Projekat 2: Kreiranje sistema agenata

Todor Marković

Računarski fakultet Univerziteta Union u Beogradu

Duboko učenje

Arhitektura

Sistem je dizajniran kao **modularna Python aplikacija** koja upravlja interakcijom između korisnika, lokalnog LLM-a i spoljnih alata. Arhitektura prati **linearnu primopredaju**, gde se podaci sekvencijalno obrađuju kroz niz specijalizovanih agenata.

Sistem se sastoji iz tri ključna sloja:

- **Sloj zaključivanja:** Odgovoran za generisanje teksta (Llama 3.2 model).
- **Sloj agenata:** Logika odlučivanja, upravljanje memorijom i izvršavanje alata.
- **Sloj alata:** Interfejsi ka spoljnom svetu (Arxiv API, funkcije za prenos podataka).

Analiza toka izvršavanja

U ovom poglavlju detaljno prikazujemo putanju jednog korisničkog zahteva kroz arhitekturu sistema.

Kao primer uzimamo upit:

"Pronađi radove o 'AI' i analiziraj ih."

Proces obrade se odvija kroz 5 sekvencijalnih faza, u fajlu main.py i klasi Agent (agenti.py).

Faza 1: Inicijalizacija i Konstrukcija Prompta

Korisnički upit ulazi u sistem kroz funkciju pokreni_sistem (main.py). Prvi agent, **Istraživač**, prima upit.

- **Kod:** agent_istrazivac.run(korisnicki_ulaz)
- **Proces:** Metoda dodaj_u_memoriju formira strukturu razgovora. Tokenizer (ucitavanje.py) konvertuje sistemski prompt (definisane uloge i pravila) i korisnički upit u niz tokena koje Llama-3.2 model razume.
- **Stanje memorije:**

  JSON

```
[
 {"role": "system", "content": "You are the RESEARCHER AGENT..."},
 {"role": "user", "content": "Pronađi radove o 'AI'..."}
]
```

Faza 2: LLM i Odluka o Alatu

Metoda _pozovi_llm šalje formatiranu istoriju razgovora modelu.

- **Model:** Llama-3.2-3B-Instruct (učitana via llama-cpp-python).
- **Odluka:** Na osnovu sistemskog prompta koji nalaže korišćenje alata, model ne generiše tekst odgovora, već generiše specifičan token za poziv funkcije.
- **Sirovi izlaz modela:** TOOL_CALL: alat_pretrazi_arxiv|AI
- **Detekcija:** while True petlja u agenti.py detektuje ključnu reč TOOL_CALL i pauzira generisanje teksta da bi izvršili Python kod.

Faza 3: Izvršavanje Alata (Python Runtime)

Kontrola prelazi sa LLM-a na Python interpreter (pretraga.py).

- **Parsiranje:** Funkcija izvrsi_alat razdvaja ime alata (alat_pretrazi_arxiv) od argumenta (AI).
- **Eksterna Akcija:** Skripta poziva arxiv API, preuzima metapodatke stvarnih naučnih radova i formatira ih u string.
- **Povratna sprega:** Rezultat pretrage se ne prikazuje korisniku, već se vraća u memoriju agenta kao "user" poruka (simulirajući sistemski odgovor).

Faza 4: Lančana Reakcija (Multi-Agent Handoff)

Istraživač sada u memoriji ima i zahtev i podatke. Njegov sistemski prompt mu nalaže da podatke ne zadržava, već da ih prosledi.

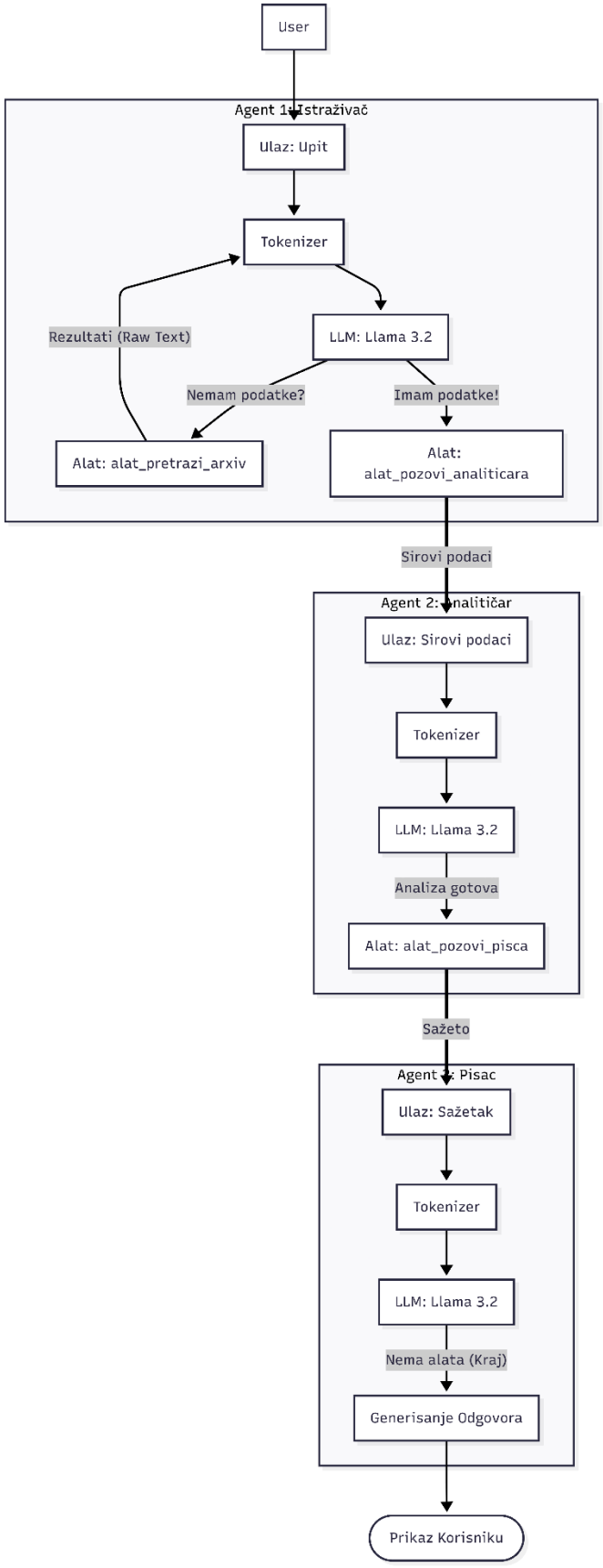
- **Nova Odluka:** Model ponovo vrši inferenciju nad novim stanjem memorije.
- **Izlaz:** TOOL_CALL: alat_pozovi_analiticara|<sirovi_tekst_radova>

- **Tranzicija:** Funkcija pozovi_analiticara u main.py instancira novog agenta (**Analitičar**) i prosleđuje mu sirove podatke kao ulaz.

Faza 5: Transformacija i Finalizacija

Proces se ponavlja za Analitičara i Pisca, ali sa različitim kognitivnim zadacima definisanim u njihovim promptovima:

1. **Analitičar:** Prima sirov tekst -> Vrši ekstrakciju (filtriranje) -> Poziva Pisca.
2. **Pisac:** Prima strukturiranu analizu -> Generiše finalni odgovor.
3. **Kraj:** Kada model više ne generiše TOOL_CALL, metoda run vraća finalni string korisniku.


User
Agent 1: Istraživač
Ulaz: Upit
Tokenizer
LLM: Llama 3.2
Rezultati (Raw Text)
Nemam podatke?
Imam podatke!
Alat: alat_pretrazi_arxiv
Alat: alat_pozovi_analiticara
Sirovi podaci
Agent 2: Analitičar
Ulaz: Sirovi podaci
Tokenizer
LLM: Llama 3.2
Analiza gotova
Alat: alat_pozovi_pisca
Sažeto
Agent 3: Pisac
Ulaz: Sažetak
Tokenizer
LLM: Llama 3.2
Nema alata (Kraj)
Generisanje Odgovora
Prikaz Korisniku

Teorijske pretpostavke

Ovaj projekat se zasniva na primeni lokalnih Velikih Jezičkih Modela (LLM) u multi-agentskom okruženju. Arhitektura sistema nije nasumična, već je izvedena iz nekoliko ključnih pretpostavki o tome kako savremeni generativni modeli procesiraju informacije, izvršavaju instrukcije i koriste alate. U nastavku su definisane tri ključne hipoteze na kojima počiva dizajn ovog sistema.

1. Hipoteza o specijalizaciji (Role-Based Prompting)

Pretpostavka: Jedan generalni LLM (npr. Llama 3.2 3B) je podložan "zaboravljanju" instrukcija i halucinacijama kada mu se zada previše kompleksan zadatak odjednom. Međutim, sposobnost modela drastično raste kada se "uokviri" (frame) u specifičnu, usku ulogu.

2. Hipoteza o upravljanju kontekstom (Context Management)

Pretpostavka: Kvalitet izlaza modela je direktno proporcionalan kvalitetu i formatu ulaznih podataka (Prompt Engineering). Model ne "razume" tekst kao čovek, već predviđa sledeći token na osnovu statističke verovatnoće. Ako format prompta (System/User/Assistant tagovi) nije matematički precizan, verovatnoća tačnog odgovora opada.

3. Hipoteza o nelinearnoj degradaciji i pragu kolapsa

Degradacija performansi manjih LLM modela (npr. 3B parametara) usled povećanja kompleksnosti prompta nije linearna. Modeli održavaju visoke performanse do određene tačke kognitivnog zasićenja, nakon čega dolazi do naglog strukturnog kolapsa ('kognitivna litica'). Primena Prompt Fine-Tuninga ne eliminiše ovu liticu, već značajno pomerera prag tolerancije na viši nivo kompleksnosti pre nego što do kolapsa dođe.

Eksperimenti

1. Hipoteza o specijalizacij

Dokaz hipoteze o specijalizaciji ćemo razložiti na tri zasebne hipoteze:

H1 — Uticaj kompleksnosti prompta

Povećanje strukturne i kognitivne kompleksnosti prompta dovodi do statistički značajnog smanjenja uspešnosti izvršavanja zadatka od strane agenta *agent_analiticar-a*.

H2 — Efekat prompt fine-tuninga

Primena prompt fine-tuninga u vidu jasne fazne strukture, eksplicitnih radnih pravila i mehanizama samoprovere dovodi do statistički značajnog poboljšanja performansi agenta *agent_analiticar* pri istoj kompleksnosti prompta.

H3 — Stabilnost i determinističnost ponašanja

Prompt fine-tuning smanjuje varijansu izlaznih odgovora agenta *agent_analiticar* pri ponovljenim izvršavanjima istog zadatka, čime se postiže stabilnije i determinističnije ponašanje.

1.2. Operacionalizacija kompleksnosti prompta

Kompleksnost prompta definiše se kao višedimenzionalna veličina sastavljena od sledećih komponenti:

- **Strukturna kompleksnost** – broj faza i zavisnosti između faza zadatka
- **Kognitivna kompleksnost** – zahtev za formalnim rezonovanjem I algoritamskom konstrukcijom
- **Kompleksnost ograničenja** – broj i strogoća eksplicitno navedenih pravila i zabrana

- **Kompleksnost izlaza** – potreba za generisanjem više tipova strukturiranih izlaza (tekst, kod, formalne specifikacije)

Na osnovu navedenih dimenzija definišu se tri nivoa prompta: nizak (L1), srednji (L2) i visok (L3) nivo kompleksnosti.

1.3. Kriterijumi evaluacije

Evaluacija performansi agenta *agent_analiticar* vrši se na osnovu sledećih merljivih kriterijuma:

- **Stopa uspešnosti zadatka (Task Success Rate)**
- **Broj prekršenih ograničenja (Violation Count)**
- **Strukturalni Integritet**
- **Odsustvo Halucinacija**
- **Kognitivno Opterećenje**

Upit

"Title: Federated Learning: Strategies for Improving Communication Efficiency. Abstract: Federated Learning (FL) enables model training on decentralized data. However, communication costs remain a bottleneck. We propose a new algorithm, 'FedCom', which uses gradient compression and quantization to reduce data transfer by 40%. The method was tested on MNIST and CIFAR-10 datasets. Results show 99% accuracy retention compared to full-precision transmission. However, non-IID data distribution remains a challenge for convergence speed."

Kriterijum Evaluacije	L1	L2	L3	L3 NAIVE	L3 FINE-TUNED
Uspešnost Zadatka	Visoka	Visoka	Srednja (Format greska)	Niska(kolaps)	Visoka

Strukturalni Integritet	N/A	N/A	Pao (Imao je uvodni tekst)	**Katastrofalan** (Više blokova, tekst)	Savršen
Odsustvo Halucinacija	N/A	Visoko	Pao	Pao	Visoko
Kognitivno Opterećenje (Kvalitet jezika)	Nisko	Srednje	Visoko	Veoma visoko	Visoko (Vidljiv napor, ali stabilan)

Zaključak analize performansi

Komparativna analiza prikazana u tabeli nedvosmisleno potvrđuje da performanse LLM agenta (na manjem modelu od 3B parametara) nisu linearne, već zavise od interakcije između **kompleksnosti zadatka** i **kvaliteta instrukcija (prompta)**.

Iz tabele se izvode tri ključna uvida:

1. Potvrda H1: Granica kognitivnog kapaciteta Poređenjem kolona **L1/L2** sa kolonom **L3 (Standard/Naive)**, uočava se jasan trend degradacije performansi.

- Dok model na nivoima L1 i L2 postiže ocenu **"Visoka"** uz nisko do srednje kognitivno opterećenje, uvođenje strogih strukturnih zahteva (JSON format, zabrane) na nivou L3 dovodi do pada uspešnosti na **"Srednja"** (L3) ili **"Niska/Kolaps"** (L3 Naive).
- Ovo dokazuje da puka kompleksnost zahteva može dovesti do "kognitivnog zasićenja" modela, rezultirajući strukturnim raspadom (curenje uvodnog teksta, višestruki blokovi).

2. Potvrda H2: Moć Prompt Fine-Tuninga Najznačajniji nalaz je kontrast između kolona **L3 NAIVE** i **L3 FINE-TUNED**. Iako je nivo kompleksnosti zadatka identičan, rezultati su dijametralno suprotni:

- **L3 NAIVE** pokazuje "Katastrofalan" strukturni integritet i prisustvo halucinacija.
- **L3 FINE-TUNED** vraća uspešnost na nivo **"Visoka"**, sa ocenom **"Savršen"** za strukturni integritet i eliminisanim halucinacijama. Ovo dokazuje da **Prompt Fine-Tuning** (kroz *Chain-of-Thought* i jasnu faznu strukturu) deluje kao stabilizator koji omogućava modelu da prevaziđe svoja prirodna ograničenja.

2. Odnos napora i stabilnosti Poslednji red tabele ("Kognitivno opterećenje") otkriva kvalitativnu razliku: kod L3 Fine-Tuned modela primećen je **"Vidljiv napor, ali stabilan"** rad. To ukazuje da optimizovan prompt ne smanjuje težinu zadatka, već efikasnije usmerava "pažnju" modela, sprečavajući haos (kolaps) koji se vidi kod Naive pristupa.

2. Hipoteza o upravljanju kontekstom

Da bih dokazao pretpostavku da kvalitet prompta direktno utiče na kvalitet izlaza, prilagodio sam funkciju **_pozovi_llm** na dva načina. U prvom slučaju korišćen je tokenizator sa tačnim tagovima za Llama 3 model. U drugom slučaju namerno su korišćeni pogrešni tagovi kao što su [INST] i <</SYS>>, kako bi se izazvala degradacija performansi.

Upit

“Find papers about 'Ethical AI'. Analyze them and extract key risks.CRITICAL OUTPUT RULES:

1. You MUST output the result strictly as a JSON object.
2. The JSON keys must be: "naslov_rada", "rizik_opis", "godina".
3. The content must be in Serbian language.
4. .DO NOT use the word "inteligencija" in your text. Use "razum" instead.”

Kriterijum evaluacije	Sistem A (Bez Tokenizera)	Sistem B (Sa Tokenizerom)
Sintaksička ispravnost	**Greška:** Curenje sistemskih tokena u odgovor ([OUT], <</SYS>>, [INST]).	**Stabilna:** Čist prirodni jezik, bez curenja sistemskih tokena.
Verodostojnost	**Halucinacija:** Navodi naučne radove iz budućnosti (godina 2026).	**Generička:** Navodi opšte informacije, ali bez izmišljanja nemogućih datuma.
Poštovanje formata	**Raspad:** Nije uspeo da formira JSON, generisao je nevalidan kod.	**Odbijanje:** Ignorisao JSON zahtev, ali zadržao koherentnu strukturu eseja.
Kontrola toka	**Beskonačna petlja:** Model nije generisao EOS token, morao je biti nasilno prekinut.	**Kontrolisan kraj:** Model je završio odgovor i vratio kontrolu programu.
Jezik	**Mešovito:** Počeo na engleskom, uprkos instrukciji za srpski.	**Srpski:** Poštovao je jezičko ograničenje.

*Napomena: Problem formatiranja u JSON objekat. Model sa tokenizerom je ispoštovao kada je piscu to prosledjeno kao pravilo u promptu

Zaključak: Eksperiment nedvosmisleno potvrđuje da je precizno upravljanje kontekstom (korišćenje AutoTokenizer-a) preduslov za funkcionalan sistem. Analiza je pokazala tri ključne stvari:

1. **Eliminacija teških halucinacija:** Loš format prompta dovodi do potpunog gubitka veze sa realnošću (izmišljanje radova iz 2026. godine), dok ispravan format drži model u okvirima stvarnih činjenica.
2. **Razlika između kvara i neposlušnosti:** Tokenizer osigurava da se sistem **ne sruši** i da poštuje jezik (srpski), dok je za striktno poštovanje formata (JSON) potrebno dodatno preciziranje sistemske uloge agenta, a ne samo tehničko formatiranje.

3. Hipoteza o nelinearnoj degradaciji i pragu kolapsa

Nelinarnu degradaciju smo dokazali korišćenjem proste python skripte. Skripta korišćenjem deset iteracija gde se svaki put kreirea novi agent omogućava da svaki test ponovo počne od nule.

Upit

"Title: Federated Learning: Strategies for Improving Communication Efficiency. Abstract: Federated Learning (FL) enables model training on decentralized data. However, communication costs remain a bottleneck. We propose a new algorithm, 'FedCom', which uses gradient compression and quantization to reduce data transfer by 40%. The method was tested on MNIST and CIFAR-10 datasets. Results show 99% accuracy retention compared to full-precision transmission. However, non-IID data distribution remains a challenge for convergence speed."

Ocenjivanje

Nakon vraćanja odgovora modela proveravaju se tri kriterijuma:

1. Da li je odgovor u JSON formatu

json.loads(clean_odg)

Proverava da li je odgovor validan JSON format. Ako pukne parsiranje, test pada (**is_json = False**).

2. Zabranjena reč

if "problem" in odgovor.lower():

Proverava da li je model iskoristio zabranjenu reč "problem". Ako jeste, pada (**no_forbidden = False**).

3. Halucinacije

if "import torch" in odgovor

Proverava da li je model izmislio Python kod koji ne postoji u tekstu. Ako nađe kod, test pada (**no_hallucination = False**).

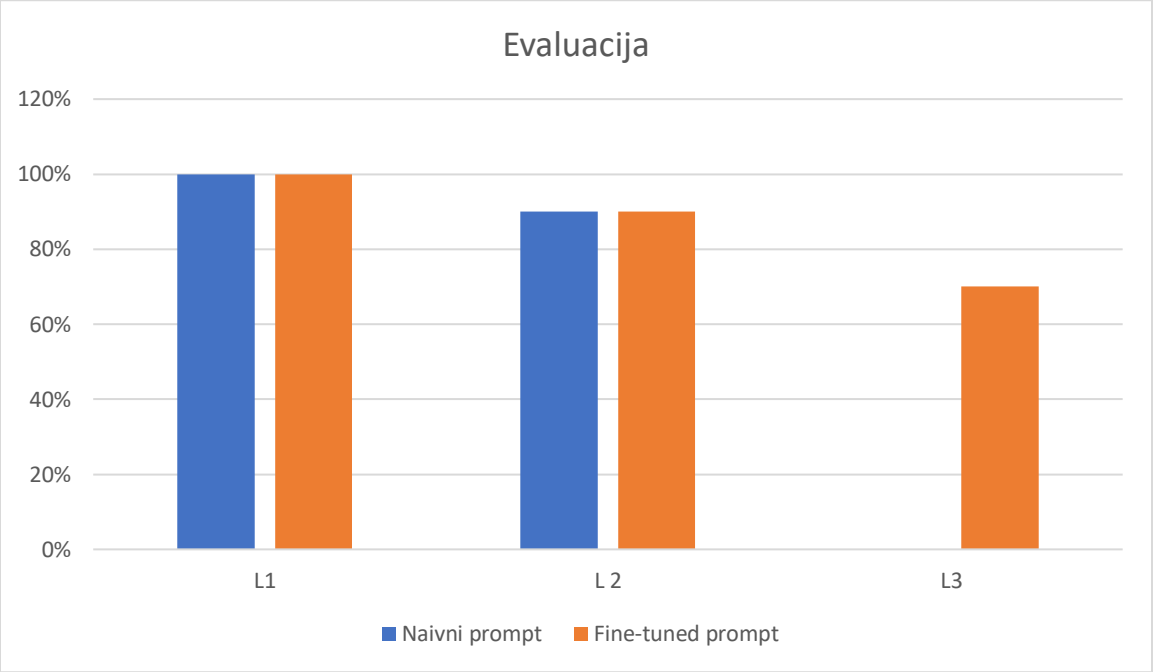
4. Bodovanje

if is_json and no_forbidden and no_hallucination:

uspesni += 1

Iteracija se računa kao uspešna samo ako su **SVA TRI** uslova ispunjena. Ako je JSON dobar, ali ima zabranjenu reč -> **NULA bodova**.

Nivo Kompleksnosti	Naivni Prompt (Plava Linija)	Fine-Tuned Prompt (Narandzasta Linija)
L1 (Niska)	100%	100%
L2 (Srednja)	90%	90%
L3 (Visoka)	**0%** (KOLAPS)	**70%** (USPEH)



Grafikon 'Evaluacija' vizuelno demonstrira fenomen kognitivne litice definisan u hipotezi.

- **Stabilna zona (L1 i L2):** Na nivoima niske i srednje kompleksnosti, oba pristupa (Naivni i Fine-tuned) pokazuju visoke performanse (90-100%). Model uspešno savladava zadatke jer kognitivno opterećenje ne prelazi njegov 'prozor pažnje'.
- **Tačka kolapsa (L3 - Naivni prompt):** Kod plavog stuba na nivou L3 vidimo nagli pad na **0%**. Ovo nije postepena degradacija, već potpuni funkcionalni otkaz. Model nije uspeo da generiše validan JSON ni u jednoj iteraciji, što ukazuje da je kompleksnost zadatka probila prag tolerancije modela bez strukturne podrške.
- **Efekat stabilizacije (L3 - Fine-tuned prompt):** Narandžasti stub na nivou L3 (**70%**) pokazuje efekat inženjeringa prompta. Iako zadatak ostaje isti, uvođenje *Chain-of-Thought* mehanizma i jasnih uloga sprečava kolaps, vraćajući sistem u operativno stanje."

2. Korelacija sa kvalitativnom tabelom (Povezivanje dve slike)

Upoređivanjem kvantitativnih podataka sa tabelom kriterijuma evaluacije (Tabela iznad), jasno je da uzrok 'nule' kod naivnog prompta leži primarno u gubitku **strukturalnog integriteta**. Dok Fine-tuned prompt održava strogu JSON strukturu (ocena 'Savršen'), naivni prompt generiše mešavinu teksta i koda ('Katastrofalan'), što automatski rezultira neuspehom pri parsiranju, bez obzira na tačnost samog sadržaja."

3. Diskusija o ograničenjima

"Važno je primetiti da čak ni optimizovani (Fine-tuned) prompt ne dostiže 100% uspešnosti na L3 nivou (zadržava se na 70%). Preostalih 30% grešaka pripisujemo inherentnim ograničenjima arhitekture modela Llama-3.2 3B. Kao model sa malim brojem parametara, on poseduje limitiran kapacitet za paralelno praćenje negativnih ograničenja i kompleksnih formata, bez obzira na kvalitet instrukcija. Ovo sugeriše da Prompt Engineering može drastično poboljšati performanse, ali ne može u potpunosti nadoknaditi nedostatak 'sirove inteligencije' manjeg modela."

Zaključak

Ovaj rad je istražio mogućnosti i ograničenja primene lokalnih velikih jezičkih modela (konkretno Llama 3.2 3B) u složenim multi-agentskim sistemima. Kroz razvoj modularne Python arhitekture i seriju kvantitativnih eksperimenata, došli smo do sledećih ključnih saznanja:

Arhitektura ispred sirove snage: Dokazano je da manji modeli, koji su sami po sebi podložni halucinacijama i zaboravljanju instrukcija , mogu postati pouzdani delovi sistema ako se primeni stroga **podela uloga, precizna tokenizacija i dobar prompt**.

Pravci budućeg istraživanja

Za dalji razvoj ovog sistema, predlažu se sledeći koraci:

- **Implementacija dugoročne memorije:** Uvođenje baze podataka kako bi agenti mogli da pamte informacije između različitih sesija.
- **Testiranje većih modela:** Replikacija istih eksperimenata na modelu Llama 3 8B ili Mistral 7B kako bi se utvrdilo da li se "kognitivna litica" pomera ili nestaje sa većim brojem parametara.
- **Samokorekcija:** Razvoj agenta "Kritičara" koji bi automatski pregledao izlaz Analitičara i vraćao ga na doradu pre slanja Piscu, čime bi se potencijalno rešilo preostalih 30% grešaka uočenih u evaluaciji.