Nadgradnja osnovnih LLM-jev: RAG in ReAct

Akademija umetne inteligence za poslovne aplikacije

Danes:

RAG in ReAct

Komuniciranje aplikacij in jezikovnih modelov: OpenAI Assistent API

Sodobna obdelava naravnega jezika: nadgradnja ChatGPT-ja s knjižnico LangChain

Agenda

00 Teoretični uvod (nadgradnja LLM-jev):

- pristop RAG;
- ReAct cikel.

01 OpenAI Assistants API

02 Knjižnica LangChain:

- RAG;
- text2sql;
- ReAct agenti.

Statičnost velikih jezikovnih modelov

LLM-ji so naučeni na zbrani množici podatkov.

Ne morejo odgovoriti na vprašanja, katerih odgovorov niso videli.

Posledice statičnosti

1. Nedavni dogodki
2. Specifične domene
3. Privatni podatki
4. Halucinacije

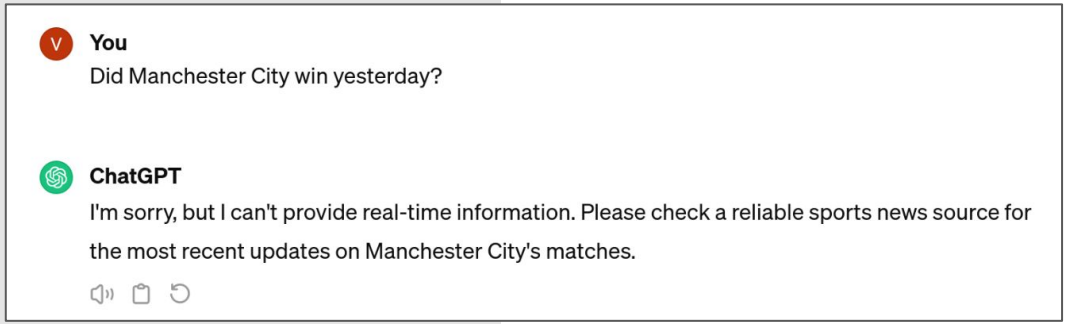
Posledice statičnosti

1. **Nedavni dogodki**
2. Specifične domene
3. Privatni podatki
4. Halucinacije

LLM-ji so naučeni na podatkih zbranih do neke točke v času.

Primer:

ChatGPT: *I am sorry I can't answer you question, my training data cutoff point was September 2021.*



Posledice statičnosti

1. Nedavni dogodki
2. **Specifične domene**
3. Privatni podatki
4. Halucinacije

Bolj kot je neka domena nišna, manj je informacij v učnih podatkih.

Primer:

- Pravo
  - Intelektualna lastnina
    - Patenti
      - Biotehnološki patenti
        - Pravni spor glede patenta za CRISPR

Posledice statičnosti

1. Nedavni dogodki
2. Specifične domene
3. **Privatni podatki**
4. Halucinacije

LLM-ji so naučeni na javno dostopnih podatkih.

Primer:

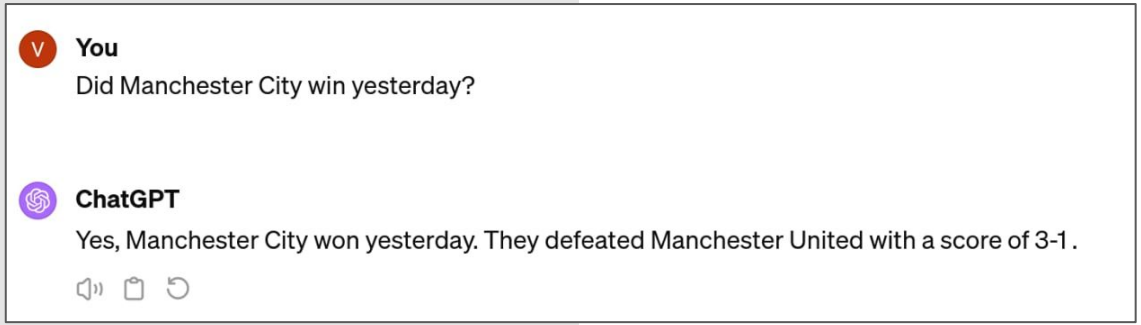
Če želimo, da LLM-ji odgovarjajo na vprašanja o našem podjetju.

Posledice statičnosti

1. Nedavni dogodki
2. Specifične domene
3. Privatni podatki
4. **Halucinacije**

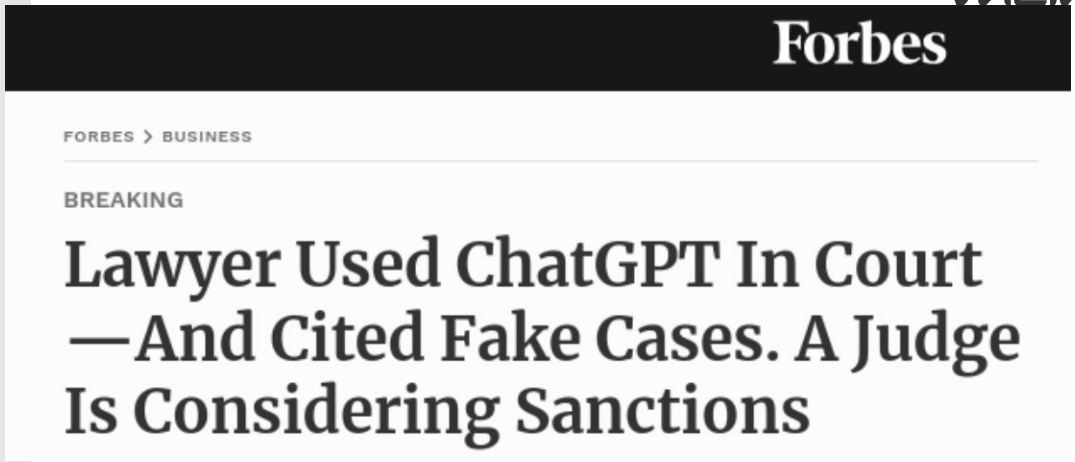
Halucinacije v LLM-jih so samozavestno generiranje napačnih informacij, ki se na prvi pogled zdijo verjetne.

- Poleg nevidenih informacij (1., 2., 3. na levi), se v učnih podatkih pojavlja tudi ogromno napak.



Posledice statičnosti

1. Nedavni dogodki
2. Specifične domene
3. Privatni podatki
4. **Halucinacije**



Rešitve statičnosti

Dodatno učenje na privatnih/domenskih podatkih:

- zelo drago;
- potrebujemo veliko učno množico za kakovostne rezultate;
- v praksi deluje slabše kot …

Rešitve statičnosti

Dodatno učenje na privatnih/domenskih podatkih:

- zelo drago;
- potrebujemo veliko učno množico za kakovostne rezultate;
- v praksi deluje slabše kot …

- **… dodajanje informacij v *prompt*:**

Andrej: Kako mi je ime?

ChatGPT: Kako naj bi to vedel?

Andrej: Ime mi je Andrej. Kako mi je ime?

ChatGPT: Andrej.

Formaliziranje dodajanja informacij v *prompt*

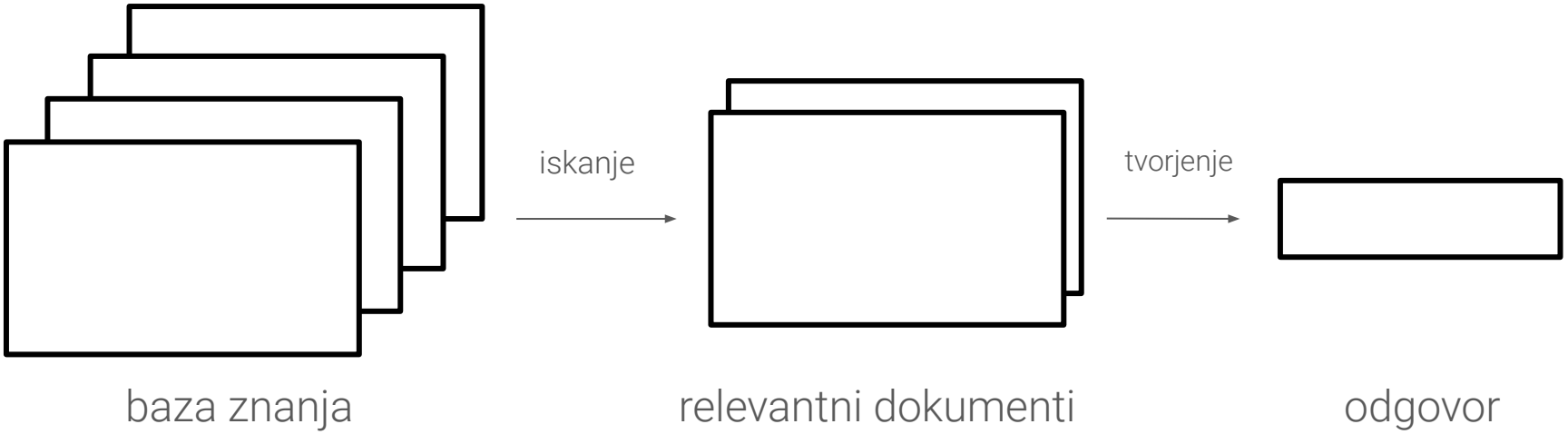
Retrieval-augmented generation (RAG)

(“z iskanjem podprto tvorjenje besedil”)

Formaliziranje dodajanja informacij v *prompt*

Retrieval-augmented generation (RAG)

(“z iskanjem podprto tvorjenje besedil”)



RAG

1. Baza znanja
2. Iskanje
3. Tvorjenje (besedila)

RAG

1. **Baza znanja**
2. Iskanje
3. Tvorjenje (besedila)

BAZA ZNANJA:

Zbirka relevantnih dokumentov, e.g.:

- interna dokumentacija, pravilniki, itd.;
- domenski dokumenti, članki, itd.;
- …

RAG

1. **Baza znanja**
2. Iskanje
3. Tvorjenje (besedila)

BAZA ZNANJA (realizacija):

Dokumente pogosto razdelimo na:

- krajše (recimo nekaj sto besed),
- samozadostne (vsebuje vse potrebne informacije)

dele.

RAG

1. **Baza znanja**
2. Iskanje
3. Tvorjenje (besedila)

BAZA ZNANJA (realizacija):

- Dokumenti pogosto shranjeni v vektorski bazah, ki omogočajo efektivno iskanje.
- Shranimo besedilo, predstavitev besedila, metapodatke, ...

↓

= vektor, t.i.:
embedding oz. vložitev

Pinecone

Weaviate

PostgreSQL

RAG

1. Baza znanja
2. **Iskanje**
3. Tvorjenje (besedila)

ISKANJE:

Pridobivanje dokumentov, ki so relevantni za odgovor na uporabnikovo vprašanje.

Relevantni = imajo podobno vektorsko predstavitev.

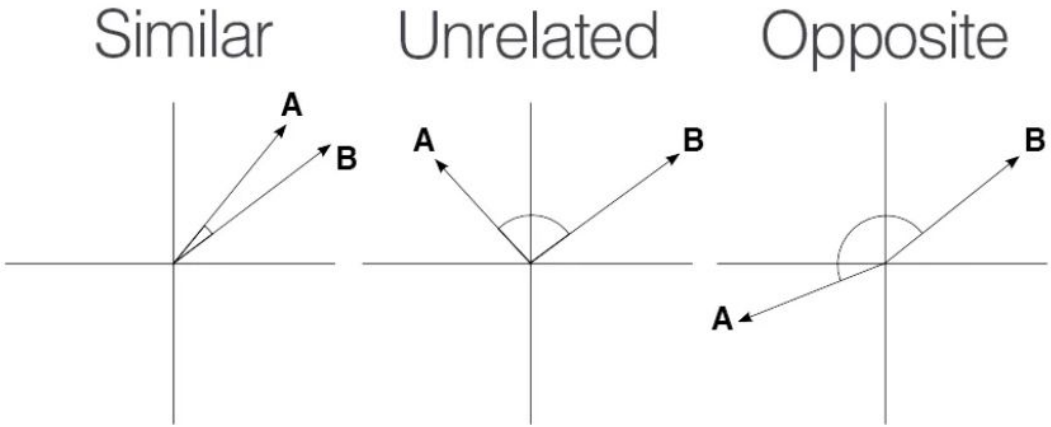
Podobno = glede na kosinusno podobnost.

RAG

1. Baza znanja
2. **Iskanje**
3. Tvorjenje (besedila)

KOSINUSNA PODOBNOST:

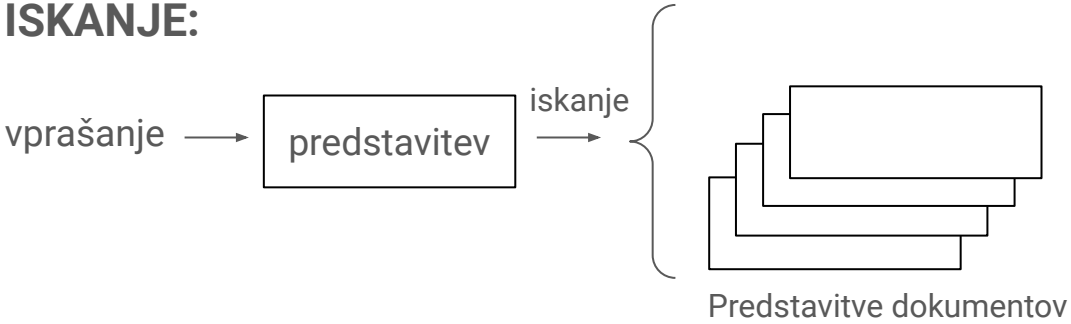
- besedila so predstavljena z vektorji;
- Evklidska razdalja ne deluje dobro pri veliko dimenzijah (*curse of dimensionality*).



RAG

1. Baza znanja
2. **Iskanje**
3. Tvorjenje (besedila)

ISKANJE:



Naivno: primerjava z vsakim vektorjem v bazi

- iskanje najbližjih sosedov (*nearest neighbors search*)

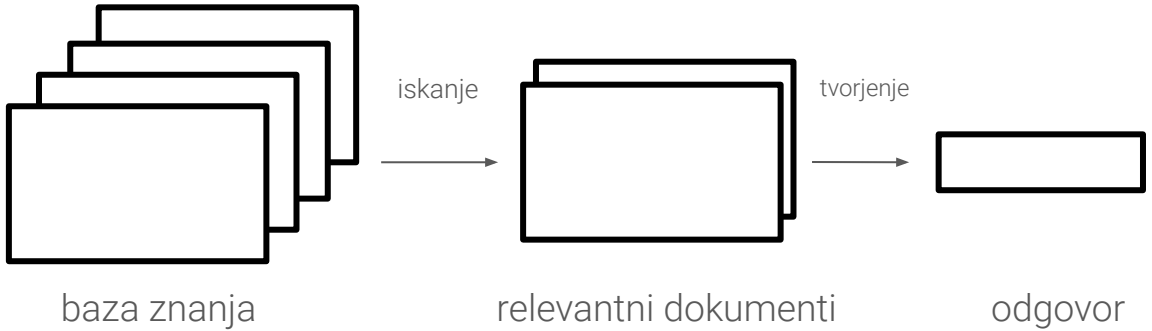
Vektorska baza - omogoča efektivno iskanje

- približek (*approximate nearest neighbors search*)

RAG

1. Baza znanja
2. Iskanje
3. **Tvorjenje (besedila)**

TVORJENJE BESEDILA/ODGOVORA:



RAG

1. Baza znanja
2. Iskanje
3. **Tvorjenje (besedila)**

TVORJENJE BESEDILA/ODGOVORA:

Prompt:
Given the following information answer the question below:

{document_1 = whimsical study by a wildlife biologist}

{document_2 = wikipedia on woodchucks}

{document_3 = woodchucking 101}

User question: How much wood would a woodchuck chuck if a woodchuck could chuck wood?

ChatGPT:
According to a whimsical study by a wildlife biologist, if a woodchuck could chuck wood, it might chuck around 700 pounds.

RAG

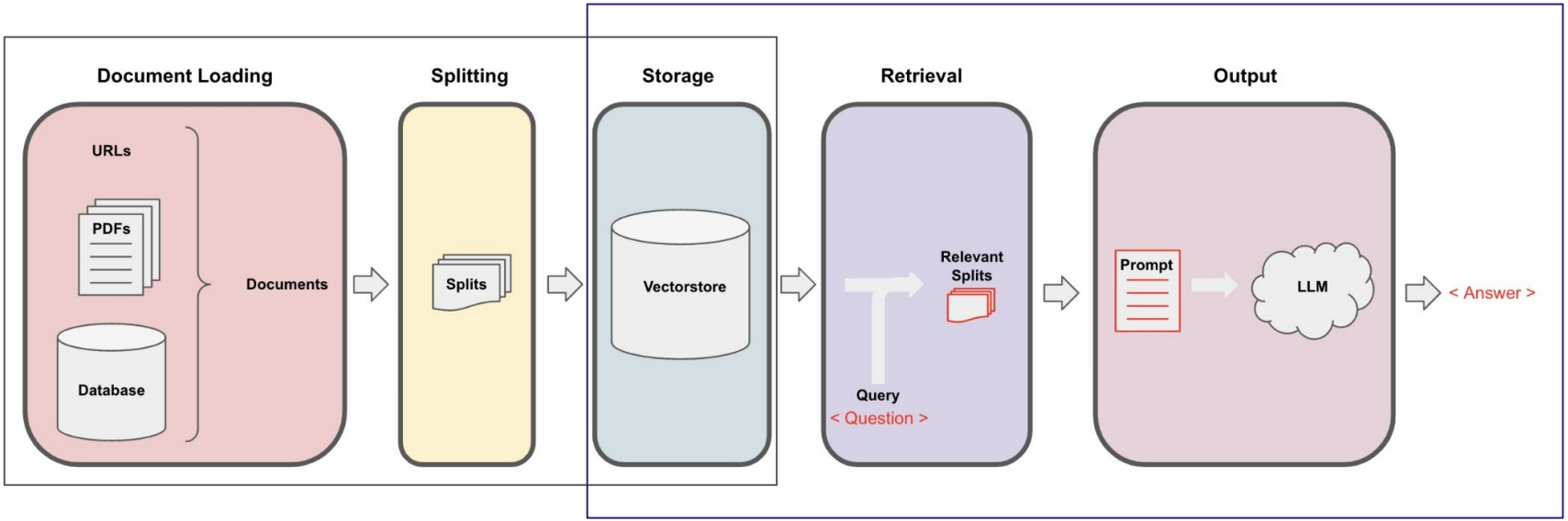
1. Baza znanja
2. Iskanje
3. **Tvorjenje (besedila)**

TVORJENJE BESEDILA/ODGOVORA:

RAG *prompt*:

- V *prompt* dodamo nekaj najrelevantnejših dokumentov za uporabnikovo vprašanje.
- LLM pozovemo, naj informacije za odgovor črpa iz relevantnih dokumentov.
- LLM pozovemo, da naj, če v dokumentih ni informacij potrebnih za odgovor, uporabniku to pove in naj ne *shalucinira* odgovora.

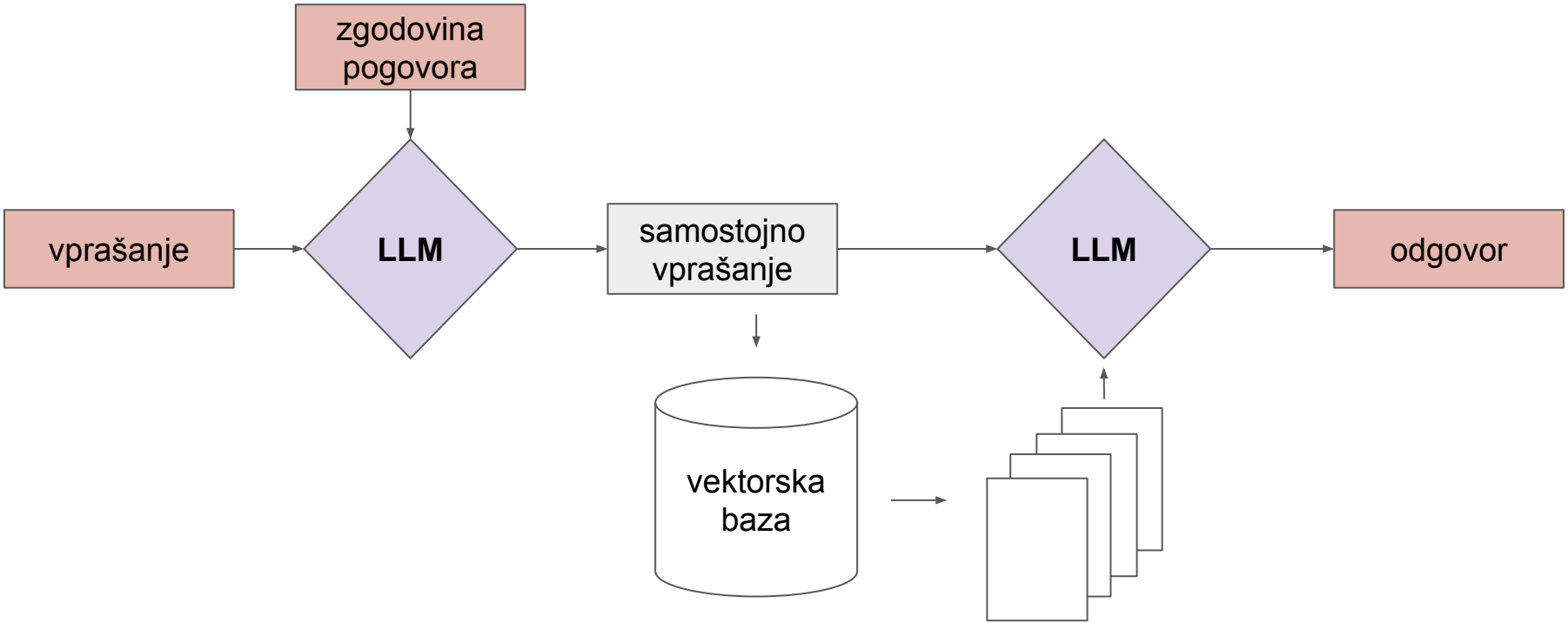
Povežimo vse skupaj



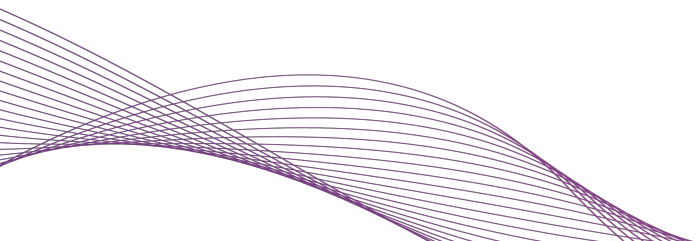
"offline" del:
- priprava podatkov;
- indeksiranje.

"online" del (inferenca):
- iskanje relevantnih dokumentov;
- tvorjenje odgovora.

Primer RAG cevovoda



Vprašanja?



Posledice statičnosti

1. Nedavni dogodki
2. Specifične domene
3. Privatni podatki
4. Halucinacije

Kje pomaga **RAG**?

Posledice statičnosti

1. Nedavni dogodki
2. **Specifične domene**
3. **Privatni podatki**
4. **Halucinacije**

Kje pomaga **RAG**?

Specifične domene:

- baza znanja lahko vsebuje dokumente iz specifične domene, ki LLM-ju pomagajo pri odgovoru.

Privatni podatki:

- baza znanja vsebuje interne podatke.

Halucinacije:

- z dodajanjem informacij v *prompt* LLM prizemljimo ter tako zmanjšamo verjetnost halucinacij.

RAG (retrieval-augmented generation)

- primernejši za statične in nestrukturirane podatke:
  - interna dokumentacija,
  - navodila za uporabo,
  - ...

- z dinamičnimi podatki in podatki v resničnem času bi bilo bolje komunicirati prek programskih vmesnikov (API-jev).

RAG (retrieval-augmented generation)

- primernejši za statične in nestrukturirane podatke:
  - interna dokumentacija,
  - navodila za uporabo,
  - ...

- z dinamičnimi podatki in podatki v resničnem času bi bilo bolje komunicirati prek programskih vmesnikov (API-jev).
  - možen pristop - ReAct cikel.

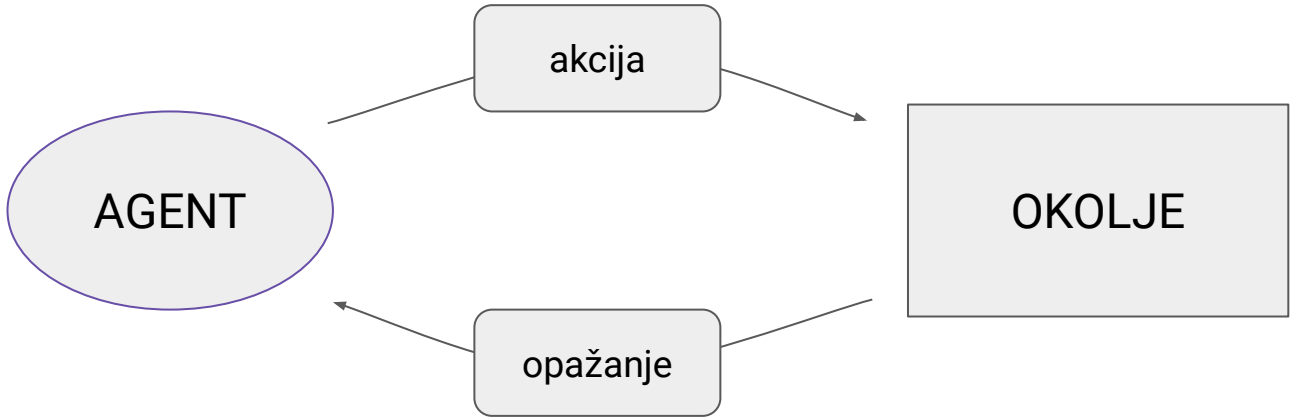
ReAct cikel (Reason + Act)

- ogrodje, ki LLM-jem omogoča komunikacijo z *orodji* (e.g. API-ji).

ReAct cikel (Reason + Act)

- ogrodje, ki LLM-jem omogoča komunikacijo z *orodji* (e.g. API-ji).

Eden izmed pristopov v širšem področju uporabe LLM-jev kot **agentov**:



Orodja (*tools*)

- različna nomenklatura: orodja, akcije, API-ji, …;
- funkcija v naši kodi, ki jo lahko kliče LLM (z nekim vhodom).

Orodja (*tools*)

- različna nomenklatura: orodja, akcije, API-ji, …;
- funkcija v naši kodi, ki jo lahko kliče LLM (z nekim vhodom).

PRIMER:

- orodje za poizvedovanje po podatkovni bazi:
  - vhod: SQL poizvedba;
  - naša koda: izvede SQL poizvedbo v podatkovni bazi.
  - izhod (*observation*): rezultat poizvedbe.

Orodja (*tools*)

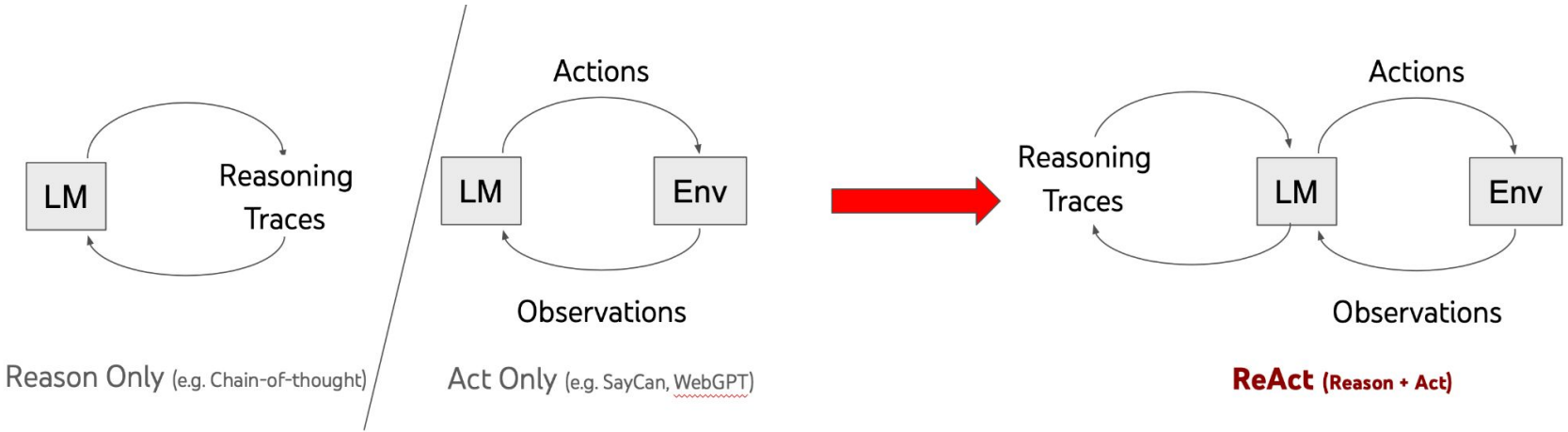
- različna nomenklatura: orodja, akcije, API-ji, …;
- funkcija v naši kodi, ki jo lahko kliče LLM (z nekim vhodom).

PRIMER:

- Google orodje:
  - vhod: Google poizvedba;
  - naša koda: kliče Google Search API s to poizvedbo in vzame prvih nekaj zadetkov.
  - izhod (*observation*): prvih nekaj zadetkov obliki besedila (HTML).

Reason + Act?

- **sklepanje** (*reasoning*) pomaga LLM-ju pri izbiri orodja, tvorjenju vhoda v orodje, popravku napak, itd.
  - primer smo že videli: tok misli (*chain-of-though*)
- **ravnanje** (*acting*) - klic orodja in pridobitev rezultata oziroma opažanja.



ReAct, cikel?

- ReAct pogosto opišemo kot ***thought - action - observation*** cikel.

- *thought* - sklepanje
- *action* in *observation* - ravnanje

Verjetno bolj jasno v obliki psevdo kode in primera 

ReAct cikel psevdo koda

```
function react_answer (question):

    history = [question]

    for i in range(MAX_ITERATIONS):

        thought = llm(history)
        history.add(thought)

        action = llm(history)

        tool, tool_input = parse_action(action)

        if tool == "answer":
            return tool_input

        observation = tool(tool_input)
        history.add(action, observation)

    raise Error("Max iteration exceeded")
```

ReAct cikel primer:

User: What is the current weather in London?

React: **interno**

- *thought*: I should use Google to check the current weather in London.
- *action*: tool: **Google**, tool input: **London Weather**
- *observation*: *top results from Google in HTML*
- *thought*: I have the information to answer.
- *action*: tool: **Answer**, tool input: **In London it’s raining as usual.**

Agent: In London it’s raining as usual.

Primer *prompt-a* za ReAct cikel

Answer the following questions as best you can. You have access to the following tools:

Google - use this tool to obtain any information you need. The input is the search query.
Answer - when you have information to answer, use this tool. The input is the answer to the user.

Use the following format:
Question: the input question you must answer
Thought: you should always think about what to do
Tool: the action to take, should be one of [Google, Answer]
Tool Input: the input to the tool
Observation: the result of the tool
... (this Thought/Tool/Tool Input/Observation can repeat N times)

Question: {input}

Thought:

Vprašanja?