Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/26BE15Dokic

RAZVOJ ČET BOTA SA DUGOROČNIM PAMĆENJEM I RAZUMEVANJEM KONTEKSTA

DEVELOPMENT OF A CHAT BOT WITH LONG-TERM MEMORY AND CONTEXT UNDERSTANDING

Dejan Dokić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIČKO I RAČUNARSKO INŽENJERSTVO

Kratak sadržaj – *U ovom radu će biti predstavljen razvoj čet bota koji je u stanju da dugoročno pamti i razume kontekst započete prepiske. Rad uključuje opis korišćenih tehnika i načina na koji su povezani zarad dobijanja sistema koji simulira ljudsku interakciju.*

Ključne reči: *čet botovi, vektorske baze podataka, dugoročna memorija, jezički modeli*

Abstract – *This work will present the development of a chatbot capable of long-term memory and understanding the context of initiated correspondence. The work includes a description of the techniques used and the way they are connected to achieve a system that simulates human interaction.*

Keywords: *chatbots, vector databases, long-term memory, language models*

1. UVOD

Čet botovi predstavljaju poseban tip digitalnih sistema osmišljenih sa ciljem da simuliraju ljudsku interakciju putem tekstualnog ili glasovnog kanala. Funkcionišu kao agenti pružajući konstantnu podršku u bilo koje doba dana, i na taj način doprinose smanjenju intenziteta potrebne radne snage kao i operativnih troškova. U stanju su da istovremeno obrade veliki broj upita, komuniciraju sa eksternim servisima ako se javi potreba za time i u svakom momentu pruže doslednu informaciju.

Prvobitno, čet botovi su bili ograničeni time što su imali predefinisan skup odgovora i na osnovu korisničkog unosa, najrelevantniji odgovor iz skupa svih odgovora bi bio vraćen korisniku. Vremenom se javila potreba za složenijim i preciznijim odgovorima, što je zahtevalo primenu veštačke inteligencije (engl. *Artificial Intelligence*, skraćeno *AI*), obrade prirodnog jezika (engl. *Natural Language Processing*, skraćeno *NLP*) i algoritama mašinskog učenja (engl. *Machine Learning*, skraćeno *ML*). Na taj način je postignuta personalizovana komunikacija u realnom vremenu sa mogućnošću učenja iz same interakcije i neprestanog poboljšanja kvaliteta samog odgovora.

Iako je samo dobijanje doslednog odgovora isključivo na osnovu pitanja postavljenog od strane korisnika veliki uspeh, još precizniji i kvalitetniji odgovor bi bio moguć ako bi čet bot bio u mogućnosti da u proces kreiranja odgovora uzme u obzir što je moguće veći nivo razumevanja konteksta, baziranih na osnovu prethodnih poruka. Na ovaj način, korisniku bi bilo omogućeno da u više različitih iteracija komunikacije sa čet botom zadrži kontekst prethodnih dopisivanja, što ujedno prouzrokuje tačnijim i približnijim odgovorom samom korisniku. Kreiranje takvog čet bota koji je u mogućnosti da dugoročno pamti i razume kontekst dosadašnje prepiske predstavlja veoma važan korak i ujedno najveći izazov u daljem razvoju ove oblasti.

2. KORIŠĆENE TEHNOLOGIJE

Zarad razumevanja načina funkcionisanja samog čet bota, neophodno je pre svega objasniti pojmove: *GPT* modela, *Pinecone* vektorske baze podataka, *LangChain* integracionog okvira i *Zep* dugoročne memorije. Ovo poglavlje ima za cilj da sa teorijske strane objasni prethodno navedene koncepte sa naglaskom na komponente svake tehnologije koje će se koristiti u procesu implementacije čet bota.

2.1. Generativni Unapred Obučeni Transformator (*GPT*)

Generativni unapred obučeni transformator (*GPT*) predstavlja jezički model[1] (engl. *language model*, skraćeno *LM*) razvijen od strane OpenAI-a[2] koji je sposoban da proizvede odgovor u vidu teksta koji se gotovo ne razlikuje u odnosu na prirodni ljudski jezik. Koncepti koji stoje iza generativnog unapred obučenog transformatora se usavršavaju kroz proces iz dva koraka. Prvi korak je generativna nenadzirana predobuka koristeći neobeležene podatke, dok je drugi korak diskriminativno, nadgledano fino podešavanje (engl. *fine-tuning*) zarad poboljšanja performansi. Tokom prve faze, model uči prirodno, simulirajući kako bi i osoba učila u novom okruženju, dok u drugoj fazi fino podešavanje uključuje više usmereno i struktuirano usavršavanje od strane kreatora [2].

Ključno postignuće u procesu razvoja jezičkih modela predstavlja *InstructGPT*. Ovaj radni okvir omogućava

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dušan Gajić, vanr. prof.

[1] Jezički model je komponenta za obradu prirodnog jezika i odnose se na sisteme koji su obučeni nad velikim korpusom podataka [1].

[2] OpenAI je istraživačka laboratorija usmerena u razvoj *AI* tehnologija i koja je objavila niz proizvoda za mašinsko učenje kao što su: *ChatGPT* i *DALL-E* [2].

fino podešavanje instrukcija prethodno obučenog jezičkog modela zasnovano na učenju pojačanjem iz ljudskih povratnih informacija (engl. *reinforcement learning from human feedback*, skraćeno *RLHF*). Korišćenje ljudskih povratih informacija doprinosi tome da se jezički model uspešno koristi za rešavanje širokog spektra zadataka u oblasti obrade prirodnog jezika, čineći ga veoma raznovrsnim i fleksibilnim. [3].

2.2. *Pinecone* vektorska baza podataka

Kako bi čet botovi pružili koristan odgovor na zadati korisnikov upit potrebno je razumeti kontekst samog upita, njegovu semantičku vrednost i na osnovu toga generisati najprikladniji odgovor. Kako bi željeni efekat bio ostvariv, neophodno je koristiti vektorske ugradnje (engl. *vector embeddings*), odnosno tip predstavljanja podataka koji omogućava da sami podaci u sebi sadrže semantičke vrednosti koje su kasnije krucijalne za rad *AI* modela u svrhu razumevanja konkteksta i održavanja dugoročne memorije [4].

Prilikom vektorizacije većeg broja podataka, primetno je da se u okviru jednog vektorskog prostora distanca između vektora razlikuje, odnosno da su neki bliži jedan drugom, dok su drugi udaljeniji [5]. Ovaj efekat je izuzetno bitan, jer grupisani vektori, odnosno vektori bliži jedni drugima imaju sličnije semantičko značenje u odnosu na one druge koji su dosta udaljeniji od njih. Vektori sami po sebi predstavljaju idealnu strukturu podataka za algoritme mašinskog učenja jer su procesorske jedinice i grafičke kartice optimizovane za obavljanje matematičkih operacija za njihovu obradu. Proces transformacije podataka u vektor je zahtevna operacija, jer je neophodno zadržati prvobitno značenje podataka. Da bismo očuvali značenje samih podataka neophodno je razumeti proces vektorizacije takav da odnosi između vektora imaju smisla.

Za postizanje ovako nečega koristi se model ugradnje (engl. *embedding model*) i to propuštanjem velike količine označenih podataka kroz neuronsku mrežu. Obučavanje neuronske mreže se vrši metodom nadgledanog učenja (engl. *supervised learning*), davajući mreži veliki skup trening podataka sastavljenih od parova ulaza (engl. *input*) i označenih izlaza (engl. *labeled output*). Svakom iteracijom obučavanja, neuronska mreža modifikuje aktivacije u svakom mrežnom sloju i to sve sa ciljem da se na kraju obučavanja izlazna oznaka može predvideti na osnovu datog ulaza, čak i za ulaz koji nije bio deo testnog skupa.

Model ugradnje je zapravo ovako obučena neuronska mreža ali bez poslednjeg sloja, kako bi se umesto označenog izlaza, dobila vektorska ugradnja [5].

Dobijene vektorske ugradnje je potrebno prikladno čuvati. Za to se koristi vektorska baza podataka (engl. *vector database*), u konkretnom slučaju korišćena je *Pinecone* vektorska baza podataka, radi optimizovanog skladištenja vektorskih ugradnji i kasnijim upitima nad istim. Ovakav tip baza podataka je specijalno dizajniran za rukovanje ovom vrstom podataka nudeći visoke performanse, skalabilnost i fleksibilnost pri radu sa vektorskim podacima. Obezbeđena je pretraga na semantičkom nivou, kao i dugoročno čuvanje podataka [4].

2.3. *LangChain*

LangChain sluzi kao radni okvir dizajniran za kreiranje aplikacija zasnovanih na velikim jezičkim modelima. Njegov primarni cilj je da omogući pogodan način integracije različitih izvora podataka i da olakša integraciju sa drugim aplikacijama. U procesu razvoja aplikacija kao što su čet botovi, ključno je da sistem poseduje svest o kontekstu, što znači da treba da poveže jezički model sa različitim izvorima konteksta [6]. Inkorporacija samog konteksta je od presudnog značaja za modele da shvate i efikasno integrišu informacije. Štaviše, radni okvir se oslanja i na jezički model zbog samog rasuđivanja, uzimajući u obzir kako generisati odgovor na osnovu datog konteksta i odrediti akcije koje je potrebno izvršiti.

U aplikacijama zasnovanim na jezičkom modelu, svakako najvažnija celina jeste sam model. *LangChain* pruža interfejse i integraciju za dva tipa modela: velike jezičke modele (engl. *Large Language Model*, skraćeno *LLM*) i konverzacijske modele (engl. *Chat Models*). Ova dva tipa modela se suptilno, ali opet značajno razlikuju. Veliki jezički model se u okviru LangChain-a odnosi na model generisanja teksta, prihvatajući tekst na ulazu i isto tako izbaciti generisani tekst na izlazu. *GPT-3* model razvijen od strane OpenAI-a je implementiran kao veliki jezički model. Konverzacijski model sa druge strane podržava veliki jezički model, ali je dodatno prilagođen za razgovore. Ovo podrazumeva da na samom ulazu ne primaju čist tekst, već listu poruka, posebno označenih ulogom kojoj pripadaju (obično se koriste: „*System*“, „*AI*“ i „*Human*“), dok je izlaz iz ovakvog tipa modela poruka obeležena oznakom „*AI*“. *GPT-4* model je jedna od implementacija koja koristi ovaj tip [7].

LangChain je skup gradivnih elemenata, od kojih je najvažniji svakako sam lanac (engl. *chain*). Lanac obično kombinuje veliki jezički model zajedno sa *prompt*-om, čineći zasebnu celinu, ali koja se dalje može kombinovati sa drugim gradivnim elementima i na taj način proširiti kako bi se postigao željeni niz operacija nad korisničkim upitom. Najjednostavniji tip lanca predstavlja onaj koji ima jedan ulaz i jedan izlaz. Više lanaca mogu biti pokrenuti jedan za drugim, gde bi izlaz jednog lanca predstavljao ulaz sledećeg. Više različitih lanaca mogu biti spojeni primenom *Simple Sequential Chain*-a. Postoje i drugi tipovi lanaca. Nešto složeniji u odnosu na prethodni je *Sequential Chain*, koji može da ima više ulaza, ali samo jedan izlaz. Poprilično uobičajen scenario je korišćenje više lanaca postavljenih tako da je usmeravanje zavisno od toga šta se nalazi na samom ulazu. Na primer, ako imamo nekoliko usko specijalizovanih lanaca za određen tip podataka, možemo ispred njih imati jedan *Router Chain*, koji će odlučiti u kojem pravcu će informacija dalje ići, odnosno kojem lancu će predati tok [8].

Jezički model je sam po sebi bez stanja (engl. *stateless*), što znači da nije svestan bilo kakve interakcije (razgovora) koji se vodio do sada. Svaki poziv ka jezičkom modelu je nezavisan od pređašnjih. Kako bi se ostvario efekat pamćenja, neophodno je uvrstiti komponentu memorije (koja čuva prethodnu prepisku) u jezički model prilikom narednog upita. *LangChain* nudi

niz različitih načina upravljanja memorijom, a najkorišćeniji su [9]:

a) ***ConversationBufferMemory.***
ConversationBufferMemory funkcioniše na taj način što čuva čitavu prepisku i prosleđuje je kao kontekst u model prilikom svakog narednog upita. Ovaj pristup može biti koristan u određenim scenarijima, ali se smatra kao izuzetno skup, iz razloga što kada se konverzacija oduži, velike količine informacija se prosleđuju pri svakom sledećem upitu.

b) ***ConversationBufferWindowMemory.***
ConversationBufferWindowMemory predstavlja tip memorije, ali specifičan po tome što je moguće definisati koliko maksimalno poslednjih interakcija [3] je moguće ubaciti u kontekst prilikom narednog korisničkog upita. Ovim se postiže to da je memorija limitirana, odnosno da je rast ograničen.

c) ***ConversationTokenBufferMemory.***
ConversationTokenBufferMemory je dosta sličan prethodnom tipu memorije. Kao što i sam naziv govori, razlika je u tome što se umesto broja interakcija, definiše limit na nivou broja tokena[4]. Pamćenjem N poslednjih tokena, memorija je takođe limitirana.

d) ***ConversationSummaryMemory.***
ConversationSummaryMemory predstavlja specifičan tip memorije. Za razliku od prethodnih tipova, u ovom pristupu se koristi jezički model radi sumarizacije prethodne prepiske. Tako sumarizovan kontekst se kasnije uzima u obzir pri generisanju novog odgovora.

Za razliku od lanaca, kod kojih je redosled akcija unapred određen, agenti predstavljaju fleksibilnije rešenje u pogledu da je odlučivanje koje akcije će se izvršiti i kojim redosledom prepušteno samom jezičkom modelu. Agenti se još zovu i lanci koji su odgovorni za određivanje narednih koraka u procesu izvršavanja. Pored samog jezičkog modela i korisničkog unosa, agentima je neophodna informacija o nizu alata (engl. Tools[5]) kojima raspolažu i prethodno izvršenim koracima. Kako bi agenti uspešno koristili ponuđene alate, neophodno je dati opis u kojim situacijama bi određeni alat bio od koristi, što će kasnije služiti jezičkom modelu prilikom odabira narednog koraka [11].

2.4. *Zep* dugoročna memorija

Zep funkcioniše kao platforma otvorenog koda (engl. *open source*) dizajnirana tako da pruži efikasan rad sa aplikacijama baziranih na jezičkom modelu. Ovu platformu odlikuje sposobnost asinhrone sumarizacije, ugrađivanja, kao i obogaćivanja interakcija u okviru prepiske, i to na način da da ne narušavaju korisničko iskustvo prilikom korišćenja samog čet bota. *Zep* ima svoju bazu podataka, gde se vrši čuvanje podataka, obezbeđujući skalabilnost potrebnu u bilo kom trenutku [12].

3. IMPLEMENTACIJA „AUTO CALL CENTER“ ČET BOTA

„Auto Call Center“ čet bot predstavlja sistem zadužen da korisnicima pruži pomoć prilikom kupovine novog automobila. Čet bot raspolaže sa informacijama o tome koja su vozila dostupna u salonu, imajući dodatne informacije o svakom. Pored samog informisanja o stanju ponude, čet bot je u mogućnosti da pruži bilo kakav uopšteni vid informacije vezan za automobil. Uspešno savetuje prilikom nedoumice o konkretnim markama automobila i njihovim modelima kao što su potrošnja, istorija itd. Takođe sposoban je da reši naprednije matematičke računice, koje su potrebne prilikom izračunavanja poreza, prilikom povrata nakon uspešne kupovine. I pored svega prethodno opisanog, čet bot ima mogućnost pamćenja same konverzacije sa korisnikom, što pruža još preciznije odgovore, približnije samom korisniku.

Serverska strana pisana u Python [13] programskom jeziku, dok je za klijentsku stranu korišćen Telegram [14]. U nastavku sledi opis komponenti na serverskoj strani koje se koriste u procesu rezonovanja prilikom generisanja odgovora:

Prva bitna komponenta jeste *Pinecone* vektorska baza podataka. Predstavljena u obliku alata korišćenog od strane *LangChain* agenta za dobijanje informacija o ponudi samih vozila kojima salon raspolaže. U okviru ove baze podataka, unete su vektorske ugradnje nastale na osnovu konkretnih primeraka automobila. Svaka vektorska ugradnja predstavlja vektor dimenzije 1536 i kasnije je za pretragu ovih entiteta korišćena kosinusna (engl. *cosine*) razdaljina. Agent pristupa ovom alatu isključivo u situacijama kada je potrebna informacija o raspoloživosti automobila, konkretnoj ponudi i potrebi da odgovori na pitanja vezana za cenu izlistanih modela.

Druga bitna komponenta je *Zep* dugoročna memorija. Kao i *Pinecone* vektorska baza podataka, *Zep* je definisan kao poseban alat korišćen od strane *LangChain* agenta za dobijanje informacija o detaljima iz dosadašnje prepiske. Ova baza sadrži čitavu konverzaciju između korisnika i čet bota. Za svakog korisnika se kreira jedinstvena sesija koja se kasnije koristi za grupisanje svih poruka i sumarizacija za konkretnog korisnika. Najčešća primena ovog alata jeste za pamćenje preferencija korisnika koje se odnose na željeni automobil.

LangChain agent pored prethodno opisanih alata, koristi još dva manja, i to: *MathTool* i *WikiTool*. *MathTool* se koristi za rešavanje matematičkih računica, u ovom slučaju najčešće za izračunavanje poreza u slučajevima kada je moguć povrat istog, dok sa druge strane *WikiTool* služi za informisanje o samim markama i modelima automobila.

[3] Interakcija predstavlja spoj korisničkog unosa i dobijenog odgovora iz jezičkog modela

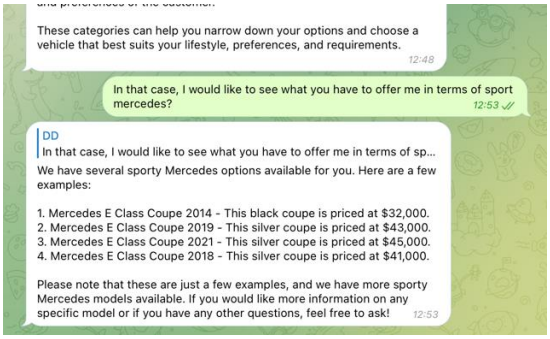
[4] Token je predstavljen kao deo reči, njegova dužina je oko 4 karaktera engleskog jezika [10].

[5] *Tool* predstavlja funkciju koja se poziva od strane *LangChain* agenata zadužena za rešavanje specifičnih operacija [11].

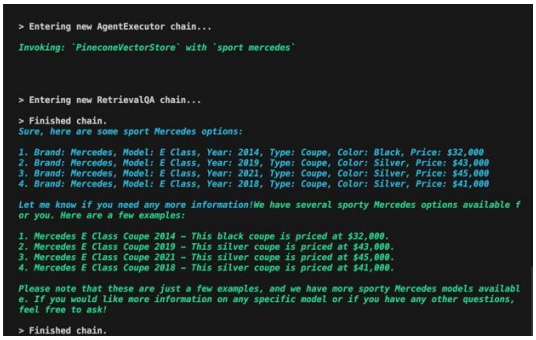
Za kraj je ostao sam *LangChain* agent, koji koristi sve prethodno navedene alate i opise za njihovo korišćenje, kao i jezički model kako bi doneo odluku o tome koji od ponuđenih izvora informisanja je najprikladniji za generisanje odgovora na osnovu korisničkog unosa.

4. REZULTATI RADA

„Auto Call Center“ čet bot predstavlja sistem zadužen da korisnicima pruži pomoć prilikom kupovine novog automobila. Zbog prirode problema koji rešava, demonstracija njegovog rada će biti propraćena kroz primer konverzacije između čet bota i potencijalnog kupca. Upravo na Slici 1 je prikazan takav scenario, gde korisnik iznosi informaciju da je zainteresovan za sportska vozila brenda Mercedes i od čet bota traži listu vozila koja ispunjavaju dati kriterijum. Čet bot pruža odgovor u vidu liste, čime ispunjava korisnikov zahtev, a sam odgovor je zasnovan na korišćenju vektorske baze podataka, jer se radi o konkretnim vozilima iz ponude salona, što je upravo i definisano u samom alatu agenta. Rezonovanje od strane agenta je prikazano na Slici 2.



Slika 1. *Prikaz konverzacije*



Slika 2. *Prikaz rezonovanja agenta*

5. ZAKLJUČAK

U ovom radu dat je opis čet bot sistema koji simulira ljudsku interakciju putem tekstualnog kanala. Glavna odlika prethodno opisanog čet bota jeste sposobnost dugoročnog pamćenja i razumevanja konteksta čime su poruke koje generiše još približnije samom korisniku, odnosno postignuta je personalizovana komunikacija u realnom vremenu. Detaljno su objašnjeni osnovni tehnološki koncepti poput: vektorske ugradnje i vektorskih baza podataka, *Zep* dugoročne memorije, *LangChain* radnog okvira i svih gradivnih komponenti koje obuhvata, kao i samog *GPT* modela. U nastavku je objašnjen način na koji je implementiran čet bot, kao i iz kojih komponenti se sastoji, na koji način su te komponente povezane i na koji deo sistema se oslanja u zavisnosti od date situacije. Takođe pružen je uvid u konkretan primer upotrebe čet bota, ispraćen kratkom konverzacijom i putem rezonovanja koji prođe od prihvatanja poruke, do generisanja smislenog odgovora koji će ispuniti potrebe korisnika.

6. LITERATURA

[1] Brown, Hannah, Katherine Lee, Fatemehsadat Mireshghallah, Reza Shokri, and Florian Tramèr. "What does it mean for a language model to preserve privacy?." In Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency, pp. 2280-229.

[2] Lund, Brady D., and Ting Wang. "Chatting about ChatGPT: how may AI and GPT impact academia and libraries?." Library Hi Tech News 40, no. 3 (2023): 26-29.

[3] Liu, Yiheng, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He et al. "Summary of chatgpt/gpt-4 research and perspective towards the future of large language models." arXiv preprint arXiv:2304.01852 (2023).

[4] https://www.pinecone.io/learn/vector-database (pristupljeno u septembru 2023.)

[5] https://www.pinecone.io/learn/vector-embeddings-for-developers (pristupljeno u septembru 2023.)

[6] https://python.langchain.com/docs/get_started/introduction (pristupljeno u septembru 2023.)

[7] https://python.langchain.com/docs/modules/model_io/models (pristupljeno u septembru 2023.)

[8] Topsakal, Oguzhan, and Tahir Cetin Akinci. "Creating Large Language Model Applications Utilizing LangChain: A Primer on Developing LLM Apps Fast." In International Conference on Applied Engineering and Natural Sciences, vol. 1, no. 1, pp. 1050-1056. 2023.

[9] https://learn.deeplearning.ai/langchain/lesson/3/memory (pristupljeno u septembru 2023.)

[10] https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them (pristupljeno u septembru 2023.)

[11] https://python.langchain.com/docs/modules/agents (pristupljeno u septembru 2023.)

[12] https://github.com/getzep/zep (pristupljeno u septembru 2023.)

[13] https://www.python.org (pristupljeno u septembru 2023.)

[14] https://telegram.org (pristupljeno u septembru 2023.)

Kratka biografija:

Dejan Dokić rođen je u Novom Sadu 1997. god. Diplomski rad na Fakultetu tehničkih nauka u Novom Sadu, smer softversko inženjerstvo i informacione tehnologije odbranio je 2020.god.

kontakt: dejan.dokic.ns@gmail.com