SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Zlatko Pračić

IZRADA AGENTA S GENERATIVNOM UMJETNOM INTELIGENCIJOM TEKSTA SPECIFIČNE NAMJENE

DIPLOMSKI RAD

Varaždin, 2026.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Zlatko Pračić

Matični broj: 0016145097

Studij: Baze podataka i baze znanja

IZRADA AGENTA S GENERATIVNOM UMJETNOM INTELIGENCIJOM TEKSTA SPECIFIČNE NAMJENE

DIPLOMSKI RAD

Mentor:

Prof. dr. sc. Markus Schatten

Varaždin, kolovoz 2026.

Zlatko Pračić

Izjava o izvornosti

Izjavljujem kako je ovaj diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI Radovi

Sažetak

Opsega od 100 do 300 riječi. Sažetak upućuje na temu rada, ukratko se iznosi čime se rad bavi, teorijsko-metodološka polazišta, glavne teze i smjer rada te zaključci.

Ključne riječi: riječ; riječ; ...riječ; Obuhvaća 7 ± 2 ključna pojma koji su glavni predmet rasprave u radu.

Sadržaj

1. Uvod

“We are drowning in information but starved for knowledge.” [1]
– John Naisbitt, Megatrends

2. Povijest Chatbotova

Popularnost chatbotova naglo je porasla predstavljanjem ChatGPT-a 2022. godine. Iz te se činjenice može zaključiti kako su chatbotovi postojali i prije ove nagle popularizacije. Stoga će se u ovom poglavlju prikazati povijesni razvoj chatbota, kao i teorijska osnova koja je utjecala na razvoj ideje o stroju koji razgovara.

2.1. Teorijske i matematičke pretpostavke

Iako je Andrey Markov uvođenjem Markovljevih lanaca u svom radu postavio važan temelj, ideja njegovog rada nije bila ni približna konceptu "stroja koji razgovara". Unatoč tome, vrijednost modela Markovljevog lanca za kasnija otkrića i razvoj znanstvene misli je neupitna.

S druge strane, Alan Turing promišlja o budućnosti strojeva čije su se sposobnosti ubrzano razvijale i postajale sve složenije. Zanimalo ga je može li stroj tijekom komunikacije uvjeriti ispitivača da nije stroj, što se u nastavku teksta formalizira u obliku Turingova testa. Iako se u tom kontekstu izravno ne spominju chatbotovi, upravo se na njih ovaj test u velikoj mjeri može primijeniti.

2.1.1. Markovljev lanac

Al-Amin i ostali u svom radu predstavljaju Markovljev lanac kao jednu od najranijih metoda koja je pokazala potencijal chatbota kroz generiranje teksta na temelju statističkih vjerojatnosti slijeda riječi. Markovljev lanac je razvio Andrey Markov 1906. godine za modeliranje stohastičkih procesa, a kasnije se počeo koristiti u zadacima poput predviđanja sljedeće riječi (npr. *autocomplete* u e-pošti). U chatbot kontekstu, Markovljev lanac procjenjuje vjerojatnost da određena riječ slijedi drugu i na temelju unaprijed naučenih prijelaznih vjerojatnosti "slaže" odgovor. Kao rana primjena navodi se HeX, koji je osvojio Loebnerovu nagradu 1997., nakon čega je interes za ovu tehniku postupno rastao [2].

Klasični Markovljevi modeli opisuju prelazak sustava između različitih stanja kroz vrijeme, pri čemu se prijelazne vjerojatnosti procjenjuju iz empirijskih podataka i pretpostavlja se da su stabilne kroz susjedne vremenske intervale. Prostorni Markovljevi modeli proširuju tu ideju na prostorne jedinice (npr. namjene zemljišta) te uključuju prostornu ovisnost između susjednih lokacija, a dodatna proširenja omogućila su i višedimenzionalne simulacije [2].

Iako su Markovljevi lanci jednostavni za implementaciju i mogu proizvesti dobre rezultate, previše su ograničeni za uvjerljive i koherentne razgovore. Kod predviđanja teksta obično "pamte" samo jednu ili nekoliko prethodnih riječi, pa bez šireg konteksta teško hvataju značenje i dugoročnu strukturu rečenica. Unatoč tome, bili su važan rani korak i dokaz koncepta za probabilističke/generativne pristupe te su svojim idejama utjecali na kasniji razvoj modernih razgovornih AI sustava [2].

2.1.2. Turingov test

Alan Turing svojim radom tvrdi kako je pitanje “Mogu li strojevi misliti?” nejasno pa ga zamjenjuje praktičnijim kriterijem: imitacijskom igrom (u nastavku kaže kako je to “test”), gdje ispitivač preko pisanih poruka pokušava razlikovati čovjeka od stroja. Fokusira se na digitalna računala, objašnjava njihovu građu (memorija, izvršna jedinica, kontrola/program) i naglašava njihovu univerzalnost. Jedno dovoljno snažno računalo može oponašati ponašanje bilo kojeg stroja s diskretnim stanjima ako je dobro programirano. Predviđa kako bi se u nekoliko desetljeća mogla pojaviti računala koja bi u takvoj igri bila dovoljno uvjerljiva u “razgovoru” s prosječnim ispitivačem [3].

Velik dio rada posvećen je odgovorima na prigovore kako strojevi ne mogu misliti: teološke tvrdnje o “duši”, strah od posljedica, matematička ograničenja, argument iz svijesti i razne “nemogućnosti” tipa da stroj ne može biti kreativan, griješiti ili učiti. Turing uglavnom pokazuje da ti prigovori ne dokazuju nemogućnost, često miješaju kvarove s pogrešnim zaključivanjem ili se oslanjaju na tadašnja ograničenja tehnologije. Kao pozitivni smjer predlaže da se ne programira “odrasli um” direktno, nego da se napravi “dječji stroj” koji se zatim uči i odgaja (nagrade/kazne, jezik naredbi, moguće i slučajnost), analogno procesu učenja i evolucije [3].

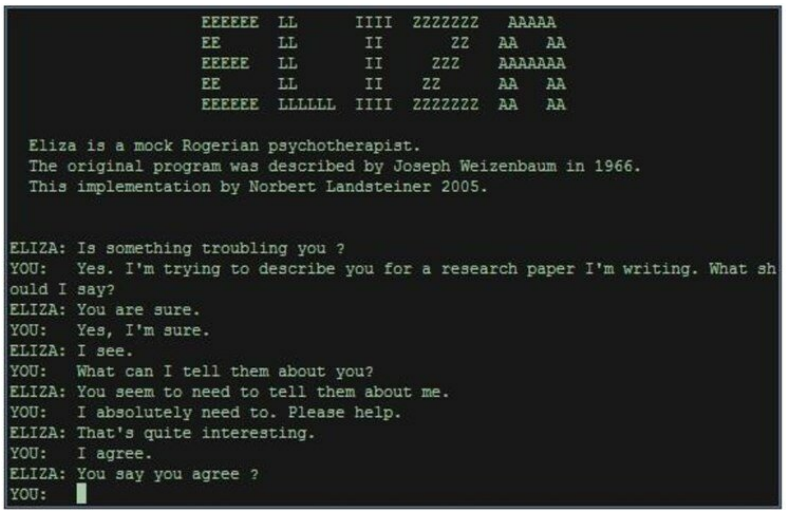
Al-Amin kaže kako se Turingov test može smatrati modelskom pretečom chatbota jer je već 1950. postavio razgovor u prirodnom jeziku kao ključni način provjere “inteligentnog” ponašanja mašine, kroz dijalog u kojem treba biti teško razlikovati mašinu od čovjeka. U tekstu se naglašava kako se evaluacija zasniva na tome može li računalo “prevariti” ljudskog ispitivača da pomisli kako razgovara s čovjekom, što je isti osnovni cilj konverzacijskih sistema poput chatbota. Također, kao utjecajan istraživački kamen temeljac, Turingov test je godinama i desetljećima motivirao razvoj obrade prirodnog jezika, čime je direktno usmjerio put prema modernim chatbot tehnologijama [2].

2.2. Prva generacija - tekstualni chatbotovi bazirani na pravilima

Prva generacija chatbotova, razvijana od sredine 1960-ih do ranih 1980-ih, označava prijelaz s teorijskih koncepata na prve praktične implementacije sustava koji simuliraju ljudski razgovor. Ovi rani programi temeljili su se na paradigmi prepoznavanja uzoraka (engl. *pattern matching*) i primjeni strogo definiranih, ručno pisanih pravila, bez stvarnog razumijevanja semantike ili konteksta. Oslanjajući se na identifikaciju ključnih riječi i njihovu transformaciju u unaprijed pripremljene odgovore, ovi su sustavi po prvi put demonstrirali mogućnost održavanja prividno koherentnog dijaloga unutar usko definiranih domena, čime su postavili temelje za razvoj interakcije čovjeka i računala te otkrili snažnu ljudsku sklonost počovječenih strojeva.

2.2.1. ELIZA

Rupert i ostali u svom radu detaljno obrađuju ELIZA, chatbot kojeg je Joseph Weizenbaum razvio na MIT-u sredinom 1960-ih godina. Po tom istraživanju, ELIZA predstavlja prvi



Slika 1: Originalni ispis koda ELIZA pronađen u Weizenbaumovim arhivama [5]

chatbot na svijetu, nastao na prvom *time-sharing* sustavu CTSS (*Compatible Time-Sharing System*) na IBM 7094 računalu. Rad opisuje kako je tim istraživača 2021. otkrio originalni ispis koda ELIZA u Weizenbaumovim arhivama na MIT-u, uključujući ranu verziju čuvenog DOCTOR skripta te gotovo potpunu verziju MAD-SLIP koda — kombinacije programskog jezika MAD i Weizenbaumovog vlastitog sustava SLIP (*Symmetric Lisp Processor*). Tim je uspješno "vratio na život" originalnu ELIZA na emuliranom IBM 7094 sustavu u prosincu 2024., čime je po prvi put nakon više od 60 godina omogućeno pokretanje autentičnog originalnog koda. Bit ovog postignuća nije samo nostalgija, rekonstrukcija je omogućila provjeru što originalna ELIZA doista jest, potvrđujući kako je bila funkcionalan, iako ograničen sustav koji je pomogao pokrenuti cijelo područje konverzacijske umjetne inteligencije [4].

Analiza restauriranog koda Ruperta i ostalih otkrila je nekoliko ključnih aspekata koji ranije nisu bili poznati. Pronađena verzija ima implementiran potpuni "*teaching mode*" (način podučavanja), spominjan u Weizenbaumovom radu iz 1966. samo jednom rečenicom, ali nikada detaljno opisan. Korisnici su mogli upisivanjem naredbi poput ADD, SUBST, APPEND, i RANK mijenjati pravila razgovora i spremati nove skripte. Također, otkrivena je značajna greška: ELIZA se "srušila" kada bi primila numerički unos (npr. "*you are 999 today*"), što autori tumače kao ironično doslovno shvaćanje Ada Lovelace uvida da računala mogu "djelovati na druge stvari osim brojeva". Važno je da tim nije popravio takve bugove, ostavili su kod autentičnim, točno 96% onakav kakav je pronađen, kako bi očuvali arheološku vrijednost nalaza. Time ELIZA postaje ne samo povijesni artefakt već i upozorenje da su mnogi kasniji opisi (npr. da je ELIZA bila napisana u Lispu) bili pogrešni, originalna je bila u MAD-SLIP jeziku, a Lisp verzija nastala je kasnije kao klon Bernieja Cosella, koji se zatim proširio preko ARPANeta [4].

2.2.2. PARRY

Al-Amin u svom radu obrađuju PARRY chatbot koji je 1972. razvio psihijatar Kenneth Colby na Stanfordu i predstavlja suprotnost ELIZI. Umjesto da glumi terapeuta, PARRY glumi paranoidnog shizofrenog pacijenta kako bi odvukao pažnju sa sebe i potaknuo korisnika na

dugačke, razrađene odgovore. U tekstu se naglašava kako je PARRY služio i kao alat za obuku psihijatara (vježbanje razgovora s pacijentima), ali i kao operativni model Colbyjeve teorije paranoje, tj. demonstracija kako "poremećena interpretacija znakova" može proizvoditi prepoznatljive obrasce paranoidnog govora i ponašanja [2].

Radom se ističe kako je PARRY jedan od prvih chatbotova koji se evaluirao kroz oblik "Turing testa" (imitacijska igra) radi procjene koliko uvjerljivo može oponašati ljudski razgovor. Ta linija evaluacije kasnije se nastavlja kroz natjecanja poput Loebner Prizea, gdje se botovi ocjenjuju po "prirodnosti" nakon kratkih razgovora. Poanta u tekstu je da su PARRY (i kasnije Loebner Prize) dio ranih pokušaja mjerenja konverzacijske "inteligencije" chatbotova prema tome koliko dobro simuliraju prirodan ljudski dijalog [2].

2.2.3. Racter

Racter (1983), kojeg su izradili Chamberlain i Etter, u tekstu se opisuje kao pionirski chatbot koji se posebno istaknuo po nasumičnom generiranju novog teksta, a ne samo biranju odgovora iz predložaka. Racter (skraćeno od *raconteur*, pripovjedač) mogao je tijekom razgovora stvarati jedinstvene rečenice i prozne ulomke koristeći pravila kontekstno-slobodne gramatike (*context-free grammar*) i element slučajnosti. Rad naglašava kako je rezultat često bio besmislen jer nije postojalo "pravo razumijevanje", ali da je sama proceduralna sposobnost generiranja originalnog teksta bila iznimno napredna i utjecajna za to vrijeme [2].

Posebno se ističe kako je autor povezan s Racterom (Chamberlain) objavio knjigu *The Policeman's Beard is Half Constructed* (1984), koja se predstavlja kao knjiga "autorski" proizvedena Racterovim generativnim mogućnostima, čime je dobio širu kulturnu vidljivost. Tekst napominje kako nije potpuno jasno u kojoj je mjeri proza doista bila u potpunosti računalno generirana, no važna je poruka da je Racter popularizirao ideju chatbotova koji mogu proizvoditi nov i originalan razgovorni sadržaj. Dodatno, rad daje primjer kako pojedini Racterovi fragmenti zvuče introspektivno i "ljudski" (npr. o snovima i željama), što je otvorilo pitanja o počovječenju i o tome kako AI može imitirati emocionalni i egzistencijalni izraz unatoč tome što nema ljudsko iskustvo [2].

2.3. Druga generacija - skriptni jezici

Nakon pionirskih sustava poput ELIZA, razvoj chatbotova ušao je u fazu dominacije skriptnih jezika i naprednog uparivanja uzoraka, što se može smatrati njihovom drugom generacijom. Umjesto jednostavnih pravila, ovi sustavi koriste strukturirane jezike poput AIML-a, RiveScripta i ChatScripta za izgradnju opsežnih baza znanja koje omogućuju chatbotovima da simuliraju kontekstualnu svjesnost i upravljaju tisućama varijacija korisničkih upita. Iako se ovi jezici i dalje oslanjaju na unaprijed definirane predloške umjesto na stvarno strojno učenje, oni su omogućili chatbotovima da postignu visoku razinu interaktivnosti i postanu prvi široko dostupni digitalni sugovornici na internetu [6].

2.3.1. Chatterbot u TINYMUD okruženju

Eleni i ostali u svom radu navode kako se pojam Chatterbot prvi put spominje 1991. godine, i to kao umjetni "igrač" u TINYMUD-u (*multiplayer real-time virtual world*), čija je primarna funkcija bila razgovor. Autori navode kako su mnogi stvarni (ljudski) igrači čak više voljeli razgovarati s Chatterbotom nego s drugim stvarnim igračem [6].

Eleni uspjeh Chatterbota u TINYMUD okruženju objašnjava kontekstom same virtualne zajednice: igrači su pretpostavljali kako su svi sugovornici ljudi pa bi sumnju izazvalo tek ako bi bot napravio veću/uočljiviju pogrešku. Drugim riječima, "prikrivenost" u okruženju gdje se očekuju ljudi smanjivala je prag za vjerovanje kako je sugovornik stvarna osoba [6].

2.3.2. ALICE i AIML

Eleni kao sljedeći veći korak u povijesti chatbotova navodi ALICE (*Artificial Linguistic Internet Computer Entity*), nastao 1995. kao prvi *online* chatbot inspiriran ELIZA. ALICE je bio temeljen na slaganju uzoraka, bez stvarnog razumijevanja cjelokupnog razgovora, ali s mogućnošću vođenja dugih razgovora na webu i pokrivanja praktički bilo koje teme [6].

Ipak, iako je kasnije poboljšavan i osvojio Loebner Prize za "najhumaniji" računalni program, autori ističu ograničenja: ALICE nije imao "inteligentne" značajke u smislu generiranja ljudski sličnih odgovora koji izražavaju emocije ili stavove. Dakle, unatoč velikoj bazi pravila i odgovora, nije bio emocionalno ni kognitivno "pametan" u modernom smislu [6].

Elena naglašava kako je ključna razlika između ALICE i ELIZA-e to što je ALICE razvijen uz novi jezik napravljen upravo za tu svrhu: AIML (*Artificial Intelligence Markup Language*). Također navode razliku u opsegu znanja: ALICE baza znanja imala je oko 41.000 predložaka i povezanih uzoraka, što je znatno više od ELIZA (oko 200 ključnih riječi i pravila) [6].

Autori kažu kako je AIML nastao u razdoblju 1995–2000 za izgradnju baze znanja chatbotova koji koriste pristup slaganja uzoraka. AIML je baziran na XML-u i *open-source*, "topics" sadrže "categories", a svaka kategorija predstavlja pravilo s uzorkom (korisnički *input*) i obrascem (odgovor bota). Sve se organizira u strukturu nazvanu Graphmaster (stablo), a AIML radi uparivanje uzoraka pretragom (*first depth search*) kako bi našao najbolje slaganje [6].

2.4. Treća generacija - komercijalni i praktični chatboti

Razvoj chatbotova prošao je put od ranih eksperimentalnih sustava do sofisticiranih asistenata koji su danas duboko integrirani u digitalni ekosustav. Ključni trenutak u toj evoluciji bio je prijelaz s izoliranih programa na masovne komunikacijske platforme, čime su chatbotovi postali dostupni širokom krugu korisnika u stvarnom vremenu. Kroz primjere poput pionirskog SmarterChilda i suvremenih rješenja na platformama kao što su Facebook Messenger, WhatsApp i WeChat, može se pratiti kako su ovi sustavi transformirali način na koji se dobijaju informacije, komunicira s brendovima i koriste društvene mreže [2] [6].

2.4.1. SmarterChild

SmarterChild je chatbot koji je razvila tvrtka ActiveBuddy 2001. godine i smatra se jednim od ranih pionira koji je chatbotove približio širokoj publici. Al-Amin u svom radu navodi kako je chatbot radio unutar *instant messaging* platformi i mogao je davati informacije o raznim temama na upit, što je pokazalo kako chatbot može funkcionirati kao svojevrsni "virtualni asistent" koji pronalazi činjenice i odgovara na korisnička pitanja [2].

Al-Amin ističe kako je posebno važno da se SmarterChild mogao dodati u MSN Messenger i AOL Instant Messenger te je na vrhuncu dosegao više od 30 milijuna korisnika. Time je "pionirski" otvorio put chatbotovima u mrežama za dopisivanje i kasnijim asistentima poput Appleova Siri-ja i Samsungova S Voice-a, jer je demonstrirao potencijal pružanja usluga kroz razgovorno sučelje direktno u okruženju poruka. Također je pokazao praktičnu vrijednost (npr. dohvat vijesti, vremena i burzovnih podataka iz baza) pa je pristup informacijama kroz razgovor umjesto kroz web-stranice označen kao važan korak prema AI sustavima koji služe ljudima putem dijaloga [2].

2.4.2. Web i društvene mreže

Chatbotovi na društvenim mrežama naglo su se proširili paralelno s rastom interneta i društvenih platformi, gdje mogu imati različite uloge: korisničku podršku, marketing i oglašavanje, zabavu te prikupljanje podataka. U radu Al-Amina se također naglašava kako se mogu koristiti i kao instrumenti "hibridnih prijetnji" s ciljem utjecaja na javno mnijenje, što pokazuje koliko su njihove primjene višestruke i društveno značajne [2].

U praksi se chatbotovi često integriraju izravno u brandove profile na društvenim mrežama kako bi pružali trenutnu podršku, automatski generirali sadržaj (npr. opise i hashtagove), omogućili personaliziran "razgovorni" kontakt s publikom, radili analizu sentimenta i prikupljali te organizirali podatke o korisnicima i trendovima. Kao primjer integracije na velikim platformama navodi se Meta, koja uvodi AI značajke te testira "Meta AI" asistenta dostupnog na WhatsAppu, Messengeru i Instagramu, uz naglasak na postupno uvođenje i zaštitne mehanizme [2].

2.5. Četvrta generacija - glasovni asistenti i konverzacijska sučelja

Četvrta generacija razvoja umjetne inteligencije u domeni komunikacije označava prijelaz s tekstualnih chatbotova na inteligentne glasovne asistente integrirane u svakodnevne uređaje. Ova faza, koja je započela početkom 2010-ih, donijela je revoluciju u interakciji čovjeka i računala omogućivši prirodan razgovor, prepoznavanje konteksta i upravljanje pametnim domovima. Za razliku od svojih prethodnika koji su se oslanjali na stroga pravila, moderni asistenti koriste duboko učenje i masovne baze podataka kako bi predvidjeli potrebe korisnika i pružili personalizirano iskustvo u stvarnom vremenu [2] [6].

2.5.1. Siri, Google Assistant, Cortana, Alexa

Pojava Siri na iPhoneu 4S 2011. godine postavila je temelje za komercijalnu upotrebu glasovnih asistenata, proizašavši iz ambicioznog vojnog projekta CALO koji je težio stvaranju kognitivnog agenta sposobnog za učenje. Siri je uvela koncept personaliziranog digitalnog pomoćnika koji ne samo da odgovara na pitanja, već i izvršava zadatke poput postavljanja podsjetnika ili slanja poruka, prilagođavajući se s vremenom jeziku i preferencijama korisnika. Iako je pionir, njezina ovisnost o internetskoj vezi i ograničena podrška za manje jezike ostali su izazovi u ranoj fazi razvoja [2] [6].

Ubrzo nakon Applea, Google je 2012. predstavio Google Now (kasnije Google Assistant), fokusirajući se na prediktivno pružanje informacija temeljem lokacije i povijesti pretraživanja, dok su Microsoft i Amazon 2014. godine ušli na tržište s Cortanom i Alexom. Dok je Cortana bila usmjerena na produktivnost unutar Windows ekosustava, Alexa je redefinirala koncept pametnog doma kroz Echo uređaje, omogućivši programerima stvaranje novih vještina putem Alexa Skills Kita [2] [6].

2.5.2. Opća arhitektura modernog asistenta - Siri

“Hey Siri” je *on-device* glasovni okidač koji koristi duboku neuronsku mrežu (DNN) za stalno osluškivanje aktivacijske fraze uz minimalnu potrošnju baterije. Sustav na iPhoneu radi u dvije faze: mali DNN na štedljivom koprocesoru (AOP) vrši početnu detekciju, a tek nakon toga glavni procesor s većim modelom potvrđuje upit. Cijeli proces prepoznavanja temelji se na analizi kratkih audio-okvira od 0,2 sekunde i vremenskoj integraciji rezultata kako bi se izračunala pouzdanost okidanja [7].

Nakon lokalne aktivacije, sustav provjerava identitet korisnika usporedbom s glasovnim profilom spremljenim na uređaju. Iako je sam detektor okidača u potpunosti lokalan radi privatnosti i brzine, većina daljnje obrade (glavno prepoznavanje govora i tumačenje upita) odvija se u oblaku. Ako Siri server utvrdi da početak govora ipak nije bio ispravan okidač, šalje signal uređaju da se vrati u stanje mirovanja [7].

2.6. Peta generacija - duboko učenje, transformeri i LLM chatbotovi

Peta generacija razvoja umjetne inteligencije i chatbot sustava označava prijelaz s krutih pravila na sustave koji uče iz golemih količina podataka, omogućujući prirodniju i kontekstualno svjesniju interakciju. Dok su raniji sustavi poput ELIZA-e koristili jednostavno prepoznavanje uzoraka, moderni pristupi temelje se na dubokom učenju i neuronskim mrežama koje simuliraju ljudske kognitivne procese. Ova evolucija omogućila je chatbotima da postanu ne samo digitalni asistenti za izvršavanje zadataka, već i empatični sugovornici sposobni za složenu komunikaciju [6].

2.6.1. Sekvencijski modeli

Sekvencijski modeli, poput rekurentnih neuronskih mreža (RNN) i mreža s dugom kratkoročnom memorijom (LSTM), predstavljali su ključni korak u omogućavanju chatbotima da zadrže kontekst razgovora. Za razliku od tradicionalnih mreža, ovi modeli koriste petlje koje omogućuju informacijama da perzistiraju, što je neophodno za razumijevanje rečenica u kojima značenje ovisi o prethodno izgovorenim riječima [6].

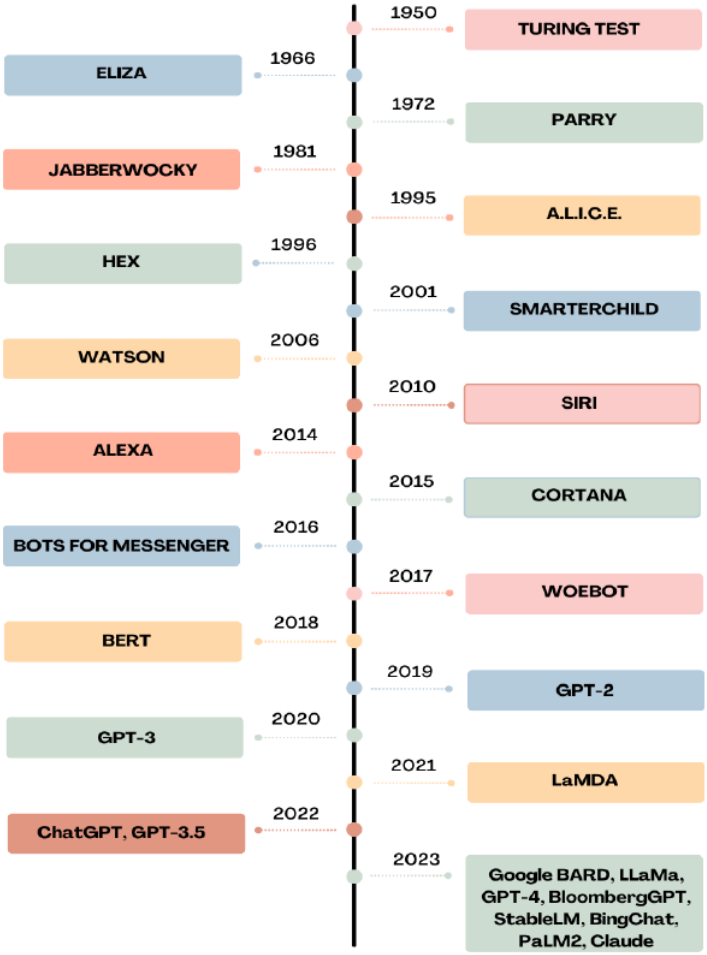
Međutim, unatoč njihovoj sposobnosti obrade sekvenci, ovi modeli često nailaze na poteškoće pri učenju dugoročnih ovisnosti u vrlo dugim tekstovima. Iako su LSTM i GRU (*Gated Recurrent Units*) značajno poboljšali preciznost u sustavima za odgovaranje na često postavljana pitanja (FAQ), njihova arhitektura i dalje zahtijeva sekvencijalnu obradu podataka, što ograničava brzinu treniranja na velikim skupovima podataka [6].

2.6.2. Transformer arhitektura

Transformer arhitektura donijela je revoluciju u obradi prirodnog jezika uvođenjem mehanizma "pažnje" (*attention mechanism*), koji omogućuje modelu da istovremeno promatra sve dijelove ulazne sekvence. Za razliku od prethodnih sekvencijskih modela, transformeri mogu paralelno obrađivati podatke, što drastično ubrzava proces treniranja i omogućuje rad s neusporedivo većim korpusima teksta [8].

Ova arhitektura omogućuje modelu da dinamički dodjeljuje važnost različitim riječima u rečenici, bez obzira na njihovu međusobnu udaljenost, čime se postiže dublje razumijevanje semantike i konteksta. Upravo je ovaj tehnološki skok postavio temelje za razvoj suvremenih velikih jezičnih modela koji dominiraju današnjim tržištem umjetne inteligencije [8].

Na Slici 2 se može vidjeti vremenski slijed pojavljivanja pojedinih chatbotova. Neki od njih nisu spomenuti u ovom radu pošto se u opisu povijesti chatbotova biralo najznačajnije zastupnike.



Slika 2: Vremenska linija chatbota [9]

2.6.3. Veliki jezični modeli i generativni chatboti (ChatGPT, Bard ...)

Veliki jezični modeli (LLM) predstavljaju vrhunac generativne umjetne inteligencije, koristeći milijarde parametara za predviđanje i generiranje teksta koji je često nerazlučiv od ljudskog. Sustavi poput ChatGPT-a ili Barda (sada Gemini) ne oslanjaju se na unaprijed definirane odgovore, već sintetiziraju nove rečenice u stvarnom vremenu, prilagođavajući se tonu i specifičnim zahtjevima korisnika kroz višestruke krugove dijaloga [8].

Iako ovi modeli pokazuju impresivne sposobnosti u rješavanju problema, kreativnom pisanju i programiranju, oni se i dalje suočavaju s izazovima poput “halucinacija” ili generiranja netočnih informacija. Unatoč tome, njihova integracija u svakodnevne alate i sposobnost učenja iz povratnih informacija korisnika čine ih najnaprednijim oblikom interakcije čovjeka i stroja do danas [8].

3. AI inženjerstvo - koncepti i ključne tehnike

Budući da se u ovom radu razmatra razvoj chatbota specifične namjene, pri čemu se njegova funkcionalnost temelji na postojećim velikim jezičnim modelima, može se zaključiti da je u svrhu izrade rješenja primijenjen pristup AI inženjerstva. Iz tog razloga, u ovom će se poglavlju detaljnije pojasniti sam pojam AI inženjerstva, kao i njegov položaj u kontekstu izgradnje i prilagodbe sustava temeljenih na velikim jezičnim modelima. Također će se obraditi i ostali ključni pojmovi i koncepti nužni za razumijevanje ovog područja, s posebnim naglaskom na one koji su relevantni za praktičnu izradu, konfiguraciju i evaluaciju chatbota namijenjenog određenoj domeni ili zadatku.

Ujedno, cilj poglavlja je pružiti teorijsku podlogu i jasnu terminološku osnovu za naredne dijelove rada, kako bi se korištene metode, tehnologije i dizajnerske odluke mogle razumjeti u širem okviru suvremenog razvoja AI rješenja. Na taj način se omogućuje lakše praćenje procesa izrade sustava, od odabira modela i načina integracije, preko prilagodbe ponašanja i ograničenja chatbota, do provjere njegove učinkovitosti u stvarnim uvjetima uporabe.

3.1. Pojam i uloga AI inženjerstva

AI inženjerstvo je specijalizirano područje u kojem stručnjaci koriste umjetnu inteligenciju i strojno učenje za razvoj aplikacija i sustava koji organizacijama pomažu povećati učinkovitost, smanjiti troškove i donositi bolje poslovne odluke. AI inženjeri razvijaju alate, sustave i procese koji omogućuju primjenu umjetne inteligencije u stvarnom svijetu [10].

Ključne odgovornosti AI inženjera:

- Stvaranje i upravljanje AI infrastrukturom za razvoj i produkciju
- Provođenje statističkih analiza za optimizaciju odlučivanja
- Automatizacija AI infrastrukture
- Izgradnja AI modela od nule
- Transformacija modela strojnog učenja u API-je koji se mogu integrirati s drugim aplikacijama [11]

Potrebne vještine:

- Programiranje (Python, R, Java, C++)
- Statistika i linearna algebra
- *Big data* tehnologije (Apache Spark, Hadoop, MongoDB)
- Algoritmi strojnog učenja i duboko učenje
- AI *frameworks* (TensorFlow, PyTorch, Keras) [10]

3.2. Temeljni modeli i veliki jezični modeli (LLM)

Današnji razvoj umjetne inteligencije više se ne vrti oko izrade malih, specijaliziranih programa za samo jedan zadatak. Umjesto toga, sve se temelji na moćnim, unaprijed obučenim sustavima koji služe kao univerzalna baza za sve ostalo. U središtu te promjene su temeljni modeli, a unutar njih svima poznati veliki jezični modeli (LLM). Dok su temeljni modeli širi pojam i mogu raditi sa slikama ili zvukom, LLM-ovi su tu tehnologiju doveli do savršenstva u radu s tekstom. Oni su pokazali kako AI može nevjerojatno dobro zaključivati i snalaziti se u raznim situacijama. Zapravo, ukoliko se želi razumjeti kako današnja AI funkcionira, najvažnije je shvatiti upravo taj odnos između te dvije skupine modela [12] [13].

3.2.1. Temeljni modeli

Temeljni modeli (često u literaturi nazivani i *Pretrained Foundation Models* - PFMs) definiraju se kao modeli velikih kapaciteta koji su pretrenirani na masivnim i heterogenim skupovima podataka, čime stječu opće reprezentacije primjenjive na niz različitih nizvodnih zadataka. Povijesni razvoj ovih modela, od ranih jezičnih reprezentacija poput BERT-a do kompleksnih multimodalnih sustava, ukazuje na trend u kojem se kroz povećanje računalne skale i količine podataka postiže visoka razina transfernog učenja. Ovakav pristup omogućuje da se jedan centralni model, uz minimalne prilagodbe, uspješno koristi u domenama koje variraju od računalnog vida do analize grafova, čime se radikalno mijenja proces dizajna inteligentnih sustava [13].

Ključna prednost temeljnih modela leži u njihovoj sposobnosti prilagodbe specifičnim kontekstima putem metoda kao što su fino podešavanje (*fine-tuning*), instrukcijsko učenje ili tehnike parametarski učinkovite adaptacije. Međutim, oslanjanje na jedan centralni model donosi i značajne izazove, prvenstveno u pogledu nasljeđivanja pristranosti iz pretreniranih podataka, sigurnosnih rizika te potrebe za rigoroznom evaluacijom robusnosti. Stoga se suvremena istraživanja ne fokusiraju samo na povećanje performansi, već i na razvoj mehanizama koji osiguravaju privatnost, etičnost i pouzdanost modela u produkcijskim okruženjima, čineći temeljne modele ne samo tehnološkim alatom, već i kompleksnim ekosustavom koji zahtijeva multidisciplinarni nadzor [12].

3.2.2. Veliki jezični modeli (LLM)

Veliki jezični modeli predstavljaju specijalizaciju temeljnih modela unutar domene prirodnog jezika, karakterizirani milijardama parametara i treniranjem na korpusima teksta koji obuhvaćaju značajan dio ljudskog znanja. Za razliku od tradicionalnih modela za obradu jezika, LLM-ovi pokazuju takozvana “izranjajuća” (emergentna) svojstva – sposobnosti poput rješavanja logičkih problema ili učenja iz konteksta koje se pojavljuju tek pri dosezanju određene kritične mase parametara i podataka. Evolucija ovih modela dovela je do pomaka s jednostavnog predviđanja sljedećeg tokena prema sustavima koji su usklađeni s ljudskim namjerama putem metoda poravnanja, čime postaju sposobni za kompleksnu interakciju i izvršavanje slo-

ženih korisničkih uputa [12], [13].

Arhitektonski temelj LLM-ova čini Transformer arhitektura, čija skalabilnost i paralelizacija omogućuju obradu ogromnih količina informacija uz korištenje naprednih varijanti mehanizama pažnje. Kako bi se prevladali visoki računalni troškovi treniranja i inferencije, istraživački fokus se sve više usmjerava prema optimizaciji arhitekture, uključujući efikasnije metode pozicijskog kodiranja i tehnike kompresije modela poput kvantizacije i "obrezivanja". Ovi tehnički napredci omogućuju širu dostupnost LLM-ova, pretvarajući ih iz isključivo istraživačkih artefakata u praktične komponente koje se mogu integrirati u različite softverske arhitekture uz zadržavanje visoke razine preciznosti [13].

U suvremenim primjenama, LLM-ovi se rijetko koriste kao izolirani sustavi. Umjesto toga, oni postaju jezgra širih okvira koji uključuju vanjske izvore znanja kroz metode poput generiranja potpomognutog dohvatom (*Retrieval-Augmented Generation* - RAG). Ovakva integracija rješava inherentne probleme modela, poput halucinacija i nedostatka ažurnih informacija, dok istovremeno omogućuje specijalizaciju za specifične domene bez potrebe za skupim ponovnim treniranjem. Konačno, proces evaluacije LLM-ova evoluirao je od jednostavnih metričkih pokazatelja prema kompleksnim *benchmark* testovima koji ispituju kognitivne sposobnosti, etičku usklađenost i praktičnu korisnost, čime se zaokružuje razvojni ciklus modela od sirovog pretreniranja do pouzdanog inteligentnog agenta [13] [12].

3.3. Tehničke osnove: transformer i *attention*

Prije pojave Transformera, rekurentne neuronske mreže poput LSTM i GRU bile su standardni izbor za obradu sekvenci teksta. Njihov glavni nedostatak bila je sekvencijalna obrada – svaki element sekvence morao se obraditi tek nakon prethodnog, što je onemogućavalo paralelizaciju i usporavalo treniranje na dugim sekvencama [14]. Transformer arhitektura riješila je ovaj problem potpunim odbacivanjem rekurencije i oslanjanjem isključivo na mehanizme pažnje (*attention*), čime je postala temeljna arhitektura za suvremene jezične modele [13].

3.3.1. Transformer arhitektura

Transformer se sastoji od dva glavna dijela: kodera (*encoder*) koji obrađuje ulazni tekst i dekodera (*decoder*) koji generira izlazni tekst. Koder pretvara ulaznu sekvencu u numeričke reprezentacije koje dekoder koristi za generiranje izlaza, element po element. Model je autoregresivan – pri generiranju svakog novog elementa koristi sve prethodno generirane elemente kao kontekst [14].

Ključna prednost ove arhitekture je sposobnost da istovremeno "vidi" sve dijelove ulazne sekvence, neovisno o njihovoj međusobnoj udaljenosti. Ovo omogućuje učinkovito hvatanje dugodosežnih ovisnosti u tekstu, što je kod rekurentnih mreža predstavljalo značajan izazov [14]. Također, budući da se sve pozicije mogu obraditi paralelno, Transformer omogućuje znatno brže treniranje na modernim GPU jedinicama [14].

3.3.2. *Attention* mehanizam

Attention mehanizam omogućuje modelu da pri obradi svakog elementa sekvence obrati pažnju na relevantne dijelove ulaza. Funkcionira tako da za svaki element (upit) izračuna koliko je povezan sa svim ostalim elementima (ključevima), a zatim te povezanosti koristi kao težine za kombiniranje vrijednosti. Matematički se to izražava sljedećom formulom [14]:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Dodati objašnjenje za Q, K i V: Q (Query) predstavlja trenutni element koji se obrađuje, K (Key) su svi elementi ulaza na koje se može obratiti pažnja, a V (Value) su vrijednosti povezane s tim elementima. Normalizacija dijeljenjem s $\sqrt{d_k}$ osigurava stabilnost gradijenata tijekom treniranja [14].

Višeglavi attention (*multi-head attention*) proširuje ovaj mehanizam izvođenjem više paralelnih *attention* operacija, svaka s različitim naučenim projekcijama. Rezultati se zatim spajaju, što modelu omogućuje istovremeno fokusiranje na različite vrste informacija – primjerice, jedna "glava" može pratiti sintaktičke odnose, dok druga prati semantičke veze [14].

3.4. Rad s LLM-ovima u aplikacijama: tokeni, kontekst i kontrola generiranja

Veliki jezični modeli obrađuju tekst na način koji se razlikuje od tradicionalnog pristupa čitanju i pisanju. Umjesto da rad s tekstom temelji na riječima ili rečenicama, LLM-ovi koriste tokene kao osnovne jedinice obrade i generiraju tekst sekvencijalno, token po token. Razumijevanje ovih procesa ključno je za učinkovitu primjenu LLM-ova u praktičnim aplikacijama.

3.4.1. Tokeni i tokenizacija

Token predstavlja osnovnu jedinicu teksta koju model obrađuje tijekom svog rada. Ovisno o postupku tokenizacije, token može biti znak, podriječ, simbol ili cijela riječ [14]. Tokenizacija je bitna faza predobrade koja tekst rastavlja na te nedjeljivije jedinice. Najčešće korištene metode tokenizacije kod LLM-ova uključuju *wordpiece*, kodiranje parom bajtova (BPE) i *unigramLM* [13].

Proces tokenizacije započinje tako što se ulazni tekst pretvara u niz tokena, pri čemu svaki token dobiva jedinstveni identifikator iz rječnika modela. Na primjer, riječ "tokenization" može biti podijeljena na više tokena poput "token", "ization" ili čak na manje jedinice ovisno o primijenjenom algoritmu. Vaswani i suradnici koriste naučene vektorske reprezentacije (*embeddings*) za pretvaranje ulaznih i izlaznih tokena u vektore dimenzije d_{model} [14]. Ove vektorske reprezentacije omogućuju modelu da razumije semantičke odnose između različitih tokena.

Model generira izlaz sekvencijalno, predviđajući sljedeći token na temelju prethodno generiranih tokena. Ovaj proces iterativnog generiranja znači da model mora za svaki novi

token izvršiti proračun kroz sve svoje slojeve, što izravno utječe na brzinu izvođenja i cijenu korištenja modela. Veći broj tokena u ulazu ili izlazu povećava računske zahtjeve i vrijeme potrebno za dobivanje odgovora. Različiti LLM-ovi koriste različite veličine rječnika tokena – primjerice, GPT-3 koristi približno 50.000 tokena, dok BLOOM koristi 250.000 tokena u svom rječniku [13].

3.4.2. Kontekst modela

Kontekst modela označava maksimalnu količinu teksta s kojom model može raditi odjednom. Može se zamisliti kao radna memorija modela – sve što model "vidi" i "pamti" tijekom obrade jednog zahtjeva ograničeno je veličinom ovog kontekstnog prozora. LLM-ovi su tradicionalno trenirani s ograničenim kontekstnim prozorima zbog računske složenosti mehanizma pažnje i visokih zahtjeva za memorijom [13].

Ograničenja konteksta predstavljaju važan praktični izazov. Model koji je treniran s ograničenom duljinom sekvence obično ne može uspješno generalizirati na veće duljine tijekom primjene. Proširivanje kontekstnog prozora tijekom finog podešavanja modela je spor, neučinkovit i računski skup proces [13]. Zbog toga su razvijene različite tehnike za proširenje konteksta poput interpolacije pozicijskih kodiranja i učinkovitijih mehanizama pažnje.

Gustim globalnim mehanizmom pažnje jedan je od glavnih uzroka ograničenja pri treniranju modela s većim kontekstnim prozorima. Korištenje učinkovitijih varijanti pažnje, poput lokalne, rijetke ili dilatirane pažnje, može značajno smanjiti računsku složenost. Na primjer, LongT5 koristi prijelaznu globalnu pažnju (TGlobal) koja kombinira lokalnu i globalnu pažnju, što omogućuje proširenje konteksta na 16.000 tokena [13].

Ograničenja konteksta motiviraju razvoj različitih tehnika za rad s dugim dokumentima. Tehnike poput sažimanja sadržaja, dijeljenja teksta na manje dijelove (*chunking*) i pristupa proširene generiranja s dohvaćanjem (RAG) omogućuju modelima pristup većoj količini informacija nego što stane u njihov kontekstni prozor. RAG pristup kombinira vanjsko pohranjivanje informacija s mogućnostima generiranja LLM-a, čime se omogućuje pristup ažurnim podacima i smanjuje pojava halucinacija [13].

3.4.3. Parametri generiranja: *temperature*, top-k i top-p

Nakon što model procesira ulazne tokene kroz svoje slojeve, dobiva se izlazna raspodjela vjerojatnosti za sve moguće sljedeće tokene. Transformer arhitektura koristi linearnu transformaciju i *softmax* funkciju za pretvaranje izlaza dekodera u vjerojatnosti sljedećeg tokena [14]. *Softmax* funkcija normalizira rezultate tako da zbroj svih vjerojatnosti bude jednak jedan, čime se dobiva valjana vjerojatnosna raspodjela.

Različiti parametri kontroliraju kako model bira sljedeći token iz ove raspodjele vjerojatnosti, što omogućuje podešavanje ravnoteže između kreativnosti i dosljednosti generiranog teksta.

Temperature je parametar koji kontrolira stupanj stohastičnosti u procesu generiranja.

Temperature modificira vjerojatnosnu raspodjelu prije primjene *softmax* funkcije tako što dijeli logite s vrijednošću temperature. Niže vrijednosti temperature (bliže nuli) čine raspodjelu oštrijom – model postaje konzervativniji i češće bira tokene s najvećom vjerojatnosti, što rezultira predvidljivijim i stabilnijim odgovorima. Više vrijednosti temperature (veće od jedan) zaglađuju raspodjelu vjerojatnosti, dajući više šanse tokenima s nižom vjerojatnosti, što potiče veću raznolikost i kreativnost u generiranom tekstu. Temperatura se često koristi pri treniranju modela za generiranje prirodnog jezika kako bi se poboljšala kvaliteta prijevoda ili generiranih sekvenci [15].

Top-k uzorkovanje ograničava izbor sljedećeg tokena na k najvjerojatnijih kandidata iz raspodjele vjerojatnosti. Nakon što se identificira prvih k tokena s najvećom vjerojatnosti, model uzorkuje sljedeći token samo iz ovog ograničenog skupa, zanemarujući sve ostale mogućnosti. Ovaj pristup spriječava model da bira vrlo nevjerojatne tokene koji mogu narušiti koherentnost teksta, ali istovremeno omogućuje određenu raznolikost među najvjerojatnijim opcijama. Vrijednost parametra k određuje koliko će kandidata biti razmatrano – manja vrijednost rezultira konzervativnijim odabirom, dok veća vrijednost dopušta više varijabilnosti [15].

Top-p uzorkovanje, poznato i kao *nucleus sampling*, predstavlja dinamičniji pristup filtriranju kandidata. Umjesto fiksnog broja tokena, ovaj metoda odabire najmanji skup tokena čija kumulativna vjerojatnost prelazi zadani prag p (obično između 0.9 i 0.95). To znači da model automatski prilagođava veličinu skupa kandidata ovisno o distribuciji vjerojatnosti – kada je jedan token dominantan, skup će biti manji, a kada je više tokena približno jednako vjerojatno, skup će biti veći. Ovaj pristup omogućuje modelu da prilagodi razinu konzervativnosti ovisno o pouzdanosti predikcije u svakom koraku generiranja [15].

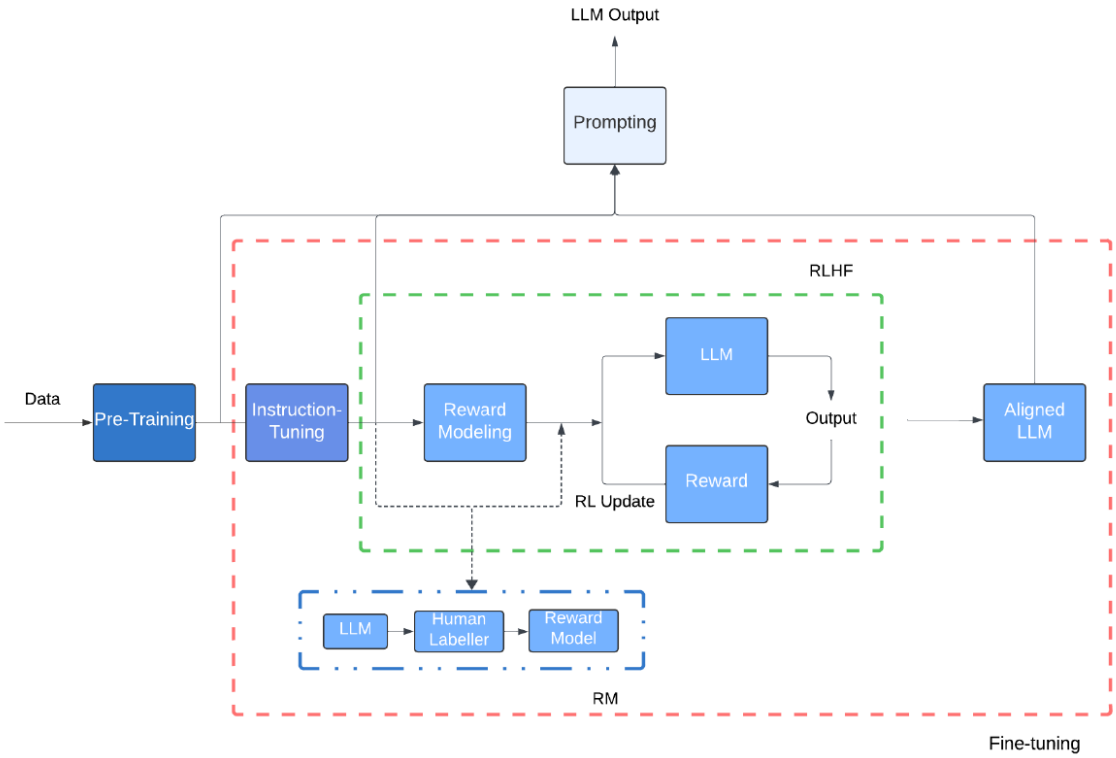
Kombinacija ovih parametara omogućuje fino podešavanje ponašanja modela prema specifičnim potrebama primjene. Za zadatke koji zahtijevaju preciznost i dosljednost, poput generiranja programskog koda ili formalnih dokumenata, koriste se niže vrijednosti temperature i konzervativnije strategije uzorkovanja. S druge strane, za kreativne zadatke poput pisanja priča ili generiranja ideja, preporučuju se više vrijednosti temperature i fleksibilnije strategije uzorkovanja koje potiču raznolikost u generiranom sadržaju.

3.5. Prilagodba ponašanja bez treniranja: prompt engineering i *in-context* učenje

Tradicionalni pristup prilagodbi jezika modela za specifične zadatke ili domene zahtijeva postupak finog podešavanja (*fine-tuning*), pri čemu se parametri prethodno treniranog modela ažuriraju korištenjem zadataka specifičnih podataka. Iako ovaj pristup omogućava značajno poboljšanje performansi za ciljane aplikacije, proces je računski skup i zahtijeva pristup kvalitetnim označenim podacima za svaki novi zadatak. Dodatno, fino podešavanje može dovesti do katastrofičnog zaboravljanja (*catastrophic forgetting*), gdje model gubi znanje stečeno tijekom inicijalnog treniranja [15]. Ova ograničenja motivirala su razvoj alternativnih pristupa koji omogućavaju prilagodbu ponašanja modela bez potrebe za izmjenom njegovih parametara.

Veliki jezični modeli pokazuju sposobnost adaptacije putem promjena u načinu pos-

tavljanja upita, odnosno kroz tehniku koja se naziva *prompting* ili *prompt engineering*. Kao što je prikazano na Slici 3, prompting predstavlja metodu postavljanja upita treniranim LLM-ovima koja omogućava prilagodbu modela bez finog podešavanja parametara. Ova neočekivana sposobnost velikih jezičnih modela omogućava korisnicima da usmjeravaju ponašanje modela isključivo kroz pažljivo oblikovane tekstualne upute i primjere unutar samog upita. Dva ključna pristupa prilagodbi bez treniranja su prompt engineering, koji se fokusira na strukturiranje i formuliranje upita, te *in-context* učenje, pri čemu model uči iz demonstracija prisutnih u promptu [15], [16].



Slika 3: Dijagram toka različitih faza razvoja LLM-a od pred-treniranja do korištenja. RL označava *reinforcement learning*, RM *reward modeling*, a RLHF *reinforcement learning with human feedback* [12]

3.5.1. Prompt engineering

Prompt engineering predstavlja disciplinu oblikovanja uputa za velike jezične modele s ciljem postizanja konzistentnijih i kvalitetnijih rezultata. Ovaj pristup omogućava prilagodbu ponašanja modela bez potrebe za dodatnim treniranjem ili izmjenom parametara. Prompt može sadržavati različite elemente poput definiranja uloge modela, specificiranja željenog formata izlaza, postavljanja pravila ponašanja ili davanja kontekstualnih informacija [17].

Istraživanja pokazuju kako kvaliteta prompta značajno utječe na performanse modela. Dobro strukturiran prompt uključuje jasne i detaljne upute, može sadržavati pozitivne i negativne primjere željenog ponašanja te specificirati očekivanu duljinu ili format odgovora. Dodatno, prompting može uključivati tehnike poput poticanja korak-po-korak rasuđivanja, gdje se model navodi da eksplicitno prikaže svoj proces razmišljanja prije davanja konačnog odgovora [13].

Prompt engineering postaje posebno važan u kontekstu *zero-shot* učenja, gdje model treba odgovoriti na zadatak koji nije vidio tijekom treniranja. Veliki jezični modeli pokazuju sposobnost *zero-shot* učenja, što znači da mogu odgovarati na upite bez ikakvog prethodnog primjera u promptu. Ova sposobnost omogućava modelima generalizaciju na nove zadatke isključivo na temelju jasno formuliranih uputa [13].

3.5.2. Sistemski i korisnički prompt

Moderni sustavi velikih jezičnih modela koriste distinkciju između sistemskog i korisničkog prompta kako bi omogućili fleksibilnije i kontroliranije interakcije. Sistemski prompt definira opće ponašanje modela kroz cijelu interakciju i postavlja osnovne smjernice za način rada. Ovaj prompt obično ostaje konstantan tijekom sesije i određuje kontekst u kojem model interpretira korisničke zahtjeve. Sistemski prompt može uključivati informacije o ulozi modela, stilskim preferencijama, ograničenjima u ponašanju ili specifičnim domenskim znanjima [16].

Korisnički prompt, s druge strane, predstavlja konkretan zahtjev ili upit koji korisnik postavlja u određenom trenutku. Ovaj prompt je dinamičan i mijenja se s svakim zahtjevom korisnika. Model kombinira informacije iz sistemskog i korisničkog prompta kako bi generirao odgovarajući odgovor. Ova arhitektura omogućava održavanje konzistentnog ponašanja modela (kroz sistemski prompt) dok istovremino omogućava fleksibilnost u odgovaranju na raznolike korisničke upite [13].

Važnost ove distinkcije posebno dolazi do izražaja u aplikacijama gdje je potrebno održavanje određenog stila komunikacije ili poštivanje specifičnih ograničenja kroz cijelu konverzaciju. Na primjer, sistemski prompt može definirati da model uvijek daje odgovore u određenom profesionalnom tonu ili da izbjegava određene teme, dok korisnički promptovi postavljaju konkretna pitanja unutar tog okvira.

3.5.3. *In-context* learning

In-context learning ili učenje u kontekstu predstavlja ključnu sposobnost velikih jezičnih modela koja im omogućava prilagodbu novim zadacima bez izmjene parametara modela. Ovaj fenomen se manifestira kroz sposobnost modela da uči iz primjera koji su dani unutar samog prompta. Kada se modelu pruži nekoliko primjera ulazno-izlaznih parova za određeni zadatak, model može generalizirati taj obrazac i primijeniti ga na nove ulaze [18].

Few-shot learning, kao specifičan oblik *in-context* učenja, pokazuje se posebno efektivnim. Istraživanja na GPT-3 modelu pokazala su kako performanse modela u *few-shot* scenariju premašuju performanse u *zero-shot* scenariju, što sugerira da veliki jezični modeli funkcioniraju kao meta-učenici (*meta-learners*) - modeli koji uče kako učiti iz malog broja primjera [18]. Broj primjera koji se daju modelu može varirati, ali čak i jedan ili dva dobro odabrana primjera mogu značajno poboljšati performanse na novom zadatku.

Format *in-context* učenja tipično uključuje strukturirane demonstracije gdje se svakom primjeru prilaže ulaz i očekivani izlaz. Model zatim koristi ove demonstracije kao reference za

generiranje odgovora na novi upit. Važno je napomenuti da kvaliteta i relevantnost odabranih primjera značajno utječu na performanse modela. Istraživanja pokazuju da primjeri trebaju biti što sličniji ciljanom zadatku te da redoslijed primjera također može utjecati na rezultate [13].

Ova tehnika omogućava privremeno usmjeravanje modela bez potrebe za skupim postupkom *fine-tuninga*. Parametri modela ostaju nepromijenjeni, a prilagodba se postiže isključivo kroz sadržaj prompta. To čini *in-context* učenje izuzetno praktičnim pristupom za brzu prilagodbu modela različitim domenama i zadacima, posebno u scenarijima gdje je dostupno samo nekoliko primjera ili gdje je potrebna česta promjena zadataka [13].

Međutim, važno je napomenuti kako *in-context* učenje ima svoja ograničenja. Svi primjeri moraju stati u kontekstni prozor modela, što ograničava broj demonstracija koje se mogu pružiti. Dodatno, performanse ovog pristupa ovise o veličini modela - veći modeli pokazuju značajno bolje sposobnosti *in-context* učenja nego manji modeli [18].

3.6. Prilagodba modela treniranjem i optimizacija primjene

Veliki jezični modeli treniraju se na ogromnim količinama tekstualnih podataka kako bi stekli opće jezične sposobnosti, no njihova primjena u stvarnim sustavima često zahtijeva dodatne korake optimizacije. Prilagodba modela specifičnim domenama ili zadacima, smanjenje računalnih zahtjeva i usklađivanje s ljudskim preferencijama ključni su izazovi pri integraciji LLM-ova u proizvodne sustave. Dok su tehnike poput *prompt engineeringa* i *in-context* učenja korisne za prilagodbu ponašanja bez izmjene parametara modela, postoje scenariji u kojima je potrebno izravno modificirati sam model kako bi se postigla stabilnija i učinkovitija izvedba. Ovaj pristup omogućuje optimizaciju modela za specifične zadatke uz kontrolu nad resursnim zahtjevima i usklađenost s očekivanim ponašanjem [12].

Tehnike prilagodbe modela kreću se od potpunog ponovnog treniranja na domenskim podacima do metoda koje selektivno modificiraju mali podskup parametara, kao i postupaka kompresije koji smanjuju veličinu modela uz minimalan gubitak performansi. *Fine-tuning* omogućuje specijalizaciju modela za određene zadatke ili domene, dok *parameter-efficient* metode postižu slične rezultate uz znatno manje računalnih resursa. Dodatno, tehnike kao što su *preference fine-tuning* usklađuju izlaze modela s ljudskim vrijednostima i preferencijama, što je ključno za primjenu u stvarnim sustavima gdje je sigurnost i korisnost prioritet. Metode kvantizacije i destilacije pak omogućuju razvoj manjih, bržih verzija modela pogodnih za implementaciju na uređajima s ograničenim resursima, čime se proširuje dostupnost i praktična primjenjivost LLM tehnologije [12].

3.6.1. Fine-tuning i PEFT

Fine-tuning predstavlja proces dodatnog treniranja prethodno naučenog jezičnog modela na skupu podataka specifičnom za određeni zadatak ili domenu. Za razliku od pretreniranja koje model izlaže širokom spektru općih jezičnih obrazaca, *fine-tuning* omogućuje modelu da specijalizira svoje znanje za uže definirane aplikacije. Ovaj proces nadograđuje postojeće

reprezentacije naučene tijekom pretreniranja, prilagođavajući ih posebnostima ciljane domene ili zadatka. Kod tradicionalnog *fine-tuninga* ažuriraju se svi parametri modela, što može rezultirati stabilnijim i pouzdanijim ponašanjem u specifičnim scenarijima, ali zahtijeva značajne računalne resurse i memoriju proporcionalnu veličini modela [12].

Tradicionalni pristup *fine-tuningu*, iako učinkovit, suočava se s praktičnim izazovima kada se primjenjuje na moderne LLM-ove koji sadrže milijarde parametara. Ažuriranje svih parametara tijekom treniranja zahtijeva pohranu gradijenata i optimizacijskih stanja za svaki parametar, što multiplicira memorijske zahtjeve i čini proces računalno zahtjevnim. Dodatno, potpuni *fine-tuning* nosi rizik od "katastrofičkog zaboravljanja" gdje model gubi opće jezične sposobnosti naučene tijekom pretreniranja dok se prilagođava novom zadatku. Ovi izazovi motivirali su razvoj pristupa koji omogućuju učinkovitu prilagodbu modela uz minimiziranje računalnih i memorijskih troškova [12].

Parameter-Efficient Fine-Tuning (PEFT) metode predstavljaju alternativni pristup koji postiže prilagodbu modela modificirajući samo mali podskup parametara ili dodavanjem dodatnih adaptivnih komponenti. Jedna od najpoznatijih PEFT tehnika je LoRA (*Low-Rank Adaptation*) koja umjesto ažuriranja svih težina matrica u modelu, dodaje male trenable matrice niskog ranga koje se kombiniraju s originalnim zamrznutim težinama. Ovaj pristup drastično smanjuje broj parametara koje je potrebno trenirati - tipično za nekoliko redova veličine - što rezultira znatno nižim memorijskim zahtjevima i bržim treniranjem, uz zadržavanje kvalitete prilagodbe usporedive s potpunim fine-tuningom. PEFT metode također omogućuju zadržavanje više različitih adaptacija istog osnovnog modela za različite zadatke, što je praktično u scenarijima gdje je potrebno podržati više specijaliziranih aplikacija [12].

3.6.2. Preference fine-tuning

Preference fine-tuning predstavlja naprednu tehniku koja omogućuje usklađivanje ponašanja jezičnih modela s ljudskim vrijednostima i preferencijama kroz učenje iz komparativnih ocjena. Za razliku od klasičnog *supervised fine-tuninga* koji model uči iz parova ulaza i željenih izlaza, *preference fine-tuning* koristi podatke u obliku usporedbi između različitih generiranih odgovora na isti upit. Ljudski korisnici ocjenjuju koji od ponuđenih odgovora je korisniji, sigurniji, informativniji ili prikladniji, a model uči da preferira generiranje odgovora koji su bolje ocijenjeni. Ovaj pristup temelji se na ideji da je ljudima često lakše usporediti kvalitetu dvaju odgovora nego precizno definirati što čini idealan odgovor, što ga čini pogodnim za domene gdje je kvaliteta subjektivna ili je teško uspostaviti formalizam [12].

Najpoznatiji pristup preference fine-tuningu je *Reinforcement Learning from Human Feedback* (RLHF) koji kombinira *supervised learning* s metodama potkrepljivanja. Proces započinje skupljanjem usporednih ocjena ljudi gdje anotatori rangiraju različite odgovore modela na iste upite. Iz tih ocjena trenira se *reward* model koji uči predvidjeti ljudske preferencije, tj. procjenjuje koja vrsta odgovora će biti pozitivno ocijenjena. Konačno, koristeći algoritme poput *Proximal Policy Optimization* (PPO), osnovni jezični model se optimizira da maksimizira očekivanu nagradu prema naučenom *reward* modelu, efektivno učeći generirati odgovore usklađene s ljudskim preferencijama. Ovaj pristup pokazao se ključnim u razvoju conversational AI sus-

tava poput ChatGPT, omogućivši modele koji ne samo da generiraju koherentne odgovore, već su također korisni i pružaju odgovore u stilu koji korisnici preferiraju [16].

Preference fine-tuning adresira fundamentalan izazov u razvoju AI sustava - usklađivanje tehničke sposobnosti s ljudskim vrijednostima. Kroz iterativni proces prikupljanja povratnih informacija i učenja iz njih, modeli postaju sve prikladniji za interakciju s ljudima u stvarnim aplikacijama. Ovaj pristup omogućuje korekciju neželjenih ponašanja poput generiranja štetnog sadržaja, nedosljednosti u odgovorima ili neprimjerenog tona, što je teško postići samo kroz *supervised learning*. Dodatno, *preference fine-tuning* omogućava kontinuiranu prilagodbu modela kako se ljudske preferencije i društveni standardi mijenjaju tijekom vremena, čineći sustave fleksibilnijima i odgovornijima u dugoročnoj primjeni [12].

3.6.3. Kvantizacija i destilacija

Kvantizacija predstavlja tehniku kompresije modela koja smanjuje preciznost numeričke reprezentacije parametara, tipično s 32-bitne ili 16-bitne aritmetike s pomičnim zarezom na 8-bitne ili čak 4-bitne reprezentacije. Ovaj proces značajno smanjuje memorijske zahtjeve modela - model kvantiziran na 8 bitova zauzima četiri puta manje memorije nego originalni 32-bitni model. Osim smanjenja memorijskog otiska, kvantizacija također ubrzava izvođenje jer operacije s nižom preciznošću se može brže izvršavati, posebno na specijaliziranom hardveru. Moderna istraživanja pokazuju kako je moguće kvantizirati veliki dio parametara LLM-a bez značajnog gubitka performansi, zahvaljujući redundanciji u naučenim reprezentacijama. Postoje različiti pristupi kvantizaciji, uključujući *post-training* kvantizaciju gdje se već trenirani model komprimira, te *quantization-aware training* gdje se kvantizacija uzima u obzir tijekom samog procesa treniranja [12].

Destilacija znanja (*knowledge distillation*) predstavlja komplementarni pristup kompresiji u kojem se manji model, nazvan *student* model, trenira da imitira ponašanje većeg, *teacher* modela. Za razliku od kvantizacije koja zadržava arhitekturu modela ali smanjuje preciznost, destilacija stvara fundamentalno manji model s manje slojeva ili parametara. *Student* model uči ne samo iz oznaka podataka (*ground truth labels*), već i iz "mekih" predviđanja (*soft predictions*) *teacher* modela - distribucija vjerojatnosti koju *teacher* generira nad mogućim izlazima. Ove distribucije sadrže bogatije informacije od tvrdih oznaka, pokazujući relativne odnose između različitih mogućih odgovora. Na primjer, pri generiranju sljedeće riječi, *teacher* model može dodijeliti vjerojatnost 0.7 jednoj riječi, 0.2 drugoj i 0.1 trećoj - ove relativne vjerojatnosti pružaju *student* modelu uvid u nijanse jezika koje je *teacher* model naučio [12].

Kombinacija kvantizacije i destilacije omogućuje razvoj izuzetno učinkovitih modela pogodnih za implementaciju u okruženjima s ograničenim resursima, poput mobilnih uređaja ili *edge computing* sustava. *Student* model može se dodatno kvantizirati nakon destilacije, postižući kumulativni učinak obje tehnike. Ovakvi komprimirani modeli omogućuju demokratizaciju AI tehnologije, čineći napredne jezične sposobnosti dostupnima na širem spektru uređaja i aplikacija. Iako komprimirani modeli tipično pokazuju određeno smanjenje performansi u odnosu na originalne velike modele, pažljivom primjenom ovih tehnika moguće je zadržati prihvatljivu kvalitetu uz drastično smanjenje računalnih zahtjeva, što je ključno za praktičnu primjenu LLM-

ova u stvarnom svijetu [12].

3.7. RAG: povezivanje modela s vanjskim znanjem

3.7.1. Osnovna ideja RAG-a

- *Retrieval-Augmented Generation* (RAG) kombinira dohvat relevantnih dokumenata iz baze znanja i generiranje odgovora na temelju dohvaćenog sadržaja.
- Cilj je povećati aktualnost i pouzdanost odgovora te smanjiti oslanjanje na implicitno znanje iz treniranja.

3.7.2. Embeddings i vektorske baze podataka

- *Embeddings* preslikavaju tekst u vektore tako da semantički slični izrazi budu blizu u vektorskom prostoru.
- Vektorske baze omogućuju efikasno pretraživanje najsličnijih vektora upitu, tj. semantički dohvat.

3.7.3. *Chunking*

- *Chunking* dijeli dokumente na manje cjeline radi lakšeg pretraživanja i zbog ograničenja konteksta modela.
- Odabir veličine i strategije segmentacije utječe na kvalitetu dohvaćenih informacija i konačni odgovor.

3.8. Agenti i alati

- Agent predstavlja pristup u kojem sustav planira korake i izvršava radnje prema cilju (npr. višekorakno rješavanje zadatka).
- Integracija alata omogućuje modelu izvođenje stvarnih radnji (npr. dohvat podataka, poziv API-ja), a ne samo generiranje teksta.

3.9. Evaluacija i operativni aspekti (inferenca, latencija, metrike)

3.9.1. Inferenca: *online* i *batch*

- *Online* inferenca isporučuje odgovore u stvarnom vremenu.

- *Batch* inferenca obrađuje veći broj zahtjeva izvan realnog vremena (npr. periodički poslovi).

3.9.2. Latencija

- Latencija je vrijeme do prvog korisnog izlaza i predstavlja kritičan faktor korisničkog iskustva u interaktivnim sustavima.

3.9.3. Benchmarkovi i metrike

- Benchmarkovi služe za usporedbu modela na standardiziranim zadacima (npr. razumijevanje teksta, matematika, kodiranje).
- Primjeri metrika:
  - **Perplexity** (mjera "iznenađenja" modela nad tekstom),
  - **BLEU** (podudaranje s referencom, česta u prijevodu),
  - **ROUGE** (naglasak na *recall*, česta u sažimanju).
- **LLM-as-Judge** koristi model za ocjenu izlaza drugog modela prema zadanim kriterijima (npr. praćenje uputa, jasnoća, relevantnost).

3.10. (Opcionalno) Model Context Protocol (MCP)

- MCP predstavlja standardizirani način povezivanja modela, aplikacija i alata kroz jedinstveno sučelje, s ciljem smanjenja potrebe za prilagođenim integracijama.

4. Evaluacija Chatbotova i halucinacije

5. Zaključak

Popis literature

[1] J. Naisbitt, *Megatrends: 10 new directions transforming our lives*. Warner Books, 1982., str. 17.

[2] M. Al-Amin, M. S. Ali, A. Salam i dr., „History of generative Artificial Intelligence (AI) chatbots: past, present, and future development," *arXiv preprint arXiv:2402.05122*, 2024.

[3] A. Turing, „A Quarterly review," *Machine Intelligence: Perspectives on the Computational Model*, sv. 1, str. 1, 1998.

[4] R. Lane, A. Hay, A. Schwarz, D. M. Berry i J. Shrager, „ELIZA Reanimated: The world's first chatbot restored on the world's first time sharing system," *arXiv preprint arXiv:2501.06707*, 2025.

[5] D. F. Murad, A. G. Iskandar, E. Fernando, T. S. Octavia i D. E. Maured, „Towards smart LMS to improve learning outcomes students using LenoBot with natural language processing," *2019 6th International Conference on Information Technology, Computer and Electrical Engineering (ICITACEE)*, IEEE, 2019., str. 1–6.

[6] E. Adamopoulou i L. Moussiades, „Chatbots: History, technology, and applications," *Machine Learning with applications*, sv. 2, str. 100 006, 2020.

[7] S. Team, *Hey Siri: An On-device DNN-powered Voice Trigger for Apple's Personal Assistant*, `https://machinelearning.apple.com/research/hey-siri`, [Pristupljeno 09.01.2026], 2017.

[8] A. Toosi, A. G. Bottino, B. Saboury, E. Siegel i A. Rahmim, „A brief history of AI: how to prevent another winter (a critical review)," *PET clinics*, sv. 16, br. 4, str. 449–469, 2021.

[9] T. T. Stoyanova i S. R. Stoyanov, „Chatbots: History, technology and application in higher education," *Annual of Konstantin Preslavsky University of Shumen Faculty of mathematics and informatics*, 2023.

[10] C. Staff, *What is an AI Engineer? (And how to become one)*, `https://www.coursera.org/articles/ai-engineer`, [Pristupljeno 13.01.2026], 2025.

[11] M. Wyss, *AI Engineering For Beginners in 14 Minutes*, `https://youtu.be/nUqgjQs5pJk?si=GyE7MNiLfHiq6Zrm`, [Pristupljeno 13.01.2026], 2025.

[12] H. Naveed, A. U. Khan, S. Qiu i dr., „A comprehensive overview of large language models," *ACM Transactions on Intelligent Systems and Technology*, sv. 16, br. 5, str. 1–72, 2025.

[13] C. Zhou, Q. Li, C. Li i dr., „A comprehensive survey on pretrained foundation models: A history from bert to chatgpt,” *International Journal of Machine Learning and Cybernetics*, sv. 16, br. 12, str. 9851–9915, 2025.

[14] A. Vaswani, N. Shazeer, N. Parmar i dr., „Attention is all you need,” *Advances in neural information processing systems*, sv. 30, 2017.

[15] C. Huyen, *AI Engineering*. USA: O'Reilly Media, 2025., ISBN: 978-1801819312.

[16] L. Ouyang, J. Wu, X. Jiang i dr., „Training Language Models to Follow Instructions with Human Feedback,” *arXiv preprint arXiv:2203.02155*, 2022.

[17] E. Saravia, *Prompt Engineering Guide*, 12. 2022.

[18] T. B. Brown, B. Mann, N. Ryder i dr., „Language Models are Few-Shot Learners,” *arXiv preprint arXiv:2005.14165*, 2020.

Popis slika

Prilozi

1. Prilog 1