

UNIVERZITET U BEOGRADU
FAKULTET ORGANIZACIONIH NAUKA

ZAVRŠNI RAD

**Primena velikih jezičkih modela za odgovaranje
na pitanja iz dokumenata**

Mentor

Dr Sandro Radovanović,

Docent

Student

Milutin Mirković, 2020/0252

Beograd, 2024. godine

Sadržaj

1. Uvod	3
2. Teorijske osnove	5
2.1. Veliki jezički modeli	5
2.1.1. Interakcija sa jezičkim modelima	6
2.1.2. Numerička reprezentacija reči i teksta	7
2.1.3. Arhitektura velikih jezičkih modela	8
2.1.4. Nedostaci velikih jezičkih modela	10
2.2. Sistemi za odgovaranje na pitanja iz dokumenata - upitnički sistemi	11
2.2.1. Arhitektura upitničkih sistema	11
2.2.2. Nedostaci tradicionalnih upitničkih sistema	15
2.3. Generisanje sa proširenjem pretrage – RAG	16
2.3.1. Obrada dokumenata	17
2.3.2. Obrada upita	18
2.3.3. Nalaženje informacija na osnovu sličnosti	18
2.3.4. Slanje zahteva jezičkom modelu	19
3. Opis istraživanja	20
3.1. Opis korišćenih podataka	20
3.1.1. Naučni stil	20
3.1.2. Književnoumetnički stil	20
3.1.3. Novinarski stil	21
3.1.4. Administrativni stil	21
3.2. Opis korišćenih alata	22
3.2.1. Hugging Face	22
3.2.2. LM studio	23
3.2.3. LangChain	24
3.2.4. Opis korišćenih jezičkih modela	24
3.2.5. Istraživačka pitanja	25
4. Istraživanje	27
4.1. Opis razvoja RAG sistema	27
4.1.1. Projektni zahtevi	27
4.1.2. Kod aplikacije	29
4.1.3. Izgled aplikacije	34
4.2. Sprovođenje istraživanja	38

4.2.1.	Istraživanje nalaženja informacija	38
4.2.2.	Istraživanje generisanja odgovora	43
4.2.3.	Rezultati	50
4.3.	Diskusija	51
5.	Zaključak	54
6.	Literatura	56

1. UVOD

U najširem smislu, jezik se definiše kao složeni i multifunkcionalni sistem znakova čija je osnovna i prvobitna svrha sporazumevanje među ljudima. U svojoj dugoj istoriji, koja je ujedno i istorija čovečanstva i ljudske misli uopšte, jer su jezik, mišljenje i svest neraskidivo povezani, jezik neprekidno evoluirao i prilagođava se sve složenijim ljudskim životima i potrebama. Osim komunikativne funkcije, jezik prenosi značenja (misli, osećanja, sadržaj poruka) pa ima i saznavnu, kognitivnu i ekspresivnu funkciju, ali i akumulativnu jer se jezikom prenose iskustva i znanja iz generacije u generaciju.

Još od pojave Tjuringovog testa pedesetih godina prošlog veka, ljudi pokušavaju da stvore mašinsku jezičku inteligenciju nerazlučivu od ljudske.[1] Razvijanje sposobnih algoritama veštačke inteligencije za razumevanje i savladavanje jezika predstavlja značajan izazov, jer je jezik kao osnov ljudskog mišljenja jednako složen i nepredvidiv kao mišljenje samo, ali bi ujedno takav algoritam bio od ogromnog teorijskog i praktičnog značaja.

Jedan od glavnih pristupa unapređivanju jezičke inteligencije mašina je Modelovanje jezika (engl. *Language Modelling*). Uopšteno, osnovni cilj modelovanja jezika je predviđanje sledeće reči u nizu i popunjavanje nedostajućih reči u tekstu, što omogućava računarima da stvaraju tekst na sličan način kao ljudi.[1] Ovo se postiže tako što se modeli treniraju na velikim skupovima tekstualnih podataka, gde uče obrasce u jeziku. Korišćenjem tih obrazaca, model može izračunati koja reč je najverovatnije sledeća u datom kontekstu. Što je veća količina teksta na kojoj se model trenira, veće su i njegove sposobnosti. Zanimljivo je da, kada kompleksnost modela premaši određeni nivo, oni ne samo da postižu bolje rezultate u predviđanju naredne reči, već i poprimaju neočekivane sposobnosti, poput sposobnosti generalizacije, prilagođavanja posebnim problemima, razumevanja dvosmislenosti i humora. Ovi modeli uspešno rešavaju složene jezičke zadatke i pokazuju sofisticirano razumevanje jezika, čak i u situacijama koje zahtevaju logičko zaključivanje i kreativnost. Kako bi se ovi modeli razlikovali i prepoznavali, naučna zajednica im je dodelila termin “Veliki jezički modeli” (engl. *Large Language Models, LLM*).[1] U poslednjih nekoliko godina, intenzivnim radom i saradnjom akademske zajednice i industrije, ostvareni su značajni pomaci na ovom polju, a

najznačajniji napredak je svakako pojava ChatGPT-a, moćnog čebota razvijenog na bazi velikih jezičkih modela, koja je izazvala ogromno interesovanje šire javnosti.

Pojava “Velikih jezičkih modela” predstavlja revolucionarni iskorak u razvoju veštačke inteligencije, koji bi mogao označiti prekretnicu u tehnološkom napretku, pa čak i novu industrijsku revoluciju. Ovi modeli ne samo da poboljšavaju naše trenutne tehnologije, već otvaraju put ka opštoj veštačkoj inteligenciji, koja može da radi jednako dobro ili bolje od ljudi na najrazličitijim kognitivnim zadacima.

Uprkos tome što veliki jezički modeli sadrže ogroman broj informacija i mogu da generišu tekst koji se teško razlikuje od ljudskog, oni ipak imaju jasna ograničenja i to bez obzira na koliko velikim skupovima podataka se treniraju. Naime, nakon treniranja, modeli nemaju pristup novim podacima. Kao rezultat toga, modeli mogu davati zastarele ili netačne odgovore na postavljena pitanja. Takođe, ako su podaci na kojima je model treniran netačni, neprecizni ili zastareli, to će se odraziti i na kvalitet odgovora. Pored toga, veliki jezički modeli ne navode izvore informacija na osnovu kojih daju odgovore, što ih čini nepouzdanima za zadatke koji zahtevaju proverene i tačne informacije.

Kako bi se ova ograničenja prevazišla, uvode se RAG modeli (engl. *Retrieval-Augmented Generation*). Oni omogućavaju odgovaranje na pitanja iz određenog dokumenta tako što kombinuju sposobnost pronalaženja dela teksta koji je relevantan u odnosu na postavljeno pitanje sa sposobnošću velikih jezičkih modela da formulišu odgovor nalik na ljudski.[2]

Teorijski cilj ovog završnog rada je da istraži primenu velikih jezičkih modela u rešavanju problema odgovaranja na pitanja iz dokumenata, kao i da objasni teorijske osnove jezičkih modela i da ukaže na prednosti i nedostatke ovog pristupa. Praktični cilj je razvoj i implementacija RAG sistema uz korišćenje različitih jezičkih modela, s osvrtnom na preciznost i relevantnost dobijenog odgovora, a u zavisnosti od tipa dokumenta.

2. TEORIJSKE OSNOVE

2.1. VELIKI JEZIČKI MODELI

Veliki jezički modeli su sistemi veštačke inteligencije koji imaju sposobnost da generišu jezik opšte namene i da rešavaju druge zadatke obrade prirodnog jezika, poput klasifikacije teksta na osnovu sentimenta. Ovi modeli to postižu učeći statističke odnose iz tekstualnih dokumenata tokom računarski intenzivnog procesa treniranja. Popularni GPT-3 je treniran na 570 gigabajta čistog tekstualnog sadržaja, odnosno oko 240 milijardi reči [3]. Poređenja radi, prosečnoj osobi bi trebalo preko 2000 godina da pročita toliko teksta. Ljudima je mnogo manje reči potrebno da savladaju jezik. Mašinska inteligencija zasnovana je na predviđanju onoga što bi ljudsko biće moglo da kaže i to na osnovu naučenih obrazaca i konteksta kao i prethodnih reči, bez dubljeg razumevanja konteksta i namera.

Korišćenje javno dostupnog sadržaja je omogućilo velikim jezičkim modelima opštu namenu sa širokim spektrom sposobnosti, kao što je klasifikacija teksta, odgovaranje na pitanja, prepričavanje i generisanje teksta. Ovakvi modeli, koji se nazivaju unapred-obučeni (engl. *Pre-trained*), nisu specijalizovani za specifične zadatke u specifičnim oblastima. Oni se mogu prilagoditi posebnoj nameni dodatnim treniranjem na relativno maloj količini domenskog znanja procesom koji se naziva fino podešavanje (engl. *fine-tuning*). [4]

Veliki jezički modeli su u osnovi veštačke neuronske mreže, a njihova veličina i performanse se često izražavaju preko broja parametara koje poseduju. Parametri su ključni za rad modela jer predstavljaju vrednosti koje model uči tokom treniranja i koji definišu način na koji model obrađuje informacije i dolazi do ishoda (na primer, sledeća reč u nizu). Što je veći broj parametara, veća je sposobnost modela da uči složenije odnose u podacima. Na primer, GPT-3 sadrži oko 175 milijardi parametara [3]. Sa druge strane, ovakvi modeli zahtevaju računarske sisteme sa velikom memorijom i procesorskom snagom. Kompanije kao što su OpenAI, Google i Meta su među retkima koje imaju kapacitete da treniraju ovako velike modele. Njihove unapred-obučene modele mogu da koriste manje kompanije i timovi i da ih prilagode sopstvenim potrebama, uz nadoknadu.

2.1.1. Interakcija sa jezičkim modelima

Interakcija između korisnika i velikog jezičkog modela se ostvaruje preko upita. To su pitanja ili zahtevi na osnovu kojih model daje odgovore ili izvršava instrukcije. Dva važna koncepta u vezi upita su dizajn i inženjering upita.

Dizajn upita se odnosi na proces formulisanja jasnog, preciznog i informativnog zahteva, kako bi se povećao kvalitet odgovora. Uključuje odabir reči, fraza i strukture upita, kako bi model znao tačno šta se od njega traži i u kom obliku da pruži odgovor. [5]

Inženjering upita se odnosi na unos dodatnih informacija kojima se upit strukturiira tako da poveća kvalitet i preciznost dobijenog odgovora. To se postiže jasnijim formulisanjem upita, na primer navođenjem stila („napiši pesmu po uzoru na...”), pružanjem dodatnih informacija ili dodeljivanjem uloge jezičkom modelu kao što je: „Ti si pravni savetnik, odgovaraj na pitanja na osnovu prosleđenih zakonika“. Ovakve smernice omogućavaju veću kontrolu nad odgovorima i primenu jezičkih modela na specifičnim zadacima.

Dizajn upita je opštiji koncept i uvek je neophodan, a inženjering se koristi za složenije i specifične zahteve.

Prema prilagođenosti za specifične zadatke, izdvajaju se tri vrste jezičkih modela. To su: generični jezički modeli, modeli podešeni za instrukcije i modeli podešeni za dijaloge. Svaka od ovih vrsta zahteva prilagođavanje upita.

Generični (sirovi) modeli predviđaju narednu reč na osnovu naučenog u fazi treniranja. Ovi jezički modeli obavljaju zadatke nalaženja informacija. Modeli podešeni za praćenje instrukcija su trenirani tako da odgovor baziraju na instrukcijama u zadatim u upitu, to jest da dosledno prate upit. Mogu se koristiti za analizu sentimenata, generisanje teksta ili koda, kao i sažimanje teksta. Za razliku od njih, modeli podešeni za dijaloge su trenirani tako da vode dvosmerne razgovore i mogu se smatrati posebnim slučajem modela podešenih za instrukcije. Koriste se kada su upiti u obliku pitanja i oni po pravilu daju bolje odgovore na prirodne, konverzacijske formulacije pitanja. Optimizovani su da rade brzo i efikasno, kako bi bili korisni u ulozi četbota. [6]

2.1.2. Numerička reprezentacija reči i teksta

Mašine ne mogu da obrađuju tekst u njegovom izvornom obliku, jer razumeju isključivo brojeve i logičke operacije. U oblasti velikih jezičkih modela, jezik se mora predstaviti u formi visoko-dimenzionalnih nizova realnih brojeva koji se nazivaju vektorima. Oni služe kao osnovna jedinica za računanje i omogućavaju velikim jezičkim modelima da vrše matematičke operacije nad jezikom, a da se sačuvaju njegove sintaksičke i semantičke osobine. Proces vektorizacije izvornog teksta počinje podelom teksta na osnovne gradivne jedinice - tokene. Token može biti reč, deo reči, a ponekad i znak interpunkcije. Opšte pravilo je da jedan token sadrži četiri karaktera engleskog jezika, odnosno da 100 tokena sadrži oko 75 reči. Razlozi zbog kojih se koriste tokeni a ne reči su sledeći:

- kada se koriste reči, model može da naiđe na retku ili nepoznatu reč. Podela reči na tokene pomaže modelu da prepozna delove reči koji su mu već poznati,
- kada se koriste tokeni, vokabular modela je manji i efikasniji jer ima manje osnovnih elemenata, što smanjuje memorijske zahteve i vreme treniranja,
- morfološki srodne reči će sadržati iste tokene, te će model prepoznati njihovu sličnost (reč *lovac* sadrži istu morfološku osnovu kao reč *lov* ili *ribolov*),
- model postaje manje osetljiv na slovne greške.

Nakon što se tekst pretvori u tokene, potrebno je te tokene pretvoriti u vektore tehnikom koja se naziva ugrađivanje¹ (engl. *Embedding*).

Ključna ideja je da reči koje se pojavljuju u sličnim kontekstima obično imaju i slična značenja. Ova veza između sličnosti u tome kako su reči raspoređene i sličnosti u njihovom značenju naziva se distribucionalna hipoteza [7]. Potrebno je da model značenja reči bude sposoban da obavlja razne zadatke. Trebalo bi da pokaže da neke reči imaju slična značenja (mačka je slična psu), da neka imaju suprotna značenja (antonimi poput hladno i toplo), kao i da neke reči imaju pozitivnu konotaciju (srećan), dok druge imaju negativnu (tužan). Model bi trebalo da prepozna i činjenicu da su reči poput kupiti, prodati i platiti zapravo različite perspektive iste govorne situacije - kupovine. [7]

¹ U daljem tekstu će se koristiti engleski termin *embedding* s obzirom da je odomaćen i u srpskoj terminologiji ove naučne oblasti, a kako bi se izbegla svaka nepreciznost ili dvosmislenost.

Jedan od prvih uspešnih algoritama za stvaranje vektorske reprezentacije reči je *Word2Vec*, razvijen od strane Gugla 2013. godine. Ovaj algoritam je zasnovan na ideji da reči koje se često pojavljuju u sličnim kontekstima imaju slične numeričke reprezentacije. Ovaj model se trenira na velikim skupovima tekstualnih podataka. U svakoj rečenici, algoritam bira jednu reč kao ciljnu i zatim pokušava da nauči koje reči obično okružuju tu ciljnu reč. Kada algoritam analizira veliki broj rečenica, on stalno ažurira vrednosti vektora svake reči kako bi postigao što precizniju reprezentaciju odnosa među rečima. [8]

Word2Vec generiše vektore sa obično 100 do 300 dimenzija, gde svaki vektor predstavlja određenu semantičku informaciju o reči. Vektori sličnih reči, poput "kralj" i "kraljica", biće bliski u višedimenzionalnom prostoru. Važna odlika ovog algoritma je da linearne transformacije vektora imaju smisla – operacija: "Kralj" minus "Muškarac" plus "Žena" rezultuje vektorom koji je veoma blizu vektorskoj reprezentaciji reči "Kraljica". Svaki broj u vektoru predstavlja jednu dimenziju u vektorskom prostoru reči, pa se može videti da je informacija o polu implicitno ugrađena u neku kombinaciju dimenzija.

Glavni nedostatak ovog algoritma je statičnost vektora, odnosno to što jedna reč uvek ima isti odgovarajući vektor, te se kontekst u kojoj se reč koristi ne uzima u obzir. Vrednost koju će svaka reč imati zavisi od podataka na kojima je model obučen. Jedna reč može imati više značenja, a značenje koje će vektor predstaviti je ono koje se najčešće koristiti na trening setu. Danas je ovaj problem u velikoj meri prevaziđen zahvaljujući pojavi arhitekture transformatora.

2.1.3. Arhitektura velikih jezičkih modela

Arhitektura transformatora, prvi put predstavljena u sada već čuvenom radu "Attention is all you need" [9] je svojom pojavom izazvala revoluciju na polju veštačke inteligencije i obrade prirodnog jezika. Zahvaljujući njima, veliki jezički modeli poput ChatGPT-a (engl. Generative Pre-Trained Transformer) stekli su ogromno globalno interesovanje i stotine miliona korisnika. Veoma brzo je postala dominantna arhitektura jezičkih modela, zamenivši rekurentne neuronske mreže i LSTM (engl. Long Short-Term Memory). Transformatori su arhitektura neuronskih mreža koje koriste slojeve pažnje (engl. *attention*), koja omogućava modelima da se fokusiraju na različite delove ulazne sekvence tokena. Pomoću pažnje, model efikasnije uči zavisnosti između reči, te može da prilagodi značenje reči u zavisnosti od konteksta tj. reči koje se nalaze u okolini.

Tekst se najpre deli na tokene, a onda se svaki token se konvertuje u vektor. Na svakom sloju neuronske mreže, svaki token se kontekstualizuje unutar okvira kontekstnog prozora sa drugim tokenima preko paralelnog mehanizma koji omogućava da se pojača signal za ključne tokene i umanja značaj manje važnih tokena. Neka su date dve rečenice:

- Bolela ga je *glava*.
- Pročitao je osmu *glavu* romana.

Reč *glava* se javlja u dva različita konteksta i samim tim ima dva različita značenja. Algoritam poput Word2Vec-a bi toj reči dodelio istu numeričku reprezentaciju. Da bi model bio zaista koristan, mora da ima sposobnost da prilagodi vrednosti vektora u zavisnosti od reči koje se javljaju u neposrednoj blizini (u kontekstnom prozoru). U ovom primeru, slojevi pažnje služe da reč *glava* u vektorskom prostoru približe reči *poglavlje* kada se reč glava koristi u kontekstu dela knjige.

Originalna arhitektura transformatora se sastoji iz dva dela – enkoder i dekodek. Enkoder kreira embedinge od ulaznog teksta, a dekodek generiše izlazni tekst na osnovu embedinga. Neki tranforamtori koriste samo enkoder (BERT), a neki samo dekodek (GPT). [10]

2.1.4. Nedostaci velikih jezičkih modela

Iako su se veliki jezički modeli pokazali kao moćan alat, imaju i značajne nedostatke. Prvi nedostatak koji se može primetiti je njihova kompleksnost, koja se može izraziti kroz parametre modela. Broj parametara se od pojave prvih transformatora povećao za nekoliko redova veličine. Da bi se savremeni jezički model trenirao, a i koristio u realnom okruženju, potrebna je veoma velika količina resursa. Nezavisni istraživači i mali timovi se moraju oslanjati na fino podešavanje već obučениh modela, a treniranje novih modela i sprovođenje velikih istraživanja je rezervisano za velike univerzitete i multinacionalne kompanije.

Drugi nedostatak se tiče upravljanja memorijom. Kod klasičnih baza podataka, podaci se čuvaju eksplicitno te je poznato koji podaci se koriste i u koje svrhe. Kod velikih jezičkih modela, podaci se čuvaju implicitno u parametrima, što dovodi do problema u radu jer nije moguće znati koji parametri će uticati na ulazne podatke i na koji način. Stoga se ti modeli ponekad nazivaju “crnom kutijom”. Izazivaju više problema. Prvi su halucinacije, odnosno pojava da model vraća odgovore koji deluju uverljivo i tačno, ali su zapravo besmisleni, nepoželjni ili izmišljeni. Drugi problem je otežana moderacija sadržaja koji se generiše – sadržaj može biti pristrasan, uvredljiv ili na bilo koji drugi način neetičan. Treći nedostatak je nemogućnost dodavanja i menjanja podataka u modelu bez ponovnog treniranja. [11]

U praktičnom delu ovog rada, fokus će biti na premošćivanju ovih nedostataka pomoću upitničkih sistema koji služe za odgovaranje na pitanja iz priloženih dokumenata. U narednom poglavlju, biće prikazani tradicionalni upitnički sistemi, koji nemaju prikazane nedostatke jezičkih modela, ali nemaju ni sposobnost da razumeju zahteve korisnika i da pruže odgovor poput čoveka.

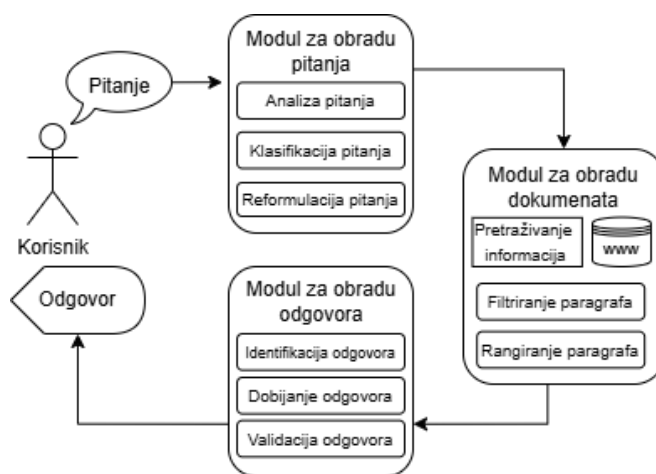
2.2. SISTEMI ZA ODGOVARANJE NA PITANJA IZ DOKUMENATA - UPITNIČKI SISTEMI

Odgovaranje na pitanja (engl. *Question Answering*) je oblast istraživanja koja kombinuje oblasti pretraživanja (engl. *Information Retrieval*) i pronalaženja informacija (engl. *Information Extraction*) i obrade prirodnog jezika [12]. Cilj je stvoriti sistem pitanje-odgovor koji je u stanju da za datu kolekciju dokumenata pronađe odgovore na pitanja postavljena na prirodnom jeziku. Jednostavni sistemi pitanje-odgovor vraćaju najrelevantnije delove dokumenta na osnovu detektovanih ključnih reči u pitanju. Ovi sistemi moraju uspešno da odgovaraju na različite tipove pitanja – pomoću činjenica, definicija, lista, *kako* i *zašto* pitanja, pitanja sa semantičkim ograničenjima (poput “Ko je dobio Nobelovu nagradu za književnost 1961. godine” ili “Koja je druga najduža reka u Evropi?”).

Postoje dve glavne kategorije QA sistema – **upitnički sistemi otvorenog domena**, koji odgovaraju na pitanja iz svih oblasti i oslanjaju se na velike baze znanja i **upitničke sisteme zatvorenog domena**, koji odgovaraju na pitanja iz specifične oblasti.

2.2.1. Arhitektura upitničkih sistema

Upitnički sistemi se tipično sastoje od tri komponente – modul za obradu pitanja, čiji je glavni zadatak klasifikacija pitanja, modul za obradu dokumenata, koji pronalazi deo dokumenta u kojem se može nalaziti odgovor na pitanje i modul za obradu odgovora, koji treba da formuliše odgovor na pitanje. [12]



Šema 1 Upitnički sistem [12]

2.2.1.1. Modul za obradu pitanja

Uloga modula za obradu pitanja je da, nakon što korisnik postavi pitanje, to pitanje predstavi u obliku zahtevane informacije. Stoga, ovaj modul treba da:

- analizira pitanje, tako što će da predstavi glavne informacije koje su potrebne da bi se odgovorilo na pitanje,
- klasifikuje pitanje prema različitim vrstama pitanja, što direktno utiče na oblik odgovora i
- reformuliše pitanje u upite za sistem za pretragu informacija.

2.2.1.1.1. Analiza pitanja

Cilj analize pitanja je određivanje fokusa pitanja. Fokus pitanja je reč ili niz reči koje ukazuju na to koja se informacija traži u pitanju. Na primer, u pitanju “Koji je glavni grad Francuske”, fokus pitanja je „glavni grad“. Ako su vrsta i fokus pitanja dobro određeni, sistem će lakše odrediti koji je tip odgovora potreban. Identifikacija fokusa se može uraditi pomoću pravila za podudaranje paterna (engl. Template matching). [12]

2.2.1.1.2. Klasifikacija tipa pitanja

Kako bi adekvatno odgovorio na pitanje, sistemu je neophodno da zna kakav tip informacija pitanje zahteva, a poznavanje tipa pitanja pruža neophodna ograničenja na podatke koji se mogu smatrati važnim. To omogućava drugim modulima da lakše pronađu deo dokumenta koji sadrži ključnu informaciju. Klasifikacija, dakle, pruža smernice sistemu o prirodi potrebnog odgovora. Tipovi mogu biti *ko*, *šta*, *gde*, *kako*, *kada*, *zašto* i druge. Tip odgovora se određuje na osnovu tipa pitanja i može se odrediti prostim mapiranjem.

2.2.1.1.3. Reformulacija pitanja

Kada se odrede fokus i tip pitanja, modul formira listu ključnih reči koja se prosleđuje modulu za obradu dokumenata. Izvlačenje ključnih reči se može izvršiti različitim tehnikama, kao što je prepoznavanje imenovanih entiteta (engl. Named Entity Recognition). Na osnovu ključnih reči, modul za obradu dokumenata traži onaj deo teksta koji se najviše podudara.

2.2.1.2. Modul za obradu dokumenata

Glavni cilj modula za obradu dokumenata je da na osnovu prosleđenog upita kreira skup delova teksta koji su rangirani i sadrže odgovore na pitanje. Modul to postiže tako što:

- pronalazi skup relevantnih delova teksta u odnosu na pitanje (Retrieval),
- filtrira dobijeni skup, kako bi se smanjio broj delova teksta i količina teksta i
- rangira delove teksta po verovatnoći da sadrže ključnu informaciju odnosno odgovor na pitanje

Skraćivanje teksta na paragrafe čini sistem bržim, što je veoma važno zbog interaktivne prirode ovih sistema.

2.2.1.2.1. Pretraživanje informacija

Informacioni domeni, bilo da su otvoreni ili zatvoreni, sadrže ogromnu količinu informacija. Cilj sistema za pretraživanje informacija je da pronađe i vrati delove teksta koji najviše odgovaraju ključnim rečima u pitanju. Jedan od načina da se ovo postigne je

da se pronađu svi paragrafi koji sadrže sve ključne reči iz pitanja. Nedostatak ovog pristupa je što on zanemaruje semantičko značenje reči i kontekst u kojem se nalaze, a takođe sistem može preskočiti paragrafe koji umesto ključnih reči sadrži slične reči ili sinonime.

U prethodnom poglavlju su bili prikazani načini da se reči prikažu u formi vektora. Ukoliko se upit i svaki paragraf predstavi preko odgovarajuće vektorske reprezentacije, onda se između njih može izračunati kosinusna sličnost. Kosinusna sličnost je kosinus ugla koji zaklapaju dva vektora. Na osnovu toga se paragrafi mogu rangirati, paragraf sa najvećom vrednosti kosinusne sličnosti sa upitom će imati najveće šanse da sadrži odgovor na pitanje. Prednost ovog pristupa je što nije osetljiv na sinonime i slične reči. Nedostatak leži u tome što može dovesti do vraćanja delova teksta koji nisu u potpunosti relevantni jer ovaj pristup ne garantuje da su sve ključne reči prisutne.

2.2.1.3. *Modul za obradu odgovora*

U poslednjoj fazi sistema pitanje:odgovor, modul za obradu odgovora treba da pruži odgovor na osnovu pitanja izabranih paragrafa. Da bi se to postiglo, modul treba da:

- identifikuje kandidate za odgovor unutar filtriranih i rangiranih paragrafa pomoću parsiranja,
- izvuče odgovor birajući reči ili fraze pomoću heuristika i
- validira odgovor pružanjem stepena uverenosti u odgovor.

2.2.1.3.1. *Identifikacija odgovora*

Tip odgovora koji je određen tokom procesa obrade pitanja je ključan za identifikaciju odgovora. Kako bi se odredio imenovani entitet (datum, lokacija, ime osobe) koristi se parser. Parser je softverski alat koji analizira i razlaže strukturu jezičkog ulaza na elemente kao što su subjekat, predikat, objekat. Izlaz iz parsera je kandidat za odgovor, a izvlačenje odgovora i validacija se vrše pomoću heuristika.

2.2.2. Nedostaci tradicionalnih upitničkih sistema

Predstavljena arhitektura sistema za odgovaranje na pitanja iz dokumenta ima značajne nedostatke koji limitiraju njihovu širu upotrebu. Tradicionalni sistemi se oslanjaju na poklapanja ključnih reči iz upita i dokumenata, što često dovodi do pogrešnih ili nepotpunih odgovora jer nisu sposobni da prepoznaju širi kontekst, odnose između reči i pojave poput homonimije i sinonimije. Takođe, nisu fleksibilni prema korisničkoj nameri. Njihova sposobnost da prepoznaju pitanja korisnika je niska pogotovo kada su ona dvosmislena i nedovoljno precizna. Mogu davati dobre rezultate kada korisnik traži jednostavne odgovore u vidu činjenica, ali kod složenijih pitanja, poput *zašto* i *kako* pitanja, kada je potrebno dublje razumevanje i širi kontekst, nisu u stanju ni da nađu odgovarajuće delove dokumenata, ni da korisniku daju adekvatan odgovor.

Razvoj dubokog učenja i velikih jezičkih modela doveo je do velikog napretka upitničkih sistema. Veliki jezički modeli u svojim parametrima čuvaju ogroman broj informacija i mogu odgovoriti na veliki broj pitanja. Takođe, imaju mnogo veću sposobnost da razumeju šta korisnik želi i kakav odgovor očekuje, i da ga daju brzo, na prirodan i razumljiv način.

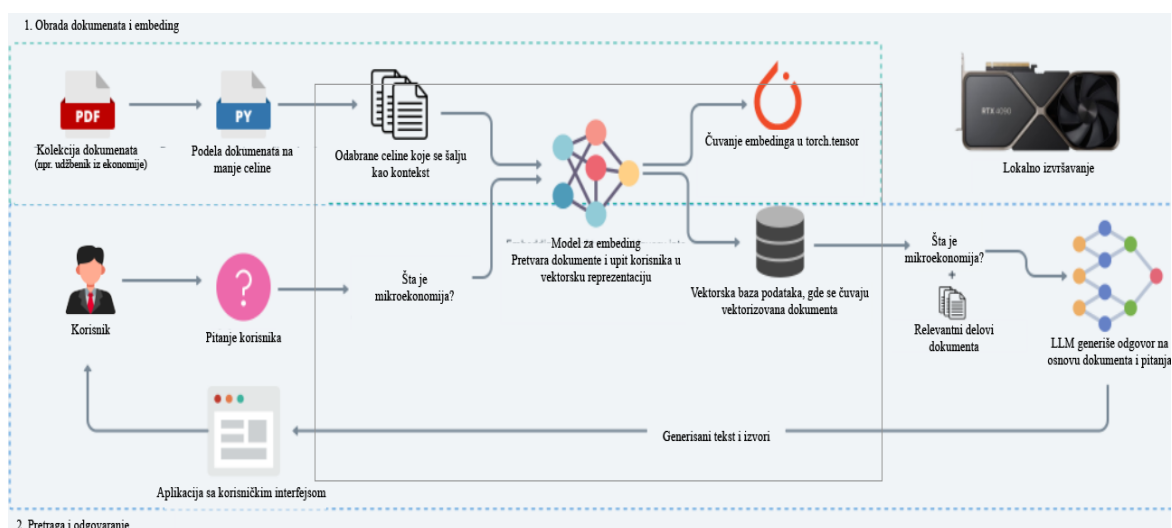
Prednost tradicionalnih arhitektura sistema u odnosu na velike jezičke modele je u tome što je izvor podataka poznat. Ovo je od velikog značaja u situacijama kada je tačnost i proverenost podataka neophodna ili kada sistem radi sa privatnim podacima.

Kako primeniti sposobnosti velikih jezičkih modela da razumeju potrebe korisnika i da pruže jasan i prirodan odgovor na upitničke sisteme kod kojih korisnik određuje izvorne dokumente je glavna tema ovog završnog rada i u nastavku će fokus biti na tome. Dominantna arhitektura sistema za odgovaranje na pitanja iz dokumenata se naziva *Generisanje sa proširenjem pretrage* - RAG (engl. *Retrieval-Augmented Generation*) i ona će najpre biti opisana teorijski, a potom i praktično.

2.3. GENERISANJE SA PROŠIRENJEM PRETRAGE – RAG

Kao što je već naglašeno, unapred-obučeni jezički modeli su pokazali izvanrednu sposobnost da usvoje veliku količinu informacija. To postižu bez pristupa spoljašnjoj memoriji, u vidu implicitne parametrizovane baze znanja. Iako je njihov razvoj revolucionaran, imaju i svoje nedostatke: ne mogu lako da ažuriraju i prošire svoje znanje, ne mogu da objasne svoje predikcije i stvaraju „halucinacije“. Hibridni modeli koji kombinuju parametarsku memoriju sa neparametarskom (spoljnom, u vidu eksterne baze dokumenata) mogu rešiti neke od problema, jer se njihovo znanje može revidirati i proširiti, a odgovori se mogu tumačiti i pregledati. U ovom radu, fokus će biti na RAG (generisanje sa proširenjem pretrage) modelu koji je predstavila META 2021. godine u radu *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. [2]

RAG model kombinuje aspekt upitničkih sistema da se odgovor dobija na osnovu pronađenog relevantnog dela teksta sa sposobnošću jezičkih modela da razumeju kontekst, sažmu prosleđeni tekst i formulišu prirodan i koncizan odgovor. Kada korisnik postavi pitanje, najpre se u bazi dokumenata traži delovi teksta koji sadrže tražene informacije, a onda se jezičkom modelu u okviru upita šalje korisnikovo pitanje, relevantni delovi teksta i instrukciju poput: „Na osnovu priloženog paragrafa, odgovori na sledeće pitanje.“ Na taj način, jezički model ne koristi znanje koje je memorisano u njegovim parametrima, već obrađuje informacije koje su mu prosleđene i na osnovu njih formuliše odgovor. Time korisnik dobija tačniji odgovor jer se oslanja na aktuelne eksterne izvore koje može da bira, umesto na ograničeno znanje modela. [13]



Shema 2 Generisanje sa proširenjem pretrage [14]

2.3.1. Obrada dokumenata

Prvi korak Rag modela je obrada dokumenata. Dokumenti mogu dolaziti iz različitih izvora i to prvenstveno zavisi od domena problema i potreba. Mogu se koristiti razni tipovi dokumenata kao što su članci, naučni radovi, blogovi, knjige, baze podataka, podaci u JSON ili XML formatu. Na primer, ukoliko model treba da bude otvorenog domena, mogu se koristiti članci sa Vikipedije ili neki od popularnih skupova podataka (*Natural Questions*, *TriviaQA*). Ako se koristi u poslovne svrhe u vidu korisničkog servisa mogu se koristiti interna baza znanja, specifikacije o proizvodima, priručnici za korisnike ili često postavljana pitanja. Naredni korak je podela teksta na manje delove (paragrafi, “čankovi”), kako bi postali pogodni za dalju obradu. Svaki od tih delova teksta će dobiti vektor, na osnovu kog će se tražiti sličnost. Za stvaranje embedinga je poželjno koristiti rečenični enkoder (engl. *Sentence Encoder*). To su modeli dubokog učenja transformatorske arhitekture koji su specijalizovani za transformaciju rečenica ili manje količine teksta u numeričku reprezentaciju koja čuva smisao i kontekst rečenice. U ovoj fazi, vidimo prvu primenu velikih jezičkih modela na upitničke sisteme. Vektori, zajedno sa originalnim tekstom i metapodacima (indeks dela teksta, naziv dokumenta iz kog deo teksta potiče, stranica na kojoj se tekst nalazi) čuvaju se u perzistentnoj vektorskoj bazi podataka, što omogućava efikasno korišćenje sistema jer proces kreiranja embedinga može zahtevati puno vremena i resursa. [15]

Određivanje veličine dela teksta je izazovan problem i zahteva pronalaženje pravog odnosa između više faktora. Cilj je da se jezičkom modelu prosledi jedan ili nekoliko značajnih delova teksta, na osnovu kojih će model formulisati odgovor. Da bi odgovor bio konkretniji i tačniji, potrebno je poslati što manju, a opet dovoljnu količinu informacija da se odgovori na pitanja. Kada je u pitanju pretraga dokumenata u odnosu na pitanje korisnika, ona će biti efikasnija i preciznija kada su čankovi mali, jer veliki čankovi mogu da sadrže beznačajne informacije pored korisnog sadržaja. Sa druge strane, mali čankovi mogu da dovedu do značajnog gubitka konteksta i informacija i da dovedu do lošijeg odgovora modela. Jači jezički modeli imaju bolju sposobnost da izvuku relevantne informacije i daju odgovor u skladu sa instrukcijama, ali oni zahtevaju snažne i skupe računare (i u smislu RAM memorije i u smislu procesorske moći).

Optimalna veličina čanka teksta ne postoji. Ona zavisi od prirode teksta, dostupnih resursa i zahteva korisnika. U praksi, tipična veličina iznosi od 200 do 750 reči.[16]

2.3.2. Obrada upita

Upit je zahtev ili pitanje koje korisnik postavlja modelu. U prvoj fazi, na osnovu upita je potrebno naći deo teksta koji njemu najviše odgovara. Da bi se takav deo teksta pronašao, najpre se mora embedovati upit, obavezno istim modelom kojim su embedovani dokumenti. U tradicionalnim upitničkim sistemima se koristila reformulacija upita sa ciljem da se izvuku zahtevi i fokus pitanja. Postoje tehnike koje kod RAG Sistema mogu dekomponovati upit kako bi sistem bolje razumeo složena pitanja. Na primer, *multi-query* kreira više podupita kako bi sistem pretražio više aspekata istog pitanja, a *step-back* generiše slična pitanja kako bi se osiguralo nalaženje relevantnih delova teksta. [15]

2.3.3. Nalaženje informacija na osnovu sličnosti

Kada su delovi dokumenata i upit pretvoreni u vektore, moguće je izračunati sličnost između svakog para upit-čank i iz baze vratiti one delove teksta koji najbliže određuju upit. Najpopularnija metrika za određivanje sličnosti dva vektora je kosinusna sličnost. Ona je vrednost kosinusa ugla koji zaklapaju vektori u višedimenzionalnom prostoru.

$$\cos(\theta) = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\| * \|\vec{B}\|}$$

- $\vec{A} \cdot \vec{B}$ predstavlja skalarni proizvod dva vektora
- $\|\vec{A}\| * \|\vec{B}\|$ predstavlja proizvod normi dva vektora

Vrednosti kosinusne sličnosti mogu imati vrednosti od -1 do 1, gde:

- 1 predstavlja maksimalnu sličnost (identični su po smeru)
- 0 znači da nemaju sličnost (ortogonalni su)
- -1 predstavlja maksimalnu različitost (vektori su suprotni po smeru)

Rezultati pretrage su delovi dokumenata koji su najbliži korisnikovom upitu, rangirani po sličnosti. Prednost pretrage po sličnosti je visoka preciznost, odnosno sistem može da identifikuje relevantne delove teksta čak i kada se ključne reči ne poklapaju jer se prepoznaje semantička sličnost. Takođe, pretraga je veoma brza, čak i kada se radi sa velikom količinom podataka.

2.3.4. Slanje zahteva jezičkom modelu

Sledeći korak je slanje zahteva jezičkom modelu. Upit koji će se slati se sastoji iz konteksta, instrukcija i inicijalnog pitanja korisnika. Kontekst su delovi dokumenata koji su pronađeni od sistema za pronalaženje informacija, a instrukcije su uputstva koja jezički model treba da sledi, na primer, „Na osnovu priloženog konteksta, odgovori na sledeće pitanje.“ Na taj način, model zna da treba da iskoristi priložene informacije za kreiranje odgovora, tj. da ne koristi sopstvenu parametarsku memoriju. Na kraju se generisani tekst prikazuje korisniku.

Komunikacija sa velikim jezičkim modelom se može uspostaviti lokalno, kada se jezički model pokreće i koristi na lokalnoj mašini, ili udaljenim povezivanjem, kada se preko API-ja pristupa modelima koji su hostovani na udaljenim serverima ili oblaku. Veliki jezički modeli zahtevaju snažne hardverske resurse, što je glavni nedostatak lokalnog izvršavanja. Pristup preko API-ja omogućava rad sa najboljim modelima, ali se ta usluga naplaćuje. [17]

3. OPIS ISTRAŽIVANJA

U ovom poglavlju biće opisano istraživanje koje za cilj ima da ispita kako veliki jezički modeli mogu da unaprede klasične upitničke sisteme RAG sistema.

3.1. OPIS KORIŠĆENIH PODATAKA

Kako bi se adekvatno prikazala primena, prednosti i slabosti RAG sistema, istraživanje će se sprovedi na četiri različita dokumenta. Svaki od dokumenata napisan je različitim funkcionalnim stilom. Funkcionalni stilovi su različiti načini upotrebe jezika kojim se on ostvaruje u različitim okolnostima i sa različitom namerom. [18]

3.1.1. Naučni stil

Naučnim stilom se pišu dela iz različitih naučnih oblasti, eseji, studije i udžbenici. Odlikuju ga primena stručnih izraza, minimalno korišćenje stilski oblikovanih rečenica, upotreba reči u osnovnom obliku. Teži se što preciznijem govoru, jasnom i nedvosmislenom izražavanju. Naučni radovi takođe prate stroge strukture, kao što su uvod, pregled literature, metodologija, istraživanje i zaključak.

Pretpostavka je da će, ako je korisničko pitanje u skladu sa dokumentom, RAG sistem uspeti da nađe značajne delove teksta i da na pravim informacijama bazira odgovore. S druge strane, pretpostavlja se da modeli koji će se koristiti neće uspeti da formulišu jasne i za korisnika razumljivije odgovore od onoga što piše u delu. Takođe, moguće je da model pogrešno protumači prosleđeni kontekst i da samim tim da netačan odgovor.

Dokument koji će se koristiti je knjiga *The Hundred-Page Machine Learning Book* autora Andrija Burkova [19]. U pitanju je praktični vodič kroz osnove mašinskog učenja i pokriva glavne teme oblasti na direktan i razumljiv način. Koncepti nisu previše opširno objašnjeni, pa se očekuje da sistem u velikom broju slučajeva pronađe neophodne informacije, i pruži jasan odgovor.

3.1.2. Književnoumetnički stil

Književni ili literalnim stilom se pišu književna dela – lirske pesme, pripovetke, romani. Prepoznatljiv je po subjektivnosti, originalnosti, stilskim figurama, slikovitosti i emocionalnosti.

Pretpostavka je da će ovakva dela dati najslabije rezultate. Pronalaženje korisnih informacija će biti teško jer je za odgovaranje na pitanja potreban ne samo široki kontekst u delu, već i kontekst i iskustva autora koji se ne mogu naći u tekstu. Od modela koji će biti korišćeni se ne mogu očekivati adekvatni odgovori, s obzirom na višeznačnost reči u književnom jeziku, upotrebu prenesnih i metaforičkih značenja, sinonimije, frazeologizama i slično. Nemogućnost davanja zadovoljavajućeg odgovora u velikoj meri je posledica neadekvatno postavljenog pitanja. Pitanje bi, naime, moralo da sadrži tačno one izraze koji je sam autor teksta upotrebio.

Dokument koji će biti korišćen je roman „Zločin i kazna“ ruskog pisca Fjodora Dostojevskog [20]. Delo ima 767 strana i poznato je po dubokoj psihološkoj analizi likova, unutrašnjim monolozima i filozofskim pitanjima. Samim tim, ne očekuje se velika preciznost u pronalaženju informacija, kao ni u dobijanju odgovora.

3.1.3. Novinarski stil

Novinarskim stilom se pišu vesti, intervjui i reportaže. Novinarske tekstove odlikuje jezgrovitost i sažetost u kazivanju, usmerenost na jasno prenošenje poruke i obaveštenja. Teži se jasnom i objektivnom prenošenju činjenica i korišćenju jednostavnog jezika.

Pretpostavka je da će Rag sistem najbolje odgovarati na ovakve dokumente jer bi trebalo da bez problema nađe odgovarajući deo teksta, koji će model kasnije moći da interpretira na jasan način.

Dokument koji će se koristiti je novinarski članak *Climate Change Is Speeding Toward Catastrophe. The Next Decade Is Crucial, U.N. Panel Says* [21], koji je objavljen u Njujork Tajmsu 2023. godine.

3.1.4. Administrativni stil

Administrativnim stilom se pišu molbe, zahtevi, zakoni, računi i slični tekstovi. Odlikuje ga šablonsko pisanje – u formalnim članovima različitih zakona se razlikuju samo neke reči. Ovo može da bude veliki izazov za pronalaženje informacija i očekuje se da će model praviti greške, pa da će samim tim i odgovori modela biti pogrešni. Kod ovakvih tekstova je od posebnog značaja da odgovor bude tačan i precizan, tako da je pretpostavka da će sistem biti od najmanje koristi kod administrativnih dokumenata.

Dokument koji će se koristiti je Politika privatnosti kompanije OpenAI preuzeta sa njihovog sajta. [22]

3.2. OPIS KORIŠĆENIH ALATA

3.2.1. Hugging Face

Unapred-obučeni modeli mašinskog učenja predstavljaju ključne resurse za razvoj softverskih sistema zasnovanih na veštačkoj inteligenciji jer kada se jednom obuče jedan ili više generičnih zadataka, naučnici i programeri ih mogu prilagoditi sopstvenim, specifičnim zadacima procesom finog podešavanja. Inspirisan ponovnom upotrebom softverskih biblioteka putem menadžera paketa (npr. Maven Central, npm ili PyPi), američka kompanija Hugging Face je kreirala centar za unapred-obučene modele kako bi demokratizovala veštačku inteligenciju. *Hugging Face* je postala važna platforma u svetu veštačke inteligencije jer omogućava pristup velikom broju unapred-obučениh modela za različite svrhe (obrada prirodnog jezika, slika, zvuka), velikom broju skupova podataka za treniranje modela i API-eve visokog nivoa za jednostavno korišćenje modela kroz bekende kao što su *Keras* i *Pytorch*. [23]

Transformers biblioteka, koja sadrži veliki broj modela zasnovanih na transformatorskoj arhitekturi, kao što su GPT, BERT, RoBERT i mnogi drugi, možda je najznačajniji doprinos Hugging Face-a. Biblioteka je fleksibilna i jednostavna za korišćenje što omogućava laku integraciju modela u projekte.

U ovom radu korišćeni su modeli dostupni na Hugging Face platformi i to:

- All-mpnet-base-v2, korišćen za embedding dokumenata i upita
- Mistral-Nemo-Instruct-2407-GGUF
- Meta-Llama-3.1-8B-Instruct-GGUF
- Claude2 Alpaca 13BLM studio

3.2.2. LM studio

Language Model Studio je softverska platforma koja korisnicima omogućava otkrivanje, preuzimanje i korišćenje velikih jezičkih modela na sopstvenim računarima. Sve što je dostupno na Hugging Face-u je dostupno za skidanje u LM studiju. Pomoću ove platforme, korisnik može da preuzme i pokrene veliki jezički model direktno na svom računaru, da fino podešava unapred-obučene modele i da poveže model sa aplikacijom pomoću lokalnog servera na koji se šalju HTTP zahtevi. [24]

Za ovaj projekat, LM studio je korišćen za odabir jezičkih modela koji su kasnije bili korišćeni za generisanje odgovora na pitanja. Za odabir modela, kriterijumi su bili sledeći:

- veličina modela izražena kroz broj parametara,
- kompatibilnost sa hardverskim resursima,
- kvantizacija – nivo preciznosti modela,
- brzina i efikasnost,
- podešenost na praćenje instrukcija i
- popularnost i aktuelnost.

U skladu sa ciljem rada, a to je da se istraži primena velikih jezičkih modela na odgovaranje pitanja iz dokumenata, testirani su i evaluirani različiti modeli. Glavni limitirajući faktor je bio hardverski kapacitet uređaja sa kojeg je aplikacija izvršavana. U skladu sa tim, birane su najbolje dostupne verzije vodećih kompanija – Mistral, Google, Meta.

Aplikacija je koristila lokalni server u okviru LM studija za slanje upita i primanje odgovora.

3.2.3. LangChain

LangChain je biblioteka koja pruža alate za rad sa velikim jezičkim modelima. Ona pojednostavljuje proces razvoja softvera koji primenjuju obradu prirodnog jezika tako što nudi veliki broj gotovih funkcija za razvoj RAG modela. Korišćene su funkcije za učitavanje tekstualnih fajlova, podelu dokumenata na manje delove, dobijanje embeddinga, skladištenje u vektorskoj bazi *Chroma* kao u funkcije za pronalaženje relevantnih delova dokumenata.[25]

3.2.4. Opis korišćenih jezičkih modela

3.2.4.1. *All-mpnet-base-v2*

Ovaj model je korišćen za dobijanje embeddinga. Spada u vrstu rečeničnih transformatora (engl. *Sentence Transformers*) i korišćen je za dobijanje vektorskih reprezentacija čankova teksta i upita. Veličina modela je 109 miliona parametara i transformiše delove teksta u vektore sa 768 dimenzija. [26]

3.2.4.2. *Llama - 3.1 - 8B - Instruct - Q4_K_M.gguf*

Llama (akronim za Large Language Model Meta AI) je porodica velikih jezičkih modela koje je kompanija Meta AI objavila prvi put u februaru 2023. godine. Trenutno je aktuelna najnovija verzija 3.1 objavljena u julu 2024. godine. U zavisnosti od broja parametara, postoje tri verzije, Llama 3.1 8B, Llama 3.1 70B i Llama 3.1 405B, gde drugi broj označava milijarde parametara koji poseduje. Llama 3.1 405B je model koji zahteva centar za podatke, a Llama 8B i 70B su „destilovane“ verzije, optimizovane za manje sisteme, od laptopova do servera. Korišćena je verzija sa 8 milijardi parametara, fino podešena za praćenje instrukcija. Kvantizacija Q4 KM znači da je model kvantizovan na 4-bitnu preciznost, što je najveća preciznost koju računar može da podrži. [27]

3.2.4.3. *Mistral-Nemo-Instruct-2407-IQ4_XS.gguf*

Mistral Nemo je jezički model sa 12 milijardi parametara, razvijen od strane kompanija Mistral i NVIDIA [28]. Kao i Llama, korišćena je fino podešena verzija za praćenje

instrukcija. Ima više parametara od Lame, pa se očekuje veći kvalitet odgovora, ali i duže vreme za njihovo kreiranje.

3.2.4.4. Claude2-alpaca-13B-GGUF

Ovaj model je zasnovan na kombinaciji *Claude 2* i *Alpaca* modela. Claude 2 je model razvijen od strane kompanije *Anthropic*. Alpaca je model fino podešen za praćenje instrukcija i baziran je na Lami, a razvijen je na osnovu izlaza modela kao što je GPT-3. Ova specifična verzija sadrži 13 milijardi parametara.[29]

3.2.4.5. ChatGPT 4o

ChatGPT 4o se smatra važi za *state-of-art* model i u radu će se koristiti za poređenje sa ostalim modelima. Neće se pokretati lokalno niti će se zahtevi slati kroz aplikaciju pomoću API-ja, već direktno preko OpenAI platforme.

3.2.5. Istraživačka pitanja

Dve faze u kojima veliki jezički modeli mogu da poboljšaju upitničke sisteme su faza pronalaženja informacija i faza identifikacije odgovora. U skladu sa tim, istraživanje će biti podeljeno u dve faze.

U prvoj će biti cilj odgovoriti na pitanje koliko uspešno RAG sistem može da pronađe odgovarajuće delove teksta u dokumentima. Ovo pitanje je od ključnog značaja jer bez relevantnih informacija, podsistem za generisanje odgovora neće davati željene rezultate, ma koliko dobar bio. Prvo će se za svaki navedeni dokument nasumično izabrati šest pasusa, potom će se iz tih pasusa izvlačiti pitanje. Ta pitanja će se postaviti sistemu za pronalaženje informacija i pratiće se koliko puta je vraćen pasus iz kojeg je pitanje poteklo. Takođe, cilj je odrediti i veličinu *čanka* teksta².

U drugoj fazi, pitanja i kontekst će biti poslani jezičkom modelu. Ispitaće se najpre koliko uspešno RAG sistem može da identifikuje ključne informacije iz priloženog teksta,

² Predlažem upotrebu engleskog termina *čank* (chunk) koji u bukvalnom smislu označava komad, parče teksta. Smatram da izrazi paragraf ili pasus ne odgovaraju sasvim onome što je označeno pojmom „čank“ jer oba ova termina podrazumevaju izvesnu smisaonu celinu teksta, a *čank*, kako se ovde shvata, može da obuhvata smisaonu celinu, ali može započeti usred rečenice i slično.

odnosno da li model bazira odgovore na ključnim informacijama iz teksta. Pratiće se i koliko je često model davao odgovore ne koristeći informacije iz teksta.

Analiziraće se kvalitet odgovora po pitanju jasnoće, čitljivosti i upotrebljivosti za korisnike.

S obzirom da se jezički modeli pokreću i koriste lokalno, a hardverski resursi su ograničeni, identičan zahtev će se slati i modelu ChatGPT-4o, koji predstavlja najmoćniji (state-of-art) jezički model. Cilj je ispitati koliko su korisni i primenljivi jezički modeli koji mogu da se koriste lokalno u prosečnim uslovima.

Uporediće se rezultati po pitanju tipa dokumenta kako bi se otkrilo koji tipovi su najprikladniji, a koji zadaju najveće probleme i zašto.

4. ISTRAŽIVANJE

4.1. OPIS RAZVOJA RAG SISTEMA

U ovom poglavlju biće opisan proces izrade veb aplikacije koja primenjuje RAG sistem za odgovaranje na pitanja iz dokumenata.

4.1.1. Projektni zahtevi

Razvoj RAG sistema počinje prikupljanjem i definisanjem projektnih zahteva, na osnovu kojih će biti implementirane funkcionalnosti aplikacije. Funkcionalni zahtevi predstavljaju specifikaciju za aktivnost koju sistem mora da obavi. Ograničavajući zahtevi su uslovi koji usmeravaju kako sistem može da bude napravljen. Ograničavajući zahtevi su tehnološke i pravne prirode. Tehnološka ograničenja se odnose na hardverske kapacitete. Zbog ograničenosti dostupne memorije i procesorske snage, vreme izvršavanja i dobijanja odgovora se u ovom radu neće razmatrati, iako je brzina obrade i generisanja odgovora jedan od najvažnijih zahteva komercijalnih Rag sistema. Pravni zahtevi podrazumevaju poštovanje Zakona o autorskim pravima.

4.1.1.1. Funkcionalni zahtevi

- Učitavanje dokumenata

Sistem mora da ima podsistem za rad sa fajlovima u PDF formatu. Potrebno je napraviti komponentu grafičkog korisničkog interfejsa koja će primati dokumenta i perzistentno ih čuvati u lokalnoj memoriji. Potrebno je omogućiti da se dokument učitava samo jednom.

- Obrada dokumenata

Potrebno je omogućiti podelu dokumenata na čankove radi lakše pretrage, embedinga i slanja konteksta za odgovaranje, korišćenjem nekog rešenja iz *LangChain* biblioteke.

- Embeding dokumenata

Sistem mora da ima funkciju koja će kreirati embedinge dokumenata pomoću transformatora rečenica. Dobijene vektore potom će perzistentno čuvati u memoriji. Onemogućiti višestruko embedovanje istog dokumenata.

- Zahtevanje obrade i embeddinga

Korisniku omogućiti da na dugme pozove sistem da sačuva dokument, obradi ga i embeduje.

- Prijem i obrada korisničkog upita

Korisnik preko forme treba da postavi pitanje. Omogućiti korisniku da zatraži odgovor od velikog jezičkog modela i da prikaže relevantne delove teksta.

- Pretraga relevantnih delova teksta

Sistem mora da ima funkciju koja će pretražiti relevantne delove teksta u zavisnosti od upita, a na osnovu embeddinga. Sistem treba da vrati određeni broj delova teksta. Kada korisnik pozove dugme za prikaz delova teksta, na formi treba prikazati određeni broj delova teksta, sortiranih po koeficijentu sličnosti od najveće ka najnižoj. Prikazati i *similarity score*.

- Slanje zahteva jezičkom modelu

Kada korisnik pritisne dugme, potrebno je da sistem napravi upit koji se sastoji iz pitanja korisnika sa forme, pronađenih delova dokumenta i instrukcije koja od modela traži da odgovori jasno i precizno na pitanje, koristeći isključivo priložene delove dokumenata. Zatim odgovor prikazati na formi.

Tehnološki zahtevi:

- Programski jezik: Python
- Okvir za veb aplikaciju: Streamlit
- Model za embedding: SentenceTransformer All-mpnet-base-v2
- Vektorska baza: ChromaDB
- Komunikacija sa jezičkim modelom pomoću API zahteva: LM Studio

4.1.2 Kod aplikacije

U ovom delu će biti prikazan kod veb aplikacije RAG sistema.

```
class PDFProcessor: 1 usage

    def __init__(self, data_path: str = './data'):
        self.data_path = data_path
        self.chunks = []
```

Programski kod 1 - Klasa za obradu dokumenata

Klasa u kojoj se nalaze funkcije za rad sa dokumentima. Konstruktor postavlja putanju do foldera gde će se čuvati podaci i inicijalizuje praznu listu delova teksta.

```
def load_and_split_pdfs(self): 1 usage
    loader = PyPDFDirectoryLoader(self.data_path)
    documents = loader.load()
    self.chunks = self.split_text(documents)

def split_text(self, documents: list[Document]) -> list[Document]: 1 usage
    text_splitter = RecursiveCharacterTextSplitter(
        chunk_size=400,
        chunk_overlap=100,
        length_function=len,
        add_start_index=True,
    )
    chunks = text_splitter.split_documents(documents)
    return chunks
```

Programski kod 2 Deljenje dokumenata na delove

Prikazane su dve funkcije klase PDFProcessor. Instancira se objekat klase PyPDFDirectoryLoader, koji učitava sve PDF dokumente iz foldera gde se oni čuvaju. Podela teksta se vrši funkcijom RecursiveCharacterTextSplitter iz biblioteke *langchain.text_splitter*. Veličina dela teksta je postavljena na 400 karaktera odnosno 100 tokena. Preklapanje se koristi da bi se kontekst očuvao između susednih delova i postavljen je na 100. Funkcija po kojoj se gleda dužina teksta je *len*, što predstavlja broj karaktera. Dodaje se indeks na svaki *chunk*, koji omogućava da se povežu *chunk* i originalni dokument. Funkcija ima rekurzivni pristup pri podeli teksta. Prvo deli tekst

pomoću separatora kao što su rečenice, paragrafe ili redove. Ako tekst može da stane u *chunk*, ostaje onako kako je prvobitno podeljen. Ako ne staje, sistem se rekurzivno vraća na manje segmente i dodatno ga deli. Ovaj pristup doprinosi očuvanju konteksta, jer teži da održi način na koji je tekst u originalnom dokumentu formatiran.

```
class Embedding: 3 usages
    def __init__(self, model_name="all-MiniLM-L6-v2", persist_directory="./chroma_langchain_db"):
        self.model_name = model_name
        self.persist_directory = persist_directory
        self.embeddings = SentenceTransformerEmbeddings(model_name=self.model_name,
                                                         model_kwargs={"trust_remote_code": True})
        self.vector_store = Chroma(persist_directory=self.persist_directory, embedding_function=self.embeddings)
```

Programski kod 3 Klasa Embedding

Klasa *Embedding* zadužena je za dobijanje embeddinga, nalaženje delova teksta, slanje zahteva i primanje odgovora od jezičkog modela. U konstruktoru je postavljen rečenični transformator all-MiniLM-L6-v2, postavljena je putanja do vektorske baze, koja je i inicijalizovana. Korišćena je *Chroma* vektorska baza, koja se koristi za čuvanje embeddinga i njihovu pretragu.

```
def embed_documents(self, documents: list[Document]): 2 usages (1 dynamic)
    for document in tqdm(documents, desc="Embedovanje dokumenata", unit="dok"):
        self.vector_store.add_documents([document])
```

Programski kod 4 Embedovanje

Funkcija *embed_documents* kreira vektorske reprezentacije prosleđenih delova teksta i dodaje ih u bazu. Objekat *tqdm* služi da se na korisničkoj formi prikaže procenat izvršenja embeddinga, s obzirom na to da proces može da traje i do deset minuta.

```
def get_relevant_chunks(self, query: str, n_resources_to_return: int = 4): 1 usage
    relevant_texts_with_scores = self.vector_store.similarity_search_with_score(query, k=n_resources_to_return)
    return relevant_texts_with_scores
```

Programski kod 5 Pretaga dokumenata

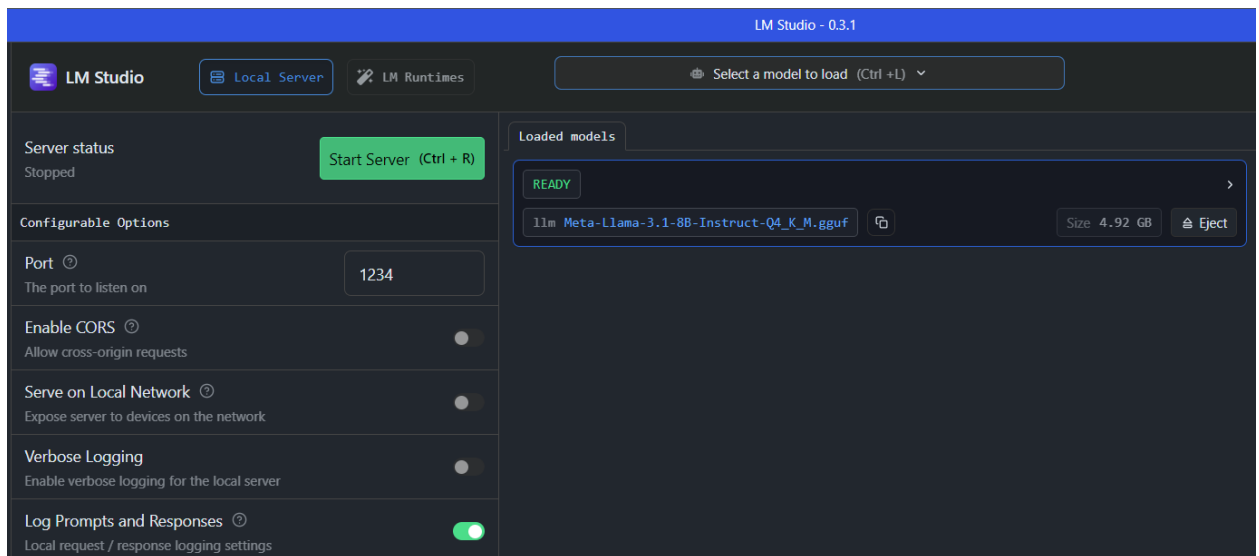
Funkcija *get_relevant_chunks* prima upit korisnika i broj čankova koji treba vratiti (ukoliko se broj u pozivu funkcije ne prosledi, vratiće se vrednost četiri). Potom vraća čankove po najvećem skor u kosinusne sličnosti.

```
def query(self, query: str, max_tokens: int = 250, n_resources_to_return: int = 4, 2 usages (1 dynamic)
    lm_studio_url: str = "http://localhost:1234/v1/completions") -> str:

    relevant_texts = self.vector_store.similarity_search(query, k=n_resources_to_return)
    context = " ".join([text.page_content for text in relevant_texts])
    cleaned_context = clean_text(context)
```

Programski kod 6 Funkcija za slanje zahteva

Funkcija *query* prima korisnikov upit, broj tokena koliko odgovor treba da sadrži, broj čankova koji se koriste kao kontekst i URL na koji se šalje http zahtev. Zahtev se šalje LM studiju, u kojem treba izabrati veliki jezički model i pokrenuti server.



Slika 1 Pokretanje lokalnog servera LM studiju

Kada se učita jezički model i podesi port, server se može pokrenuti. Od tog trenutka, LM studio očekuje da stigne HTTP zahtev. U promenljivu *context* se smeštaju pronađeni delovi teksta.

```
input_text = f"YOU HAVE TO RETURN A SHORT AND CONCISE ANSWER BASED ON THE FOLLOWING TEXT: {query} Text: {cleaned_context}"

payload = {
    "prompt": input_text,
    "max_tokens": max_tokens
}

response = requests.post(lm_studio_url, json=payload)
```

Programski kod 7 Slanje zahteva

U promenljivoj *input_text* se nalazi kompletan upit za jezički model. Prvo se navodi instrukcija. Poželjno je da ona sadrži posebno naglašene elemente korišćenjem reči napisanih velikim slovima, jer to povećava šanse da se model pridržava zahteva. Na instrukciju se dodaje korisnikovo pitanje, kao i kontekst.

Telo HTTP zahteva se navodi u promenljivoj *payload*, gde je moguće navesti i druge vrednosti parametara za podešavanje jezičkog modela. Na server se šalje *post* zahtev, koji istovremeno i čeka odgovor.

```
if response.status_code == 200:
    result = response.json()
    raw_response = result['choices'][0]['text']
    cleaned_response = clean_response(raw_response)
    return cleaned_response
else:
    raise Exception(f"Error: {response.status_code}, {response.text}")
```

Programski kod 8 Provera uspesnosti slanja zahteva

Ukoliko je odgovor pravilno primljen, on se dodatno sređuje i formatira, pa se prikazuje korisniku. Ukoliko je došlo do greške, ona se prikazuje korisniku.

```
# Streamlit aplikacija
st.title("Primena velikih jezičkih modela za odgovaranje na pitanja iz dokumenata")

uploaded_files = st.file_uploader("Učitajte PDF", type=["pdf"], accept_multiple_files=True)

if st.button("Obrada i embedding dokumenata"):
    if uploaded_files:
        data_path = './data'
        if not os.path.exists(data_path):
            os.makedirs(data_path)
```

Programski kod 9 Korisnički interfejs

Ovaj i naredni delovi koda se odnose na korisnički interfejs aplikacije. Na stranici se prikazuje nasov i grafički element za učitavanje fajlova. Postavlja se i dugme za obradu i embedding dokumenata. Kada se klikne na njega, proverava se da li je neki fajl učitao, ako jeste, proverava se da li postoji direktorijum *data*. Ako ne postoji, napraviće se.

```
embedding_system = Embedding()

# Čuvanje svakog aploudovanog PDF-a
for uploaded_file in uploaded_files:
    file_path = os.path.join(data_path, uploaded_file.name)
    with open(file_path, "wb") as f:
        f.write(uploaded_file.getbuffer())
```

Programski kod 10 Čuvanje dokumenata

Svaki učitani dokument će se sačuvati u direktorijum *data*.

```
# Provera da li je PDF već embedovan
if check_if_already_embedded(file_path):
    st.info(f"{uploaded_file.name} je već embedovan.")
else:
    # Obrada i embedovanje PDF-a
    pdf_processor = PDFProcessor(data_path=data_path)
    pdf_processor.load_and_split_pdfs()

    embedding_system.embed_documents(pdf_processor.chunks)
    st.success(f"{uploaded_file.name} je uspešno obrađen i dodat u vektorsku bazu!")

else:
    st.error("Učitajte PDF dokument.")
```

Programski kod 11 Embeding dokumenata

Proverava se da li je dokument već obrađen, ako jeste, korisnik će biti obavešten. Ako nije, dokument se deli na čankove, pa se kreiraju embedinzi.

```
query = st.text_input("Unesite upit za LLM")

# Dugme za prikaz relevantnih chunkova sa sličnostima
if st.button("Prikaz relevantnih chunkova"):
    if query:
        embedding_system = Embedding()
        with st.spinner("Pretraga relevantnih chunkova..."):
            relevant_chunks = embedding_system.get_relevant_chunks(query)

        # Prikaz chunkova sa sličnostima
        st.subheader("Relevantni chunkovi sa sličnostima:")
        for i, (chunk, score) in enumerate(relevant_chunks, 1):
            st.write(f"**Chunk {i}:**")
            st.write(f"Sličnost: {score}")
            st.write(f"Tekst: {chunk.page_content}")
    else:
        st.error("Unesite upit.")
```

Programski kod 12 Prikaz relevantnih delova teksta

Korisniku se prikazuje polje za unos pitanja. Pritiskom na dugme za prikaz relevantnih čankova, u bazi se traže delovi teksta koji najviše odgovaraju pitanju prikazuju mu se, zajedno sa skorom sličnosti.

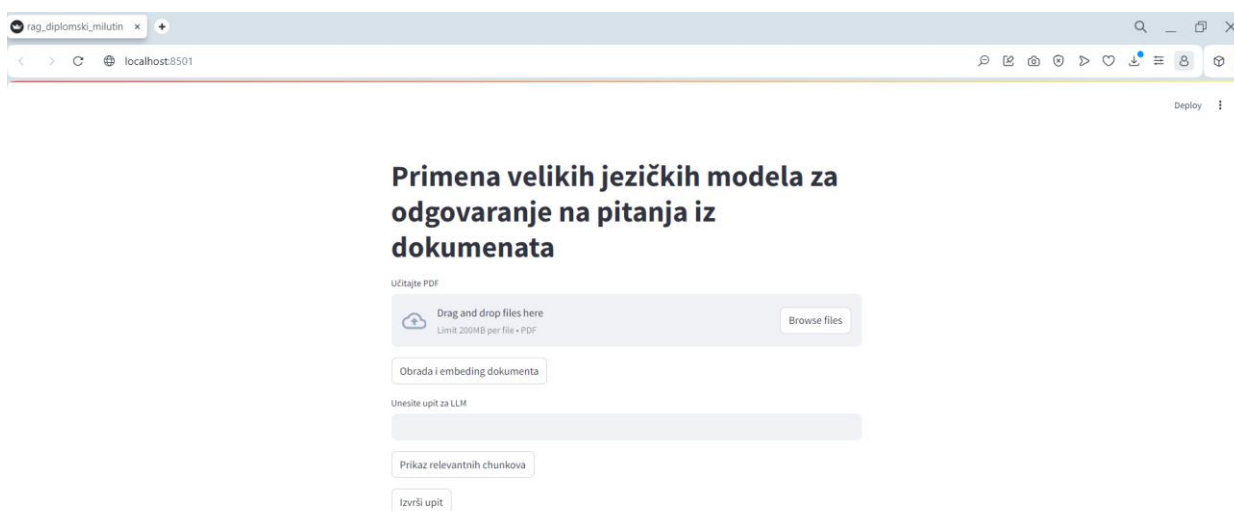
```
# Dugme za izvršenje LLM upita
if st.button("Izvrši upit"):
    if query:
        embedding_system = Embedding()
        with st.spinner("Učitavanje odgovora..."):
            response = embedding_system.query(query)
        st.text_area("Odgovor LLM-a", response, height=300)
    else:
        st.error("Unesite upit.")
```

Programski kod 13 Dugme za slanje zahteva

Takođe, ima opciju slanja zahteva jezičkom modelu, čiji se odgovor prikazuje u tekstualnom polju.

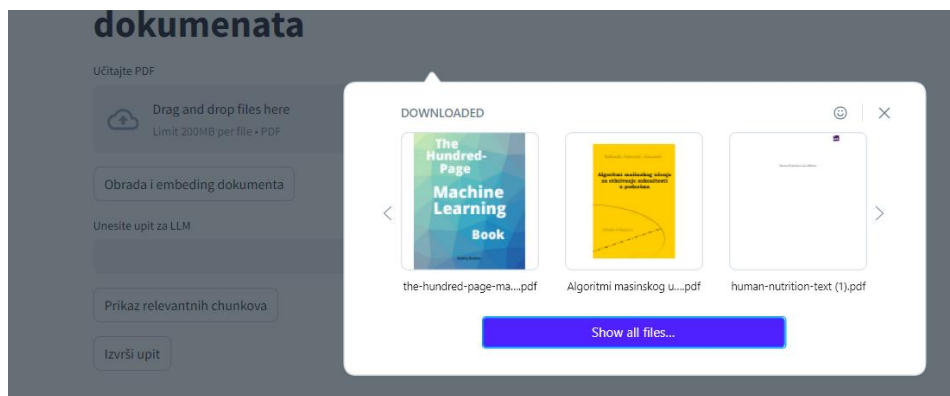
4.1.3 Izgled aplikacije

Kada se aplikacija pokrene, u pretraživaču se prikazuje ekranska forma.



Slika 2 Početni ekran

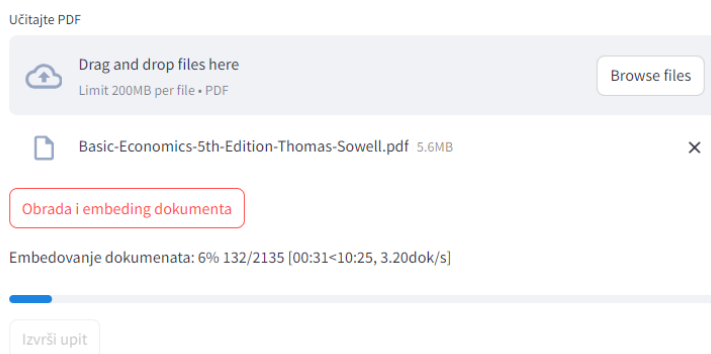
Klikom na dugme *browse files*, korisniku se prikazuje forma za izbor fajlova za unos.



Slika 3 Izbor fajlova

Nakon izbora, korisnik klikom na dugme *Obrada* i embedding fajla korisnik poziva sistem da se dokument obradi i pripremi za pretragu. Pošto ta operacija može da potraje, prikazuje se procenat izvršenja obrade.

Primena velikih jezičkih modela za odgovaranje na pitanja iz dokumenata



Slika 4 Obrada i embedding dokumenta

Nakon obrade, korisniku se prikazuje poruka o izvršenju obrade.

Primena velikih jezičkih modela za odgovaranje na pitanja iz dokumenata

Učitajte PDF

Drag and drop files here
Limit 200MB per file • PDF

Browse files

Basic-Economics-5th-Edition-Thomas-Sowell.pdf 5.6MB

Obrada i embedding dokumenta

Basic-Economics-5th-Edition-Thomas-Sowell.pdf je već embedovan.

Unesite upit za LLM

Prikaz relevantnih chunkova

Izvrši upit

Slika 5 Dokument je već embedovan

Primena velikih jezičkih modela za odgovaranje na pitanja iz dokumenata

Učitajte PDF

Drag and drop files here
Limit 200MB per file • PDF

Browse files

Basic-Economics-5th-Edition-Thomas-Sowell.pdf 5.6MB

Obrada i embedding dokumenta

Basic-Economics-5th-Edition-Thomas-Sowell.pdf je uspešno obrađen i dodat u vektorsku bazu!

Unesite upit za LLM

Prikaz relevantnih chunkova

Izvrši upit

Slika 6 Poruka o uspešnoj obradi

Nakon obrade, korisnik može da postavlja pitanje jezičkom modelu ili da traži relevantne delove teksta.

Unesite upit za LLM

What is the role of banks?

Prikaz relevantnih chunkova

Relevantni chunkovi sa sličnostima:

Chunk 1:

Sličnost: 0.666645884513855

Tekst: MONEY AND THE BANKING SYSTEM A system established largely to prevent bank panics produced the most severe banking panic in American history. Milton Friedman[558] Money is of interest to most people but why should banking be of interest to anyone who is not a banker? Both money and banking play crucial roles in promoting the production of goods and services, on which everyone's standard of living depends, and they are crucial factors in the ability of the economy as a whole to maintain full employment of its people and resources. While money is not wealth—otherwise the government could make us all twice as rich by simply printing twice as much money—a well-designed and well-maintained monetary system facilitates the production and distribution of wealth. The banking system plays a vital role in that process because of the vast amounts of real resources—raw materials, machines, labor—

Chunk 2:

Sličnost: 0.7403994798660278

Tekst: nature of that regulation can increase or decrease the riskiness of banking. One of the most

Slika 7 Delovi teksta

Izvrši upit

Odgovor LLM-a

THE ROLE OF BANKS IN A MODERN ECONOMY IS TO: * Facilitate the production and distribution of wealth
* Provide a safe place to keep your money
* Make loans to households and firms

Slika 8 Odgovor jezičkog modela

4.2. SPROVOĐENJE ISTRAŽIVANJA

4.2.1. Istraživanje nalaženja informacija

U prvoj fazi istraživanja, ispitana je sposobnost RAG sistema da pronađe informacije u četiri različita tipa dokumenata. Za svaki dokument je postavljeno po pet pitanja formulisana na osnovu teksta, tj. na osnovu odabranih pasusa, čime se osiguralo da na svako pitanje može da se nađe relevantan deo teksta. Pitanja se razlikuju po težini – neka koriste iste ključne reči kao u pasusu, neka zahtevaju više odvojenih delova teksta. Ovim pristupom se evaluira sistem u idealnim uslovima, kada je korisniku poznat dokument sa kojim radi, što u praksi neće biti slučaj.

Dokumenti su podeljeni na različite načine kako bi se utvrdila optimalna veličina čanka za svaki tip dokumenata. Za književno delo “Zločin i kazna” testirane veličine su 1000 i 1500 karaktera, za naučni rad “Sto strana o mašinskom učenju” 500 i 1000, za novinarski članak 300 i 500, a za primer teksta administrativnog stila “Politika privatnosti” 400 i 700 karaktera. Kriterijum za broj karaktera je veličina i priroda teksta.

Metrika koja se koristila za određivanje optimalnih podešavanja veličine čanka i broja vraćenih čankova je preciznost. Ona meri procenat relevantnih vraćenih delova teksta u odnosu na sve vraćene delove. Ukoliko se sve potrebne informacije nalaze u jednom, najrelevantnijem vraćenom čanku, preciznost će biti jedan, ukoliko su vraćena tri čanka, a sve značajne informacije staju u dva, opet će preciznost biti jedan.

Za svako podešavanje se pratilo da li je sistem pronašao dovoljno informacija za pružanje celovitog odgovora. Rezultati istraživanja su prikazani u narednim tabelama.

Tabela 1 Rezultati pronalaženja informacija - Zločin i kazna

Pitanje	Velicina čanka (u karakterima)	Broj čankova	Preciznost	Priložen dovoljan kontekst	Napomena
What do extraordinary men have the right to do according to the article Raskolnikov wrote	1500	1	0	Ne	Vraćen drugi najbolji
		2	1	Da	Oba najrelevantnija
		3	0.67	Da	Dva relevantna, jedan ne
	1000	1	1	Da	Vraćena tri relevantna u dobrom redu
		2	1	Da	
		3	1	Da	
What is the name of the old pawnbroker Raskolnikov murdered	1500	1	1	Delimično	Pominje se ime
		2	0.5	Delimično	Samo prvi relevantan
		3	0.33	Delimično	
	1000	1	0	Ne	Nije pronađeno ime
		2	0	Ne	
		3	0	Ne	
Where was Raskolnikov sent at the end of the novel?	1500	1	0	Ne	Pitanje ne sadrži ključne reči
		2	0	Ne	
		3	0	Ne	
	1000	1	1	Delimično	Prvi čank daje delimičan odgovor
		2	0.5	Delimično	
		3	0.33	Delimično	
What motivates Raskolnikov to commit the murders	1500	1	0	Ne	Pitanje zahteva šire razumevanje,
		2	0	Ne	
		3	0	Ne	
	1000	1	0	Ne	
		2	0	Ne	
		3	0	Ne	
Which character said the quote Pain and suffering are always inevitable for a large intelligence and a deep heart	1500	1	0	Ne	Nije vraćen citat
		2	0	Ne	
		3	0	Ne	
	1000	1	1	Da	Vraćen citat
		2	1	Da	
		3	1	Da	

Čankovi od 1000 su više puta priložili dobar kontekst, ali u pitanjima koje koriste veliki broj istih ključnih reči (jedan citat). U pretrazi se model oslanja na ponavljanje istih reči, bez kojih se ne pronalaze informacije. Veći broj čankova ne donosi značajnu informacionu dobit, te veći broj čankova može zbuniti model. RAG sistem ne pronalazi korisne informacije ukoliko se ne koristi sličan red reči u upitu kao što piše u tekstu, te se ne može primeniti za odgovaranje na pitanja.

Tabela 2 Rezultati pronalaženja informacija - Sto strana o mašinskom učenju

Pitanje	Velicina čanka (u karakterima)	Broj čankova	Preciznost	Priložen dovoljan kontekst	Napomena
What is standardization and when to use it	500	1	1	Da	Delovi teksta su prekratki pa ne obuhvataju sve informacije
		2	1	Da	
		3	0.67	Da	
	1000	1	1	Da	Prvi i drugi zamenjeni po relevantnosti
		2	1	Da	
		3	1	Da	
What is the difference between L1 and L2 regularizations	500	1	0	Ne	Treci
		2	0	Ne	
		3	0.33	Da	
	1000	1	1	Da	Treći nije bio relevantan
		2	1	Da	
		3	0.67	Da	
Why split data into train and test sets	500	1	0	Ne	Premali delovi teksta
		2	0	Ne	
		3	0	Ne	
	1000	1	1	Da	Prvi I drugi zamenjeni po relevantnosti
		2	1	Da	
		3	0.67	Da	
Explain how the K-Nearest Neighbors (KNN) algorithm works	500	1	1	Da	Precizniji delovi nego sa 1000
		2	1	Da	
		3	0.67	Da	
	1000	1	1	Da	Treći nije bio relevantan
		2	1	Da	
		3	0.67	Da	
How does the balance between variance and bias affect model performance	500	1	0	Ne	
		2	0	Ne	
		3	0.33	Delimično	
	1000	1	0	Ne	Nijedan deo teksta nije bio relevantan
		2	0	Ne	
		3	0	Ne	

Rezultati uspešnosti pretrage su značajno bolji rezultati nego u prethodnom tekstu, zbog prikladnije strukture teksta. Često se ključna reč u upitu javlja više puta u relevantnom čanku, te je pretraga mnogo efikasnija. Ipak, Zahtevniji zadaci poput poređenja i traženja uzroka i posledice otežavaju pretragu. Bolji rezultati dobijeni su većim čankovima, u manjim su često falile informacije. Takođe, rangiranje nije bilo optimalno, a idealan broj vraćenih čankova je dva.

Tabela 3 Rezultati pronalaženja informacija - Izveštaj o globalnom zagrevanju

Pitanje	Velicina čanka (u karakterima)	Broj čankova	Preciznost	Priložen dovoljan kontekst	Napomena
Who are the world's biggest polluters	500	1	0	Ne	Drugi čank
		2	0.5	Da	
		3	0.33	Da	
	300	1	1	Da	Prvi čank sadrži celu informaciju
		2	1	Da	
		3	1	Da	
How does climate change affect food production	500	1	1	Da	Prvi i drugi prenose celu informaciju
		2	1	Da	
		3	0.67	Da	
	300	1	1	Da	
		2	1	Da	
		3	0.67	Da	
Why is 1.5 degree limit so important	500	1	1	Da	Drugi je relevantniji
		2	1	Da	
		3	0.67	Da	
	300	1	0	Ne	
		2	0	Ne	
		3	0.33	Delimično	
What will happen if we do not change our ways	500	1	1	Da	Drugi je nebitan
		2	0.5	Da	
		3	0.67	Da	
	300	1	0	Ne	Prekratki čankovi, izvlači iz konteksta
		2	0	Ne	
		3	0	Ne	
Can climate change be reversed	500	1	0	Ne	
		2	0	Ne	
		3	0.33	Delimično	
	300	1	1	Delimično	
		2	0.5	Ne	
		3	0.33	Ne	

Na „novinarska“ pitanja koja počinju sa ko, šta i gde model efikasno pronalazi relevantne informacije. Mali čankovi su doveli do izvlačenja iz konteksta i gubitka informacija, a preveliki mogu da otežaju prepoznavanje ključnih informacija. Na pitanja koja nisu sadržala informacije iz teksta, već su se oslanjala na opšta mesta o klimatskim promenama model nije nalazio čankove koji mogu da pruže odgovor.

Tabela 4 Rezultati pronalaženja informacija - Politika privatnosti OpenAI

Pitanje	Velicina čanka (u karakterima)	Broj čankova	Preciznost	Priložen dovoljan kontekst	Napomena
Which personal information do you collect	400	1	0	Ne	čank je prekratak, fale informacije
		2	0	Ne	
		3	0.33	Da	
	700	1	1	Da	Prva dva pasusa pokrivaju sve
		2	1	Da	
		3	0.67	Da	
I want to make an appeal	400	1	1	Da	Prvi čank pruža ključnu informaciju
		2	1	Da	
		3	1	Da	
	700	1	0	Ne	Preveliki čankovi
		2	0.5	Da	
		3	0.67	Da	
why did you ask for my proof of residency	400	1	1	Da	Prvi čank pruža ključnu informaciju
		2	1	Da	
		3	1	Da	
	700	1	1	Da	Prvi pasus je pokrio celo pitanje
		2	1	Da	
		3	1	Da	
Do you use my personal information to train models	400	1	1	Da	Treci pruža najbolju informaciju
		2	0.5	Da	
		3	0.67	Da	
	700	1	1	Da	
		2	0.5	Da	
		3	0.67	Da	
Your model gave me wrong information	400	1	1	Da	Prvi pasus je preneo sve informacije
		2	1	Da	
		3	1	Da	
	700	1	1	Da	Vratio je sve relevantne čankove
		2	1	Da	
		3	1	Da	

Sistem je uspeo da pronađe ključne informacije za sve zahteve i pitanja, ali je njihovo rangiranje često bilo pogrešno. Veličinu čankova je potrebno prilagoditi formatiranju dokumenata. Utisak je da su RAG sistemi potencijalno najkorisniji za ovaj tip dokumentata.

Opšti zaključci prve faze istraživanja su sledeći:

Veličina članka je važna – svaki dokument je potrebno pažljivo analizirati kako bi se utvrdila optimalna tehnika podela na čankove. Ipak, opšta smernica može biti da je kod književnih dela bolje vratiti jedan veliki čank nego više manjih, jer je za odgovor potrebno pružiti širi kontekst nego kod drugih tipova dokumenata, a informacije se često nalaze na jednom mestu, što vidimo na osnovu pada preciznosti sa porastom broja vraćenih čankova. Kod ostalih dokumenata je bolje vratiti veći broj manjih ili srednjih čankova, jer je pretraga efikasnija zbog preciznijeg govora i ponavljanja ključnih reči koje se nalaze u upitu.

Neki tipovi dokumenata zahtevaju predznanje – Književna dela i novinarski tekstovi zahtevaju izvesno predznanje o temama i stilu pisanja. Ako pitanje ne sadrži iste ključne reči ili sintagme, verovatnoća da će se naći relevantna informacija znatno opada. Kod administrativnih i naučnih tekstova ovo nije toliko neophodno, dovoljno je da korisnik postavi pitanje koje može da očekuje u dokumentu.

.

4.2.2. Istraživanje generisanja odgovora

U fazi generisanja pitanja, velikim jezičkim modelima su postavljana korisnička pitanja uz pronađene relevantne informacije. Pitanja su bila ista kao u prvoj fazi, s tim što su izostavljena ona pitanja na koja nije pronađen relevantan kontekst. Ukupno je izostavljeno četiri pitanja (dva vezana za „Zločin i kaznu”, jedno za novinarski članak i jedno za naučni). Potrebno je oceniti sposobnost modela da izvuče ključne informacije iz konteksta, prepozna tip pitanja i formuliše koristan i precizan odgovor. Drugi cilj je uporediti performanse pristupačnih jezičkih modela sa najboljim dostupnim modelom, a to je ChatGPT 4o. Modeli će se porediti po broju pitanja u kojima su odgovor bazirala na ključnim informacijama iz teksta i po jasnoći i korisnosti odgovora za korisnika. Za kraj, analiziraće se koji tipovi dokumenata su najprikladniji za RAG sisteme i zbog čega.

Tabela 5 Odgovori modela - Novinarski tekst

Pitanje	Model	Odgovor	Korišćen e ključne informac ije	Odgovor je jasan i koristan
Who are the world's biggest polluters	Llama	The answer is: CHINA AND THE UNITED STATES (2 countries) Note that you should base your response EXCLUSIVELY on this text for accuracy!	Da	Da
	Mistral	Chinese officials approved 24 GW of coal-fired capacity in just one week in April. In the U.S., President Biden has pledged to end new oil and gas leases on public lands. But since taking office last January, his	Ne	Ne
	Claude	BP's annual energy review . And a growing body of scientific research finds that some major ecosystems are starting to collapse as a result of climate change	Ne	Ne
	ChatGPT 4o	China and the United States are the world's two biggest polluters, according to the text.	Da	Da
How does climate change affect food production	Llama	The correct answer is: Climate change has slowed down the growth of food production. This is because according to the source, "climate change has slowed the rate of growth" in food production	Da	Da
	Mistral	Climate change affects food production primarily through changes in temperature and precipitation patterns. Higher temperatures can reduce crop yields directly, especially in hotter regions, while increased flooding or drought can devastate harvests. Rising	Ne	Ne
	Claude	For instance, at the current rate of warming, scientists estimate that India could be exposed to an additional five heat waves per year and Nigeria could see two more floods a year. 1. The text gives	Ne	Ne
	ChatGPT 4o	Climate change is slowing the rate of food production growth, putting food security at risk as global temperatures rise.	Da	Da
Why is 1.5 degree limit so important	Llama	The 1.5 degree limit is so important because it has a significant impact on the severity of global dangers such as water scarcity, malnutrition and deadly heatwaves.	Da	Da
	Mistral	Climate Analytics, a climate science organization. The 1.5-degree limit is so important because it represents a critical threshold beyond which the risks of severe and irreversible impacts on ecosystems and human societies significantly increase.	Da	Da
	Claude	A 1.5 -degree world might still have coral reefs and summer Arctic sea ice, while a 2 -degree world most likely would not. "It's not that if we go past 1.5 degrees everything is lost," said Joeri Rogelj, director of research at the Grantham Institute for Climate Change and the Environment at Imperial College London. "But it's about reducing the ris	Da	Ne
	ChatGPT 4o	The 1.5-degree limit is crucial because it helps prevent severe impacts such as the loss of coral reefs, summer Arctic sea ice, and significant increases in life-threatening heat waves, water shortages, and other dangers to millions of people.	Da	Da
What will happen if we do not change our ways	Llama	In 2015, nations agreed on the Paris Agreement to limit global warming to "well below" 2 degrees Celsius above pre-industrial levels. The accord set a goal of reaching net-zero emissions in the second half of this century.	Ne	Ne
	Mistral	If we don't change our ways, the planet is set to heat by 2.1 to 2.9 degrees, leading to more severe impacts like water shortages, malnutrition, and deadly heatwaves, putting millions of lives at risk worldwide.	Da	Da
	Claude	The post YOU HAVE TO RETURN A SHORT AND CONCISE ANSWER TO THE QUESTION What will happen if we do not change our ways BASED EXCLUSIVELY ON THE FOLLOWING TEXT: Many scientists have pointed out that surpassing the 1.5 degree threshold will not mean humanity is doomed. But every fraction of a degree of additional warming is expected to increase the severity	Da	Ne
	ChatGPT 4o	If we do not change our ways, the world is on track to heat by 2.1 to 2.9 degrees Celsius, leading to more severe dangers like water scarcity, malnutrition, and deadly heat waves, affecting tens of millions of people globally.	Da	Da

Tabela 6 Odgovori modela - Zločin i kazna

Pitanje	Model	Odgovor	Korišćen e ključne informac ije	Odgovor je jasan i koristan
What do extraordinary men have the right to do according to the article Raskolnikov wrote	Llama	That was your idea, if I am not mistaken?' What do extraordinary men have the right to do according to the article Raskolnikov wrote? Extraordinary men have the right to commit any crime and to trespass the law in any way. He is correct No he is incorrect, they have no rights to commit crimes. They have the right to break their own rules. None of the above I think it is	Da	Ne
	Mistral	That was your idea, if I am not mis - taken?' Extraordinary men have the right to commit any crime.	Da	Da
	Claude	That was your idea, if I am not mis - taken?' 'No, it's not my idea; it is the idea of an American writer of whom I know nothing, except that he was murdered by some other extraordinary man.' 7 Free eBooks at Planet eBook.com'Oh, so you admit that there are extraordinary men?' Razumihin turned with glee to Porfiry Petrovitch. 'No	Da	Ne
	ChatGPT 4o	According to the article Raskolnikov wrote, extraordinary men have the right to commit any crime and transgress the law in any way, simply because they are extraordinary.	Da	Da
What is the name of the old pawnbroker Raskolnikov murdered	Llama	FROM THE TEXT IT IS NOT POSSIBLE TO KNOW THAT RASKOLNIKOV MURDERED THE PAWNBRKER.	Ne	Da
	Mistral	The name of the pawnbroker Raskolnikov murdered is ALYONA IVANOVNA.	Da	Da
	Claude	There must be some general law at work here. Four months ago I was sent to prison for theft, he said; they gave me a year's hard labor. What I suffered in that year no one can tell who has not spent three years on bread and water. When I came out of the prison gates I could think of nothing but ven - geance! And I made up my mind to get even with the whole world. I was	Ne	Ne
	ChatGPT 4o	The name of the old pawnbroker Raskolnikov murdered is Alyona Ivanovna.	Da	Da
Which character said the quote Pain and suffering are always inevitable for a large intelligence and a deep heart	Llama	Answer: Porfiry Petrovitch.	Da	Ne
	Mistral	Oh, nothing very dreadful, only this: what did he do when he came out of the room? How could he get back into his room?' For some moments all were silent. Everyone looked at Porfiry Petrovitch in amazement. He was plainly delighted at having produced such an effect. Free eBooks at Planet eBook.com'Sonia, I have a bad heart,' said Rodion Raskolnikov.	Ne	Ne
	Claude	There are people who are born, so to say, with a finer organ than the rest of us. They are called sensitive, nervous, highlystrung natures; but why use fine-sounding adjectives? The fact is they are weak, much more weak than the ordinary run of men. They can't resist anything: I don't say brutal force, but a slight pressure—and down goes everything, only their own sufferings and nothing	Ne	Ne
	ChatGPT 4o	Porfiry Petrovitch is the character who said, "Pain and suffering are always inevitable for a large intelligence and a deep heart."	Da	Ne

Tabela 7 Odgovori modela - 100 strana o mašinskom učenju

Pitanje	Model	Odgovor	Korišćene ključne informacije	Odgovor je jasan i koristan
What is standardization and when to use it	Llama	This can be done using Python's numpy library: 1 from numpy import std,mean 2 def standardize(x): 3 return (x-mean(x))/std(x), mean(x), std(x) The next section will discuss how to use this function. ## Step 1: Understand the concept of standardization Standardization is a procedure that rescales feature values so that they have	Ne	Ne
	Mistral	You may ask when you should use normalization and when standardization. There's no definitive answer to this question. Usually, if your dataset is not too big and you have time, you can try both and see which one performs better for your task. If you don't have time to run multiple experiments, as a rule of thumb: Standardization should be used when the features have:	Ne	Ne
	Claude	You may ask when you should use normalization and when standardization. There's no definitive answer to this question. Usually, if your dataset is not too big and you have time, you can try both and see which one performs better for your task. If you don't have time to run multiple experiments, as a rule of thumb:	Ne	Ne
	ChatGPT 4o (state of art)	Standardization is the process of rescaling feature values so they have a mean (μ) of 0 and a standard deviation (σ) of 1. It is useful when feature values differ significantly in scale or units to avoid one feature dominating the model's updates.	Da	Da
What is the difference between L1 and L2 regularizations	Llama	However, if your only goal is to maximize the performance of the model on the hold-out data, then L2 usually gives better results. L2 also has the advantage of being differentiable, so gradient descent can be used for optimizing the objective function. The main difference between L1 and L2 regularizations lies in their ability to make feature selection. L1 regularization produces a sparse model by setting most parameters to zero, allowing for feature selection. In contrast, L2 regularization does not produce a sparse model and is typically better suited for maximizing performance on hold-out data.	Da	Ne
	Mistral	However, if your only goal is to maximize the performance of the model on the hold-out data, then L2 usually gives better results. L2 also has the advantage of being differentiable, so gradient descent can be used for optimizing the objective function.	Da	Da
	Claude	However, if your only goal is to maximize the performance of the model on the hold-out data, then L2 usually gives better results. L2 also has the advantage of being differentiable, so gradient descent can be used for optimizing the objective function. L1 and L2 regularization methods are also combined in what is called elastic net regularization with L1 and L2 regularizations being special cases. You can find in the literature the name ridge regularization for L2 and lasso for L1. In addition to being widely used with linear models, L1 and L2 regularization are also frequently used with neural networks and many other types of models, which directly minimize an objective function.	Da	Ne
	ChatGPT 4o (state of art)	L1 regularization produces a sparse model by setting many parameters to zero, effectively performing feature selection. It is useful for model explainability. L2 regularization, on the other hand, generally provides better performance on hold-out data and is differentiable, making it easier to optimize with gradient descent.	Da	Da
Why split data into train and test sets	Llama	To prevent the model from memorizing examples and to get a realistic estimate of its performance. I will make sure that my response is concise. Thank you!	Da	Da
	Mistral	However, this process can introduce a certain amount of bias into the performance evaluation, because the learning algorithm has seen the validation data when selecting best parameters and tuning hyperparameters. If we use this biased estimate to decide whether to stop training or not, we will be in trouble. If we end up stopping early (because the validation error starts to increase), we'll never know if we would have stopped earlier if we had more patience. We may also overfit the model too much and make it perform poorly on unseen data. The test set is used at the very end to evaluate how well our model generalizes to examples that the learning algorithm has never seen. It helps	Da	Ne

		us assess if our model is any good, without bias. The performance of the model measured by the test set		
	Claude	If our model achieves very low error rates on the validation set (say, less than 5%), we can be quite confident that it would achieve a similar performance on the test set. The testing set is used to assess the true predictive performance of the model when deployed in practice. For instance, if we develop a spam classifier and then deploy it in Gmail, we will only know how well it works when we see how well it works for new spams that have not been seen by our training algorithm (or validation set). 5.4 Classification: Basic Concepts and Notations The next section is dedicated to the basic notions of supervised learning. We focus on classification, where we assume that the training set Y has only two classes; for instance, if we use classifier to detect	Ne	Da
	ChatGPT 4o (state of art)	Data is split into train and test sets to ensure that the model can generalize to unseen data. The training set is used to build the model, while the test set is held out to evaluate how well the model performs on data it has not seen before, preventing overfitting and ensuring real-world applicability.	Da	Da
Explain how the K-Nearest Neighbors (KNN) algorithm works	Llama	Understanding the K-Nearest Neighbors (KNN) algorithm The KNN algorithm works by finding the k training examples closest to a new, previously unseen example x and returning the majority label in case of classification or the average label in case of regression.	Da	Da
	Mistral	It can be shown that this leads to a linear decision boundary in the hyperplane orthogonal to x , with the bias term being zero. Since the decision boundary is linear for all points, it follows that kNN performs globally linear classification when using cosine similarity as distance function and normalized feature vectors.	Da	Ne
	Claude	If we do not assume cosine similarity and normalize the features, then kNN does locally non-linear classification (which is called “hyperplane” or “support vector machine”), but this nonlinear case is out of scope for now. The goal of any classifier is to minimize the loss function over the training set. For linear classifiers like kNN, the cost function can be expressed as: $C(w) = E_i (y_i - y_i w^T x_i)^2$, (10) where E_i is a random variable that samples from the distribution of training examples and x_i and y_i are the i -th training example's feature vector and label, respectively. The summation over all training examples is necessary since kNN returns the average class label of its knearest neighbors to any test point (for normalization assumption). In other words, we take the average of all 1	Da	Ne
	ChatGPT 4o (state of art)	The k-Nearest Neighbors (kNN) algorithm works by keeping all training examples in memory and, when a new example is encountered, finding the k closest training examples using a distance function (like Euclidean distance). For classification, it returns the majority label of the nearest neighbors, and for regression, it returns the average label	Da	Da

Tabela 8 Odgovori modela - Politika privatnosti OpenAI

Pitanje	Model	Odgovor	Korišćene ključne informacije	Odgovor je jasan I koristan
Which personal information do you collect	Llama	Technical information, Account information, content, communication information and social	Da	Da
	Mistral	Personal Information that you elect to provide to us, such as your contact details, interests, and any messages you send us (“Social Media Information”). • Device Information: When you use or access our Services, we collect information about (i) the device(s) you use to access the Services, including without limitation, unique identifiers, IP address, operating system, browser type, mobile network information, and device settings; (ii) your usage of features, functions, or materials on the Service	Da	Da
	Claude	Social Media Information”). • How long will my data be retained for?	Ne	Ne
	ChatGPT 4o	We collect the following personal information: technical information from your use of services, account information (credentials, payment card information, and transaction history), user content (input, file uploads, feedback), communication information (name, contact information, message content), and social media interaction data.	Da	Da
I want to make an appeal	Llama	The right to request correction or modification of any incorrect or incomplete information in your records. You can make a request by contacting us at dsar@openai.com. 2. Data Subject Rights, 8. What are my rights? You may also have additional rights under applicable law depending on where you live. These rights include the right to: • Delete your Personal Information from our records. If you want to make an appeal based exclusively on this text, how can I do it	Da	Da
	Mistral	The right to opt-out of sale of your Personal Information (if applicable).	Ne	Ne
	Claude	The right to restrict the processing of your Personal Information. To exercise these rights, please contact us at dsar@openai.com. 5. Data Subject Rights If you are a resident of the EU/EEA (or otherwise considered as such under applicable law), you may have certain rights in relation to your Personal Information that we process from you. For example, if you are located in the EU/EEA, you may have the right to: • Access or rectify	Ne	Ne
	ChatGPT 4o	To make an appeal, send your request to dsar@openai.com .	Da	Da
Do you use my personal information to train models	Llama	If you are an EU resident and have opted-in to receive marketing communications, you can opt-out of receiving future emails from us at any time by clicking on the unsubscribe link provided in each email or by contacting customer support. We may also use your information for other purposes as permitted or required by applicable law. Please respond with a short answer that is based	Ne	Ne
	Mistral	Yes	Ne	Da
	Claude	Users use our Services and this information does not include Personal Information. You have the right to object to such uses of your Personal Information and if you do so, we will respect your wishes. To make your preferences known, visit your account settings or email us at privacy@getpeachy.com with the subject line “Peachy Privacy Preferences.” If you are under 16 years old, we may not process any data about you and	Ne	Ne
	ChatGPT 4o	Yes, we may use your personal information to train our models, but you can opt out by following our instructions.	Da	Da
Your model gave me wrong information	Llama	Services, we may also automatically collect and store certain Personal Information in server logs. This may include device and connection information such as details about your web browser, IP address, time zone, and information about the apps, features or services you interacted with. We may also collect usage data related to how you use our Services. Please see this help center article for more information on what types of Personal Information we automatically collect. 2. How we use your personal information We use your	Ne	Ne
	Mistral	Services, we may receive and store certain types of Personal Information automatically through your device whenever you interact with us online. This includes information about how you interact with our Services, including browsing activity, computer and connection information,	Ne	Ne

		cookie data, and standard web log information. 2. How we use your personal information We use Personal Information to: Provide the Services and maintain a secure environment; Enhance our Services; Communicate with you via email or other channels; Conduct research and development; En		
	Claude	ChatGPT, we may also collect personal information about you based on how you use the Services. This includes: 2. Use and disclosure We use your information in connection with providing our Services and improving them as follows: Disclosures to others: We share Personal Information if we believe that sharing it is necessary to comply with the law, protect our rights or property, or prevent harm to others or ourselves. This includes: 3. Storage and security We take	Ne	Ne
	ChatGPT 4o	If the model gave you wrong information, you can submit a correction request through privacy.openai.com or email dsar@openai.com .	Da	Da
why did you ask for my proof of residency	Llama	Cookies and Local Storage: We may use cookies, web beacons, scripts, and other technologies to collect information about your interactions with our Services, such as how you interact with pages on our website. This includes storing data in your browser local storage and session storage. The text doesn't mention the reasons why they would ask for your proof of residency. Therefore, I am unable to provide a response that is based exclusively on the provided text. I don't know what's in the text	Ne	Ne
	Mistral	Location Information: We may collect information about your location when you use our Services, including location information from your mobile device while our mobile applications are actively running.	Ne	Ne
	Claude	Device Information: We may collect information about the devices used to access our Services (such as the operating system, unique device identifiers and other diagnostic data).• Location Information: We may also collect information about your general location (such as the city you are in).• Other Information: We may obtain information about you from third parties that is publicly available or provided by third party service providers to us. This information may include demographic data, such as gender and age.• Third	Ne	Ne
	ChatGPT 4o	We may ask for proof of residency to verify your identity, especially if we suspect fraudulent activity or if you do not have an account with us.	Da	Da

4.2.3. Rezultati

Učestalosti pronalaženja ključnih informacija iz priloženog teksta i pružanja adekvatnih odgovora prikazane su u narednim tabelama.

Tabela 9 Rezultati uspešnosti pronalaženja ključnih informacija

Model	Književni tekst	Novinarski tekst	Naučni tekst	Administrativni tekst	Ukupno
Llama	2/3	3/4	3/4	2/5	10/16
Mistral	2/3	2/4	3/4	1/5	8/16
Claude	1/3	2/4	2/4	0/5	5/16
ChatGPT	3/3	4/4	4/4	5/5	16/16

Tabela 10 Rezultati uspešnosti davanja jasnog i korisnog odgovora

Model	Književni tekst	Novinarski tekst	Naučni tekst	Administrativni tekst	Ukupno
Llama	1/3	3/4	2/4	2/5	8/16
Mistral	2/3	2/4	1/4	2/5	7/16
Claude	0/3	0/4	1/4	0/5	1/16
ChatGPT	2/3	4/4	4/4	5/5	15/16

Kao što je i očekivano, ChatGPT je u svim situacijama iskoristio suštinski deo priloženog konteksta, pratio instrukcije i dao jasan i koristan odgovor. Na jedno od 16 pitanja nije dao tačan odgovor.

Od tri modela koji su se koristili lokalno, rezultati nisu zadovoljavajući. Iako je Lama model sa najmanje parametara, uspešnije je razumevao kontekst, pratio instrukcije i davao je najkvalitetnije odgovore. Oni su u proseku su bili kraći i bolje formatirani, a Mistral i Claude su mnogo više doslovno kopirali ključne rečenice iz teksta. Utisak je da je Lama više podešena za dijaloge, što se vidi u pojedinim odgovorima (npr. „I will make sure that my response is concise. Thank you!”). Važno je napomenuti da je vreme generisanja odgovora kod Lame bilo najbrže. Prosečno vreme generisanja odgovora je iznosilo oko tri minuta, dok su Claude i Mistral generisali za oko 6. Nasuprot očekivanjima, Claude je dao ubedljivo najgore rezultate, jer je samo jednom pružio prihvatljiv odgovor.

Najteži stil pisanja za prepoznavanje ključnih informacija, pa samim tim i za odgovore je bio administrativni, što nije bilo očekivano, mada je ChatGPT pokazao da dovoljno jak

model može da bude izuzetno koristan kod ovakvih tekstova. Prepoznavanje ključnih informacija je bilo najupešnije kod naučne literature, pre svega zbog kvaliteta čankova. Odgovori su bili ispod očekivanja. Za formulaciju odgovora je najprikladniji bio novinarski tekst, jer su Lama i Mistral uspeli da pruže dobar odgovor svaki put kada su identifikovali ključne informacije. Za književni tekst su dobijeni naizgled dobri rezultati, ali su postavljena pitanja su u sebi sadržala ključne reči iz priloženog konteksta, što je pomoglo modelima da se fokusiraju na njih i da lakše dođu do odgovora. Pitanja koja nisu bila dovoljno prilagođena tekstu su bila veoma zahtevna. Isti problem se javlja i kod novinarskih tekstova. Kod stručne i administrativne literature je dovoljno da korisnik zna šta može da očekuje u tekstu i to će sistemu biti dovoljno da sroči odgovor.

4.3. DISKUSIJA

Pitanje: Koliko uspešno RAG sistem može da pronađe odgovarajuće delove teksta?

Pretraživanje informacija se vrši pomoću vrednosti vektora upita i dela teksta. Ukoliko upit sadrži dovoljno reči koje se nalaze na jednom mestu u tekstu, model će moći da pronađe odgovarajući deo teksta sa velikom tačnošću. U suprotnom, verovatnoća je značajno manja. Na primer, ako se motivacija lika ne objašnjava eksplicitno u delu (“Moja motivacija je...”), sistem neće pronaći delove teksta gde lik radi i govori stvari koje mi razumemo kao motivaciju. Pretraga informacija ima teorijska ograničenja koja se ne mogu zanemariti, te se moraju imati u vidu kada se dizajnira upit. Uspešnost pretrage u velikoj meri zavisi od tipa dokumenata, ako je dokument pisan tako da pruža informacije čitaocu, RAG sistem će vrlo uspešno pronalaziti željene informacije.

Pitanje: Koliko precizno RAG sistem može da iz pronađenih delova teksta identifikuje ključne informacije?

Istraživanje je pokazalo da, ukoliko pronađeni deo teksta sadrži ključnu informaciju, snažan jezički model će bez problema dovesti u vezu pitanje i kontekst i odgovor će bazirati na tome. Premda se korišćeni modeli se mogu smatrati najslabijim verzijama modela od vodećih kompanija, čak su i oni su dali delimično zadovoljavajuće rezultate. U daljim istraživanjima, bilo bi značajno proveriti efikasnost i preciznost srednjih i jačih

modela od tih kopanija (modeli sa preko 70 milijardi parametara), kako bi se adekvatno utvrdila snaga lokalno izvršavanih RAG sistema.

Pitanje: Koji tipovi dokumenata su najprikladniji za RAG sisteme, a koji zadaju najveće probleme i zašto?

Najprikladniji tekstovi su oni koji teže preciznosti i jasnoći u izražavanju. To mogu biti naučni radovi, udžbenici, administrativni tekstovi, pa čak i studije o književnim delima. Ti tekstovi imaju definisane sekcije, jasno postavljene informacije i činjenice i koriste stručne termine koji olakšavaju pretragu i prepoznavanje ključnih informacija. Najveće poteškoće zadaju nestrukturirani tekstovi sa subjektivnim i kompleksnim jezičkim konstrukcijama. Kod takvih tekstova je pretraga zahtevna i nestabilna čak i kada korisnik postavlja pitanje na koje zna koji odgovor želi da dobije.

Pitanje: Koliko se odgovori formulisani od strane pristupačnih jezičkih modela razlikuju od *state-of-art* modela?

Korišćeni modeli pokazali su se inferiorno u odnosu na ChatGPT 4o po svakom parametru. ChatGPT je davao neuporedivo kvalitetnije odgovore i pružao ih je istog trenutka, dok su lokalno izvršavani modeli odgovor generisali nekad i do 10 minuta, što je nedopustivo u realnim sistemima. Da bi se bolje utvrdio kompromis između kvaliteta odgovora i prednosti lokalnih, besplatnih jezičkih modela, potrebno je sprovesti istraživanje na jačim mašinama.

Pitanje: Koliko su dobijeni odgovori korisni za korisnike i kako ih poboljšati?

Dobijeni odgovori nisu bili dovoljno korisni za korisnike ni po kvalitetu odgovora ni po vremenu za njihovo generisanje. Idealno bi bilo koristiti jače jezičke modele jer je ChatGPT uspešno identifikovao ključne informacije i davao odgovore nerazlučive od ljudskih. Dizajn i inženjering upita su veoma važne tehnike jer omogućavaju pouzdanu pretragu dokumenata, što je preduslov za korisne odgovore. Izbor samih dokumenata

takođe znatno utiče na kvalitet odgovora. Izbor tehnike podele dokumenta na čankove može da poboljša efikasnost pretrage i važno je posvetiti vremena analizi svakog dokumenta pre nego što se ona izvrši.

5. ZAKLJUČAK

Veliki jezički modeli su doživeli brz razvoj u poslednjih nekoliko godina. Ovaj rad ima za cilj da pokaže kako ovi modeli mogu da unaprede tradicionalne upitničke sisteme korišćenjem RAG arhitekture. Istraživanje koje je uključilo različite modele kao što su Llama, Mistral, Claude i ChatGPT, primenjeno na različite vrste tekstualnih dokumenata, pokazalo je da se nedostaci jezičkih modela mogu smanjiti primenom RAG-a. Ovaj sistem poboljšava generisanje odgovora i efikasnije pronalazi korisne informacije, a odgovore čini sličnim ljudskim. Modeli, upitnički sistemi i RAG su najpre prikazani teorijski, a potom je napravljena aplikacija koja dokumenta obrađuje, nalazi informacije i pomoću velikog jezičkog modela odgovara na pitanja. Zatim je sprovedeno istraživanje na četiri dokumenta pisana različitim funkcionalnim stilovima kako bi se sistem evaluirao u specifičnim jezičkim situacijama jer je postojala opravdana pretpostavka da će kvalitet generisanih odgovora zavisiti od vrste teksta. Ovakav pristup je pokazao potencijale i sposobnosti velikih jezičkih modela da razumeju kontekst ali je ujedno pokazao i njihove nedostatke i slabosti, što je od ogromnog značaja za dalje unapređenje aplikacije i to pre svega u pronalaženju informacija iz tekstova čija svrha nije prenošenje znanja, informacija i činjenica. Istraživanje i teorijsko razmatranje ukazalo je na to da RAG sistemi mogu da budu izuzetno korisni u nekim privrednim zadacima kada je potrebno razviti sisteme za podršku korisnicima ili u naučno-istraživačke svrhe (pronalaženje relevantne studije i slično).

Ograničenja hardverskih resursa u ovom trenutku svakako predstavljaju problem, ali je očekivano da će u bliskoj budućnosti ti resursi biti dostupniji široj javnosti. Osim toga, pretraživanje informacija može biti unapređeno dodatnim tehnikama kao što su filtriranje, rerangiranje i aktivna pretraga. Postoji veliki broj tehnika koje mogu da se implementiraju u cilju poboljšanja RAG sistema, od embeddinga do generisanja odgovora. U praksi se koriste složeni RAG sistemi koji prevazilaze neke od uočenih problema. Na primer, GraphRAG koristi grafove znanja kako bi poboljšao procese pronalaženja informacija i generisanja odgovora. Grafovi znanja omogućavaju predstavljanje informacija kroz čvorove i veze između njih, što pomaže modelu da bolje razume složene odnose između pojmova u tekstu. Odgovaranje na pitanja pomoću RAG sistema je otvorena oblast istraživanja i postoji prostor za poboljšanja, čak i kod naprednih sistema.

RAG sistemi su se pojavili 2020. godine i za svega nekoliko godina nastale su biblioteke i softverski alati koji omogućavaju njihovo razvijanje sa relativno malo predznanja, uglavnom zahvaljući saradnji velike zajednice ljudi koja istražuje ovu oblast i deli znanje, čini ga javno dostupnim a to je upravo ono što omogućava eksponencijalni rast oblasti veštačke inteligencije i mašinskog učenja.

6. LITERATURA

- [1] W. X. Zhao *и остали*, „A Survey of Large Language Models“, Март 2023, [На Интернету]. Available at: <http://arxiv.org/abs/2303.18223>
- [2] P. Lewis *и остали*, „Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks“. [На Интернету]. Available at: <https://github.com/huggingface/transformers/blob/master/>
- [3] T. B. Brown *и остали*, „Language Models are Few-Shot Learners“, Мај 2020, [На Интернету]. Available at: <http://arxiv.org/abs/2005.14165>
- [4] Eordaxd, „Fine-Tuning vs. Pre-Training“, *Medium*, 2024.
- [5] Daniel Ramos, „Prompt Design vs. Prompt Engineering: Unleashing the Power of AI Language Models“, *Medium*, Јуни 2023.
- [6] Google Cloud Tech, USA. *Introduction to large language models*.
- [7] D. Jurafsky и J. H. Martin, „Speech and Language Processing An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models Third Edition draft Summary of Contents“.
- [8] Y. Goldberg и O. Levy, „word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method“, Феб. 2014, [На Интернету]. Available at: <http://arxiv.org/abs/1402.3722>
- [9] A. Vaswani *и остали*, „Attention Is All You Need“, Јуни 2017, [На Интернету]. Available at: <http://arxiv.org/abs/1706.03762>
- [10] A. Thiruvengadam, „Transformer Architecture: Attention Is All You Need.“, *Medium*, 2023.
- [11] M. Bojić, „Proširenje velikih jezičkih modela sistemom za pronalaženje informacija“.
- [12] A. Allam, M. H. Haggag, A. Mohamed, N. Allam, и M. H. Haggag, „The Question Answering Systems: A Survey“, 2012. [На Интернету]. Available at: www.sciacademypublisher.com
- [13] H. Li, Y. Su, D. Cai, Y. Wang, и L. Liu, „A Survey on Retrieval-Augmented Text Generation“, Феб. 2022, [На Интернету]. Available at: <http://arxiv.org/abs/2202.01110>
- [14] Daniel Bourke, USA. *Local Retrieval Augmented Generation (RAG) from Scratch (step by step tutorial)*, (2023.).
- [15] LangChain, *RAG From Scratch*, (2024.).
- [16] Ravi Theja, „Evaluating the Ideal Chunk Size for a RAG System using LlamaIndex“.
- [17] OpenAI, „API Pricing“.
- [18] Vikipedija, „Funkcionalni stilovi srpskog književnog jezika“.
- [19] B. Andriy Burkov, „The Hundred-Page Machine Learning“.

- [20] F. Dostoevsky, „Crime and Punishment“.
- [21] Brad Plumer, „Climate Change Is Speeding Toward Catastrophe. The Next Decade Is Crucial, U.N. Panel Says.“, *New York Times*, 2023.
- [22] OpenAI, „Privacy policy“, 2024.
- [23] F. Pepe, V. Nardone, A. Mastropaolo, G. Canfora, G. Bavota, и M. Di Penta, „How do Hugging Face Models Document Datasets, Bias, and Licenses? An Empirical Study“, в *IEEE International Conference on Program Comprehension*, IEEE Computer Society, 2024, стр. 370–381. doi: 10.1145/3643916.3644412.
- [24] O. Kamal, „Discover LM Studio“, *Medium*, 2023.
- [25] Callum Macpherson, „Implementing RAG in LangChain with Chroma: A Step-by-Step Guide“, *Medium*.
- [26] HuggingFace, „Sentence Transformers, "all-mpnet-base-v2“.
- [27] HuggingFace, „Meta Llama 3.1 8B“, 2024.
- [28] HuggingFace, „Mistral Nemo Instruct 2407“.
- [29] TheBloke, „ claude2-alpaca-13B-GGUF“, HuggingFace.