Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 4.9
DOI: https://doi.org/10.24867/33BE21Kasikovic

KOMPARATIVNA ANALIZA AGENTSKIH *RAG* PRISTUPA U KONTEKSTU IZGRADNJE KONVERZACIONOG ASISTENTA

COMPARATIVE ANALYSIS OF AGENTIC *RAG* APPROACHES IN THE CONTEXT OF BUILDING A CONVERSATIONAL ASSISTANT

Ivana Kašiković, Aleksandar Kovačević, *Fakultet tehničkih nauka, Novi Sad*

Oblast – SOFTVERSKO INŽENJERSTVO

Kratak sadržaj – *U okviru rada je predstavljeno istraživanje i implementacija dvije verzije konverzacionog asistenta. Fokus je na konceptu Retrieval Augmented Generation (RAG) koji pored „ugrađenog“ koristi i eksterno znanje za generisanje odgovora i interakciju sa korisnikom. Prva implementacija koristi osnovnu kombinaciju dobavljanja znanja i generisanja odgovora, a zatim je razvijen složeniji pristup sa agentima i specijalizovanim alatima za obradu upita. Cilj je detaljno opisati proces izrade sistema, uporediti arhitekture po kompleksnosti, proširivosti, performansama i relevantnosti odgovora. Evaluacija je prvo rađena ručno, a kasnije automatizovana pomoću Ragas okvira. Na kraju su istaknute prednosti i mane oba pristupa, primjeri njihove primjene te prijedlozi za unapređenje i dalja istraživanja.*

Ključne reči: *RAG, Vanilla RAG, agentski RAG, agenti, alati, konverzacioni asistent*

Abstract – *As part of this paper, we present research and the implementation of two versions of a conversational assistant. The focus was on exploring Retrieval-Augmented Generation (RAG), which uses external knowledge in addition to built-in knowledge to generate responses and interact with users. The implementation started with a simple version that combines retrieval and generation. Afterwards, a more advanced approach was developed using agents and specialized tools for query processing. The goal of the paper is to provide a detailed description of the system development process, compare the architectures in terms of complexity, scalability, performance, and answer relevance. Initially, system evaluation was performed manually, while later performance analysis was automated using the Ragas framework. Finally, the advantages and disadvantages of both approaches are highlighted, along with scenarios where each would be appropriate. The paper concludes with suggestions for improvements and directions for future research.*

Keywords: *RAG, Vanilla RAG, agentic RAG, agents, tools, chatbot*

NAPOMENA: Ovaj rad proistekao je iz master rada čiji mentor je bio dr Aleksandar Kovačević, red. prof.

1. UVOD

Arhitektura sistema konverzacionih asistenata, organizacija komponenti, veliki jezički model i još mnogo drugih stvari značajno utiču na ponašanje I rezultate pomenutog sistema. LLM—ovi [1] iako veoma moćni alati imaju ograničeno znanje, tj. znanje na kome su se obučavali što je dosta umanjivalo vrijednost konverzacionih asistenata koji su se bazirali isključivo na njima. Kao rješenje za taj problem uveden je RAG [2] koncept, pomoću kog se stvara mogućnost da se prilikom generisanja odgovora uključi i neko eksterno znanje koje do tada nije bilo poznato velikom jezičkom modelu.

Brzim razvojem ove oblasti stvorene su različite implementacije ovih sistema. Stoga, cilj ovog rada je da izvrši komparativnu analiza izmedju najjednostavnije verzije i kompleksnijeg pristupa implementiranog uz pomoć agenta i specializovanih alata, pri čemu smo prvo dali detaljan opis arhitekture oba rješenja. Kao rezultat imaćemo realnu sliku o tome koliko je implementacija kog sistema kompleksna, koliko pouzdane i relevantne odgovore možemo očekivati za tu količinu utrošenog vremena i na taj način ćemo moći procijeniti koja vrsta arhitekture i implementacije bi bila adekvatna za koji slučaj upotrebe.

U drugom poglavlju biće opisane teorijske osnove RAG sistema kao koncepta, ali i osnove jednostavnog i agentskog RAG-a. Naredno poglavlje baviće se detaljnim opisivanjem procesa implementacije i koraka koji su sprovedeni kako bismo došli do krajnjih verzija sistema. Nakon toga slijedi poglavlje gdje smo opisali proces evaluacije. U petom poglavlju diskutovali smo o rezultatima sistema i data su potencijalna objašnjenja za određene metrike i neka ponašanja. Posljednje poglavlje donosi zaključak sa sažetkom istraživanja.

2. TEORIJSKE OSNOVE

U ovom poglavlju biće date teorijske osnove RAG koncepta, kao i teorijske osnove oba pristupa za implementaciju koji će kasnije biti i realizovana.

2.1. Šta je RAG?

Retrieval-Augmented Generation (RAG) je arhitektonski obrazac koji kombinuje dvije bitne komponente, a то sy: **pretraga (retrieval)** relevantnih informacija iz izvora i **generacija (generation)** odgovora pomoću velikog jezičkog modela. Uvođenjem ovog koncepta riješili smo problem stagnirajućeg znanja, niske

objašnjivosti i otvorili smo mogućnost generisanja odgovora i na osnovu domenski specifičnog znanja.

2.2 Eksterno znanje i njegova integracija u RAG sistem

Eksterno znanje predstavlja osnovni benefit RAG koncepta. Može biti bilo koji struktuirani ili nestruktuirani tekst, a sama procedura indeksiranja je ista i svodi se na pretprocesiranje teksta i podjelu na manje segmente, zatim generisanje embedding reprezentacija od tih elemenata i čuvanje u odabranu vektorsku bazu. Nakon toga, proces pretrage zasniva se na konvertovanju korisničkog upita u embedding reprezentaciju i vrši se pretraga po sličnost gdje su semantički slični vektori nalaze blizu u prostoru [3].

2.3 Mehanizmi promptovanja

Prompt [4] je jedna od ključnih stavki koje utiču na kvalitet generisanog odgovora. To predstavlja niz instrukcija koje se prosljeđuju velikom jezičkom modelu tokom inferencije. Za što bolje rezultate dobra je praksa ispoštovati karakteristike efektivnog prompta, a to su da bude jasan i precizan, da u okviru prompta bude uključen relevantan kontekst, kao i da instukcije budu neutralne tj. da ne sugerišu neki odgovor i da ne nameću određeno mišljenje i stav.

Sa druge strane, postoje i različite tehnike i strategije za unapređenje prompta posebno u slučaju RAG-a od kojih možemo izdvojiti instruction based promptovanje, few shot promptovanje itd.

2.4 Vanilla RAG

Vanilla RAG predstavlja osnovni i najjednostavniji vid RAG koncepta. Ovu implementaciju odlikuje linearan tok obrade upita, prvo se pokreće proces dobavljanja relevantnog konteksta, a zatim se na osnovu toga i generiše odgovor. U nastavku ćemo istaći osnovne osobine ovog pristupa.

2.4.1 Ključne osobine

Karakterisične osobine značajne za ovaj pristup su jednostavnost implementacije i statička logika. To znači da se ovakvi sistemi poprilično brzo stavljaju u upotrebu i da se svi upiti obrađuju na isti način. Kao posljedica svega toga možemo zaključiti da bi debagovanje ovakvog sistema bilo veoma lako.

2.4.2 Nedostaci

Iako je Vanilla RAG veoma jednostavan za implementaciju, ovo rješenje imamo određenih mana, pogotovo kada imamo kompleksnije upite. Glavni problem je što nema mogućnost adaptacije logike na osnovu upita, tako da ukoliko postavimo neki složeni upit vrlo vjerovatno nećemo dobiti adekvatan i složen odgovor.

2.4.3 Praktična primjena

Kao posljedica pomenutih osobina možemo reći da bi ova implementacija najbolje funkcionisala u manjim okruženjima gdje upiti nisu toliko česti i kompleksni, a baza znanja je statična i nije podložna čestim promjenama.

2.5 Agentski RAG

Agentski RAG [5] predstavlja napredni oblik RAG arhitekture u kojoj ključnu ulogu ima inteligentni agent, odnosno entitet sposoban za donošenje odluka, upravljanje kontekstom i strategijsko izvršavanje zadataka u više koraka uz pomoć alata.

2.5.1 Agenti

Agenti [6] predstavljaju samostalne jedinice koje imaju sposobnost da rezonuju, donose odluke i izaberu adekvatne alate za izvršavanje određenih akcija. Postoje različite vrste agenata u zavisnosti od toga koji je njihov cilj. U okviru ovog rada koristiće se ReAct agent koji može da rezonuje i na osnovu međurezultata može da mijenja strategiju i prilagođava je.

2.5.2 Alati

Alati [7] su pomoćne strukture koje agent koristi tokom procesa odlučivanja. U suštini, to je interfejs za funkcije koje mogu da dobavljaju određeno znanje ili izvršavaju neku akciju adekvatnu za obradu upita. Bitna stavka prilikom specificiranja nekog alata je opis, na osnovu koga inteligenti koordinator, agent zna šta je uloga tog alata i u zavisnosti od toga ga pozove ili ne.

2.5.3 Ključne osobine

Ključne osobine karakteristične za agentski RAG su modularnost i fleksibilnost. U nastavku će biti prikazano da se cijela arhitektura ove implementacije sastoji od komponenti, tako da je veoma praktično proširiti neku funkcionalnost, zamijeniti neku komponentu. Pored toga, ima mogućnost orkestracije složenih zadataka i generisanje adekvatnog odgovora i za kompleksnije zadatke.

2.5.4 Nedostaci

S obzirom da raste kompleksnost sistema, to sa sobom povlači neke mane kao što je složenost implementacije. Za kreiranje ovakvog sistema potrebno je više vremena i javlja se povećana latencija pošto se izvršava veći broj koraka. Povezano sa tim, javlja se i problem većih troškova izvršavanja.

2.5.5 Praktična primjena

Upotreba agentskog RAG-a je opravdana u sistemima koji zahtjevaju složenije radnje i laku proširivost sistema dodatnim alatima. Npr. To bi bio slučaj u velikim kompanijama koje zahtjevaju pretragu dokumentacije, generisanje izvještaja slanje mejlova itd.

3. METODOLOGIJA

U ovom poglavlju opisane su metode i koraci koji su primijenjeni tokom rada na projektu, čiji je cilj bio istraživanje i implementacija RAG sistema. Projekat je obuhvatio specifikaciju dizajna i implementaciju obje vrste RAG sistema. U nastavku slijedi detaljniji opis.

3.1 Specifikacija zahtjeva

Projekat je realizovan korištenjem programskog jezika Typescript i Nodej.js radnog okvira, pri čemu je serverski dio organizovan upotrebom modularne monolitne arhitekture. Korisnički interfejs je razvijen u React-u sa ciljem da omogući jednostavnu interakciju sa sistemom.

Aplikacija je odrađena za imaginarnu kompaniju koja bi imala mogućnost da pretražuje kompanijsku dokumentaciju u jednostavnijoj verziji sistema, a u agentskoj verziji ta implementacija je proširena i pristupom internetu i mogućnošću riješavanja matematičkih izraza

preko aplikacije. Prvo je dat opis pripreme podataka za testiranje aplikacije, a onda ćemo opisati i konkretno implementaciju.

3.2 Indeksiranje dokumenata

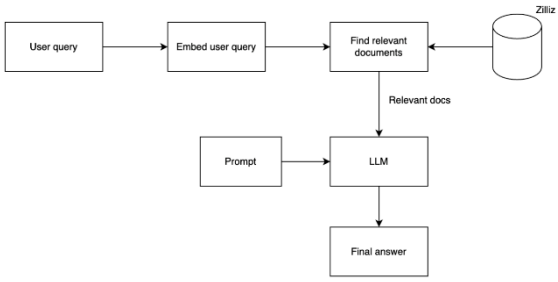
Prvi koraci u razvoju sistema su indeksiranje dokumenata i priprema eksternog znanja koje će se koristiti za testiranje ovih sistema. Za čuvanje generalnog znanja odlučili smo se za vektorsku bazu Zilliz [8]. Ovaj proces započinje raščlanjivanjem dokumenata na manje segmente, u našem slučaju vršena je semantička podjela teksta po pasusima.

Sljedeći korak je kreiranje vektorskih reprezentacija od tih pasusa upotrebom modela za embedovanje. U ovom slučaju koristili smo **OpenAIEmbeddings**, koji u pozadini podrazumjevano koristi napredni **model text-embedding-3-small**.

Posljednji korak pri indeksiranju dokumenata je čuvanje u vektorsku bazu. Odabrana vektorska baza nudi više načina ažuriranja baze, a u ovom radu iskorišten je inkrementalni način ažuriranja baze, pri čemu se tokom ažuriranja unose samo novi ili izmijenjeni segmenti, dok se zastarjeli automatski uklanjaju.

3.3 Implementacija Vanilla RAG-a

Vanilla RAG arhitektura implementirana je kao prva verzija konverzacionog asistenta i kombinuje proces za dobavljanje relevantnih dokumenata i proces generisanja odgovora. Proces dobavljanja relevantnih dokumenata oslanja se isključivo na veliki jezički model i nema mogućnost rezonovanja. Za implementaciju korišten je **Langchain** [9] radni okvir i **ChatGPT Turbo 3.5**. Slika čitave arhitekture prikazana je na slici 1.

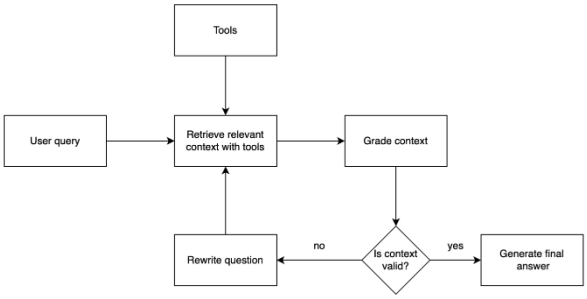


Slika 1. *Arhitektura Vanilla RAG-a*

3.4 Implementacija agentskog RAG-a

Agentski RAG predstavlja napredniju verziju RAG koncepta, u kom se umjesto velikog jezičkog modela kod jednostavne verzije koristi čitav agentski sistem sa različitim alatima, Uz pomoć agenta ovaj pristup dobija mogućnost rezonovanja I adaptacije logike za rješavanje kompleksnih korisničkih zahtjeva.

Za realizaciju ovog sistema korišten je **LangGraph** [10] gdje je svaki čvor predstavljao neki korak u obradi zahtjeva. Dok smo za definisanje alata koristili Langchain interfejs. Detaljniji prikaz arhitekture prikazan je na slici 2.



Slika 2 *Arhitektura agentskog RAG-A*

Na osnovu priloženog možemo zaključiti da prvi korak je dobavljanje relevantnog znanja pomoću agenta i alata. U ovom slučaju korišten je reaktivni agent koji planira dalji korak u skladu sa trenutnim stanjem.

Što se tiče alata za demonstraciju rada kreirali smo tri alata, a to su:

- Alat za dobavljanje znanja o kompaniji
- Alat za internet pretragu
- Alat za rješavanje matematičkih izraza

Zatim slijedi korak provjere da li je kontekst relevantan, ukoliko jeste ide se na kraj i generiše se odgovor, a u suprotnom pitanje se reformuliše i proces se ponavlja.

4. EVALUACIJA

Pod procesom evaluacije podrazumjeva se da se „izmjeri“ koliko tačne i relevantne odgovore sistem daje. Pošto se generalno proces sastoji od dvije faze dobavljanja znanja i generisanja odgovora, dobra praksa je evaluirati te dvije faze odvojeno.

U početku evaluacija je rađena ručno i empirijski na osnovu testnog skupa pitanja, a naknadno je taj proces automatizovan upotrebom Ragas radnog okvira.

5. REZULTATI

U ovom poglavlju biće dato finalno poređenje ove dvije implementacije na osnovu tehničkih metrika koje definišu tačnost odgovora, ali i poređenje po sekundarnim osobinama bitnim za razvoj ovih rješenja.

U prilogu je data tabela sa vrijednostima tehničkih metrika dobijenih preko Ragas [11] radnog okvira.

Tabela 1. *Evaluirane vrijednosti*

	Vanilla RAG	Agentski RAG
Relevantnost konteksta	0.6987	0.8903
Pridržavanje konteksta	0.2951	0.2698

Povraćaj konteksta	0.4773	0.4833
Relevantnost odgovora	0.7818	0.8196
Tačnost odgovora	0.6192	0.8684
Semantička sličnost	0.8434	0.9020

Na osnovu tabele, agentski pristup ima dosta bolje rezultate u relevantnosti konteksta, razlog za to su složeni upiti gdje drugi pristup dosta bolje rezonuje i pronalazi adekvatna dokumenta. Takođe, agentski pristup se neznatno bolje pokazao I u metrici povraćaj konteksta, do toga je došlo, jer agenti malo bolje rade u slučaju kadaa se kontekst vraća iz više dokumenata. Sa druge strane, posmatrajući pridržavanje konteksta jednostavniji model neočekivano daje bolji rezultat. Razlog za to je što agentski pristup daje dosta informacija da što bolje objasni koncept, ali u ovom slučaju to smanjuje vrijednost ove metrike.

Sa druge strane, što se tiče metrika vezanih za generisanje odgovora agentski pristup se dosta bolje pokazao kod tačnosti odgovora, jer ovo rješenje detaljnije vraća odgovor I detalje koji naizgled ne djeluju bitni. Kao posljedica svega toga možemo vidjeti da agentski prisup prednjači I u metrici semantičkoj sličnosti odgovora koji mjeri usklađenost generisanog i referentnog odgovora. Isto objašnjenje važi I za dosta bolju relevantnost odgovora kod agentskog rješenja.

Poredeći sekundarne osobine ovih sistema, očigledno je da je agentski pristup dosta kompleksniji, zahtjeva više vremena za razvoj I troškovi održavanja arhitekture su dosta veći. Ali glavna prednost ovog pristupa je što je lako proširiv i daje bolje, relevantnije I pouzdanije odgovore.

6. ZAKLJUČAK

U radu je predstavljen razvoj i implementacija jednostavne i agentske implementacije RAG sistema. Motivacija za razvoj sistema je uočiti prednost agenata u okviru ovog koncepta s obzirom na brzi razvoj tehnologije. Glavni cilj ovog rada je uporediti spomenute arhitekture, istaći prednosti i mane oba pristupa, kao i potencijalne prijedloge praktične primjene.

Na osnovu rezultata možemo zaključiti da jednostavnija implementacija RAG sistema se relativno brzo postavlja, nema složenu arhitekturu i ima linearan tok izvršavanja akcija. Dok sa druge strane agentski RAG daje detaljnije odgovore, ima sposobnost rezonovanja i značajno prednjači pri odgovaranju na složene upite. Takođe, arhitektura agentskog pristupa je lako proširiva, ali nosi veće troškove i kompleksnija je za razvoj. Stoga prilikom izbora načina implementacije RAG sistema bitno je naći balans između performansi, održavanja i budućeg razvoja.

Za buduća unapređenja, predlaže se optimizacija promptova uz pomoć predloženih alata. Takođe, korisno bi bilo da sistem ima mogućnost učenja tokom konverzacija i pamćenja dobrih praksi. Još jedno ograničenje je ograničen kontekst koji se šalje velikom jezičkom modelu, ukoliko bi se ovaj problem riješio model bi imao više informacija za odgovaranje što bi proizvelo relevantnije odgovore.

7. LITERATURA

[1] What are large language models (LLMs)? https://www.ibm.com/think/topics/large-language-models [датум приступа јун 2025]

[2] Newhauser, Mary; (2024). Introduction to Retrieval Augmented Generation (RAG) https://weaviate.io/blog/introduction-to-rag [датум приступа јун 2025]

[3] Vector Databases: Tutorial, Best Practices & Examples https://nexla.com/ai-infrastructure/vector-databases/ [датум приступа јун 2025]

[4] Prompt Engineering Guide https://www.promptingguide.ai [датум приступа јун 2025]

[5] What is agentic RAG? https://www.ibm.com/think/topics/agentic-rag [датум приступа јун 2025]

[6] Tahir; (2024). What are AI Agents? https://medium.com/@tahirbalarabe2/what-are-ai-agents-f06ef775e78f [датум приступа јун 2025]

[7] What are tools? https://huggingface.co/learn/agents-course/en/unit1/tools [датум приступа јун 2025]

[8] Zilliz званична документација https://docs.zilliz.com/ [датум приступа јул 2025]

[9] Званична документација Langchain-a https://python.langchain.com/docs/introduction/ [датум приступа јун 2025]

[10] Званична документација Langgraph-a https://langchain-ai.github.io/langgraph/concepts/why-langgraph/ [датум приступа јун 2025]

[11] Званична документација за Ragas радни оквир https://docs.ragas.io/en/stable/getstarted/ [датум приступа август 2025]

Kratka biografija:

Ivana Kašiković rođena je 03. oktobra 2000. godine u Trebinju. Kao nosilac diplome „Vuk Karadžić“ 2019. godine završava gimnaziju „Jovan Dučić“, nakon čega upisuje Fakultet tehničkih nauka, smer Softversko inženjerstvo i informacione tehnologije. Po završetku studija u roku, upisuje master akademske studije i iste završava 2025. godine.