Implementacija konverzacijskog agenta za personalizaciju nastavnih materijala pomoću jezičnih modela

Blašković, Luka

SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET INFORMATIKE

Luka Blašković

IMPLEMENTACIJA KONVERZACIJSKOG AGENTA ZA PERSONALIZACIJU NASTAVNIH MATERIJALA POMOĆU JEZIČNIH MODELA

Diplomski rad

Pula, srpanj, 2024. godine

SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET INFORMATIKE

Luka Blašković

IMPLEMENTACIJA KONVERZACIJSKOG AGENTA ZA PERSONALIZACIJU NASTAVNIH MATERIJALA POMOĆU JEZIČNIH MODELA

Diplomski rad

JMBAG: 0303088177, redovni student
Studijski smjer: Informatika

Kolegij: Razvoj IT rješenja
Znanstveno područje : Društvene znanosti
Znanstveno polje : Informacijske i komunikacijske znanosti
Znanstvena grana : Informacijski sustavi i informatologija

Mentor: doc. dr. sc. Nikola Tanković

Pula, srpanj, 2024. godine

Sažetak

Generativna umjetna inteligencija, temeljena na velikim jezičnim modelima (LLM), postaje sveprisutna u obrazovanju, omogućujući personalizaciju kroz generiranje prilagođenih odgovora i napredno rasuđivanje. No, ograničeno znanje LLM-ova dovodi do halucinacija, gdje modeli generiraju dezinformacije. RAG (Retrieval Augmented Generation) tehnike omogućuju pristup vanjskim izvorima podataka, povećavajući točnost odgovora. U ovom radu prezentiran je EduBot, konverzacijski agent koji koristi RAG module za pomoć studentima i nastavnicima. Evaluacija EduBota provedena je ručnom analizom 20 pitanja vezanih uz nastavne materijale i sveučilišne pravilnike, pokazujući značajna poboljšanja u točnosti odgovora. Komercijalna rješenja poput GPT-4 i Claude 3 Sonnet postigla su izvanredne performanse, dok su manji modeli poput Mistral, Gemma i Llama3 također pokazali značajan potencijal. Rad također obuhvaća pregled razvoja NLP-a i velikih jezičnih modela baziranih na transformer arhitekturi. Evaluacija je provedena ručnom analizom 20 akademskih pitanja te evaluacijom pomoću mjera vjernosti (FAITH) i relevantnosti odgovora (ANS_REL), pokazujući zadovoljavajuće rezultate kod komercijalnih rješenja.

Ključne riječi : konverzacijski agent, veliki jezični modeli (LLM), transformer arhitektura, duboko učenje, obrada prirodnog jezika (NLP), metode oblikovanja konteksta (RAG)

Abstract

Generative artificial intelligence, based on large language models (LLM), is becoming ubiquitous in education, enabling personalization through generating tailored responses and advanced reasoning. However, the limited knowledge of LLMs leads to hallucinations, where models generate misinformation. Retrieval Augmented Generation (RAG) techniques enable access to external data sources, increasing the accuracy of responses. This paper presents EduBot, a conversational agent that uses RAG modules to assist students and teachers. The evaluation of EduBot was conducted through manual analysis of 20 questions related to teaching materials and university regulations, showing significant improvements in response accuracy. Commercial solutions such as GPT-4 and Claude 3 Sonnet achieved outstanding performance, while smaller models like Mistral, Gemma, and Llama3 also demonstrated significant potential. The paper also includes an overview of the development of NLP and large language models based on transformer architecture. The evaluation was carried out through manual analysis of 20 academic questions and evaluation using measures of fidelity (FAITH) and response relevance (ANS_REL), showing satisfactory results for commercial solutions.

Keywords : chatbot, large language models (LLM), transformer architecture, deep learning, natural language processing (NLP), retrieval augmented generation (RAG)

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Luka Blašković, kandidat za magistra informatike ovime izjavljujem da je ovaj Diplomski rad isključivo rezultat mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Diplomskog rada nije napisan na nedozvoljeni način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

Luka Blašković

Pula, srpanj, 2024. godine

IZJAVA O KORIŠTENJU AUTORSKOG DJELA

Ja, Luka Blašković, dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj Diplomski rad pod nazivom „Implementacija konverzacijskog agenta za personalizaciju nastavnih materijala pomoću jezičnih modela“ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

Student

Luka Blašković

Pula, srpanj, 2024. godine

ZAHVALA

Želim izraziti iskrenu zahvalnost svom mentoru, doc. dr. sc. Nikoli Tankoviću, na stručnoj podršci tijekom izrade ovog rada. Njegova susretljivost, neiscrpna motivacija i svi vrijedni materijali koje mi je slao mjesecima bili su od neprocjenjive važnosti. Posebno se zahvaljujem svojoj obitelji na pomoći i ohrabrenju koje su mi pružali tijekom studiranja. Bez njihove podrške, ova avantura ne bi bila moguća.

Sadržaj

1 Uvod **1**

2 Korisnički zahtjevi **3**

2.1 Platforme za online učenje 3

2.2 Izazovi i prilike u personalizaciji podrške 5

2.3 Kako bi izgledao idealan chatbot? 7

2.4 EduBot - UML model obrazaca korištenja 8

3 Veliki jezični modeli **10**

3.1 Definicija modela 10

3.2 Neuronske mreže i duboko učenje 12

3.3 Obrada prirodnog jezika (NLP) 18

3.3.1 Povijest strojnog prijevoda jezika 18

3.3.2 RNN i Sekvencijalni modeli 22

3.3.3 Transformer arhitektura 25

3.3.4 Kako rade transformer modeli? 26

3.4 Foundation modeli 31

3.4.1 Otvoreni modeli 31

3.4.2 *text-to-image* modeli 32

3.4.3 Razvoj AI zajednice 33

3.4.4 Fine-tuning 34

3.4.5 RLHF 37

4 Metode oblikovanja konteksta **39**

4.1 Uvod u RAG 39

4.2 Osnovni RAG 40

4.2.1 Nedostaci osnovne RAG tehnike 48

4.3 Kako se ostvaruju kvalitetne semantičke značajke? 49

4.3.1 Metode dohvaćanja (*eng. Retrieval methods)* 49

4.3.2 Suvremeni pristup modelima tekstualnog ugrađivanja 51

4.4 Napredni RAG 53

4.4.1 Proces prije dohvaćanja (*eng. Pre-retrieval process*) 54

4.4.2 Proces nakon dohvaćanja (*eng. Post-retrieval process*) 64

4.5 Modularni RAG 67

4.5.1 RAG vs Finetuning 71

4.5.2 RAG evaluacija 72

5 Implementacija **73**
5.1 Streamlit 73
5.2 Elementi personalizacije 74
5.3 RAG moduli 76
5.3.1 RAPTOR modul 76
5.3.2 *text-to-SQL* modul 78
5.3.3 *web scraper* modul 80
5.4 Semantičko usmjeravanje upita 80

6 Evaluacija **83**
6.1 Ručna evaluacija 83
6.2 LLM evaluacija 88
6.2.1 Metrike 89

7 Zaključak **93**

Literatura

Popis slika

Popis tablica

1 Uvod

Razvoj umjetne inteligencije donosi značajne promjene u poslovanje organizacija različitih djelatnosti, uključujući medicinu, obrazovanje, financije, proizvodnju, zabavnu industriju i mnoge druge. Sve veća primjena AI tehnologija značajno doprinosi rastu globalnog tržišta umjetne inteligencije. Prema podacima platforme *Statista*, predviđa se da će tržište umjetne inteligencije dosegnuti vrijednost od oko 184 milijarde američkih dolara u 2024. godini, uz očekivani godišnji rast od gotovo 30% [1]. Danas, uvođenje tzv. konverzacijskih agenata u poslovanje, poznatijih kao *chatbotovi*, predstavlja revolucionarno rješenje za poboljšanje i optimizaciju poslovnih procesa, što istovremeno unapređuje korisničko iskustvo. Iako *chatbotovi* nisu novost, njihova integracija s velikim jezičnim modelima omogućuje implementaciju naprednijih i sofisticiranijih softverskih rješenja.

Veliki jezični modeli (*eng. Large language models (LLMs)*) predstavljaju podskup generativne umjetne inteligencije, specijalizirane za generiranje sadržaja. Trenirani na ogromnim količinama tekstualnih podataka, ovi modeli mogu razumjeti, prepoznati relevantne obrasce i izuzetno kvalitetno generirati tekst [2]. Danas se smatraju izuzetno moćnim alatima, često korištenim kao osobni asistenti za svakodnevne zadatke ili za generiranje raznovrsnog sadržaja. Među njima se ističe ChatGPT, konverzacijski agent koji je razvila tvrtka OpenAI, koristeći njihove napredne jezične modele GPT-3 i GPT-4. Unatoč tome što je ChatGPT izvanredan alat, on i slični konverzacijski agenti imaju određena ograničenja: sadrže ograničeno znanje do datuma treniranja, imaju poteškoća s razumijevanjem složenijih konteksta, podložni su sigurnosnim ograničenjima i imaju ograničene mogućnosti personalizacije. Najnoviji modeli, kao što su GPT-4 i GPT-4o, omogućuju pretraživanje interneta i učitavanje datoteka u svrhu proširenja konteksta. Međutim, sigurnosni izazovi i dalje postoje. Primjerice, ako tvrtka za koju radimo ne dozvoljava dijeljenje datoteka ili programskog kôda izvan organizacije, što je često upravo i slučaj.

Glavni problem nastaje kada je za dobivanje odgovora na pitanje potreban kontekst iz različitih izvora podataka ili iz velikog skupa podataka koji se sastoji od desetaka ili stotina datoteka. Iako je moguće kopirati sadržaj svih tih datoteka i koristiti ga kao kontekst, istraživanja i praksa pokazuju da zbog ogromne količine podataka, LLM-ovi često isporučuju lošije i nepredvidljive rezultate. To često dovodi do tzv. halucinacija. Halucinacije u ovom kontekstu odnose se na situacije gdje model generira odgovor koji je sintaktički ispravan, ali nema stvarnog oslonca u podacima ili kontekstu. Takvi odgovori mogu sadržavati izmišljene činjenice (dezinformacije) ili nelogične zaključke.

Iz navedenog se jasno uočava potreba za optimizacijskim tehnikama, od kojih je jedna od najučinkovitijih skup metoda poznatih kao **Retrieval-Augmented Generation** (RAG). RAG je skupina metoda koja omogućava modelima dohvaćanje relevantnih informacija iz unaprijed definiranih izvora znanja, čime se stvara bolji kontekst i poboljšavaju odgovori. Ovo područje trenutno doživljava izuzetno brz razvoj, s novim idejama i rješenjima koja se pojavljuju gotovo svakodnevno. RAG metode temelje se na integraciji faze dohvaćanja informacija i generiranja odgovora, omogućujući modelima da aktivno pretražuju relevantne informacije iz vanjskih izvora i obogaćuju kontekst za generiranje odgovora. Ovaj pristup omogućava bolju prilagodbu specifičnim upitima korisnika i povećava relevantnost odgovora. Unatoč brzom razvoju i brojnim inovacijama, RAG rješenja dijele iste ciljeve: poboljšanje točnosti odgovora kroz pristup relevantnim podacima, smanjenje količine podataka koje LLM mora istovremeno obraditi, smanjenje rizika od halucinacija te ušteda računalnih resursa, što uključuje i vrijeme i financijska sredstva. U konačnici, RAG metode predstavljaju značajan napredak u području konverzacijskih agenata, omogućujući im da budu precizniji, učinkovitiji i pouzdaniji u pružanju informacija korisnicima.

Za potrebe ovog diplomskog rada razvijen je konverzacijski agent (*eng. chatbot*) za personalizaciju nastavnih materijala i dobivanje odgovora na druga akademska pitanja, posebno namijenjen studentima, ali i nastavnom osoblju - EduBot. **EduBot** temelji se na najnovijim LLM-ovima dostupnim na tržištu putem plaćenih API licenci, kao što su OpenAI GPT-4 ili Anthropic Claude-3. Osim toga, nudi mogućnost korištenja manjih otvorenih modela, poput LLaMA 3 i Gemma, koji se mogu jednostavno pokrenuti na osobnom računalu.

EduBot prepoznaje korisničke namjere i na temelju njih prilagođava svoje odgovore. Kada je u pitanju student, EduBot uzima u obzir njegove podatke, preferencije i postojeće znanje iz programiranja kako bi odabrao odgovarajući alat za generiranje odgovora. Ovisno o prepoznatoj namjeri, EduBot provodi upit kroz odgovarajući RAG modul. Svaki alat predstavlja specifičnu metodu oblikovanja konteksta koja je asistentu potrebna za pružanje točnog i korisnog odgovora na postavljeno pitanje. Metode su integrirane u aplikacijske module:

1. Analiza više dokumenata različitih veličina i formata podataka te izrada stabla znanja (*eng. knowledge tree*)

2. Generiranje SQL upita na temelju korisničkog pitanja (*eng. text-to-SQL*) te izvršavanje upita nad bazom podataka

3. Dohvaćanje najnovijih informacija sa sveučilišne web stranice koristeći tehniku *web-scraping* za ekstrakciju sadržaja.

U završnom poglavlju prikazani su rezultati evaluacije odgovora na 20 pitanja koristeći tri različite RAG tehnike. Evaluacija obuhvaća usporedbu rezultata različitih velikih jezičnih modela u kontekstu konzistentnosti odgovora s izvorom znanja, kao i s postavljenim pitanjem. Također, evaluacija se provodi i na manjim, otvorenim izvorima, kako bi se pokazala učinkovitost RAG metodologija i na slabijim modelima.

2 Korisnički zahtjevi

Danas je prilagodljivost jedan od ključnih elemenata za uspješan pristup i izgradnju pozitivnih odnosa s klijentima. Bilo da se radi o proizvodima, uslugama ili sadržaju, prilagođavanje ponude individualnim potrebama i preferencijama korisnika polako postaje neophodno za postizanje konkurentske prednosti. Jednako kao u poslovnom svijetu, personalizirano učenje u obrazovanju još i od ranijih dana igra ključnu ulogu u postizanju uspješnih rezultata. Svaki učenik, bilo osnovne ili srednje školi kao i student, ima jedinstvene potrebe, stilove učenja, interese i razine znanja. Tradicionalni obrazovni sustavi često ne uspijevaju adekvatno adresirati ove razlike, što može rezultirati slabijim uspjehom i psihičkim opterećenjem učenika pa i nastavnika.

Individualizirani pristup omogućava prilagodbu nastavnog plana i programa svakom učeniku, što potiče dublje razumijevanje i povećava motivaciju za učenje [3]. Finski obrazovni sustav često se ističe kao jedan od najboljih na svijetu, a mnogi analitičari ga smatraju i najboljim [4] [5]. Prema izvješću "*The Secret to Finnish Education: Trust*", finski obrazovni sustav postiže izvanredne rezultate zahvaljujući visoko kvalificiranim učiteljima koji uživaju veliku autonomiju u svom radu [6]. Naglasak je na dobrobiti učenika kroz pružanje besplatnih obroka, zdravstvene skrbi, opuštenije atmosfere te praksu "Learning over Testing" koja se odmiče od standardiziranih testova. Izvješće ističe važnost personaliziranog pristupa učenju. Učitelji primjenjuju različite metode procjene kako bi pratili napredak učenika i prilagodili strategije podučavanja, dok učenici imaju priliku postavljati vlastite ciljeve, pratiti svoj napredak i aktivno sudjelovati u procesu učenja. Najvažnije, obrazovni sustav ih potiče da preuzmu odgovornost za svoje učenje i razvijaju kritičko mišljenje [7]. Sličan pristup usvojen je i u Estoniji, koja također postiže visoke rezultate na međunarodnim obrazovnim ljestvicama, poput PISA i TIMSS [8].

Korištenjem modernih tehnologija i metoda, nastavnici mogu lakše prepoznati i odgovoriti na specifične potrebe svakog učenika, ali i oni mogu svoje znanje usavršavati samostalno.

2.1 Platforme za online učenje

S razvojem digitalnih tehnologija u proteklom desetljeću, na tržište su stupile brojne digitalne edukacijske platforme koje se temelje na personaliziranom pristupu korisnicima kao glavnom poslovnom motivu. Primjeri najuspješnijih edukacijskih platformi uključuju:

1. **Khan Academy** - neprofitna organizacija koja nudi besplatne obrazovne materijale namijenjene svim dobnim skupinama. Lekcije su u obliku videozapisa, članaka i interaktivnih vježbi, pokrivajući teme kao što su matematika, prirodne znanosti, povijest, računalno programiranje, ekonomija itd. Platforma se prilagođava napretku svakog učenika, omogućujući im da uče vlastitim tempom. Učitelji i mentori mogu pratiti napredak učenika putem detaljnih izvješća, što im omogućava pružanje individualizirane podrške i usmjeravanja.

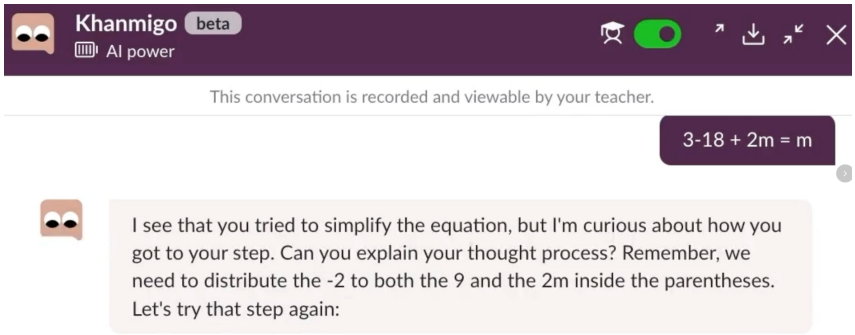
2. **Duolingo** - popularna aplikacija za učenje stranih jezika koja nudi lekcije iz gotovo 40 jezika. Ova platforma koristi pristup gamifikacije (*eng. gamification*) kako bi učenje učinila zabavnim i motivirajućim. Gamifikacija podrazumijeva korištenje elemenata igre u "ne-igrama",

primjerice u obrazovanju, kako bi se povećala angažiranost i motivacija korisnika. Dodatno, Duolingo koristi personalizirano učenje prilagođavajući se korisnikovoj razini znanja i napretku, omogućavajući učinkovitije učenje. Kroz elemente *gamifikacije* poput bodova, razina, znački i dnevnih ciljeva, korisnici su motivirani da redovito vježbaju i napreduju.

3. **Brilliant** - obrazovna platforma usmjerena na učenje prirodnih znanosti iz *STEM* područja, kao što su matematika, fizika i programiranje, putem interaktivnih lekcija i izazova. Umjesto pasivnog gledanja video materijala, učenici aktivno sudjeluju u rješavanju problema. Platforma nudi prilagodljive staze učenja, preporuke, dnevne kvizove i praćenje napretka, što omogućuje personalizirani pristup učenju. Interaktivno rješavanje problema čini ovu platformu posebnom i izuzetno zanimljivom. Na temelju rezultata rješavanja problema, sustav može preusmjeriti učenike na prethodne lekcije, ako procijeni da je to potrebno.

4. **Scrimba** - inovativna platforma za učenje programiranja koja kombinira video lekcije s interaktivnim pisanjem programskog kôda. Pokrenuta 2017. godine, Scrimba omogućava korisnicima da zaustave video, izmijene kôd i odmah vide rezultate svojih promjena. Osim toga, platforma nudi personalizirane elemente učenja, uključujući prilagođene kvizove i zadatke, praćenje napretka te mentorski program, što sve zajedno čini učenje programiranja učinkovitijim i pristupačnijim.

Posljednjih godina, ove i mnoge druge platforme integriraju napredne jezične modele kako bi unaprijedile personalizirano učenje za svoje korisnike. Primjerice, Khan Academy je predstavio *Khanmigo*, asistenta pogonjenog *GPT-4* jezičnim modelom. Njegova uloga je biti virtualni tutor za studente i asistent nastavnicima, pomažući studentima kroz dublje diskusije i poticanje istraživanja problema u dubinu. Umjesto generičkih odgovora, Khanmigo potiče kritičko razmišljanje i bolje razumijevanje složenih koncepata. Duolingo koristi svoj interni AI model pod nazivom *Birdbrain*, koji pomaže u kreiranju lekcija za učenje stranih jezika, značajno smanjujući vrijeme potrebno za prilagodbu lekcija specifičnim korisničkim grupama. Birdbrain kontinuirano analizira znanje učenika, prepoznaje koje dijelove jezika savladavaju bolje ili lošije te procjenjuje koliko dobro učenici poznaju pojedine riječi i jezične konstrukte. Također, Duolingo nudi *Duolingo Max* pretplatničku uslugu koja donosi najnovije mogućnosti, poput simulacija stvarnih razgovora, personaliziranih objašnjenja za odgovore korisnika i igranja uloga. Slika 1 prikazuje korisničko sučelje aplikacije Khamingo.



Slika 1: Sučelje Khanmigo asistenta

2.2 Izazovi i prilike u personalizaciji podrške

Glavni motiv za razvoj konverzacijskog agenta leži u potrebi za primjenom dobrih praksi, kao što je opisano u prethodnom potpoglavlju. Uvođenje personaliziranih pristupa u obrazovanje, podržanih modernim tehnologijama i metodama, može značajno unaprijediti proces učenja, posebno u zemljama gdje se još uvijek prakticiraju tradicionalni obrazovni sustavi. Platforme za online učenje omogućuju svim učenicima da uče vlastitim tempom i prema vlastitim potrebama, razvijaju kritičko razmišljanje i stječu znanja koja su im do sada možda nedostajala.

Visoko obrazovanje često predstavlja izazov za veliki broj novih studenata. Suočavaju se s poteškoćama koje mogu uključivati prilagodbu na novi način učenja, veću količinu gradiva, teže ispite, kao i potrebu za boljim upravljanjem vremenom. Mnogi studenti dolaze direktno iz srednjoškolskog sustava, koji je prilično različit u svojoj organizaciji i pristupu učenju od onog akademskog. Na visokim učilištima se od studenata očekuje veća razina samostalnosti, odgovornosti i proaktivnosti u pristupu učenju. Također, tu je specifična akademska kultura koja uključuje pravila citiranja, izradu seminarskih radova, prisustvovanje diskusijama i sudjelovanje u istraživačkim projektima.

Jedan od ključnih izazova s kojim se suočavaju studenti u visokom obrazovanju jest upravljanje vlastitim vremenom. Nakon svakodnevne rutine stečene tijekom 12 godina osnovne i srednje škole, mnogi se na fakultetima bore s pronalaženjem balansa između predavanja, samostalnog učenja i društvenog života. Uz akademske izazove, tu su i socijalni aspekti prilagodbe na novi životni stil, novu socijalnu okolinu, sklapanje novih prijateljstava i financijsko upravljanje. Mnoge institucije visokog obrazovanja prepoznaju ove izazove i često nude različite oblike podrške kako bi olakšale tranziciju. To uključuje radionice o upravljanju vremenom i stresom, pristup savjetodavnim službama te psihološku pomoć.

Ono što nedostaje i na čemu treba aktivno raditi jest sustavna i personalizirana podrška koja bi studentima pomogla u rješavanju specifičnih problema s kojima se suočavaju. Primjerice, studenti često imaju pitanja vezana uz:

- raspored ispitnih rokova i procedure za prijavu ili odjavu ispita,
- nastavni sadržaj i materijale,
- načine i pravila polaganja kolegija,
- mogućnosti za obavljanje stručne prakse,
- administrativne procedure i razno-razne zahtjeve,
- konzultacije s nastavnikom i sl.

Digitalni konverzacijski agenti mogu značajno unaprijediti svakodnevicu studenata i nastavnika, posebno u rješavanju akademskih izazova. Dok klasični *chatbotovi* temeljeni na unaprijed definiranim pravilima (*eng. rule-based chatbots*) nude solidna rješenja, poput usmjeravanja

na relevantne resurse na stranicama Sveučilišta, njihova učinkovitost ima svoje granice. Ovi *chatbotovi* često ne mogu razumjeti složenije upite niti pružiti personalizirane odgovore zbog specifičnih situacija (*eng. edge cases*) koje nisu pokrivene pravilnicima dostupnim na Sveučilištu. Zbog toga, mnogi studenti traže dodatnu podršku kako bi riješili specifične izazove s kojima se suočavaju.

Veliki jezični modeli mogu značajno doprinijeti rješavanju ovih problema prepoznavanjem točne namjere ili problema studenta putem prirodnog jezika, bez potrebe za unaprijed definiranim opcijama koje korisnik mora odabrati. Iako su ovi modeli trenirani na velikim i raznovrsnim skupovima podataka, često neće moći pružiti precizan odgovor na specifične upite korisnika, već će ponuditi generički odgovor. Uzmimo sljedeći primjer gdje korisnik (npr. student) postavlja pitanje:

> *"Koja je procedura upisa na online studij Fakulteta informatike?"*

Generički LLM odgovor mogao bi biti sljedeći:

> "Procedura upisa na online studij može se razlikovati ovisno o instituciji. Molimo vas da provjerite službenu stranicu Fakulteta ili kontaktirate administrativnu osobnu za točne informacije."

Iako je ovaj odgovor tehnički ispravan, često nije dovoljno specifičan niti koristan studentu koji traži konkretne informacije. Puno veća opasnost koju predstavljaju veliki jezični modeli jest njihova sklonost **haluciniranju**. Haluciniranje se odnosi na situaciju kada LLM generira tekst, izjavu ili rečenicu koja je netočna, izmišljena ili nepovezana sa stvarnim svijetom [9]. Jednostavan primjer halucinacije je kada model lažno tvrdi da je određena povijesna ličnost učinila nešto što se zapravo nikada nije dogodilo. Zbog svojih naprednih jezičnih sposobnosti (npr. mogućnost generiranja gramatički točnog i stilski prikladnog teksta te upotrebe detalja), **veliki jezični modeli su vrlo dobri u održavanju koherencije u kontekstu**, pa njihovi odgovori često zvuče logično i povezano, čak i kad su netočni. Halucinacije su jedan od glavnih izazova u primjeni jezičnih modela u osjetljivim područjima poput medicine, prava ili financija, gdje točnost informacija mora biti izuzetno visoka.

Primjer halucinacije u odgovoru na isto pitanje mogao bi izgledati ovako:

> "Procedura upisa na online studij Fakulteta informatike uključuje ispunjavanje prijavnog obrasca na web stranici Fakulteta sa svim traženim podacima. Nakon toga morate priložiti potrebne dokumente, uključujući sken osobne iskaznice, potvrdu o završenoj srednjoj školi, životopis i motivacijsko pismo. Nakon što vaša prijava bude pregledana, morate prisustvovati obaveznom webinaru o pravilima online studija."

Obogaćivanjem našeg *chatbota* točnim informacijama o raznovrsnim akademskim pitanjima i aktualnostima, stvaramo suvremeni alat koji može značajno unaprijediti iskustvo studenata i nastavnika tako da pruža precizne i personalizirane odgovore na specifične upite.

2.3 Kako bi izgledao idealan chatbot?

Idealni *chatbot* kombinirao bi napredne tehnološke mogućnosti prirodne obrade jezika (NLP) koristeći velike jezične modele (LLM-ove) kao temelj te primjenu RAG tehnika za pretraživanje različitih izvora znanja kako bi odgovori bili što točniji i personaliziraniji. Korištenjem sofisticiranih jezičnih modela, *chatbot* može precizno tumačiti korisničke upite i formulirati relevantne odgovore. Primjena različitih RAG tehnika omogućila bi mu pretraživanje te integraciju informacija iz različitih izvora, rezultirajući bogatijim i informiranijim odgovorima. Funkcionalnosti "idealnog chatbota" za personalizirani pristup studentima mogu se razvrstati u nekoliko ključnih aspekata:

1. **Odgovaranje na specifična pitanja**

   - Precizno odgovaranje na pitanja o ispitnim rokovima, nastavnom gradivu, pravilima polaganja kolegija, stipendijama, upisima, akademskom rasporedu i sl.

   - Korištenjem metoda pretraživanja konteksta, *chatbot* može brzo pretražiti baze podataka i druge relevantne izvore informacija te pružiti točne odgovore.

2. **Personalizirani savjeti za učenje**

   - Preporuke za dodatnu literaturu ovisno o predznanju studenta.

   - Prilagođavanje odgovora ovisno o znanju pojedinog dijela gradiva.

   - Preporuka relevantnih tečajeva i zadataka koji zahtijevaju veću pažnju.

   - Analiza napretka studenta i izrada personaliziranih planova učenja.

3. **Interaktivno vođenje kroz procese**

   - Sposobnost vođenja korisnika kroz administrativne procese poput: prijave na stručnu praksu, dobivanje određenih potvrda, prijava završnog ili diplomskog rada, predaja studentskih projekata, zadaće ili prijave/odjave ispitnih rokova.

   - Kroz interaktivne dijaloge, *chatbot* može korak po korak voditi korisnika kroz složene procese, osiguravajući da su svi koraci pravilno izvršeni.

4. **Korisnički profil i elementi *gamifikacije***

   - Praćenje napretka studenta kroz personalizirane korisničke profile.

   - Prikaz individualnih statistika, bodova, preporuka, ispunjenih zadataka i postignutih ciljeva za svakog studenta.

   - Davanje informacija o rokovima za predaju zadataka, terminima nadolazećih kolokvija i sl.

   - Implementacija elemenata *gamifikacije* tako da se studente motivira da aktivno sudjeluju u procesu učenja.

5. **Sigurnosni aspekti**

   - Implementacija sigurnosnih protokola kako bi se osigurala privatnost korisnika i zaštita njihovih osobnih podataka.

6. **Multimodalna podrška**

   - Idealni *chatbot* trebao bi imati sposobnost primanja i obrade različitih formata podataka, a ne samo tekstualnih. To uključuje slike, PDF dokumente, prezentacije, datoteke programskog kôda i druge formate.

   - Osim obrade različitih formata ulaznih podataka, moderan *chatbot* trebao bi biti sposoban prikazati multimedijalni sadržaj kao dio svojih odgovora. To uključuje grafove za bolju vizualizaciju informacija, slike koje objašnjavaju koncepte, interaktivne dijagrame i sl.

7. **Integracija s postojećim IS**

   - Integracija sa sustavima e-obrazovanja, kao što su: *Moodle*, *Microsoft Teams*, *Google Classroom* i sl.

   - Sinkronizacija s kalendarom.

   - Sinkronizacija s ostalim relevantnim akademskim IS, npr. sa *ISVU* sustavom (*Informacijski sustav visokih učilišta*).

2.4 EduBot - UML model obrazaca korištenja

EduBot aplikacija razvijena je za potrebe ovog diplomskog rada, s ciljem zadovoljavanja potreba i smjernica opisanih u kontekstu personaliziranog pristupa obrazovanju. Aplikacija je osmišljena kako bi olakšala studentima i profesorima svakodnevne akademske zadatke, s posebnim naglaskom na podršku studentima. EduBot koristi naprednije RAG tehnike koje omogućuju precizne i personalizirane odgovore na specifične upite korisnika. Međutim, razvoj takvog alata zahtijeva još mnogo rada kako bi se postigla optimalna učinkovitost i pouzdanost.

U nastavku je prikazan UML model obrazaca korištenja kroz dijagram te su ukratko objašnjene osnovne funkcionalnosti EduBot aplikacije. Detaljnija implementacija je razrađena u 5. poglavlju.

Glavne radnje koje korisnik može obavljati u EduBot aplikaciji:

1. **Postavljanje pitanja** povezanih s raznim akademskim temama, uključujući: pitanja iz nastavnog gradiva, stanje bodova na kolegiju, nastavni plan i program, novosti sa Sveučilišta, zadaće, projekte i drugo.

2. **Učitavanje vlastitih materijala za učenje**, što može obuhvaćati više datoteka različitih formata, poput PDF-a.

3. **Konfiguriranje postavki chatbota**, uključujući odabir LLM-a i konfiguracija postavki modula za oblikovanje konteksta.

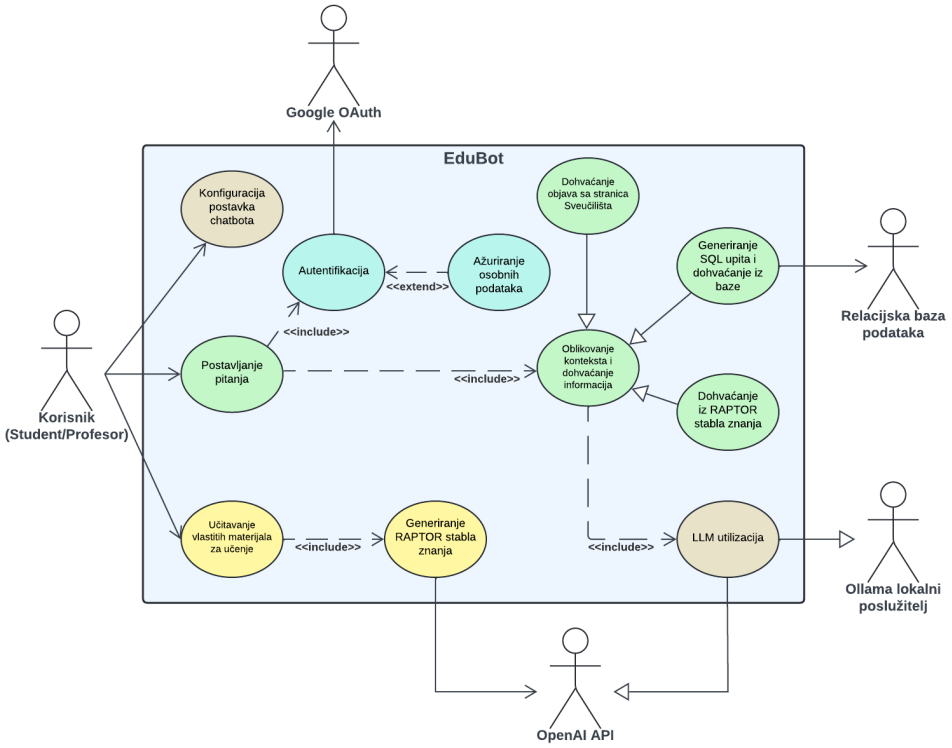
Postavljanje pitanja zahtijeva autentifikaciju kako bi se omogućilo personalizirano učenje. Proces autentifikacije provodi se putem vanjskog sustava *Google OAuth*, a nakon uspješne autentifikacije korisnik može dodatno ažurirati svoje osobne podatke. Ovi podaci uključuju osobni opis (o meni), razinu znanja iz programiranja na skali od 1 do 10 te godinu studija (1. prijediplomski do 2. diplomski). Ako korisnik odluči učitati vlastite materijale, izrađuje se RAPTOR stablo znanja, omogućavajući EduBotu da "nauči" iz svih priloženih datoteka. RAPTOR metoda za oblikovanje konteksta razrađena je detaljno u 3. poglavlju. Oblikovanje konteksta i dohvaćanje informacija uvijek prethodi generiranju konačnog odgovora. Aplikacija koristi tri modula, ovisno o prepoznatoj namjeri korisnika:

1. **Dohvaćanje informacija iz RAPTOR-ovog stabla znanja**
2. **Generiranje SQL upita, izvršenje tog upita i dohvaćanje iz baze podataka**
3. **Dohvaćanje objava sa stranica Sveučilišta**

U ovim koracima, veliki jezični modeli sudjeluju kao vanjski akteri, a mogu biti:

1. **Lokalni** - putem Ollama lokalnog poslužitelja.
2. **Vanjski** - putem OpenAI API servisa.

Opisani proces može se detaljno proučiti kroz *UML Use Case* dijagram prikazan na Slici 2.



Slika 2: UML dijagram obrazaca upotrebe EduBot aplikacije

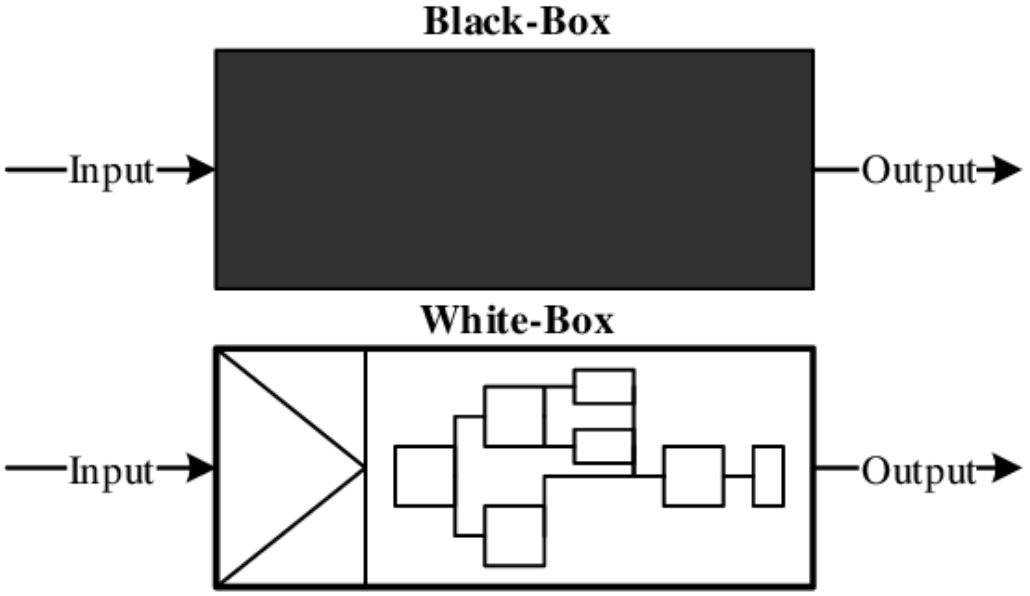
3 Veliki jezični modeli

Veliki jezični modeli *(eng. Large Language Models, LLMs*) spadaju u kategoriju tzv. *foundation* modela, koji su trenirani na ogromnim količinama podataka. Ogromne količine podatka te napredna arhitektura ovih modela, omogućava im da "razumiju" i generiraju prirodni jezik, kao i druge oblike sadržaja s visokim stupnjem točnosti i svestranosti [10].

U ovom poglavlju istražit ćemo ključne koncepte koji su neophodni za razumijevanje pojma velikih jezičnih modela. Obradit ćemo osnove dubokog učenja, povijest razvoja obrade prirodnog jezika, te ćemo se detaljno posvetiti *seq2seq* arhitekturama i modernim transformer modelima.

3.1 Definicija modela

Kako bismo bolje razumjeli što su zapravo veliki jezični modeli, ali i jezični modeli općenito, potrebno je definirati pojam "model" u kontekstu umjetne inteligencije i strojnog učenja. Općenito u računarstvu, model se odnosi na apstraktni opis sustava pomoću matematičkih principa [11]. Svaki model koji nije *white-box*, odnosno model s apsolutno transparentnim i razumljivim internim strukturama i pravilima, naziva se *black-box* modelom. U *black-box* modelima, interne strukture i pravila donošenja odluka su nedostupni ili su prekompleksni čovjeku za razumijevanje. Slika 3 ilustrira opisanu razliku između *white-box* i *black-box* modela.



Slika 3: Black-box VS White-box

***white-box* model:** Jedan od poznatijih primjera *white-box* modela je linearna regresija. U statistici, linearna regresija odnosi se na svaki pristup modeliranju relacija između jedne zavisne varijable, često označene s Y, te jedne ili više nezavisnih varijabli, označenih s X, na način da takav model linearno ovisi o nepoznatim parametrima procijenjenima iz podataka [12].

Zašto linearnu regresiju možemo interpretirati kao *white-box*?

1. **Jednostavnost strukture**

   Linearni regresijski model temelji se na jednostavnoj linearnoj funkciji za modeliranje relacije između zavisne varijable i nezavisnih varijabli. Prema tome, opći oblik linearnog regresijskog modela možemo prikazati na sljedeći način:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \epsilon \tag{1}$$

   gdje su: Y zavisna varijabla, $X_1, X_2, \ldots, X_n$ nezavisne varijable, $\beta_0, \beta_1, \ldots, \beta_n$ koeficijenti, a ϵ je pogreška modela (*eng. residual*).

2. **Jasna interpretacija**

   Svaki koeficijent β_i ima jasno tumačenje. Na primer, β_1 predstavlja promjenu u Y kada se X_1 promijeni za jednu jediničnu vrijednost, uz uvjet da su ostale varijable konstantne.

3. **Deterministički pristup**

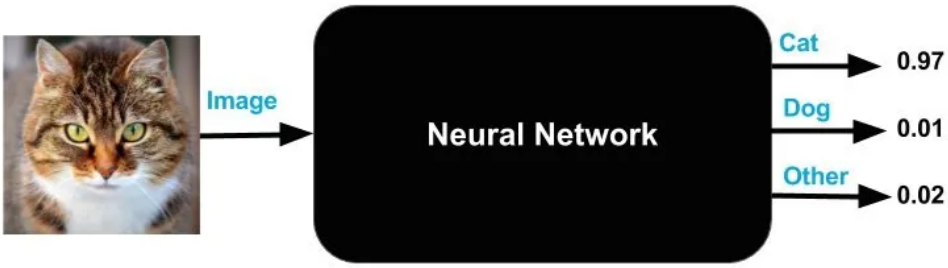
   Koeficijenti $\beta_0, \beta_1, \ldots, \beta_n$ se mogu jednostavno izračunati metodom najmanjih kvadrata (*eng. Least squares*). Cilj je minimizirati sumu kvadrata odstupanja između stvarnih vrijednosti zavisne varijable i predviđenih vrijednosti koje daje model. Postoji jednoznačnost rješenja, odnosno postoji jedinstvena kombinacija koeficijenata koja minimizira sumu kvadrata odstupanja. Također, kako je gore prikazano, postoji jasno definirana matematička jednadžba za izračunavanje navedenog; i ono što je najvažnije, postupak je reproducibilan. Drugim riječima, ako se postupak linearne regresije provodi na istim podacima koristeći metodu najmanjih kvadrata, rezultati će uvijek biti isti.

***black-box* model:** U *black-box* modelima, interna struktura i pravila donošenja odluka su ili skriveni, ili prekompleksni za razumijevanje čovjeku. Ovi modeli mogu pružiti točna predviđanja, ali postupak kako dolaze do tih predviđanja je netransparentan.

Za razliku od *white-box* modela i danog primjera linearne regresije, u *black-box* modelima parametri ne prenose konkretno značenje. Jedan od najpoznatijih i aktualnijih primjera *black-box* modela današnjice su modeli **dubokih neuronskih mreža**. Oni sadrže veliki broj parametara (težinski faktori + pristranost) raspoređenih u slojeve koji su međusobno povezani. Bez obzira na visoke performanse dubokih neuronskih mreža za određene probleme, parametri nemaju jasno fizičko značenje (npr. nisu povezani sa stvarnim fizičkim ili logičkim karakteristikama problema) i teško ih je interpretirati u kontekstu problema koji se rješava [13].

Poznat primjer upotrebe dubokih neuronskih mreža je prepoznavanje objekata na slici. Ovaj proces često uključuje korištenje konvolucijskih neuronskih mreža (*eng. Convolutional neural*

network (CNN)), koje su posebno dizajnirane za rad s vizualnim podacima. Slika 4 ilustrira proces klasifikacije slika korištenjem neuronske mreže, gdje je neuronska mreža apstraktno prikazana *black-box* modelom, a rezultati su vjerojatnosti, u rasponu [0-1], tj. pripadnost pojedinoj klasi. U sljedećem poglavlju detaljnije ćemo proučiti *black-box* model dubokih neuronskih mreža.



Slika 4: Klasifikacija slike korištenjem neuronskih mreža

3.2 Neuronske mreže i duboko učenje

Neuronske mreže, u strojnom učenju, posebna su vrsta modela koji se koristi za aproksimaciju nelinearnih funkcija, a inspirirane su strukturom i funkcijom ljudskog mozga [14]. Međutim, kod analogije s biološkim neuronima i mozgom treba biti vrlo oprezan, budući da biološki neuroni kompleksnije strukture koje mogu obavljati različite funkcije. Umjetne duboke neuronske mreže sastoje se od međusobno povezanih čvorova ili "neurona" koji su raspoređeni kroz višeslojnu strukturu [15].

U grubo, osnovna struktura neuronske mreže može se podijeliti na 3 dijela:

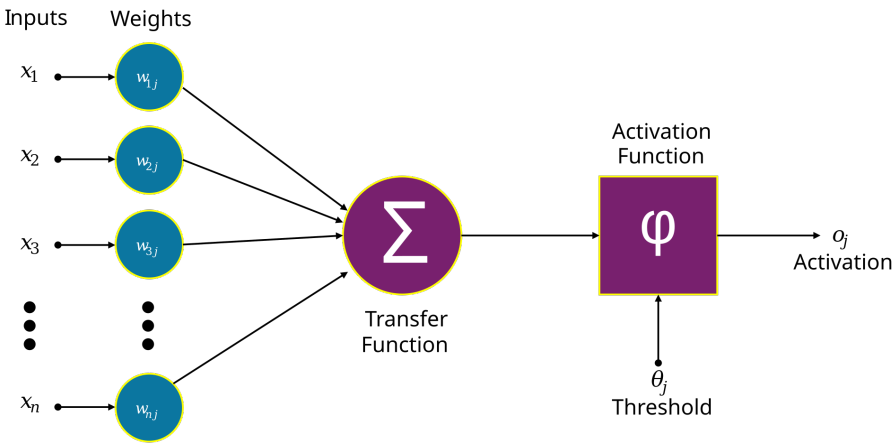
1. **Ulazni sloj** (*eng. Input layer*): ulazni podaci (npr. tekst, video, slika)

2. **Skriveni slojevi** (*eng. Hidden layers*): Slojevi između ulaznog i izlaznog sloja. Oni obrađuju podatke kroz niz transformacija i predstavljaju upravo ono što čini mrežu "dubokom". Ukratko, skriveni slojevi primjenjuju težinske faktore na ulazne podatke i propuštaju ih kroz aktivacijsku funkciju u sljedeći skriveni sloj ili izlazni sloj ako se radi o predzadnjem sloju. Aktivacijske funkcije su matematičke funkcije koje omogućavaju mreži da uči složene nelinearne odnose.

3. **Izlazni sloj** (*eng. Output layer*): Posljednji sloj neuronske mreže koji daje konačni rezultat.

Ključne komponente svake neuronske mreže su: umjetni neuroni, težinski faktori (*eng. weights*), aktivacijske funkcije (*eng. activation functions*), funkcije gubitka (*eng. loss functions*) i postupak optimizacije/učenja mreže (*eng. training process*) odnosno minimizacije funkcije gubitka.

Umjetni neuroni (*eng. Artificial neurons*): osnovne su jedinice u neuronskoj mreži. Svaki neuron prima ulazne vrijednosti $x_1, x_2, \ldots, x_n$, primjenjuje težinske faktore $w_1, w_2, \ldots, w_n$, zbraja ih i prosljeđuje aktivacijskoj funkciji. Opisani postupak prikazan je na Slici 5.



Slika 5: Struktura umjetnog neurona

Aktivacijske funkcije uvode nelinearnost u model, što omogućava neuronskim mrežama da uče složene obrasce u podacima. Bez pojma nelinearnosti, mreža bi se svela na linearnu kombinaciju ulaza te u konačnici na linearni klasifikator. Poznate aktivacijske funkcije uključuju:

- Sigmoidna funkcija (*eng. Sigmoid function*)
- Hiperbolična tangens funkcija (*eng. Hyperbolic tangent function (tanh)*)
- ReLU funkcija (*eng. Rectified Linear Unit*)
- Leaky ReLU funkcija (*eng. Leaky Rectified Linear Unit*)

Odabir aktivacijske funkcije ovisi o zadanom problemu i specifičnim potrebama neuronske mreže.

Klasifikacija slika je zadatak koji se odnosi na dodjeljivanje jedne ili više oznaka (klasa) slici na temelju njenog sadržaja. Tradicionalno, ovaj zadatak bio je izuzetno izazovan za računala zbog kompleksnosti vizualnih podataka i velikog broja varijacija u slikama (različiti kutovi, osvjetljenja, pozadine, itd.). Međutim, u posljednjih desetak godina, zahvaljujući razvoju strojnog učenja, a posebno dubokog učenja, kao i velikom napretku u razvoju hardvera, postignut je ogroman napredak u ovom području [16] [17] [18].

Problemu klasifikacije slika možemo pristupi i kroz linearne modele, npr. koristeći jednostavni linearni klasifikator. Prvo je potrebno odraditi korak pretprocesiranja podataka, u ovom slučaju ako se radi o slikama, potrebno ih je učitati, svesti na jedinstvenu dimenziju (npr. $32x32$ piksela) te normalizirati (obično između 0 i 1). Kako su slike u biti matrice piksela, za linearnu klasifikaciju trebamo ih pretvoriti u jednodimenzionalne vektore. Primjerice, slika veličine $32x32x3$ (3 RGB kanala) može se pretvoriti u vektor od 3072 elemenata.

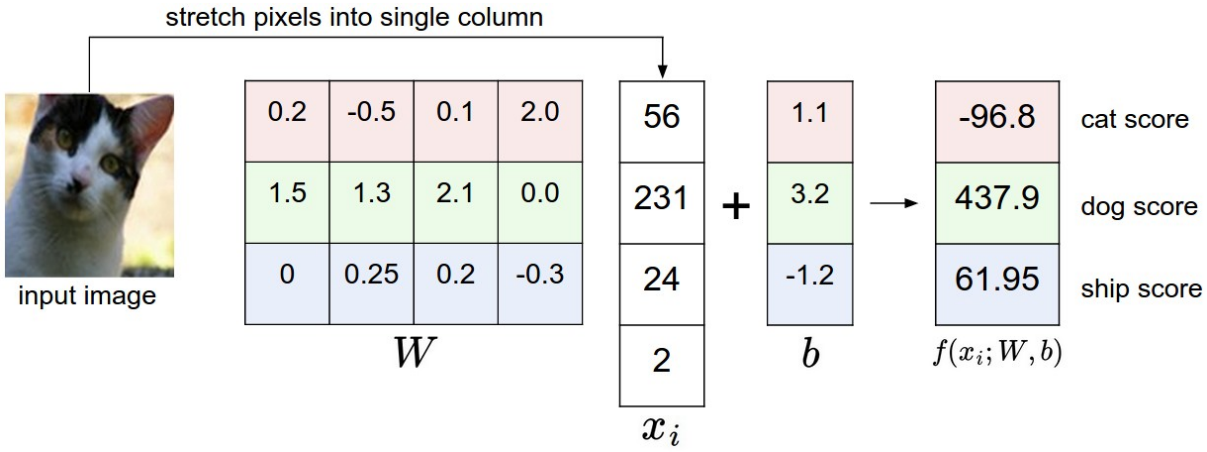
Naš linearni klasifikator sada možemo predstaviti na sljedeći način:

$$f(x) = W \cdot x + b \tag{2}$$

gdje su:

- x - vektor piksela ulazne slike,
- W - matrica težinskih faktora (*eng. weights*),
- b - vektor pomaka (*eng. bias*),
- $f(x)$ - rezultat klasifikatora koji određuje pripadnost klasi.

Slika 6 ilustrira postupak klasifikacije slike koristeći jednostavni linearni klasifikator.



Slika 6: Ilustracija linearnog klasifikatora na primjeru klasifikacije slika

Klasifikator bez treninga neće dati zadovoljavajuće rezultate. Za treniranje linearnog klasifikatora potrebno je definirati **funkciju gubitka** (*eng. loss function*). Funkcija gubitka ključna je komponenta u modelima strojnog učenja jer omogućava mjerenje koliko je klasifikator dobar odnosno loš. Bez nje nije moguće procijeniti performanse modela niti ga učinkovito trenirati.

Recimo da imamo 10 različitih klasa (oznaka) u koje želimo kategorizirati slike. To mogu biti primjerice: *avion, pas, ptica, mačka, srna, pas, žaba, konj, brod i kamion*. Kako bismo nastavili, podatke je potrebno podijeliti na trening skup, validacijski skup i testni skup, kako bismo kasnije u fazi inferencije modela dobili pouzdane procjene na neviđenim podacima. Trening skup koristi se za treniranje modela, validacijski skup koristi se tijekom treniranja kako bi se pratila izvedba modela na podacima koji nisu korišteni za treniranje, a testni skup se koristi jednom kad je model u potpunosti istreniran, u tzv. fazi inferencije (iskorištavanja) [15].

Definirat ćemo sljedeću funkciju gubitka: *Multiclass SVM (Support vector machine) loss*, poznatija kao **hinge loss** (često se koristi za treniranje klasifikatora [19]).

Funkcija je dana formulom:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

gdje su:

1. L_i - Vrijednost gubitka za pojedini uzorak treniranja i.

2. $\sum_{j \neq y_i}$ - Iteracija kroz sve klase j koje nisu ispravne. y_i predstavlja ispravnu klasu uzorka u trenutnoj iteraciji.

3. $\max(0, s_j - s_{y_i} + 1)$ - Ovaj dio jednadžbe računa funkciju gubitka za svaku krivu klasu j.

   - s_j: Predstavlja vrijednost *score funkcije* (rezultat) za klasu j koju je model predvidio.

   - s_{y_i}: Ovo je vrijednost *score funkcije* (rezultat) za istinitu (stvarnu) klasu y_i.

   - $s_j - s_{y_i} + 1$: Računa razliku između rezultata za klasu j i istinitu (stvarnu) klasu y_i, uz dodatak fiksne marginale od 1.

   - $\max(0, \cdot)$: Osigurava da samo pozitivne razlike u rezultatima doprinose gubitku.

Ako je rezultat *score funkcije* za klasu j, odnosno s_j veći za više od 1 + s_{y_i} (1 + rezultat za stvarnu klasu) tada će $s_j - s_{y_i} + 1$ biti pozitivan broj i to će biti dodano u gubitak. Ako je manji, tada će $s_j - s_{y_i} + 1$ biti 0 i nema "doprinosa" gubitku.

Dakle, *Multiclass SVM* funkcija penalizira model kada je rezultat za pogrešnu klasu previše blizu ili veći od rezultata za stvarnu tj. istinitu (*eng. ground truth*) klasu. Ukupni gubitak možemo definirati kao prosječni gubitak na cijelom skupu podataka:

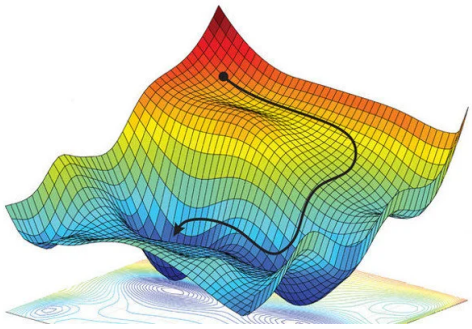
$$L = \frac{1}{N} \sum_{i=1}^{N} L_i$$

U praksi, potrebno je nadograditi funkciju gubitka regularizacijom (*eng. regularization*). Izrazi regularizacije dodaju se u svrhu smanjenja prenaučenosti (*eng. overfitting*) modela penalizacijom velikih težinskih faktora. Poznatije metode uključuju *L1* i *L2* regularizaciju [20].
Ukupan gubitak s dodanom regularizacijom izgleda ovako:

$$L = \frac{1}{N} \sum_{i} L_i(f(x_i, W), y_i) + \lambda R(W)$$

Jednom kada imamo definiranu funkciju gubitka, možemo istrenirati naš linearni klasifikator, primjerice koristeći algoritam stohastičkog gradijentnog spusta (*eng. Stochastic Gradient Descent (SGD)*). Metoda stohastičnog gradijentnog spusta koristi iterativni pristup za optimizaciju modela, ažurirajući težinske faktore modela na temelju izračunate funkcije gubitka na malim nasumičnim podskupovima (*eng. mini-batches*) trening skupa podataka. Parametri mreže koje algoritam može mijenjati su faktorske težine (W) i pomaci (b), dok su ulazni podaci, u ovom slučaju slike i njihove oznake, nepromjenjivi (X, y). Linearni klasifikatori mogu modelirati

samo linearno-razdvojive podatke (*eng. Linear separability*), što često nije dovoljno za složenije probleme. Uz to, često nemaju dovoljnu fleksibilnost za prilagodbu složenijim strukturama podataka, što može dovesti do niske točnosti na stvarnim podacima. Na Slici 7 prikazana je trajektorija gradijentnog spusta (eng. *trajectory for gradient descent*) koja ilustrira kretanje funkcije cilja prema lokalnom minimumu u trodimenzionalnom prostoru. Lokalni minimum je označen tamno plavom bojom, dok se početno inicijalizirani težinski faktori nalaze oko crvenog područja gdje je vrijednost funkcije gubitka visoka.



Slika 7: Trajektorija gradijentnog spusta prikazuje kretanje funkcije cilja prema lokalnom minimumu

Neuronske mreže su sofisticiran pristup za rješavanje problema kao što je klasifikacija, koji se ističe u suočavanju s kompleksnim i linearno nerazdvojivim podacima. U ovim mrežama, neuroni su slojevito organizirani, pri čemu svaki neuron koristi aktivacijsku funkciju kako bi unio nelinearnost u model.

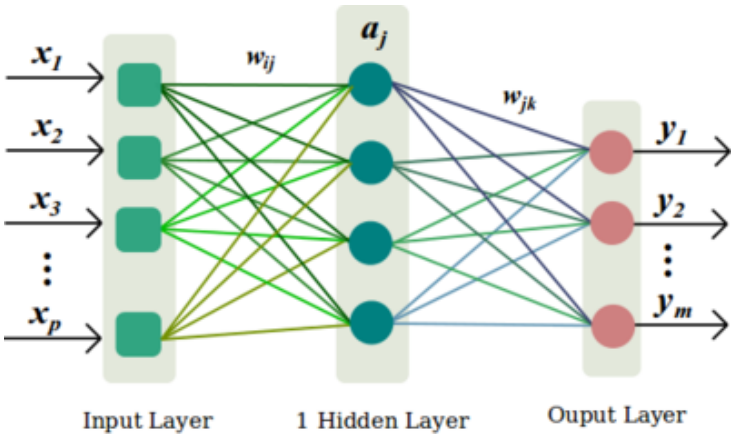
Uvođenjem aktivacijske funkcije, primjerice *ReLU*, naš jednostavni linearni klasifikator postaje neuronska mreža s 2 sloja.

$$f = W_2 \max(0, W_1 x)$$

gdje su:

- x - vektor piksela ulazne slike,
- W_1 - matrica težinskih faktora 1. sloja koja transformira ulazni vektor,
- $\max(0, \cdot)$ - aktivacijska funkcija ReLU,
- W_2 - matrica težinskih faktora 2. sloja. koja transformira rezultat ReLU funkcije,
- f - rezultat neuronske mreže nakon W_2 transformacija.

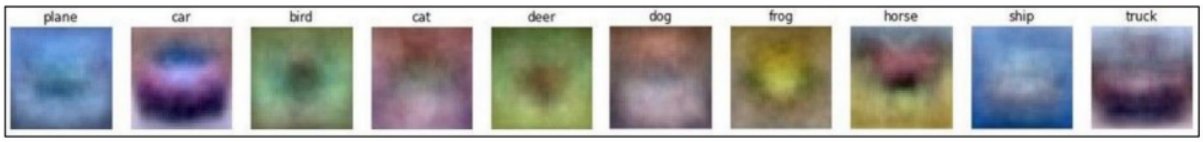
Izraz $\max(0, W_1 x)$ koristi aktivacijsku funkciju *ReLU* ($\max(0, z)$), koja zapravo sprječava "povratak" na linearni klasifikator. Grafički, ovu neuronsku mrežu s jednim skrivenim slojem možemo prikazati na sljedeći način (Slika 8):



Slika 8: Neuronska mreža s jednim skrivenim slojem

Pretvaranjem linearnog klasifikatora u jednostavnu neuronsku mrežu s jednim skrivenim slojem značajno se poboljšava sposobnost modela za učenje i prepoznavanje složenih obrazaca u slikama. Skriveni sloj omogućava modelu da uči nelinearne odnose između ulaznih podataka i klasa, što rezultira boljom točnošću klasifikacije. Recimo da je veličina skrivenog sloja mreže 100, neuronska mreža uči 100 predložaka i raspoređuje ih među klasama, umjesto da se oslanja samo na 10 klasa kao što je slučaj kod linearnog klasifikatora. Dodavanjem slojeva povećava se fleksibilnost i preciznost u prepoznavanju obrazaca te mreža dobiva na moći generalizacije [21]. Slika 9 prikazuje primjer predložaka koje CNN uči. U ovom slučaju, budući da mreža ima samo jedan skriveni sloj, na predlošcima se mogu uočiti obrisi stvarne slike.

Osim toga, neuronska mreža s jednim skrivenim slojem može bolje generalizirati na nove - neviđene podatke, u usporedbi s jednostavnim linearnim klasifikatorom.



Slika 9: Primjer predložaka koje neuronska mreža uči i dodjeljuje klasama

Duboko učenje predstavlja podskup strojnog učenja koji se specifično odnosi na višeslojne neuronske mreže. Ove mreže koriste višestruke slojeve kako bi prepoznale i naučile složene obrasce i strukture unutar podataka. Zahvaljujući ovoj višeslojnoj arhitekturi, duboke mreže postižu superiorne performanse u poređenju s plitkim modelima, kao što su jednostavne neuronske mreže s jednim skrivenim slojem ili linearni klasifikatori. Kroz hijerarhiju slojeva, duboke mreže mogu izvući značajke na različitim razinama apstrakcije, što im omogućava rješavanje kompleksnih zadataka koji su izazovni za tradicionalne algoritme strojnog učenja [21].

Iako koncept dubokog učenja nije nov i datira još iz 1940-ih godina, značajan napredak u ovom polju ostvaren je još 1989. godine kada su razvijene tehnike unazadne propagacije (*eng. backpropagation*) i konvolucijskih neuronskih mreža za prepoznavanje ručno pisanih znamenki kroz arhitekturu **LeNet-5** na kojoj je radio Yann LeCun [22]. Međutim, duboko učenje je doži-

vjelo pravu eksploziju u zadnjih desetak godina zbog nekoliko ključnih faktora. Prvo, količina dostupnih podataka je drastično porasla s rastom interneta i razvojem digitalizacije, omogućujući treniranje dubokih mreža na velikim skupovima podataka. Drugo, skorašnji napredak u računalnoj snazi, posebno s grafičkim procesorskim jedinicama (*GPU-ovima*), omogućio je bržu i efikasniju obradu tih podataka. Treće, algoritamski napreci, uključujući bolje tehnike regularizacije, optimizacije i arhitekture mreža, značajno su poboljšali performanse i stabilnost modela dubokog učenja.

3.3 Obrada prirodnog jezika (NLP)

Natural Language Processing (NLP) je interdisciplinarno područje računalne znanosti koje uključuje umjetnu inteligenciju i jezikoslovlje (lingvistiku) - znanost u kojoj su predmet istraživanja jezik i govor. *NLP* područje vrlo je složeno budući da je jezikoslovlje samo po sebi interdisciplinarna znanost koja uključuje proučavanje jezika te kognitivnih i psiholoških procesa pri njegovoj uporabi [23].

3.3.1 Povijest strojnog prijevoda jezika

Obrada prirodnog jezika, u daljnjem tekstu *NLP*, predstavlja područje istraživanja koje se bavi razumijevanjem, interpretacijom i generiranjem ljudskog jezika putem računalnih sustava. Cilj je razvoj tehnologija koje omogućuju računalima da komuniciraju na način koji je prirodan za ljude. Premda se radi o relativno modernom području, njegovi korijeni sežu u rane 1950-e godine. Jedan od prvih značajnih događaja u povijesti NLP-a bio je "Georgetown–IBM eksperiment" proveden na Sveučilištu Georgetown u SAD-u, u suradnji s IBM-om. Ovaj eksperiment demonstrirao je strojno prevođenje više od 60 rečenica s ruskog na engleski jezik [24].

Statističko strojno prevođenje: *"Georgetown–IBM"* eksperiment pokazao se uspješnim, no do 1980-ih godina nije bilo značajnijih pomaka u području strojnog prevođenja. Pravi napredak započeo je tek s razvojem statističkog strojnog prevođenja. Statističko strojno prevođenje (*eng. Statistical machine translation (SMT)*) metoda je prevođenja teksta pomoću statističkih modela temeljenih na analizi dvojezičnih tekstova (paralelnih korpusa). Ovaj model zamijenio je inicijalna rješenja koja su bila bazirana na pristupu temeljenom na pravilima (*eng. Rule-based approach*) gdje je bilo potrebno eksplicitno definirati jezična pravila. Takav pristup je vrlo skup u pogledu računalnih resursa i vremena, a mogućnosti generalizacije su slabe. Statističko strojno prevođenje proizlazi iz teorije informacija (*eng. Information theory*). Ovom metodom, dokument se prevodi prema distribuciji vjerojatnosti: $p(e|f)$ vjerojatnost da je niz znakova e u ciljanom jeziku (npr. engleski) ispravan prijevod niza znakova f u izvornom jeziku (npr. francuskom). Problem modeliranja ove distribucije vjerojatnosti pokušao se rješavati na mnoge načine, uključujući i primjenu *Bayesovog teorema* [25].

Problemi koje metoda statističkog strojnog prevođenja nije uspjela riješiti uključuju:

- **Poravnanje rečenica**: U paralelnim korpusima često se događa da pojedinačne rečenice u jednom jeziku budu prevedene u više rečenica u drugom jeziku ili obrnuto. Dodatno, neki jezici nemaju jasne oznake kraja rečenice, što otežava pravilno poravnanje.
- **Poravnanje riječi**: Često se susreću riječi koje nemaju jasan ekvivalent u ciljanom jeziku, što predstavlja izazov pri prevođenju. Primjerice, pri prevođenju s engleskog na hrvatski, u rečenici "She does not eat meat," riječ "does" nema direktan ekvivalent u rečenici "Ona ne jede meso." U ovom slučaju, riječ "does" sadrži gramatičku informaciju koja je u hrvatskom jeziku obuhvaćena unutar glagola "jede" i negacije "ne."
- **Prevođenje idioma**: Idiom je fraza čije se značenje ne može razumjeti doslovnim tumačenjem pojedinačnih riječi. U englesko-francuskom prevođenju, idiom "piece of cake" mogao bi se doslovno prevesti kao "morceau de gâteau" (komad kolača), umjesto pravilnog idiomatskog prijevoda "C'est du gâteau" koji glasi "to je lako" ili "ništa lakše". Iz ovog razloga, idiomi su se mogli poravnati samo frazno; budući da se ne mogu rastaviti bez gubitka značenja.
- **Različit redoslijed riječi u rečenici**: Redoslijed riječi u rečenici značajno varira među jezicima. Primjerice, engleski jezik obično slijedi redoslijed *Subjekt-Glagol-Objekt* (*eng. Subject-Verb-Object (SVO)*), dok neki drugi jezici, poput japanskog, imaju drugačiji redoslijed riječi, što otežava izravno prevođenje rečenica bez gubitka značenja i prirodnog toka.

$$\text{Subject} \longrightarrow \text{Verb} \longrightarrow \text{Object}$$

Naravno, u mnogim jezicima redoslijed riječi nije uvijek isti, a promjena poretka često mijenja značenje rečenice. Hrvatski jezik tipično koristi SVO (subjekt-glagol-objekt) redoslijed, no zbog padeža ima fleksibilniji raspored riječi. Primjerice, rečenicu "Mačka (subjekt) je lovila (glagol) miša (objekt)" možemo preoblikovati u "Miša je lovila mačka" bez promjene značenja, jer padeži određuju ulogu svake riječi u rečenici.

Pasivni oblik, koji je u engleskom jeziku vrlo zastupljen, u hrvatskom jeziku ima drugačiju upotrebu i rijetko se koristi. Stoga rečenicu na engleskom "The mouse was being chased by the cat." nećemo prevesti u pasivu: "Miš je bio lovljen od strane mačke." Umjesto toga, koristit ćemo aktivni oblik ili refleksivni glagol kako bi rečenica zvučala prirodnije.

Prevođenje teksta uz pomoć neuronskih mreža: Počevši od 2003. godine pa sve do sredine 2010-ih, polagano ali sigurno došao je kraj statističkom strojnom prevođenju zahvaljujući razvoju dubokog učenja i neuronskih mreža. Google, kao jedan od vodećih korisnika statističkog strojnog prevođenja sa svojim alatom *Google Translate*, 2016. godine najavljuje kroz znanstveni

rad "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation" prijelaz na **prevođenje temeljeno na neuronskim mrežama** *(eng. Neural Machine Translation - NMT*) [26].

NMT se temelji na korištenju neuronskih mreža za predikciju vjerojatnosti sljedećeg niza riječi, tipično modelirajući cijele rečenice u jedan integrirani model. Paralelno s razvojem različitih rješenja i metoda za prijevod jezika, razvijale su se i generativne metode za generaciju samog teksta.

Prema tome, razvoj NLP-a kao znanosti, možemo jednako tako podijeliti u 3 faze:

1. **Simbolički NLP** (1950-e - 1990-ih): U razdoblju od 1950-ih do 1990-ih, simbolički pristup obradi prirodnog jezika (NLP) temeljio se na skupu definiranih pravila. Računala su simulirala razumijevanje jezika (ili obavljala druge NLP zadatke poput prevođenja) primjenom tih pravila na podatke. Jedan od prvih značajnih projekata bio je *ELIZA*, razvijena na MIT-u između 1964. i 1967. godine. ELIZA je simulirala razgovor s psihoterapeutom koristeći jednostavna pravila i obrasce za prepoznavanje teksta, čime je pokazala kako računala mogu imitirati ljudski razgovor [27]. Drugi značajan primjer iz tog razdoblja je Mindscapeov *Racter*, koji je bio sposoban generirati prozu na engleskom jeziku [28]. Napredak u području obrade prirodnog jezika bio je usporen ograničenjima tadašnjih računala i nedostatkom dostupnih podataka, što je značajno utjecalo na mogućnosti primjene i daljnji razvoj NLP tehnologija.

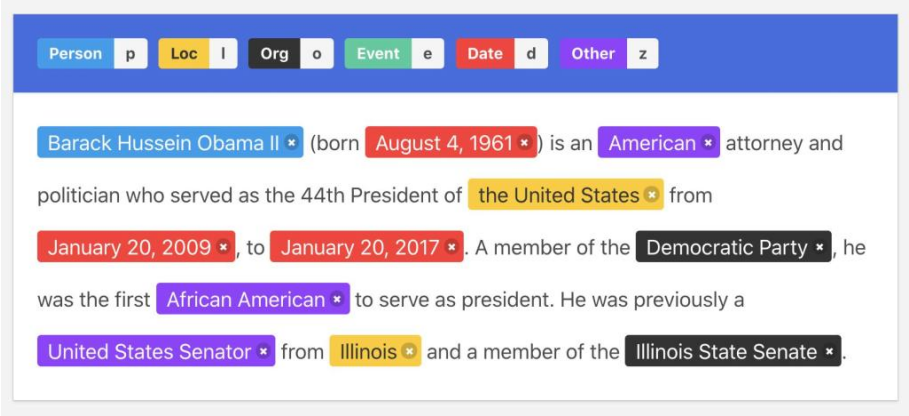
2. **Statistički NLP** (1990-e - 2010-ih): Veliki broj uspješnih metoda iz ovog područja vezano je uz strojno prevođenje, ponajviše zbog velikog angažmana IBM-a. Ove metode temeljile su se na statističkim modelima koji su koristili velike korpuse podataka za treniranje. Međutim, veliki broj tih sustava bio je izrazito ovisan o korpusu podataka, što je ujedno bio i glavni nedostatak. Velikim razvojem Interneta ranih 2000-ih, povećao se broj jezičnih podataka koji su postali javno dostupni, što je omogućilo razvoj novih tehnika nenadziranog (*eng. unsupervised*) učenja, budući da glavnina podataka s web stranica tada nije bila označena/labelirana. Također, tijekom ovog razdoblja razvijeni su algoritmi kao što su skriveni Markovljevi modeli (*eng. Hidden Markov Models (HMMs)*) i modeli maksimalne entropije (*eng. Maximum entropy models*) koji su postali standardni alati u statističkom NLP-u. Korištenje statističkih metoda omogućilo je veću fleksibilnost i prilagodljivost sustava, ali su i dalje postojali izazovi vezani uz razumijevanje konteksta i semantike jezika.

3. **NLP baziran na dubokom učenju** (danas): Višeslojni perceptron (Standardna *feed-forward* neuronska mreža) nadmašuje 2003. godine *word-n-gram* statistički model, koji je do tada bio *de facto* standard u statističkom NLP-u [29]. Godine 2010., Tomáš Mikolov, tada doktorand na Sveučilištu Brno u Češkoj, zajedno s koautorima primijenio je jednostavni RNN s jednim skrivenim slojem za modeliranje jezika. Tijekom narednih godina, zajedno s kolegama na Googleu, razvio je **Word2vec**, tehniku koja pretvara riječi s

njihovim semantičkim značenjem u vektore koristeći neuronske mreže. Tijekom 2010-ih, razvoj dubokog učenja naglo je proširio i NLP područje. Popularnost tih metoda dijelom je rezultat niza istraživanja koja su pokazala da takve tehnike mogu postići vrhunske rezultate u mnogim NLP zadacima [30] [15].

Zaključno, u kontekstu NLP-a, glavni nedostatak metoda baziranih na statistici je potreba za složenim inženjeringom značajki (*eng. Feature engineering*). Inženjering značajki odnosi se na proces odabira i transformacije ulaznih podataka, poput riječi i rečenica, u korisnije značajke za model. Ovaj proces uključuje razne metode za reprezentaciju tekstualnih podataka, uključujući:

- **BoW** (*eng. Bag of Words*): jednostavna tehnika reprezentacije teksta/dokumenta pomoću frekvencije pojavljivanja svake riječi u dokumentu.
- **TF-IDF** (*eng. Term Frequency-Inverse Document Frequency*): mjera važnosti riječi u dokumentu ili korpusu, uz pretpostavku da se neke riječi pojavljuju češće od drugih.
- **N-grami** (*eng. n-grams*): Kontinuirane sekvence od *n* riječi ili simbola.
- **POS oznake** (eng. *Part-of-Speech (POS) tags*): oznake koje označavaju gramatičko značenje riječi, (npr. glagol, imenica, pridjev, zamjenica).
- **NER** (*eng. Named Entity Recognition*): Identifikacija i klasifikacija entiteta u tekstu (npr. osobno ime, lokacija, datum, grad itd.). Prikazano na Slici 10.

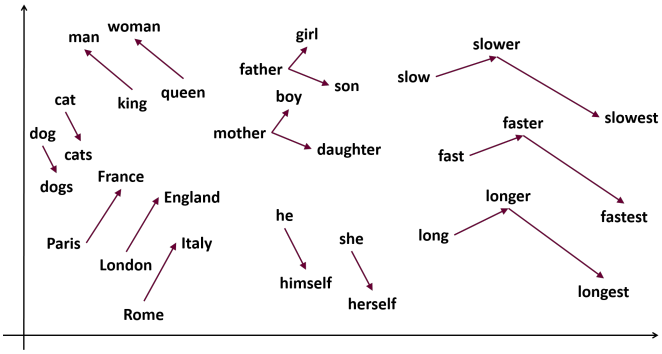


Slika 10: Primjer *NER* tehnike za identifikaciju i klasifikaciju entiteta iz teksta

Izrada kvalitetnih značajki nije jednostavan posao i iziskuje puno vremena i tehničko-domenskog znanja. Uz to, inženjering značajki za određeni NLP problem/zadatak često ima loše mogućnosti generalizacije na druge zadatke. Ručni inženjering značajki također pojednostavljuje podatke te se na taj način teže može obuhvatiti puna semantika riječi i nijanse prirodnog jezika. Primjerice, pri analizi sentimenta teksta, ručni inženjering može uključivati jednostavno brojanje pozitivnih i negativnih riječi. Međutim, ovakav pristup ne može prepoznati sarkazam

ili riječi koje mijenjaju značenje ovisno o kontekstu, poput "Ovo je baš fantastično" izrečeno s negativnom konotacijom.

Tehnike ugradnje riječi (*eng. Word embedding techniques*) poput Word2Vec automatski "uče" vektorske reprezentacije riječi koje odražavaju njihove semantičke odnose. Ove tehnike koriste kontinuirani prostor visokih dimenzija u kojem se svaka riječ prikazuje kao vektor. Vektori riječi koji su semantički slični nalaze se blizu jedan drugome u ovom prostoru, omogućujući da slične riječi imaju slične numeričke reprezentacije. Slika 11 ilustrira semantički srodne riječi u 2D reprezentaciji višedimenzionalnog vektorskog prostora.



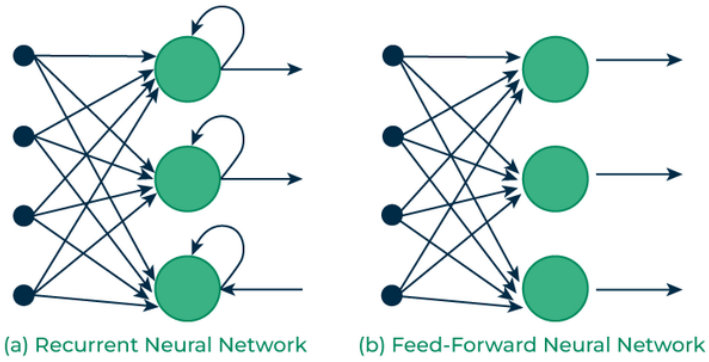
Slika 11: *Word2Vec* - Ilustracija odnosa semantički srodnih riječi u 2D reprezentaciji vektorskog prostora

3.3.2 RNN i Sekvencijalni modeli

Rekurentne neuronske mreže (RNN): poznate kao i povratne neuronske mreže, posebna su vrsta neuronske mreže dizajnirana za rad s podacima u nizu, poput vremenskih serija, ali češće prirodnog jezika. Rekurentne neuronske mreže (*eng. Recurrent Neural Networks*) imaju "unutarnje petlje" koje im omogućuje očuvanje informacija kroz različite vremenske korake. Za razliku od standardnih mreža s prolaskom unaprijed (*eng. feedforward*), koje su objašnjene u poglavlju 3.2., RNN mreže omogućuju očuvanje informacija kroz različite vremenske korake.

Drugim riječima, ovaj oblik mreže prosljeđuje rezultate aktivacija prethodnog koraka kao ulazne podatke u trenutni korak. U usporedbi s tradicionalnim neuronskim mrežama, gdje su svi ulazi i izlazi nezavisni jedni o drugima, ovaj pristup koristi **sekvencijalnu ovisnost podataka**.

U *feedforward* mrežama, "informacije" se kreću samo u jednom smjeru: od ulaznog sloja prema izlaznom sloju, te kroz skrivene slojeve ako postoje. Ove mreže pogodne su za zadatke poput klasifikacije slika gdje su ulazni i izlazni podaci potpuno različiti i neovisni (*ulazi* su vektori piksela slika, a *izlazi* su vrijednosti koje predstavljaju odabrane klase). Međutim, to nije slučaj kod problema gdje su ulazni podaci tekstulanog tipa, primjerice prirodni govor. Slika 12 prikazuje osnovnu usporedbu u strukturi *feedforward* i rekurentne neuronske mreže.



Slika 12: *Recurrent* VS *Feedforward* neuronske mreže

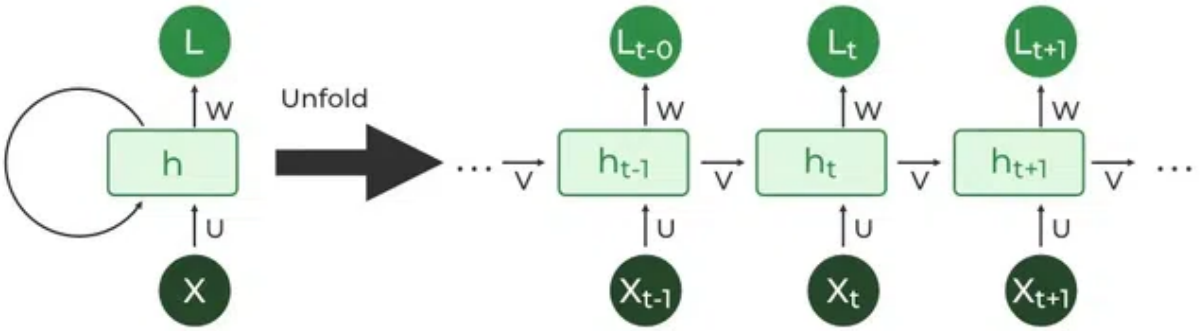
Slika prikazuje saženu (lijevo) i razmotanu (desno) jednostavnu rekurentnu neuronsku mrežu. Glavna i najvažnija značajka RNN-a je tzv. skriveno stanje (*eng. hidden state*), koje pamti neke informacije o sekvenci. Povratni neuron, ili RNN blok (*eng. Recurrent Unit*), predstavlja osnovnu gradivnu jedinicu ove mreže. Ovaj blok posjeduje sposobnost pamćenja skrivenog stanja, što u konačnici omogućuje mreži da zadrži određeno pamćenje povijesti niza.

Na Slici 13 (lijevo) se može uočiti skriveno stanje RNN mreže h koje pamti informacije o sekvenci iz prethodnih koraka. X je ulazna sekvenca na početku svakog koraka, a L je vrijednost funkcije gubitka, odnosno *izlaz* svakog koraka.

Oznake U, W, V predstavljaju vrijednosti težinskih faktora u različitim trenucima:

- U predstavlja težine između ulaznog vektora i skrivenog stanja h
- W predstavlja težine između skrivenog stanja i izlaza L
- U predstavlja rekurentne težinske faktora iz prethodnih stanja $h - 1$ prema trenutnom skrivenom stanju h

Desni dio ilustracije na Slici 13 prikazuje skrivena stanja u "razmotanom" obliku, odnosno prikazuje kako izgleda opisani iterativni proces unutar RNN blokova.



Slika 13: Recurrent Neural Network

S obzirom na broj ulaza odnosno izlaza RNN-a, u literaturi se navodi 4 vrste ovih mreža:

1. **Ono to One** - Standardna *feedforward* neuronska mreža gdje jedan ulaz vodi do jednog izlaza, ovdje nema sekvenci.

2. **One to Many** - Jedan ulaz proizvodi sekvencu izlaza, npr. tekst koji opisuje ulaznu sliku.

3. **Many to One** - Sekvenca ulaza, poput rečenice, vodi do jednog izlaza, npr. u svrhu analize sentimenta rečenice.

4. **Many to Many** - tzv. *sequence to sequence* modeli, npr. za potrebe strojnog prevođenja, generiranja teksta, odgovaranje na pitanja i sl.

U praksi, klasični RNN-ovi imaju probleme s učenjem dalekosežnih veza zbog problema nestajanja gradijenta i ograničenog kapaciteta memorije [31]. Tijekom treniranja RNN-ova, gradijenti pogrešaka za potrebe izračuna funkcije gubitka propagiraju se unatrag koristeći prilagođeni *Backpropagation through time* algoritam - algoritam unazadne propagacije kroz vrijeme. Kao varijanta algoritma *backpropagacije*, prilagođen je radu s rekurentnim arhitekturama kako bi se nosio s njenim posebnostima, kao što su povratne petlje u mreži. Kod dalekosežnih veza, gradijenti mogu postati vrlo mali (nestajanje gradijenta) ili vrlo veliki (eksplodiranje gradijenta). Nestajanje gradijenta dovodi do toga da se težine faktora minimalno mijenjaju tijekom treniranja, što mreži otežava učenje dugoročnih relacija [32].

Kako bi se adresirali ovi problemi, razvijene su naprednije arhitekture poput *Long Short-Term Memory* (LSTM) mreže, modifikacije rekurentne neuronske mreže gdje se perceptron zamjenjuje s tzv. LSTM jedinicom koja "pamti" informacije proizvoljno dugo te na neki način doprinosi kratkotrajnoj memoriji RNN-a koja sada može trajati na tisuće iteracija, odakle i naziv *Long Short-Term Memory*. LSTM ćelija može se zamisliti kao svojevrsna "podmreža" koja ima vlastitu rekurenciju. Svaka takva mreža ima svoj sustav kontrolnih sklopova, koji se sastoji od ulaznog sklopa, sklopa za zaboravljanje i izlaznog sklopa. LSTM mreže su 1997. uveli Hochreiter i Schmidhuber kao prijedlog rješenja problema nestajućeg ili eksplodirajućeg gradijenta [33].

Sequence to sequence modeli (*Seq2Seq*): posebna su vrsta RNN-ova koja radi na *"Many to many"* principu, odnosno transformiraju ulaznu sekvencu podataka u izlaznu sekvencu, npr. rečenicu. Ovaj oblik RNN-a predstavili su prvi puta znanstvenici iz Googlea u radu "Sequence to Sequence Learning with Neural Networks". Cilj ovih modela je preslikavanje ulazne sekvence fiksne duljine na izlaznu sekvencu fiksne duljine, pri čemu duljina ulazne sekvence može biti različita od duljine izlazne sekvence. Primjerice, želimo rečenicu *"Pozdrav kako si?"* prevesti na engleski jezik.

$$\text{Pozdrav kako si? (3)} \longrightarrow \text{Hello how are you? (4)}$$

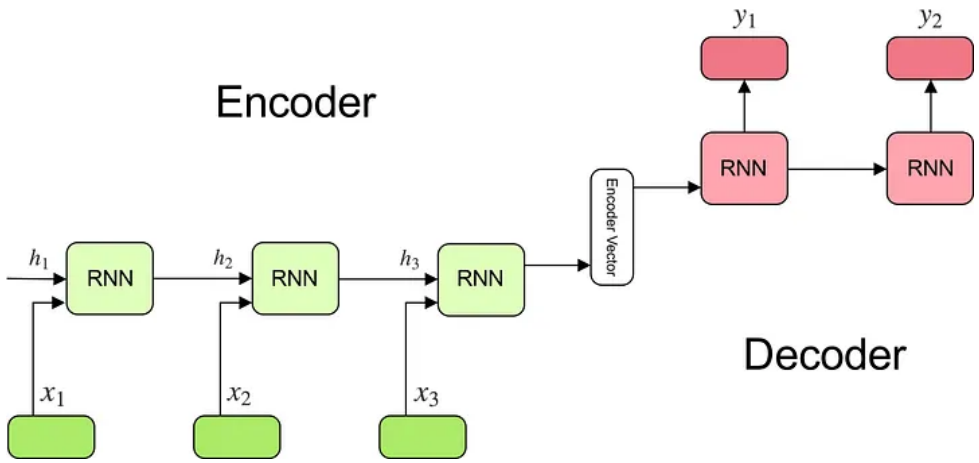
Obična LSTM mreža ne može riješiti ovaj problem budući da su duljine ulaza i izlaza različite, odnosno rečenica na hrvatskom ima 3 riječi, dok na engleskom ima 4 riječi. Model

Seq2Seq posebno je dizajniran za rješavanje ovakvih problema. Sastoji se od dva ključna dijela: *encoder* i *decoder*.

1. **Encoder**: skup sekvencijalnih rekurentnih jedinica (u pravilu LSTM ili GRU radi boljih performansi) koji prima ulaznu sekvencu (npr. rečenicu na engleskom) i pretvara ju u kontekstni vektor (*eng. context vector*), koji sažima sve informacije ulazne sekvence u fiksne duljine. Međutim, *encoder* ne mora imati nužnu fiksnu duljinu za kontekstni vektor; ona može biti i promjenjiva ako se koristi *mehanizam pažnje*, više o tome u nastavku.

2. **Decoder**: skup sekvencijalnih rekurentnih jedinica koje primaju kontekstni vektor od *encodera* i generiraju izlaznu sekvencu (npr. rečenicu na hrvatskom). Svaka rekurentna jedinica prima skriveno stanje od prethodne jedinice i vraća sekvencu na temelju tog i svog vlastitog skrivenog stanja. *Decoder* se također može nadograditi mehanizmom pažnje (*eng. attention mechanism*) kako bi dinamički odabrao relevantne dijelove kontekstnog vektora pri generiranju svake riječi u izlaznoj sekvenci.

Opisana *encoder-decoder* struktura sekvencijalnog modela može se vidjeti na Slici 14. Kontekstni vektor prikazan je na sredini, kao rezultat *encodera*.



Slika 14: Ilustracija *encoder-decoder* strukture *seq2seq* modela

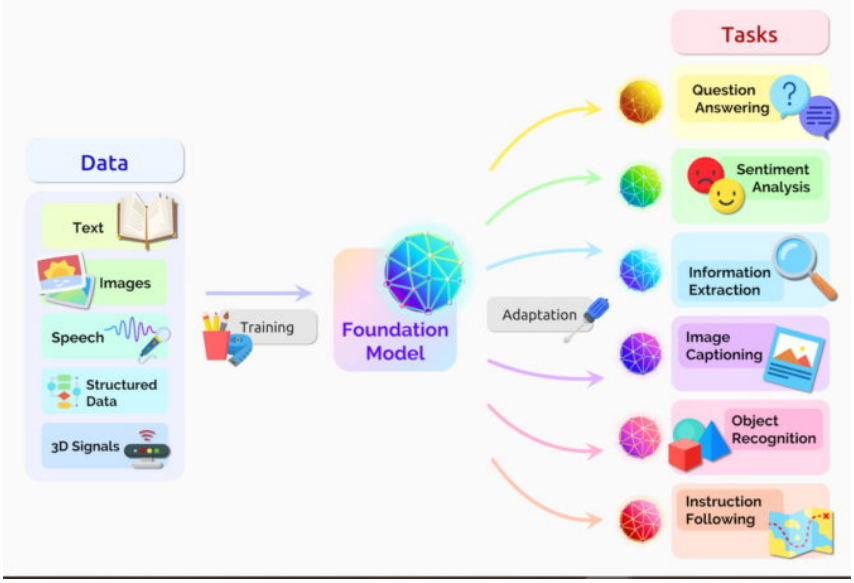
3.3.3 Transformer arhitektura

Transformer je arhitektura modela dubokog učenja koja se temelji na mehanizmu pažnje (*eng. Attention mechanism*) za efikasnu obradu sekvencijalnih podataka, prvenstveno teksta. Ovaj mehanizam omogućuje modelima obradu teksta na dosad neviđen način, uz dublje razumijevanje konteksta i semantičkog značenja riječi i fraza. Sve je započelo sada već legendarnim radom "Attention is All You Need", koji su 2017. godine objavili Ashish Vaswani i njegov tim znanstvenika iz Googlea. Ovaj rad predstavlja evoluciju Bahdanauovih mehanizama pažnje iz 2015. godine [34], omogućujući modelima da dinamički "obraćaju pažnju" na relevantne dijelove ulazne sekvence tijekom procesa prevođenja ili generiranja teksta. Ova inovacija

značajno je poboljšala performanse u zadacima poput strojnog prevođenja i obrade prirodnog jezika. U radu "Attention is All You Need" predstavljen je *encoder-decoder* transformer model veličine od oko 65 milijuna parametra, što je bilo revolucionarno za to, ne tako davno, vrijeme.

Znanstvenici sa Stanforda nazvali su, u jednom radu iz 2021. godine, ove modele "temeljnim modelima" (*eng. Foundation models*) budući da ih vide kao ključne komponente na kojima se temelji sav daljnji razvoj umjetne inteligencije [35]. Transformer arhitektura ističe se zbog svoje sposobnosti da paralelno obrađuje podatke, što je značajna prednost u odnosu na prethodne sekvencijalne modele poput RNN-a ili LSTM-a koji obrađuju ulazne podatke "riječ po riječ". Ovi modeli danas predstavljaju temelje mnogih modernih AI aplikacija te imaju široki spektar primjene: od klasičnih NLP problema poput generiranja teksta, prijevoda, odgovaranja na pitanja, sažimanja teksta pa sve do računalnog vida; uključujući probleme klasifikacije, detekcije objekata i lokalizacije objekata. Slika 15 prikazuje primjene modernih transformer arhitektura.

Stoga, nije iznenađujuće da transformer modeli danas postupno zamjenjuju CNN i RNN arhitekture, koje su do prije samo nekoliko godina bile daleko najpopularnije.



Slika 15: Transformer modeli, ponekad zvani i *foundation* modelima, danas imaju jako široku upotrebu

3.3.4 Kako rade transformer modeli?

Kako je već rečeno, transformer modeli revolucionizirali su cijelo znanstveno područje, a posebno NLP ogranak, zahvaljujući svojoj sposobnosti da paralelno obrađuju ulazne sekvence, što ih čini iznimno brzim u treniranju i inferenciji. Štoviše, osim brzine, transformeri rješavaju još jedan problem kod RNN i LSTM arhitektura, a to je ograničeno pamćenje konteksta kada je "udaljenost" između dijelova rečenica velika [34].

Postoje dvije velike inovacije koje dolaze s **transformer modelima**:

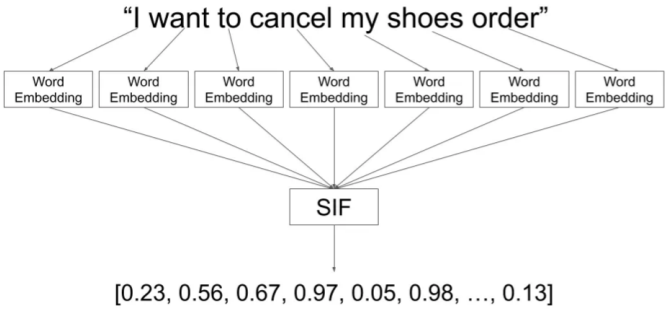
1. **Pozicijsko kodiranje** (*eng. Positional encoding*): Kako transformeri sami po sebi ne prate

redoslijed ulaznih elemenata (sekvenci) budući da se ulazne sekvence obrađuju paralelno, potreban je mehanizam za predstavljanje informacije o njihovom redoslijedu. Pozicijsko kodiranje je dodatni vektor koji se dodaje svakom ulaznom vektoru, pružajući informacije o položaju svakog *tokena* (*tokeni* su dijelovi sekvence, poput cijelih riječi ili dijelova riječi) u nizu.

2. **Mehanizam samopozornosti** (*eng. Self-attention mechanism*): Mehanizam samopozornosti izračunava težinske faktore za svaku riječ u rečenici u odnosu na svaku drugu riječ, omogućujući modelu predviđanje riječi koje će vjerojatno slijediti u nizu. Ovaj proces se uči tijekom vremena dok se model trenira na velikoj količini podataka. *Self-attention* mehanizam omogućuje svakoj riječi da se paralelno usredotoči na svaku drugu riječ u nizu, važući njihovu važnost za trenutni *token*. Na taj način modeli strojnog učenja mogu "naučiti" gramatička pravila na temelju statističkih vjerojatnosti upotrebe riječi u jeziku.

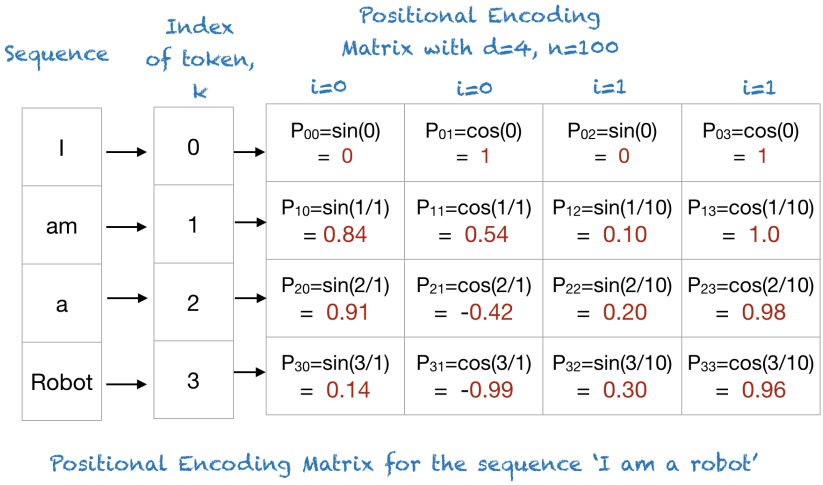
Transformer modeli obrađuju ulazne podatke kroz niz slojeva koji sadrže mehanizme samopozornosti *feedforward* mreže. Princip rada transformer modela može se podijeliti u nekoliko ključnih koraka. Uzmimo za primjer prijevod rečenice s engleskog na hrvatski jezik. U nastavku je detaljno opisan postupak rada transformer modela [36]:

1. **Pretvaranje ulaznih podataka u vektorske reprezentacije** (*eng. Input vector embeddings*): Prvi korak uključuje transformaciju ulazne rečenice u numeričke reprezentacije poznate kao *vector embeddings*. Ove reprezentacije utjelovljuju semantička značenja *tokena* u ulaznom nizu. Podsjetimo se, *tokeni* su osnovne jedinice obrade u jezičnim modelima i mogu predstavljati riječi, podriječi ili čak pojedinačne znakove. U kontekstu sekvenci riječi (obično rečenica), vektorske reprezentacije mogu se naučiti tijekom treniranja modela ili se mogu koristiti unaprijed trenirane reprezentacije (*eng. Pre-trained word embeddings*). Slika 16 prikazuje primjer numeričkih reprezentacija rečenice rastavljene na *tokene*.



Slika 16: Primjer kodiranja rečenice rastavljene na *tokene*

2. **Pozicijsko kodiranje** (*eng. Positional encoding*): obično se uvodi kao skup dodatnih vektora koji se dodaju ulaznim vektorima prije pohrane u sam transformer model. Ove vrijednosti imaju specifične obrasce koji kodiraju informaciju o poziciji riječi u rečenici, omogućujući modelu da prepozna redoslijed riječi i zadrži kontekstualne veze unutar rečenice [37]. Na slici 17 prikazan je primjer pozicijskog kodiranja za rečenicu "I am a robot". Najčešće se koriste sinusoidi različitih frekvencija.

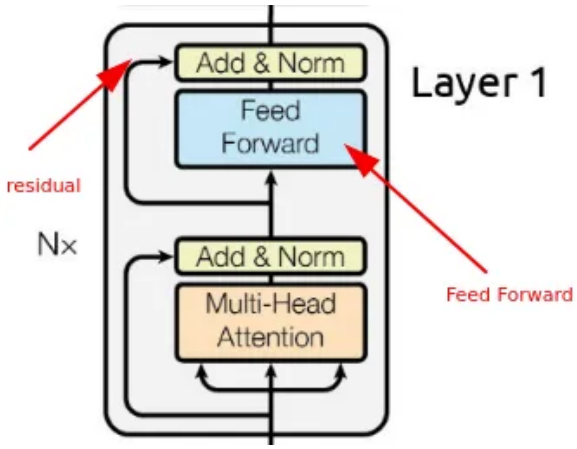


Slika 17: Primjer pozicijskog kodiranja za rečenicu "I am a robot".

3. **Višestruki mehanizam samopozornosti** (*eng. Multi-head attention*): *Self-attention* djeluje kroz više paralelnih mehanizama kako bi se naučili različiti odnosi između *tokena*. Svaki od tih mehanizama omogućava modelu da se usredotoči na različite karakteristike ulaznih podataka istovremeno, što poboljšava sposobnost modela da razumije složene obrasce i kontekste.

4. **Normalizacija slojeva i povezivanje reziduala** (*eng. Layer normalization and residual connections*): Transformer model koristi normalizaciju slojeva i povezivanje reziduala (dodatne putanje signala) kako bi stabilizirao i ubrzao proces treniranja. Normalizacija slojeva smanjuje unutarnju promjenjivost podataka tijekom treniranja, dok povezivanje reziduala omogućuje lakši protok informacija kroz mrežu, smanjujući problem gubitka gradijenta u dubokim mrežama.

5. **Neuronske mreže s prolazom unaprijed** (*eng. Feedforward neural networks*): Izlazni podaci *self-attention* sloja prosljeđuju se *feedforward NN*. Ove mreže primjenjuju nelinearne transformacije na vektorske reprezentacije, omogućujući modelu da uči složene obrasce i odnose u podacima. Slika 18 prikazuje *encoder* modul transformer modela.



Slika 18: Prikaz *encoder* modula transformera s *feedforward* operacijom, *self-attention* slojem i rezidualom kao alternativnom putanjom.

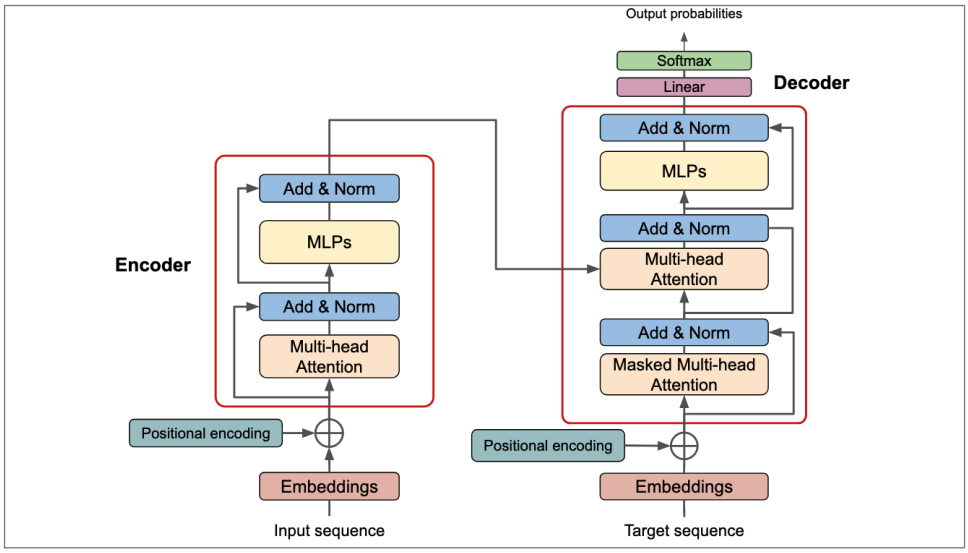
6. **Naslagani slojevi** (*eng. Stacked layers*): Cijela struktura transformer modela sastoji se od više međusobno naslaganih slojeva, poput: *self-attention* sloja, slojeva normalizacije, *feedforward* slojeva itd. Općenito u dubokom učenju, smisleno slaganje slojeva jedan na drugi (*eng. stacking*) je upravo ono što čini ove mreže snažnima zbog tendencije da se obuhvati što veći broj apstraktnih obrazaca u podacima.

7. **Output sloj** (*eng. Output layer*): U *sequence-to-sequence* zadacima, poput problema strojnog prevođenja, moguće je dodati dodatni *decoder* modul (kao što je objašnjeno u *seq2seq* poglavlju) koji generira izlaznu sekvencu, odnosno u ovom slučaju prevedenu rečenicu.

8. **Faza treniranja** (*eng. Training phase*): Transformer modeli se obično treniraju kombinacijom metoda nenadziranog (*eng. unsupervised learning*) i nadziranog učenja (*eng. supervised learning*). Prvobitno, modeli prolaze kroz fazu nenadziranog učenja, gdje je cilj minimizacija funkcije gubitka koja kvantificira razliku između predikcije modela i stvarnih rezultata na temelju velikog korpusa neoznačenih podataka. Treniranje transformer modela tipično uključuje korištenje *Adam optimizatora* ili SGD optimizatora. Nakon početne faze treniranja, obično slijedi *fine-tuning* koristeći nadzirano učenje kako bi se model prilagodio specifičnim zadacima koristeći označene podatke.

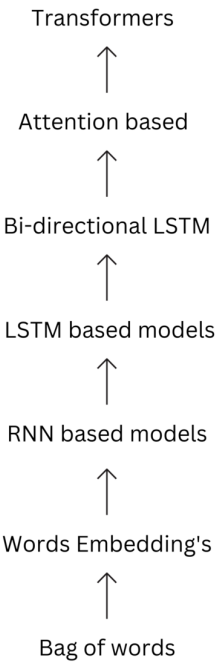
9. **Faza inferencije** (*eng. Inference phase*): Nakon treniranja, model se može koristiti za testiranje na novim podacima. Za vrijeme ove faze, ulazna sekvenca se prosljeđuje predtreniranom modelu, koji zatim generira predikcije.

   Arhitektura transformer modela može se vidjeti na Slici 19.



Slika 19: Arhitektura transformer modela

Slika 20 prikazuje tijek razvoja NLP-a: od spomenutih ranih simboličkih pristupa, preko statističkih modela 1970-ih poput *bag of words*, pa sve do suvremenih transformer arhitektura temeljenih na dubokom učenju.



Slika 20: Tijek razvoja NLP modela

3.4 Foundation modeli

Danas se *foundation* modeli gotovo poistovjećuju s velikim jezičnim modelima, koji su postali ključna osnova modernih AI aplikacija i privlače značajnu pažnju razvojne i istraživačke zajednice. *Foundation* modeli su transformirali cijelo polje umjetne inteligencije, kroz popularne alate poput ChatGPT-a. Važno je napomenuti da se termin *foundation* modela danas koristi i za modele koji nisu bazirani na transformer arhitekturi, poput popularnog YOLO (*You Only Look Once*) modela za detekciju objekata [38]. Međutim, u kontekstu ovog rada, izraz *foundation* modela odnosit će se na LLM-ove temeljene na modernoj transformer arhitekturi. U lipnju 2024. godine, za vrijeme pisanja ovog diplomskog rada, najsuvremeniji i najnapredniji veliki jezični modeli koriste *decoder-only* transformer arhitekturu. Ova se arhitektura razlikuje od prvobitne *encoder-decoder* arhitekture, koja je detaljno opisana u poglavlju 3.3.3. Rani primjeri *foundation* modela uključuju jezične modele poput Googleovog **BERT** modela te OpenAI-evih **GPT-1** i **GPT-2** modela.

Godina 2022. ostavila je neizbrisiv trag u svijetu umjetne inteligencije, posebno kroz razvoj i popularizaciju LLM *foundation* modela. Ovi modeli, zahvaljujući svojim naprednim arhitekturama i treniranju na ogromnim količinama podataka, omogućili su postizanje dosad neviđenih razina performansi u različitim AI aplikacijama. Izgradnja ovakvih modela iznimno je skupa, a treniranje najskupljih *flagship* modela može doseći cijenu i od nekoliko stotina milijuna američkih dolara zbog visokih troškova računalnih resursa i plaćanja velikih količina podataka.

Jedan od najvažnijih trenutaka bio je lansiranje **ChatGPT**-a na tržište, koji je u početku koristio model GPT-3. Ovaj alat ubrzo je postao izuzetno popularan za brojne primjene u obradi prirodnog jezika, uključujući odgovaranje na raznolika pitanja, kreativno pisanje, pomoć u obrazovanju, programiranje i kao osobni asistent u svakodnevnim zadacima.

3.4.1 Otvoreni modeli

Osim tehnološkog napretka, ovi događaji potaknuli su i važne diskusije o etici i regulaciji umjetne inteligencije. Pitanja privatnosti podataka, autorskih prava i potencijalnih zloupotreba sadržaja generiranih velikim jezičnim modelima postala su goruće teme [39] [40] [41]. Prema tome, važno je napomenuti i razvoj "open source" modela, koji nude alternativu velikim komercijalnim rješenjima. Treba biti oprezan s *open source* izrazom kada pričamo o velikim jezičnim modelima. Termin *open source* kod modela nezgrapan je budući da mnoge organizacije koje reklamiraju svoje modele kao *open source*, ustvari objavljuju javno samo određene dijelove modela. Neke organizacije objavljuju ukupan kôd, dok druge objavljuju podatke na kojima su modeli trenirani ili težinske faktore. Primjerice, OpenAI je objavio kôd za GPT-2, ali je zadržao težinske faktore za kasnije verzije, kao što je GPT-3, zbog zabrinutosti oko zloupotrebe tehnologije [42]. Ovakva selektivna objava informacija stvara zbrku oko stvarne transparentnosti modela i njihove dostupnosti široj zajednici [43] [44].

Izraz *Open* prikladniji je za modele koji se mogu preuzeti i pokretati lokalno na vlastitim

računalima, nego za modele koji su potpuno otvorenog kôda (kojih je vrlo malo). *Open* modeli omogućavaju korisnicima veću kontrolu i prilagodbu, čak i ako nisu potpuno transparentni u svim aspektima svog razvoja i treniranja.

U 2023. godini, značaj otvorenih *foundation* modela dodatno je porastao zahvaljujući izdanjima modela *LlaMA-2*, *Falcon* i *Mistral*. Ovi modeli su istaknuli prednosti otvorenog pristupa, omogućavajući široj zajednici priliku za prilagodbu, učenje i korištenje modela za razne primjene, te dodatno, potencijala za istraživanje, inovacije i transparentnost. Ovakav pristup modelima omogućava istraživačima detaljniju analizu njihovih efekata, potiče konkurenciju i inovacije, te poboljšava znanstvena istraživanja i reproducibilnost [45].

Međutim, otvorenost modela nosi i rizike, poput mogućnosti zloupotrebe u svrhu generiranja dezinformacija ili zabranjenog sadržaja. Ovi rizici potaknuli su ozbiljne rasprave o etičnosti i potrebi za regulacijom otvorenih *foundation* modela [46].

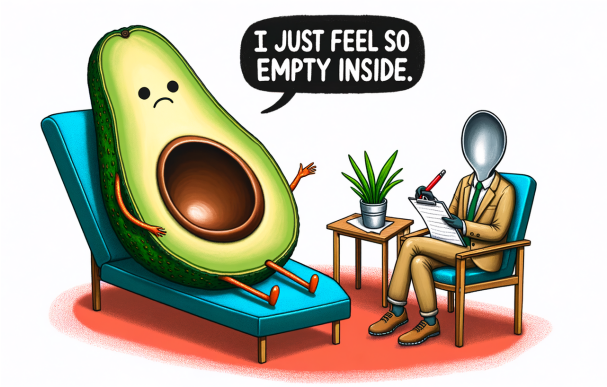
3.4.2 *text-to-image* modeli

Ubrzo nakon eksplozije velikih jezičnih modela, pojavili su se i *text-to-image* modeli, koji su proširili mogućnosti generativne umjetne inteligencije u novim - vizualnim smjerovima. Iako to možda nije odmah očito, ovi modeli također spadaju u kategoriju velikih jezičnih modela jer koriste slične arhitekture i tehnike za generiranje slika na temelju tekstualnih opisa. U osnovi, ovi modeli "zamjenjuju tekst pikselima", zadržavajući pritom temeljnu strukturu *sequence-to-sequence* modela.

Među najpoznatijim *text-to-image* modelima su *DALL·E*, *Midjourney* i *Stable Diffusion*. *DALL·E*, razvijen od strane tvrtke OpenAI, koristi transformere za generiranje slika iz tekstualnih opisa, demonstrirajući impresivne sposobnosti stvaranja vizualno koherentnih i estetski privlačnih slika. Midjourney je još jedan značajan model koji je brzo stekao popularnost zbog svojih visokokvalitetnih i kreativnih vizualnih izlaza. *Stable Diffusion*, s druge strane, poznat je po svojoj sposobnosti da generira slike visoke rezolucije uz zadržavanje detalja i složenosti zadanih opisa. Osim generiranja slika iz tekstualnih opisa, ovi modeli su postali multimodalni, što znači da omogućuju izmjene postojećih slika na temelju dodatnih tekstualnih uputa. Drugim riječima, korisnici mogu zatražiti izmjenu boje, dodavanje elemenata ili promjenu stila slike davanjem jednostavnih tekstualnih uputa.

Međutim, razvoj i primjena *text-to-image* modela nisu prošli bez kontroverzi i izazova. Jedan od glavnih problema je etička upotreba i regulacija ovih tehnologija. Naime, ovi modeli trenirani su na velikim skupovima podataka poput *WebImageText* koji sadrže slike s opisima. Mnogi autori i umjetnici izrazili su zabrinutost zbog neovlaštenog korištenja njihovih radova u procesu treniranja modela. Još jedan ozbiljan problem je potencijal za zloupotrebu ovih modela u svrhu generiranja *deepfake* slika i drugih oblika dezinformacija [47]. Generiranje realističnih, ali lažnih slika može se koristiti za manipulaciju javnim mnijenjem, širenje lažnih vijesti ili stvaranje kompromitirajućih sadržaja, što dodatno naglašava potrebu za regulacijom i etičkim smjernicama [48]. Konačno, dok *text-to-image* modeli predstavljaju značajan korak naprijed

u razvoju umjetne inteligencije, važno je nastaviti istraživati i razvijati načine kako osigurati njihovu etičku upotrebu, zaštitu intelektualnih prava i sprječavanje zloupotreba. Ove tehnologije nude ogromne potencijale, ali samo uz odgovoran pristup mogu uistinu doprinijeti društvu na pozitivan način. Slika 21 prikazuje primjer slike generirane DALL·E 3 modelom.



Slika 21: Fotografija generirana DALL·E 3 modelom uz *prompt*: "An illustration of an avocado sitting in a therapist's chair, saying 'I just feel so empty inside' with a pit-sized hole in its center. The therapist, a spoon, scribbles notes"

3.4.3 Razvoj AI zajednice

Razvoj umjetne inteligencije doživio je nevjerojatan rast, stvarajući dinamičnu zajednicu koja transformira tehnologiju i društvo. *Hugging Face* platforma postala je centralno mjesto za AI programere diljem svijeta, omogućavajući dijeljenje i preuzimanje otvorenih AI modela, kao i skupova podataka, potičući suradnju i inovacije. Značajne investicije dolaze iz privatnog i javnog sektora. Tehnološki giganti poput Googlea, Microsofta i Amazona ulažu milijarde dolara u AI tehnologije: Google razvija projekte kao što su *DeepMind*, Microsoft unapređuje *Azure AI* platformu i surađuje s OpenAI, dok Amazon integrira AI rješenja kroz *AWS* usluge. Apple je najavio suradnju s OpenAI 2024. godine kako bi integrirao AI alate u iOS uređaje, poboljšavajući korisničko iskustvo. Medijski prostor posvećen AI tehnologijama značajno je porastao, uz redovite rasprave o etici, utjecaju na tržište rada i budućnosti obrazovanja, AI postaje ključna tema u vijestima, stručnim časopisima i na tehnološkim konferencijama. Zakonodavna tijela širom svijeta počinju usvajati nove zakone i regulative kako bi osigurala odgovorno korištenje AI tehnologija. U SAD-u se radi na regulaciji privatnosti podataka, dok Europska unija razvija *AI Act* pritom jasno definirajući smjernice za razvoj i implementaciju AI tehnologija. Ove regulative pomažu u sprječavanju zloupotrebe te promicanju transparentnosti i povjerenja u nove tehnologije. Sve ove inicijative oblikuju budućnost umjetne inteligencije kao takve, omogućujući kontinuirani napredak uz adresiranje izazova koje donosi ova revolucionarna tehnologija. Tablica 1 prikazuje neke od najpopularnijih *foundation* modela za vrijeme pisanja ovog diplomskog rada (lipanj 2024).

Tvrtka	**Familija modela**	**Primjena**	**Popularne aplikacije**	**Pristup**
Open AI	GPT-3, GPT-4, GPT-4o	Multimodalni LLM, široke primjene.	ChatGPT, Microsoft Copilot, Duolingo	*chatbot* + API
Open AI	DALL·E 2, DALL·E 3	*text-to-image*	ChatGPT	*chatbot* + API
Open AI	Whisper	speech-recognition (ASR)	-	API
Open AI	Sora	*text-to-video*	-	-
Google	Gemini	Multimodalni LLM, široke primjene	Gemini *chatbot* i Google aplikacije	*chatbot* + API
Google	Gemma	Multimodalni LLM, široke primjene	-	lokalno
Meta	Llama2 i Llama3	Multimodalni LLM, široke primjene	Meta *chatbot* i aplikacije	lokalno
Anthropic	Claude 3 i Claude 3.5	Multimodalni LLM, široke primjene	Slack, Notion, Zoom	*chatbot* + API
Mistral AI	Mistral	Multimodalni LLM, široke primjene	-	*chatbot* + API + lokalno
Midjourney	Midjourney	*text-to-image*	-	Discord *chatbot*
Stability AI	Stable Diffusion	*text-to-image*	-	*chatbot* + API + lokalno
Cohere	Coral	Multimodalni LLM, široke primjene	HyperWrite, Jasper, Notion, LongShot	*chatbot* + API
Technology Innovation Institute	Falcon	Multimodalni LLM, široke primjene	-	lokalno
Suno AI	Suno	*text-to-music*	Suno aplikacija + Microsoft Copilot	web aplikacija + API

Tablica 1: Tablica popularnih *foundation* modela (lipanj 2024.)

3.4.4 Fine-tuning

Fine-tuning predstavlja proces prilagođavanja već unaprijed treniranog modela kako bi se optimizirao za određeni zadatak ili prilagodio novom skupu podataka. Dok su LLM-ovi poput GPT-4 trenirani na opsežnim skupovima podataka kako bi razumjeli široki spektar jezičnih struk-

tura i mogli generirati tekst u širokom rasponu konteksta, *fine-tuning* omogućava dodatnu obuku modela na specifičnim, često manjim skupovima podataka kako bi se poboljšale performanse na određenim zadacima. To može uključivati specifične aplikacije poput prepoznavanja entiteta, klasifikacije teksta, prevođenja, sažimanja informacija ili dijaloga u specifičnom području znanosti, medicine, prava i sl. Prednosti *fine-tuninga* uključuju značajna poboljšanja modela na specifičnim, domenskim zadacima, iskorištavanje postojeće infrastrukture te smanjenje potrebe za treniranjem modela "od nule".

Slučajevi gdje *fine-tuning* može poboljšati performanse modela uključuju:

1. Usavršavanje modela za rad s manje poznatim jezicima ili dijalektima.
2. Poboljšanje točnosti u praćenju složenih uputa i izvršavanju zadataka.
3. Primjena modela u specifičnim domenama, kao što su pravosuđe i medicina.
4. Svi zadaci koji zahtijevaju visoku preciznost i točnost u rezultatima.

Iako je *fine-tuning* izvrsna tehnika, nosi sa sobom određene izazove. Proces zahtijeva visoko tehničko znanje, uključujući razumijevanje modela, algoritama i infrastrukture potrebne za treniranje. Priprema podataka može biti rigorozna i zahtjevna, s naglaskom na pažljivo čišćenje, označavanje i formatiranje. Visoki troškovi također predstavljaju prepreku, jer procesi *fine-tuninga* i *model-deploymenta* zahtijevaju znatne računalne resurse. Korištenje rješenja za *fine-tuning*, poput onih koje nudi OpenAI, donosi nove izazove vezane uz sigurnost i privatnost podataka, uz dodatno povećanje financijskih troškova [49].

Prompt engineering: *Prompt engineering* predstavlja izvrsnu besplatnu alternativu *fine-tuningu*, ali također dolazi s određenim izazovima. Prompt engineering podrazumijeva pažljivo osmišljavanje i oblikovanje upita (*eng. prompt*) kako bi se postigli željeni odgovori LLM-ova bez potrebe za dodatnom obukom modela, odnosno za *fine-tuning*. Glavni izazov leži u obimu konteksta koji treba prenijeti. Veća količina konteksta svaki put povećava troškove ulaznog *tokena*, što povećava troškove korištenja modela. Svejedno, pravilno provedeni *fine-tuning* obično donosi bolje rezultate od *prompt engineeringa*.

Prompt engineering omogućava modelu da privremeno "uči" iz zadanih *promptova*. Ova značajka, poznata kao *in-context* učenje, postaje sve izraženija kako se model povećava. Za razliku od *fine-tuninga*, učenje iz konteksta predstavlja samo privremeno znanje koje model koristi, ali ne zadržava dugoročno [50].

Popularne *prompt engineering* tehnike uključuju:

1. **Davanje jasnih uputa**: Jezični modeli ne mogu čitati misli, stoga je važno pružiti precizne i konkretne upute. Ako su dobiveni rezultati predugi, može se zatražiti kraći odgovor. Kada odgovori nisu dovoljno složeni, korisno je zatražiti odgovore na stručnoj ili ekspertnoj razini. Također, format odgovora može se prilagoditi; umjesto razgovornog jezika,

može se preferirati književni ili akademski stil. Smanjenjem potrebe za pretpostavkama o korisničkim očekivanjima, povećava se vjerojatnost dobivanja željenih odgovora. Primjerice, umjesto da zatražimo "Napiši nešto o povijesti umjetnosti," možemo reći: "Napiši sažetak povijesti renesansne umjetnosti koristeći akademski stil i ne duži od 500 riječi."

2. **Zatražiti model da usvoji određenu personu**: Često je korisno zamoliti jezični model da preuzme specifičnu personu ili ulogu. LLM može preuzeti identitet određene osobe, izmišljenog lika, stručnjaka u nekom području, učitelja i slično. Komunikacijski ton može biti precizno definiran (formalan, neformalan, prijateljski, stručan itd.). Uz to, može se pružiti i dodatni kontekst, poput "ponašaj se kao stručnjak na konferenciji" ili "prijatelj koji daje savjete". U kontekstu obrazovanja, možemo pak upute definirati i na sljedeće načine: "Zamisli da si profesor povijesti koji objašnjava uzroke Drugog svjetskog rata" ili "Preuzmi ulogu profesora informatike koji pomaže studentu riješiti programerski problem".

3. **Korištenje oznaka za naglašavanje**: Oznake ili delimitere, poput navodnih znakova, XML/HTML oznaka, podnaslova, zagrada i sličnog, mogu pomoći jezičnom modelu u razumijevanju strukture i konteksta u *promptu*. Na ovaj način, mogu se označiti blokovi kôda ili specifični dijelovi teksta koje treba posebno tretirati, na primjer, kao entitete. Markdown oznake su također popularan način strukturiranja teksta, posebno u tehničkoj dokumentaciji. Oznake olakšavaju čitljivost i naglašavanje ključnih uputa, omogućavajući modelu da bolje interpretira i odgovori na zadani *prompt*.

4. **"Korak po korak" instrukcije**: Razlaganje zadataka na manje korake može biti vrlo korisno jer omogućuje temeljitije i jasnije razumijevanje problema te učinkovitije rješavanje. Kada dajemo instrukcije jezičnom modelu, možemo koristiti pristup "korak po korak" kako bi model bolje razumio i izvršio zadatak. Na sličan način kao što ljudi razmišljaju o problemu: prvo identificiraju ključne komponente, zatim ih analiziraju pojedinačno, te na kraju integriraju svoja saznanja kako bi došli do cjelovitog rješenja. Primjerice, ako pitamo osobu da napamet pomnoži 17 i 28, vjerojatno neće odmah znati rezultat. Međutim, promišljenim pristupom i primjenom određenih strategija, osoba može doći do točnog odgovora.

5. **Davanje primjera i referentnih tekstova**: Veliki jezični modeli mogu vrlo pouzdano iznositi lažne odgovore i činjenice, osobito kada ih se pita o konkretnim temama ili citatima. Na isti način kao što skripta, ili papir s formulama iz matematike može pomoći studentu na ispitu, pružanje referentnog teksta ovim modelima može pomoći u odgovaranju s manje "izmišljanja".

Transformeri imaju izvrsne sposobnosti pamćenja i kontekstualnog razumijevanja. *Self-Attention* mehanizam omogućava modelima da se fokusiraju na relevantne dijelove ulaznog

teksta, čime se značajno poboljšava njihova sposobnost razumijevanja i generiranja koherentnih odgovora. Međutim, kada se suoče s velikim kontekstom, mnogim pravilima i uputama, ili informacijama iz različitih izvora, ovi modeli ponovno mogu imati poteškoća (pogotovo oni manji). Upravo ta složenost i količina informacija može uzrokovati probleme s točnošću i dosljednošću odgovora koje model pruža.

Kako bi se pokušali riješiti ovi problemi, razvijaju se napredne tehnike poput **Retrieval-Augmented Generation** (RAG). RAG tehnike spajaju generativne sposobnosti velikih jezičnih modela s mogućnošću pretraživanja relevantnih informacija iz vanjskih izvora. Ove tehnike omogućuju modelima da identificiraju i preuzmu informacije iz specifičnih dokumenata ili baza podataka prije samog generiranja odgovora, čime se značajno smanjuje rizik od dezinformacija i generiranja izmišljenih podataka.

U četvrtom odjeljku - *Metode oblikovanja konteksta*, detaljno se analiziraju RAG tehnike.

3.4.5 RLHF

Reinforcement Learning from Human Feedback (RLHF) je metoda usklađivanja AI sustava prema preferencijama korisnika. Ovaj pristup ima za cilj uskladiti ponašanje umjetne inteligencije s ljudskim vrijednostima, preferencijama i povratnim informacijama. RLHF dio je *reinforcement learning* područja koje se bavi učenjem iz povratnih informacija koje dolaze direktno od ljudi, čime se model trenira da optimizira svoje akcije kako bi postigao što bolje rezultate prema zadanom cilju.

U tradicionalnom *reinforcement learning*-u, agent uči iz interakcije s okolinom putem **nagrada** i **kazni** koje dobiva na temelju svojih akcija. Međutim, teško je definirati funkciju koja eksplicitno aproksimira preferencije čovjeka. Kod RLHF-a, ljudske povratne informacije služe kao ključan izvor učenja i njima se direktno trenira model nagrade (*eng. reward model*). Ovaj model koristi se za poboljšanje koherencije agenta putem algoritama optimizacije kao što je *proximal policy optimization*. Iako je metoda učinkovita, izazovi se javljaju u prikupljanju kvalitetnih podataka kroz ljudske preferencije, koje mogu biti pristrane [51].

Optimizacija modela na temelju povratnih informacija korisnika (RLHF) poželjna je kada je zadatak teško specificirati, ali ga je lako procijeniti. U zadacima poput generiranja prirodnog jezika, teško je unaprijed odrediti sve kriterije za kvalitetu odgovora, ali je relativno jednostavno ocijeniti koliko je odgovor koristan ili relevantan nakon što je generiran [52].

Primjera radi, ako pitamo *chatbot* kakvo je vrijeme vani, odgovor bi mogao biti:

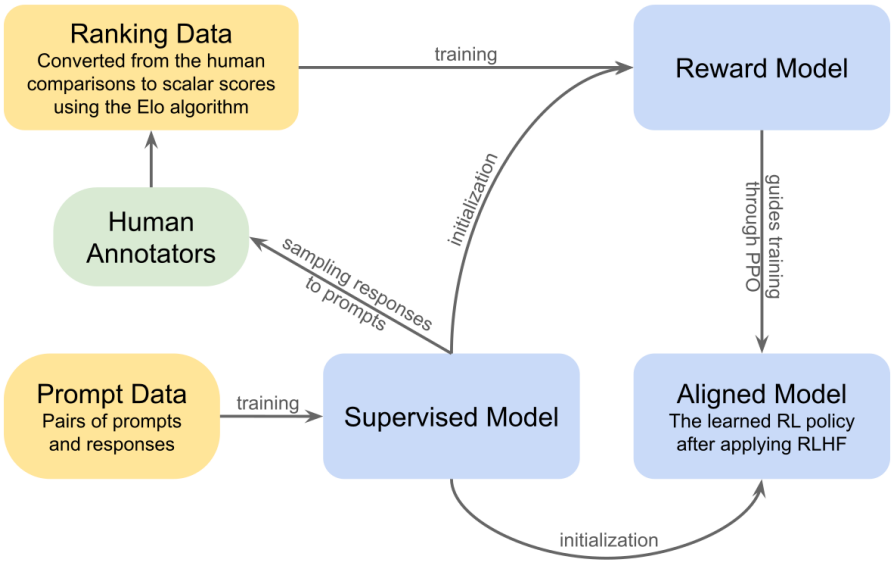
"Trenutno je vani vedro i suho, s temperaturom od 25 stupnjeva."

ili

"Trenutno je vani vedro i suho, uz temperaturu od ugodnih 25 stupnjeva. Mjestimice možemo osjetiti blagi povjetarac, što dodatno doprinosi ugodnom osjećaju."

Iako oba odgovora pružaju iste osnovne informacije, drugi odgovor zvuči prirodnije i pruža više konteksta, što može poboljšati korisničko iskustvo. Raniji napori suočavali su se s izazovima

poput generalizacije i rijetkih ili nejasnih funkcija nagrađivanja. Ipak, RLHF je postao popularna i široko primjenjivana metoda, koja se koristi u raznim domenama NLP-a, uključujući razvoj konverzacijskih agenata, sažimanje teksta i prevođenje. Slika 22 prikazuje *high-level* prikaz rada RLHF metode.



Slika 22: *High-level* prikaz rada RLHF metode

Iako povećanje veličine jezičnih modela ne čini iste nužno boljima u praćenju korisničkih namjera, OpenAI je pokazao obećavajući napredak usklađivanju modela s ljudskim namjerama kroz RLHF tehnike u njihovom radu: "Training language models to follow instructions with human feedback" iz 2022. godine [53]. Izradili su skup podataka koji se sastoji od *promptova* napisanih od strane stručnjaka i korisnika OpenAI API-ja, koji sadrži primjere željenog ponašanja modela. Taj skup podataka koristio se početno nadzirano učenje GPT-3 modela. Nakon toga, sastavili su bazu podataka rangiranih *outputa* modela , koja se koristi za daljnje usavršavanje modela kroz RLHF tehniku. Rezultantni model nazvali su InstructGPT. Ručnom evaluacijom, *outputi* InstructGPT modela, koji se sastoji od 1.3 milijarde parametara, preferirani su u donosu na *outpute* modela GPT-3 s 175 milijardi parametara.

OpenAI je popularizirao ovu tehnologiju putem ChatGPT-a, omogućujući modelima da generiraju relevantnije odgovore i odbacuju neprikladne upite. Povratne informacije korisnika kontinuirano poboljšavaju modele, prilagođavajući ih potrebama korisnika.

4 Metode oblikovanja konteksta

Jezični modeli su pokazali da posjeduju značajnu količinu dubinskog znanja stečenog iz podataka, ostvarujući to bez potrebe za pristupom vanjskoj memoriji, djelujući kao **parametrizirana implicitna baza znanja** [54].

Premda je ovaj napredak značajan, takvi modeli pokazuju određene nedostatke: ne mogu lako proširiti ili revidirati svoje znanje, ne mogu jednostavno pružiti uvid u svoje predikcije te mogu proizvoditi halucinacije [55]. Hibridni modeli koji kombiniraju parametarsku memoriju s neparametarskom (tj. *retrieval-based*) mogu riješiti neke od tih problema jer se znanje može izravno revidirati i proširiti, a pristupljeno znanje (*eng. retrieved knowledge*) može se pregledati i interpretirati [56].

4.1 Uvod u RAG

Retrieval-Augmented Generation (RAG) predstavlja napredni skup tehnika u području obrade prirodnog jezika (NLP) koji kombinira dva pristupa: pretraživanje informacija (*eng. retrieval*) i generiranje teksta (*eng. generation*). Metoda je prvi put predstavljena u radu iz 2020. godine pod naslovom "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks" autora Patricka Lewisa i suradnika [57]. U radu su predstavljena dva modela za generiranje teksta koja kombiniraju parametarsku i neparametarsku memoriju (tj. pamćenje). Ovi modeli, poznati kao RAG modeli, popularizirali su termin RAG koji se danas koristi kao *umbrella term* za skup tehnika namijenjenih optimizaciji velikih jezičnih modela putem referenciranja vanjskih izvora podataka [36]. Rana faza RAG-a obilježena je redefiniranjem *pre-training* tehnika pretreniranih modela.

Parametarska memorija odnosi se na dio modela u kojem se tijekom treniranja pohranjuje "naučeno" znanje unutar samih parametara modela. Kada dođe do promjena u stvarnom svijetu, odnosno kada činjenice pohranjene u parametarskoj memoriji postanu zastarjele ili se pojave nove informacije, obično je potrebno ponovno trenirati cijeli model kako bi se ažuriralo znanje.

Neparametarska memorija predstavlja vanjski izvor informacija. Kada se model susretne s pitanjem ili zadatkom, koristi pretraživač (*retriever*) za dohvaćanje relevantnih informacija iz ovog vanjskog izvora.

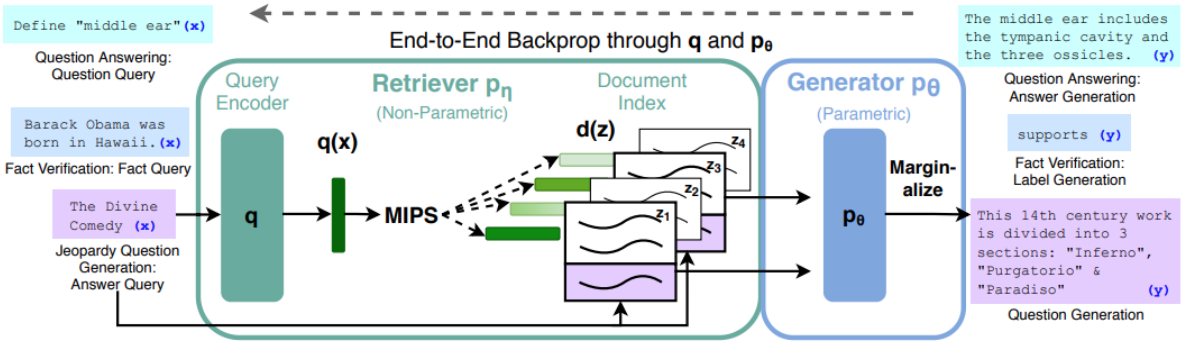
Ključni elementi RAG modela predstavljenih u ranom radu autora Lewisa su:

1. **Retriever**: Koristi *Dense Passage Retriever* (DPR) pretraživač koji se temelji na BERT arhitekturi. BERT (*Bidirectional Encoder Representations from Transformers*) je pretrenirani jezični model koji koristi transformere za stvaranje kontekstualnih reprezentacija riječi, omogućavajući modelu razumijevanje značenja riječi u njihovom kontekstu. Upit i dokumenti se pretvaraju u vektorske reprezentacije, a najrelevantniji dokumenti se dohvaćaju pomoću algoritma *Maximum Inner Product Search* (MIPS). MIPS omogućava brzo pronalaženje dokumenata čiji vektori imaju najveći unutarnji umnožak s vektorom

upita. Parametri ovog *retrievera* unaprijed su trenirani na zadacima kao što su TriviaQA i Natural Questions.

2. **Generator**: Ova komponenta generira konačni odgovor koristeći BART jezični model - pretrenirani *seq2seq* model temeljen na transformer arhitekturi. BART je treniran na zadacima uklanjanja šuma (*eng. denoising*) i generiranja teksta, što mu omogućava izvršavanje različitih generativnih zadataka. Generator kreira izlazni tekst kombinirajući ulazni upit i dohvaćene dokumente. Pri tome može koristiti različite dokumente za generiranje svakog *tokena* u odgovoru (*RAG-Token*) ili isti dokument za cijelu sekvencu (*RAG-Sequence*).

U ovim RAG modelima, *retriever* komponenta dohvaća relevantne dokumente s Wikipedije. Slika 23 prikazuje opisani RAG pristup.



Slika 23: Ilustracija opisanog RAG pristupa - kombiniranje pretreniranog *retrievera* s pretreniranim *seq2seq generator* modelom.

Kroz ovaj pristup, autori su smanjili sklonost modela halucinacijama i postigli veću sposobnost generiranja specifičnih odgovora. Prednost RAG modela leži u njihovoj sposobnosti da, umjesto ponovnog treniranja cijelog modela pri promjeni određenih informacija iz općeg znanja, jednostavno ažuriraju ili zamijene sadržaj neparametarske memorije. Primjerice, dodavanjem novog članka na Wikipediju ili ažuriranjem postojećeg, model će automatski koristiti te nove informacije pri generiranju odgovora [57].

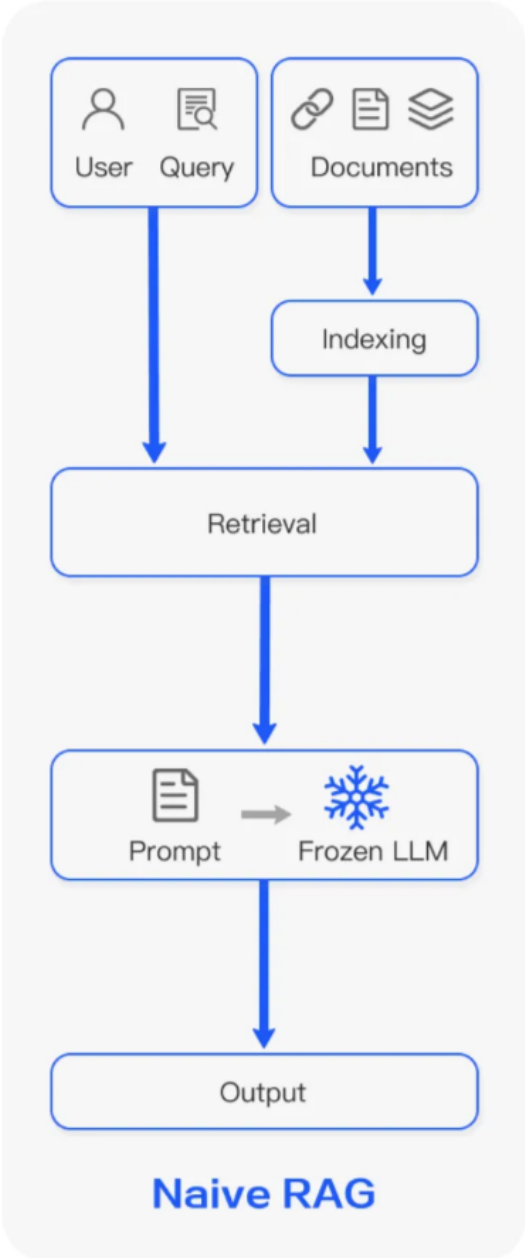
4.2 Osnovni RAG

Izlaskom ChatGPT-a krajem 2022. godine, došlo je do ključne prekretnice u razvoju RAG-a, donoseći sa sobom značajno parametarsko znanje i dotad neviđene sposobnosti učenja u kontekstu. Učenje u kontekstu (*eng. in-context learning - ICL*) odnosi se na sposobnost velikih jezičnih modela da razumiju i odgovaraju na nova pitanja koristeći nekoliko primjera/uputa danih u *promptu*, bez potrebe za eksplicitnim učenjem novog znanja. Istraživanja u RAG domeni usmjeravaju se na pružanje boljih informacija velikim jezičnim modelima tijekom faze

izvođenja (inferencije), u usporedbi s prethodnim generacijama modela, koje su se oslanjale isključivo na parametarsku memoriju.

Osnovni (*eng.* *Naive*) RAG predstavlja jedan od ranih metodoloških pristupa koji je brzo stekao popularnost nakon širokog prihvaćanja naprednih jezičnih modela. Slika 24 prikazuje osnovni RAG pristup, poznat i kao *Retrieve-Read framework*. Ovaj pristup sastoji se od tri ključna koraka:

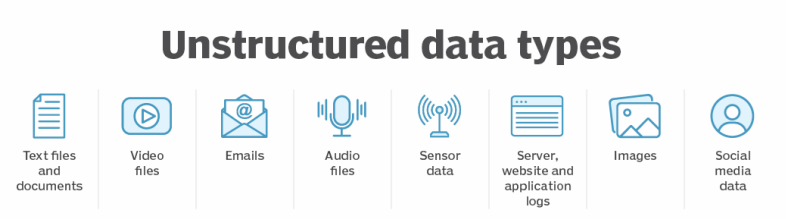
1. **Indeksiranje** (*eng. indexing*)
2. **Dohvaćanje** (*eng. retrieval*)
3. **Generiranje** (*eng. generation*)



Slika 24: Osnovni (*eng. naive*) RAG pristup

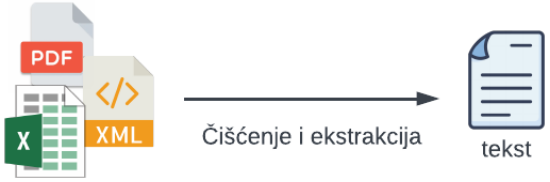
Indeksiranje (*eng. Indexing*) proces je koji započinje čišćenjem i ekstrakcijom sirovih nestrukturiranih podataka iz različitih formata poput PDF-a, tekstualnih datoteka, Word dokumenata, Excel tablica, HTML datoteka i drugih tekstualnih formata. Ti podaci se zatim pretvaraju u ujednačeni format običnog teksta. Kako bi se prilagodili kontekstualnim ograničenjima velikih jezičnih modela, tekst se segmentira u manje, lakše obradive dijelove. Nakon segmentacije, ovi dijelovi se kodiraju u vektorske reprezentacije pomoću modela ugradnje (*eng. embedding models*) i pohranjuju u vektorsku bazu podataka. Ovaj korak je ključan za omogućavanje učinkovitog pretraživanja u kasnijim fazama dohvaćanja informacija.

1. **Čišćenje i ekstrakcija podataka**: Cijeli proces započinje čišćenjem i ekstrakcijom sirovih podataka iz različitih formata poput PDF-a, HTML-a, Worda i Markdowna. Ovo uključuje parsiranje dokumenata kako bi se uklonili nepotrebni formati, metapodaci i drugi nebitni elementi. Primjerice, ako imamo PDF dokument koji sadrži znanstveni rad, ovaj korak uključuje izdvajanje teksta iz dokumenta, uklanjajući slike, fusnote i složeno formatiranje. Kroz ovaj proces osigurava se da se samo relevantni tekstualni sadržaj koristi za daljnju obradu. Slika 25 prikazuje oblike nestrukturiranih podataka, poput dokumenata, videozapisa, slika, audiozapisa itd.



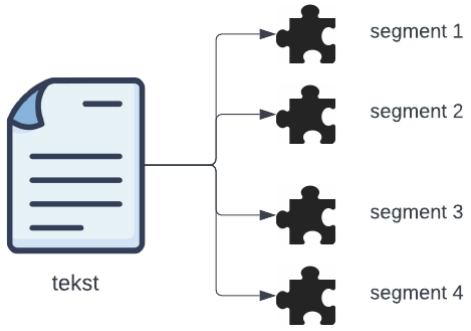
Slika 25: Oblici nestrukturiranih podataka

2. **Pretvaranje u unificirani format**: Jednom kad se tekst izdvoji, pretvara se u unificirani format podataka (*eng. unified data format*), najčešće je to običan tekst. Ovaj korak osigurava da su svi podaci, bez obzira na originalni format, standardizirani, što olakšava daljnju obradu. Kao primjer možemo uzeti pretvorbu Word dokumenta s različitim fontovima, zaglavljima i stilovima u običnu tekstualnu datoteku gdje je sačuvan samo esencijalni tekstualni sadržaj. Slika 26 prikazuje opisane korake 1 i 2.



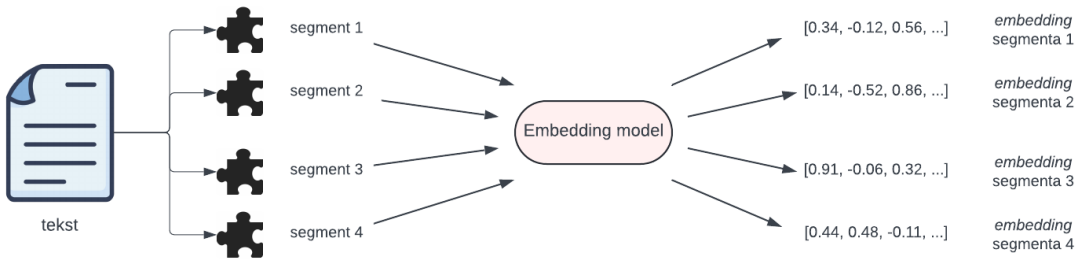
Slika 26: Pretvaranje nestrukturiranih podataka u unificirani format običnog teksta nakon odrađenog čišćenja i ekstrakcije

3. **Segmentacija teksta**: Kako bi se učinkovito prilagodili kontekstualnim ograničenjima velikih jezičnih modela, običan tekst se segmentira u manje, "probavljive" dijelove. Ovo je važno jer jezični modeli mogu kvalitetno obraditi samo ograničenu količinu teksta odjednom. Primjer, ako običan tekst sadrži cijelu knjigu, može se podijeliti na odlomke ili sekcije od nekoliko stotina riječi.



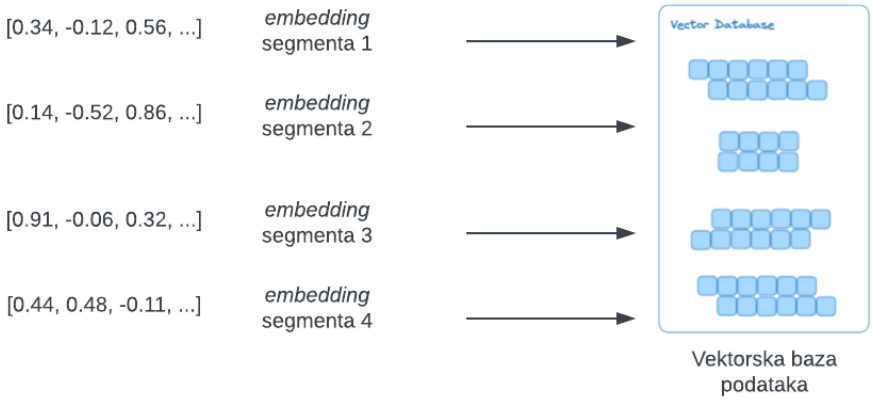
Slika 27: Prikaz segmentacije običnog teksta na manje "probavljive" segmente

4. **Kodiranje u vektorske reprezentacije**: Svaki segmentirani dio teksta zatim se kodira u vektorske reprezentacije pomoću modela kodiranja/ugradnje (*eng. embedding model*). Ovi modeli transformiraju ulazne podatke (segmente) u guste vektore realnih brojeva unutar kontinuiranog vektorskog prostora (Slika 28). Vektori predstavljaju semantičke odnose i značenja riječi i teksta, npr. rečenica "Strojno učenje je podskup umjetne inteligencije" pretvara se u vektor visokodimenzionalnog prostora koji se može prikazati kao jednodimenzionalni niz decimalnih vrijednosti: $[0.34, -0.12, 0.56, \ldots]$.



Slika 28: Kodiranje segmenata u vektorske reprezentacije pomoću *embedding* modela

5. **Pohrana u vektorsku bazu podataka**: Dobiveni vektori pohranjuju se u vektorsku bazu podataka (Slika 29). Vektorske baze podataka su specijalizirane baze dizajnirane za pohranu i pretraživanje vektorskih reprezentacija. Ove baze često imaju ugrađene različite algoritme za pretraživanje po sličnosti, što omogućuje brzo dohvaćanje relevantnih dijelova teksta na temelju njihovog semantičkog sadržaja.

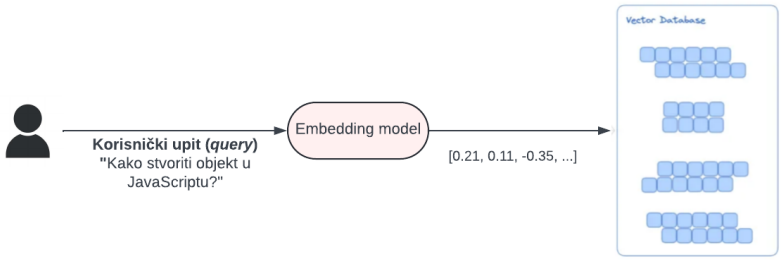


Slika 29: Pohrana dobivenih ugradbi (*eng. embeddings*) u vektorsku bazu podataka

Dohvaćanje (*eng. Retrieval*) Nakon primanja korisničkog upita, RAG koristi isti model kodiranja koji je korišten tijekom faze indeksiranja za pretvaranje upita u vektorsku reprezentaciju. Sustav zatim izračunava sličnost između vektora upita i vektora segmenata unutar indeksiranog korpusa. Na temelju te sličnosti, sustav prioritizira i dohvaća *top-K* dijelove koji pokazuju najveću sličnost s upitom. Ovi dijelovi se potom koriste kao prošireni kontekst u *promptu* za generaciju.

1. **Primanje korisničkog upita**: *Retriever* modul najprije prima upit od korisnika, obično u obliku prirodnog jezika. Primjerice, korisnik može pitati: "Kako stvoriti objekt u JavaScriptu?"

2. **Kodiranje upita**: *Retriever* modul koristi isti *embedding* model koji je korišten tijekom faze indeksiranja, no ovaj put za pretvaranje korisničkog upita u vektorsku reprezentaciju. To osigurava da su upit i indeksirani dokumenti u istom vektorskom prostoru (*eng. vector space*), što omogućuje precizno izračunavanje sličnosti. Slika 30 prikazuje postupak kodiranja korisničkog upita prije sljedeće faze izračunavanja sličnosti.



Slika 30: Kodiranje korisničkog upita

3. **Izračunavanje sličnosti**: Sljedeći korak je izračunavanje sličnosti između vektora upita i vektora segmenata unutar indeksiranog korpusa. Ova operacija obično se izvodi unutar vektorske baze podataka, najčešće koristeći *cosine similarity* metriku. Sličnost vektora

dana je u intervalu [0, 1], gdje veća vrijednost označava veću sličnost između vektora upita i vektora dokumenta. Tablica 2 prikazuje 5 segmenata pohranjenih unutar vektorske baze s izračunatim sličnostima za postavljeno pitanje o stvaranju objekta u JavaScriptu.

Redni broj	**Dokument/Segment**	**Sličnost**
1	U JavaScriptu možete stvoriti objekt koristeći literal objekta: `const obj = { key: 'value' };`. Ovaj način je jednostavan i često se koristi za definiranje objekata s ključ-vrijednost parovima.	0.96
2	ES6 uvodi klasu za stvaranje objekata: `class Car { constructor(model) { this.model = model; } } const myCar = new Car('Toyota');`. Ovaj pristup omogućava korištenje metoda konstruktora za definiranje objekta.	0.91
3	Konstruktor funkcije se također može koristiti za stvaranje objekta: `function Person(name) { this.name = name; } const me = new Person('John');`. Ovaj način je bio popularan prije uvođenja klasa u ES6.	0.81
4	Možete koristiti metodu `Object.create()` za stvaranje novog objekta s određenim prototipom: `const newObj = Object.create(proto);`. Ovaj pristup daje veću kontrolu nad nasljeđivanjem prototipa.	0.79
5	JavaScript omogućava korištenje JSON formata za stvaranje i prijenos objekata: `const obj = JSON.parse('"key": "value"');`. Ovaj način je koristan za rad s podacima u JSON formatu.	0.46

Tablica 2: Segmenti s izračunatim semantičkim sličnostima za postavljeno korisničko pitanje o stvaranju objekta u JavaScriptu.

Kosinusna sličnost (*eng. Cosine similarity*) predstavlja mjeru sličnosti između dva vektora, koja se izračunava kao kosinus kuta između njih. Vektori se obično koriste za reprezentaciju tekstualnih podataka. Formula za izračunavanje kosinusne sličnosti je sljedeća:

$$\textit{cosine-similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}|| \cdot ||\mathbf{B}||}$$

U ovoj formuli, $\mathbf{A}$ i $\mathbf{B}$ predstavljaju vektore upita i segmenta, $\mathbf{A} \cdot \mathbf{B}$ je njihov skalarni produkt, a $||\mathbf{A}||$ i $||\mathbf{B}||$ su norme (veličine) vektora $\mathbf{A}$ i $\mathbf{B}$.

Cosine similarity daje vrijednosti u intervalu [0, 1], gdje veća vrijednost označava veću sličnost između upita i dokumenta.

Druge metrike za mjerenje sličnosti, koje se češće koriste kod drugih problema ili ulaznih podataka, uključuju Euklidsku udaljenost, Manhattan udaljenost i Jaccardov indeks.

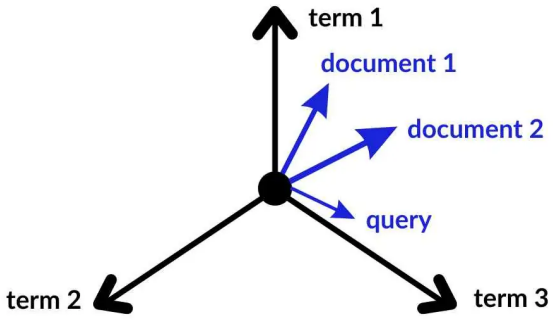
4. **Dohvaćanje top-K segmenata**: *Retriever* prioritizira i dohvaća *top-K* segmenata koji pokazuju najveću sličnost s korisničkim upitom. Ti segmenti se smatraju najrelevantnijima i odabiru se kako bi pružili prošireni kontekst za generiranje odgovora. U ovom kontekstu, K predstavlja broj segmenata s najvećom sličnosti koji će biti korišteni za proširivanje konteksta.

   Na primjeru korisničkog upita "Kako stvoriti objekt u JavaScriptu?" izračunate su sličnosti za segmente unutar vektorske baze podataka (prikazano u Tablici 2). Za odabrani $K = 2$, segmenti s najvećom sličnosti su:

   Segment 1 s izračunatom sličnosti od 0.96: "U JavaScriptu možete stvoriti objekt koristeći literal objekta: `const obj = { key: 'value' };` Ovaj način je jednostavan i često se koristi za definiranje objekata s ključ-vrijednost parovima."

   Segment 2 s izračunatom sličnosti od 0.91: "ES6 uvodi klasu za stvaranje objekata: `class Car { constructor(model) { this.model = model; } } const myCar = new Car('Toyota');` Ovaj pristup omogućava korištenje metoda konstruktora za definiranje objekta."

   Odabir segmenata 1 i 2 kao najrelevantnijih odgovora na korisnički upit o stvaranju objekta u JavaScriptu demonstrira njihovu visoku semantičku usklađenost s upitom. Zbog toga će ovi segmenti biti korišteni u daljnjem procesu generiranja konačnog odgovora. Slika 31 ilustrira pretraživanje vektorskog prostora za navedeni upit.



Slika 31: Prikaz pretraživanja sličnosti između vektora dokumenata/segmenata i vektora upita u višedimenzionalnom prostoru

Generiranje (*eng.* *Generation*) Nakon što je upit postavljen i relevantni segmenti su dohvaćeni, slijedi faza generiranja konačnog odgovora. U ovoj fazi, upit korisnika i dohvaćeni segmenti se kombiniraju kako bi se kreirao koherentan *prompt* za LLM. Proces generiranja odgovora može se podijeliti u dva koraka:

1. **Priprema *prompta***: Dohvaćeni segmenti i korisnički upit kombiniraju se u jedinstveni *prompt* koji se prosljeđuje LLM-u. Primjerice, za upit "Kako stvoriti objekt u JavaScriptu?" i dohvaćene segmente iz prethodne faze, sastavljeni *prompt* može izgledati ovako:

"Korisnik je postavio pitanje 'Kako stvoriti objekt u JavaScriptu?'. U nastavku su navedene relevantne informacije koje ćeš upotrijebiti za generiranje odgovora:

1. U JavaScriptu možete stvoriti objekt koristeći literal objekta: `const obj = { key: 'value' };`. Ovaj način je jednostavan i često se koristi za definiranje objekata s ključ-vrijednost parovima.

2. ES6 uvodi klasu za stvaranje objekata: `class Car { constructor(model) { this.model = model; } } const myCar = new Car('Toyota');`. Ovaj pristup omogućava korištenje metoda konstruktora za definiranje objekta.

Na temelju ovih informacija, odgovori na postavljeno pitanje korisnika."

2. **Generiranje konačnog odgovora**: Na temelju sastavljenog *prompta*, LLM će sada generirati precizniji odgovor, koji može izgledati ovako:

"Postoji nekoliko načina kako možete stvoriti objekt u JavaScriptu. Jedan od najjednostavnijih načina je korištenje literala objekta:

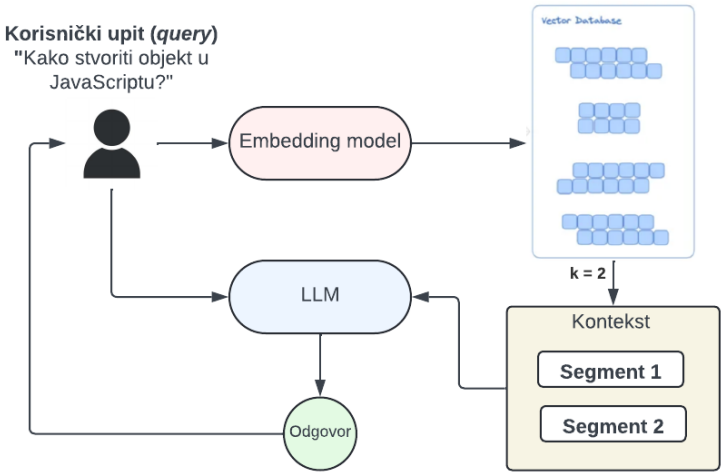
```
const obj = { key:  'value' };
```

Ovaj način je jednostavan i često se koristi za definiranje objekata s ključ-vrijednost parovima. Također, možete koristiti sintaksu klase uvedenu u ES6:

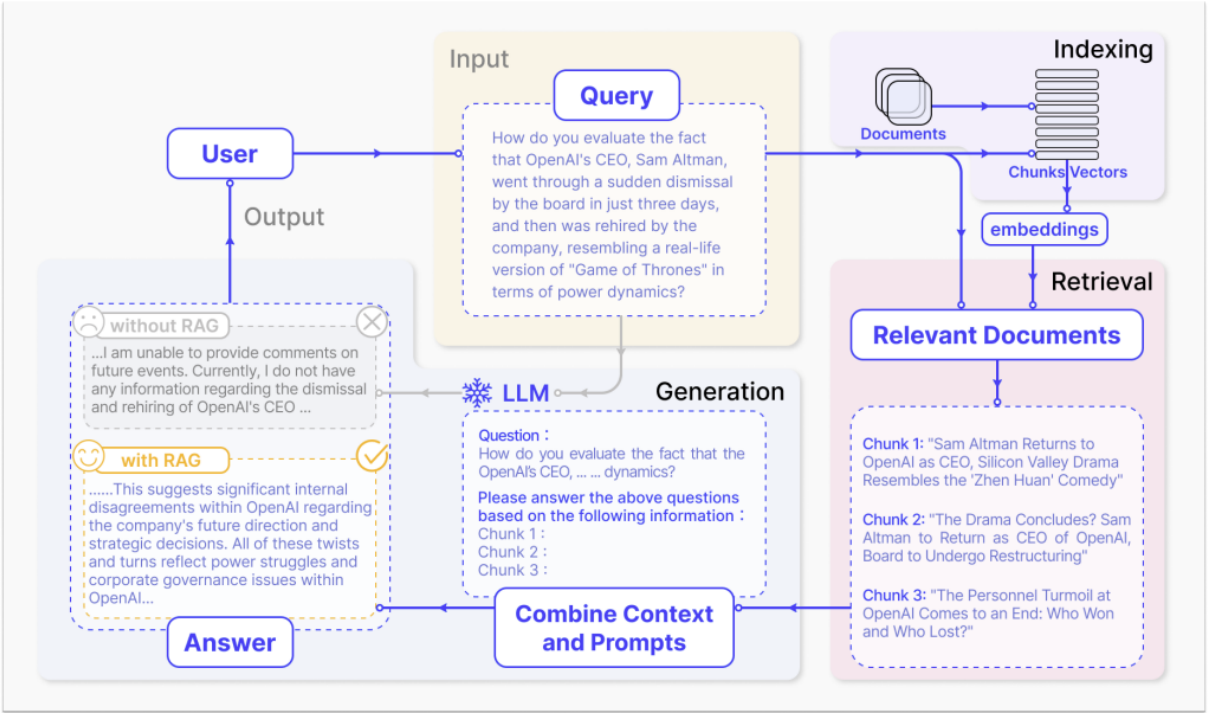
```
class Car { constructor(model) { this.model = model; } } const myCar = new Car('Toyota');
```

Ovaj pristup omogućava korištenje konstruktor metode za definiranje objekata."

Slika 32 prikazuje cijeli postupak generiranja konačnog odgovora za dani upit o JavaScript objektima, dok Slika 33 prikazuje generički RAG proces kod problema odgovaranja na pitanja.



Slika 32: Ilustracija opisanih procesa dohvaćanja relevantnih segmenata (1 i 2) na temelju semantičke sličnosti te generiranje odgovora koristeći novi kontekst i originalni korisnički upit.



Slika 33: Ilustracija osnovnog (*env. naive*) RAG procesa na primjeru odgovaranja na pitanja. Prikazana su 3 koraka: 1) **Indeksiranje**: Učitavanje datoteka, njihova podjela na segmente, kodiranje u vektore i pohrana u vektorsku bazu podataka. 2) **Dohvaćanje**: dohvaćanje top-k segmenata koji su najrelevantniji s postavljenim pitanjem obzirom na semantičku sličnost. 3) **Generacija**: kombiniranje korisničkog upita i dobivenih segmenata kao *prompt* LLM-u koji generira konačni odgovor.

4.2.1 Nedostaci osnovne RAG tehnike

Osnovna RAG tehnika ima nekoliko nedostataka [58] koji se manifestiraju kroz različite faze procesa:

1. **Izazovi u fazi pretrage**: Faza pretrage često se suočava s problemima preciznosti i pronalaženja relevantnih informacija. Ovo može dovesti do odabira nepovezanih ili nebitnih dijelova teksta te do propuštanja ključnih podataka. Oslanjanje na početne upute može biti nedovoljno za dobivanje odgovarajućeg konteksta za složene probleme.

2. **Teškoće prilikom generiranja**: Model ponekad može generirati sadržaj koji nije utemeljen na dobivenom kontekstu, što rezultira halucinacijama. Generirani rezultati mogu biti nevažni ili pristrani, što smanjuje kvalitetu i pouzdanost odgovora.

3. **Izazovi u integraciji informacija**: Integracija pretraženih informacija može biti zahtjevna, što može dovesti do nepovezanih ili neusuglašenih odgovora. Može doći do ponavljanja kada se slične informacije pretraže iz više izvora. Utvrđivanje značaja i relevantnosti različitih pasusa te osiguravanje stilske i tonalne dosljednosti dodatno otežava proces.

4.3 Kako se ostvaruju kvalitetne semantičke značajke?

Da bismo razumjeli RAG tehnike, od ključne je važnosti prikazati razvoj modela ugradnje (*eng. embedding models*), koji danas omogućuju postizanje visokokvalitetnih semantičkih značajki teksta.

4.3.1 Metode dohvaćanja (*eng. Retrieval methods)*

Dohvaćanje informacija predstavlja kritičnu komponentu koja omogućava modelima pristup relevantnim vanjskim informacijama u stvarnom vremenu, čime se povećava točnost i relevantnost generiranih odgovora. U nastavku su navedene tehnike dohvaćanja (*eng. Retrieval methods*) podijeljene u dvije kategorije, prema vrsti vektora na kojima su temeljene: rijetki vektori (*eng. sparse vectors*) i gusti vektori (*eng. dense vectors*).

1. **Metode rijetkog dohvaćanja** (*eng. Sparse retrieval methods*): metode pretraživanja informacija koje se oslanjaju na korištenje rijetkih (*eng. sparse*) vektorskih reprezentacija teksta. *Sparse* vektori predstavljaju riječi ili rečenice pomoću vektorskih reprezentacija gdje svaka dimenzija korespondira određenoj riječi u vokabularu. Ove dimenzije su uglavnom nula (0) osim na mjestima koja odgovaraju prisutnim riječima. Rijetki vektori zahtijevaju manje memorije u usporedbi s gustim vektorima, posebno kada se radi o visokodimenzionalnim podacima gdje je većina elemenata nula, kao što su tekstualni podaci predstavljeni *bag-of-words* modelom ili vektorima frekvencije (*eng. frequency vectors*).

   TF-IDF (*eng. Term Frequency-Inverse Document Frequency*) je tradicionalna statistička metoda za procjenu važnosti riječi unutar dokumenta u korpusu. Izračunava frekvenciju riječi u dokumentu (TF) i uspoređuje je s učestalošću pojavljivanja riječi u cijelom korpusu (IDF). Na taj način, metoda identificira ključne riječi specifične za dokument. Metodu su razvili Gerard Salton i Christopher Buckley 1972. godine. Iako je TF-IDF efikasan za mnoge zadatke pretraživanja, ovisnost o točnom podudaranju ključnih riječi može rezultirati propuštanjem relevantnih dokumenata sa sinonimima. U modernom dubokom učenju, TF-IDF se sve manje koristi zbog naprednijih modela poput transformera i vektorskih reprezentacija koje bolje obrađuju semantičku sličnost i kontekst.

   Okapi BM25 tehnika temelji se na probabilističkom modelu rangiranja. Ova tehnika koristi slične principe kao TF-IDF, ali uključuje dodatna poboljšanja za bolje upravljanje duljinom dokumenata i procjenom relevantnosti. BM25 su razvili Stephen E. Robertson i Karen Spärck Jones tijekom 1970-ih i 1980-ih godina u okviru projekta Okapi na Sveučilištu City u Londonu [59].

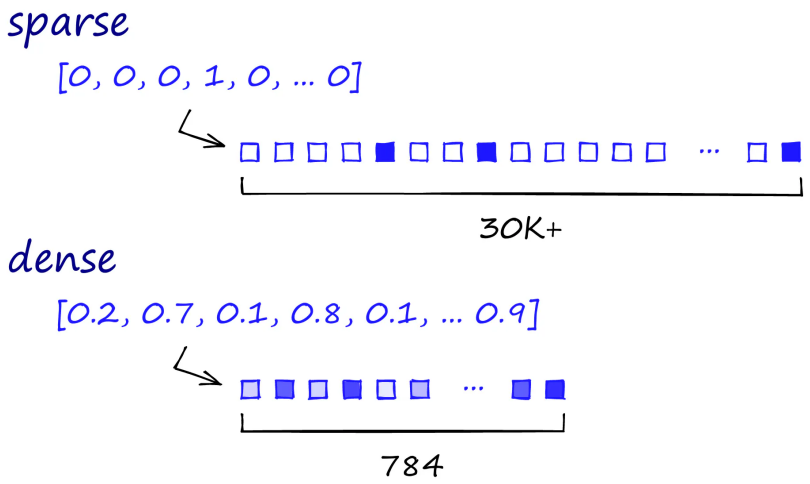
   Rijetki vektori omogućuju učinkovitije pohranjivanje podataka i izvođenje sintaktičkih usporedbi dviju sekvenci. Uzet ćemo kao primjer sljedeće dvije rečenice:

- "Bill ran from the giraffe toward the dolphin"
- "Bill ran from the dolphin toward the giraffe"

Unatoč različitom značenju rečenica, one su sastavljene od istih riječi, ali u različitom redoslijedu. Kao rezultat toga, rijetki vektori za ove rečenice bili bi vrlo slični ili čak identični, ovisno o načinu njihove izrade.

Rijetki vektori nazivaju se rijetkima jer sadrže malo informacija; tipično, sadrže mnogo nula, s tek nekoliko jedinica koje predstavljaju relevantne informacije. Posljedično, ovi vektori mogu imati mnogo dimenzija, često na desetke tisuća.

2. **Metode gustog dohvaćanja** (*eng. Dense retrieval methods*): familija metoda pretraživanja informacija koja se ne oslanja na točno podudaranje ključnih riječi, već na semantičko značenje gustih vektora (*eng. dense vectors*). Kod gustog dohvaćanja koriste se modeli gdje je semantika teksta ugrađena u kontinuirani vektorski prostor pomoću modela temeljenih na neuronskim mrežama. Slika 34 prikazuje usporedbu rijetkog i gustog vektora.



Slika 34: Usporedba rijetkog (*eng. sparse*) i gustog (*eng. dense*) vektora. Rijetki vektori sadrže rijetko raspoređene bitove informacija (imaju mnogo nula), dok su gusti vektori bogatiji informacijama koje su gusto "upakirane" u svakoj dimenziji.

Rane metode gustog dohvaćanja uključuju već spomenuti Word2Vec algoritam, koji se sastoji od ranih arhitektura temeljenih na neuronskim mrežama, poput CBOW (*eng. Continuous Bag-of-Words*) i Skip-gram. GloVe, koji je izašao 2014., razlikuje se od Word2Vec-a po tome što koristi globalne statistike učestalosti riječi u velikim tekstovima, a ne samo lokalni kontekst riječi [60].

Razvoj transformer arhitekture i modela temeljenih na pozornosti započeo je jednu novu eru razvoja tekstualnih modela ugrađivanja. 2018. godine dolazi do razvoja *contextual word embeddings* modela, poput: ELMo (*eng. Embeddings from Language Models*), BERT (*eng. Bidirectional Encoder Representations from Transformers*) i GPT (*eng. Generative Pre-trained Transformer*).

BERT: Legendarni model predstavljen 2018. godine od strane Google AI, BERT je revolucionirao način na koji se generiraju vektorske reprezentacije riječi. Vektorske reprezentacije za svaku riječ (ili *token*) generiraju se na sličan način kao kod Word2Vec metode, ali su znatno bogatije zahvaljujući dubljim mrežama [61]. BERT koristi *attention mehanizam* za kodiranje konteksta riječi, omogućujući modelu da "obraća pažnju" na specifične riječi ovisno o kontekstu, čime se postiže dublje razumijevanje semantičkih odnosa u tekstu. Kao *bidirectional* model, BERT analizira kontekst riječi s obje strane, i s lijeve i s desne strane, što dodatno poboljšava razumijevanje jezika. Međutim, jednostavne metode kombiniranja vektora pojedinih *tokena* nisu davale zadovoljavajuće rezultate za cijele rečenice, jer nisu uspjele adekvatno sažeti cjelokupno značenje rečenice u jedan koherentan vektor.

Sentence Transformers: Sentence Transformers modeli, poput Sentence-BERT-a, razvijeni su 2019. godine i predstavljaju značajan napredak u gustim vektorskim reprezentacijama rečenica. Ovi modeli ugrađuju cijele rečenice ili odlomke, omogućujući preciznije hvatanje semantičkog značenja. Sentence-BERT koristi BERT kao bazu i prilagođava ga za stvaranje visokokvalitetnih rečeničnih ugradbi (*eng. sentence embeddings*) koje su korisne u zadacima poput pretraživanja i semantičke sličnosti. Danas broje oko 5000 pretreniranih modela dostupnih na Hugging Face platformi [62].

4.3.2 Suvremeni pristup modelima tekstualnog ugrađivanja

U 2024. godini, najnapredniji modeli za ugrađivanje teksta temelje se na dubokom učenju i suvremenoj transformer arhitekturi. Međutim, evaluacija ovih modela predstavlja značajan izazov. Postoji mnogo različitih modela za tekstualnu ugradnju, a s obzirom na tu raznolikost, razvijeno je i mnogo *benchmark* testova za njihovu evaluaciju. Jedan od najpoznatijih benchmark testova je *Massive Text Embedding Benchmark* (MTEB) koji mjeri performanse putem raznih zadataka, uključujući 56 *datasetova* raspoređenih u 8 zadataka. Trenutno, MTEB broji preko 350 modela na svojoj rang listi [63]. Evaluacija modela za tekstualnu ugradnju je složena zbog različitih zahtjeva zadataka. Model koji je dobar u određivanju sličnosti teksta može loše performirati kod zadataka pretraživanja. Sličnost teksta zahtijeva semantičku bliskost, dok se pretraživanje fokusira na pronalaženje relevantnog sadržaja temeljenog na specifičnom upitu. Osim toga, određeni jezici sa svojim jedinstvenim lingvističkim osobinama zahtijevaju specijalno prilagođene *benchmark* testove. Primjerice, arapski jezik, predstavlja poseban izazov za modele i zahtijeva posebne metode evaluacije. Uz performanse na zadacima, važni su i drugi parametri prilikom procjene modela za tekstualnu ugradnju. Efikasnost, veličina i dimenzionalnost ugrađivanja su ključni, osobito kod primjene u stvarnim svjetovima. Ovi faktori mogu utjecati na brzinu modela, potrebne resurse i ukupnu korisnost u stvarnim aplikacijama.

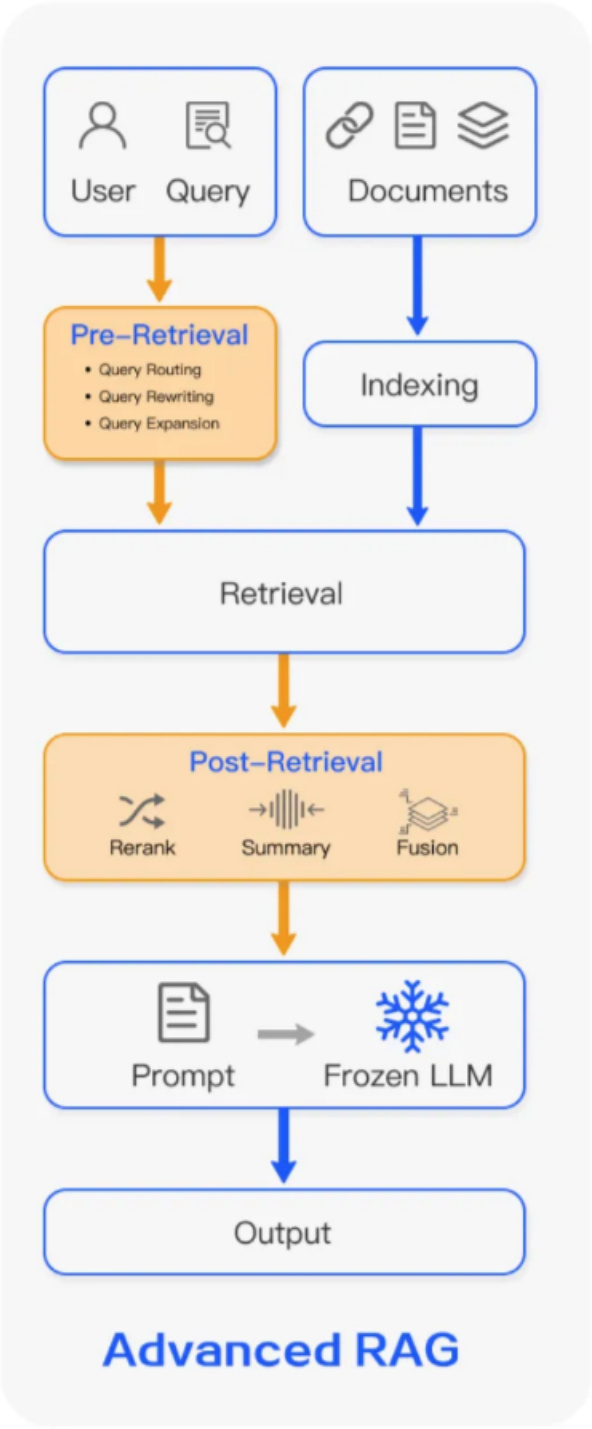
U posljednjih desetak godina, *embedding* modeli, posebno oni temeljeni na velikim jezičnim

modelima, postali su stalni fokus istraživanja i daljnjeg razvoja zbog svoje sposobnosti da precizno ugrađuju semantičke informacije i kontekst.

Moćna rješenja velikih kompanija, poput Alibaba GTE, Salesforce SFR, Nvidia te Intel *neural embedding* modeli, značajno su unaprijedila ovu tehnologiju. S druge strane, komercijalna rješenja, poput najnovijih OpenAI *text-embedding-3* modela, Voyage-evog *Instruct* modela i Google *Gecko*, također postavljaju visoke standarde u industriji, nudeći svojim klijentima pristup ovim modelima putem API-ja. Osim ovih, postoji i veliki broj otvorenih rješenja koji se može preuzeti, često putem Hugging Face platforme, kao što su intfloat E5 modeli, Snowflake Artic i mnogi drugi. Ovi napredni modeli omogućuju bogate kontekstualizirane reprezentacije teksta, nadmašujući tradicionalne *encoder* modele u raznim zadacima tekstualnog ugrađivanja, uključujući MTEB. Suvremeni veliki jezični modeli postižu izvrsne rezultate na zadacima semantičke tekstualne sličnosti koristeći dobro osmišljene *zero-shot* ili *few-shot* strategije upita, često nadmašujući modele kao što su SBERT i RoBERTa [64] [65].

4.4 Napredni RAG

Napredni RAG donosi značajna poboljšanja u odnosu na osnovni RAG, fokusirajući se na optimizaciju pretraživanja i generacije kako bi se prevladali nedostaci osnovne tehnike. Ova tehnika koristi strategije pretraživanja prije (*eng. pre-retrieval*) i nakon dohvaćanja (*eng. post-retrieval*), uključujući optimizaciju indeksiranja, precizniju segmentaciju i dodavanje metapodataka kako bi se poboljšala kvaliteta sadržaja koji se indeksira (Slika 35).



Slika 35: Napredni (*eng. advanced*) RAG pristup

4.4.1 Proces prije dohvaćanja (*eng. Pre-retrieval process*)

U ovoj fazi primarni je cilj optimizirati strukturu indeksiranja i početni upit korisnika. Optimizacija indeksiranja uključuje povećanje granularnosti podataka, poboljšanje struktura indeksa, dodavanje metapodataka, optimizaciju usklađivanja i mješovito dohvaćanje. Optimizacija upita ima za cilj učiniti korisničko pitanje jasnijim i pogodnijim za dohvaćanje relevantnih informacija. Uobičajene metode uključuju prepisivanje upita (*eng. query rewriting*), transformaciju upita (*eng. query transformation*), proširenje upita (*eng. query expansion*), ali i one sofisticiranije poput odabira ispravnog indeksa, dodavanje metapodataka (*eng. metadata attachment*) i optimizacije segmenata (*eng. chunk optimization*) [58].

1. Optimizacija indeksiranja (*eng. Indexing Optimization*): Kako bismo poboljšali kvalitetu indeksiranog sadržaja, trebamo unaprijediti kvalitetu sirovih podataka, povećati njihovu granularnost i koristiti odgovarajuće indekse.

- **Poboljšanje kvalitete podataka:**
  - **Uklanjanje nevažnih informacija:** Prije nego što se podaci indeksiraju, potrebno je ukloniti sve informacije koje nisu relevantne za predmet ili upit. Primjerice, prilikom indeksiranja članaka o zdravlju, uklanjaju se informacije koje nisu povezane s medicinskim savjetima.
  - **Uklanjanje dvosmislenosti:** Koriste se standardizirani nazivi i pojmovi kako bi se izbjegla zabuna. Recimo za osobu s imenom "Ivan Horvat", potrebno je osigurati da se uvijek referira kao "Ivan Horvat" umjesto "I. Horvat" ili "Ivan H.".
  - **Provjera točnosti:** Potrebno je potvrditi činjenice i informacije prije nego što se uključe u indeks. Ako se radi o znanstvenim podacima, potrebno je provjeriti vjerodostojnost izvora.
  - **Održavanje konteksta:** Potrebno je osigurati da informacije zadrže svoj originalni kontekst kako bi bile ispravno interpretirane. Na primjer, citati se zadržavaju u kontekstu njihovog izvorno objavljenog članka.
- **Odabir odgovarajuće vektorske baze podataka ili biblioteke**

  Vektorske biblioteke su namijenjene radu sa statičnim podacima gdje su indeksi nepromjenjivi. One pohranjuju isključivo vektorske reprezentacije, bez povezanih objekata iz kojih su generirane. Zbog toga vektorske biblioteke poput FAISS i ANNOY ne podržavaju CRUD operacije, što otežava dodavanje novih dokumenata postojećem indeksu.

  Vektorski indeks je struktura podataka koja omogućuje brzu pretragu i dohvaćanje vektorskih reprezentacija na temelju sličnosti. U vektorskim indeksima, podaci su predstavljeni kao vektori u višedimenzionalnom prostoru, a pretraga se temelji na mjerenju sličnosti

između tih vektora, često korištenjem metrika poput kosinusne sličnosti ili euklidske udaljenosti [66].

S druge strane, klasični indeksi u bazama podataka koriste strukture poput *B-stabla* ili *hash tablica* za brzo dohvaćanje podataka na temelju točnog podudaranja ili raspona vrijednosti. Dok klasični indeksi omogućuju učinkovitu pretragu i dohvaćanje točnih vrijednosti ili raspona vrijednosti, vektorski indeksi su optimizirani za pretragu sličnosti u velikim skupovima podataka, što je ključno za aplikacije poput prepoznavanja slika, pretraživanja tekstualnih podataka i preporučenih sustava (*eng. recommender systems*) [66].

Vektorske baze podataka dizajnirane su za upravljanje velikim količinama podataka, podržavaju CRUD operacije, distribuirane su, te omogućuju replikaciju i toleranciju na greške. Također, podržavaju različite metode indeksiranja i omogućuju brze pretrage vektorske sličnosti.

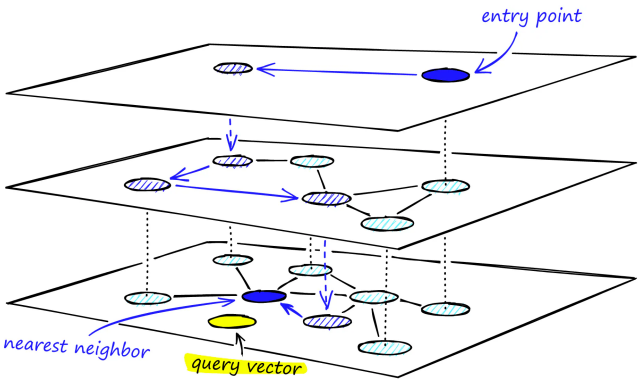
Baze podataka s podrškom za vektorske operacije uključuju:

1. **SQL baze podataka s vektorskom podrškom**: pgvector, Supabase, StarRocks
2. **Baze podataka za pretraživanje cijelog teksta**: ElasticSearch, OpenSearch
3. **NoSQL baze podataka s vektorskom podrškom**: Redis, MongoDB
4. **Namjenske vektorske baze podataka**: Pinecone, Milvus, Weaviate, Qdrant, Vald, Chroma, Vespa, Vearch

- **Odabir odgovarajuće vrste indeksa**

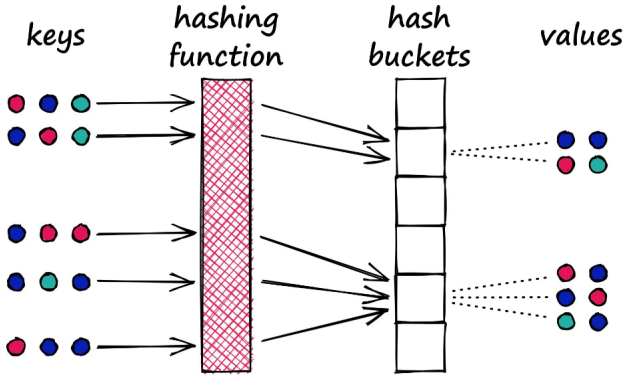
  Indeksi su ključni za učinkovitost pretraživanja u vektorskim bazama podataka i bibliotekama. Postoji nekoliko vrsta indeksa, od kojih svaki nudi različite kompromise između učinkovitosti, brzine i točnosti [67]:

  - **FLAT indeks**: Ovaj tip indeksa ne koristi nikakve tehnike optimizacije ili aproksimacije. Pruža potpunu točnost (100%), ali je spor i neučinkovit jer se pretraga vrši kroz cijeli skup podataka bez ubrzanja.
  - **IVF_FLAT**: Ovaj indeks koristi invertirane datoteke za grupiranje vektora u manje podskupove, što povećava brzinu pretraživanja. Međutim, točnost je kompromitirana u odnosu na FLAT indeks jer se pretraga vrši samo unutar odabranih podskupova.
  - **HNSW** (*Hierarchical Navigable Small World*): Ovaj indeks koristi hijerarhijsku navigaciju za brzo pretraživanje vektorskih podataka. Pruža dobar balans između točnosti i brzine pretraživanja. HNSW koristi strukturu grafova za povezivanje vektora (*eng. Graph-based index*), omogućujući brzo pronalaženje sličnih vektora navigacijom kroz graf [68].



Slika 36: Proces pretraživanja u hijerarhijskoj strukturi HNSW grafa

– **LSH** (*Locality-Sensitive Hashing*): LSH koristi specijalizirane hash funkcije koje s velikom vjerojatnošću mapiraju slične ulazne vrijednosti na isti hash kôd. Ova tehnika omogućava brže pretraživanje sličnih vektora, iako uz određeni kompromis u točnosti, jer vektori koji su slični u izvornom prostoru ponekad mogu biti raspoređeni u različite "hash kante" [69]. Slika 37 ilustrira rad LSH algoritma.



Slika 37: LSH funkcija ima za cilj smjestiti slične vrijednosti u iste "hash kante"

– **PQ** (*Product Quantization*): PQ dijeli vektore na manje podvektore i kvantizira svaki podvektor posebno. Ovaj pristup značajno smanjuje memorijsku složenost i ubrzava pretraživanje, ali može smanjiti točnost zbog kvantizacijskih pogrešaka.
– **ANNOY** (*Approximate Nearest Neighbors Oh Yeah*): ANNOY koristi šume binarnih stabala (*eng. random decision forests*) za ubrzanje pretraživanja najbližih susjeda. Ovaj pristup omogućava brzo pretraživanje uz dobar kompromis između brzine i točnosti, i često se koristi u sustavima za preporuke (*eng. Recommender system*) [70]. Metodu je razvio Erik Bernhardsson 2015. godine dok je radio u Spotifyju. ANNOY je dizajniran za pretraživanje gustih vektora do 1000 dimenzija. Za izračun najbližih susjeda, dijeli skup točaka na polovice i to radi rekurzivno dok svaki skup ne sadrži k stavki, gdje je k obično oko 100.
– **FAISS** (*Facebook AI Similarity Search*): Faiss je biblioteka za efikasno pretraživanje i kvantizaciju gustih vektora, koja podržava nekoliko vrsta indeksa, uključujući

FLAT, IVF i HNSW, kao i njihove kombinacije, kako bi se postigao optimalan balans između brzine i točnosti [71].

- **Performanse različitih vektorskih baza podataka**

  Performanse vektorskih baza podataka značajno variraju ovisno o specifičnim zahtjevima i skupovima podataka. Analiza nekoliko istaknutih baza podataka pokazuje sljedeće karakteristike:

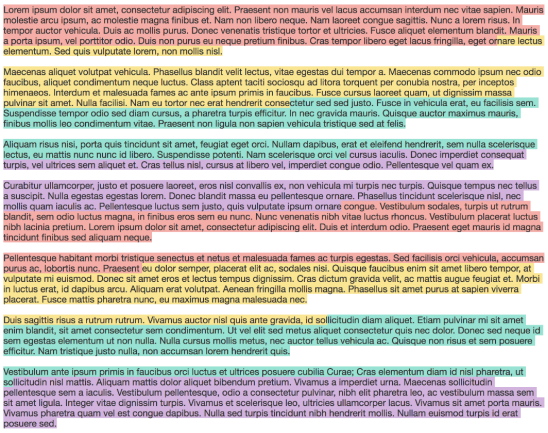
  - **Milvus**: Ističe se visokim *throughputom* za različite skupove podataka, poput *glove-100-angular* i *nytimes-256-angular*. Pruža brzu pretragu i umjereno vrijeme izrade indeksa [72]. Osim toga, Milvus je fleksibilan i podržava različite algoritme pretrage vektora, što ga čini pogodnim za širok spektar primjena.
  - **Weaviate**: Poznata po najmanjoj veličini indeksa i dobrim performansama. Iako izrada indeksa traje dulje, nudi visoku točnost pretrage i fleksibilnost. Weaviate podržava integraciju s popularnim ML modelima, brzu pretragu i dodatne funkcije poput preporuka i sažimanja.
  - **Pinecone**: Izvrsna zbog niske latencije i visokog broja upita po sekundi (QPS). Idealna za aplikacije koje zahtijevaju brze odgovore. Nudi potpuno upravljanu uslugu, ali nije otvorenog kôda. Pinecone je visoko skalabilan i podržava stvarnu vremensku obradu podataka te integraciju s Langchain okvirom za razvoj LLM aplikacija.
  - **Qdrant**: Nudi dobar omjer cijene i performansi, s niskim troškovima za manje skupove podataka. Prikladna za *startupove* i projekte s ograničenim budžetom [73]. Qdrant omogućuje brze i precizne pretrage koristeći prilagođeni HNSW algoritam te podržava napredne filtere i razne tipove podataka.
  - **Chroma**: Ističe se visokom preciznošću pretrage i skalabilnošću. Prilagodljiva je za specifične industrijske potrebe, s dobrom integracijom s modernim AI i ML alatima (Langchain i LlamaIndex), što je čini izvrsnom za napredne analitičke zadatke. Chroma je otvorenog kôda i omogućuje jednostavno upravljanje tekstualnim dokumentima, konverziju teksta u vektore te slične pretrage.

  Odabir odgovarajuće vektorske baze podataka ili biblioteke ključan je za postizanje visokih performansi i skalabilnosti RAG metode. Pravilnim razumijevanjem razlika između vektorskih baza podataka i biblioteka, kao i karakteristika različitih indeksa, sustav se optimizira za specifične potrebe i ciljeve. Naravno, postoje i druge vektorske baze podataka koje također mogu biti relevantne za različite primjene.

2. Optimizacija segmenata (*eng. Chunk Optimization*): S obzirom na to da RAG tehnike koriste vanjske izvore za dohvaćanje podataka, prvi korak je inteligentna podjela tih podataka

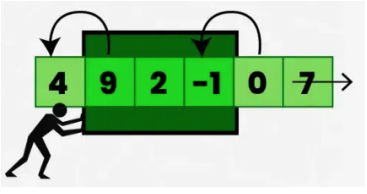
u manje dijelove (*eng. chunks*) kako bi se izdvojile ključne značajke svakog segmenta. Nakon toga, ti se segmenti ugrađuju u vektorske značajke. Međutim, problem s prevelikim ili premalim segmentima može smanjiti performanse RAG tehnika. Zato se koriste metode optimizacije veličine blokova (*eng. chunk size*) i različite tehnike segmentacije koje pomažu u postizanju optimalne ravnoteže.

- **Segmentacija fiksne duljine** (*eng. Fixed-sized Chunking*): Segmentacija fiksne duljine je tehnika podjele podataka u segmente s unaprijed definiranom konstantnom veličinom (*eng. chunk size*). Primjer, raspodjela dokumenta od 1000 riječi na 10 segmenata od 100 riječi. Prednosti uključuju jednostavnu implementaciju te je metoda računalno efikasna. Također, segmentacija fiksne duljine je računalno jeftina i ne zahtijeva korištenje NLP biblioteka. Međutim, često dolazi do gubitka konteksta prekidanjem rečenice ili paragrafa te se na taj način ignorira inherentna struktura teksta [74].



Slika 38: Segmentacija fiksne duljine

- **Tehnika kliznog prozora** (*eng. Sliding Window*): *Sliding Window* uključuje stvaranje preklapajućih segmenata teksta pomoću prozora fiksne veličine koji se pomiče preko dokumenta za zadani korak (*eng. stride*). Na primjer, s prozorom od 1000 znakova i korakom od 500 znakova, prvi segment bi obuhvaćao znakove od 1. do 1000., drugi segment bi obuhvaćao znakove od 501. do 1500., itd. Prednosti ove metode uključuju bolju očuvanost konteksta, jer se ključne informacije blizu granica segmenata uključuju u više segmenata. Ova tehnika može dovesti do boljih u usporedbi sa segmentacijom fiksne duljine [75].



Slika 39: Ilustracija okvira kod tehnike kliznog prozora

- **Segmentacija obzirom na sadržaj** (*eng. "Content-aware" Chunking*): Segmentacija prema sadržaju koristi karakteristike samog sadržaja koji segmentiramo, primjenjujući naprednije metode segmentacije. Neke od tih metoda uključuju:

  - **Razdvajanje rečenica** (*eng. Sentence Splitting*): Mnogi modeli optimizirani su za ugrađivanje sadržaja na razini rečenice, a segmentacija rečenica omogućuje iskorištavanje prednosti takvih modela. Postoji nekoliko pristupa i alata za razdvajanje rečenica. Najjednostavniji je naivno razdvajanje, koje koristi točke i nove redove za razdvajanje rečenica. Iako je brz i jednostavan, ovaj pristup ne uzima u obzir sve moguće rubne slučajeve. S druge strane, **Natural Language Toolkit** (NLTK) je popularna Python biblioteka za rad s podacima ljudskog jezika, koja pruža razne *tokenizer* module u svrhu ekstrakcije smislenijih segmenata [76]. Još jedna moćna i modernija Python biblioteka za NLP zadatke je **spaCy**, koja nudi sofisticirane alate za segmentaciju rečenica, efikasno dijeleći tekst u zasebne rečenice i omogućujući bolje očuvanje konteksta u rezultirajućim segmentima [77].
  - **Rekurzivna segmentacija** (*eng. Recursive Chunking*): Rekurzivna segmentacija je metoda koja dijeli ulazni tekst na manje dijelove na hijerarhijski i iterativni način, koristeći skup separatora. Ako početni pokušaj podjele teksta ne rezultira segmentima željene veličine ili strukture, metoda se rekurzivno primjenjuje na dobivene segmente s drugim separatorom ili kriterijem sve dok se ne postigne željena veličina ili struktura segmenta.

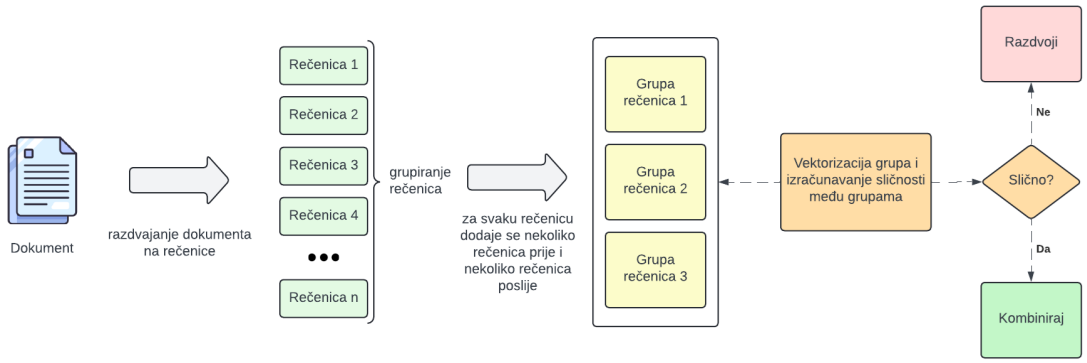
    Primjer ove metode može se pronaći u Langchain okviru pod *recursive text splitter* kategorijom, koji koristi parametarski definiranu listu znakova za segmentaciju. Ovaj alat pokušava podijeliti tekst prema ovim znakovima redom dok segmenti ne postanu dovoljno mali. Zadana lista znakova je `["\n\n", "\n", " ", ""]` što omogućava da se paragrafi (a zatim rečenice i riječi) drže zajedno što je dulje moguće, jer su ti dijelovi obično najjače semantički povezani.
  - **Specijalizirana segmentacija** (*eng. Specialized Chunking*): Markdown i LaTeX su dva primjera strukturiranog i formatiranog sadržaja. U slučajevima kada radimo s takvim podacima, možemo koristiti specijalizirane metode segmentacije koje zadržavaju originalnu strukturu i sadržaj ovih formata. Primjerice, kod Markdown formata, ove metode prepoznaju sintaksu (npr. naslove, liste, tablice, blokove kôda...), te mogu inteligentno segmentirati sadržaj na temelju takve strukture, što rezultira semantički koherentnijim dijelovima.
  - **Semantička segmentacija** (*eng. Semantic Chunking*): Greg Kamradt je prvi predstavio inovativnu tehniku segmentacije teksta temeljenu na značenju [78]. U svom radu, Kamradt je uvjerljivo argumentirao da je korištenje uniformnih veličina segmenata previše trivijalno jer ne uzima u obzir semantičke odnose unutar dokumenta. Takav pristup ne omogućuje razlikovanje tematski povezanih segmenata od onih koji

nisu. Koristeći napredne jezične modele u razvoju aplikacija, možemo primijeniti modele za tekstualnu ugradnju koji omogućavaju izdvajanje dubokog značenja iz podataka. Ova semantička analiza može se koristiti za stvaranje segmenata sastavljenih od rečenica koje se bave istom temom ili konceptom, čime se poboljšava jasnoća i relevantnost informacija.

Algoritam semantičke segmentacije sastoji se od sljedećih koraka:

1. **Podijeli dokument na rečenice**
2. **Grupiraj rečenice**: za svaku rečenicu, stvori grupu koja sadrži nekoliko rečenica prije i poslije dotične rečenice. Grupa je "usidrena" u rečenici koja se koristi kao centralna. Specifični broj rečenica prije i poslije određujemo kao hiperparametar, ali sve rečenice u grupi vežu se uz jednu glavnu rečenicu.
3. **Generiraj vektorske reprezentacije za svaku grupu rečenica**: pri čemu su sve rečenice povezane s glavnom rečenicom.
4. **Usporedi sličnosti (udaljenosti) između svake uzastopne grupe**: Dok analiziramo uzastopne rečenice, ako tema/značenje ostaje isto, udaljenost između vektora trenutne grupe i prethodne grupe bit će mala. S druge strane, veća semantička udaljenost ukazuje na promjenu teme. Na taj način, segmenti se mogu učinkovito odvojiti jedan od drugog.

Opisani algoritam prikazan je na slici 40:



Slika 40: Postupak semantičke segmentacije dijelova rečenice

3. Dodavanje metapodataka (*eng. Metadata attachments*): Metapodaci mogu obogatiti segmente informacija dodavanjem podataka kao što su: **broj stranice**, **naziv datoteke**, **autor**, **kategorija**, **sažetak** i **vremenski žig**. Ovo omogućava filtriranje pretrage na temelju metapodataka, čime se ograničava opseg pretrage. Dodjeljivanje različitih težina vremenskim oznakama dokumenata tijekom pretrage može ostvariti vremenski svjestan RAG, osiguravajući svježinu informacija i izbjegavanje zastarjelih podataka.

Osim izdvajanja metapodataka iz originalnih dokumenata, metapodaci se mogu i umjetno konstruirati, poput **dodavanja sažetaka paragrafa** ili **uvođenja hipotetskih pitanja**. Konkretno, korištenje velikih jezičnih modela za generiranje pitanja koja se mogu odgovoriti pomoću

dokumenta može dodatno obogatiti metapodatke. Ova metoda uključuje kreiranje hipotetskih pitanja koja odgovaraju sadržaju dokumenta, a zatim izračunavanje sličnosti između originalnog pitanja i tih hipotetskih pitanja tijekom pretraživanja. Na taj način smanjuje se semantički jaz između postavljenog pitanja i odgovora, čime se povećava preciznost pretraživanja.

Na primjer, pretpostavimo da imamo dokument o povijesti rimskog carstva. LLM može generirati hipotetsko pitanje poput "Koji su glavni uzroci pada rimskog carstva?" čak i ako originalni dokument sadrži rečenice koje detaljno objašnjavaju političke, ekonomske i vojne razloge pada. Tijekom pretraživanja, ako korisnik postavi pitanje "Zašto je Rimsko carstvo propalo?", sustav može prepoznati visoku sličnost između ovog pitanja i generiranog hipotetskog pitanja, omogućavajući preciznije i relevantnije rezultate.

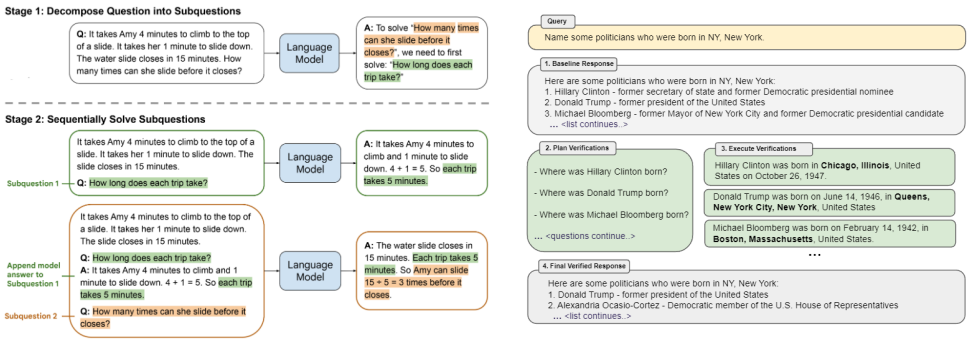
4. Optimizacija upita (*eng. Query optimization*): Jedan od većih izazova kod osnovne RAG tehnike je preveliko oslanjanje na početni korisnički upit kao bazu za daljnje dohvaćanje. Formulacija preciznog upita je izazovna, a nepromišljeni upiti rezultiraju lošom učinkovitošću u procesu dohvaćanja. Ponekad je samo pitanje složeno, ili jezik nije dobro organiziran (leksički stil). Još jedna poteškoća leži u samoj prirodi jezika, odnosno njegovoj višeznačnosti. Veliki jezični modeli često imaju poteškoća s razumijevanjem specijaliziranih termina ili višeznačnih kratica. Na primjer, možda neće uvijek moći razlikovati odnosi li se "LLM" na veliki jezični model (Large Language Model) ili na "Master of Laws" (titulu magistra prava).

Optimizacija upita postiže se kroz nekoliko tehnika:

1. **Proširenje upita** (*eng. Query expansion*): Ova tehnika uključuje proširenje jednog upita u više upita kako bi se obogatio sadržaj i kontekst, omogućujući preciznije odgovore. Na primjer, složeno pitanje može se rastaviti na niz jednostavnijih potpitanja koristeći metodu *least-to-most prompting*. Korištenjem LLM-ova za proširenje upita, ovi upiti mogu se paralelno izvršavati i validirati kako bi se smanjile halucinacije i povećala pouzdanost odgovora.

    - Metoda ***least-to-most prompting***: Temelji se na razlaganju složenih problema na niz jednostavnijih potproblema koji se rješavaju sekvencijalno [79]. Svako rješenje prethodnog potproblema olakšava rješavanje sljedećeg. Eksperimenti pokazuju da ova metoda omogućava modelima poput GPT-3 da preciznije rješavaju teže zadatke nego s metodom *chain-of-thought*. Na primjer, model GPT-3 postigao je 99% točnosti na SCAN *benchmarku* koristeći samo 14 primjera, dok je s metodom *chain-of-thought* postigao samo 16% točnosti.
    - Metoda ***Chain-of-Verification (CoVe)***: Generiranje plauzibilnih, ali netočnih informacija, poznatih kao halucinacije, neriješen je problem velikih jezičnih modela. CoVe metoda temelji se na sposobnošću LLM-ova da promišljaju o svojim odgovorima kako bi ispravili potencijalne pogreške [80].

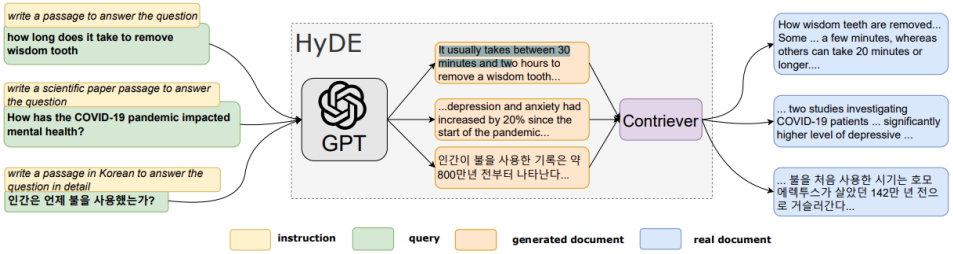
Unutar ove metode, model najprije sastavlja početni odgovor, a zatim planira pitanja za provjeru činjenica vezanih uz taj odgovor. Sljedeći korak je neovisno odgovaranje na ta pitanja kako bi se osiguralo da odgovori nisu pristrani prema drugim odgovorima. Na kraju, model generira konačan, verificirani odgovor. Eksperimenti pokazuju da metoda CoVe smanjuje pojavu halucinacija u raznim zadacima, uključujući pitanja iz Wikidata, zatvorena pitanja iz MultiSpanQA i generiranje dugih tekstova. Validirani prošireni upiti obično pokazuju veću pouzdanost. Slika 41 prikazuje tijek rada *least-to-most* (lijevo) i CoVe metode (desno).



a) *least-to-most* b) CoVe

Slika 41: a) *least-to-most* metoda, b) CoVe metoda

2. **Transformacija upita** (*eng. Query transformation*): Ova metoda uključuje preoblikovanje izvornog upita radi poboljšanja preuzimanja informacija. Primjer je korištenje LLM-a za prepisivanje upita (*eng. Query rewrite*) ili generiranje hipotetičkih dokumenata HyDE tehnikom koji pomažu u boljoj usporedbi s postojećim odgovorima. Kod HyDE tehnike, inicijalni upit se koristi za generiranje hipotetskog dokumenta koji odgovara na postavljeno pitanje. Ovaj generirani dokument može sadržavati netočne informacije, ali je zamišljen da bude sličan relevantnom dokumentu. Generativni proces hvata "relevantnost" pružajući primjer, a generirani dokument se zatim kodira u vektorsku reprezentaciju. Na taj način se dohvaćaju stvarni dokumenti koji su najsličniji generiranom dokumentu na temelju sličnosti vektora. Slika 42 prikazuje princip rada Hyde tehnike.



Slika 42: Primjer rada HyDe metode za optimizaciju upita. Zelenom bojom prikazani su stvarni upiti korisnika, a žutom dodatne instrukcije koje se prilažu u svrhu optimizacije upita. Smeđom bojom prikazani su generirani dokumenti, koji se potom uspoređuju sa stvarnim dokumentima prikazanim u plavoj boji.

3. **Usmjeravanje upita** (*eng. Query routing*): Usmjerivači (*eng. Router*) su moduli koji na temelju korisničkog upita i skupa pravila mogu usmjeriti pretragu prema relevantnim dijelovima podataka ili drugim RAG modulima, optimizirajući proces pretraživanja. To su jednostavni, ali moćni moduli koji koriste moć rasuđivanja LLM-ova za potrebe donošenja odluka. Primjerice, mogu se koristiti za sljedeće radnje:

   - Odabir pravog izvora podataka iz širokog spektra dostupnih mogućnosti.
   - Odabir ispravne *retrieval* tehnike koja je najprimjerenija za dani upit.
   - Primjena više tehnika istovremeno te kombiniranje njihovih rezultata za poboljšanje preciznosti i relevantnosti.

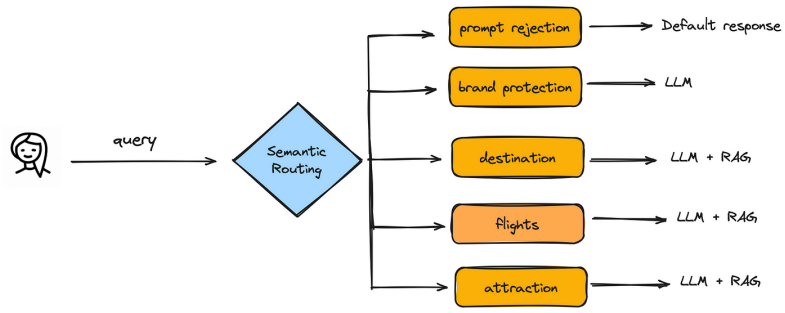
   Usmjerivače upita možemo podijeliti u 2 kategorije:

   - **Usmjerivač temeljen na metapodacima (*eng. Metadata router*)**: Ovaj korak započinje izdvajanjem ključnih riječi (entiteta) iz korisničkog upita, a zatim filtriranjem tih ključnih riječi i metapodataka unutar različitih dijelova kako bi se preciznije odredio opseg pretrage.

     Na primjer, ako korisnik postavi pitanje: "Koje su karakteristike mediteranske klime?", *Metadata router* će prepoznati ključne riječi poput "mediteranska klima" i "karakteristike".

     Upotrebom metapodataka kao što su *Tema*: "klima" i *Lokacija*: "mediteranska regija", upit će se usmjeriti prema relevantnim dijelovima podataka koji sadrže informacije o klimatskim značajkama mediteranske regije ili će se proslijediti odgovarajućem RAG pipeline-u za temeljitije pretraživanje.
   - **Semantički usmjerivač (*eng. Semantic router*)**: Ovaj pristup koristi semantičke informacije upita za inteligentno usmjeravanje. Specifične metode uključuju primjenu semantičkih modela koji interpretiraju značenje upita i usmjeravaju ga prema najrelevantnijim dijelovima podataka.

     Na primjer, kada korisnik postavi pitanje: "Koje su karakteristike mediteranske klime?", *Semantic router* analizira značenje upita koristeći napredne jezične modele. Umjesto da se oslanja samo na ključne riječi, Semantic Router razumije da korisnik traži detaljan opis klimatskih uvjeta specifičnih za mediteransku regiju. Na temelju ove semantičke analize, upit se usmjerava prema podacima koji objašnjavaju specifičnosti mediteranske klime, uključujući temperaturne obrasce, količinu oborina, vjetrove i sezonske promjene. Slika 43 ilustrira usmjeravanje korisničkog upita koristeći semantički usmjerivač.

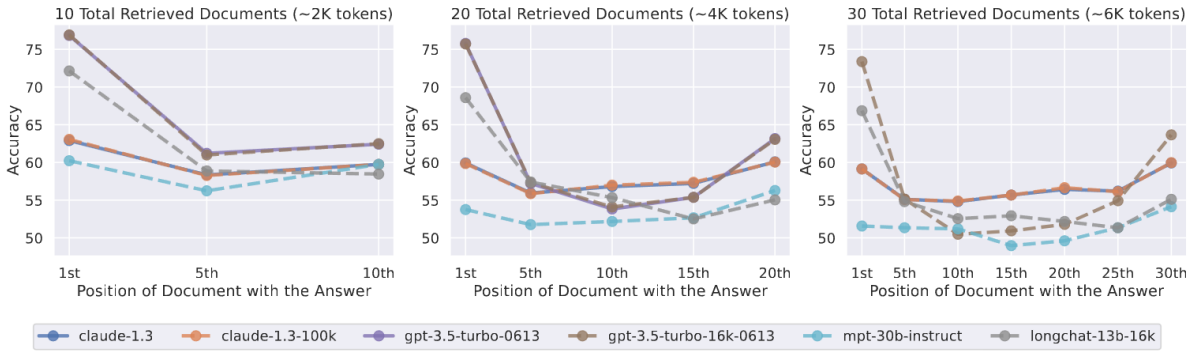


Slika 43: Ilustracija usmjeravanja koristeći semantički usmjerivač (*eng. semantic router*)

4.4.2 Proces nakon dohvaćanja (*eng. Post-retrieval process*)

Nakon što su relevantni dokumenti dohvaćeni, ključno je učinkovito integrirati ih s upitom. Glavne metode u ovoj fazi uključuju ponovno rangiranje segmenata (*eng. Reranking*) i kompresiju konteksta (*eng. Context compressing*). Ponovno rangiranje dohvaćenih informacija pomaže u isticanju najrelevantnijeg sadržaja, smanjujući preopterećenje informacijama i osiguravajući fokus na ključnim detaljima. Proces nakon dohvaćanja od ključne je važnosti u RAG-u zbog prirode semantičkog pretraživanja nad velikim skupovima tekstualnih dokumenata. Naime, prilikom vektorskog pretraživanja, tekstovi se kodiraju u vektore koji predstavljaju semantički sažetak značenja teksta. Iako je ovo učinkovito za brzo pretraživanje, dolazi do gubitka informacija zbog kompresije teksta u gusti vektor. To može rezultirati time da relevantni dokumenti budu rangirani niže od top_k praga, što negativno utječe na kvalitetu odgovora LLM-a.

Kako bismo riješili ovaj problem, možemo povećati broj dokumenata koji dohvaćamo, ali zbog ograničenja kontekstualnog prozora (*eng. context window*) LLM-a, ne možemo prenijeti sve. Dodatno, povećanjem količine teksta unutar kontekstualnog prozora smanjuje se sposobnost modela da prepozna ključne informacije (*eng. LLM Recall problems*) [81]. Stoga, nužno je maksimizirati učinkovitost pretraživanja i istovremeno smanjiti broj dokumenata koje model obrađuje.



Slika 44: Učinak promjene položaja relevantnih informacija (dokumenata koji sadrže odgovor) na performanse odgovaranja na pitanja iz većeg korpusa dokumenata. Performanse se pokazuju najboljima kada se relevantne informacije nalaze na samom početku ili samom kraju i rapidno degradiraju kada modeli moraju rasuđivati na temelju informacija u sredini ulaznog konteksta.

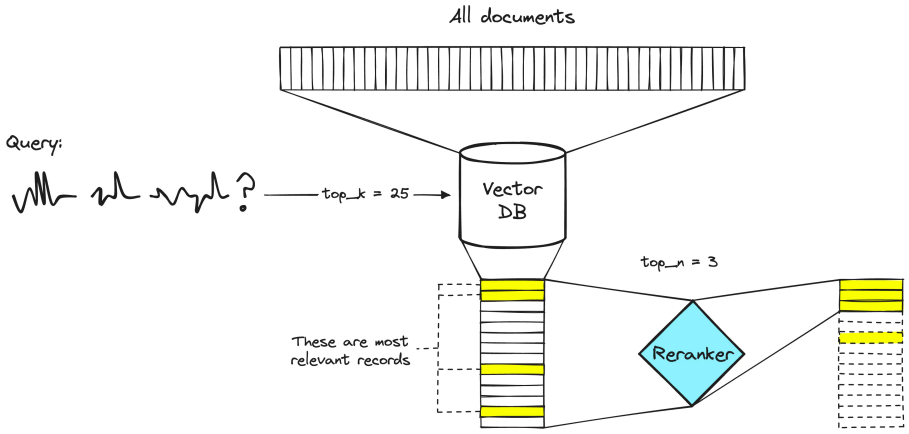
1. Ponovno rangiranje (*eng. Re-ranking*): Ponovno rangiranje (*eng. Re-ranking*) popularna je tehnika optimizacije performansi RAG modela koja se temelji na dodatnom koraku evaluacije i sortiranja dokumenata ili rezultata pretraživanja prije generiranja odgovora. Kako je već navedeno, proces kompresije dokumenata u guste vektore uključuje gubitak informacija. Isto tako, kako se veći dokumenti često dijele u manje segmente prilikom vektorske ugradnje, održavanje konteksta kroz te manje dijelove predstavlja novi problem.

Re-ranking model je oblik modela koji računa *score* poklapanja između danog korisničkog upita i dokument para. Ovaj se rezultat zatim može upotrijebiti za preuređivanje rezultata vektorskog pretraživanja, osiguravajući da najrelevantniji rezultati imaju prioritet [82].

1. **Korak**: Dohvaćanje relevantnih dokumenata iz velike baze podataka pomoću vektorske pretrage zbog njezine brzine.

2. **Korak**: Nakon što se dobiju ovi povezani dokumenti, primjenjuje se ponovno rangiranje kako bi se najrelevantniji dokumenti prioritetno postavili na vrh.

3. **Korak**: Ovi najviše rangirani dokumenti, koji se usko podudaraju s korisničkim upitom, zatim se prosljeđuju LLM-u kako bi se poboljšala točnost i preciznost rezultata.



Slika 45: Ponovno rangiranje (*eng. Re-ranking*)

Važno je napomenuti da su modeli za ponovno rangiranje obično sporiji u usporedbi s vektorskom pretragom [82]. Stoga se ne koriste u početnom koraku pronalaženja relevantnih dokumenata vezanih za korisnički upit kako bi se održala učinkovitost. Rangiranje se izvršava u fazi inferencije i koristi cross-encoder, gdje je reranking model ustvari cross-encoder. Cross-encoder je vrsta modela koja, za zadani par upita i dokumenta, daje rezultat sličnosti. Ovaj rezultat koristimo za preuređivanje dokumenata prema njihovoj relevantnosti za naš upit.

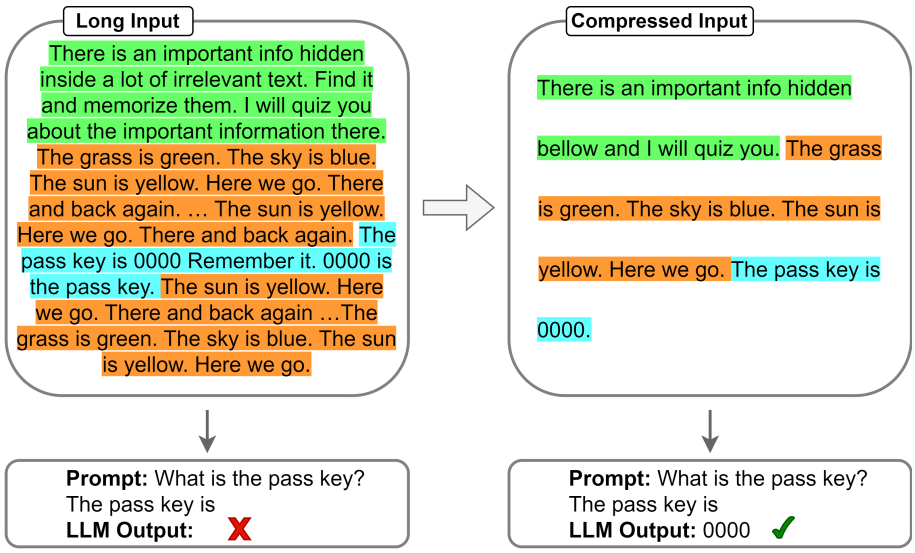
U RAG modelima, *bi-encoder* (*eng. bidirectional encoder*) se koristi u fazi pohrane relevantnih dokumenata iz vektorske baze podataka. Intuicija iza slabije točnosti *bi-encodera* je da ovi modeli moraju komprimirati sva moguća značenja dokumenta u jedan gusti vektor, što dovodi do gubitka informacija. Osim toga, *bi-encoders* modeli nemaju kontekst upita jer

ne znamo upit dok ga ne dobijemo (vektorske reprezentacije korpusa stvaramo prije nego što korisnik postavi upit).

S druge strane, *reranker* može primiti sirove informacije izravno za vrijeme samostalne obrade transformera, što znači manje gubitka informacija. Budući da *reranker* radi u trenutku kada korisnik postavlja upit, imamo dodatnu prednost analiziranja značenja dokumenta specifično za korisnički upit, umjesto pokušaja stvaranja generičkog, prosječnog značenja [83].

Reranker modeli uspješno izbjegavaju gubitak informacija koji je prisutan kod *bi-encodera*, ali imaju drugi nedostatak - vremensku zahtjevnost, što ih čini sporijima od procesa dohvaćanja.

2. Kompresija konteksta (*eng. Context compressing*): Kompresija konteksta jednostavna je tehnika koja omogućava učinkovitu integraciju velikih količina informacija unutar ograničenog kontekstualnog prozora LLM-a. Ovaj proces uključuje sažimanje ključnih dijelova dohvaćenih dokumenata, kako bi se zadržale najvažnije informacije dok se eliminiraju manje relevantni dijelovi. Primjer kompresije konteksta može biti sažimanje duljeg članka u nekoliko rečenica koje sažimaju glavne točke, čime se omogućava modelu da obradi veći broj relevantnih informacija unutar svojih ograničenja [84]. Slika 46 prikazuje semantičke kompresije upita.

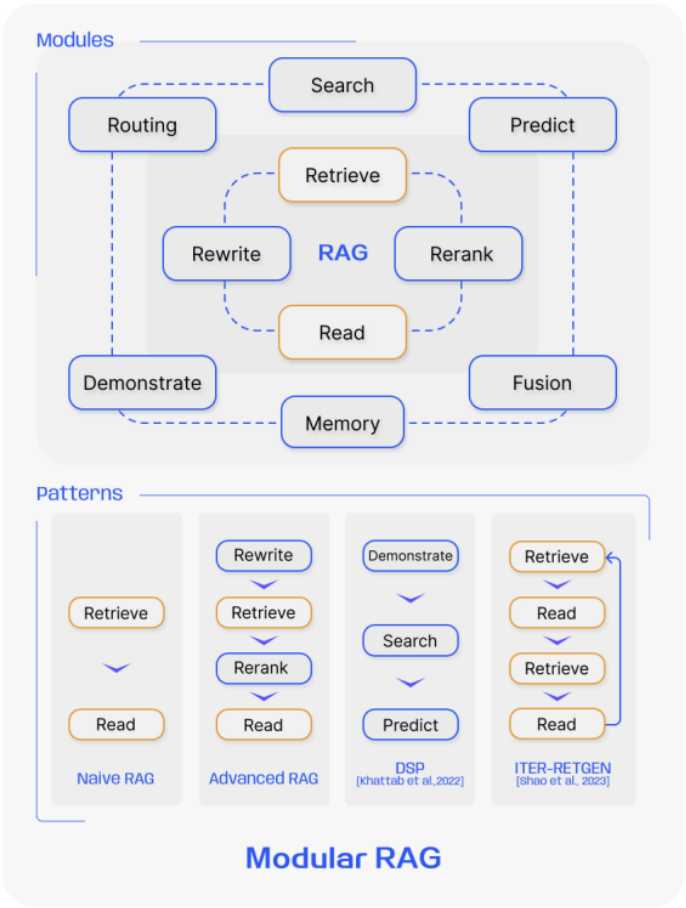


Slika 46: Primjer sintetičkog upita za zadatak dohvaćanja pristupnog ključa. Pretrenirani LLM nije sposoban obraditi dugi kontekst zbog ograničenja duljine konteksta. Primjenom semantičke kompresije konteksta, suvišne informacije u drugom dokumentu se uklanjaju, a kompresirani unos sadržava najbitnije informacije koje LLM lako obrađuje i generira točan odgovor.

4.5 Modularni RAG

Modularni RAG predstavlja napredak u odnosu na prethodne dvije paradigme RAG-a (osnovni i napredni), nudeći poboljšanja u prilagodljivosti i skalabilnosti. Ovaj napredni pristup uključuje raznolike strategije za unapređenje svojih komponenata, kao što su dodavanje modula za pretragu sličnosti i *finetuning retriever* modula. Inovacije poput restrukturiranja RAG modula i preuređenih arhitektura uvedene su kako bi se riješili specifični izazovi. Pomak prema modularnom RAG pristupu postaje sve raširenij, podržavajući sekvencijalnu obradu, kao i *end-to-end* treniranje svih njegovih komponenti. U modularnom RAG sustavu, i *retriver* i *generator* su dizajnirani tako da budu modularni, što znači da se mogu pojedinačno razvijati, optimizirati ili zamijeniti bez utjecaja na druge komponente, što omogućuje:

- **Fleksibilnost**: Različite metode pretraživanja ili izvori podataka mogu se lako testirati i zamijeniti.
- **Prilagodljivost**: Sustav se može prilagoditi specifičnim zadacima ili domenama podešavanjem modula.
- **Skalabilnost**: Svaki modul se može skalirati neovisno kako bi se nosio s većim skupovima podataka ili složenijim zadacima generiranja.



Slika 47: Modularni RAG

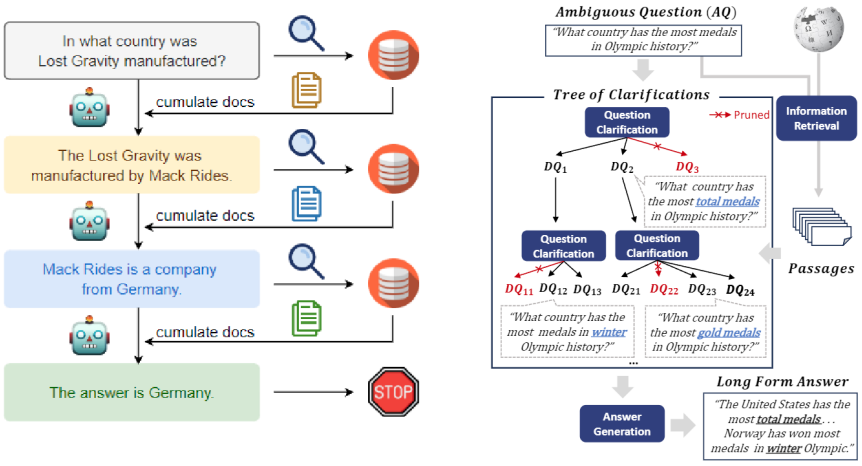
U RAG području, standardna praksa često uključuje singularni (jednokratni) korak preuzimanja informacija nakon kojeg slijedi generiranje, što može dovesti do neučinkovitosti i u pravilu je nedovoljno za složene probleme koji zahtijevaju razmišljanje u više koraka, jer pruža ograničen opseg informacija. Mnoge su studije optimizirale proces preuzimanja informacija kao odgovor na ovaj problem.

1. **Iterativno preuzimanje informacija (*eng. Iterative Retrieval*)** Iterativno preuzimanje informacija je proces u kojem se baza znanja više puta pretražuje na temelju početnog upita i dosadašnjeg generiranog teksta, pružajući LLM-ovima sveobuhvatniju bazu znanja. Ovaj pristup poboljšava robusnost generiranja odgovora jer omogućava dodatne kontekstualne reference kroz višestruke iteracije preuzimanja. Ipak, pristup je ranjiv prilikom pojavljivanja semantičkog diskontinuiteta i akumulacije irelevantnih informacija. Na primjer, model ITERRETGEN koristi sinergijski pristup koji kombinira *retrieval-enhanced generation* i *generation-enhanced retrieval* kako bi učinkovito obradio zadatke koji zahtijevaju preciznu reprodukciju specifičnih informacija [85]. Na ovaj način, odgovor na zadatak pruža ključne informacije potrebne za njegovo uspješno dovršavanje, omogućujući prepoznavanje relevantnog znanja. To, zauzvrat, omogućava generiranje poboljšanih odgovora u narednim iteracijama. Koristeći te informacije kao osnovu za preuzimanje relevantnog znanja, postiže se sve precizniji i kvalitetniji odgovor kroz svaku sljedeću iteraciju.

2. **Rekurzivno preuzimanje informacija (*eng. Recursive Retrieval*)**
   Rekurzivno preuzimanje informacija često se koristi u pretraživanju informacija i općenito NLP-u, kako bi se poboljšala dubina i relevantnost rezultata pretraživanja. Proces uključuje iterativno usavršavanje pretraživačkih upita na temelju rezultata dobivenih iz prethodnih pretraga. Cilj rekurzivnog preuzimanja je poboljšati iskustvo pretraživanja postupnim približavanjem najrelevantnijim informacijama kroz povratnu petlju.

   - IRCoT koristi *chain-of-thought* tehniku za vođenje procesa preuzimanja i usavršava lanac razmišljanja s dobivenim rezultatima preuzimanja [86].
   - ToC (*eng. Tree of Clarification*) stvara stablo pojašnjenja (*eng. Clarification tree*) koje sustavno optimizira nejasne dijelove upita. Ovaj proces omogućava kontinuirano učenje i prilagodbu korisnikovim potrebama, često rezultirajući poboljšanim zadovoljstvom rezultatima pretraživanja [87].

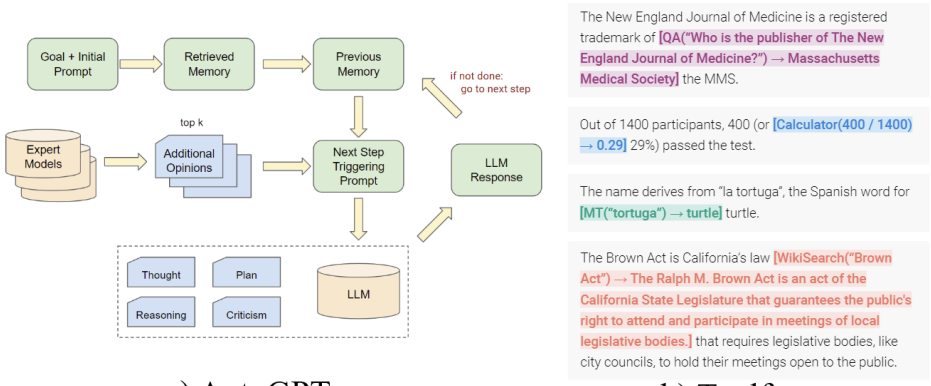
   Primjeri ovih tehnika (IRCoT lijevo, ToC desno) prikazani su na slici 48.



a) IRCoT b) ToC

Slika 48: a) IRCot, b) ToC

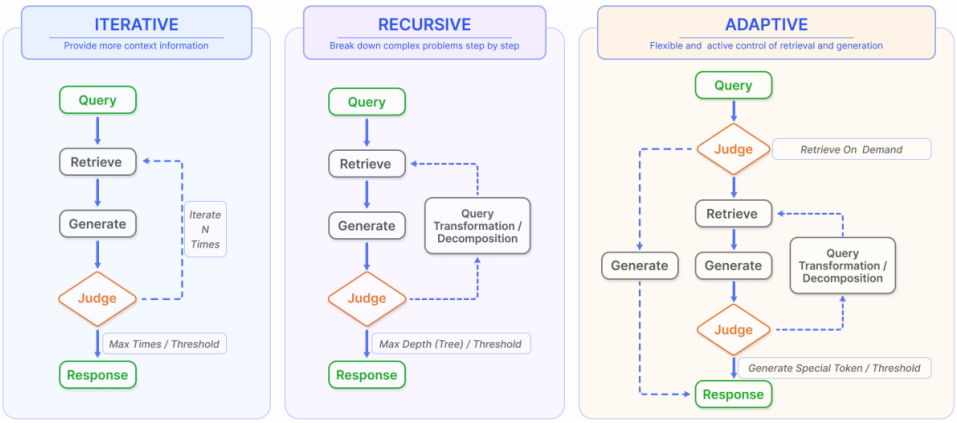
3. **Adaptivno preuzimanje informacija (*eng. Adaptive Retrieval*)**
   Adaptivne metode prikupljanja informacija, poput modela Flare i Self-RAG, poboljšavaju RAG okvir omogućujući velikim jezičnim modelima (LLM-ovima) da aktivno odrede optimalne trenutke i sadržaj za preuzimanje, čime se povećava učinkovitost i relevantnost prikupljenih podataka. Ove metode odražavaju širi trend u kojem LLM-ovi koriste aktivno prosuđivanje tijekom rada, što se može vidjeti kod agenata poput AutoGPT-a, Toolformera i Graph-Toolformera. Na primjer, WebGPT integrira okvir za učenje pojačanjem kako bi obučio GPT-3 model da autonomno koristi pretraživač tijekom generiranja teksta [88]. Flare automatizira vrijeme preuzimanja praćenjem povjerenja u proces generiranja, aktivirajući sustav preuzimanja kada povjerenje padne ispod određenog praga. Self-RAG uvodi "refleksijske tokene" koji omogućuju modelu introspekciju svojih izlaza, autonomno odlučujući kada aktivirati preuzimanje ili postavljajući unaprijed definirani prag za pokretanje procesa [89]. Slika 49 prikazuje metodologiju AutoGPT i primjer rada Toolformer tehnike.



a) AutoGPT b) Toolformer

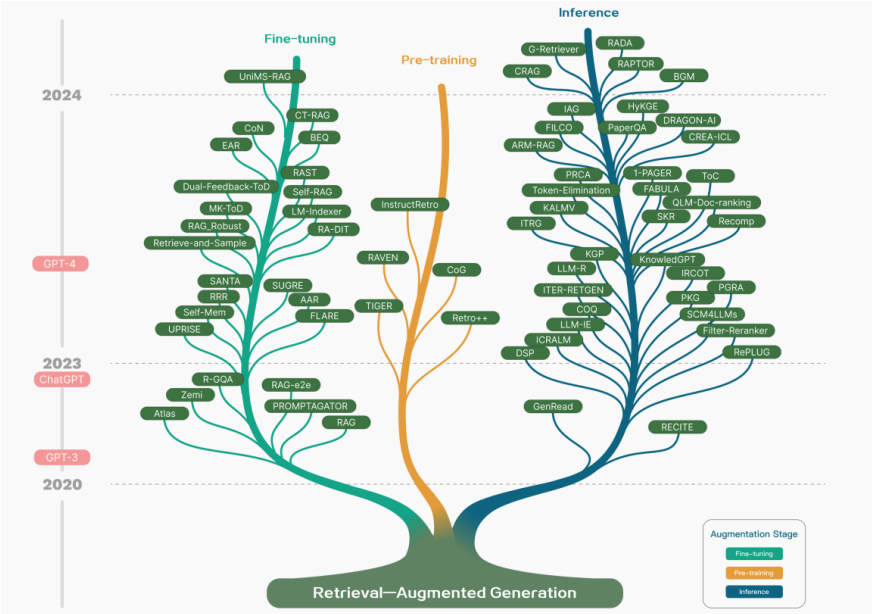
Slika 49: a) AutoGPT, b) Toolformer. Kod Toolformer tehnike može se vidjeti pozivanje raznih modula kroz sam proces (QA, Calculator, MT, WikiSearch)

Iterativni, rekurzivni i adaptivni pristupi preuzimanju informacija (Slika 50) svaki na svoj način optimiziraju proces prikupljanja podataka. Iterativni pristup omogućava ponovljene iteracije kako bi se postigli bolji rezultati, rekurzivni se usredotočuje na usavršavanje upita kroz povratne petlje, dok se adaptivni pristup prilagođava trenutnim potrebama modela kako bi optimizirao trenutke preuzimanja podataka.



Slika 50: Procesi augmentacije (1. Iterativni, 2. Rekurzivni, 3. Adaptivni)

Razvoj tehnika RAG može se podijeliti u tri faze, ovisno o tome kada se odvija proces augmentacije: *pre-training*, *fine-tuning* i *inference*. U ranim danima RAG-a 2020. godine, kao što je opisano u poglavlju 4.1, znanstvenici su se prvenstveno bavili fazom pretreniranja. S pojavom ChatGPT-a i općim napretkom velikih jezičnih modela, fokus razvoja RAG tehnika prebacio se na fazu *inference* i istraživanje kako iskoristiti snažne kontekstualne značajke koje pružaju moderni LLM-ovi. Ipak, nova istraživanja aktivno istražuju i integraciju *fine-tuning* tehnika. Slika 51 prikazuje tehnološko stablo razvoja RAG istraživanja u razdoblju od 2020. do 2024. godine [58].

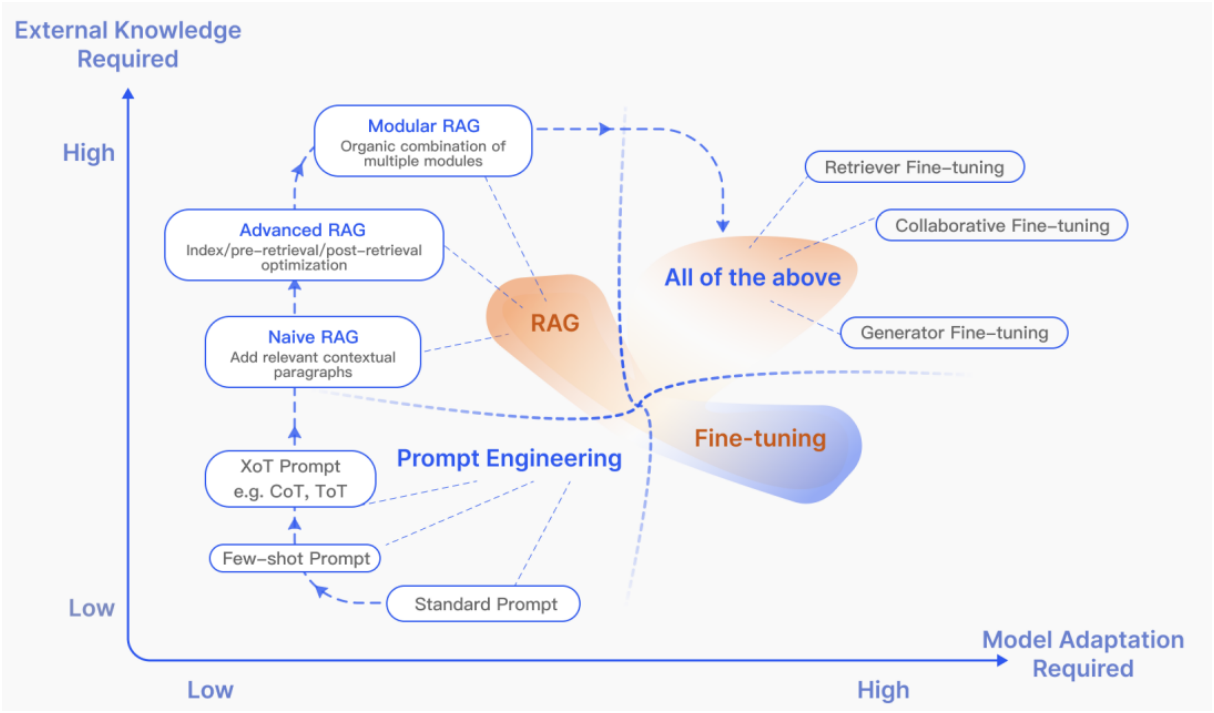


Slika 51: Tehnološko stablo istraživanja RAG područja

4.5.1 RAG vs Finetuning

Augmentacija velikih jezičnih modela privukla je značajnu pozornost zbog njihove sve veće rasprostranjenosti. Među metodama optimizacije LLM-ova, RAG se često uspoređuje s *finetuningom* i *prompt inženjeringom*. Svaka metoda ima različite karakteristike, kako je prikazano na Slici 52. Ilustracija prikazuje grafikon podijeljen u 4 kvadranta kako bi se ilustrirale razlike u trima metodama (*prompt engineering*, RAG i *finetuning*):

- **Zahtjevi za vanjskim znanjem na *Y* osi** prikazuju koliko model treba dodatnih informacija iz vanjskih izvora za točne odgovore. Pri dnu *Y* osi (blizu 0) ti zahtjevi su niski, dok su pri vrhu (npr. kod modularnog RAG-a) vrlo visoki, što znači da model treba značajnu količinu vanjskih podataka za učinkovito funkcioniranje.
- **Zahtjevi za prilagodbom modela na *X* osi** prikazuju koliko je model potrebno prilagoditi; prema 0 ti su zahtjevi vrlo niski, dok su kod *finetuninga* na krajnjem dijelu osi vrlo visoki.



Slika 52: Usporedba RAG metodologije s *prompt engineering* i *finetuning* pristupima [58]

Prompt inženjering koristi sposobnosti modela s minimalnom potrebom za vanjskim znanjem i prilagodbom. RAG se može usporediti s pružanjem modela prilagođenim udžbenikom za preuzimanje informacija, idealnim za precizno preuzimanje podataka. Nasuprot tome, **fine-tuning** je poput učenika koji s vremenom usvaja znanje, pogodan za reproduciranje specifičnih struktura, stilova ili formata. **RAG** se ističe u dinamičkim okruženjima omogućujući ažuriranje znanja u stvarnom vremenu i učinkovito korištenje vanjskih izvora s visokom interpretabilnošću, ali dolazi s većom latencijom i etičkim pitanjima u vezi prikupljanja podataka. *Finetuning* je

statičan, zahtijeva ponovno treniranje za ažuriranja, ali omogućuje duboku prilagodbu ponašanja i stila modela. Također zahtijeva značajne računalne resurse i može smanjiti halucinacije, ali se suočava s izazovima pri radu s nepoznatim podacima [58].

4.5.2 RAG evaluacija

U literaturi postoji niz referentnih testova i alata razvijenih u svrhu lakše evaluacije RAG modela. Ovi alati pružaju kvantitativne metrike koje, osim što procjenjuju performanse RAG modela, poboljšavaju razumijevanje sposobnosti modela kroz razne aspekte evaluacije. Istaknute *benchmark* metrike poput RGB, RECALL i CRUD primarno se fokusiraju na procjenu ključnih sposobnosti RAG modela, dok alati poput RAGAS, ARES i TruLens8 koriste LLM-ove za ocjenjivanje kvalitativnih rezultata [58]. Spomenuti alati i *benchmark* testovi, te njihove primjene, mogu se vidjeti u Tablici 3.

Alat	**Predmet evaluacije**	**Aspekti**	**Kvantitativne metrike**
RGB †	Retrieval Quality Generation Quality	Noise Robustness Negative Rejection Information Integration Counterfactual Robustness	Accuracy EM Accuracy Accuracy
RECALL †	Generation Quality	Counterfactual Robustness	R-Rate
RAGAS ‡	Retrieval Quality Generation Quality	Context Relevance Faithfulness Answer Relevance	* * Cosine Similarity
ARES ‡	Retrieval Quality Generation Quality	Context Relevance Faithfulness Answer Relevance	Accuracy Accuracy Accuracy
TruLens ‡	Retrieval Quality Generation Quality	Context Relevance Faithfulness Answer Relevance	* * *
CRUD †	Retrieval Quality Generation Quality	Creative Generation Knowledge-intensive QA Error Correction Summarization	BLUE ROUGE-L BertScore RAGQuestEval

Tablica 3: Usporedba evaluacijskih metrika za RAG modele: simbolom † se označava *benchmark*; ‡ predstavlja alate; zvjezdicom (*) su označene prilagođene kvantitativne metrike, koje se razlikuju od onih standardnih.

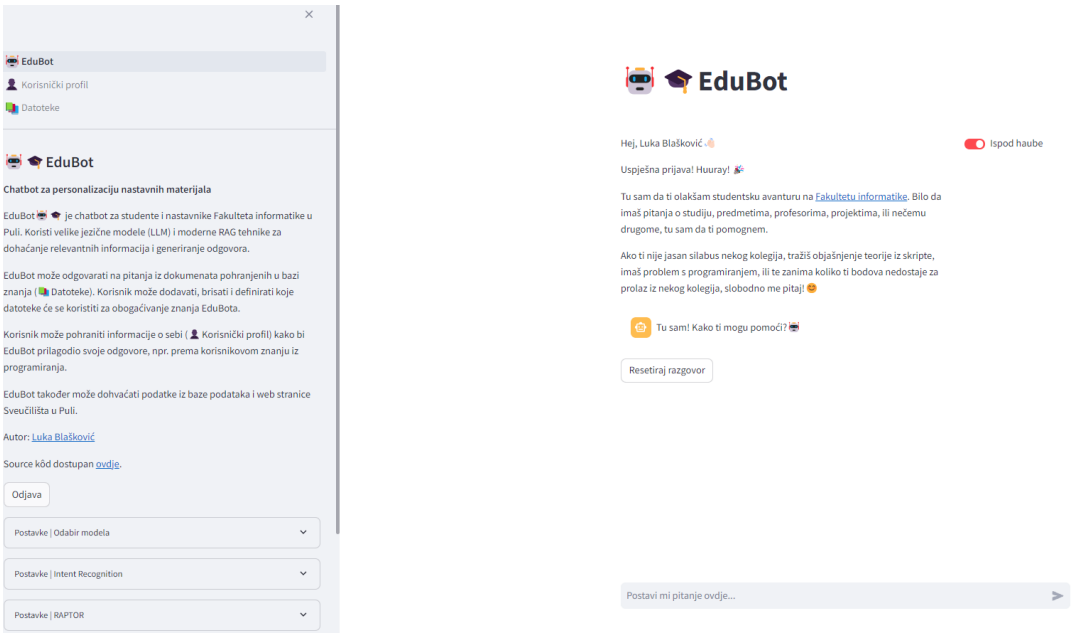
5 Implementacija

EduBot je napredni konverzacijski agent koji pruža personalizirani pristup učenju svakom studentu i odgovara na razna akademska pitanja. Opremljen velikim jezičnim modelima i modernim RAG tehnikama, EduBot pristupa informacijama iz raznih vanjskih izvora podataka. Ovi izvori uključuju SQL baze podataka, učitane nastavne materijale poput studentskih skripti, nastavnih planova, raznih pravilnika i knjiga, kao i podatke dohvaćene s web stranica Sveučilišta Jurja Dobile i Fakulteta informatike. Aplikacija je razvijena u Pythonu, koristeći Streamlit biblioteku za izradu frontend komponenti i pokretanje lokalnog poslužitelja. Svaki izvor podataka ima svoj RAG modul koji dohvaća relevantne informacije i prosljeđuje ih velikom jezičnom modelu (LLM) na daljnju obradu. Korisnik ne mora birati koji će se RAG modul koristiti jer to automatski određuje semantički usmjerivač (*eng. semantic router*). Usmjerivač prepoznaje namjeru korisnika i na temelju toga upit prosljeđuje odgovarajućem RAG modulu na daljnju obradu.

Izvorni kôd aplikacije dostupan je javno na GitHub repozitoriju:
https://github.com/lukablaskovic/edu_bot

5.1 Streamlit

Streamlit je Python biblioteka otvorenog kôda koja omogućuje jednostavnu izradu web aplikacija s grafičkim sučeljem [90]. Namijenjen je prvenstveno stručnjacima u području podatkovne znanosti i strojnog učenja, pružajući im alat za brzo i efikasno vizualiziranje i interaktivno istraživanje podataka. Streamlit omogućuje razvojnim programerima da brzo izgrade interaktivne web aplikacije kroz nekoliko linija kôda, koristeći samo Python, bez potrebe za opsežnim znanjem HTML-a, CSS-a ili JavaScript-a. Slika 53 prikazuje početno sučelje EduBot aplikacije.



Slika 53: EduBot GUI - početna stranica

Grafičko sučelje aplikacije sastoji se od dva glavna dijela:

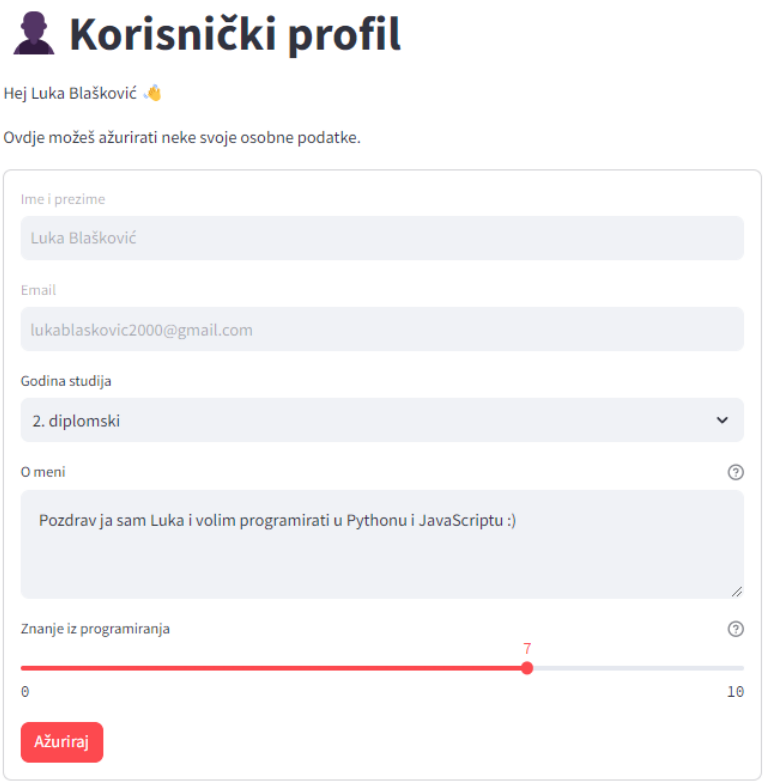
1. **Chatbot sučelje**: Klasično *chatbot* sučelje koje omogućava vođenje razgovora s konverzacijskim agentom. Korisnik postavlja pitanja, a agent ispisuje odgovore ispod svakog postavljenog pitanja.

2. **Bočni izbornik**: Izbornik koji sadrži upute za korištenje, poveznice na tri podstranice (1. EduBot, 2. Korisnički profil, 3. Datoteke) te postavke prikazane u obliku padajućih izbornika.

EduBot koristi **Streamlit Google Auth** biblioteku kako bi omogućio jednostavnu integraciju Google autentifikacije [91]. Za lakše dijeljenje varijabli između modula i stranica unutar aplikacije koristi se Streamlitov **Session State API**, koji omogućava pohranu varijabli u obliku ključ-parova, čime se stvara privremena memorija tijekom korištenja aplikacije.

5.2 Elementi personalizacije

Kako bi se omogućila personalizacija unutar aplikacije, korisnici se prvo trebaju autentificirati putem Google OAuth 2.0 servisa. Proces autentifikacije je jednostavan - korisnici se prijavljuju svojim Google računom i time započinju korištenje aplikacije. Nakon uspješne prijave, aplikacija pohranjuje korisnikovu e-mail adresu te ime i prezime preuzeto s Google računa. Korisnici mogu pristupiti svom **Korisničkom profilu** putem bočnog izbornika, gdje mogu ažurirati svoje osobne informacije. Slika 54 prikazuje sučelje korisničkog profila.

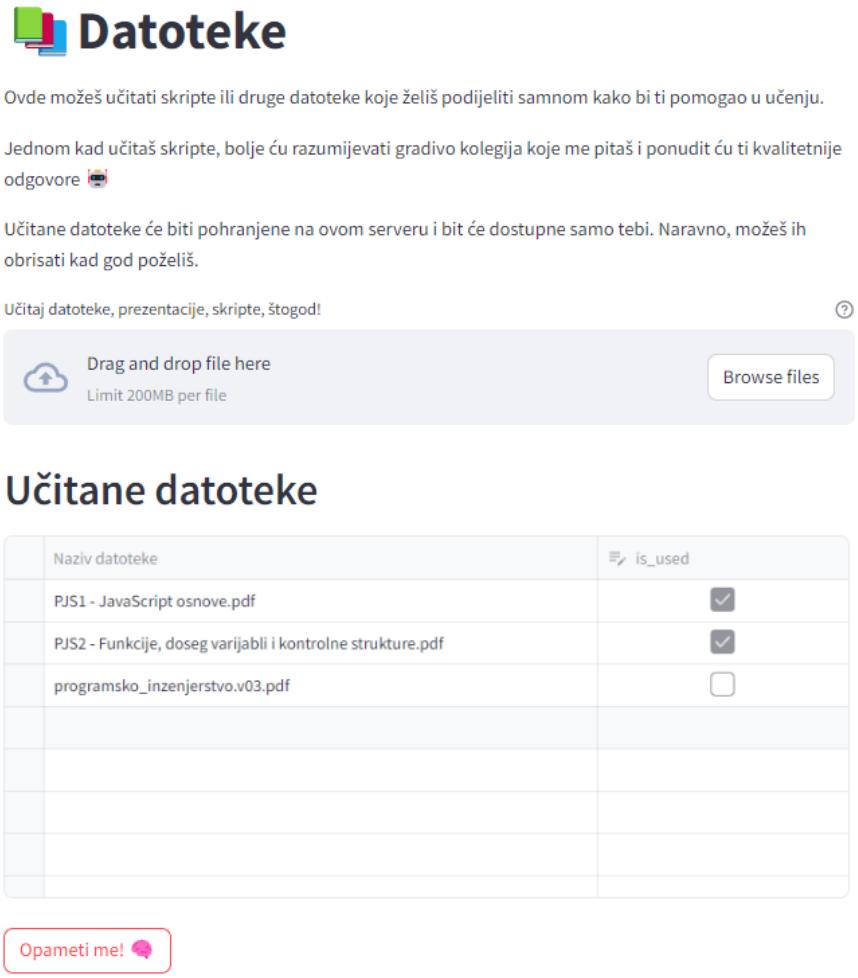


Slika 54: Edubot GUI - Korisnički profil

Korisnici mogu odabrati godinu studija, napisati kratak opis o sebi te procijeniti svoje znanje iz programiranja pomoću klizača s rasponom od 0 do 10. EduBot koristi ove informacije kako bi prilagodio svoje odgovore na tehnička pitanja, pružajući korisnicima relevantne i precizne savjete temeljene na njihovom definiranim znanju.

Dalje, korisnici mogu putem stranice *Datoteke* učitati vlastite nastavne materijale poput studentskih skripti, knjiga, znanstvenih članaka, programskog kôda i slično. Učitane datoteke pohranjuju se lokalno u datotečni sustav same aplikacije. Korisnici mogu pohraniti više datoteka te obrisati postojeće. Nakon toga, mogu odabrati koje će se datoteke koristiti za potrebe RAG dohvaćanja i pokrenuti postupak indeksiranja.

Na Slici 55 prikazane su učitane datoteke: *"PJS1 - JavaScript osnove.pdf"*, *"PJS2 - Funkcije, doseg varijabli i kontrolne strukture.pdf"* i *"programsko_inzenjerstvo.v03.pdf"*. Od te tri, samo posljednja datoteka nije odabrana za indeksiranje.



Slika 55: EduBot GUI - Učitavanje datoteka

Kada korisnik učita datoteke, može stvoriti RAPTOR stablo znanja na temelju tih podataka. RAPTOR je ključna RAG tehnika koja služi kao temelj za EduBot aplikaciju. Više informacija o tome možete pronaći u sljedećem poglavlju.

5.3 RAG moduli

Implementirana su ukupno tri RAG modula za potrebe dohvaćanja raznih informacija:

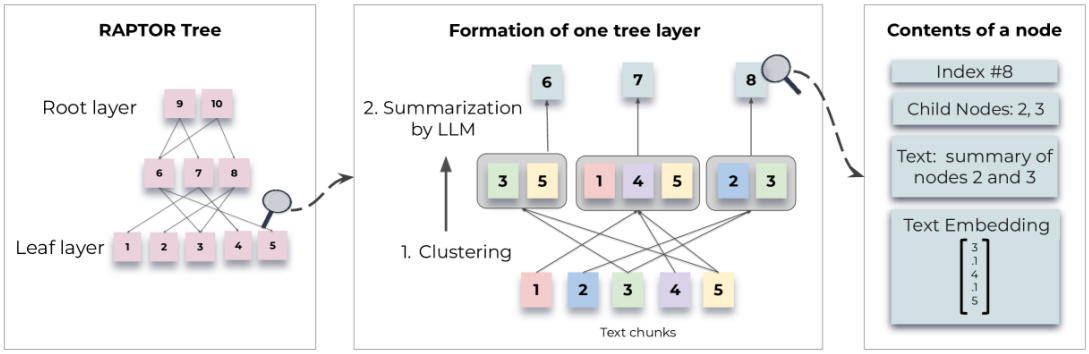
1. **RAPTOR**: Ovaj modul koristi RAPTOR tehniku rekurzivnog dohvaćanja kroz hijerarhijsku stablastu strukturu, što omogućuje indeksiranje tekstualnih datoteka koje korisnik učita putem stranice Datoteke.

2. **text-to-SQL**: Ovaj modul omogućava dohvaćanje podataka iz baze podataka generiranjem odgovarajućeg SQL upita temeljenog na korisničkom upitu.

3. **web-scraper**: Koristi se za prikupljanje najnovijih informacija sa web stranica Sveučilišta Jurja Dobrile u Puli ili Fakulteta informatike.

5.3.1 RAPTOR modul

RAPTOR (*eng. Recursive Abstractive Processing for Tree-Organized Retrieval*) je inovativna tehnika za dohvaćanje informacija koja koristi princip izrade stabla rekurzivnim ugrađivanjem, grupiranjem i sažimanjem tekstualnih segmenata, čime omogućuje dohvaćanje informacija na različitim razinama apstrakcije [92]. Ova metoda rekurzivno gradi stablo grupiranjem (*eng. clustering*) tekstualnih segmenata na temelju vektorskih reprezentacija, postupno generirajući sažetke viših razina za te segmente. Primjenjuje *bottom-up* pristup, tako da se stablo gradi od listova (*eng. leaf nodes*) prema korijenu (*eng. root node*). Čvorovi koji su grupirani zajedno nazivaju se braćom (*eng. siblings*), dok roditelj čvora sadrži tekstualni sažetak svoje grupe. Na slici 56 prikazan je opisani postupak.



Slika 56: Proces izgradnje RAPTOR stabla

Algoritam klasteriranja (*eng. Clustering Algorithm*) Proces izrade RAPTOR stabla započinje segmentiranjem korpusa podataka na manje dijelove teksta veličine 100 tokena. Ako broj premaši tu veličinu, cijela rečenica se prenosi u sljedeći segment (*eng. chunk*), umjesto da se prekida u sredini. Time se osigurava očuvanje semantike i koherencije teksta unutar svakog segmenta. U originalnoj RAPTOR implementaciji, ovi segmenti se kodiraju koristeći

varijantu SBERT modela. Međutim, u EduBot implementaciji, za brže izvođenje mogu se koristiti *text-embedding-3* modeli tvrtke OpenAI.

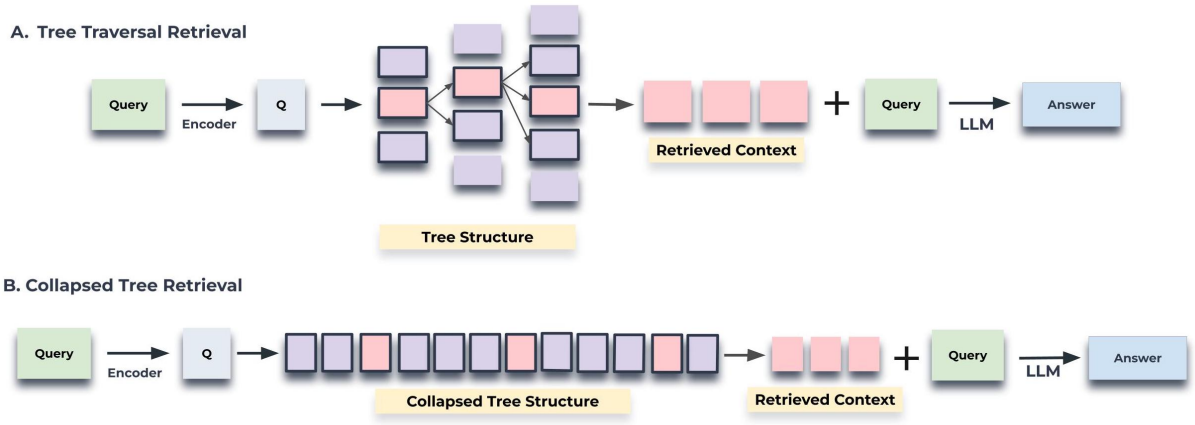
Algoritam grupiranja ključan je za RAPTOR implementaciju jer omogućuje organizaciju segmenata u grupe, pri čemu su sekvencijalno poredane grupe povezane jedna s drugom. Ovaj algoritam temelji se na *Gaussian Mixture Models* (GMMs) algoritmu grupiranja koji omogućuje da se isti čvorovi nalaze u više različitih grupa (*eng. clusters*). Ova fleksibilnost je važna jer individualni segmenti teksta često sadrže informacije relevantne za različite teme, čime se osigurava njihovo uključivanje u više sažetaka.

Jednom kada se segmenti sažetaka čvorova prve razine (originalnih dijelova teksta) izrade, oni se rekurzivno kodiraju i grupiraju na višim razinama, koje sada sadrže apstraktnije dijelove informacija. Na taj način, korijen stabla sadrži sažetak cijelog dokumenta.

RAPTOR *pipeline* : Kao kod osnovnog RAG principa, proces započinje postavljanjem upita korisnika koji se kodira u vektorsku reprezentaciju kako bi se izračunala sličnost s kodiranim dokumentima. U RAPTOR-u, ovaj proces pretraživanja (*retrieval*) može se provesti na dva načina, kao što je i prikazano na slici 57.

1. **Tree Traversal Retrieval**: Obilazak stabla započinje od korijena i dohvaća *top-k* čvorova na temelju izračuna *cosine similarity* s vektorom upita na svakom sloju. Dakle, ako je *k=2*, dohvaćaju se po dva čvora na svakom sloju (razini) počevši od korijena, zatim se dohvaćaju dva čvora na sljedećoj razini, koji su njegova djeca, itd.

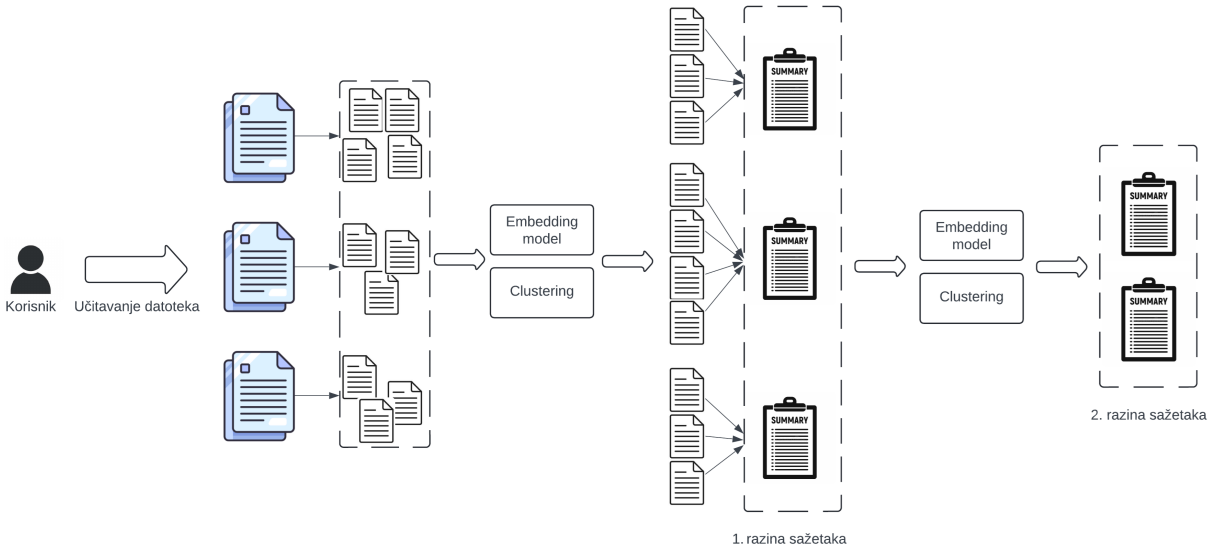
2. **Collapsed Tree Retrieval**: Kod ove metode, stablo se svodi na sekvencijalni niz čvorova iste razine, čime se dohvaćanje znatno pojednostavljuje. Nedostatak ove metode je što se *cosine similarity* mora izračunati za sve čvorove s obzirom na dani upit.



Slika 57: Ilustracija metoda pretraživanja RAPTOR stabla: A. Tree Traversal Retrieval, B. Collapsed Tree Retrieval

Jednom kada korisnik učita datoteke koje će se koristiti, započinje proces izrade RAPTOR stabla. Vektorske reprezentacije pohranjuju se u Chroma vektorsku bazu podataka, dok RAPTOR dodatno sprema binarnu datoteku koja opisuje odnose između čvorova. Kada korisnik započne interakciju s EduBotom, provjerava se postoji li već pohranjeno RAPTOR stablo znanja i odgovarajuće vektorske reprezentacije. Ako stablo već postoji i nije bilo promjena u datotekama ili hiperparametrima RAPTOR-a, koristi se postojeće stablo. U suprotnom, stablo se izgrađuje ponovno. Proces učitavanja datoteka i izrade RAPTOR stabla prikazan je na slici 58.

U postavkama EduBota korisnik može odabrati metodu dohvaćanja podataka: *Tree Traversal* ili *Collapsed Tree*. Također može postaviti top-k hiperparametar (broj najrelevantnijih *clustera* koji će se pretraživati u svakom sloju) te odabrati *embedding* model za izradu semantičkih vektorskih reprezentacija.



Slika 58: Učitavanje datoteka i izrada RAPTOR stabla

5.3.2 *text-to-SQL* modul

Ovaj modul je namijenjen za dohvat podataka pohranjenih u SQL bazi. Koristi SQLite bazu podataka koja se lokalno izvršava na računalu i sastoji se od tablica *korisnik* i *PJS_bodovi*. U Tablici *korisnik* nalaze se podaci o korisnicima, uključujući one povučene s Google računa kao i podatke koje korisnik sam unese. Tablica *PJS_bodovi* sadrži informacije o bodovima iz kolegija Programiranje u skriptnim jezicima, s umjetno unesenim podacima za nekoliko studenata, bodovima iz pet ispita te dodatnim stupcem za ukupne bodove.

Text-to-SQL RAG modul dizajniran je za precizno dohvaćanje podataka iz baze podataka i pružanje točnih odgovora korisnicima. Proces uključuje sljedeće korake:

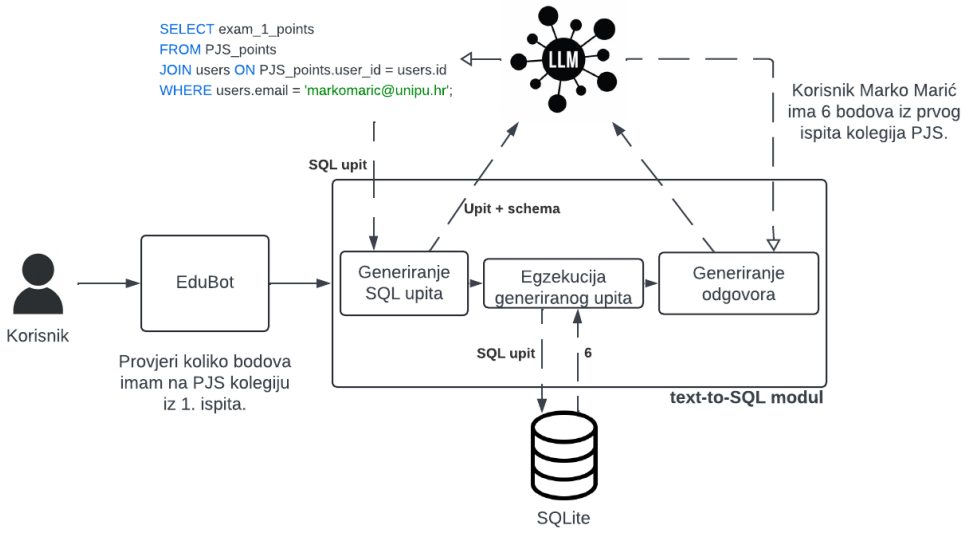
1. **Primanje korisničkog upita**: Korisnik postavlja upit, primjerice: "Provjeri koliko bodova imam na PJS kolegiju iz 1. ispita."

2. **Generacija odgovarajućeg SQL upita**: Sljedeći korak uključuje generiranje odgovarajućeg SQL upita pomoću LLM-a i to na temelju strukture relacijske tablice i korisničkog upita. Korisnik može u postavkama EduBota odabrati koje tablice će se koristiti za dohvaćanje podataka, čime se proširuje ili sužava kontekst za LLM. Primjer *prompta* prikazan je u nastavku: nakon definiranja persone i davanja uputa za generiranje SQL upita, u kontekst se dodaje struktura tablica *(schemas_str)* u obliku "SQL CREATE" naredbe te korisnički upit *(query_str)*.

```
f"Context information is below.\n"
f"---------------------\n"
f"You are a professional SQL developer. You are given a task to
write a SQL query to retrieve data from the database.\n"
f"You must ALWAYS return SQL query only and nothing else.\n"
f"---------------------\n"
f"{schemas_str}\n"
f"---------------------\n"
f"Given the database schemas and example rows,
structure the SQL query from given User prompt\n"
f"User prompt: {{query_str}}\n"
```

3. **Izvršavanje SQL upita**: Generirani upit iz prethodnog koraka se izvršava. Ako je upit točan, baza podataka vraća tražene informacije.

4. **Generiranje konačnog odgovora**: Završni korak, generacijski dio svakog RAG modula, uključuje ponovno korištenje LLM-a s kontekstom početnog korisničkog upita i rezultata dobivenog iz baze podataka. Cilj je integrirati informacije u čovjeku prirodan odgovor. Na primjer: "Korisnik Marko Marić ima 6 bodova iz prvog ispita kolegija PJS."



Slika 59: Opisani postupak rada SQL RAG modula

5.3.3 *web scraper* modul

Web scraper modul implementiran je pomoću *BeautifulSoup* Python biblioteke, koja omogućava preuzimanje podataka iz HTML i XML struktura (tzv. *web scraping*) [93]. Korisnik može odabrati želi li povući podatke sa stranica novosti Sveučilišta Jurja Dobrile ili Fakulteta informatike, te postaviti maksimalni broj objava koje želi pročitati. Nakon toga, šalju se GET zahtjevi koristeći *requests* biblioteku te se vraća HTML struktura web stranice koja se dalje obrađuje pomoću *BeautifulSoup* parsera. Dobivene objave koriste se kao kontekst jednostavnog RAG modula koji generira tekst na temelju ekstrahiranih podataka. Podaci uključuju: naslov objave, poveznicu, datum objave, autora, sliku i sažetak.

5.4 Semantičko usmjeravanje upita

Odabir modula izvršava se automatski na temelju korisničkog upita. To se postiže kroz semantički usmjerivač (*eng. semantic router*) koji je implementiran kao zasebna komponenta u aplikaciji. Ova komponenta koristi LlamaIndex **Router Query Engine**. LlamaIndex je razvojni okvir (*eng. framework*) dizajniran za jednostavniji i brži razvoj aplikacija temeljenih na velikim jezičnim modelima. Pruža gotove komponente za razvoj konverzacijskih agenata, multimodalnih aplikacija, RAG sustava i drugih sličnih rješenja [94]. U LlamaIndexu, **Query Engine** predstavlja osnovno sučelje koje omogućuje postavljanje pitanja o podacima putem prirodnog jezika. Možemo ga zamisliti kao osnovnu RAG komponentu na koju možemo graditi složenije module. Ova komponenta korištena je u implementaciji triju spomenutih RAG modula, međutim, oni se razlikuju po svojim principima rada. Primjerice, RAPTOR komponenta koristi poseban *RAPTOR retriever* i vlastiti algoritam indeksiranja podataka, u usporedbi sa *SQL RAG* i *web scraper* komponentama.

Router Query Engine je komponenta koja agregira tri modula i, na temelju korisničkog upita, odabire jedan od njih. Svaki modul ima definirane metapodatke koji uključuju naziv modula, *query engine* i detaljni *use-case* opis. Ako ova komponenta utvrdi da dani upit nije prikladan za niti jedan od modula, proslijedit će upit direktno LLM-u, bez RAG utilizacije.

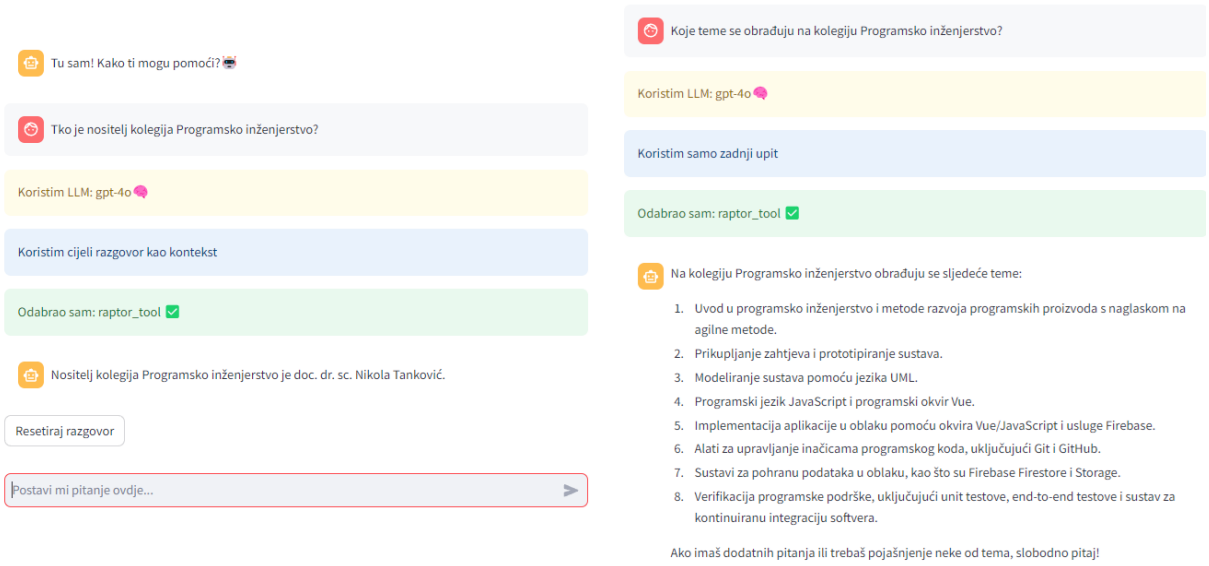
Primjerice, *use-case* opis za RAPTOR *query engine* izgleda ovako:

```
Useful for retrieving specific context about the faculty,
syllabus, and course information such as:
- syllabus information (how to access exam, what to expect from class, etc.)
- course information (course schedule, course materials, etc.)
- programming knowledge (programming languages, tools, paradigms, functions)
- specific questions about programming (how to debug or how to write a program)

Query: {query}
```

U nastavku su prikazani neki isječci razgovora s EduBotom. Dodatna opcija "Ispod haube" korisnicima pruža vizualni uvid u aktivnosti EduBota u pozadini:

Slika 60 prikazuje razgovor s Edubotom, gdje korisnik postavlja pitanja koja zahtijevaju RAPTOR RAG modul za dohvaćanje informacija iz RAPTOR stabla znanja. U ovom kontekstu, stablo je izgrađeno pomoću tri datoteke: dvije studentske skripte iz JavaScripta, koje nisu povezane s pitanjem, te silabusom kolegija - dokument iz kojeg očekujemo informacije.

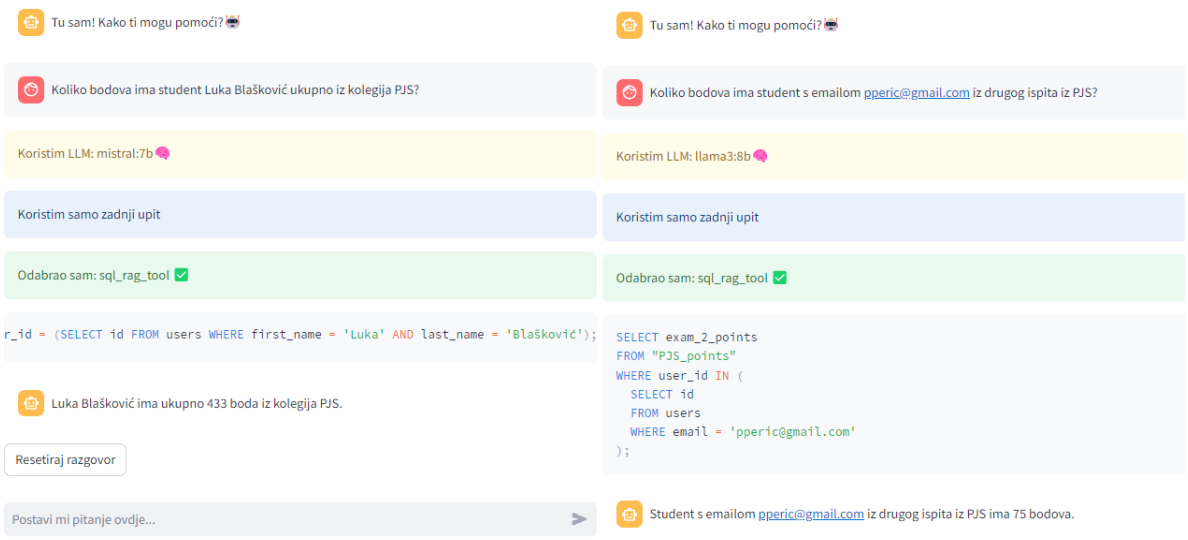


Slika 60: EduBot interakcija - RAPTOR modul

Slika 61 prikazuje interakciju gdje korisnik postavlja pitanje: "Objasni mi koncept rekurzije u JavaScriptu.". Nakon što EduBot dohvati relevantne segmente iz učitane JavaScript skripte, dodatno prilagođava odgovor u fazi generacije ovisno o predznanju korisnika.

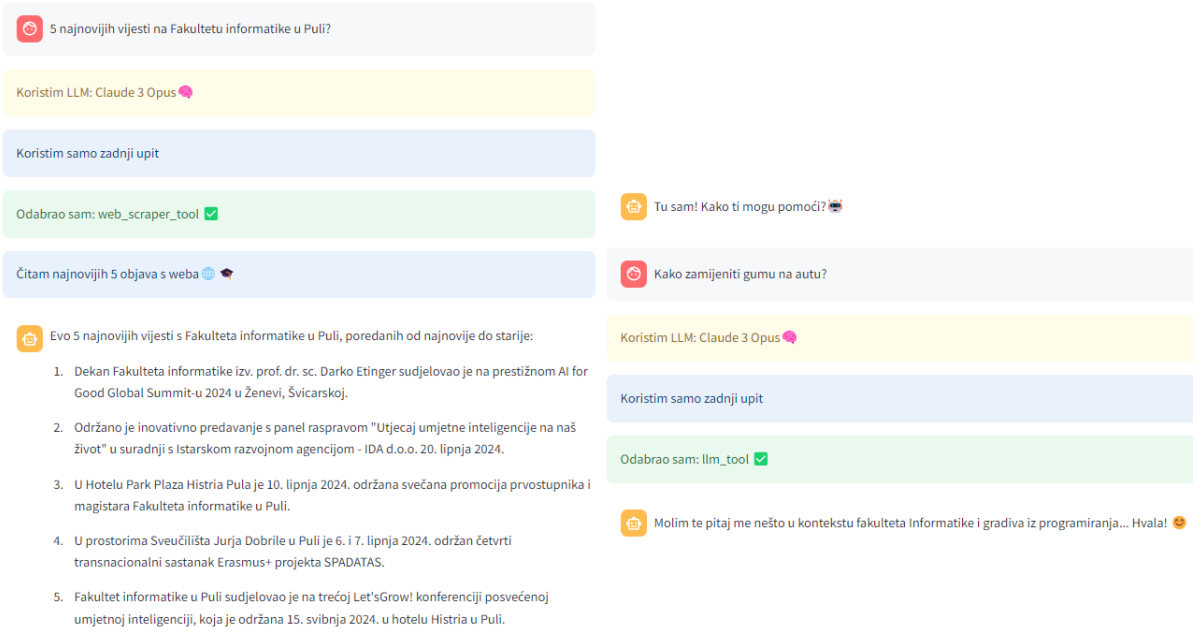
Slika 61: EduBot interakcija - RAPTOR modul: Prilagodba odgovora ovisno o predznanju

Slika 62 prikazuje interakciju u kojoj *chatbot* prepoznaje potrebu za korištenjem SQL RAG modula. Kada korisnik postavi pitanje o bodovima na kolegiju, *text-to-SQL* modul generira odgovarajući upit, izvršava ga i prikazuje odgovor korisniku.



Slika 62: EduBot interakcija - SQL RAG modul

Na slici 63 prikazana je interakcija u kojoj korisnik traži najnovije vijesti s fakulteta, a za to se koristi RAG modul *web scraper*. Desna strana prikazuje isječak razgovora u kojoj korisnik postavlja pitanje koje nije povezano s tematikom chatbota. U tom slučaju, modul se ne aktivira, već se upit automatski prosljeđuje LLM-u za obradu.



Slika 63: EduBot interakcija - Web scraper modul (lijevo) i direktni LLM (desno)

6 Evaluacija

Provedena je sveobuhvatna evaluacija kvalitete odgovora EduBot konverzacijskog agenta koristeći skup od 20 pitanja koja se odnose na nastavno gradivo na fakultetu, sveučilišne pravilnike, izvedbene planove nastave za kolegije Programsko inženjerstvo i Upravljanje poslovnim procesima, status bodova na kolegijima te novosti sa sveučilišta. Pitanja je definirao autor prema informacijama iz navedenih dokumenata, SQL baze podataka i web mjesta. Izravnim pristupom velikim jezičnim modelima nije moguće dobiti odgovore na navedena pitanja, već je potrebna primjena jedne od tri RAG tehnike. Zbog toga se provodi evaluacija konverzacijskog agenta koristeći sedam različitih LLM-ova kako bi se testirala kvaliteta RAG tehnika s manjim modelima.

6.1 Ručna evaluacija

Prvo je provedena ručna evaluacija odgovora na postavljena pitanja. Svaki odgovor je klasificiran u jednu od četiri kategorije, uzimajući u obzir njegovu kvalitetu u odnosu na učitani dokument i relevantnost u odnosu na postavljeno pitanje. Kvalitativna metrika za ručnu evaluaciju odgovora RAG sustava, koja se temelji na kvantitativnim metodama iz poglavlja 4.5.2, prikazana je u nastavku:

- **Točno**: Generirani odgovor je u potpunosti točan i u skladu s izvorom znanja. Odgovor na pitanje je koherentan, bez dodatnih potpitanja, pozdrava, nelogičnosti i drugih oblika halucinacija.
- **Djelomično točno**: Odgovor je djelomično točan, ali usklađen s izvorom znanja. Sadrži manje halucinacije, poput pozdrava prema korisniku.
- **Pogrešan alat**: Semantički usmjerivač je odabrao pogrešan RAG alat, što je rezultiralo potpuno netočnim odgovorom.
- **Netočno**: Odgovor je potpuno netočan ili predstavlja potpunu halucinaciju - model je izmislio odgovore koji nisu u skladu s dohvaćenim izvorom znanja; model je potpuno ignorirao inicijalno pitanje; model je naveo previše nerelevantnih informacija koje nemaju veze s postavljenim pitanjem.

Odgovori i dohvaćanje kroz RAG provedeni su korištenjem 7 različitih LLM-ova, od kojih su 3 modela otvorenog kôda pokrenuta lokalno. Ovi modeli su primijenjeni u svim koracima osim pri izgradnji RAPTOR stabla, za što je uvijek korišten OpenAI *text-embedding-3-small* model. Evaluacija je obuhvatila sljedeće modele:

- **OpenAI** gpt-4 (API)
- **OpenAI** gpt-3.5-turbo (API)
- **Google** gemma:7b (Ollama lokalno)
- **Meta** llama3b:8b (Ollama lokalno)
- **Mistral** mistral:7b (Ollama lokalno)
- **Anthropic** Claude 3 Opus (API)
- **Anthropic** Claude 3 Sonnet (API)

Definicija pitanja temelji se na informacijama iz vanjskih izvora, uz korištenje tri RAG tehnike opisane u poglavlju o implementaciji EduBota:

- RAPTOR
- SQL-RAG
- Web Scraper

Od ukupno 20 pitanja, za točan odgovor na 15 pitanja (Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q12, Q13, Q14, Q15, Q16, Q17, Q20) potreban je **RAPTOR** modul. Tri pitanja ($Q1, Q2, Q3$) zahtijevaju **SQL-RAG** modul, dok je za dva pitanja ($Q18, Q19$) potreban **Web scraper** modul za davanje ispravnog odgovora.

Rezultati ručne evaluacije prikazani su u Tablici 4.

Pitanje	gpt-4o	gpt-3.5-turbo	gemma:7b	llama3:8b	mistral:7b	Claude 3 Opus	Claude 3 Sonnet
1. Tko je nositelj kolegija Programsko inženjerstvo?	točno	točno	točno	točno	djelomično točno	točno	točno
2. Tko je nositelj kolegija UPP	točno	točno	točno	točno	djelomično točno	točno	točno
3. Koji su ishodi učenja na kolegiju PI?	točno	točno	pogrešan alat	pogrešan alat	djelomično točno	točno	točno
4. Koje su studentske obaveze na kolegiju upravljanje poslovnim procesima?	točno	djelomično točno	netočno	točno	netočno	točno	točno
5. Kako se radi funkcija u JavaScriptu?	točno	točno	netočno	točno	točno	točno	točno
6. Objasni mi koncept rekurzije u JavaScriptu kroz primjer.	točno	točno	točno	točno	točno	točno	točno
7. Na kojim jezicima se izvodi online nastava na Sveučilištu?	točno	točno	točno	točno	pogrešan alat	pogrešan alat	točno
8. Koji je maksimalni broj mentorstva po nastavniku?	točno	pogrešan alat	netočno	pogrešan alat	pogrešan alat	pogrešan alat	točno
9. Reci mi ime i prezime studenta s emailom ppetro-vic@gmail.com	točno	točno	točno	točno	točno	točno	točno

Pitanje	gpt-4o	gpt-3.5-turbo	gemma:7b	llama3:8b	mistral:7b	Claude 3 Opus	Claude 3 Sonnet
10. Koliko bodova ima studentica Ana Anić iz PJS na 3. ispitu?	točno	točno	točno	točno	točno	točno	točno
11. Koji student ima najveći ukupni broj bodova iz predmeta PJS?	točno	točno	točno	točno	netočno	točno	točno
12. Navedi mi obaveznu literaturu za kolegij Upravljanje poslovnim procesima.	točno	točno	točno	točno	točno	točno	točno
13. Koji su preduvjeti za upis i cilj kolegija programsko inženjerstvo?	točno	točno	djelomično točno	netočno	točno	točno	točno
14. Što su studenti dužni ispuniti za sudjelovanje u online nastavi?	točno	djelomično točno	pogrešan alat	djelomično točno	netočno	točno	točno
15. Navedi mi točke Članka 9. Opterećenja studenata i provedbe ispita.	točno	točno	pogrešan alat	pogrešan alat	pogrešan alat	djelomično točno	točno
16. Kada se izvodi Programsko inž., a kada upravljanje poslovnim procesima?	točno	točno	netočno	djelomično točno	netočno	točno	točno
17. Glavne razlike studentskih obaveza na kolegijima UPP i PI?	točno	točno	netočno	pogrešan alat	djelomično točno	točno	točno

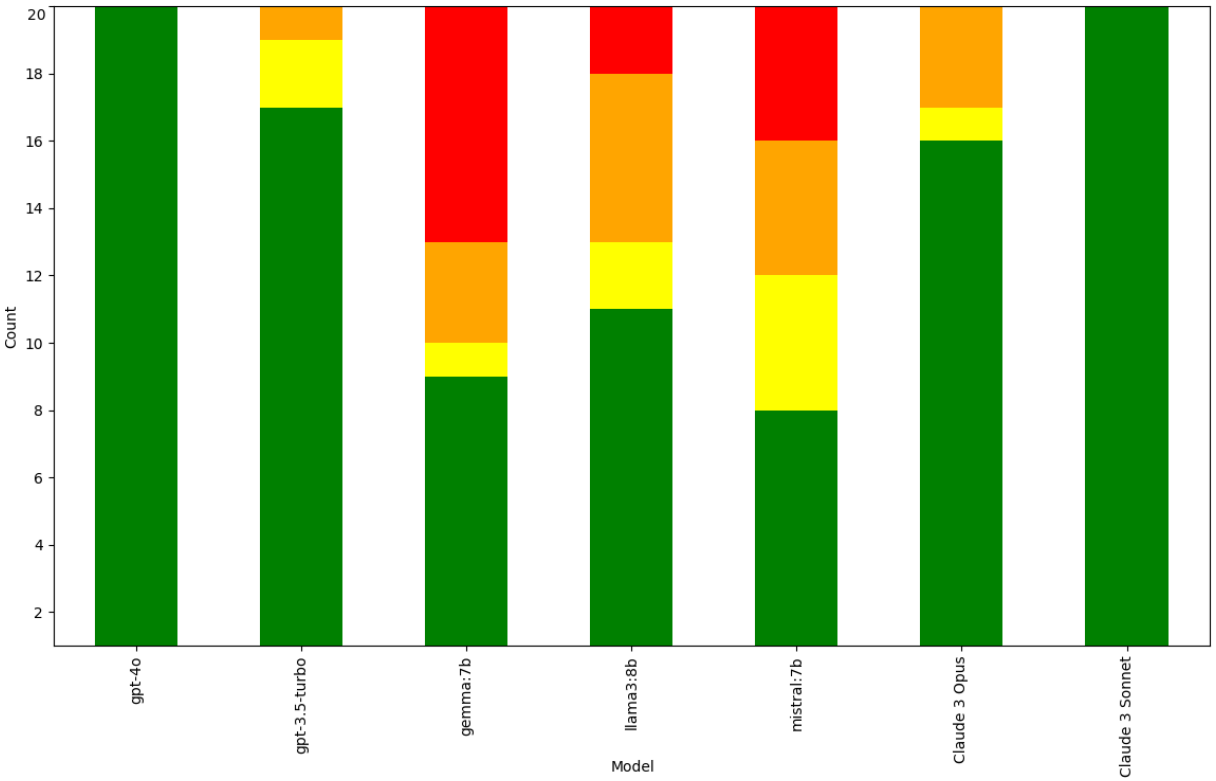
Pitanje	gpt-4o	gpt-3.5-turbo	gemma:7b	llama3:8b	mistral:7b	Claude 3 Opus	Claude 3 Sonnet
18. Koje su 3 najnovije vijesti sa Sveučilišta Jurja Dobrile u Puli?	točno	točno	točno	točno	točno	točno	točno
19. Koja je najnovija vijest na Fakultetu informatike u Puli?	točno	točno	netočno	netočno	točno	točno	točno
20. Koje su šifre kolegija Upravljanje poslovnim procesima i Programsko inženjerstvo?	točno	točno	netočno	točno	pogrešan alat	pogrešan alat	točno

Tablica 4: Evaluacija 20 pitanja u EduBotu uz korištenje 7 različitih modela

GPT-4o i Claude 3 Sonnet dali su zadovoljavajuće odgovore na svih 20 pitanja. Claude 3 Opus i GPT-3.5 Turbo imali su nekoliko pogrešnih odabira alata, ali nijedan potpuno netočan odgovor. Lokalni model Gemma imao je najlošije rezultate s čak sedam netočnih odgovora, dok je Llama3 bio najbolji među lokalnim modelima. Zbog velikog broja netočnih odgovora lokalnih modela uslijed pogrešnih odabira alata, daljnja istraživanja trebala bi se fokusirati na poboljšanje semantičkog usmjeravanja. Ipak, lokalni modeli pokazuju zadovoljavajuće rezultate s RAG tehnikama. Slika 64 prikazuje rezultate ručne evaluacije odgovora.

U fazi generiranja, modeli često dodaju dodatne nepovezane komentare, upute ili pitanja koja se ne odnose na postavljeno pitanje, poput: "Srdačan pozdrav studentu", "Dragi studenti", "Obratite mi se ako imate još pitanja" i slično. Iz tog razloga aktivno se razrađuju tehnike za evaluaciju LLM-ova, ali i RAG tehnika, budući da postoji više faza evaluacije kod velikih jezičnih modela. Kod RAG-a to uključuje i kvalitetu dohvaćanja, točnost generiranja samog odgovora te relevantnost odgovora.

U poglavlju 6.2 bit će prikazana evaluacija na istim pitanjima i modelima, ali koristeći LLM evaluaciju kroz dvije metrike vjernosti s dohvaćenim kontekstom i relevantnosti odgovora u odnosu na pitanje.

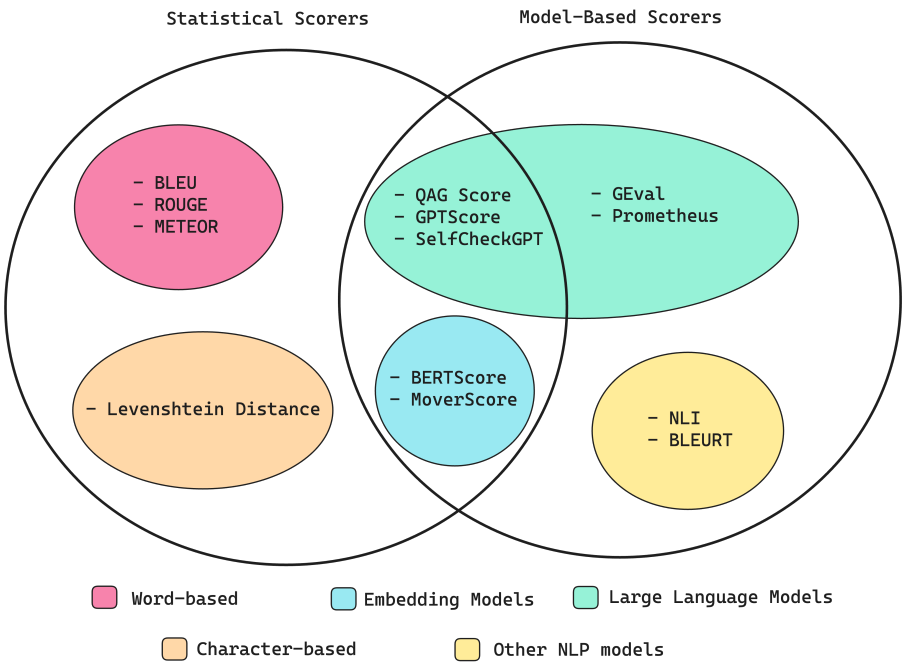


Slika 64: Rezultati EduBot evaluacije; točno (zelena), djelomično točno (žuta), pogrešan alat (narančasta), netočno (crvena)

6.2 LLM evaluacija

Evaluacija rezultata velikih jezičkih modela (LLM) ključna je za razvoj robusnih aplikacija, ali predstavlja izazov zbog složenosti same procjene. LLM evaluacijske metrike, poput točnosti odgovora, semantičke sličnosti i halucinacija, kvantificiraju performanse modela na temelju različitih kriterija. Pravilno definirane metrike omogućuju preciznu procjenu uspješnosti modela, bilo da se radi o samom modelu ili sustavu temeljenom na dohvaćanju i generaciji (RAG). Odabir odgovarajućih metrika, uključujući specifične zadatke kao što su sažimanje i relevantnost konteksta, ključan je za uspostavu efikasnog evaluacijskog procesa.

Statistički evaluatori (*eng. Statistical Scorers*), poput BLEU, ROUGE, METEOR i Levenshteinove udaljenosti, fokusiraju se na kvantitativne podudarnosti između izlaza LLM-a i referentnih tekstova, ali često ne uspijevaju uhvatiti semantičke nijanse i rezoniranje, što ih čini manje točnima za složene evaluacije LLM-a. Nasuprot tome, evaluatori temeljeni na modelima (*eng. Model-Based Scorers*), poput NLI i BLEURT, koriste NLP modele za procjenu logičke konzistentnosti i semantičkog usklađivanja, nudeći veću točnost, ali pate od problema pouzdanosti zbog svoje probabilističke prirode i ovisnosti o kvaliteti podataka za treniranje. Dakle, statistički evaluatori su jednostavniji i pouzdaniji, dok ocjenjivači temeljeni na modelima pružaju bolju semantičku evaluaciju, ali na račun konzistentnosti. Slika 65 prikazuje vrste alata za evaluaciju velikih jezičnih modela.



Slika 65: Alati za evaluaciju rezultata velikih jezičnih modela

Kombiniranje statističkih evaluatora i evaluatora baziranih na modelima može ponuditi uravnotežen pristup za procjenu jezičnih modela. Statistički evaluatori osiguravaju pouzdanost i jednostavnost, dok evaluatori bazirani na modelima dodaju dubinsku semantičku procjenu. Time se maksimizira preciznost evaluacije, uz zadržavanje dosljednosti. Primjeri takvih kombinacija uključuju korištenje BLEU ili ROUGE u kombinaciji s BERTScore ili MoverScore, što omogućuje sveobuhvatnu evaluaciju koja uzima u obzir i kvantitativne i kvalitativne aspekte generiranog teksta.

6.2.1 Metrike

Visokokvalitetni odgovori kod LLM-ova rezultat su dobro odrađenih RAG tehnika, preciznije pretraživača (*eng. retriever*) i generatora (*eng. generator*). Ove tehnike omogućavaju modelima da pretražuju i prikupe relevantne informacije, te ih zatim koherentno i precizno predstave korisniku. Za potrebe evaluacije, koriste se Faithfulness i Answer Relevancy metrike bazirane na QAG evaluatoru. **QAG Scorer** evaluator koristi se za procjenu točnosti i relevantnosti odgovora uspoređujući generirane odgovore s referentnim podacima [95]. Evaluacija se provodi koristeći okvir *deepeval*, koji omogućava sveobuhvatnu i preciznu analizu performansi modela [96].

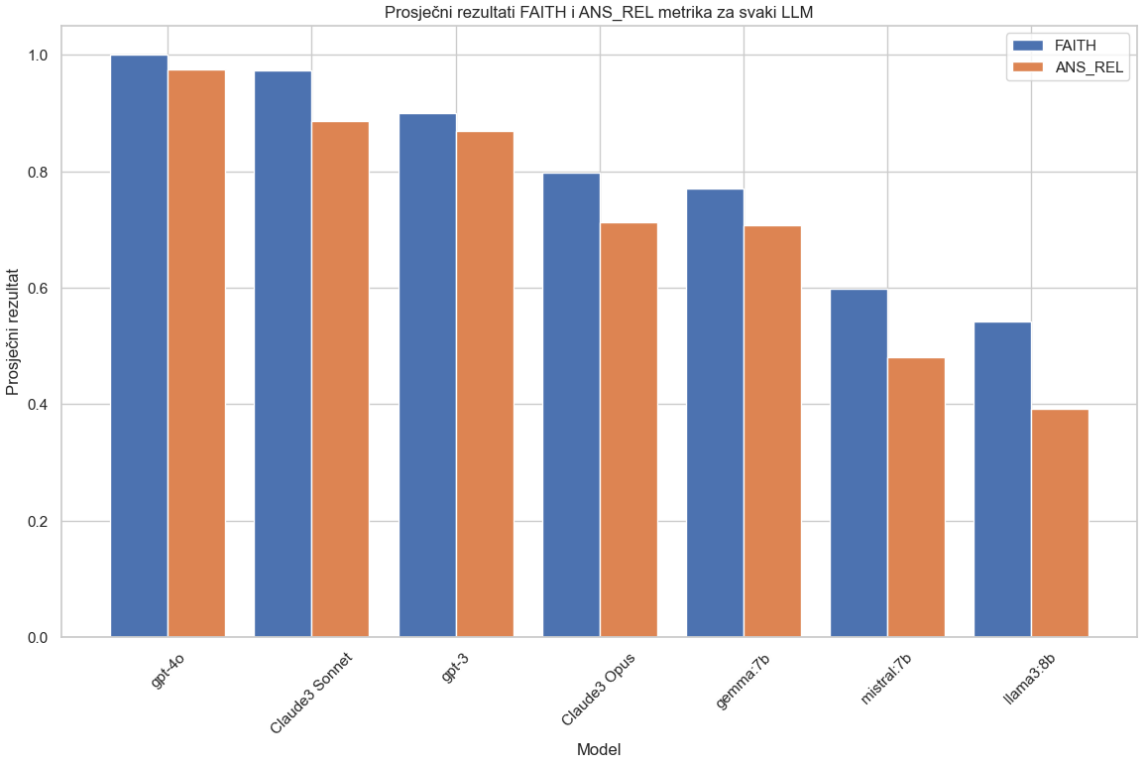
- **Faithfulness (FAITH)**: procjenjuje generira li generator modul u RAG sustavu odgovore koji se činjenično podudaraju s informacijama dohvaćenim iz konteksta pretraživanja. Koristi *QAG Scorer* i definirana je kao omjer istinitih tvrdnji u odgovoru LLM-a u odnosu na kontekst pretraživanja. Primjenjuje se algoritam opisan sljedećim pseudokodom:
  - Ekstrakcija tvrdnji: ekstrakcija tvrdnji iz generiranih odgovora.

- Evaluacija tvrdnji: provjerava za svaku tvrdnju slaže li se sa svakim pojedinačnim čvorom u dohvaćenom kontekstu. U QAG evaluatoru se postavlja sljedeće pitanje: "Slaže li se dana tvrdnja s referentnim tekstom?", gdje referentni tekst predstavlja svaki pojedinačno dohvaćeni čvor.
- Ograničavanje odgovora: odgovori na pitanje se ograničavaju na "da", "ne" i "ne znam".
- Izračunavanje ocjene: zbraja se ukupan broj istinitih tvrdnji ("da" i "ne znam") te se dijeli s ukupnim brojem tvrdnji.

- **Answer Relevancy (ANS_REL)**: procjenjuje generira li RAG generator sažete odgovore, a može se izračunati određivanjem omjera rečenica u odgovoru LLM-a koje su relevantne za postavljeno pitanje (tj. dijeljenje broja relevantnih rečenica s ukupnim brojem rečenica). Za izgradnju robusne metrike relevantnosti odgovora ključno je uzeti u obzir kontekst pretraživanja, budući da dodatni kontekst može opravdati naizgled nerelevantnu rečenicu. Evaluacija relevantnosti odgovora uključuje:
  - Procjena relevantnosti rečenica: provjera svake rečenice u odgovoru kako bi se utvrdilo je li relevantna za postavljeno pitanje.
  - Korištenje konteksta pretraživanja: uzimanje u obzir dodatnog konteksta pretraživanja kako bi se opravdala relevantnost pojedinih rečenica.

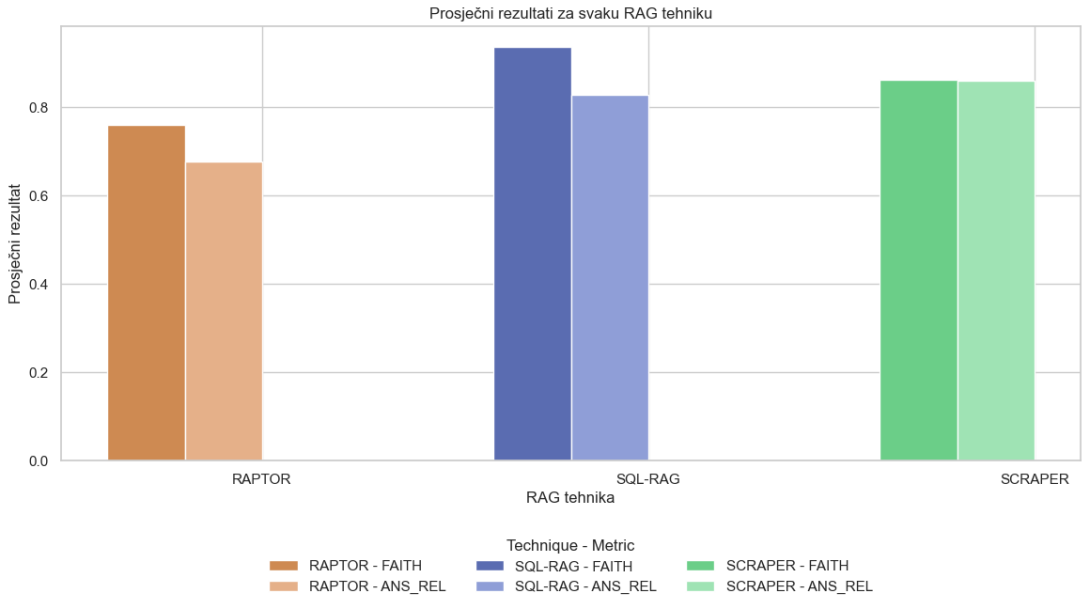
  Razlika između metrike relevantnosti odgovora (*eng. Answer Relevancy*) i metrike vjernosti (*eng. Faithfulness*) leži u njihovom fokusu. Vjernost se usredotočuje na činjeničnu točnost tvrdnji u odgovoru s obzirom na pretraživački kontekst, dok se relevantnost odgovora fokusira na konzistentnost generiranog odgovora u odnosu na inicijalno postavljeno pitanje. Primarni fokus relevantnosti je na odnosu prema korisniku i postavljenom pitanju, uključujući ton komunikacije, dodatne komentare, potpitanja i kvalitetu odgovora, te izbjegavanje odgovaranja na nepostavljena pitanja.

  Za svaki od sedam modela provedene su tri evaluacije za svaku od metrika ($FAITH$ i ANS_REL), nakon čega su uzete prosječne vrijednosti. Vrijednosti ovih metrika su u intervalu [0-1]. Slika 66 prikazuje prosječne vrijednosti obje metrike za svaki model.



Slika 66: Prosječni rezultati FAITH i ANS_REL metrika za svaki LLM

S obzirom na to da se LLM-ovi prikazani u tablicama koriste za RAG (faza dohvaćanja i faza generiranja), ali isti model služi i za semantičko usmjeravanje, koje nije predmet ovog istraživanja, neki modeli su pogrešno usmjeravali prema RAG modulu. To je dovelo do situacije u kojoj su obje metrike ocijenjene s 0.00. Slika 67 prikazuje rezultate $FAITH$ i ANS_REL metrika za svaku RAG tehniku.

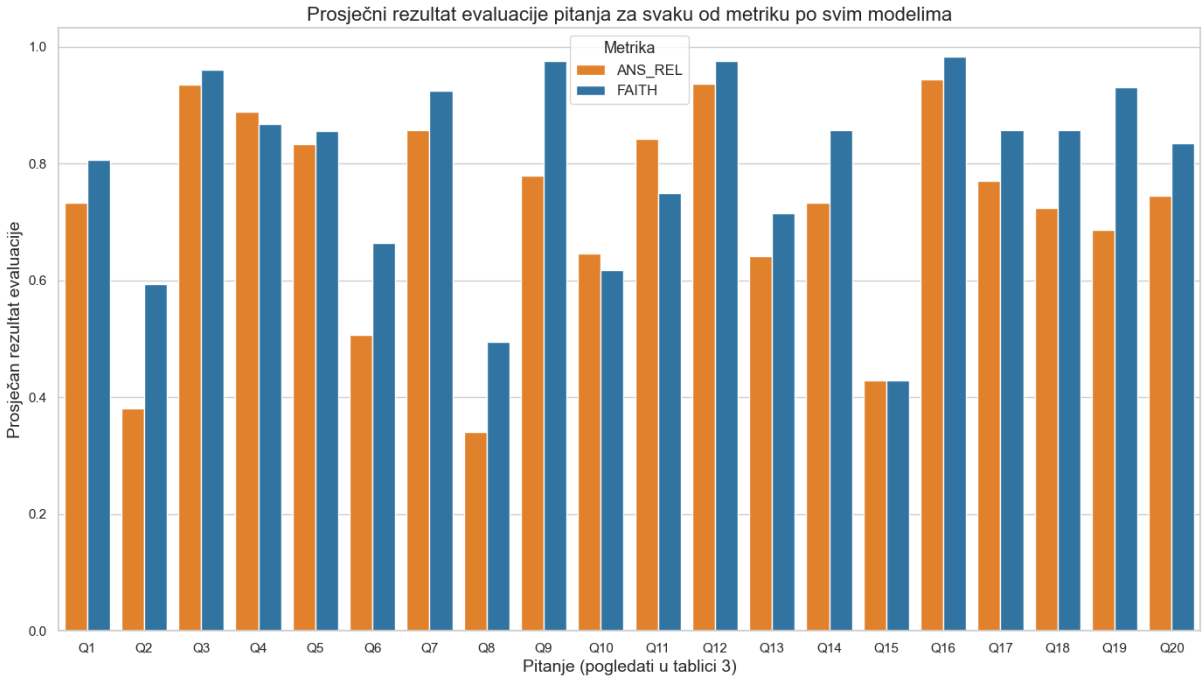


Slika 67: Prosječni rezultati FAITH i ANS_REL metrika za svaku RAG tehniku

Rezultati istraživanja ukazuju na izvanredne rezultate SOTA modela poput GPT-4o i Claude 3 Sonnet, posebno u semantičkom usmjeravanju, ali i još važnije, u fazama dohvaćanja i generiranja konačnih odgovora (RAG). Nešto slabije varijante, kao što su GPT-3.5 Turbo i Claude 3 Opus, također postižu odlične rezultate. Ipak, ti modeli povremeno dodaju nerelevantne komentare u svoje odgovore, poput "Dragi studenti", "Srdačan pozdrav", "Ako imate još pitanja" i slične fraze, što može umanjiti kvalitetu generiranih odgovora. Lokalni modeli poput Gemma i Mistral također su pokazali dobre rezultate, ali slične probleme u obraćanju korisnicima. Najslabije rezultate pokazuje model Llama3, prvenstveno zbog lošeg semantičkog usmjeravanja, što često dovodi do pogrešaka u RAG modulu.

Konačno, Slika 68 prikazuje prosječne rezultate evaluacije za svako od pitanja (označeno Q1 - Q20, pri čemu se pitanja mogu vidjeti u Tablici 4). Uočava se viša vrijednost FAITH metrike, što ukazuje na to da su generirani odgovori uglavnom u skladu s dohvaćenim kontekstom. Međutim, ANS_REL metrika pokazuje slabije rezultate, posebno kod manjih modela, jer odgovori često sadrže različite oblike halucinacija i dodatnih komentara, kako je prethodno naglašeno.

Primijećena je značajno slabija izvedba većine modela kod pitanja koja zahtijevaju odgovore iz skeniranog dokumenta pravilnika Sveučilišta, posebno za pitanja Q8 i Q15, zbog pogrešaka u kodiranju takvog teksta. Pitanja Q20, Q17 i Q16 odnose se na više dokumenata (dva nastavna plana - Programsko inženjerstvo i Upravljanje poslovnim procesima, koji su sadržani u zasebnim dokumentima). Unatoč toj prepreci, rezultati kod tih pitanja su vrlo zadovoljavajući. To predstavlja ključnu prednost RAPTOR metode, koja omogućuje kvalitetnu integraciju značajki udaljenih dokumenata i izradu dobrih sažetaka informacija koje se u njima nalaze.



Slika 68: Prosječni rezultat evaluacije pitanja za svaku od metriku po svim modelima

7 Zaključak

Generativna umjetna inteligencija temeljena na velikim jezičnim modelima postaje sveprisutna, pojavljujući se u brojnim aplikacijama i digitalnim alatima za različite svrhe. Integrira se u razne djelatnosti, uključujući informacijsko-komunikacijske tehnologije, financije, medicinu, inženjerstvo, proizvodnju, transport i zabavu. Posebno se ističe u obrazovanju, gdje organizacije poput Khan Academy i Duolingo već godinama koriste AI rješenja u svojim aplikacijama. Veliki jezični modeli otvorili su nova vrata personalizaciji obrazovanja zahvaljujući svojoj sposobnosti generiranja prilagođenih odgovora i naprednog rasuđivanja.

Jedna od glavnih prednosti velikih jezičnih modela je njihovo opsežno parametarsko znanje, koje im omogućuje pristup širokom spektru informacija, sposobnost kvalitetnog rasuđivanja i kreativnost. Međutim, upravo to znanje može biti i nedostatak zbog ograničenosti na podatke trenirane do određenog datuma i nemogućnosti pristupa vanjskim izvorima podataka, što često dovodi do tzv. halucinacija – stanja u kojem model na uvjerljiv način iznosi netočne tvrdnje ili dezinformacije.

RAG (*Retrieval-Augmented Generation*) predstavlja skup tehnika koje kombiniraju *prompt engineering* i *fine-tuning* kako bi se model obogatio znanjem iz vanjskih izvora podataka prije generiranja odgovora. U tu svrhu razvijen je velik broj tehnika koje omogućuju dohvaćanje podataka iz izvora kao što su internetski članci, PDF dokumenti, baze podataka i API servisi. U ovom radu, uz sveobuhvatni pregled razvoja NLP-a, od statističkih i *rule-based* modela do modernih LLM-ova temeljenih na transformer arhitekturi, razvijen je EduBot – konverzacijski agent koji koristi RAG module kako bi pomogao studentima i nastavnicima pružajući personalizirane i precizne odgovore na pitanja.

EduBot prepoznaje namjeru korisnika i na temelju nje poziva odgovarajući RAG modul. Provedena evaluacija EduBota pokazala je značajna poboljšanja i prednosti korištenja RAG pristupa. Rezultati su dobiveni ručnom evaluacijom 20 pitanja vezanih uz nastavne materijale, gradivo raznih kolegija, nastavne planove te ostale pravilnike sveučilišta. Moćna komercijalna rješenja poput GPT-4 i Claude 3 Sonnet pokazala su izvanredne performanse, pružajući točne odgovore na sva pitanja. Manji modeli poput Mistral, Gemma i Llama3 također su pokazali značajne mogućnosti, uz nekoliko pogrešaka. Zaključuje se da moderni RAG pristup otvara vrata mogućnostima i potencijalu ovih manjih otvorenih modela.

Evaluacija EduBota temeljena na modelima pokazala je da SOTA modeli poput GPT-4o i Claude 3 Sonnet postižu izvanredne rezultate u fazama dohvaćanja i generiranja odgovora (RAG), dok su slabiji modeli poput GPT-3.5 Turbo i Claude 3 Opus skloni dodavanju nerelevantnih komentara. Lokalni modeli, kao što su Gemma i Mistral, pokazuju dobre rezultate, ali imaju slične probleme u obraćanju prema korisniku, dok je model Llama3 pokazao najslabije rezultate zbog lošeg semantičkog usmjeravanja. RAPTOR metoda se istaknula u kvalitetnoj integraciji značajki udaljenih dokumenata, omogućujući izradu dobrih sažetaka informacija, što je posebno korisno kod pitanja koja zahtijevaju odgovore iz više dokumenata.

Literatura

[1] Statista. Artificial intelligence - global | statista market forecast, 2024. Pristupljeno: 6.6.2024.

[2] AWS. What are large language models? - llm ai explained - amazon aws, Jun 2024. Pristupljeno: 6.6.2024.

[3] Luiz Rodrigues, Paula T Palomino, Armando M Toda, Ana CT Klock, Marcela Pessoa, Filipe D Pereira, Elaine HT Oliveira, David F Oliveira, Alexandra I Cristea, Isabela Gasparini, et al. How personalization affects motivation in gamified review assessments. *International Journal of Artificial Intelligence in Education*, pages 1–38, 2023.

[4] Ashok Federick. Finland education system. *International Journal of Science and Society*, 2(2):21–32, 2020.

[5] Hani Morgan. Review of research: The education system in finland: A success story other countries can emulate. *Childhood Education*, 90(6):453–457, 2014.

[6] OECD. *Education at a Glance 2018: OECD Indicators*. OECD Publishing, Paris, 2018. Pristupljeno: 28.6.2024.

[7] QRIDI. Unveiling the secrets of finnish teaching methods: A comprehensive guide, Jun 2024.

[8] Gunda Tire. Estonia: A positive pisa experience. *Improving a Country's Education: PISA 2018 Results in 10 Countries*, pages 101–120, 2021.

[9] Ziwei Ji, Tiezheng Yu, Yan Xu, Nayeon Lee, Etsuko Ishii, and Pascale Fung. Towards mitigating LLM hallucination via self reflection. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1827–1843, Singapore, December 2023. Association for Computational Linguistics.

[10] IBM-LLM. What are Large Language Models (LLMs)? | IBM. Pristupljeno: 10.6.2024.

[11] Dilwyn Edwards and Michael Hamson. Mathematical guides. In *Guide to Mathematical Modelling*, 1990.

[12] David Freedman. *Statistical Models: Theory and Practice*. Cambridge University Press, 01 2005.

[13] Jabbar Hussain. Deep learning black box problem, 2019. Pristupljeno: 12.7.2024.

[14] IBM. What is a Neural Network? | IBM. Pristupljeno: 1.7.2024.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[16] Farhana Sultana, Abu Sufian, and Paramartha Dutta. Advancements in image classification using convolutional neural network. In *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, pages 122–129. IEEE, 2018.

[17] Tausif Diwan, G Anirudh, and Jitendra V Tembhurne. Object detection using yolo: Challenges, architectural successors, datasets and applications. *multimedia Tools and Applications*, 82(6):9243–9275, 2023.

[18] Leiyu Chen, Shaobo Li, Qiang Bai, Jing Yang, Sanlong Jiang, and Yanming Miao. Review of image classification algorithms based on convolutional neural networks. *Remote Sensing*, 13(22):4712, 2021.

[19] Lorenzo Rosasco, Ernesto De Vito, Andrea Caponnetto, Michele Piana, and Alessandro Verri. Are Loss Functions All the Same? *Neural Computation*, 16(5):1063–1076, 05 2004.

[20] Robert Moore and John DeNero. L1 and l2 regularization for multiclass hinge loss models. In *Symposium on machine learning in speech and language processing*, 2011.

[21] Stanford University CS231N. Stanford University CS231N: Deep Learning for Computer Vision. Pristupljeno: 1.7.2024.

[22] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[23] R.L. Trask and P. Stockwell. *Language and Linguistics: The Key Concepts*. Routledge key guides. Routledge, 2007.

[24] Mary Jo Nye. Speaking in tongues: Science's centuries-long hunt for a common language. *Distillations*, 2(1):40–43, 2016.

[25] Adam Lopez. Statistical machine translation. *ACM Comput. Surv.*, 40(3), aug 2008.

[26] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016.

[27] Marianna Baranovska and Stefan Höltgen. *Hello, I'm Eliza*. Projekt Verlag; 2nd edition (4 Sept. 2023), 12 2018.

[28] Bill Chamberlain. *The Policeman's Beard is Half-constructed*. Grand Central Pub, 1984.

[29] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155, mar 2003.

[30] Daniel W. Otter, Julian R. Medina, and Jugal K. Kalita. A survey of the usages of deep learning in natural language processing, 2019.

[31] Jianqiong Xiao and Zhiyong Zhou. Research progress of rnn language model. In *2020 ieee international conference on artificial intelligence and computer applications (icaica)*, pages 1285–1288. IEEE, 2020.

[32] Seol-Hyun Noh. Analysis of gradient vanishing of rnns and performance comparison. *Information*, 12(11):442, 2021.

[33] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[35] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[36] Rick Merritt. What Is Retrieval-Augmented Generation aka RAG | NVIDIA Blogs, 6 2024. Pristupljeno: 12.6.2024.

[37] M. Saeed. A gentle introduction to positional encoding in transformer models, part 1, 2023. Pristupljeno: 15.6.2024.

[38] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016.

[39] Hao-Ping (Hank) Lee, Yu-Ju Yang, Thomas Serban Von Davier, Jodi Forlizzi, and Sauvik Das. Deepfakes, phrenology, surveillance, and more! a taxonomy of ai privacy risks. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA, 2024. Association for Computing Machinery.

[40] Maanak Gupta, Charankumar Akiri, Kshitiz Aryal, Eli Parker, and Lopamudra Praharaj. From chatgpt to threatgpt: Impact of generative ai in cybersecurity and privacy. *IEEE Access*, 11:80218–80245, 2023.

[41] J. King and C. Meinhardt. Rethinking privacy in the ai era: Policy provocations for a data-centric world. *Stanford HAI*, February 22 2024.

[42] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. Pristupljeno: 2.7.2024.

[43] Andreas Liesenfeld, Alianda Lopez, and Mark Dingemanse. Opening up chatgpt: Tracking openness, transparency, and accountability in instruction-tuned text generators. In *Proceedings of the 5th International Conference on Conversational User Interfaces*, CUI '23, New York, NY, USA, 2023. Association for Computing Machinery.

[44] Andreas Liesenfeld and Mark Dingemanse. Rethinking open source generative ai: open washing and the eu ai act. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '24, page 1774–1787, New York, NY, USA, 2024. Association for Computing Machinery.

[45] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[46] Sayash Kapoor, Rishi Bommasani, Daniel E. Ho, Percy Liang, and Arvind Narayanan. On the societal impact of open foundation models. *Stanford HAI*, Feb 2024.

[47] Mika Westerlund. The emergence of deepfake technology: A review. *Technology innovation management review*, 9(11), 2019.

[48] Katarina Kertysova. Artificial intelligence and disinformation: How ai changes the way disinformation is produced, disseminated, and can be countered. *Security and Human Rights*, 29(1-4):55–81, 2018.

[49] J. Quinn, J. McEachen, M. Fullan, M. Gardner, and M. Drummy. *Dive Into Deep Learning: Tools for Engagement*. SAGE Publications, 2019.

[50] IBM. What is Prompt Engineering? | IBM. Pristupljeno: 1.7.2024.

[51] Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences, 2020.

[52] Dongfang Li, Zetian Sun, Baotian Hu, Zhenyu Liu, Xinshuo Hu, Xuebo Liu, and Min Zhang. Improving attributed text generation of large language models via preference learning, 2024.

[53] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

[54] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*, 2019.

[55] Gary Marcus. The next decade in ai: Four steps towards robust artificial intelligence, 2020.

[56] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR, 13–18 Jul 2020.

[57] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[58] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey, 2024.

[59] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.

[60] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[61] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[62] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[63] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2022.

[64] Shuhe Wang, Beiming Cao, Shengyu Zhang, Xiaoya Li, Jiwei Li, Fei Wu, Guoyin Wang, and Eduard Hovy. Sim-gpt: Text similarity via gpt annotated data, 2023.

[65] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models, 2024.

[66] Weaviate dokumentacija. Vector Indexing | Weaviate - Vector Database. Pristupljeno: 1.7.2024.

[67] Milvus dokumentacija. Vector Index Milvus v2.0.x documentation. Pristupljeno: 1.7.2024.

[68] Yu. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs, 2018.

[69] Kai Zhao, Haiping Lu, and Jianguo Mei. Locality preserving hashing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

[70] Erik Bernhardsson. Annoy: Approximate nearest neighbors in c++/python, 2018. Pristupljeno: 17.6.2024.

[71] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *arXiv preprint arXiv:2401.08281*, 2024.

[72] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, Kun Yu, Yuxing Yuan, Yinghao Zou, Jiquan Long, Yudong Cai, Zhenxiang Li, Zhifeng Zhang, Yihua Mo, Jun Gu, Ruiyi Jiang, Yi Wei, and Charles Xie. Milvus: A purpose-built vector data management system. In *Proceedings of the 2021 International Conference on Management of Data*, SIGMOD '21, page 2614–2627, New York, NY, USA, 2021. Association for Computing Machinery.

[73] Qdrant. Qdrant vector database, 2024. Pristupljeno: 30.6.2024.

[74] Myungkeun Yoon. A constant-time chunking algorithm for packet-level deduplication. *ICT Express*, 5:120–145, 09 2018.

[75] GeeksforGeeks. Sliding window technique, 4 2024. Pristupljeno: 2.7.2024.

[76] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[77] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. Pristupljeno: 2.7.2024, 2017.

[78] Greg Kamradt. Greg Kamradt: Semantic chunking. Pristupljeno: 1.7.2024.

[79] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. Least-to-most prompting enables complex reasoning in large language models, 2023.

[80] Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models. *arXiv preprint arXiv:2309.11495*, 2023.

[81] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

[82] Pinecone dokumentacija. Rerankers and Two-Stage retrieval. Pristupljeno: 1.7.2024.

[83] Jaekeol Choi, Euna Jung, Jangwon Suh, and Wonjong Rhee. Improving bi-encoder document ranking models with two rankers and multi-teacher distillation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 2192–2196, 2021.

[84] Weizhi Fei, Xueyan Niu, Pingyi Zhou, Lu Hou, Bo Bai, Lei Deng, and Wei Han. Extending context window of large language models via semantic compression, 2023.

[85] Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy, 2023.

[86] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*, 2022.

[87] Gangwoo Kim, Sungdong Kim, Byeongguk Jeon, Joonsuk Park, and Jaewoo Kang. Tree of clarifications: Answering ambiguous questions with retrieval-augmented large language models, 2023.

[88] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. Webgpt: Browser-assisted question-answering with human feedback, 2022.

[89] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection, 2023.

[90] Snowflake Inc. Streamlit: A faster way to build and share data apps, 2019. Pristupljeno: 2.7.2024.

[91] Adrien Bouvais. streamlit-google-auth, 4 2024. Pristupljeno: 2.7.2024.

[92] Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. Raptor: Recursive abstractive processing for tree-organized retrieval, 2024.

[93] Leonard Richardson. Beautiful Soup Documentation — Beautiful Soup 4.12.0 documentation. Pristupljeno: 2.7.2024.

[94] llamaindex dokumentacija. LlamaIndex, data framework for LLM applications. Pristupljeno: 2.7.2024.

[95] Asahi Ushio, Fernando Alva-Manchego, and Jose Camacho-Collados. An empirical comparison of lm-based question and answer generation methods. *arXiv preprint arXiv:2305.17002*, 2023.

[96] Yixin Yang, Zheng Li, Qingxiu Dong, Heming Xia, and Zhifang Sui. Can large multimodal models uncover deep semantics behind images?, 2024.

Popis slika

1 Sučelje Khanmigo asistenta 4
2 UML dijagram obrazaca upotrebe EduBot aplikacije 9
3 Black-box VS White-box 10
4 Klasifikacija slike korištenjem neuronskih mreža 12
5 Struktura umjetnog neurona 13
6 Ilustracija linearnog klasifikatora na primjeru klasifikacije slika 14
7 Trajektorija gradijentnog spusta prikazuje kretanje funkcije cilja prema lokalnom minimumu 16
8 Neuronska mreža s jednim skrivenim slojem 17
9 Primjer predložaka koje neuronska mreža uči i dodjeljuje klasama 17
10 Primjer *NER* tehnike za identifikaciju i klasifikaciju entiteta iz teksta 21
11 *Word2Vec* - Ilustracija odnosa semantički srodnih riječi u 2D reprezentaciji vektorskog prostora 22
12 *Recurrent* VS *Feedforward* neuronske mreže 23
13 Recurrent Neural Network 23
14 Ilustracija *encoder-decoder* strukture *seq2seq* modela 25
15 Transformer modeli, ponekad zvani i *foundation* modelima, danas imaju jako široku upotrebu 26
16 Primjer kodiranja rečenice rastavljene na *tokene* 27
17 Primjer pozicijskog kodiranja za rečenicu "I am a robot". 28
18 Prikaz *encoder* modula transformera s *feedforward* operacijom, *self-attention* slojem i rezidualom kao alternativnom putanjom. 29
19 Arhitektura transformer modela 30
20 Tijek razvoja NLP modela 30
21 Fotografija generirana DALL·E 3 modelom uz *prompt*: "An illustration of an avocado sitting in a therapist's chair, saying 'I just feel so empty inside' with a pit-sized hole in its center. The therapist, a spoon, scribbles notes" 33
22 *High-level* prikaz rada RLHF metode 38
23 Ilustracija opisanog RAG pristupa - kombiniranje pretreniranog *retrievera* s pretreniranim *seq2seq generator* modelom. 40
24 Osnovni (*eng. naive*) RAG pristup 41
25 Oblici nestrukturiranih podataka 42
26 Pretvaranje nestrukturiranih podataka u unificirani format običnog teksta nakon odrađenog čišćenja i ekstrakcije 42
27 Prikaz segmentacije običnog teksta na manje "probavljive" segmente 43
28 Kodiranje segmenata u vektorske reprezentacije pomoću *embedding* modela . . 43
29 Pohrana dobivenih ugradbi (*eng. embeddings*) u vektorsku bazu podataka . . . 44

30 Kodiranje korisničkog upita 44
31 Prikaz pretraživanja sličnosti između vektora dokumenata/segmenata i vektora upita u višedimenzionalnom prostoru 46
32 Ilustracija opisanih procesa dohvaćanja relevantnih segmenata (1 i 2) na temelju semantičke sličnosti te generiranje odgovora koristeći novi kontekst i originalni korisnički upit. 47
33 Ilustracija osnovnog (*env. naive*) RAG procesa na primjeru odgovaranja na pitanja. Prikazana su 3 koraka: 1) **Indeksiranje**: Učitavanje datoteka, njihova podjela na segmente, kodiranje u vektore i pohrana u vektorsku bazu podataka. 2) **Dohvaćanje**: dohvaćanje top-k segmenata koji su najrelevantniji s postavljenim pitanjem obzirom na semantičku sličnost. 3) **Generacija**: kombiniranje korisničkog upita i dobivenih segmenata kao *prompt* LLM-u koji generira konačni odgovor. 48
34 Usporedba rijetkog (*eng. sparse*) i gustog (*eng. dense*) vektora. Rijetki vektori sadrže rijetko raspoređene bitove informacija (imaju mnogo nula), dok su gusti vektori bogatiji informacijama koje su gusto "upakirane" u svakoj dimenziji. . . 50
35 Napredni (*eng. advanced*) RAG pristup 53
36 Proces pretraživanja u hijerarhijskoj strukturi HNSW grafa 56
37 LSH funkcija ima za cilj smjestiti slične vrijednosti u iste "hash kante" 56
38 Segmentacija fiksne duljine 58
39 Ilustracija okvira kod tehnike kliznog prozora 58
40 Postupak semantičke segmentacije dijelova rečenice 60
41 a) *least-to-most* metoda, b) CoVe metoda 62
42 Primjer rada HyDe metode za optimizaciju upita. Zelenom bojom prikazani su stvarni upiti korisnika, a žutom dodatne instrukcije koje se prilažu u svrhu optimizacije upita. Smeđom bojom prikazani su generirani dokumenti, koji se potom uspoređuju sa stvarnim dokumentima prikazanim u plavoj boji. 62
43 Ilustracija usmjeravanja koristeći semantički usmjerivač (*eng. semantic router*) 64
44 Učinak promjene položaja relevantnih informacija (dokumenata koji sadrže odgovor) na performanse odgovaranja na pitanja iz većeg korpusa dokumenata. Performanse se pokazuju najboljima kada se relevantne informacije nalaze na samom početku ili samom kraju i rapidno degradiraju kada modeli moraju rasuđivati na temelju informacija u sredini ulaznog konteksta. 64
45 Ponovno rangiranje (*eng. Re-ranking*) 65
46 Primjer sintetičkog upita za zadatak dohvaćanja pristupnog ključa. Pretrenirani LLM nije sposoban obraditi dugi kontekst zbog ograničenja duljine konteksta. Primjenom semantičke kompresije konteksta, suvišne informacije u drugom dokumentu se uklanjaju, a kompresirani unos sadržava najbitnije informacije koje LLM lako obrađuje i generira točan odgovor. 66

47 Modularni RAG 67
48 a) IRCot, b) ToC 69
49 a) AutoGPT, b) Toolformer. Kod Toolformer tehnike može se vidjeti pozivanje raznih modula kroz sam proces (QA, Calculator, MT, WikiSearch) 69
50 Procesi augmentacije (1. Iterativni, 2. Rekurzivni, 3. Adaptivni) 70
51 Tehnološko stablo istraživanja RAG područja 70
52 Usporedba RAG metodologije s *prompt engineering* i *finetuning* pristupima [58] 71
53 EduBot GUI - početna stranica 73
54 Edubot GUI - Korisnički profil 74
55 EduBot GUI - Učitavanje datoteka 75
56 Proces izgradnje RAPTOR stabla 76
57 Ilustracija metoda pretraživanja RAPTOR stabla: A. Tree Traversal Retrieval, B. Collapsed Tree Retrieval 77
58 Učitavanje datoteka i izrada RAPTOR stabla 78
59 Opisani postupak rada SQL RAG modula 79
60 EduBot interakcija - RAPTOR modul 81
61 EduBot interakcija - RAPTOR modul: Prilagodba odgovora ovisno o predznanju 81
62 EduBot interakcija - SQL RAG modul 82
63 EduBot interakcija - Web scraper modul (lijevo) i direktni LLM (desno) 82
64 Rezultati EduBot evaluacije; točno (zelena), djelomično točno (žuta), pogrešan alat (narančasta), netočno (crvena) 88
65 Alati za evaluaciju rezultata velikih jezičnih modela 89
66 Prosječni rezultati FAITH i ANS_REL metrika za svaki LLM 91
67 Prosječni rezultati FAITH i ANS_REL metrika za svaku RAG tehniku 91
68 Prosječni rezultat evaluacije pitanja za svaku od metriku po svim modelima . . 92

Popis tablica

1 Tablica popularnih *foundation* modela (lipanj 2024.) 34

2 Segmenti s izračunatim semantičkim sličnostima za postavljeno korisničko pitanje o stvaranju objekta u JavaScriptu. 45

3 Usporedba evaluacijskih metrika za RAG modele: simbolom † se označava *benchmark*; ‡ predstavlja alate; zvjezdicom (*) su označene prilagođene kvantitativne metrike, koje se razlikuju od onih standardnih. 72

4 Evaluacija 20 pitanja u EduBotu uz korištenje 7 različitih modela 87