Testiranje učinkovitosti i točnosti chatbota

Za Raiffesein Bank testiranje, pozicija Junior developer za chatbot tim

Filip Matić

Testiranje se provodi u svrhu provjeravanja točnosti prepoznavanja namjere korisnika (intent) na osnovu njegovog upita, odnosno unosa korisnika. Testiranjem su provjerene osnovne metrike poput točnosti (omjer točnih odgovora i ukupnog broja pitanja) i povjerenja modela (confidence).

Chatbot za testiranje ima 10 namjera, ili intenta, koje bi trebao prepoznati iz korisničkog upita.
To su:

- radno_vrijeme
- ulaznice
- adresa
- danas_izložbe
- kafic
- toaleti
- pristupacnost
- parking
- clanstvo
- kontakt

Za svaki intent su predloženi primjeri za koje bi chatbot trebao sigurno pogoditi namjeru, odnosno odgovor. Za svaki intent postoji samo jedan točan odgovor.

Testiranje je provedeno lokalno, odnosno u Python virtualnom okruženju napravljenom na temelju priložene .env.example datoteke.

Za testiranje je pripremljeno 92 test case-a. Test caseovi su napisani tako da testiraju sposobnost prepoznati točan intent u uvjetima kada upiti nisu optimalno napisani, odnosno da simuliraju svakodnevnu situaciju gdje korisnik ne zna kako treba optimalno formulirati pitanje za unos.

Napisani test caseovi su priloženi uz izvještaj, a njihova namjera je bila testirati različite vrste upita:

- Točni upiti
- Upiti s neformalnim govorom
- Upiti s greškama u tipkanju
- Upiti s gramatičkim/pravopisnim greškama
- Upiti bez specijalnih znakova č,ć,š,ž,đ

Za svaki test case je poslan upit API-u, te su iz njegovog odgovora izvučeni podaci potrebni za testiranje točnosti chatbota.

Test caseovi su bili formulirani u obliku:

test_ID	unos	ocekivani_intent
T1	„Koje radno vrijeme?“	radno_vrijeme
T2	„Kad radite?“	radno_vrijeme

.
.
.

Osim sposobnosti bota da prepozna intent, napisani upiti su zamišljeni tako da ispitaju sljedeći sposobnosti:

Jezična varijabilnost
Kontekstualno razumijevanje
Fallback odgovore
Ponašanje u slučaju negativnih testova

Za testiranje su ponuđene dvije opcije, Selenium i REST. Selenium je opcija koja eventualno može pokriti više sadržaja i brže to uraditi, ali zbog koderske prirode pozicije za koju se prijavljujem, odabrao sam više programerski pristup.

Kod je pisan u lokalnom Python virtualnom okruženju, koristeći REST API. Kod je također priložen uz izvještaj, a osnovne funkcije napisane su:

Čitanje CSV testnih slučajeva
Slanje poruka API-u
Provjera rezultata
Izračun statistike
Spremanje rezultata u CSV

Podaci korišteni su raspoređeni u test_cases.csv, gdje se nalaze svi upiti za testiranje, i tocni_odgovori.csv, gdje su uneseni upiti koji su priloženi uz popis namjera kako bi razumjeli vrstu pitanja koje trebamo postavljati. Test_cases je služio za samo testiranje, dok je tocni_odgovori služio za računanje prosjeka i mediana „confidence-a“ na pitanjima za koje znamo da chatbot ima točan odgovor.

Rezultati

Cijeli kod se vrtio __, što je posljedica prolazaka kroz sve upite za testiranje iz dvije tablice.
Što se tiče podataka iz tocni_odgovori, prosjek i median su dosta niski. S obzirom da za testiranje nemamo točan broj koji možemo koristiti kao granicu za nizak ili visok prag sigurnosti modela (confidence), improvizirao sam računajući median sigurnosti za pitanja koja model sigurno točno odgovara. Ta tablica sadrži 22 upita te za svaki točno klasificira intent, dok je pak confidence dosta nizak. Median confidence je 0.2, dok je prosjek oko 0.13. Za daljnju klasifikaciju testnih upita korišten je ipak median confidence.

Od __ testnih upita, njih __ je zadovoljilo, odnosno točno pogodilo namjeru korisnika, što nam daje __% točnosti. Gledajući rezultate, zapažamo razne razloge pogrešne klasifikacije, ali i raznolik prag sigurnosti modela.

S obzirom na nizak median sigurnosti koji smo ranije izračunali, čak 81 upit je ispod mediana, ali i dalje dio njih daje točan intent. 43 upita, odnosno __ % daju nizak prag sigurnosti za neki intent, ali ga i dalje pogađaju, stoga razina sigurnosti nije bila metrika koju smo mogli uzeti u obzir za točnost klasifikacije.

Točnost klasifikacije je stoga u results.csv jedino određena osnovnom, jednostavnom mjerom, a to je usporedba točnog intenta i predviđenog intenta.

Analiza rezultata

Pogrešna predviđanja intenta su kroz ovo testiranje uzrokovana raznim propustima bota. Iako je test automatiziran, odlučio sam se za osobni pregled loših rezultata kako bih uvidio zbog čega je došlo do pogreške, umjesto da npr. kod svakog test casea upišem razlog zbog kojeg testiram baš taj upit (npr. raspodjela po gramatičkim greškama).

Za startni test bota, ovakav pristup ima smisla zbog malog broja testnih upita, ali već kod nekih kasnijih testiranja bilo bi dobro automatizirati i raspodjelu uzroka pogreške zbog uštede vremena.

Pregledavajući 38 pogrešnih predviđanja, greške možemo podijeliti na:

1. **Jezična varijabilnost:** Bot nije sposoban prepoznati riječi na drugim jezicima, kao ni uvedenice u neformalnom hrvatskom.
2. **Greške u unosu:** Greške kod tipkanja su česta pojava, pogotovo kod populacije koja nije toliko informatički obrazovana, stoga je bilo logično uključiti nekoliko ovakvih primjera unosa. 5/10 primjera s greškom u unosu su neprepoznati. Prepoznati su većinom primjeri čija je ključna riječ raspoznavanja (npr. parking) minimalno iskrivljena (npr. parkng)
3. **Neformalni govor:** Kod ovakvih unosa, bot nema problema dok god je neka od ključnih riječi spomenuta. Npr., kod pitanja „Kad otvoreno?“ dobijemo točan odgovor zbog pitanja „Kada“, dok „U koliko otvarate?“ dobijemo netočan odgovor zbog pitanja „koliko“, pa bot pretpostavlja da je riječ o cijenama ulaznice.
4. **Dvosmislenost pitanja:** Kod nekih pitanja, gdje zbog neformalnog govora (kao u primjeru iznad) koristimo pitanje poput npr.“Koliko je za parking?“, bot zbog pitanja „koliko“ pretpostavlja da je riječ o ulaznicama. Zbog ovakvih situacija, gdje bi bot trebao raspologati o informacijama cijena za više usluga, ne bi se trebao bazirati samo na pitanju „Koliko“, nego bi trebao i gledati npr. na riječ „parking“ koja bi mu puno više pomogla.
5. **Upiti bez specijalnih znakova:** Što se tiče specijalnih znakova, točnost odgovora varira, ali nema previše problema. Npr. u situaciji „nudite li clanstvo?“, bot točno prepoznaje intent, ali negdje gdje je specijalni znak u riječi bitnijoj za prepoznavanje, npr. kod „Koje su izlozbe danas?“, daje pogrešan odgovor. Ponovno dolazimo do zaključka da bot previše ovisi o jednoj ili dvije riječi po svakom intentu, po kojima prepoznaje namjeru.
6. **Fallback:** Ono što sam primijetio prilikom testiranja, ali nisam uključio u konačni skup testnih upita jer nema smisla, je da bot nema fallback opciju, odnosno prilikom nekog totalno nepoznatog pitanja, nevezanog za temu, neće odgovoriti s npr. „Nisam Vas razumio, molim ponovite pitanje.“, nego će dati neki odgovor po upitnoj riječi iz unosa.
7. **Kontekstualno razumijevanje:** Za ovakav jednostavan chatbot je previše za očekivati da će pratiti prošla pitanja i odgovore, ali je možda vrijedno spomenuti da bot nema sposobnost

kontekstualnog praćenja razgovora. Npr. kod pitanja „Imate li parking?“, dati će točan odgovor, no s pratećim pitanjem „Koliko košta?“, naravno da će se referencirati na cijenu ulaznica kao intent. To je nešto što bi se moglo optimizirati kada bi takav bot bio stvarno pušten u produkciju.

Zaključno, postoji više slabih strana bota na kojima bi se trebalo/moglo poraditi. Vjerujem da se većina problema navedenih u gornjim točkama može riješiti dodatnim treniranjem modela, tako da osim što prepoznaje upite po upitnim zamjenicama, može doći do odgovora i iz drugih ključnih riječi, ako ne čak i iz konteksta razgovora.

Bilingualnost bi mogla biti nešto što bi unaprijedilo bot, ako uzmemo u obzir turiste koji potencijalno posjećuju taj muzej. Ove dvije popravke bi na ovako niskoj razini chatbota bile dosta jednostavne za implementirati, dodatno trenirajući model na širem skupu pitanja ili na skupu pitanja na stranim jezicima.

Što se tiče unapređivanja modela, neke stvari bi bile jednostavne za implementirati. Npr., kod fallback odgovora, najjednostavnije bi bilo za svako pitanje ispod neke granice sigurnosti, dati generičan, općenit odgovor koji će potaknuti korisnika na reformuliranje pitanja.

Kontekstualno razumijevanje bi bilo potrebno za neki veći, razvijeniji model, ali možda ga je vrijedno spomenuti i ovdje. Umjesto da se svako pitanje tretira zasebno, izolirano, može se dodati memorija konverzacije. Model može uzimati u obzir posljednjih N pitanja kako bi mogao voditi razgovor s korisnikom, a to je moguće implementacijom RNN-ova ili LSTM-ova koji su danas dosta jednostavni za implementirati zahvaljujući raznim gotovim rješenjima i bibliotekama. Rekurentne neuronske mreže su često rješenje kod današnjih botova za razgovor s LLM-om, jer mogu uzeti u obzir cijelu povijest razgovora za jako malo vremena.