Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/29BE01Atanackovic

GPT ARHITEKTURA I PRIMENA U SOFTVERSKOJ INDUSTRIJI

GPT ARCHITECTURE AND APPLICATION IN THE SOFTWARE INDUSTRY

Ana Atanacković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *GPT modeli su modeli za predviđanje jezika opšte namene. To su kompjuterski programi koji mogu da analiziraju, izdvajaju, sumiraju i na drugi način koriste informacije za generisanje sadržaja. Oni kreiraju tekst sličan čoveku, a da nisu eksplicitno programirani da to urade. Kao rezultat, mogu se podesiti za niz zadataka obrade prirodnog jezika, uključujući odgovaranje na pitanja, prevod jezika i rezimiranje teksta.*

Ključne reči: *GPT, Transformer, veštačka inteligencija*

Abstract – *GPT models are general purpose language prediction models. These are computer programs that can analyze, extract, summarize and otherwise use information to generate content. They create human-like text without being explicitly programmed to do so. As a result, they can be tuned for a range of natural language processing tasks, including answering questions, translating languages and summarizing text..*

Keywords: *GPT, Transformer, AI*

1. UVOD

GPT jeste skraćenica za *Generative Pre-trained Transformers*. Generativno označava tehnologiju sposobnu da proizvodi sadržaj, kao što su tekst i slike. Prethodno obučeno označava sačuvane mreže koje su već naučene da reše problem ili da ostvare određeni zadatak koristeći veliki skup podataka. Transformer označava bazu GPT modela. Modela za predviđanje jezika zasnovanog na neuronskim mrežama koji je izgrađen na arhitekturi Transformera [1]. Ovakvi modeli analiziraju upite prirodnog jezika i predviđaju najbolji mogući odgovor na osnovu njihovog razumevanja jezika. Oni su prethodno obučeni za ogromne količine podataka, kao što su knjige i veb stranice, da bi se generisao kontekstualno relevantan i semantički koherentan jezik. To su računarski programi koji mogu da kreiraju tekst sličan čoveku, a da nisu eksplicitno programirani da to urade. Kao rezultat, mogu se podesiti za niz zadataka obrade prirodnog jezika, uključujući odgovaranje na pitanja, prevod jezika i rezimiranje teksta.

U okviru ovog teksta provešćemo detaljnu istragu o tome kako izgleda arhitektura GPT-a i kako funkcioniše tok obrade nekog unosa, kako bi se dobio valjani ishod.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Srđan Vukmirović, red. prof.

Zatim ćemo se fokusirati na softverski aspekt GPT-a, gde ćemo pokazati konkretne primere korišćenja.

2. MODEL GPT-A

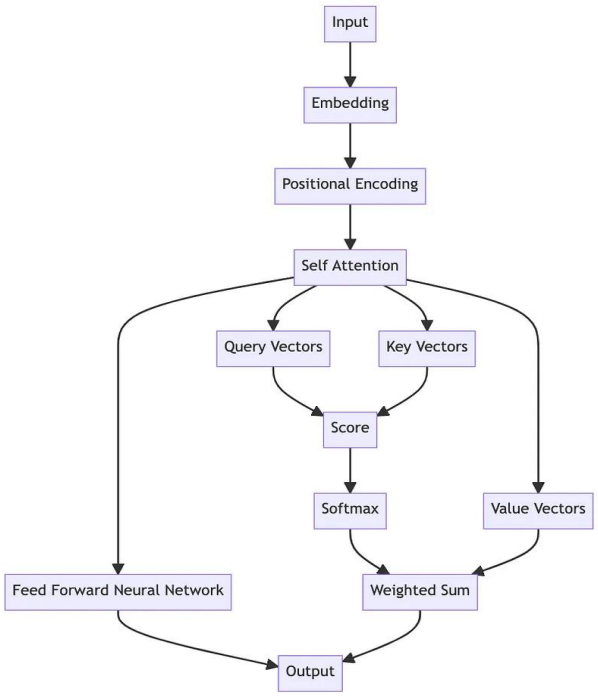
GPT arhitektura prestavlja značajan korak napred u obradi prirodnog jezika, omogućavajući mašinama da generišu tekst sličan čoveku i obavljaju bezbroj zadataka vezanih za jezik. Temelj ovog modela bazira se na arhitekturi Transformera, koja je objavljena još 2017-te godine [2]. Sledeća poglavlja otkriće ključne komponente modela, kao i njegove osnovne mehanizme.

Ovaj model prolazi kroz dva koraka pre njegovog korišćenja, obuku i usavršavanje. Tokom obuke, model je izložen ogromnoj količini raznovrsnih tekstualnih podataka, koje mu omogućavaju da shvati kompleksnost jezika i njegovih sastavnih veza. Ključnost ove faze leži u učenju bez nadzora, tako da model sam razvije široko razumevanje jezičkih termina i njihovih mogućih značenja. Za namene usavršavanja, potrebno je prvo izabrati odgovarajuće zadatke, koji si planirani da se izvrše pomoću ovog modela. Onda kada imamo krajnji cilj, model će se prilagoditi specifičnom zadatku za koji je adaptiran. Ova prilagodljivost čini GPT svestranim alatom za različite aplikacije za obradu prirodnog jezika.

2.1. Transformer

Transformatori su postali baza mnogih najsavremenijih modela obrade prirodnog jezika zbog svoje sposobnosti da efikasno pronađu zavisnosti i veze između podataka. Model Transformera sastoji se od enkodera i dekodera, od kojih je svaki sastavljen od niza identičnih slojeva, koji se sastoje od dva podsloja (videti izgled na slici 1.):

1. Višeglavi samosvesni mehanizam – omogućava modelu da odmeri i kodira različite reči u ulaznoj sekvenci na osnovu njihove relevantnosti za trenutnu reč koja se obrađuje, i koristeći više glava detektuje različite tipove odnosa između reči. Time se podržava paralelna obrada (model se istovremeno fokusira na različite delove ulazne sekvence) i poboljšano prikazivanje učenja. Ovaj mehanizam pomaže da se nauči o kontekstualnom odnosu između reči.
2. Poziciono povezana mreža za napredovanje – standardna neuronska mreža koja se primenjuje nezavisno za svaku poziciju, kako bi se promenio izlaz samosvesnog sloja. Ova mreža pomaže da se nauči o samim rečima.



Slika 1. *Model transformera*

Struktura po slojevima omogućava modelu da uhvati hijerarhijske karakteristike i složene paterne u podacima. Ovim se olakšava i skalabilnost, davajući prostora za proširenje modela u neke kompleksnije modele.

2.2. Samosvestan mehanizam

Sam temelj arhitekture Transformera jeste mehanizam pažnje/svesnosti, koji hvata zavisnosti i odnose između bilo koje dve tačke u nizu, bez obzira na njihovu udaljenost. Sa ovim, moguće je iskoristiti i znanje iz prošlosti, kako bi se uticalo na buduća predviđanja. Ovaj mehanizam izračunava i težinu koja predstavlja važnost ili relevantnost tačke u nizu. Tako se model fokusira na najrelevantnije delove ulaznih podataka za predviđanje.

Razumejući ovo, vidimo da on hvata zavisnosti i odnose unutar ulaznih sekvenci, i omogućava modelu da identifikuje i odmeri važnost različitih delova ulazne sekvence bivajući samosvestan.

Višeglavi samosvestan mehanizam podržava paralelizam. Ovo ga čini mnogo bržim i efikasnijim kada se radi o velikim količinama ulaznih parametara. Ipak, on ima i kvadratnu računsku složenost u odnosu na dužinu sekvence, što može učiniti mehanizam neefikasnim za veoma duge sekvence.

Samosvesni slojevi mogu pružiti oblik interpretabilnosti, pošto težine pažnje mogu dati uvid u to koje parametre model smatra važnim za svako predviđanje. Oni se koriste u obradi prirodnog jezika, za akcije prevođenja, sumiranja i analize osećanja.

2.3. Poziciono kodiranje

Tehnika koja se koristi da se modelu daju informacije o relevantnosti položaja reči u rečenicama - poziciono kodiranje. Transformatori su zasnovani na samosvesnim mehanizmima, koji sami po sebi ne uzimaju u obzir redosled reči u rečenicama, te se poziciono kodiranje dodaje ulaznim ugrađenim elementima (rečima), pre nego što se proslede u samosvesni sloj. Ovim model može da uhvati sekvencijalnu prirodu podataka, koja je ključna za zadatke koji zahtevaju razumevanje reda reči i konteksta.

2.4. Računanje težine

Već prethodno napomenuta, težina ili značaj ulaznog parametra, može se matematički sračunati. Kada gledamo ceo ulazni niz, važnost svake reči za trenutnu reč određuje se na osnovu njihovog semantičkog i sintaksnog odnosa. Ova mera se računa u koraku obuke modela, i može da se menja u zavisnosti od konteksta u kome se reč koristi. Drugi naziv za nju jeste i ocena pažnje.

Ako uzmemo za primer rečenicu „Rad je završen u toku prethodnog dana.“, kada se kodira reč „rad“, samosvestan mehanizam može da oceni reč „završen“ sa visokom ocenom, misleći da je za razumevanje konteksta reči „rad“ ova reč bitna.

Unos u sloj se transformiše u vektore upita, ključa i vrednosti kroz množenje sa tri različite matrice težine. Vektori upita i ključa se koriste za izračunavanje težine, a vektori vrednosti za kreiranje kombinacije težine na osnovu ovih rezultata. Kombinacije težine su ujedno i izlazi ovog sloja.

2.5. Ugrađivanje reči

Postoji više načina na koje se reč može numerički predstaviti, jedan od njih je ugrađivanjem reči. Ovaj pristup koristi Transformer. Svaka reč u rečniku se preslikava u vektor realnih brojeva, nasumično izabranih, koji kreira njenu ugradnju reči. Tokom treninga, ugradnja se menja u odnosu na kontekst u kom je reč korišćena. Slične ugradnje reči bi na primer bile „diplomski rad“ i „master rad“, jer su pronađene da se često koriste u rečenicama o radovima. Ovakav vid obrade zove se kontekstualno učenje.

Model pravi predviđanja na osnovu trenutnih ugradnji, a zatim se izračunava greška između ovih predviđanja i stvarnih vrednosti. Ova greška se zatim koristi za prilagođavanje ugrađivanja kako bi se predviđanja približila stvarnim vrednostima. Ovim se ostvaruje proces nazadnog širenja znanja modela. Ovakav model je fleksibilan i jednostavan, kako za ovakvu akciju nije potrebno neko predznanje. On uključuje rad unazad od izlaza do ulaza, kako bi se otkrilo kako smanjiti broj grešaka i učiniti model pouzdanijim. Ovakvi algoritmi primenljivi su za mnoge scenarije, što ih čini idealnim metodom za poboljšanje performansi neuronskih mreža.

2.6. Višeglava svesnost

Višeglava svesnost je ključna komponenta arhitekture modela Transformatora, koja se koristi da omogući modelu da se fokusira na različite vrste informacija. Mehanizam svesnosti se primenjuje više puta paralelno (različite glave su aktivne), i time se dobijaju drugačiji tipovi veza između podataka.

Svaka glava pažnje ima sopstvene linearne transformacije koje primenjuje na ugrađen unos kako bi dobila svoje upite, ključeve i vrednosti. Ovim glave bivaju nezavisne i

mogu učiti odvojene stvari, te se jedna grana može fokusirati na sintaksne veze a druga na semantičke. Svi rezultati (izlazi glava), spajaju se i linearno se transformišu za konačan rezultat.

Svaka glava će biti nezavisna, ne namerno već na osnovu nasumične (stohastičke) prirode procesa obuke, kao i činjenice da svaka glava počinje se drugačijim slučajno izabranim parametrima. Sa ovim, potencijalno moguće je da se ostvari nejedinstvenost grana, ali šanse za to su male.

3. PRIMENE U SOFTVERU

GPT pomaže programerima da poboljšaju svoju produktivnost, kreativnost i kvalitet svog rada. On može automatizovati zadatke kao što su pisanje koda, otklanjanje greški, testirati i dokumentovati, edukovati i odgovarati na pitanja [3]. Isto tako, on nudi sugestije i ocenjuje već postojeće rezultate [4].

3.1. Generisanje koda

Sposobnost generisanja koda na osnovu opisa prirodnog jezika jedna je od boljih funkcija GPT-a. Programeri mogu da obezbede model sa opisom željene funkcionalnosti, a on će generisati odgovarajući kod u ciljnom programskom jeziku. Programeri mogu da opišu svoje zahteve ili da traže specifične šablone koda, a GPT može da generiše šablone koda ili da pruži primere implementacije kako bi pomogao u procesu razvoja. Ova sposobnost ubrzava proces razvoja, smanjuje ljudske greške i omogućava programerima da se fokusiraju na složenije zadatke. Sa pružanjem jasnih i sažetih opisa mogu se dobiti bolji rezultati upita. Ovo će pomoći modelu veštačke inteligencije da generiše tačan i efikasan kod uz minimiziranje potencijalnih nesporazuma.

3.2. Optimizacija koda

GPT može da koristi tehnologije obrade prirodnog jezika da brzo analizira i identifikuje oblasti koda koje treba poboljšati. Pruža se lako razumljiv korisnički interfejs za programere da identifikuju funkcionalne specifikacije, nefunkcionalne zahteve, slučajeve upotrebe, arhitekturu softvera, obrasce dizajna, implementaciju i testiranje. Dobijaju se preporuke za lako kreiranje i testiranje različitih verzija koda, dok se ne ispune željeni nivoi performansi. Na taj način se štedi vreme, novac i energija, kreiranjem optimizovanog koda u manjem broju iteracija nego bez pomoći modela.

3.3. Automatski pregledi koda i asistencija

Pregledi koda su suštinski deo procesa razvoja softvera, obezbeđujući da kod bude visokog kvaliteta i da se održava u skladu sa standardima kodiranja. Ovaj model se može koristiti za automatizaciju pregleda koda otkrivanjem potencijalnih problema, kao što su sintaksne greške, nesavršenosti u stilu, neefikasni algoritmi i potencijalni bezbednosni propusti. Ova automatizacija smanjuje vreme i trud koje programer mora da uloži na ručne preglede koda, omogućavajući mu da se fokusira na vrednije zadatke.

GPT može da analizira kod za potencijalne ranjivosti kao što su *SQL injection* i *XSS*. Takođe, može da otkrije potencijalne logičke greške i mrtav kod, smanjujući vreme potrebno za otklanjanje grešaka kod testa. Ove funkcije olakšavaju programerima da identifikuju i poprave probleme u kodu pre nego što uđu u produkciju.

3.4. Testiranje

GPT modeli, sa svojim sofisticiranim razumevanjem jezika i konteksta, sada su uključeni u različite scenarije testiranja. U obrazovanju, oni pomažu u kreiranju i ocenjivanju testova, nudeći personalizovane povratne informacije učenicima. U razvoju softvera, on pomaže u generisanju i izvršavanju test slučajeva, povećavajući efikasnost i pokrivenost. Sposobnosti modela da oponašaju ljudske reakcije takođe ih čini idealnim za testiranje scenarija korisničke službe i čet-bota.

3.5. Otklanjanje grešaka na bazi AI-a

Programeri mogu da unesu delove svog koda u GPT, a model može da analizira ove isečke da bi identifikovao greške ili predložio poboljšanja. Ovaj proces se ne odnosi samo na pronalaženje sintaksnih grešaka, on takođe može da predloži bolje prakse kodiranja ili efikasnije načine za obavljanje zadataka.

GPT može pomoći u identifikaciji uobičajenih i složenih grešaka u kodu. Što je još važnije, on objašnjava ove greške na način koji je razumljiv i početnicima i iskusnim programerima. Ovaj obrazovni aspekt pomaže programerima ne samo da poprave svoje trenutne probleme već i da uče iz svojih grešaka. Kada se problem identifikuje, GPT predlaže više rešenja ili zaobilazna rešenja. Ova mogućnost je posebno korisna u scenarijima u kojima može postojati više od jednog načina za rešavanje problema, omogućavajući programerima da izaberu onaj koji najbolje odgovara njihovom specifičnom kontekstu.

3.6. Dokumentovanje i deljenje znanja

GPT je u mogućnosti da pojednostavi proces dokumentovanje generisanjem tačne i precizne dokumentacije analizom generisanog koda. Tehnologija razume zahteve korisnika i kreira dokumentaciju sa minimalnim znanjem. Isto tako, sposoban je da napravi tutorijale i da odgovori na često pitana pitanja, čime programer uspeva da brže shvati nove koncepte i tehnologije.

3.7. Komunikacija

GPT, čiji zadatak jeste da bude inteligentni posrednik komunikacionog toka, može automatski rezimirati duže diskusije, odgovoriti na pitanja koja su kontekstualno povezana sa diskusijama i predložiti potencijalna rešenja za postojeće probleme u razgovoru.

Jezičke barijere mogu ometati efikasnu komunikaciju i saradnju. GPT ima potencijal da premosti ove praznine pružanjem mogućnosti prevođenja u realnom vremenu [5].

3.8. Personalizovana obuka

Analizom skupa veština svakog programera i praćenjem preferenci učenja GPT može da generiše personalizovane planove učenja koji se fokusiraju samo na relevantne teme i resurse. Uz ovakvo učenje, programeri uspevaju da nauče nove koncepte brže i efikasnije, povećavajući njihovu produktivnost i doprinos timu.

3.9. Kontrola verzije

Krucijalna komponenta za razvoj većih projekata, gde je uključen veći broj programera, jeste kontrola koda, tj. njegov izgled u svakom momentu. Za te potrebe vrši se kontrola verzije koda. GPT takođe nudi lako skladištenje, upravljanje i deljenja koda u okviru svojih projekata razvoja softvera, praćenje napretka pojedinaca i identifikovanje konflikata u kodu.

Moguće je i vratiti se na prethodne verzije koda, kao i vršiti automatske izgradnje, testiranja i puštanja koda. Kada se pristupi nekoj verziji koda, možemo dobiti uvid u detaljne informacije verzije, za potrebe analize i deljenja.

3.10. Kontuirana integracija

Kontinuirana integracija je proces koji spaja sve promene od programera u jednu glavnu granu u bazi koda. GPT olakšava programerima da identifikuju greške, prate napredak i testiraju promene pre stizanja koda do glavne grane, automatizacijom zadataka kao što su pokretanje testova i pravljenje slika.

3.11. Puštanje koda

GPT pomaže pri puštanju koda, te on upravlja paketima i integracijom, pruža razne prilagođene usluge razvoja softvera kao što su primena u *Cloud*-u i servisi podrške. To znači da kompanije mogu biti sigurne da se njihova prilagođena rešenja primenjuju ispravno i na vreme. Isto, on ima snažan sistem praćenja koji omogućava programerima da prate performanse svojih rešenja u proizvodnji.

3.12. Analiza podataka

GPT model može pomoći analitičarima da efikasno sakupe velike količine podataka. Jezički modeli pronalaze tražene podatke i izračunavaju i prikazuju rezultate u tabeli sa podacima. Neke aplikacije mogu i da iscrtaju rezultate na grafikonu ili da kreiraju sveobuhvatne izveštaje [6].

3.13. Organizacija projekta

Da bi razvoj softvera bio završen efikasno i na vreme, kao i da bi se upravljalo resursima i budžetima, potrebni su menadžeri projekta. GPT može da obezbedi moćno rešenje za upravljanje projektima za razvoj softvera. Automatizacijom radnji kao što su zakazivanje sastanaka, praćenje resursa i rokova, on omogućava menadžerima projekata da pojednostave procese, davajući im obaveštenja i upozorenja.

4. ZAKLJUČAK

GPT je prilično mlado otkriće koje se razvija ubrzanim tempom, ali nema sumnje, da će usled popularnosti i uspeha, ova oblast biti sve više izučavana i unapređivana. Očekujemo sve veće pogodnosti ovog modela, u vidu aplikacija i alata. Što se tiče softvera, već sada vidimo ogromnu količinu slučajeva korišćenja u realnom sistemu. Isto tako postoje i rizici na koje moramo ozbiljno posvetiti pažnju. Moramo obezbediti bezbednost naših podataka i shvatiti moguću nepouzdanost nekih alata GPT-a. Adekvatna protekcija i pametno korišćenje GPT-a, proizvešće odlične rezultate za bilo koju oblast.

5. LITERATURA

[1] https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[2] https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[3] https://arxiv.org/pdf/2311.07361.pdf

[4] https://arxiv.org/pdf/2302.14520.pdf

[5] https://arxiv.org/pdf/2304.01852.pdf

[6] https://aws.amazon.com/what-is/gpt/

Kratka biografija:

Ana Atanacković rođena je u Novom Sadu 1998. god. Studijski program Računarstvo i automatika, upisala je 2017. godine, a završila 2021. godine. Nakon toga upisuje master akademske studije Primenjene računarske nauke i informatika, iz oblasti Elektrotehnike i računarstva.