Large language model in game development

Uvod

Industrija video-igara beleži izuzetan rast tokom poslednjih nekoliko decenija, koji je u velikoj meri podstaknut tehnološkim napretkom i sve većom dostupnošću igara široj publici. Sa porastom upotrebe pametnih telefona i drugih mobilnih uređaja, igranje igara postalo je praktičnije i pristupačnije nego ikada ranije, na šta je industrija odgovorila razvojem raznovrsnijih i sadržajnijih naslova koji zadovoljavaju širi spektar interesovanja, preferencija i tipova igrača.

U mnogim aspektima, savremene video-igre postale su interaktivni filmovi i snažan instrument za pripovedanje, u kojima narativ, dijalog i interakcija sa igračem imaju ključnu ulogu. Tradicijalno, dijalozima NPC-jeva upravlja grananje dijaloga, to je sistem u kome programeri unapred definišu odgovore na pitanja, koji najčešće zahtevaju veliki trud za izradu. Veliki jezički modeli mogu rešiti taj problem jer omogućavaju stvaranje bogatijih i fleksibilnijih narativnih struktura, doprinoseći dinamičnijem, jedinstvenijem i imerzivnijem igračkom iskustvu. Veliki jezički modeli mogu se primeniti za generisanje dodatnih sadržaja unutar igre, poput dijaloga sporednih neigračkih likova (NPC-jeva).

Potencijalne prednosti ovakvog pristupa su višestruke. Ne samo da bi NPC-jevi pokretani velikim jezičkim modelima mogli da nude jedinstvene i neponovljive dijaloge, već bi igračima omogućili i da sa njima vode razgovore koji u velikoj meri podsećaju na prirodnu ljudsku komunikaciju. Ovakav pristup mogao bi značajno da unapredi i ponovno igranje igre, kao i iskustvo nakon završetka glavne priče.

Large Language Model

Large Language Model (LLM) predstavlja naprednu veštačku inteligenciju sposobnu za razumevanje i generisanje prirodnog jezika. U kontekstu video igara, LLM se može koristiti za omogućavanje dinamičkog dijaloga sa NPC-jevima, njihovo raznovrsno ponašanje koje zavisi od prethodnih interakcija igrača u svetu, umesto da se oslanja na fiksne skripte. Međutim, uprkos širokim perspektivama, primena LLM-a u igricama je još uvek u začetku i suočena je sa tri ključna tehnička izazova:

- Prvi je adaptacija modela - postoji nesklad između opšteg semantičkog znanja LLM-a i sveta i priče specifične za igricu, što lako može dovesti do nepovezanih dijaloga koji narušavaju osećaj igre.
- Drugi je optimizacija za realno vreme - kašnjenje u zaključivanju LLM-a često je jako dugo, dok igre zahtevaju da kašnjenje odgovora NPC-a bude minimalno.
- Treći je doslednost karaktera – neki LLM-ovi mogu generisati sadržaj koji prevazilazi znanje tog NPC-a. Dodatno, memorijska ograničenost tih modela može otežati praćenje konteksta u dužim interakcijama, što vodi ka generisanju manje relevantnih odgovora.

Neka od rešenja za prevazilaženje ovih ključnih izazova su:

- Fine-tuning - Dodatno treniranje modela na skupu podataka specifičnim za svet igre. Ovaj skup podataka može sadržati lore igre, karakteristike likova, istoriju sveta, dijaloge i druge relevantne informacije. Kroz fine-tuning, model uči da prioritizuje znanje o svetu igre nad svojim opštim znanjem, što rezultuje konzistentnijim i relevantnijim odgovorima.
- Retrieval-Augmented Generation (RAG) - Tehnika koja omogućava modelu da pristupa eksternoj bazi znanja tokom generisanja odgovora. Pre nego što model generiše odgovor, sistem pretražuje relevantne informacije iz baze podataka igre (wiki sveta igre, karakterizacije NPC-jeva, trenutno stanje quest-ova) i uključuje ih u prompt. Na ovaj način, model "konsultuje" relevantne činjenice pre odgovora, značajno smanjujući verovatnoću generisanja informacija koje nisu u skladu sa svetom igre.
- Prompt sa jasnim ograničenjima - Pažljivo dizajniran system prompt koji eksplicitno definiše šta NPC zna, šta ne zna, i kako treba da se ponaša kada mu se postavi pitanje van njegove oblasti znanja. Na primer, NPC trgovac u srednjovekovnom fantasy svetu ne bi trebalo da zna ništa o savremenim tehnologijama, pa prompt može sadržati instrukciju: "Ako igrač pita o nečemu van tvog znanja ili sveta igre, ljubazno odgovori da o tome nemaš saznanja ili da ne razumeš o čemu se radi."
- Strukturirani system prompt - Detaljno definisanje NPC-a kroz višeslojni prompt koji uključuje: osnovne informacije (ime, uzrast, zanimanje), ličnost i temperament, znanje i veštine, motivacije i ciljeve, stil govora i dijalekt, odnos prema igraču (zasnovan na prethodnim interakcijama), trenutno emocionalno stanje i relevantne quest informacije. Ovaj prompt osigurava da model konzistentno tumači karakter kroz sve interakcije.
- Upravljanje memorijom razgovora - Implementacija sistema koji održava istoriju interakcija između igrača i NPC-a. Za kraće razgovore, cela istorija može biti

uključena u svaki prompt. Za duže sesije, koriste se tehnike koje sumiraju istoriju komunikacije gde se raniji delovi razgovora kompresuju u sažetke, zadržavajući ključne informacije, dok se skorašnji deo razgovora čuva u potpunosti. Ovo omogućava NPC-u da "pamti" šta je rečeno ranije i održava koherentnost tokom vremena.

- Keširanje i memorija dijaloga - Čuvanje prethodno generisanih odgovora za identična ili vrlo slična pitanja. Ako igrač ponovo pita isto pitanje ili ako drugi igrač postavi istu vrstu pitanja NPC-u, sistem može vratiti keširani odgovor instantno umesto ponovnog generisanja.

Tipovi modela

Implementacija LLM-a u video igrama može koristiti tri osnovna pristupa: isključivo lokalne modele, isključivo remote (cloud) modele, ili hibridnu arhitekturu koja kombinuje oba. Svaki pristup ima specifične prednosti i nedostatke, a izbor zavisi od prioriteta igre, ciljne platforme i raspoloživih resursa.

Lokalni modeli

Lokalni modeli se instaliraju direktno na uređaju igrača ili game serveru i izvršavaju se bez potrebe za internet konekcijom. Latencija odgovora zavisi od hardverskih mogućnosti uređaja.

Ovaj pristup donosi nekoliko značajnih prednosti.

- Potpuna kontrola nad podacima i privatnošću korisnika je obezbeđena jer nijedan deo komunikacije ne napušta uređaj.
- Performanse su predvidljive i ne zavise od kvaliteta ili dostupnosti internet konekcije.

Međutim, postoje i mane:

- Instalacija igre se povećava.
- Sistem zahteva značajne hardverske resurse, posebno GPU ili CPU procesorsku moć i RAM memoriju.
- Kvalitet odgovora je ograničen u poređenju sa mnogo većim cloud modelima.

Remote modeli

Remote modeli su hostovani u cloud-u i pristupa im se preko API poziva. Latencija odgovora zavisi od izabranog modela, trenutnog opterećenja servera i kvaliteta internet konekcije.

Prednosti ovog pristupa su višestruke:

- Kvalitet i koherencija odgovora su na većem nivou zahvaljujući ogromnim modelima treniranim na velikim datasetovima.
- Sistem ne opterećuje igrač jer se sva obrada odvija u cloud-u, što znači da nema specijalnih hardverskih zahteva.
- Skalabilnost sistema omogućava podršku za milione korisnika simultano bez degradacije performansi.

Nedostaci sistema su takođe značajni:

- Potpuna zavisnost od internet konekcije.
- Potencijalni privacy concerns postoje jer se svi dijalozi šalju trećoj strani, što može biti problematično za korisnike osetljive na privatnost.

Ovaj tip modela je korisniji za multiplayer igre u odnosu na lokalni, jer one već zahtevaju internet konekciju.

Hibridni model

Hibridni pristup kombinuje lokalne i remote modele, birajući koji model koristiti na osnovu konteksta, dostupnosti resursa i prioriteta. Ovaj pristup je najfleksibilniji i omogućava balansiranje između kvaliteta, brzine i troškova.

U ovom slučaju, sistem bi trebao da sadrži centralnu komponentu koja analizira svaki zahtev i donosi odluku o tome da li bi koristili lokalni ili remote model. Odluke o tome koji se model koristi može biti određen na različite načine:

- Važnost NPC-a igra ulogu pri čemu se main story karakteri obrađuju remote modelom, a sporedni NPC-jevi poput vendora, gardova i villagera lokalnim modelom.
- Na osnovu toga da li je to neka real-time situacija poput combat-a ili brzih interakcija, i takve sitruacije bi koristile remote model, dok se nebrzinske situacije poput exploration-a ili story moments obrađuju local modelom.
- Dostupnost konekcije utiče tako da se pri stabilnoj internet konekciji preferira remote model, dok se pri offline ili nestabilnoj konekciji koristi lokalni model.
- Broj reči ili slova - na primer ako imamo manje od 10 charactera, koristimo lokalni model.
- Tip interakcije - trgovinske interakcije će na primer koristiti lokalni model, dok neki bitni NPC-jevi (glavni negativci, saputnici i slično) mogu koristiti remote model.
- Može se u krajnjem slučaju koristiti i Natural Language Processing (NLP). NLP klasifikator je model mašinskog učenja treniran da analizira tekstualne upite i automatski ih kategorizuje u predefinisane klase. U kontekstu odlučivanja između lokalnog i remote modela, NLP klasifikator služi za automatsku procenu

kompleksnosti igrač upita, što eliminiše potrebu za ručnim definisanjem pravila za svaku moguću vrstu pitanja.

Integracija LLM-a u NPC dijalogu

Integracija LLM-a u multiplayer igru zahteva pažljivo dizajniranu arhitekturu koja balansira između performansi, skalabilnosti i fleksibilnosti u generisanju dijaloga.

Komponente sistema

Klijentska strana (Unity Client) - Unity klijent služi kao interfejs između igrača i backend sistema. Kada igrač inicira interakciju sa NPC-em, Unity:

- Detektuje trigger event (klik na NPC-a, quest interakcija)
- Prikuplja kontekstualni podatke (ID igrača, ID NPC-a, lokacija, quest status...) i proverava da li je poruka prazna i dužinu same poruke
- Formira zahtev i šalje ga ka Rust serveru preko WebSocket ili HTTP protokola
- Prima odgovor i renderuje ga u game UI-u (voice synthesis, subtitle sistem)

Rust Game Server - Rust server predstavlja centralni deo koji prihvata sve zahteve i upravlja logikom igre:

- Request Handler - Prima zahteve od Unity klijenata i validira ih
- Context Builder - Prikuplja sve relevantne podatke o igraču i NPC-u iz baze podataka
- Router Logic - Odlučuje da li će koristiti lokalni model, remote model ili keširani odgovor
- Response Manager - Obrađuje generisane odgovore, validira ih, i šalje nazad klijentu

RAG (Retrieval-Augmented Generation) Modul - RAG sistem omogućava dinamičko obogaćivanje promptova relevantnim informacijama. Funkcioniše na sledeći način:

- Korisnički unos - Proces započinje kada korisnik postavi upit ili unese zahtev u RAG model. Upit može biti pitanje, prompt ili zahtev za informacijama na koji model treba da odgovori.
- Encoding upita - Uneseni upit se transformiše u numeričku reprezentaciju pomoću jezičkog modela kao što su BERT ili GPT.
- Retrieval system - Sistem zatim pronalazi relevantne informacije iz spoljne baze znanja, kao što su skup dokumenata i baza podataka. Pretraga može koristiti tradicionalne metode zasnovane na ključnim rečima ili savremenije metode zasnovane na vektorima. Sistem pretražuje izvor i izdvaja najrelevantnije informacije.

- Rangiranje i filtriranje - Pronađene informacije se rangiraju prema relevantnosti i uzimaju se prvih N dokumenata koje smo mi odabrali.
- Generisanje kontekstualnih embeddinga - Svaki preuzeti dokument ili deo teksta se takođe pretvara u numerički embedding. Na taj način generativni model može efikasno da uključi pronađene informacije prilikom formiranja odgovora.
- Ugrađivanje konteksta (embedding context) - Preuzeti dokumenti ili njihovi relevantni delovi predstavljaju se kao vektori, što omogućava RAG modelu da razume i obradi njihov sadržaj u istom semantičkom prostoru kao i korisnički upit.
- Spajanje preuzetih informacija - Preuzeti dokumenti se spajaju sa korisničkim upitom i zajedno se prosleđuju modelu kako bi se generisao odgovor.
- Generisanje odgovora - Model koristi i korisnički upit i preuzeto znanje kako bi generisao prirodan i informativan odgovor. RAG modeli kombinuju originalni kontekst pitanja sa eksternim informacijama kako bi poboljšali tačnost i relevantnost odgovora.
- Isporuka odgovora - Na kraju, generisani i eventualno obrađeni odgovor se šalje korisniku.

LLM Integration Layer - Ovaj sloj upravlja direktnom komunikacijom sa jezičkim modelima:

- Local Model Runtime - Ako se koristi lokalni model, ovaj modul upravlja inference-om.
- API Client - Za remote modele, šalje HTTP zahteve ka cloud API-jima
- Response Parser - Parsira odgovor od modela i izvlači čist tekst dijaloga

Memory & Cache System - Sistem za upravljanje memorijom i keširanje:

- Conversation History Database - Redis ili PostgreSQL koji čuva istoriju razgovora
- Response Cache - Brzi cache (Redis) za često postavljana pitanja
- Summary Engine - Periodično kompresuje duge razgovore u sažetke kako bi se sačuvao kontekst ali smanjio broj tokena

Promt

Prompt predstavlja strukturisan tekstualni ulaz (instrukciju, upit ili specifikaciju) koji se prosleđuje LLM-u radi generisanja sadržaja, narativa ili interaktivnih elemenata za NPC-jeve.

Generisanje Promptova

Generisanje promptova predstavlja kritičan aspekt implementacije LLM-a u video igrama, jer kvalitet i struktura prompta direktno određuju konzistentnost i relevantnost NPC dijaloga.

Prompt se sastoji iz dva glavna dela: system prompt (definiše NPC-a) i user prompt (pitanje igrača).

Struktura system prompta

System prompt služi kao temelj karaktera NPC-a i mora biti detaljno definisan kako bi model konzistentno tumačio ulogu kroz sve interakcije. Ovaj deo prompta ostaje nepromenjen tokom razgovora i sadrži sledeće elemente:

- Kontekst iz RAG sistema - Pre nego što model generiše odgovor, RAG modul pretražuje vector bazu i vraća relevantna dokumenta. On može da nam vrati sledeće bitne informacije:
    - Osnovne informacije - Ime, uzrast, zanimanje, lokacija i fizički opis NPC-a. Ove informacije pomažu modelu da razume osnovni identitet karaktera.
    - Ličnost i temperament - Detaljno opisuje karakter NPC-a, njegove vrednosti, stavove i način na koji reaguje na različite situacije.
    - Znanje i veštine - Eksplicitno definiše šta NPC zna i šta ne zna. Ovo je kritično za sprečavanje netačnih izjava. Na primer, srednjovekovni kovač zna o oružju, metalurgiji i lokalnim pričama, ali ne zna ništa o modernoj tehnologiji ili događajima van svog regiona. Uključuje se i instrukcija šta NPC treba da kaže kada mu se postavi pitanje van njegove oblasti znanja - na primer, "ljubazno odgovori da o tome nemaš saznanja".
    - Motivacije i ciljevi - Šta NPC želi da postigne i zašto. Trgovac želi profit, ali takođe ceni reputaciju i dugoročne odnose sa klijentima.
    - Stil govora i dijalekt - Određuje kako NPC govori - formalno ili neformalno, koristi li lokalne izraze itd.
    - Odnos prema igraču - Dinamički deo koji se menja zavisno od prethodnih interakcija. NPC može biti neutralan prema novom igraču, prijateljski raspoložen prema onome ko mu je pomogao u quest-u, ili neprijateljski prema onome ko ga je prevario. Ovaj deo prompta se ažurira na osnovu reputation sistema igre.
- Player-specific kontekst - Informacije o trenutnom stanju igrača koje utiču na interakciju: nivo igrača, aktivni quest-ovi, items u inventory-ju, lokacija u svetu igre. Na primer, ako igrač ima quest da nađe retki mineral, NPC kovač može spomenuti da je čuo priče o rudniku gde se taj mineral nalazi.
- Ograničenja odgovora - Definiše pravila formatiranja i strukture odgovora:
    - Dužina odgovora: Maksimalni broj rečenica zavisno od tipa interakcije
    - Format: JSON objekat na primer

    - Instrukcije šta NPC nikada ne sme da radi ili kaže, zavisno od age rating-a igre, karaktera NPC-a, onoga što programer zapravo želi.
- Prethodna konverzacija sa NPC-jem

Struktura user prompta

User prompt se generiše za svaki pojedinačni zahtev i sadrži kontekst specifičan za trenutnu situaciju:

- Trenutna poruka igrača tačnije sve što igrač prosledi - Tekst koji je igrač uneo, prethodno očišćen od malicioznih pattern-a.

Skaliranje

Skaliranje ovakvog sistema kako bi se podržao veliki broj korisnika zahteva specifičan pristup. Klientska strana šalje zahteve preko WebSocket ili HTTP-a ka Rust game serveru, koji deli opterećenje na više instanci. Za svaki zahtev, router logika određuje da li će se koristiti keširani odgovor, lokalni ili remote model (ako koristimo hibridni pristup). Keširanje često postavljanih pitanja značajno smanjuje latenciju i opterećenje modela.

Za obradu zahteva koji zahtevaju LLM, hibridni pristup omogućava dinamičko usmeravanje. Manje zahtevne interakcije (trgovina, jednostavni dijalozi) obrađuju se lokalnim modelima na edge serverima blizu korisnika. Kompleksniji zahtevi (glavni narativi, dinamički questovi) prosleđuju se skalabilnim cloud LLM servisima, koji automatski skaliraju resurse prema potražnji.

Load balancer predstavlja ulaznu tačku koja prima sve zahteve od Unity klijenata i distribuira ih ka pool-u backend servera. Može koristiti različite strategije:

- Round-robin - Zahtevi se distribuiraju ciklično svim dostupnim serverima
- Least connections - Novi zahtev se šalje serveru sa najmanje aktivnih konekcija
- Response time based - Preferira se server sa najbržim istorijskim vremenima odgovora
- Geographic routing - Zahtevi se usmeravaju ka najbližem serveru radi smanjenja latencije

Sistem implementira višeslojni rate limiting kako bi sprečio preopterećenje. Svaki igrač ima definisan maksimalan broj zahteva koji može iskoristiti u nekom periodu. Priority queue sistem kategorizuje zahteve prema važnosti - main story NPC-jevi i quest-critical dijalozi

dobijaju visok prioritet i procesuiraju se prvi, dok generic NPC-jevi (seljani, gardovi) čekaju u redu sa nižim prioritetom.

Security

Implementacija LLM sistema u multiplayer igrama otvara nove vektore napada i zloupotrebe koje tradicionalni game sistemi nisu morali da razmatraju. Bezbednost mora biti implementirana na više nivoa kako bi se zaštitili i sistem i igrači.

Prompt Injection napadi

Prompt injection predstavlja jednu od najozbiljnijih pretnji sistemima koji koriste AI. Igrači mogu pokušati da "hakuju" NPC-jeve ubacivanjem malicioznih instrukcija u svoj input koji prevaziđe originalne system prompte. Strategije koje se koriste kako bi prevazišli ovaj problem su:

- Proveravanje inputa - Prvi nivo odbrane je validacija korisničkog input-a pre slanja ka LLM-u. Sistem proverava dužinu poruke, detektuje poznate injection pattern-e (kao što su "ignore previous", "you are now", "system override"), i uklanja specijalne karaktere koji mogu zbuniti parser.
- Menjanje promptova - System prompt mora biti dizajniran da aktivno odbija injection pokušaje. Kritična je eksplicitna instrukcija da NPC nikada ne sme napustiti svoju ulogu ili promeniti ponašanje na osnovu zahteva igrača. Efikasan pristup uključuje meta-instrukcije koje definišu kako NPC treba da reaguje kada detektuje pokušaj manipulacije.

Neprimeran sadržaj

LLM-ovi mogu generisati neprimeran sadržaj uprkos built-in safety filters. U multiplayer okruženju, ovo je posebno problematično jer jedan igrač može namerno uzrokovati da NPC kaže nešto loše, što zatim vidi cela zajednica. Nivoi zaštitte su:

- Praćenje inputa - Korisnički input se analizira pre slanja LLM-u radi detekcije govora mržnje, eksplicitnog sadržaja i namerne provokacije modela.
- Praćenje outputa - Svaki odgovor NPC-a se proverava pre prikazivanja igračima. Ukoliko se detektuje neprimeran sadržaj, odgovor se blokira ili zamenjuje neutralnom, kontekstualno bezbednom replikom.
- Ograničenja u system promptu - System prompt mora jasno definisati jezička i etička ograničenja NPC-a, uključujući zabranu psovki, diskriminatornog govora i eksplicitnih tema, kao i instrukcije za defanzivno ponašanje u slučaju provokativnih upita.

Preuzimanje naloga

Napadač može pokušati da forsira NPC-a da se pretvara da je admin i da recimo traži od ostalih igrača da “verifikuju” svoj nalog davanjem svojih kredencijala. Da bi se ovaj rizik mitigovao, NPC-evima mora biti strogo zabranjeno da se predstavljaju kao admin ili da ikada traže lične ili autentifikacione podatke.

Lažne informacije u igri

Napadač može da da NPC-ju lažne informacije kako bi ostali igrači izgubili vreme ili resurse na nešto što im nije korisno. Kako bi se ovaj problem ublažio, NPC-evi ne smeju prihvatati informacije dobijene od igrača kao tačne. Svi odgovori moraju biti zasnovani isključivo na proverenim izvorima, kao što su game state, server-side logika i RAG baza sa podacima. Informacije koje potiču od igrača treba tretirati kao neproverene i jasno označene kao glasine ili mišljenja unutar narativa igre.

Rate limiting

LLM endpointi su posebno osetljivi na spam i automatske napade. U multiplayer igrama napadač može koristiti botove za masovno slanje promptova NPC-jevima kako bi opteretio sistem. Potencijalno rešenje je implementirati mehanizme ograničavanja broja zahteva po jedinici vremena (rate limiting) po korisniku.

Reference

1. https://medium.com/@sean_skimmer/npc-gpt-3f4cb5272773
2. https://jurnal.itscience.org/index.php/brilliance/article/view/6779/4964
3. https://arxiv.org/pdf/2404.19721
4. https://arxiv.org/html/2504.13928v1
5. https://lutpub.lut.fi/bitstream/handle/10024/167809/bachelorthesis_Huang_Junyang.pdf
6. https://www.weka.io/learn/guide/ai-ml/retrieval-augmented-generation/
7.