Izrada programske podrške za interakciju s jezičnim modelima umjetne inteligencije

Dvojković, Sara

VELEUČILIŠTE U BJELOVARU
STRUČNI PRIJEDIPLOMSKI STUDIJ RAČUNARSTVO

IZRADA PROGRAMSKE PODRŠKE ZA INTERAKCIJU S JEZIČNIM MODELIMA UMJETNE INTELIGENCIJE

Završni rad br. 03/RAČ/2023

Sara Dvojković

Bjelovar, listopad 2023.

obrazac ZR - 001

Veleučilište u Bjelovaru

Trg E. Kvaternika 4, Bjelovar

1. DEFINIRANJE TEME ZAVRŠNOG RADA I POVJERENSTVA

Student: **Sara Dvojković** JMBAG: **0314022940**

Naslov rada (tema): **Izrada programske podrške za interakciju s jezičnim modelima umjetne inteligencije**

Područje: **Tehničke znanosti** Polje: **Računarstvo**

Grana: **Programsko inženjerstvo**

Mentor: **Krunoslav Husak, dipl. ing. rač.** zvanje: **predavač**

Članovi Povjerenstva za ocjenjivanje i obranu završnog rada:

1. **dr. sc. Zoran Vrhovski, predsjednik**
2. **Krunoslav Husak, dipl. ing. rač., mentor**
3. **Krešimir Markota, mag. ing. comp., član**

2. ZADATAK ZAVRŠNOG RADA BROJ: 03/RAČ/2023

U sklopu završnog rada potrebno je:

1. Analizirati i opisati dostupne jezične modele umjetne inteligencije.
2. Analizirati i opisati programska sučelja različitih modela umjetne inteligencije.
3. Predložiti i opisati programsku podršku koja omogućava interakciju s različitim jezičnim modelima umjetne inteligencije.
4. Izraditi i opisati .NET biblioteku koja omogućava interakciju s različitim jezičnim modelima umjetne inteligencije.
5. Izraditi podršku za glasovnu interakciju s različitim jezičnim modelima umjetne inteligencije.

Datum: 09.06.2023. godine

Mentor: **Krunoslav Husak, dipl. ing. rač.**

Sadržaj

1. UVOD

Razvoj umjetne inteligencije otvorio je vrata novim mogućnostima u različitim industrijama i područjima. Modeli umjetne inteligencije postali su ključni alati za rješavanje raznih problema, od analize podataka do obrade prirodnog jezika. Jedan od najznačajnijih aspekata korištenja umjetne inteligencije je interakcija s njima putem API-ja (engl. *Application Programming Interface*). Ovaj rad usredotočuje se na ovu važnu temu, izrađujući .NET biblioteku koja omogućuje interakciju s različitim modelima umjetne inteligencije.

Svrha ovog završnog rada je izrada programske podrške za interakciju s modelima umjetne inteligencije. Rad će pružiti dublje razumijevanje o tome kako koristiti modele umjetne inteligencija kao usluge, omogućujući raznim aplikacijama i sustavima da koriste njihove sposobnosti. Cilj je omogućiti integraciju modela u vlastite projekte te pružiti uvid u značaj i potencijal API-ja za interakciju s modelima umjetne inteligencije.

Dosadašnje spoznaje o interakciji s modelima umjetne inteligencije putem API-ja pokazuju da je ova tehnologija postala ključna za različita područja. Iako su postignuti znatni napreci, postoje izazovi u vezi sa sigurnošću podataka, performansama modela i etičkim pitanjima u vezi korištenja umjetne inteligencije.

Također, završni rad sastoji se od tri glavna poglavlja. U poglavlju 2, detaljnije će se istražiti što su to modeli umjetne inteligencije te će se istražiti različite vrste umjetne inteligencije, uključujući generativne modele umjetne inteligencije. Također, istražit će se kako se ti modeli treniraju i optimiziraju za različite zadatke. Poglavlje 3 fokusira se na koncepte interakcije s jezičnim modelima putem API-ja, uključujući razumijevanje samog pojma API-ja te različite primjene ove tehnologije. Uz to, detaljnije će se analizirati API koji nudi tvrtka OpenAI, kao i Hugging Face API koji omogućuje interakciju s velikim brojem različitih modela. U posljednjem poglavlju, poglavlju 4, opisat će se izrada .NET biblioteke koja olakšava integraciju različitih jezičnih modela u vlastite projekte te omogućuje glasovnu interakciju s jezičnim modelima.

Kroz ova tri poglavlja, ovaj rad pružit će sveobuhvatno razumijevanje interakcije s jezičnim modelima umjetne inteligencije, istražujući njihovu svrhu, način rada i utjecaj na budućnost AI tehnologija.

2. MODELI UMJETNE INTELIGENCIJE

Umjetna inteligencija aktualna je tema te se novi modeli umjetne inteligencije kontinuirano razvijaju i pojavljuju, dok se već postojeći unaprjeđuju.

Umjetna inteligencija uključuje teoriju i razvoj računalnih sustava koji su sposobni obavljati zadatke koji obično zahtijevaju ljudsku inteligenciju. Može se reći da je umjetna inteligencija sposobnost računala da oponaša ljudsku inteligenciju, što obuhvaća razumijevanje i generiranje prirodnog jezika i tako omogućuje komunikaciju s čovjekom. Također, samostalno donošenje odluka i zaključaka, učenje iz iskustva i rješavanje složenih problema još su neke od sposobnosti tih računalnih sustava.

Umjetna inteligencija se temelji na strojnom učenju koje omogućuje računalima samostalno učenje iz podataka i savladavanje velike količine znanja te poboljšanje svog rada bez ljudskog djelovanja. Strojno učenje se koristi za razvoj različitih vrsta algoritama i modela, uključujući neuronske mreže. Inspirirane strukturom ljudskog mozga, neuronske mreže sastoje se od međusobno povezanih elemenata (umjetnih neurona) koji služe za paralelnu obradu i analizu podataka. Zahvaljujući ovoj sposobnosti, neuronske mreže jedan su od ključnih alata u području umjetne inteligencije.

Umjetna inteligencija se primjenjuje u širokom rasponu područja. Primjenjuje se za potrebe generiranja teksta, slika i drugih multimedijskih sadržaja. Također, jedna od ključnih primjena umjetne inteligencije je i Google pretraživač. Pokrenuta umjetnom inteligencijom, omogućuje milijunima korisnika da svakodnevno brže i preciznije pretražuju veliku količinu podataka na internetu. U području računalnog vida, umjetna inteligencija se koristi za prepoznavanje objekata i lica na slikama i videozapisima, što je korisno za napredne sustave za potporu vozačima automobila (ADAS) koji koriste računalni vid kako bi unaprijedili sigurnost u vožnji.

Umjetna inteligencija pruža brojne prednosti i mogućnosti te potiče inovacije u različitim područjima.

2.1 Generativni modeli umjetne inteligencije

Generativna umjetna inteligencija spada u kategoriju strojnog učenja. To je grana umjetne inteligencije u kojoj računala, uz tumačenje i predviđanje, samostalno generiraju novi i jedinstveni sadržaj kao što su tekst, slike, zvuk i videozapisi.

Temelj generativnih modela su duboke neuronske mreže. To su specifične vrste neuronskih mreža koje imaju mnogo slojeva međusobno povezanih neurona. Dubina se

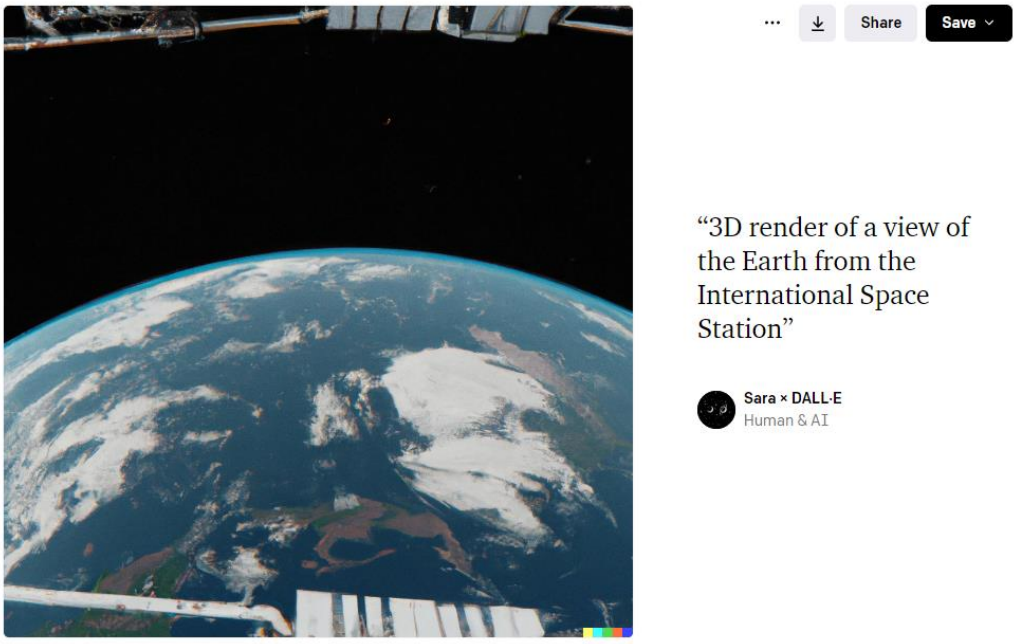
odnosi na broj slojeva u mreži. Duboke neuronske mreže su obično dizajnirane za obradu velikih i kompleksnih skupova podataka te su sposobne naučiti složene obrasce i značajke u skupu podataka. Analizom i obradom različitih varijacija podataka, ove mreže služe kao osnova za mnoge generativne modele.

Generativni modeli umjetne inteligencije treniraju se na velikoj količini podataka kako bi usvojili uzorke i karakteristike tih podataka, što im omogućuje stvaranje novog sadržaja koji je sličan onome na kojem su trenirani.

Midjourney i DALL-E 2 su primjeri generativne umjetne inteligencije. Midjourney koristi obradu prirodnog jezika i računalni vid kako bi generirao slike na temelju tekstualnog opisa. Midjourney je treniran na više od 100 milijuna slika i na velikoj količini teksta što mu omogućuje generiranje realističnih i uvjerljivih slika.

DALL-E 2 je nadogradnja originalnog DALL-E modela. Razvio ga je OpenAI te kao i Midjourney, spoj je tekstualnih i vizualnih mogućnosti umjetne inteligencije.

Midjourney i DALL-E 2, iako još uvijek u razvoju, već su sposobni generirati impresivne slike, uključujući izmišljene scene, stvarne prizore i pejzaže, i likove iz popularne kulture, poput likova iz filmova ili stripova. Vrste slika koje mogu generirati uglavnom ovise o unesenom tekstualnom upitu. Dobro osmišljen upit može pomoći u stvaranju odgovarajućih jedinstvenih slika.



Slika 2.1: Primjer slike generirane pomoću DALL-E modela

Slika 2.1 prikazuje primjer upita i generirane slike pomoću DALL-E modela. Oba modela imaju kao glavnu svrhu omogućiti korisnicima generiranje slika bez potrebe za naprednim umjetničkim vještinama ili tehničkim stručnim znanjem što može biti posebno korisno za industrije poput marketinga, grafičkog dizajna i razvoja video igara. Midjourney i DALL-E 2 moćni su alati s velikim potencijalom za kreativnost i inovacije.

Od stvaranja fotorealističnih slika do skladanja glazbe, generativni modeli ostvaruju napredak i razvijaju se na načine koji su se nekada činili nemogućima. Cilj generativne umjetne inteligencije je istražiti granice mogućeg s umjetnom inteligencijom te pokazati da umjetna inteligencija može biti primijenjena u kreativnim područjima izvan svoje tradicionalne upotrebe za analizu podataka i automatizaciju.

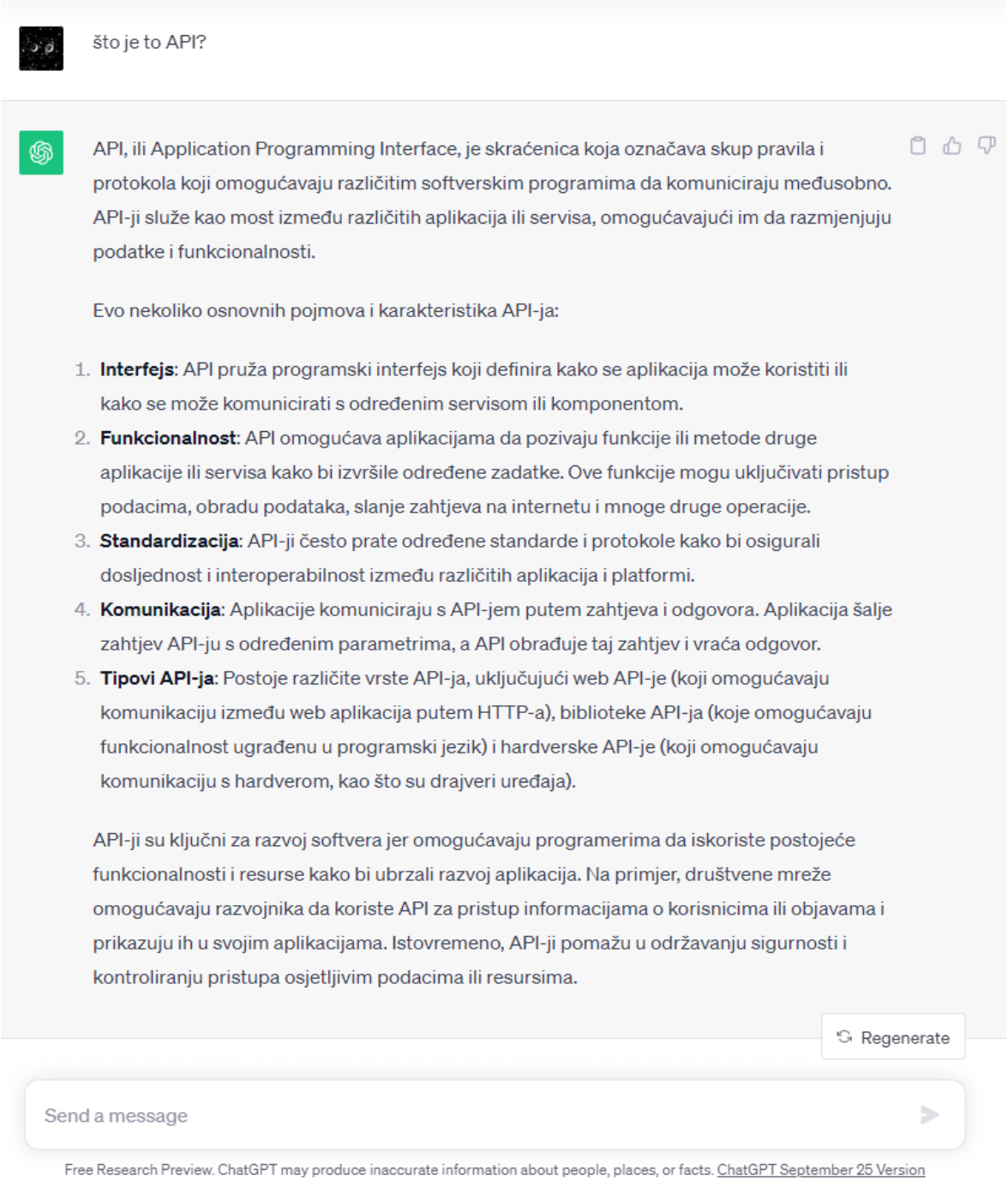
2.2 ChatGPT

Razvijen od strane tvrtke OpenAI, ChatGPT je veliki jezični model koji predstavlja vrhunsko dostignuće u području generativne umjetne inteligencije. Njegova sposobnost generiranja prirodnog jezika otvara vrata širokom spektru primjena. Veliki jezični model ChatGPT temelji se na obradi prirodnog jezika (engl. *Natural Language Processing*) što mu omogućuje komunikaciju s ljudima na način koji je sličan ljudskom razgovoru.

Za posebno komplicirane zadatke kao što je obrada i razumijevanje prirodnog jezika, neuronske mreže mogu imati milijarde parametara. ChatGPT se temelji na arhitekturi GPT-3.5 (engl. *Generative Pre-trained Transformer*) koja koristi transformatorske neurone i ima 175 milijardi parametara [1]. Transformer arhitektura je napredna arhitektura neuronskih mreža koja se temelji na mehanizmu pozornosti. Tijekom treniranja, modelu se prezentira niz tokena te se od njega traži da predvidi sljedeći token u nizu, pri čemu su neki od tokena u ulaznom nizu maskirani. To prisiljava model da koristi kontekst nemaskiranih tokena kako bi napravio predviđanje za maskirani token. Upravo to ChatGPT čini posebno učinkovitim za obradu prirodnog jezika. Optimiziran je za razumijevanje i generiranje teksta koji je smislen i kontekstualan te je treniran na velikoj količini tekstualnih podataka iz različitih izvora kao što su knjige, dokumenti, znanstveni radovi, internetski sadržaj, novine i mnogi drugi [2].

Jedna od prednosti ChatGPT-a je ta što se može trenirati na širok raspon tema te se stalno nadograđuje i razvija. To ga čini prilagodljivim i može se koristiti za različite primjene. Osim toga, vrlo je skalabilan, što znači da može obraditi velike količine zahtjeva i generirati mnogo odgovora u vrlo kratkom vremenskom razdoblju.

Poput Midjourney i DALL-E, veliki jezični model ChatGPT prima upute u obliku teksta te se može koristiti za stvaranje različitih vrsta kreativnog sadržaja poput pjesmi ili eseja. Također, može se koristiti i za prevođenje jezika ili odgovaranje na pitanja na informativan način kao što je prikazano na slici 2.2.



Slika 2.2: Primjer upita ChatGPT-u i odgovora

ChatGPT pruža rješenja u različitim područjima kao što su obrazovni i informativni sustavi, stvaranje sadržaja, pisanje koda u različitim programskim jezicima i mnogim drugim. Iako je sposoban generirati informativne i relevantne sadržaje na različite teme, postoji mogućnost da generira netočne ili nerelevantne odgovore. Uz to, iako je model treniran na velikoj količini podataka, treniran je na podacima prikupljenima do 2021. godine te mu nedostaje znanje o nedavnim događajima. Još nekoliko ograničenja ChatGPT-a su nedosljednost, neprikladni ili pristrani odgovori te ograničena korisnost u složenim zadacima.

OpenAI planira proširiti ponudu ChatGPT-a putem ChatGPT API-a, uvesti ChatGPT Enterprise te istražiti dodatne plaćene planove i dodatke za proširenje funkcionalnosti modela.

2.3 Bard

Google već duže vrijeme razvija rješenja za rad s umjetnom inteligencijom. Štoviše, Googleovi inženjeri iz Google Braina, tima za područje dubokog učenja, postavili su nove temelje za razvoj generativne umjetne inteligencije osmišljavanjem transformera, modela strojnog učenja odnosno arhitekturu neuronske mreže koja je značajno unaprijedila obradu prirodnog jezika. Ova arhitektura je poslužila kao temelj za razvoj rješenja generativne umjetne inteligencije kao što su ChatGPT i Google Bard.

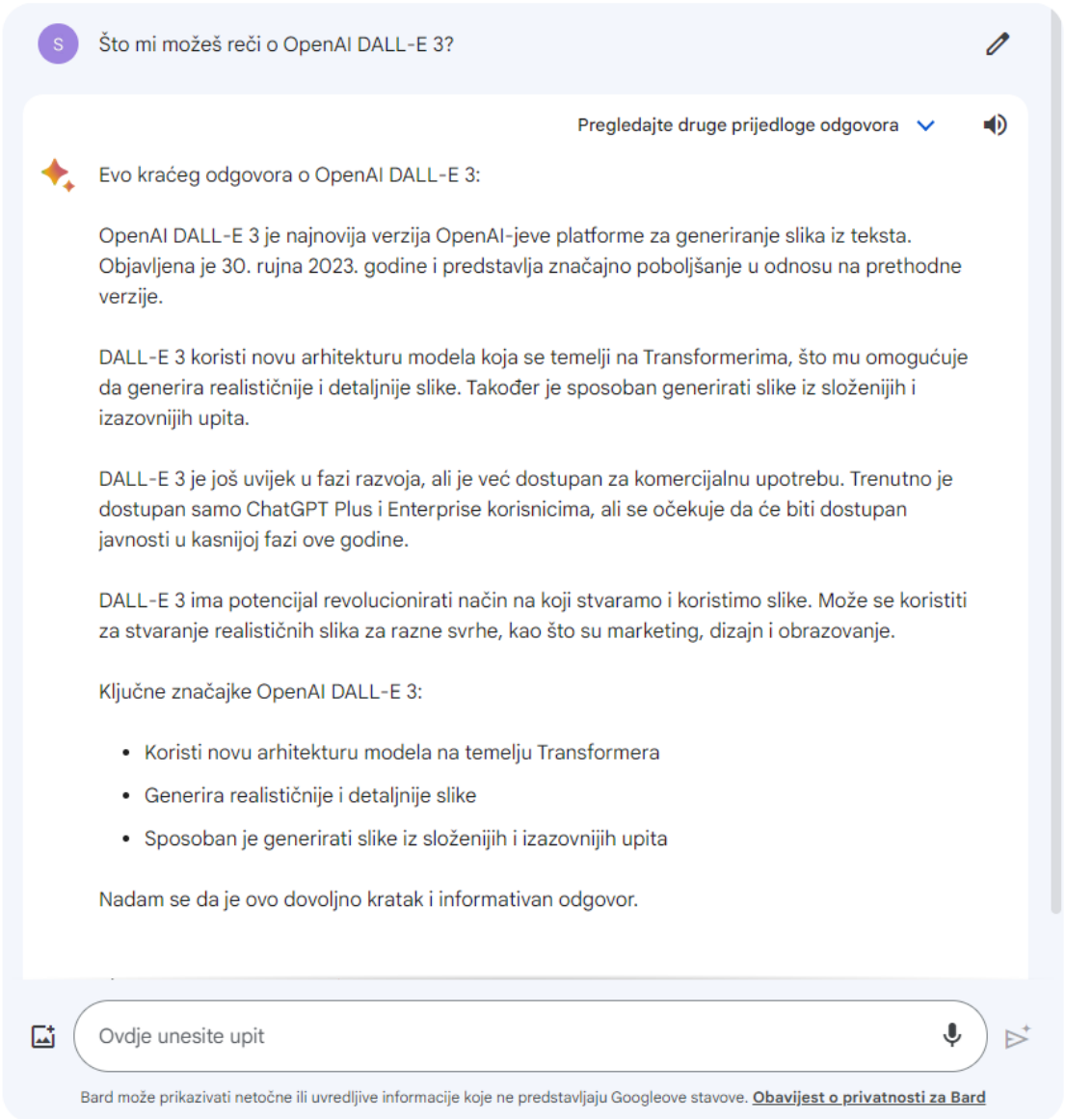
Google Bard je još jedan od generativnih modela umjetne inteligencije. Bard je model za razgovor koji se temelji na velikom jezičnom modelu LaMDA (engl. *Language Model for Dialogue Applications*).

Poput mnogih jezičnih modela, uključujući GPT-3, LaMDA je izgrađen na transformer arhitekturi kojom se stvara model koji se može istrenirati da analizira skup riječi kao što su rečenice ili paragrafi, prateći kako se te riječi odnose jedna na drugu. Nakon treniranja, model može predvidjeti koje će se riječi pojaviti sljedeće. LaMDA je treniran na ogromnom skupu teksta, uključujući dijaloge i druge javno dostupne web dokumente. To daje LaMDA-i široko razumijevanje svijeta i omogućuje generiranje odgovora koji su informativniji i sveobuhvatniji.

Međutim, za razliku od drugih jezičnih modela, LaMDA je fino podešen (engl. *fine-tuned*) na skupu dijaloga kako bi se poboljšala njegova sposobnost generiranja prirodnih i zanimljivih razgovora. Ovaj proces podešavanja pomaže LaMDA-i naučiti obrasce i konvencije ljudskog dijaloga, kao što su način odgovora na pitanja, način praćenja tema i

način održavanja toka razgovora. Kombinacija treniranja na ogromnom skupu teksta i podešavanje na skupu dijaloga čini LaMDA-u raznovrsnim i moćnim jezičnim modelom koji se može koristiti za različite primjene.

S obzirom na to da se Bard temelji na manjem modelu koji zahtjeva značajno manje računalne snage, može se skalirati prema većem broju korisnika, što omogućava više povratnih informacija i poboljšava kvalitetu odgovora. Bard je još uvijek u razvoju, ali je već postao svestrani model za razgovor. Može voditi razgovore na različite teme, uključujući znanost, filozofiju, književnost i pop kulturu. Također može generirati različite kreativne tekstualne formate, sve od pjesama pa do programskog koda.



Slika 2.3: Primjer upita Bardu i odgovora

Slika 2.3 prikazuje primjer upita Bardu i njegovog odgovora. Bard nastoji kombinirati opseg svjetskog znanja s moćnim kapacitetima, inteligencijom i kreativnošću velikih jezičnih modela. Koristeći podatke s interneta, ovaj model pruža aktualne, visokokvalitetne odgovore na raznovrsna pitanja. Bard omogućuje korisnicima brz pristup osnovnim činjenicama, ali također podržava njihovu potrebu za dubljim razumijevanjem i analizom.

Bard nudi slične mogućnosti kao ChatGPT, ali ima i nekoliko jedinstvenih značajki koje ga izdvajaju od ostalih generativnih modela. Ističe se brzinom u generiranju odgovora i istovremeno pruža pristup internetu, otvarajući vrata mnogim prijedlozima i resursima koji pomažu korisnicima u njihovim istraživanjima i kreativnim projektima. Također, odgovori Barda mogu se i poslušati te se ton i stil odgovora može promijeniti. Slušanje odgovora može biti korisno ako korisnik želi čuti ispravan izgovor riječi prilikom, na primjer, učenja novog jezika [3].

Uz prevođenje jezika ili učenje novog jezika, Bard se može koristiti i za kreativno pisanje, može se koristiti kao asistent u programiranju, za analiziranje podataka i na još mnogo različitih načina.

3. INTERAKCIJA S JEZIČNIM MODELIMA UMJETNE INTELIGENCIJE PUTEM API-JA

Jezični modeli umjetne inteligencije se sve češće koriste u različitim aplikacijama, od prevođenja i generiranja teksta do odgovaranja na pitanja. Jedan od načina interakcije s ovim naprednim jezičnim modelima je putem API-ja odnosno sučelja za programiranje aplikacija (engl. *Application Programming Interface*). API predstavlja skup funkcija i podataka koji omogućuju programerima pristup i korištenje usluge ili resursa na lak i efikasan način.

Interakcija s jezičnim modelima putem API-ja ima nekoliko značajnih prednosti. Prvo, API-ji mogu biti intuitivan i učinkovit način za pristup jezičnim modelima. Oni obično koriste standardne protokole i formate što ih čini lakim za integriranje u postojeće aplikacije. Također, API-ji omogućuju programerima da ugrade jezične modele u vlastite aplikacije, čime se izbjegava potreba za razvojem vlastitih modela. To programerima može uštedjeti vrijeme i novac jer ne moraju sami trenirati i održavati jezične modele. Također, korištenjem API-ja može se poboljšati i kvaliteta i performanse aplikacija.

API-ji omogućuju pristup jezičnim modelima koji su razvijeni od strane različitih tvrtki i organizacija, što povećava fleksibilnost i mogućnosti korisnika. To korisnicima daje više mogućnosti za odabir jezičnog modela koji najbolje odgovara njihovim potrebama.

Postoje dvije glavne vrste API-ja za interakciju s jezičnim modelima, a to su API-ji za javnu i API-ji za privatnu upotrebu. API za javnu upotrebu je često besplatan za korištenje. Jedan primjer takvog API-ja je Google Cloud Natural Language API. Međutim, Google Cloud Natural Language API je besplatan za prvih 1000 poziva dnevno po projektu. Nakon toga, troškovi se naplaćuju na temelji količine korištenja.

API-ji za privatnu upotrebu su razvijeni i održavani od strane privatnih tvrtki ili organizacija. Primjer ovih API-ja uključuju OpenAI API te API-je za jezične modele koji se koriste unutar Googleovih proizvoda, kao što su Google Translate i Google Assistant.

Za korištenje API-ja za interakciju s jezičnim modelima, potrebno je dobiti pristup API-ju putem registracije kod pružatelja API-ja. No unatoč brojnim prednostima korištenja API-ja za interakciju s jezičnim modelima, važno je istaknuti i određene nedostatke. API-ji mogu biti ograničeni u smislu svojih funkcionalnosti i sposobnosti u usporedbi s razvojem

vlastitih jezičnih modela i može biti financijski zahtjevno, s obzirom na troškove licenciranja i korištenja API-ja.

API-ji za jezične modele pružaju širok spektar mogućnosti i primjena. Neki od primjera interakcije s jezičnim modelima putem API-ja uključuje korištenje jezičnih modela u industriji obrazovanja, gdje se API-ji mogu koristiti u aplikacijama za personalizaciju učenja i pružanje prilagođenih povratnih informacija studentima. Na primjer, API-ji za interakciju s jezičnim modelima mogu se upotrijebiti u aplikaciji za praćenje napretka studenata i pružanje povratnih informacija o njihovom učenju.

Kroz primjenu API-ja, korisnici dobivaju pristup širokom spektru funkcionalnosti, a jezični modeli postaju ključni resurs za različite aplikacije i industrije. Razvoj i implementacija API-ja za interakciju s jezičnim modelima predstavlja značajan korak prema unaprjeđenju različitih aspekata komunikacije i kreativnosti u digitalnom svijetu.

3.1 Interakcija s OpenAI modelima putem API-ja

OpenAI je tvrtka koja se bavi razvojem i istraživanjem umjetne inteligencije. Jedna od njihovih glavnih aktivnosti je razvoj modela umjetne inteligencije koji se mogu koristiti u raznim primjenama uključujući stvaranje teksta, prevođenje jezika, generiranje različitih kreativnih formata sadržaja i odgovaranje na pitanja.

Ključni alat koji OpenAI nudi svojim korisnicima je API koji omogućuje pristup i korištenje njihovih naprednih modela umjetne inteligencije. OpenAI API je RESTful API, vrsta API-ja koja slijedi principe REST (engl. *Representational State Transfer*) arhitekture i omogućuje komunikaciju s modelima umjetne inteligencije putem HTTP zahtjeva. Svaki zahtjev se sastoji od URL-a, HTTP metode, te tijela zahtjeva. Kako bi se izvodili pozivi na OpenAI API potreban je API ključ za autentifikaciju.

Autentifikacija putem API ključa je jedan od najčešćih načina autentifikacije u RESTful API-jima. API ključ je jedinstveni token koji se korisniku dodijeli kada se registrira za korištenje API-ja. API ključ se zatim koristi za autentifikaciju svakog zahtjeva koje korisnik pošalje API-ju.

OpenAI API nudi širok raspon krajnjih API točaka (engl. *endpoint*) za interakciju s modelima umjetne inteligencije. *Chat completions*, *models* i *completions* neke su od dostupnih API točki. Krajnja točka API-ja je URL koji predstavlja resurs u RESTful API-ju. Svaka krajnja točka API-ja predstavlja određenu operaciju koja se može izvesti na resursu. Na primjer, krajnja točka API-ja *„/v1/chat/completions*“ omogućuje korisnicima

da uspostave interaktivne razgovore s modelima umjetne inteligencije. Ova funkcionalnost može se koristiti za razvoj virtualnih asistenata, simulaciju razgovora i slične svrhe.

OpenAI API nudi interakciju s raznim modelima umjetne inteligencije. Model GPT-4 je najnoviji model koji OpenAI nudi. To je veliki jezični model koji je treniran na masovnom skupu podataka teksta i koda. Model GPT-3.5-turbo je prethodna verzija GPT-4 modela i može se koristiti za iste zadatke kao i GPT-4, uključujući stvaranje teksta, prevođenje jezika i odgovaranje na pitanja. Još jedan od dostupnih modela je i text-davinci-003. Model text-davinci-003 se može koristiti za, na primjer, prevođenje teksta s jednog jezika na drugi u aplikaciji koja treba biti podržana na više jezika. Na primjer, ako aplikacija ima korisnika koji govori engleski i koji želi čitati sadržaj na hrvatskom jeziku, text-davinci-003 se može koristiti za prevođenje tog sadržaja s engleskog na hrvatski.

OpenAI API predstavlja moćan i fleksibilan alat koji omogućuje pristup visoko razvijenim modelima umjetne inteligencije. Njegova jednostavnost korištenja i širok spektar primjena čine ga vrijednim resursom za širok krug korisnika s različitim potrebama i s različitim razinama tehničke stručnosti. Kroz API, OpenAI nastavlja poticati razvoj i širenje korištenja umjetne inteligencije u različitim sektorima i aplikacijama.

3.1.1 **Interakcija s jezičnim modelom ChatGPT**

Objavom svog API-ja, OpenAI je svima omogućio pristup mogućnostima ChatGPT-a, olakšavajući integraciju mogućnosti ChatGPT-a u različite aplikacije. Važno je napomenuti da ChatGPT API predstavlja općeniti termin koji obuhvaća OpenAI API-je koji koriste GPT temeljene modele za razvoj razgovornih sustava, uključujući modele poput GPT-3.5-turbo i GPT-4.

Modeli GPT-3.5-turbo i GPT-4 dostupni putem OpenAI API-ja su isti modeli koji se koriste u ChatGPT i ChatGPT+ uslugama. ChatGPT API je prvenstveno optimiziran za razgovor, ali također je učinkovit i u zadacima dopune teksta (engl. *completions*). Oba modela, GPT-3.5-turbo i GPT-4 su moćniji i jeftiniji od prethodnih GPT-3 modela.

GPT modeli pružaju tekstualne odgovore na ulazne upite. Oblikovanjem upita, pružanjem uputa ili primjera kako uspješno izvršiti zadatak, je zapravo način kako se GPT model programira odnosno kako mu se kaže što da učini. Korištenjem GPT modela moguće je izgraditi različite aplikacije koje se mogu koristiti za sastavljanje dokumenata, pisanje računalnih kodova, analizu tekstova, prevođenje jezika i mnogo više.

Za korištenje GPT modela putem OpenAI API-ja, potrebno je poslati zahtjev koji sadrži upit i API ključ. Modeli GPT-4 i GPT-3.5-turbo pristupaju se putem API točke za dopune razgovora (engl. *chat completions*). Ovi modeli uzimaju popis poruka kao ulaz i vračaju poruku generiranu modelom kao izlaz. Iako je format razgovora osmišljen kako bi olakšao razgovore s višestrukim upitima, jednako je koristan i za zadatke s jednim upitom bez ikakvog razgovora.

Upit koji se šalje u zahtjevu, definiran je parametrom *messages*. Ovaj parametar sadrži niz objekata, gdje svaki objekt ima svoju ulogu, koja može biti označena kao „system“, „user“ ili „assistant“, te ima sadržaj koji predstavlja upit. Razgovori mogu biti različite duljine, uključujući kratke interakcije s jednim upitom i odgovorom, ili duže sekvence gdje se izmjenjuju upiti i odgovori.

Razlika između *chat completions* krajnje točke API-ja i *completions* točke nalazi se u načinu kako se koriste i prilagođavaju upiti odnosno zahtjevi. *Chat completions* ima mogućnost prilagodbe formata tako da zahtjev koristi samo jednu korisničku poruku odnosno upit, što ga čini sličnim *completions* formatu. Na primjer, umjesto upita za *completions* da bi se preveo engleski tekst na francuski: „Prevedi sljedeći engleski tekst na francuski: '{text}'“, za *chat completions* API točku, ekvivalentni upit bio bi: „[{"role": "user", "content": 'Prevedite sljedeći engleski tekst na francuski: "{text}"'}]”. S druge strane, *completions* krajnja točka API-ja može se prilagoditi da simulira razgovor između korisnika i asistenta oblikovanjem unosa prema tom obrascu.

Važno je napomenuti da glavna razlika između ovih API-ja proizlazi iz temeljnih GPT modela koji su im dostupni. *Chat completions* API koristi najmoćnije modele i najučinkovitije modele. Osim toga, *chat completions* koristi se za slanje niza poruka u kontekstu razgovora s modelom, dok se *completions* koristi za slanje pojedinačnih poruka modelu.

OpenAI API omogućuje raznolike primjene umjetne inteligencije u različitim aplikacijama. Krajnji korisnici već koriste aplikacije koje koriste ChatGPT kako bi poboljšale svoje funkcionalnosti. Otvorivši vrata naprednim jezičnim sposobnostima, ChatGPT API postaje ključan alat za razvoj aplikacija koje poboljšavaju korisničko iskustvo.

3.1.2 **Glasovna interakcija s jezičnim modelima (Whisper)**

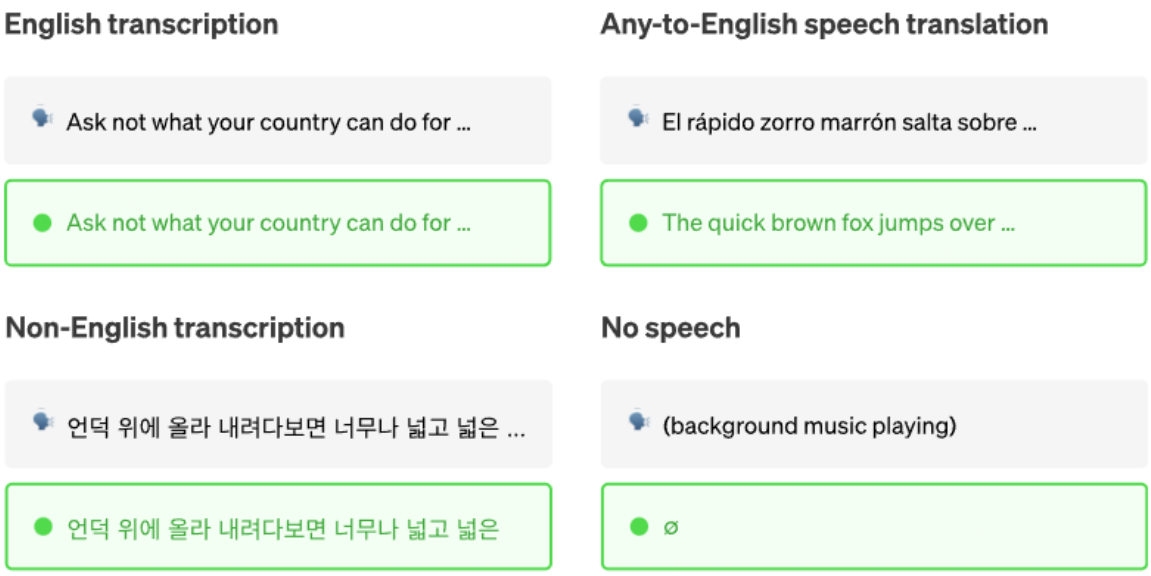
Kako mogućnosti OpenAI-ja naglo rastu, OpenAI API se sve više i više primjenjuje za zadatke koji su se prije smatrali nemogućima. Još jedan od modela koje OpenAI API nudi je Whisper v2-large model koji se može koristiti za širok raspon aplikacija koje iskorištavaju najnovije mogućnosti pretvaranja govora u tekst.

Whisper je model za prepoznavanje govora opće namjene i koristi tehnologiju automatskog prepoznavanja govora (engl. *Automatic Speech Recognition, ASR*). ASR je sustav koji pretvara govor u pisani tekst (engl. *speech-to-text*). Model Whisper v2-large trenutno je dostupan putem OpenAI API-ja kao model whisper-1.

Whisper je treniran na velikom skupu različitih audio zapisa, točnije na 680 000 sati višejezičnih i višezadaćnih podataka prikupljenih s weba. Za treniranje Whisper modela korištena je *sequence-to-sequence* metoda učenja, što znači da je model treniran na mnogo različitih zadataka obrade govora, uključujući višejezično prepoznavanje govora, prevođenje s jednog jezika na drugi, identifikaciju govora na različitim jezicima te detekciju glasa. Whisper v2-large treniran je na audio segmentima duljine od 30 sekundi i ne može obraditi dulje audio zapise odjednom što nije problem sa skupovima podataka koji se sastoje od kratkih izjava. Međutim, predstavlja izazove u stvarnim primjenama koje često zahtijevaju transkripciju audio materijala koji su dugi nekoliko minuta ili sati. No OpenAI je razvio strategiju za izvođenje transkripcije dugih audio zapisa pomoću postupnog transkribiranja 30-sekundnih segmenata zvuka i pomicanja prozora prema vremenskim oznakama koje predviđa model.

Whisper je podržan na velikom broju različitih jezika, premašujući čak 50 jezika. Osnovni model treniran je na 98 različitih jezika, no u OpenAI dokumentaciji [4] navedeni su samo oni jezici čija stopa pogreške riječi (engl. *word error rate*) iznosi manje od 50%. Stopa pogreške riječi je vrijednost koja predstavlja priznatu industrijsku normu za procjenu točnosti modela u prepoznavanju govora i pretvaranju govora u tekst. Izvedba Whisper modela u prepoznavanju govora je daleko od savršenstva, ali njegova izvedba na engleskom jeziku vrlo je blizu ljudskoj razini točnosti.

Whisper model se ističe po svojoj izvanrednoj sposobnosti prevođenja bilo kojeg jezika na engleski jezik bez potrebe za posrednim koracima. To znači da korisnici mogu koristiti Whisper u različitim jezičnim okruženjima, bez obzira na jezik izgovornog sadržaja. Slika 3.1 prikazuje mogućnosti Whisper modela.



Slika 3.1: Mogućnosti Whisper v2-large modela [4]

Osim toga, Whisper podržava različite formate ulaznih audio datoteka, uključujući sljedeće formate:

- m4a
- mp3
- webm
- mp4
- mpga
- wav
- mpeg

OpenAI API nudi dvije API točke za pristup i korištenje Whisper v2-large modela, a to su *transcriptions* i *translations*. To pruža programerima mogućnost korištenja Whisper v2-large modela u različitim aplikacijama koje iskorištavaju najnovije mogućnosti pretvaranja govora u tekst.

Whisper model se može koristiti u različitim scenarijima, pružajući brojne koristi u područjima kao što su transkripcija, obrazovni alati, istraživanje tržišta, indeksiranje audio sadržaja, korisnička podrška te glasovno pretraživanje. Whisper donosi brojne prednosti i mogućnosti koje obogaćuju digitalno iskustvo korisnika.

Na primjer, indeksiranje podcasta i audio sadržaja postaje jednostavnije uz pomoć Whisper modela koji omogućuje transkripciju te tako pristup sadržaju osobama oštećena sluha. Također, poboljšava pretraživanje i dostupnost audio materijala. U službi za korisnike, Whisper može biti koristan za analizu i poboljšanje korisničkog iskustva putem transkripcije i analize poziva u stvarnom vremenu. Također, alati za učenje jezika mogu integrirati Whisper kako bi omogućili studentima vježbanje izgovora i razumijevanje govora u stvarnom vremenu, tako poboljšavajući njihovo jezično znanje.

Dodatna vrijednost dolazi kroz mogućnost kombiniranja Whisper modela s drugim modelima poput GPT-3 ili GPT-4 modela. To omogućuje stvaranje inovativnih aplikacija koje poboljšavaju interakciju između korisnika i računalnih sustava, kao što su aplikacije za kvizove ili generiranje blog postova na temelju audio sadržaja.

Cilj OpenAI-a, kada je u pitanju Whisper model, je razviti jedan svestrani sustav za obradu govora koji pouzdano radi bez potrebe za posebnim fino podešavanjem (engl. *fine-tuning*) skupa podataka kako bi postigao visokokvalitetne rezultate na određenim distribucijama. Whisper model donosi revolucionarne mogućnosti u području obrade govora i audio sadržaja, te kao i drugi modeli dostupni putem OpenAI API-ja, otvara vrata za brojne inovacije u digitalnom svijetu.

3.2 Interakcija s Bard modelom za razgovor putem API-ja

Google Bard API je API koji omogućuje komunikaciju s Bard modelom za razgovor. Međutim, važno je napomenuti da je trenutno Google Bard API još uvijek u beta verziji te se ponaša drugačije od verzije Barda koji je dostupan putem web preglednika. Za razliku od verzije Barda koji je dostupan preko web preglednika, putem Bard API-ja može se koristiti PaLM model koji nema pristup internetu. Također, kako bi se dobio pristup Google Bard API-ju, potrebno je prijaviti se na listu čekanja.

Ukoliko je pristup odobren, Bard se može integrirati u vlastite aplikacije putem Google Bard API-ja korištenjem Google Cloud Platforme ili korištenjem API ključa i HTTP zahtjeva. Google Cloud Platforma je skup usluga i alata za razvoj, puštanje u rad i upravljanje aplikacijama i podacima u oblaku. Bard je dostupan kao API za Google Cloud Platformu, što omogućuje programerima da ga integriraju u vlastite aplikacije koje se nalaze u oblaku.

Osim korištenja Google Cloud Platforme, Bard odnosno PaLM model, može se integrirati u vlastitu aplikaciju i korištenjem API ključa i slanjem HTTP zahtjeva. Nakon

što se stvori zahtjev, potrebno ga je poslati na API točku putem HTTP protokola. API će obraditi zahtjev te vratiti odgovor na temelju upita i argumenata koji su navedeni u zahtjevu.

Model dostupan putem API-ja je, za razliku od modela dostupnog putem web preglednika, dobar u izvođenju zadataka koji uključuju obradu ili generiranje teksta. Međutim, zbog nedostatka pristupa internetu, loš je u pružanju činjeničnih podataka ili odgovaranju na pitanja gdje se traže aktualne informacije.

Iako se trenutno nalazi u beta verziji, Google Bard API predstavlja mogućnost komunikacije s Bard modelom za razgovor. Ovaj API omogućuje integraciju Barda u različite aplikacije. Iako je model koji je dostupan putem API-ja ograničen u pružanju aktualnih informacija, koristan je za obradu i generiranje teksta u različitim aplikacijama. Odobrenim pristupom Google Bard API-ju otvaraju se brojne mogućnosti za razvoj inovativnih projekata.

3.3 Interakcija s jezičnim modelima putem Hugging Face API-ja

Hugging Face je platforma za strojno učenje i znanost o podacima te zajednica koja pomaže korisnicima izgraditi, implementirati i trenirati različite modele umjetne inteligencije. Platforma Hugging Face pruža alate i resurse za izgradnju i implementaciju modela umjetne inteligencije, a Hugging Face API je jedan od njenih najpopularnijih proizvoda. Hugging Face API predstavlja snažan alat za interakciju s jezičnim modelima iz različitih izvora. Ovaj RESTful API omogućuje korisnicima korištenje velikog broja jezičnih modela za različite svrhe.

Hugging Face API je otvorenog koda i dostupan je svima koji se registriraju na web stranici Hugging Face. Nakon registracije, moguće je generirati API ključ koji omogućuje pristup samom API-ju. Ovaj API može se koristiti u različitim programskim jezicima, što ga čini izuzetno pristupačnim i fleksibilnim za različite korisnike.

Hugging Face API podržava preko 150 000 javno dostupnih modela uključujući mnoge od najpopularnijih transformer modela, kao što su GPT-2, BERT, T5, RoBERTa, DistilBERT, BART, i mnoge druge. Osim toga, korisnici imaju mogućnost preuzimanja i upravljanja vlastitim privatnim modelima.

Zahvaljujući ogromnom broju različitih modela koji su dostupni putem Hugging Face API-ja, ključna prednost je mogućnost odabira modela ovisno o specifičnim potrebama aplikacije koju želimo razviti. Različiti modeli su obučeni na različitim skupovima

podataka i imaju različite sposobnosti. Na primjer, ukoliko je cilj razviti aplikaciju za prijevod teksta iz jednog jezika u drugi, potrebno je odabrati model koji je specifično obučen na skupu podataka teksta na različitim jezicima. Ovdje bi se modeli poput transformer modela za prijevod jezika, kao što su BERT ili RoBERTa, mogli pokazati izuzetno korisnima. Također, model BERT koji je dobar u razumijevanju konteksta te pružanju preciznih odgovora na pitanja, može se koristiti i u izradi aplikacije koja je sposobna odgovarati na pitanja o određenoj temi.

Hugging Face API predstavlja izvanredan alat za interakciju s jezičnim modelima umjetne inteligencije. Njegova pristupačnost, širok spektar dostupnih modela te mogućnost odabira modela prema specifičnim potrebama čine ga popularnim izborom među istraživačima, poduzećima i pojedincima koji žele izraditi aplikacije za generiranje teksta, prijevod jezika ili odgovaranje na pitanja.

4. PROGRAMSKA PODRŠKA ZA INTERAKCIJU S JEZIČNIM MODELIMA UMJETNE INTELIGENCIJE

Za izradu biblioteke koja pruža programsku podršku za interakciju s jezičnim modelima korišten je koncept klasnih biblioteka u .NET-u koji predstavlja ideju dijeljenih biblioteka. Dijeljene biblioteke omogućuju organizaciju i razdvajanje korisnih funkcionalnosti u samostalne komponente koje se zatim mogu koristiti u različitim aplikacijama [5]. Dijeljena biblioteka predstavlja datoteku koja se koristi za dijeljenje resursa između izvršnih datoteka i drugih objektnih datoteka.

Klasna biblioteka definira i pruža metode koje se koriste i pozivaju iz aplikacija. Ova biblioteka pruža mogućnost slanja http zahtjeva OpenAI API-ju i Hugging Face API-ju. Pomoću metode CallAI može se zvati jedan od ova dva API-ja ovisno o parametru *APIProvider* koji se proslijedi metodi. Također, biblioteka sadrži sljedeća svojstva kojima se mogu, a određenim navedenim svojstvima i moraju, pridružiti vrijednosti:

- Endpoint
- APIKey
- Model
- Temperature
- MaxTokens
- AudioFilePath

Važno je da se prije poziva metode koja omogućuje poziv API-ja, dodijele vrijednosti sljedećim svojstvima:

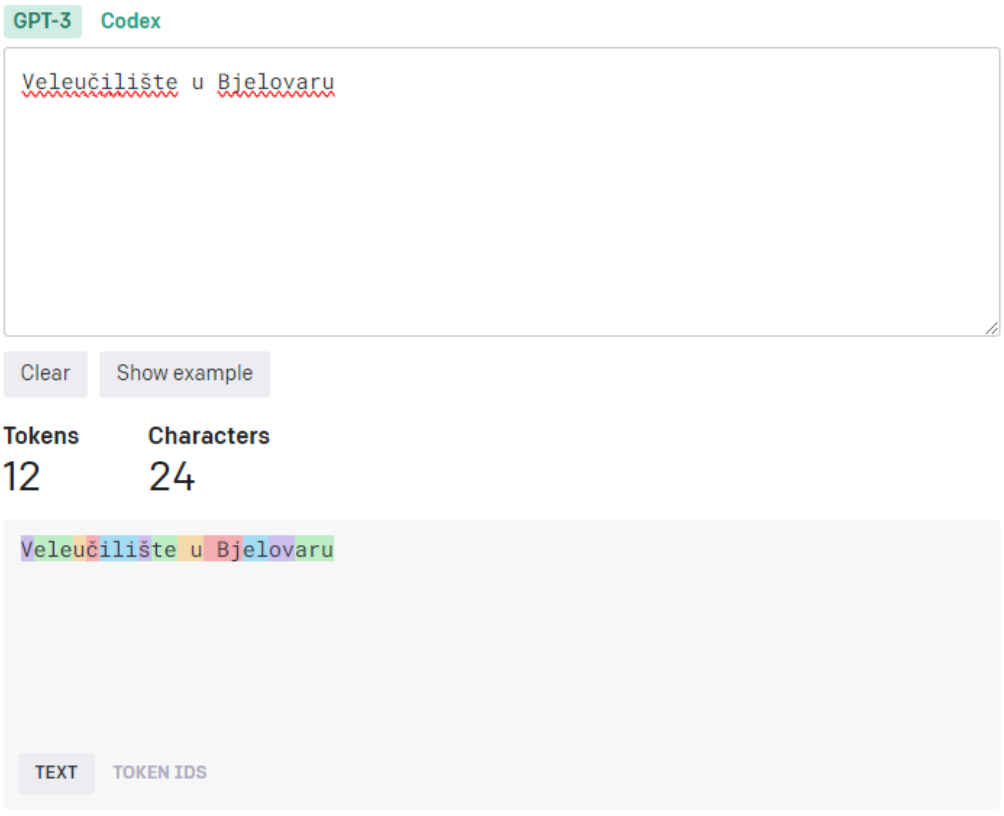
- Endpoint
- APIKey
- Model

Endpoint je svojstvo koje pohranjuje URL odnosno krajnju točku API-ja s kojim se komunicira. To je adresa na koju će se slati HTTP zahtjevi kako bi se omogućila interakcija s modelima umjetne inteligencije. APIKey je svojstvo koje pohranjuje API ključ koji se koristi za autentifikaciju kod komunikacije s API-jem, a svojstvo Model pohranjuje informaciju o modelu koji će se koristiti u aplikaciji.

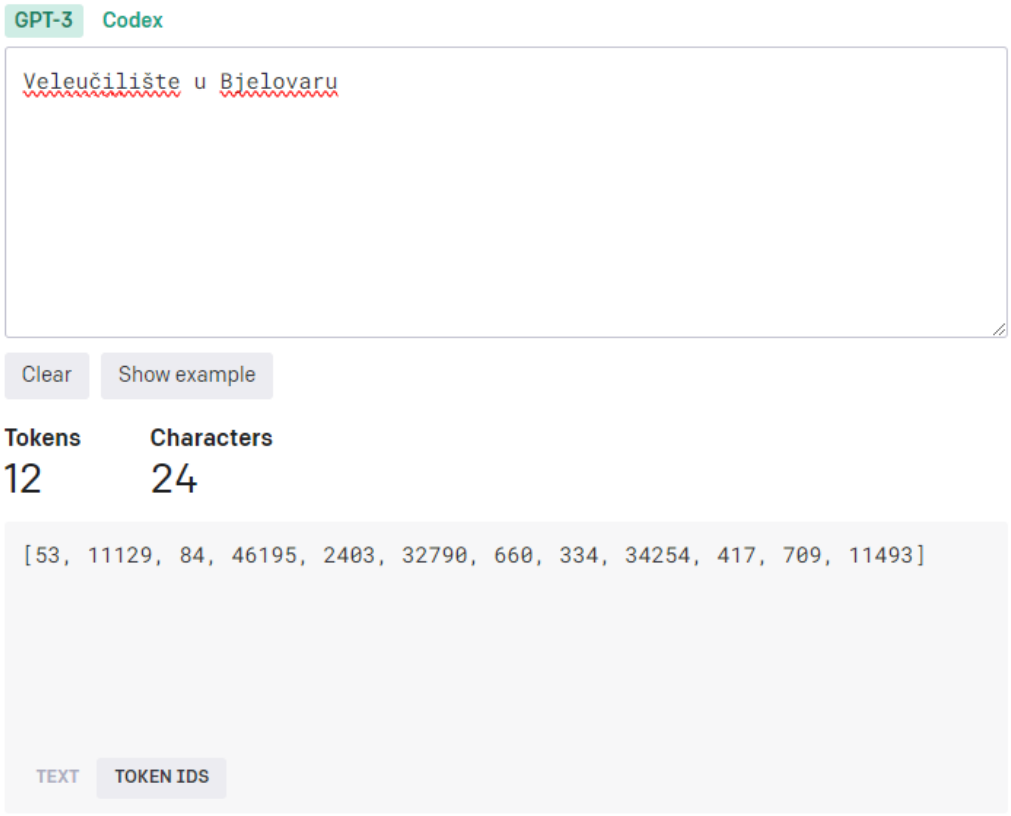
Ako se svojstvima Temperature i MaxTokens ne pridruže vrijednosti, Temperature će iznositi 1, a MaxTokens će iznositi 1000. Temperatura je svojstvo koje pohranjuje numeričku vrijednost (cijeli broj) koja može utjecati na generirane rezultate. Kada je

temperatura iznad 0, slanje istog upita nekoliko puta za redom rezultirat će različitim generiranim odgovorom svaki put. Model procjenjuje koji tekst je najvjerojatniji da će slijediti tekst koji prethodi. Temperature je vrijednost između 0 i 1 koja zapravo omogućuje kontrolu koliko model treba biti samouvjeren i kreativan tijekom svojih predviđanja. Smanjenje temperature znači da će model preuzeti manje rizika, a odgovori će biti precizniji i predvidljiviji, dok će povećanje temperature donijeti raznovrsnije odgovore. MaxTokens svojstvo pohranjuje cijeli broj koji predstavlja maksimalni broj tokena. Prije nego što API obradi poslani upit, taj upit se razbija na tokene. Tokeni se mogu zamisliti kao dijelovi riječi. Ponekad su to slogovi, a ponekad dio riječi ili jedno slovo. Tokeni mogu uključivati i prateće praznine te je važno imati na umu da upiti koji završavaju znakom razmaka mogu rezultirati manje kvalitetnim odgovorom.

Na slikama 4.1 i 4.2 prikazan je način kako API generira tokene za unos “Veleučilište u Bjelovaru”.



Slika 4.1: Primjer kako API pretvara upit u tokene (tekst)



Slika 4.2: Primjer kako API pretvara upit u tokene (token ID)

Ovisno o korištenom modelu, upiti mogu koristiti do ukupno 4097 tokena za upit i odgovor. Ako je upit 4000 tokena, odgovor može imati najviše 97 tokena. Ograničenje je trenutno tehničke prirode, no često postoje kreativni načini za rješavanje problema unutar tog ograničenja, primjerice, skraćivanjem samog upita, razbijanjem teksta na manje dijelove i slično.

Tokeni funkcioniraju na sljedeći način, na primjer, GPT-3 uzima upit i pretvara ga u listu tokena, zatim obrađuje upit te potom pretvara predviđene tokene natrag u riječi koje vidimo u odgovoru.

Ono što nama može izgledati kao dvije identične riječi može biti generirano u različite tokene ovisno o tome kako su strukturirane unutar teksta. Korisno okvirno pravilo je da se jedan token obično podudara s otprilike 4 znaka ili slova engleske abecede, pa tako 100 tokena jednako je otprilike 75 riječi.

Svojstvu AudioFilePath potrebno je pridružiti vrijednost ukoliko se želi koristiti OpenAI model Whisper v2-large za glasovnu interakciju s modelima.

Sva ova svojstva omogućuju pohranjivanje vrijednosti i pristupanje različitim parametrima koji se koriste za konfiguriranje i komunikaciju s API-jem.

Jedna od metoda je već spomenuta metoda AICall, koja prima parametre *APIProvider*, *promptText* i *voiceInteraction*. To je metoda koja se koristi za poziv Hugging Face ili OpenAI API-ju. U programskom kodu 4.1 prikazana je metoda AICall.

Programski kod 4.1: Metoda za poziv OpenAI ili Hugging Face API-ju

```
public async Task<string> AICall(string APIProvider, string promptText, bool
 voiceInteraction = false) {
   PromptText = promptText;

   if (string.IsNullOrWhiteSpace(Endpoint) ||
      string.IsNullOrWhiteSpace(APIKey) ||
      string.IsNullOrWhiteSpace(PromptText) ||
      string.IsNullOrWhiteSpace(Model)) {

         throw new ArgumentException("Endpoint, APIkey and model must be
         defined and non-empty.");
   }

   if (APIProvider == "OpenAI") {
      if (voiceInteraction) {
         AudioFilePath = promptText;
         promptText = await VoiceAICall();
         return await OpenAICall(promptText);
      }
         return await OpenAICall(promptText);
   } else if (APIProvider == "HuggingFace") {
      return await HuggingFaceCall(PromptText);
   } else {
      throw new ArgumentException("APIProvider must be 'OpenAI' or
      'HuggingFace'.");
   }
}
```

Za potrebe testiranja biblioteke napravljena je i .NET konzolna aplikacija. Konzolna aplikacija se izvršava putem naredbenog retka i konzolne aplikacije obično koriste naredbeni redak za unos podataka od korisnika te za ispis rezultata na ekran. Kada je stvoren, novi projekt konzolne aplikacije nema pristup klasnoj biblioteci. Da bi se omogućilo pozivanje metoda iz biblioteke, stvara se referenca na projekt klasne biblioteke koristeći sljedeću naredbu koju je potrebno okinuti u naredbenom retku: "*dotnet add ShowCase/ShowCase.csproj reference AILang/AILang.csproj*". Nakon dodane reference, moguće je u konzolnoj aplikaciji koristiti metode koje klasna biblioteka nudi. U

programskom kodu 4.2 prikazan je primjer korištenja biblioteke za glasovnu interakciju s modelom text-davinci-001.

Programski kod 4.2: Primjer korištenja biblioteke za glasovnu interakciju

```
static void Main(string[] args) {
        int row = 0;
        var AIInteraction = new AIInteract();
        AIInteraction.Endpoint = "https://api.openai.com/v1/completions";
        AIInteraction.APIKey = "myAPIKey";
        AIInteraction.Model = "text-davinci-001";

        do {
            if (row == 0 || row >= 25)
                ResetConsole();

            string? input = Console.ReadLine();
            if (string.IsNullOrEmpty(input)) break;
            Console.WriteLine($"Input: {input}");
            Console.WriteLine("OpenAI response: " +
AIInteraction.AICall("OpenAI", input, true).GetAwaiter().GetResult());
            row += 4;
        } while (true);
        return;

        void ResetConsole() {
            if (row > 0) {
                Console.WriteLine("Press any key to continue...");
                Console.ReadKey();
            }
            Console.Clear();
            Console.WriteLine($"{Environment.NewLine}Press <Enter> only to
exit; otherwise, enter a string and press <Enter>:{Environment.NewLine}");
            row = 3;
        }
    }
```

4.1 Podrška za interakciju putem OpenAI API-ja

S OpenAI API-jem može se komunicirati putem HTTP zahtjeva. Svi API zahtjevi trebaju sadržavati API ključ u *Authorization* HTTP zaglavlju na sljedeći način: Authorization: Bearer OPENAI_API_KEY.

Metoda OpenAICall, prikazana u programskom kodu 4.2, obavlja poziv na OpenAI API kako bi generirala tekst na temelju proslijeđenog *prompt* parametra. Metoda je asinkrona i vratit će rezultat odnosno odgovor modela umjetne inteligencije kao niz znakova kada se završi izvršavanje.

Programski kod 4.3: Metoda za poziv OpenAI API-ja

```
private async Task<string> OpenAICall(string prompt) {

   HttpClient httpClient = new HttpClient();
   httpClient.DefaultRequestHeaders.Add("Authorization", $"Bearer
{APIKey}");

   var request = new StringContent($"{{\"model\": \"{Model}\", \"prompt\":
\"{prompt}\", \"temperature\": {Temperature}, \"max_tokens\":
{MaxTokens}}}", Encoding.UTF8, "application/json");
   string requestContent = await request.ReadAsStringAsync();
   HttpResponseMessage response = await httpClient.PostAsync(endpoint,
request);

   if (!response.IsSuccessStatusCode) {
      throw new Exception("Failed to call OpenAI API, " +
      response.StatusCode);
   }

   string responseContent = response.Content.ReadAsStringAsync().Result;
   JObject jsonResponse = JObject.Parse(responseContent);

   if (jsonResponse["choices"] is JArray choicesArray && choicesArray.Count
> 0) {
      var textToken = choicesArray[0]["text"];
      if (textToken != null) {
         string text = textToken.ToString();
         return text;
      }
   }
   return "No text found in the response";
}
```

Prvo se stvara novi HttpClient objekt, koji će se koristiti za izvršavanje HTTP zahtjeva prema OpenAI API-ju. HttpClient je dio .NET Framework-a koji omogućava komunikaciju s web servisima. Zatim se postavlja *Authorization* zaglavlje HTTP zahtjeva. Varijabla request sadrži informacije o modelu, upitu, temperaturi i maksimalnom broju tokena. Svi ovi podaci šalju se u JSON (engl. *JavaScript Object Notation)* formatu u tijelu

zahtjeva. Asinkroni HTTP POST zahtjev prema OpenAI API-ju koristi stvoren httpClient objekt za slanje zahtjeva prema krajnjoj točki API-ja s prethodno stvorenim JSON sadržajem. Ako je HTTP zahtjev uspješno poslan, sprema se sadržaj odgovora u obliku niza znakova te se zatim parsira kao JSON objekt pomoću klase JObject iz Newtonsoft.Json biblioteke. Nakon parsiranja odgovora, provjerava se postoji li u odgovoru polje *choices* i ima li polje barem jedan element. Ako je to slučaj, dohvaća se prvi element i iz njega se dohvaća *text* polje koje je zapravo sam odgovor modela na upit.

Ovaj kod omogućuje komunikaciju s OpenAI API-om, slanje zahtjeva s odgovarajućim parametrima te obradu odgovora kako bi se dobio generirani tekst na temelju zadanog upita.

4.2 Podrška za glasovnu interakciju s jezičnim modelima umjetne inteligencije

Ova biblioteka omogućuje glasovnu interakciju s jezičnim modelima umjetne inteligencije pomoću Whisper v2-large modela. Whisper model se može koristiti za prevođenje ili transkripciju audio zapisa. U ovoj biblioteci, omogućena je glasovna interakcija s jezičnim modelima na engleskom jeziku te se Whisper model koristi za transkribiranje audio zapisa. Kako bi se koristio Whisper model, potrebno je AICall metodi proslijediti parametar *voiceInteraction* kao *true*.

U programskom kodu 4.4 prikazan je dio AICall metode pomoću kojeg se izvodi glasovna interakcija s modelima umjetne inteligencije.

Programski kod 4.4: Poziv metode VoiceAICall unutar AICall metode

```
public async Task<string> AICall(string APIProvider, string promptText, bool
voiceInteraction = false) {
   ...

   if (APIProvider == "OpenAI") {
      if (voiceInteraction) {
        AudioFilePath = promptText;
        promptText = await VoiceAICall();
        return await OpenAICall(promptText);
      }
      return await OpenAICall(promptText);
   }
   ...
```

Prilikom korištenja Whisper modela, API za transkripciju, umjesto tekstualnog upita, prima audio datoteku kao ulaz i obavlja transkripciju te rezultat isporučuje u željenom formatu. Važno je napomenuti da audio datoteka koja se prosljeđuje mora biti u jednom od podržanih formata. Koristeći Whisper v2-large model, napravi se transkripcija zadanog audio zapisa. Nakon toga, dobivena transkripcija se šalje OpenAICall metodi i koristi se kao ulaz, odnosno upit, za druge modele.

Metoda za glasovnu interakciju s jezičnim modelima umjetne inteligencije prikazana je u programskom kodu 4.5.

Programski kod 4.5: VoiceAICall metoda

```
private async Task<String> VoiceAICall() {
   using var content = new MultipartFormDataContent();
   using var audioData = new MemoryStream(File.ReadAllBytes(AudioFilePath));

   content.Add(new ByteArrayContent(audioData.ToArray()), "file",
    AudioFilePath);
   content.Add(new StringContent("whisper-1"), "model");

   HttpClient httpClient = new HttpClient();
   httpClient.DefaultRequestHeaders.Add("Authorization", $"Bearer
    {APIKey}");
   var response = await
    httpClient.PostAsync("https://api.openai.com/v1/audio/transcriptions",
    content);

   if (!response.IsSuccessStatusCode) {
      throw new Exception("Failed to call OpenAI API (voice), " +
       response.StatusCode);
   }

   string responseContent = await response.Content.ReadAsStringAsync();

   JObject jsonResponse = JObject.Parse(responseContent);

   if (jsonResponse["text"] != null) {
      string text = jsonResponse["text"].ToString();
      return text;
   }
      return "No text found in the response";
}
```

U metodi se prvo stvara MultipartFormDataContent objekt koji će se koristiti za slanje podataka na OpenAI API. MultipartFormDataContent se obično koristi za slanje podataka koji uključuju različite vrste sadržaja, u ovom slučaju, audio datoteku i ime modela koji će se koristiti. Zatim se koristi MemoryStream za učitavanje sadržaja audio datoteke. Audio datoteka se čita kao niz bajtova iz datoteke koja se nalazi na putanji AudioFilePath.

Nakon što se audio datoteka učita u memoriju, dodaje se u MultipartFormDataContent objekt kao ByteArrayContent pod imenom „*file*“ i predstavlja audio datoteku koju Whisper v2-large model mora transkribirati. Također, u tijelo zahtjeva se dodaje i informacija o modelu koji će se koristiti za transkripciju, pa se model postavlja na „whisper-1“ odnosno Whisper v2-large.

Za slanje HTTP zahtjeva na OpenAI API koristi se objekt httpClient. U zahtjevu se također postavlja *Authorization* zaglavlje, koristeći API ključ.

Zatim se šalje POST zahtjev na krajnju točku API-ja za transkripcije „*https://api.openai.com/v1/audio/transcriptions*“. Na slici 4.3 prikazan je primjer odgovora Whisper modela.

```
OpenAI response: {"text":"Give me the planets of our solar system."}
```

Slika 4.3: Primjer Whisper odgovora

Ako je odgovor uspješan, tekst iz odgovora se čita i sprema u varijablu *text*. Ako postoji tekst u odgovoru, vraća se taj tekst, ako nema teksta u odgovoru, vraća se poruka *„No text found in the response“*. Ova metoda omogućuje komunikaciju s OpenAI API-jem za pretvaranje govora u tekst i vrača rezultat transkripcije (tekst) koji se zatim šalje OpenAICall metodi pomoću koje se zatim dobiva željeni odgovor na temelju glasovnog upita. Na slici 4.4 prikazan je primjer glasovne interakcije s modelom text-davinci-001. U konzolnoj aplikaciji dan je ulaz koji je zapravo putanja do audio datoteke koju želimo proslijediti Whisper modelu i ispisan je odgovor modela text-davinci-001.

```
Press <Enter> only to exit; otherwise, enter a string and press <Enter>:

C:\Users\sarad\Documents\solarSystem.mp3
Input: C:\Users\sarad\Documents\solarSystem.mp3
OpenAI response:

Sun, Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto.
```

Slika 4.4: Primjer glasovne interakcije s text-davinci-001 modelom

4.3 Podrška za interakciju putem Hugging Face API-ja

Hugging Face API može pružiti odgovore po zahtjevu od više od 100 000 modela koji su postavljeni na Hugging Face Hub. S obzirom na to da Hugging Face API nudi tako velik broj različitih modela na korištenje, prvi korak je odabir modela. Ovisno o zadatku za koji je model konfiguriran, zahtjev će prihvatiti određene parametre. Obavezan parametar, bez obzira na model, je *inputs* parametar, odnosno upit koji se šalje modelu [6].

Na primjer, za zadatak popunjavanja rečenice koriste se BERT modeli, čiji je to osnovni zadatak. U takvom zadatku, model pokušava popuniti prazninu s nedostajućom riječi, točnije određenim tokenom. U ovakvom zadatku upit mora biti tekst koji mora sadržavati *„[MASK]"*. Odgovor se dobije u JSON formatu. Slika 4.5 prikazuje primjer upita i odgovora bert-base-uncased modela. U primjeru odgovora može se vidjeti da se sam odgovor modela na korisnikov upit nalazi u polju „*sequence*".


```
Press <Enter> only to exit; otherwise, enter a string and press <Enter>:

The answer to the universe is [MASK].
Input: The answer to the universe is [MASK].
Response: [{"score":0.16964051127433777,"token":2053,"token_str":"no","sequence"
:"the answer to the universe is no."},{"score":0.07344779372215271,"token":2498,
"token_str":"nothing","sequence":"the answer to the universe is nothing."},{"sco
re":0.05803246051073074,"token":2748,"token_str":"yes","sequence":"the answer to
 the universe is yes."},{"score":0.043957971036434174,"token":4242,"token_str":"
unknown","sequence":"the answer to the universe is unknown."},{"score":0.0401572
7713704109,"token":3722,"token_str":"simple","sequence":"the answer to the unive
rse is simple."}]
```


Slika 4.5: Primjer upita i odgovora bert-base-modela

U programskom kodu 4.6 prikazana je metoda HuggingFaceCall pomoću koje se šalju zahtjevi na Hugging Face API.

Programski kod 4.6: HuggingFaceCall metoda

```
private async Task<string> HuggingFaceCall(string prompt) {

   HttpClient httpClient = new HttpClient();
   httpClient.DefaultRequestHeaders.Add("Authorization", $"Bearer
    {APIKey}");

   var requestContent = new
    StringContent(Newtonsoft.Json.JsonConvert.SerializeObject(prompt),
    Encoding.UTF8, "application/json");
   HttpResponseMessage response = await httpClient.PostAsync(Endpoint,
    requestContent);

   if (!response.IsSuccessStatusCode){
      throw new Exception("Failed to call Hugging Face API, " +
       response.StatusCode);
   }

   string responseContent = await response.Content.ReadAsStringAsync();
      return responseContent;
}
```

Prvo, metoda stvara HttpClient objekt koji će se koristiti za slanje HTTP zahtjeva prema Hugging Face API-ju. Nakon stvaranja HttpClient objekta, dodaje se *Authorization* zaglavlje koje koristi API ključ za autentifikaciju korisnika.

Zatim se priprema sadržaj za HTTP POST zahtjev. Upit koji se šalje API-ju serijalizira se u JSON format jer Hugging Face API, prilikom slanja zahtjeva, zahtjeva JSON enkodirani sadržaj. Nakon pripreme zahtjeva, metoda šalje HTTP POST zahtjev prema Hugging Face API-ju putem httpClient.PostAsync metode. Zahtjev uključuje autorizacijsko zaglavlje i JSON sadržaj. Ako je zahtjev uspješan, čita se sadržaj odgovora kao niz znakova koristeći Content.ReadAsStringAsync metodu. Konačno, metoda vraća odgovor kao niz znakova u JSON obliku te se odgovor može dalje analizirati ili koristiti u aplikaciji.

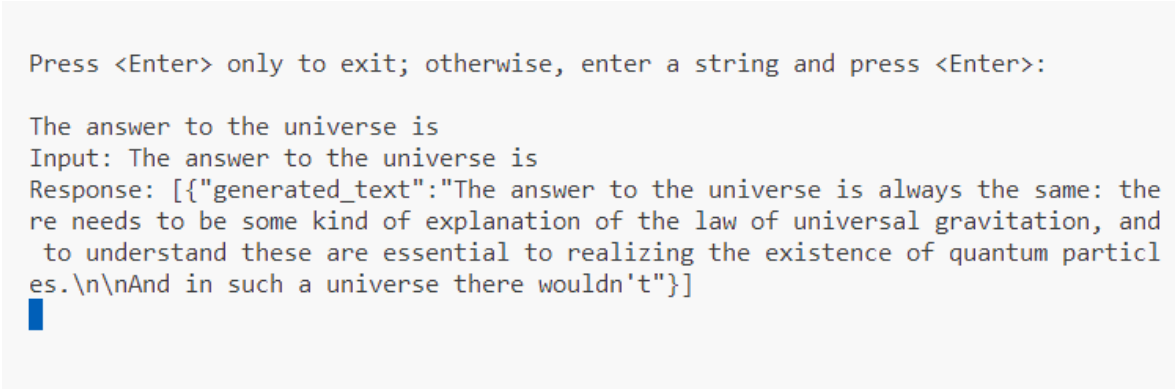
U programskom kodu 4.7 je prikazan primjer pozivanja Hugging Face API-ja u konzolnoj aplikaciji, a na slici 4.6 je prikazan ulaz i pripadajući odgovor modela.

Programski kod 4.7: Primjer poziva Hugging Face API-ja

```
static void Main(string[] args) {
        int row = 0;
        var AIInteraction = new AIInteract("https://api-
inference.huggingface.co/models/gpt2", "myAPIKey", "gpt2");

        do {
            if (row == 0 || row >= 25)
                ResetConsole();

            string? input = Console.ReadLine();
            if (string.IsNullOrEmpty(input)) break;
            Console.WriteLine($"Input: {input}");
            Console.WriteLine("Response: " +
AIInteraction.AICall("HuggingFace", input).GetAwaiter().GetResult());
            row += 4;
        } while (true);
        return;
```



Slika 4.6: Primjer upita i odgovora gpt2 modela putem Hugging Face API-ja

U primjeru se koristi model gpt2 koji je vrlo jednostavan. Koristi se kako bi se nastavio tekst prema uputi, dakle model će nastaviti generirati tekst na poslani upit. Uz gpt2 i BERT modele, Hugging Face nudi interakciju s još mnogo različitih modela za različite zadatke te se pomoću metode AICall, odnosno HuggingFaceCall metode, mogu koristiti ti modeli i u vlastitim aplikacijama.

5. ZAKLJUČAK

U ovom završnom radu opisana je izrada programske podrške za interakciju s modelima umjetne inteligencije. Razvoj umjetne inteligencije i pristup različitim API-jima kao što su OpenAI API i Hugging Face API, omogućio je integraciju modela umjetne inteligencije u različite projekte i aplikacije, što je dovelo do brojnih novih mogućnosti i rješenja za brojne izazove i probleme. No unatoč postignutim naprecima, postoje i izazovi u vezi sa sigurnošću podataka, performansama modela i etičkim pitanjima koja proizlaze iz korištenja umjetne inteligencije. Daljnja istraživanja i razvoj u ovom području trebali bi biti usmjereni na rješavanje ovih izazova kako bi se osigurala sigurna i učinkovita interakcija s različitim modelima umjetne inteligencije putem API-ja.

Ovaj rad je dao sveobuhvatan pregled interakcije s jezičnim modelima umjetne inteligencije putem API-ja. Interakcija s modelima umjetne inteligencije i njihova integracija u vlastite projekte i aplikacije putem API-ja ima potencijal promijeniti način na koji razmišljamo o aplikacijama i sustavima, nudeći nove mogućnosti i rješenja za brojne izazove.

6. LITERATURA

[1] Vrhovski, Zoran; Dvojković, Sara; Husak, Krunoslav: PREDNOSTI I IZAZOVI KORIŠTENJA AI JEZIČNOG MODELA CHATGPT PRI UČENJU PROGRAMSKIH JEZIKA U VISOKOM OBRAZOVANJU // ZBORNIK RADOVA SA MEĐUNARODNE STUDENTSKE KONFERENCIJE „OBRAZOVANJE ZA ODRŽIVI RAZVOJ“/ / Kiseljak: CEPS – Centar za poslovne studije, 2023. str. 150 - 167

[2] Chakraborty U., Soumyadeep R., Sumit K. Rise of Generative AI and ChatGPT: Understand how Generative AI and ChatGPT are transforming and reshaping the business world (English Edition). London. 2023.

[3] Google Bard Službeni Blog [Online] 2023. Dostupno na: https://blog.google/products/bard/

[4] OpenAI dokumentacija [Online]. 2021. Dostupno na: https://platform.openai.com/docs/introduction

[5] Microsoft. .NET klasne biblioteke [Online]. 2022. Dostupno na: https://learn.microsoft.com/en-us/dotnet/standard/class-libraries.

[6] Hugging Face dokumentacija [Online]. 2021. Dostupno na: https://huggingface.co/docs

7. OZNAKE I KRATICE

ADAS – Advanced Driver Assistance Systems (napredni sustavi za potporu vozačima automobila)

AI – Artificial Intelligence (umjetna inteligencija)

API – Application Programming Interface (sučelje za programiranje aplikacija)

ASR - Automatic Speech Recognition (automatsko prepoznavanje govora)

HTTP – Hypertext Transfer Protocol (protokol prijenosa hiperteksta)

JSON - JavaScript Object Notation

REST - Representational State Transfer (Reprezentacijski prijenos stanja)

URL – Uniform Resource Locator (jedinstveni lokator izvora)

8. SAŽETAK

Naslov: Izrada programske podrške za interakciju s jezičnim modelima umjetne inteligencije

Ovaj rad istražuje interakciju s modelima umjetne inteligencije putem API-ja. Cilj rada je izraditi .NET biblioteku koja olakšava integraciju različitih jezičnih modela u vlastite projekte te omogućuje glasovnu interakciju s jezičnim modelima. Korištenjem OpenAI API-ja i Hugging Face API-ja, izrađena biblioteka omogućuje integraciju modela u brojne projekte i aplikacije, otvarajući vrata novim rješenjima i izazovima. U radu su istražene različite vrste jezičnih modela, uključujući generativne modele umjetne inteligencije. Također, istraženi su koncepti interakcije s jezičnim modelima putem API-ja, uključujući razumijevanje samog pojma API-ja. Interakcija s jezičnim modelima umjetne inteligencije putem API-ja sve je važnija jer ima potencijal transformirati način na koji ljudi koriste umjetnu inteligenciju u različitim područjima. Rad nudi sveobuhvatan pregled interakcije s modelima umjetne inteligencije putem API-ja.

Ključne riječi: umjetna inteligencija, jezični modeli, API, .NET biblioteka.

9. ABSTRACT

Title: Development of Software Support for Interaction with Artificial Intelligence Models

This project investigates the interaction with artificial intelligence models through APIs. The goal of the project is to develop a .NET library that facilitates the integration of different language models into custom projects and enables voice interaction with language models. Using the OpenAI API and the Hugging Face API, the developed library enables the integration of models into numerous projects and applications, opening the door to new solutions and challenges. The project explores different types of language models, including generative artificial intelligence models. Additionally, the concepts of interaction with language models through APIs, including the understanding of the API itself, are explored. Interaction with artificial intelligence models through APIs has the potential to transform the way people use artificial intelligence in different areas. The project provides a comprehensive overview of interaction with artificial intelligence models through APIs.

Keywords: artificial intelligence, language models, API, .NET library.

IZJAVA O AUTORSTVU ZAVRŠNOG RADA

Pod punom odgovornošću izjavljujem da sam ovaj rad izradio/la samostalno, poštujući načela akademske čestitosti, pravila struke te pravila i norme standardnog hrvatskog jezika. Rad je moje autorsko djelo i svi su preuzeti citati i parafraze u njemu primjereno označeni.

Mjesto i datum	Ime i prezime studenta/ice	Potpis studenta/ice
U Bjelovaru, 9.10.2023.	Sara Dvojković	

U skladu s čl. 58, st. 5 Zakona o visokom obrazovanju i znanstvenoj djelatnosti, Veleučilište u Bjelovaru dužno je u roku od 30 dana od dana obrane završnog rada objaviti elektroničke inačice završnih radova studenata Veleučilišta u Bjelovaru u nacionalnom repozitoriju.

Suglasnost za pravo pristupa elektroničkoj inačici završnog rada u nacionalnom repozitoriju

Sara Dvojković
ime i prezime studenta/ice

Dajem suglasnost da tekst mojeg završnog rada u repozitorij Nacionalne i sveučilišne knjižnice u Zagrebu bude pohranjen s pravom pristupa (zaokružiti jedno od ponuđenog):

a) Rad javno dostupan
b) Rad javno dostupan nakon _____ (upisati datum)
c) Rad dostupan svim korisnicima iz sustava znanosti i visokog obrazovanja RH
d) Rad dostupan samo korisnicima matične ustanove (Veleučilište u Bjelovaru)
e) Rad nije dostupan.

Svojim potpisom potvrđujem istovjetnost tiskane i elektroničke inačice završnog rada.

U Bjelovaru, 9.10.2023.

potpis studenta/ice