësia e një sistemi për të gjurmuar, inspektuar dhe justifikuar mënyrën se si është prodhuar një rezultat, duke identifikuar qartë burimet, hapat dhe vendimet që çuan në atë rezultat.***

Në kontekstin e RAG, auditi nënkupton që për çdo përgjigje të gjeneruar mund të dihet:

- cilat dokumente ose segmente janë rikthyer,
- si kanë ndikuar ato në përgjigje,
- dhe të rikonstruktohet procesi i gjenerimit për verifikim ose përgjegjësi institucionale.

Kjo lejon analiza post-hoc, përgjegjësi institucionale, dhe përdorim në domeine të rregulluara.

# RAG si problem sistemor, jo vetëm NLP

RAG nuk është vetëm problem NLP.

Ai përfshin:

- Information Retrieval
- Distributed Systems
- Monitoring dhe logging
- Prompt control

Performanca e RAG varet nga komponenti më i dobët, jo nga modeli më i fortë.

# Latency breakdown në RAG

***Latency në një sistem Retrieval-Augmented Generation përkufizohet si koha totale e kaluar midis momentit kur përdoruesi paraqet një pyetje dhe momentit kur sistemi kthen përgjigjen finale.***

Në sisteme reale, latenca ndahet në:

- embedimi i pyetjes
- kërkimi në vector DB
- gjenerimi i përgjigjes

# Versionimi i njohurive

Njohuritë ndryshojnë më shpesh se modelet.

RAG lejon:

- përditësim të dokumenteve
- pa ndryshuar modelin
- pa retraining

Por kërkon:

- menaxhim versionesh
- invalidim indeksesh
- kontroll konsistence

# RAG kundrejt alternativave

RAG duhet krahasuar me alternativa reale:

- MMGj pa retrieval (hallucination risk)
- Search-only systems (pa gjenerim)
- Fine-tuned models (njohuri statike)
- QA klasike

RAG ofron kompromisin më të mirë për fleksibilitet, kontroll, kosto.

# RAG vs MMGj (pa Retrieval)

- Modeli gjeneron përgjigje vetëm bazuar në njohurinë parametrike. Nuk përdor dokumente të j ashtme në kohë inferencë.

**Avantazhe**

- Arkitekturë e thjeshtë
- Latencë më e ulët
- Implementim i lehtë

**Kufizime**

- Rrezik i lartë i halucinimeve
- Pa burim ose gjurmueshmëri
- Njohuri statike dhe të paverifikueshme

**Krahasim me RAG:**

- RAG ul ndjeshëm hallucinations duke kushtëzuar gjenerimin me evidencë.
- RAG ofron auditability, që mungon plotësisht te LLM pa retrieval.

# RAG vs Search-only systems (pa gjenerim)

Kthejnë dokumente ose fragmente tekstuale, jo përgjigje të sintetizuara.

### Avantazhe

- Transparencë e lartë
- Pa hulucinime
- Kontroll i plotë mbi burimet

### Kufizime

- Nuk ofrojnë sintezë ose arsyetim
- Ngarkesë e lartë për përdoruesin
- Përvojë e dobët përdoruesi për pyetje komplekse

**Krahasim me RAG**

- RAG kombinon transparencën e kërkimit me aftësinë gjenerative të LLM.
- RAG ul ngarkesën kognitive duke sintetizuar informacionin.

# RAG vs Fine-tuned (pa Retrieval)

Modele të përshtatura me fine-tuning mbi të dhëna specifike. Njohuria inkorporohet në parametrat e modelit.

## Avantazhe

- Performancë e mirë në domain të kufizuar
- Gjenerim i qëndrueshëm dhe i kontrolluar
- Pa varësi nga infrastruktura e retrieval

## Kufizime

- Njohuri statike (kërkon retraining)
- Mungesë burimesh të qarta
- Kosto e lartë mirëmbajtje

**Krahasim me RAG**

- RAG ofron njohuri dinamike dhe të përditësueshme pa retraining.
- Fine-tuning është më i përshtatshëm për stil dhe sjellje, jo për njohuri faktike që ndryshojnë.

# RAG kundrejt sistemeve QA klasike

Sisteme që identifikojnë dhe nxjerrin (extract) një fragment ekzistues nga dokumentet. Përgjigja është zakonisht një nënvarg i tekstit burimor. Bazohet në modele si BERT për span extraction.

## Avantazhe

- Saktësi e lartë nëse përgjigja ekziston drejtpërdrejt në tekst.
- Transparencë e plotë
- Pa halucinime

## Kufizime

- Nuk odron sintezë apo arsyetim
- Dështon në pyetje abstrakte ose përmbledhese
- Varësi e fortë nga formulimi i saktë i pyetjes

Kjo e bën RAG më të fuqishëm për: pyetje komplekse, dokumente të gjata, dhe domene ku informacioni është i shpërndarë.

# RAG kundrejt sistemeve të tjera

| Arkitektura | Hallucinations | Sintesë | Transparencë | Fleksibilitet |
|---|---|---|---|---|
| QA Klasike | ❌ | ❌ | ✅ | ❌ |
| Search-only | ❌ | ❌ | ✅ | ⚠️ |
| LLM pa Retrieval | ❌❌ | ✅ | ❌ | ✅ |
| Fine-tuned LLM | ⚠️ | ✅ | ❌ | ⚠️ |
| RAG | ⚠️ Ulët | ✅ | ✅ | ✅ |

# RAG dhe vlerësimi shkencor

**Vlerësimi i retrieval**

- *Recall@k, Precision@k*: matin nëse dokumentet relevante rikthehen në top-k rezultate.
- *MRR (Mean Reciprocal Rank)*: mat sa shpejt shfaqet dokumenti korrekt në renditje.

**Vlerësimi i faithfulness (grounding)**

- Kontrollon nëse përgjigja e gjeneruar bazohet realisht në dokumentet e rikthyera.
- Përdoren annotime manuale ose metrika si *attribution accuracy*.

# RAG dhe vlerësimi shkencor

**Vlerësimi i saktësisë faktike**

- Krahasim i përgjigjes me burime të verifikuara (gold answers).
- Shpesh kërkon evaluim njerëzor në domene kritike.

**Vlerësimi end-to-end i detyrës**

- Mat nëse sistemi përmbush objektivin praktik (p.sh. përgjigje të sakta dhe të dobishme).
- Përfshin user studies ose task success rate.

# Kufizime strukturore të RAG

RAG nuk është pa kosto:

- Kërkon infrastrukturë shtesë
- Rrit latencën
- E bën vlerësimin më kompleks

- Në disa raste, një model i vogël fine-tuned mund të jetë më i përshtatshëm se një RAG.

- RAG nuk është zgjidhje universale. Si arkitekturë nuk garanton të vërtetën absolute, nuk zgjidh kontradikta dhe nuk zëvendëson logjikën formale.

# Çfarë do të trajtojmë në vazhdimësi?

- ✓ Hyrje
- ✓ Tokenizimi & Embeddings
- ✓ Arkitektura Transformer
- ✓ Bazat e të dhënave vektoriale (Vector Databases)
- ✓ RAG (Gjenerim i përforcuar nga kërkimi)
6. Promptimi
7. Agjentët (Agents)
8. Përshtatja e modelit (Fine-Tuning)
9. Vlerësimi (Evaluation)
10. Siguria & Përafrimi (Safety & Alignment)
11. Vendosja e modelit (Deployment)
12. Integrimi i Projektit Final

# Bibliografia


- [1] P. Lewis *et al.*, “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” *Proc. NeurIPS*, 2020.
  [2] S. Karpukhin *et al.*, “Dense Passage Retrieval for Open-Domain Question Answering,” *Proc. EMNLP*, 2020.
  [3] N. Reimers and I. Gurevych, “Sentence-BERT,” *Proc. EMNLP*, 2019.
  [4] J. Johnson, M. Douze, and H. Jégou, “Billion-Scale Similarity Search with GPUs,” *IEEE Trans. Big Data*, 2021.
  [5] J. Gao *et al.*, “Rethinking Search: Making Domain Experts out of Large Language Models,” *arXiv*, 2023.