Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 4.9
DOI: https://doi.org/10.24867/30BE29Jevtic

GLASOVNO I TEKSTUALNO UPRAVLJANJE INFORMACIONIM SISTEMIMA ZASNOVANO NA KORIŠĆENJU VELIKIH JEZIČKIH MODELA VEŠTAČKE INTELIGENCIJE

USING AI LARGE LANGUAGE MODELS FOR VOICE AND TEXT BASED MANAGEMENT OF INFORMATION SYSTEMS

Jovana Jevtić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – SOFTVERSKO INŽENJERSTVO I INFORMACIONE TEHNOLOGIJE

Kratak sadržaj - *U radu su predstavljeni interfejsi prirodnog jezika (eng. Natural Language Interfaces) kao i prednosti koje se dobijaju njihovim korišćenjem. Opisana je obrada prirodnog jezika i veliki jezički modeli koji se mogu iskoristiti za realizovanje informacionog sistema sa glasovnim i tekstualnim upravljanjem. Takođe, prikazana je softverska arhitektura jednog takvog sistema i data studija aplikacije za upravljanje restoranom kao primer.*

Ključne reči: *interfejsi prirodnog jezika, obrada prirodnog jezika, veliki jezički modeli*

Abstract – *The paper presents natural language interfaces and the advantages gained from their use. It describes natural language processing and large language models that can be utilized for implementing systems with voice and text control. Additionally, it shows the software architecture of such a system and provides a case study of a restaurant management application as an example*

Keywords: *Natural Language Interfaces, Natural Language Processing, Large Language Models*

1. UVOD

Informacioni sistemi predstavljaju skup softvera, hardvera, podataka, ljudi i procedura koji su objedinjeni tako da generišu informacije koje omogućavaju aktivnosti neke organizacije. Pod tim imenom počeli su da se razvijaju sredinom 20. veka i od tada je postojalo više različitih načina za njihovu upotrebu. Prva era je era komandne linije kada su se računari koristili tako što su korisnici unosili tekstualne komande. Nakon toga, da bi se postigla lakoća korišćenja počinju da se razvijaju grafički korisnički interfejsi. Nakon komandne linije prešlo se na korišćenje desktop aplikacija koje su obično bile specifične za operativne sisteme. Sa razvojem interneta, postaju popularne veb aplikacije za čiju upotrebu su bili neophodni veb pretraživači. Kako su veb aplikacije na početku bile spore, pojavljuju su Client Side Rendering i Single Page Applications, kao njihova optimizacija. Pored veb aplikacija danas su u velikoj upotrebi i mobilne aplikacije. Ipak, jedna stvar se može izdvojiti kao zajednička za prethodno opisane vrste aplikacija, a to je da korisnik interaguje sa njima preko dobro struktuiranih formi koje mora da popuni na specifičan način da bi dobio željeni odgovor.

Obrada prirodnog jezika [1] je grana veštačke inteligencije kojoj je cilj da omogući da računari mogu da razumeju, obrade i generišu prirodni jezik, dok veliki jezički modeli [2] predstavljaju njene napredne modele koji su trenirani nad velikim količinama podataka. Razvoj ove grane veštačke inteligencije i njenih modela prethodnih godina, daju nam mogućnost da razvijamo sistem sa glasovnim i tekstualnim upravljanjem. Korisnik bi prirodnim jezikom mogao da iskaže šta želi da dobije od sistema. Takav način upotrebe bi doneo veću efikasnost, brže upravljanje i smanjeno vreme obuke. Upravo to daje nam motivaciju za razvijanje informacionog sistema sa glasovnim i tekstualnim upravljanjem koji je opisan u ovom radu.

2. STANJE U OBLASTI

Interfejsi prirodnog jezika [3] su tehnologije koje omogućavaju korisnicima da komuniciraju sa računarom koristeći prirodni jezik, kako u pisanoj, tako i u govornoj formi. Njihov cilj je da pojednostave komunikaciju između ljudi i računara, čineći je priodnom i intuitivnom. Razvoj ovakvih interfejsa uveliko zavisi od obrade prirodnog jezika čiji će osnovni koncepti i relevantni modeli biti opisani u nastavku poglavlja.

Obrada prirodnog jezika (eng. Natural Language Processing – NLP) je grana veštačke inteligencije koja se odnosi na računarsku obradu jezika ljudi. Tu se ubrajaju algoritmi kojima je ulaz tekst ili govor nastao od strane čoveka, ali i oni kojima je cilj da proizvedu takav izlaz. NLP ima mnoštvo zadataka koje rešava, a neki od njih su: prepoznavanje govora, pretvaranje teksta u govor, prevođenje, izdvajanje informacija, pretraga informacija, ogovaranje na pitanja, tekstualna klasifikacija. Iako NLP uspešno rešava pomenute zadatke, postoji mnoštvo izazova koji su morali da se prevaziđu, ali i onih koji se i dalje prevazilaze. Neki od uobičajenih izazova su kontekstualne reči i fraze, sinonimi, homofoni, homonimi, različite gramatičke strukture, sarkazam i ironija, ali i sve ostale dvosmislenosti, nejasnoće i ne tako dobro definisani delovi ljudskog jezika.

Na početku, prirodni jezici obrađivani su pomoću sistema koji su zasnovani na semantičkoj analizi i rasčlanjivanju. Oni su doveli do određenog uspeha, ali su bili ograničeni velikom složenošću jezičkih pojava. Zbog toga su počeli

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bila prof. dr Gordana Milosavljević.

da se koriste modeli koji se zasnivaju na mašinskom učenju kao što su konvolucione i rekurentne neuronske mreže, da bi se i oni prevazišli velikim jezičkim modelima zasnovanim na arhitekturi transformatora. Njihova arhitektura biće opisana u sledećem potpoglavlju.

2.1. Transformator modeli

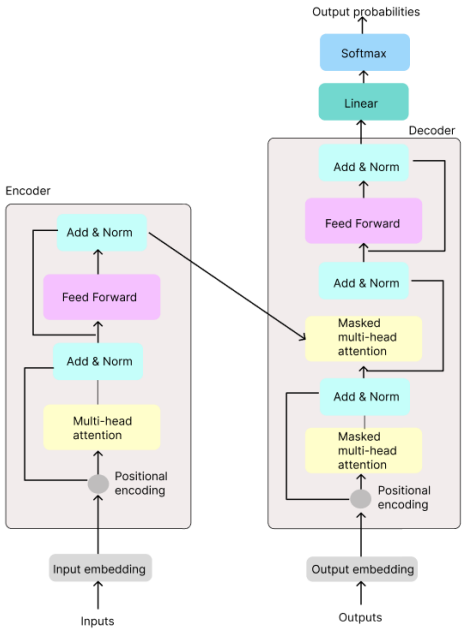
Transformator modeli [4] razlikuju se od svojih prethodnika po tome što uspešno rešavaju problem mogućnosti obraćanja pažnje na određene reči bez obzira koliko su one udaljene jedna od druge i po tome što imaju bolje performanse. Ovi modeli imaju sposobnost da prevaziđu oba koristeći prednosti mehanizma pažnje. Glavna karakteristika arhitekture ovih modela je da sadrže koder i dekoder. Pojednostavljeno, moglo bi se reći da koder uzima ulaz i daje matričnu reprezentaciju tog ulaza, dok dekoder preuzima tu matričnu reprezentaciju i iterativno generiše izlaz. Koder i dekoder su zapravo stek sa više slojeva, ali tako da oba imaju isti broj slojeva. Svi koderi imaju istu strukturu, a ulaz ulazi u svaki od njih i prosleđuje se sledećem koderu u nizu. Takođe, svi dekoderi imaju istu strukturu i dobijaju ulaz od poslednjeg kodera i prethodnog dekodera. Radni tok kodera može se podeliti na sledeće korake:

- ugrađivanje ulaza - dešava se samo u prvom koderu i predstavlja pretvaranje ulaznih tokena u vektore. Ova ugrađivanja obuhvataju semantička značenja leksema i pretvaraju ih u numeričke vrednosti. Svi koderi dobijaju listu vektora, ali u svakom koderu sem u prvom to bi bio izlaz kodera direktno ispod njega.
- poziciona kodiranje – dodaju se ulaznim ugrađenim elementima da bi pružili informacije o poziciji svakog tokena u nizu. Ovo transformer modelima omogućava da razumeju poziciju svake reči u rečenici. To se postiže upotrebom različitih sinusnih i kosinusnih funkcija.
- mehanizam pažnje - ovaj mehanizam može da se nazove i samopažnja i on omogućava da se svaka reč iz ulaza poveže sa drugim rečima. Rezultat pažnje se izračunava na osnovu upita, ključa i vrednosi. Upit je vektor koji predstavlja određenu reč ili token iz ulazne sekvence. Ključ je, takođe, vektor koji odgovara svakoj reči ili tokenu iz ulazne sekvence. Vrednost je povezana sa ključem i koristi se za konstruisanje izlaza pažnje. Kada se ključ i upit dobro podudaraju, što znači da imaju visoku ocenu pažnje, odgovarajuća vrednost se naglašava na izlazu. Vrednosti dobijene izračunavanje pažnje se nakon toga normalizuju kako bi se dobije težine pažnje, koje se koriste za generisanje izlaznog vektora.
- potpuno povezana mreža (normalizacija) - svaki izlaz podsloja se dodaje svoj ulazu (preostala veza) kako bi se ublažio problem nestajanja gradijenta. Ovaj proces će se ponoviti i nakon sledećeg koraka.
- neuronska mreža sa propagacijom unapred – kombinacija linearnih slojeva sa ReLu aktivacionom funckojim smeštenom između njih.
- izlaz kodera – skup vektora od kojih svaki predstavlja ulaznu sekvencu sa bogatim kontekstualnim razumevanjem.

Što se tiče dekodera, njegov radni tok može se podeliti na sledeće korake:

- ugrađivanje izlaza - deo isti kao kod kodera, dakle ulaz prvo prolazi kroz ugrađivanje.
- poziciono kodiranje - baš kao i kod kodera, ulaz prolazi kroz sloj pozicionog kodiranja.
- maskirani mehanizam samopažnje - ima jednu ključnu razliku u odnosu na isti sloj u koderu, a to je da sprečava pozicije da prisustvuju narednim pozicijama, što znači da svaka reč u nizu nije pod uticajem budućih tokena. Ovo maskiranje osigurava da predviđanja za određenu poziciju mogu zavisiti sam od poznatih izlaza na pozicijama pre nje.
- koder-dekoder pažnja više glava – ovaj sloj omogućava dekoderu da pronađe i nagalsi najrelevantnije delove ulaze kodera.
- neuronska mreža sa propagacijom unapred – svaki sloj dekodera uključuje potpuno povezanu mrežu za prosleđivanje, primenjenu na svaku poziciju posebno.
- linearni klasifikator i softmax za generisanje izlaznih verovatnoća - ovo je konačni linearni sloj koji funkcioniše kao klasifikator.

Opisana arhitektura kodera i dekodera može se videti na slici Slika 2.1.

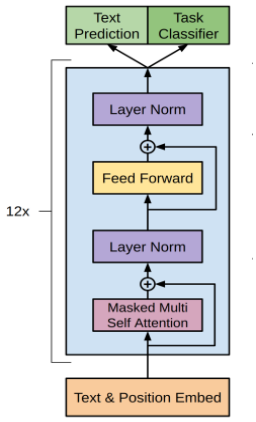


Slika 2.1 Arhitektura kodera i dekodera transformator modela

Modeli zasnovani na arhitekturi transformatora su Generative Pre-trained Transforme (GPT) modeli razvijeni od strane OpenAI kompanije. Tačnije, GPT koristi neizmenjeni dekoder deo transformatora, s tim što mu nedostaje deo vezan za pažnje dobijene koderom. Arhitektura ove porodice modela može se videti na slici Slika 2.2.

Porodicu modela čine mnogobrojne verzije GPT, GPT-2, GPT-3 i GPT-4 modela. U ovom radu, baziraćemo se na GPT-3 modele. Model ima 175 milijardi parametara koji predstavljaju težine na osnovu kojih model donosi odluke prilikom obrade i generisanja teksta. Treniran je nad više od 570 GB podataka koji uključuju vesti, knjige, vikipediju i mnoge druge izvore teksta. Može da se koristi za generisanje teksta koji je relevantan za zadatu temu, da razume zadati kontekst i koristi ga za generisanje odgovora, može prevodi tekstove sa jednog jezika na drugi, da vrši sumarizaciju teksta, generište tekstove i slično. Konkretan model koji se koristio za potrebe demonstracije sistema sa glasovnim i tekstualnim

upravljanjem je GPT-3.5-turbo [6], unapređena verzija GPT-3 modela, dizajnirana tako da bude jeftinija i brža od prethodne verzije.



Slika 2.2 Arhitektura GPT modela[5]

Još jedan model koji ima arhitekturu transformatora koji je, takođe, iskorišćen za potrebe demonstracije sistema sa glasovnim i tekstualnim upravljanjem je Whisper [7]. To je sistem za automatsko prepoznavanje govora obučen nad 680 hiljada sati višejezičkog materijala. Što se tiče arhitekture, implementiran je kao potupni koder-dekoder transformator.

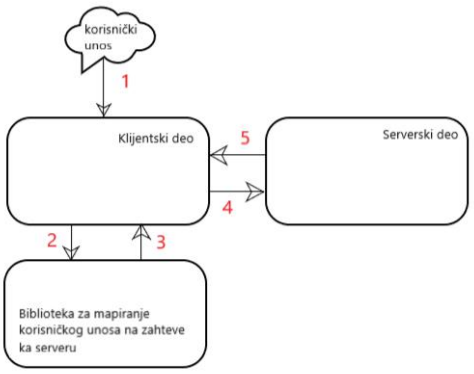
3. SOFTVERSKA ARHITEKTURA SISTEMA SA GLASOVNIM I TEKSTUALNIM UPRAVLJANJEM

Informacioni sistem sa glasovnim i tekstualnim upravljanjem, pored toga što bi se koristio na drugačiji način, drugačije bi se i razvijao. U nastavku ovog poglavlja biće prikazana moguća arhitektura jednog takvog veb sistema.

Sistem bi se sastojao od serverskog dela, koji opciono može da bude skup više manjih servisa i baza podataka i od klijentskog sloja. Na klijentskom sloju se uvodi razlika u odnosu na klasične veb sisteme. Pošto će ulaz u sistem biti prirodni jezik, klijentski sloj će morati da ga obradi pre nego što ga prosledi serverskom sloju. U tu svrhu, nastala je biblioteka kojoj se prosleđuje korisnički unos na osnovu kog ona vraća informacije o tome kakav je zahtev potrebno proslediti na serverski deo. Sam sistem i tok informacija kroz njega prikazani su na slici Slika 3.1. Iako je biblioteka za obradu korisničkog unosa, radi lakšeg prikaza, predstavljena kao posebna komponenta, ipak je potrebno naglasiti da je ona samo jedna zavisnost klijentskog sloja. Dakle, klijentski sloj će korisnički unos proslediti biblioteci koja će obradom da izdvoji putanju ka krajnjoj tački servera, HTTP metodu i telo zahteva. Nakon toga klijentski sloj će proslediti zahtev ka serveru, a kada primi odgovor, iskoristiće neku od svojih generičkih komponenti da taj odgovor prikaže.

Pomenuta biblioteka za mapiranje korisničkog zahteva na zahtev ka serveru, realizovana je za potrebe ovog rada, kao Node Package Manager (npm) paket [8]. Deo paketa koji se tiče komunikacije sa velikim jezičkim modelom implementiran je pomoću OpenaAI biblioteke [9]. Konkretno korišćene su dve njene funkcije:

- *openai.chat.completions.create* - funkcija koja vraća odgovor jezičkog modela na prosleđena pitanja. Bitno je napomenuti da postoji ograničenje u broju tokena koje jezički model može da obradi.



Slika 3.1 Arhitektura sistema sa glasovnim i tekstualnim upravljanjem

- *openai.audio.transcriptions.create* - funkcija koja obrađuje prosleđeni audio fajl tako da generiše tekst na osnovu njega.

Ono što je neophodno proslediti pomenutoj biblioteci pre same obrade zahteva, je Swagger [10] specifikaciju serverskog dela kao i podatke o velikom jezičkom modelu koji će se koristiti za obradu tekstualnih podataka. Pomenuta biblioteka, nudi dve funkcije koje se mogu pozvati:

- *processRequest* – prima tekstualni zahtev korisnika, a vraća putanju do krajnje tačke, HTTP merodu i opciono telo zahteva. U pozadini, biće poslato nekoliko zahteva jezičkom modelu. Prvo će se zahtevati krajnja tačka i HTTP metoda, a nakon toga u zavisnosti od tipa HTTP metode i telo zahteva. Ukoliko postoji telo zahteva, zahtevaće se i njegova validacija od strane jezičkog modela.
- *processRequestFromAudio* – prima audio zahtev korisnika, dok je povratna vrednost ista kao za prethodno opisanu funkciju. Ova funkcija ima jedan dodatni korak pre izvršavanja svih koraka prethodne funckije, a to je generisanje teksta iz prosleđenog audio fajla.

4. STUDIJA SLUČAJA APLIKACIJE ZA UPRAVLJANJE RESTORANOM

Za demonstraciju sistema sa glasovnim i tekstualnim upravljanjem izabran je informacioni sistem restorana koji je dovoljno raznovrstan da pokrije tipične scenarije ali sužen tako da se ne bavimo detaljima koji nisu relevantni za temu.

Aplikacija je namenjena za korišćenje, isključivo, od strane zaposlenih restorana. Prema tome postoji nekoliko tipova korisnika koji mogu da je koriste: sistem administratori, menadžeri, kuvari, konobari i šankeri. Što se tiče funkcionalnosti sistema, moguće je upravljati jelovnicima, stavkama jelovnika, podacima o zaposlenima i njihovim platama, porudžbinama i stavkama porudžbina, i rezervacijama stolova. Takođe, moguće je generisati izveštaje o poslovanju u određenom vremenu.

U nastavku potpoglavlja biće prikazani korisnički zahtevi izraženi prirodnim jezikom i rezultati obrade biblioteke za mapiranje korisničkog zahteva na zahtev ka serveru.

- Ukoliko korisnik želi da dobavi informacije o svim konobarima, dovoljno je da uputi zahtev “list all waiters”. Biblioteka za mapiranje zahteva će vratiti informacije u sledećoj formi:

{ “httpMethod”: “GET”, “path”: “/api/v1/waiter”}.

- Ukoliko korisnik želi da vidi sve porudžbine određenog konobara dovoljno je da uputi zahtev sličan “find all orders that have waiter with id 7”. Biblioteka za mapiranje zahteva će vratiti informacije u sledećoj formi:
{ “httpMethod”: “GET”, “path”: “/api/v1/orders/byWaiter/7” }.

- Ukoliko korisnik želi da vidi sve porudžbine određenog konobara koje imaju status “Poručeno”, dovoljno je da uputi zahtev “filter orders with status Poručeno and waiter id 7”. Biblioteka za mapiranje zahteva će vratiti informacije u sledećoj formi:
{ “httpMethod”: “GET”, “path”: “/api/v1/orders/filter/7/Poručeno” }

- Ukoliko korisnik želi da doda novog konobara u sistem potrebno je da uputi zahtev u kom se nalaze svi potrebni podaci za dodavanje novog konobara. Zahtev bi trebao da bude sličan “Create a new waiter. His first name is Marko, his last name is Markovic, his email is mare@example.rs, his phone number is 1234, his account number is 123”. Biblioteka za mapiranje zahteva će vratiti informacije u sledećoj formi:

```
{ “httpMethod”: “POST”,
“path”: “/api/v1/waiters”,
“payload”: {
   "name": "Marko",
   "lastName": "Markovic",
    "phoneNumber": "1234",
     "email": "mare@example.rs",
      "accountNumber": "123"
} }.
```

Ukoliko korisnik prilikom slanja zahteva izostavi neki od obaveznih podataka, biblioteka za mapiranje će to prepoznati i korisniku će biti prikazana poruka o tome. Primer poruke za prethodni zahtev je:
“In order for the request body to be valid for creating a waiter, we need to include the following information:
- First name of the waiter (name)
- Last name of the waiter (lastName)
- Phone number of the waiter (phoneNumber)
- Account number of the waiter (accountNumber)
Without this information, the request body would be considered invalid.“

5. ZAKLJUČAK

U današnjem svetu, računari imaju sveprisutnu i ključnu ulogu u gotovo svim aspektima života. Upravo zbog toga, postaje sve bitniji način na koji se odvija interakcija između korisnika i računarskih sistema. Interfejsi prirodnog jezika omogućavaju korisnicima da komuniciraju sa računarima koristeći isključivo prirodni jezik i time pružaju intuitivnije i pristupačnije korisničko iskustvo. To je dalo motivaciju za razvijanje sistema sa glasovnim i tekstualnim upravljanjem koji je opisan u ovom radu. Mogućnost implementacije ovakvih sistema omogućio je napredak obrade prirodnog jezika, a najviše veliki jezički modeli zasnovani na arhitekturi transformatora.
U radu je opisana softverska arhitektura sistema sa glasovnim i tekstualnim upravljanjem. Posebna pažnja posvećena je biblioteci koja mapira korisnički zahtev na zahtev ka serveru i koja je implementirana u svrhe ovog rada. Takođe, prikazano je nekoliko primera upotrebe na slučaju aplikacije za upravljanje restoranom.
Prikazani primeri upotrebe dovode nas do zaključka da implementirani sistem uspešno obrađuje tipične zahteve korisnika restoranske aplikacije i time mu olakšava upotrebu. Ipak, bitno je pomenuti i ograničenja koja se ogledaju u nepodržavanju obrade kompleksnijih upita, brzini odgovora, ali i broju tokena koje veliki jezički model može da obradi u određenom vremenskom periodu. Ono što bi doprinelo još većoj uspešnosti ovih sistema je svakako rad na razvijanju velikih jezičkih modela kako bi se uklonila pomenuta ograničenja. Takođe, ovakva vrsta upravljanja mogla bi da se oproba i u drugim tipovima aplikacija, kao što su one za upravljanje pametnim kućnim uređajima.

LITERATURA

[1] Nadkarni, Prakash M., Lucila Ohno-Machado, and Wendy W. Chapman. "Natural language processing: an introduction." *Journal of the American Medical Informatics Association* 18.5, 2011

[2] Chang, Yupeng, et al. "A survey on evaluation of large language models." *ACM Transactions on Intelligent Systems and Technology* 15.3, 2024

[3] Ogden, William C., and P. Bernick. "Using natural language interfaces." *Handbook of human-computer interaction*. North-Holland, 1997.

[4] Radford, Alec. "Improving language understanding by generative pre-training." (2018).

[5] Brown, Tom B. "Language models are few-shot learners." *arXiv preprint arXiv:2005.14165* (2020).

[6] GPT-3.5 Turbo https://platform.openai.com/docs/models/gpt-3-5-turbo [datum pristupa 25.08.2024.]

[7] Introducing Whisper https://openai.com/index/whisper [datum pristupa 25.08.2024.]

[8] npm https://www.npmjs.com/ [datum pristupa 28.08.2024.]

[9] OpenAI https://platform.openai.com/docs/api-reference/introduction [datum pristupa 28.08.2024.]

[10] Swagger https://swagger.io/ [datum pristupa 29.08.2024.]

Kratka biografija:

Jovana Jevtić rođena je 22.06.1999. godine u Zvorniku. Završila je osnovnu školu „Branko Radičević“ u Malom Zvorniku, a nakon toga gimnaziju u srednjoj školi Mali Zvornik. Godine 2018. upisala je Fakultet tehničkih nauka u Novom Sadu, smer Softversko inženjerstvo i informacione tehnologije. Sve ispite polaže i studije završava u roku 2022. godine. Iste godine upisuje master akademske studije na istom smeru koje završava 2024. godine.