UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

Dragana Zdravković

METODE ZA AUTOMATSKU OPTIMIZACIJU I STRUKTURIRANJE UPITA ZA VELIKE JEZIČKE MODELE

master rad

Beograd, 2026.

Mentor:

prof. dr Aleksandar Kartelj
Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:

prof. dr Mladen Nikolić
Univerzitet u Beogradu, Matematički fakultet

dr Staša Vujičić Stanković
Univerzitet u Beogradu, Matematički fakultet

Datum odbrane: _____

Naslov master rada: Metode za automatsku optimizaciju i strukturiranje upita za velike jezičke modele

Rezime: Kvalitet odgovora velikih jezičkih modela u velikoj meri zavisi od kvaliteta upita koji im se prosleđuje. Upiti koje pišu stvarni korisnici često su zašumljeni: sadrže suvišne reči i pozdrave, ponavljaju isti zahtev na više načina, imaju loš redosled instrukcija i protivrečnosti. U ovom radu predstavljen je sistem koji upit automatski čisti, uklanja ponavljanja i preuređuje redosled instrukcija, sa ciljem da model da bolje strukturiran odgovor uz manji broj ulaznih tokena. Optimizator je zasnovan na tri komplementarne, potpuno algoritamske tehnike: lingvističkim heuristikama, deduplikaciji zasnovanoj na vektorskim reprezentacijama rečenica i grafovskom uređivanju redosleda instrukcija. Za evaluaciju je konstruisan testni skup sa parovima čistih i namerno zašumljenih upita, izveden iz skupa podataka BPO, i uveden je okvir merenja *oporavka*: koliko se odgovor na optimizovan upit približi odgovoru na čist (referentni) upit u poređenju sa odgovorom na zašumljen upit - to jest koliki deo kvaliteta izgubljenog zbog šuma optimizator vrati nazad. Rezultati pokazuju da instrukciono podešeni modeli implicitno čiste i loše formulisan upit, pa je prostor za popravku *tačnosti* na blago zašumljenim upitima mali; glavna vrednost optimizacije na tom režimu leži u efikasnosti (kraći i uredniji upit) i strukturi, a ne u skoku tačnosti. Kontrast sa baznim (instrukciono nepodešenim) modelom pokazuje da ista optimizacija donosi približno dvostruko veći oporavak na referentnim merama sličnosti kod baznog nego kod instrukcionog modela. Na opširnijim, jako zašumljenim upitima optimizator smanjuje dužinu ulaza za oko 54% (mereno tokenizatorom modela) bez gubitka informacija, a tada se javlja i skroman ali merljiv porast tačnosti odgovora na oba tipa modela.

Ključne reči: veliki jezički modeli, optimizacija upita, deduplikacija, instrukciono podešavanje, evaluacija generisanog teksta, BERTScore

Sadržaj

Glava 1

Uvod

Veliki jezički modeli (engl. *large language models*, u nastavku LLM) poslednjih godina postali su osnovni alat za širok spektar zadataka obrade prirodnog jezika, od sažimanja i prevođenja do odgovaranja na pitanja i pisanja koda. Njihova upotreba u praksi gotovo uvek se svodi na isti obrazac: korisnik formuliše upit (engl. *prompt*) na prirodnom jeziku, a model na osnovu tog upita generiše odgovor. Zbog toga kvalitet i preciznost dobijenog odgovora presudno zavise od toga kako je upit napisan. Ista suštinska namera, izražena na dva različita načina, može dovesti do osetno različitih odgovora.

U praksi, međutim, malo ko piše uredno. Upiti koje stvarni korisnici zaista otkucaju retko liče na pažljivo sročene instrukcije. Oni su najčešće zašumljeni: sadrže pozdrave i uvode, suvišne reči i uzrečice, ponavljanje istog zahteva u više navrata, loš redosled u kome se traženi zaključak pojavljuje pre ulaznih podataka, kao i sitne protivrečnosti i digresije koje ne menjaju sam zadatak, ali otežavaju njegovo razumevanje. Takav upit i dalje sadrži sve informacije potrebne za tačan odgovor, ali je ta informacija zatrpana u šumu.

Prirodno se nameće pitanje da li je moguće takav zašumljen upit automatski transformisati u čistiju i bolje strukturiranu verziju, pre nego što se prosledi modelu, tako da model da bolji odgovor, a da se pritom ne izgubi nijedan podatak iz originalnog zahteva. Upravo to je predmet ovog rada.

1.1 Predmet i cilj rada

Cilj rada je da se projektuje, implementira i evaluira sistem za automatsku optimizaciju i strukturiranje upita namenjenih velikim jezičkim modelima. Pod

optimizacijom se ovde podrazumeva niz transformacija koje upit čine jasnijim i sažetijim: uklanjanje ćaskanja i uzrečica, uklanjanje ponovljenih instrukcija i uređivanje redosleda tako da kontekst i podaci dođu prvi, zatim glavni zadatak, a ograničenja na kraju. Ključno ograničenje je da nijedna suštinska informacija (svaka činjenica, broj, ulazni podatak ili zahtev) ne sme biti izgubljena tokom optimizacije.

Za razliku od pristupa koji za prepisivanje upita koriste sam jezički model, u ovom radu optimizator je namerno napravljen kao potpuno algoritamski postupak, bez pozivanja LLM-a. Takav izbor donosi predvidivost, ponovljivost, niske troškove izvršavanja i, što je najvažnije za istraživanje, mogućnost da se svaka pojedinačna tehnika uključuje i isključuje i tako meri njen doprinos.

1.2 Doprinosi rada

Glavni doprinosi rada su sledeći:

- Predložen je i implementiran algoritamski optimizator upita koji objedinjuje tri komplementarne tehnike: lingvističke heuristike za čišćenje, deduplikaciju zasnovanu na vektorskim reprezentacijama rečenica i grafovsko uređivanje redosleda instrukcija, uz mehanizam zaštite podataka u upitu od izmene.

- Konstruisan je testni skup prilagođen problemu, sa parovima čistih i namerno zašumljenih upita, i uveden je okvir merenja *oporavka* koji dozvoljava objektivno poređenje odgovora pre i posle optimizacije u odnosu na odgovor na čist upit.

- Empirijski je pokazano da instrukciono podešeni modeli u velikoj meri sami čiste zašumljen upit, zbog čega je prostor za popravku tačnosti mali; ovaj nalaz je potvrđen na dva nezavisna načina (smanjivanjem veličine modela i agresivnijim zašumljivanjem).

- Kroz kontrast instrukcionog i baznog modela pokazano je da ista optimizacija donosi korist na oba tipa modela, ali u različitom obimu: na baznom modelu oporavak na referentnim merama sličnosti je približno dvostruko veći nego na instrukcionom, jer instrukcioni model svojom robusnošću sam nadoknađuje deo onoga što optimizator popravlja.

- Pokazano je da korist od optimizacije raste sa količinom uklonjivog šuma u upitu: na opširnim, jako zašumljenim upitima optimizator smanjuje broj tokena za oko 54% bez gubitka informacija i istovremeno donosi merljiv porast tačnosti odgovora na oba tipa modela, dok je na blago zašumljenim, urednijim upitima glavni doprinos ušteda tokena (oko 4%) uz mali pomak u tačnosti. Oba nivoa predstavljaju realistične tačke na rasponu koliko korisnički upiti mogu biti zašumljeni.
- Pokazano je da optimizacija čini model doslednijim: kada isti zahtev stigne formulisan na više različitih načina, odgovori posle optimizacije postaju međusobno mnogo sličniji, jer optimizator sve te formulacije svede na skoro isti čist upit. Efekat je prisutan kod oba tipa modela, a najizraženiji kod baznog.
- Glavni nalazi o oporavku potkrepljeni su formalnom statističkom analizom (bootstrap intervali poverenja i Vilkoksonov test) i proverom LLM sudije modelom iz druge porodice, čime je pokazano da zaključci nisu slučajnost izbora testnog skupa ni posledica pristrasnosti sudije prema odgovorima iz sopstvene porodice modela.

1.3 Struktura rada

U glavi 2 dat je pregled potrebnih teorijskih osnova i srodnih radova. Glava 3 opisuje konstrukciju testnog skupa i okvir merenja oporavka. U glavi 4 detaljno je opisan predloženi optimizator upita. Glava 5 predstavlja metodologiju evaluacije, modele i mere kvaliteta. Rezultati eksperimenata izloženi su u glavi 6. Glava 7 sadrži interpretaciju nalaza, ograničenja, zaključak i pravce daljeg rada.

Glava 2

Teorijske osnove i srodni radovi

2.1 Veliki jezički modeli

Savremeni jezički modeli zasnovani su na arhitekturi transformera [17], koja je mehanizmom pažnje omogućila efikasno modelovanje dugih zavisnosti u tekstu i obučavanje na velikim korpusima. Modeli obučeni na dovoljno velikoj količini teksta pokazuju sposobnost rešavanja zadataka koje tokom obučavanja nisu eksplicitno videli, i to samo na osnovu opisa zadatka u upitu [2]. Ova sposobnost učenja iz konteksta (engl. *in-context learning*) upravo je razlog zbog kog je formulacija upita postala bitno pitanje.

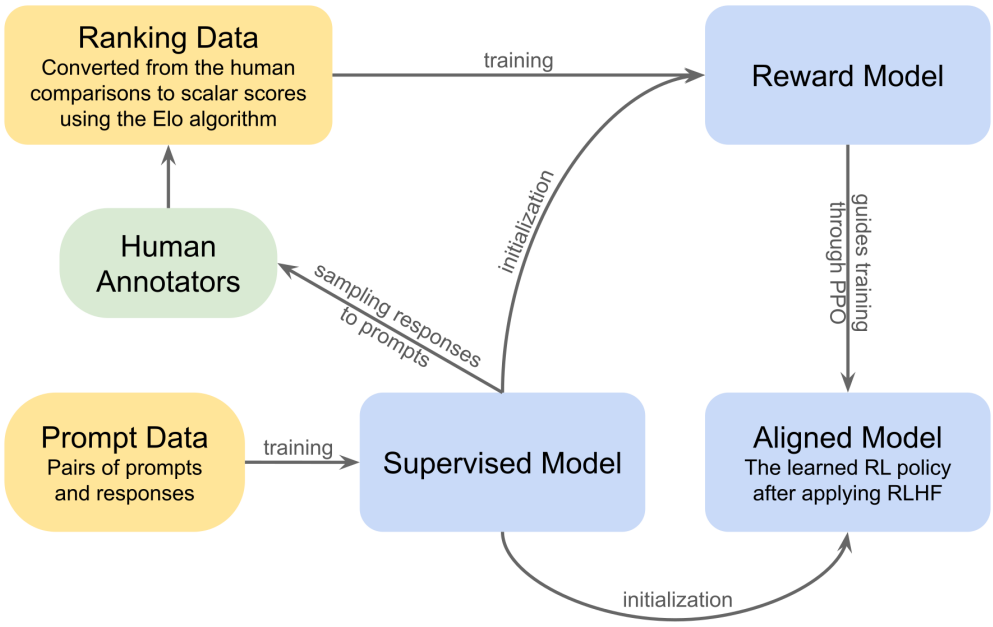
U radu se koristi porodica modela Qwen2.5 [21], koja je dostupna u više veličina i u dve varijante: baznoj i instrukciono podešenoj. Ta podela je od suštinskog značaja za zaključke rada, pa je posebno obrađena u nastavku.

2.2 Bazni i instrukciono podešeni modeli

Bazni model je model koji je samo prethodno obučen (engl. *pretrained*) na velikom korpusu teksta, sa jedinim ciljem da predviđa sledeći token. Takav model u suštini nastavlja zadati tekst i veoma je osetljiv na to kako je upit formulisan i strukturiran; ako upit ne liči na tekst iz kojeg bi prirodno sledio željeni odgovor, kvalitet izlaza brzo opada.

Instrukciono podešen model (engl. *instruction-tuned*) dobija se dodatnim obučavanjem baznog modela na parovima instrukcija i željenih odgovora, često uz učenje potpomognuto ljudskim povratnim informacijama [11, 20] (slika 2.1). Cilj tog dodatnog treninga je da model nauči da prati uputstva i vodi razgovor.

Posledica je da su instrukcioni modeli znatno robusniji: oni relativno pouzdano prepoznaju nameru korisnika čak i kada je upit loše formulisan. Kao što će biti pokazano u glavi 6, upravo ta robusnost otežava merenje efekta optimizacije upita, jer model i sam obavlja deo posla koji bi optimizator inače uradio.



Slika 2.1: Pojednostavljen prikaz instrukcionog podešavanja uz ljudske povratne informacije (RLHF): nadgledani model se inicijalizuje iz baznog i dodatno obučava na parovima upit-odgovor; ljudski anotatori rangiraju odgovore, a od tih rangiranja se obučava model nagrade, koji zatim (kroz optimizaciju polise, npr. PPO) vodi poravnavanje finalnog modela. Ovakav postupak daje robusnost instrukcionih modela, ali je skup jer se oslanja na ljudski rad. Izvor: [13].

2.3 Formulacija i optimizacija upita

Osetljivost modela na formulaciju upita motivisala je čitavu oblast inženjeringa upita (engl. *prompt engineering*) i njegove automatizacije. Jedan pravac istraživanja automatski pretražuje ili generiše upite koji maksimizuju kvalitet odgovora; tako, na primer, metod APE (engl. *Automatic Prompt Engineer*) koristi sam jezički model da predloži i izabere formulacije instrukcija [24]. Bliži ovom radu je

pristup optimizacije upita kao crne kutije (engl. *black-box prompt optimization*, BPO) [3], gde se upit prepisuje u bolju verziju bez ikakvog menjanja parametara samog modela. Skup podataka BPO, koji sadrži parove izvornih i optimizovanih (prepisanih) upita, poslužio je kao polazna osnova za konstrukciju testnog skupa u ovom radu.

Druga linija istraživanja upit ne prepisuje, već ga sažima: LLMLingua [6] tokenski (a ne rečenički) uklanja delove upita male informativnosti, koristeći pomoćni jezički model da proceni važnost svakog tokena, i tako postiže i do dvadesetostruku kompresiju uz mali gubitak kvaliteta. Optimizator predložen u ovom radu deli cilj uštede tokena, ali se razlikuje po tome što je potpuno bez modela (ne koristi nikakav pomoćni LLM) i radi na nivou rečenica i fraza vođen lingvističkim pravilima, a ne statističkom procenom važnosti pojedinačnih tokena.

Nezavisno od pitanja *šta* upit sadrži, poznato je i da su jezički modeli osetljivi na *kako* je upit oblikovan i uređen: redosled few-shot primera može promeniti tačnost od skoro savršene do nivoa slučajnog pogađanja [9], a čak i semantički ekvivalentne promene u formatiranju upita (razmaci, separatori, redosled polja) mogu povećati tačnost značajno, nezavisno od veličine modela ili instrukcionog podešavanja [16]. Ovi nalazi direktno motivišu korak grafovskog uređivanja redosleda (glava 4) i eksperiment robusnosti na parafraze (odeljak 6.8) u ovom radu: ako je model osetljiv na uređenje upita, uređivanje ka doslednom, kanonskom obliku treba da smanji tu osetljivost.

Za razliku od navedenih pristupa koji se oslanjaju na jezički model prilikom prepisivanja ili sažimanja upita, optimizator predložen u ovom radu je algoritamski i deterministički. Time se gubi deo fleksibilnosti, ali se dobijaju predvidivost, ponovljivost i transparentnost, kao i mogućnost preciznog merenja doprinosa svake tehnike.

2.4 Vektorske reprezentacije rečenica

Za prepoznavanje ponovljenih instrukcija u zašumljenom upitu potrebno je meriti semantičku sličnost rečenica, a ne samo doslovno poklapanje reči. U tu svrhu koriste se vektorske reprezentacije (engl. *embeddings*) dobijene modelom tipa Sentence-BERT [15], konkretno kompaktnim modelom `all-MiniLM-L6-v2` izvedenim destilacijom (engl. *knowledge distillation* – postupak u kome manji model uči da oponaša izlaze većeg, čime se dobija znatno brži i lakši model uz mali gubitak

kvaliteta) [19]. Rečenice se preslikavaju u vektore fiksne dimenzije tako da semantički bliske rečenice imaju veliku kosinusnu sličnost, što omogućava pouzdano otkrivanje parafraziranih ponavljanja.

Slika 2.2 ilustruje ovo svojstvo na jednostavnom primeru. Korišćeno je šest rečenica: tri parafraze istog zahteva za sažimanje članka (oznake `S1`–`S3`), dve parafraze zahteva za prevođenje (`T1`–`T2`) i jedan nepovezan zahtev (`L1`):

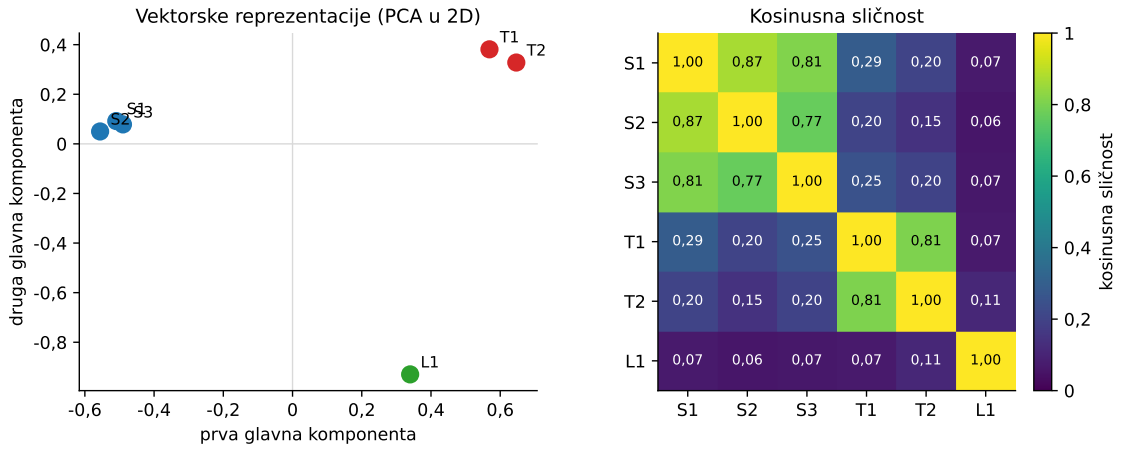
- `S1`: *„Summarize this article in three sentences."*
- `S2`: *„Give me a three-sentence summary of the article."*
- `S3`: *„Can you sum up the article in about three sentences?"*
- `T1`: *„Translate the following text into French."*
- `T2`: *„Please render this text in French."*
- `L1`: *„List all prime numbers below twenty."*

Tri parafraze prvog zahteva i dve parafraze drugog grupišu se u vektorskom prostoru, dok je nepovezan zahtev jasno izdvojen. Kosinusna sličnost unutar grupe parafraza visoka je (najbliži par dostiže 0,87, iznad praga 0,85 koji koristi deduplikacija), a između različitih zahteva niska (oko 0,15). Upravo ta razlika – visoka sličnost za preformulacije istog zahteva, niska za suštinski različite – signal je na kome se zasniva prepoznavanje ponovljenih instrukcija u koraku deduplikacije (glava 4).

2.5 Mere kvaliteta generisanog teksta

Evaluacija generisanog teksta oslanja se na više komplementarnih mera. Referentne mere porede generisani odgovor sa ciljnim (referentnim) odgovorom:

- **BERTScore** [22] meri semantičku sličnost preko kontekstualnih reprezentacija reči (u ovom radu na osnovu modela tipa RoBERTa [8]), pa dobro hvata poklapanje po značenju i onda kada se formulacije razlikuju.
- **BLEU** [12] meri preklapanje n-grama i osetljiv je na doslovno poklapanje reči. Pošto se rezultat BLEU razlikuje od implementacije do implementacije



Slika 2.2: Vektorske reprezentacije šest primera rečenica dobijene modelom `all-MiniLM-L6-v2`. Levo: dvodimenziona projekcija (PCA) na kojoj se parafraze istog zahteva grupišu. Desno: matrica kosinusne sličnosti; parafraze istog zahteva imaju visoku sličnost (svetlije ćelije), a različiti zahtevi nisku.

(usled različite tokenizacije i zaglađivanja, engl. *smoothing*), koristi se SacreBLEU [14] – standardizovana implementacija sa fiksiranom tokenizacijom, koja daje uporedive i ponovljive rezultate.

- **ROUGE-L** [7] zasnovan je na najdužem zajedničkom podnizu i meri poklapanje na nivou sekvence.

Ove tri mere odabrane su zato što su komplementarne i pokrivaju različite nivoe poklapanja: BERTScore hvata sličnost po značenju, BLEU doslovno poklapanje reči i kraćih fraza, a ROUGE-L poklapanje na nivou dužih sekvenci. Time se razdvaja da li se odgovor približio referentnom po smislu od toga da li se približio i po samoj formulaciji, a to je upravo ono što je relevantno pri merenju oporavka (glava 3). Uz to, sve tri mere su jeftine, determinističke i ponovljive, pa se lako računaju nad celim skupom i ne unose dodatnu nasumičnost u evaluaciju.

Za ovaj rad, međutim, važno je i njihovo ograničenje. Sve tri porede odgovor sa jednom referencom i mere sličnost površinskog oblika, a ne stvarnu tačnost. Suštinski tačan odgovor, formulisan drugačije od reference, dobiće nisku BLEU i ROUGE-L ocenu iako je sadržinski ispravan, a nijedna od ovih mera ne procenjuje da li je odgovor zaista tačan. Zato su referentne mere adekvatne da pokažu koliko se odgovor po obliku i značenju približio idealnom (što je i suština oporavka), ali

nisu dovoljne da potvrde ispravnost - i upravo to je motiv za dodatnu procenu koja sledi.

Pored referentnih mera, sve je rasprostranjenija upotreba samog jezičkog modela kao sudije (engl. *LLM-as-a-judge*) [23]. Takav sudija može da proceni suštinsku tačnost odgovora u odnosu na referencu, nezavisno od formulacije, što ga čini korisnom dopunom automatskim merama sličnosti. U ovom radu korišćene su i referentne mere i sudija zasnovan na modelu, kako bi se razdvojile promene u formulaciji od promena u samoj tačnosti.

Pored kvaliteta odgovora, meri se i efikasnost samog upita. Kako veliki jezički modeli obrađuju i naplaćuju ulaz u tokenima, glavna mera efikasnosti je broj tokena upita pre i posle optimizacije, izračunat tokenizatorom samog generativnog modela (Qwen2.5); broj reči i broj karaktera navode se kao grublje, ali intuitivnije aproksimacije. Ušteda tokena je merljiva korist optimizatora nezavisna od promene u tačnosti odgovora.

Glava 3

Podaci i eksperimentalna postavka

3.1 Okvir merenja oporavka

Osnovna teškoća u evaluaciji optimizacije upita jeste odabir reference u odnosu na koju se meri kvalitet odgovora. Pošto je cilj da zašumljen upit počne da se ponaša kao čist, kao referenca je uzet upravo odgovor modela na čist upit. Taj odgovor predstavlja najbolje čemu optimizacija upita realno može da teži, jer je dobijen od istog modela pod idealnim ulazom.

Za svaki zadatak postoje tri varijante upita i, posledično, tri odgovora. Radi jednostavnosti, odgovor imenujemo prema upitu koji ga je proizveo:

- *čist upit* daje *referentni odgovor*, koji je gornja granica dostižnog kvaliteta;
- *zašumljen upit* daje *zašumljen odgovor*, koji se ocenjuje u odnosu na referentni;
- *optimizovan upit* (izlaz predloženog sistema) daje *optimizovan odgovor*, koji bi trebalo da bude bliži referentnom nego što je zašumljen.

Sistem je uspešan ako je optimizovan odgovor bliži referentnom odgovoru nego što je to zašumljen odgovor. Tu razliku nazivamo *oporavkom* (engl. *recovery*): to je poboljšanje mere sličnosti koje optimizator ostvaruje u odnosu na zašumljen upit. Formalno, ako sa $m(\cdot)$ označimo neku meru sličnosti odgovora sa referentnim odgovorom, oporavak je

$$\Delta m \;=\; m(\text{optimizovan}) - m(\text{zašumljen}).$$

Pozitivna vrednost Δm znači da je optimizator približio zašumljen odgovor idealnom.

3.2 Izbor i priprema podataka

Kao izvor upita korišćen je skup podataka BPO [3], koji za svaki primer sadrži par (izvorni, optimizovani) upit. U ovom radu polazi se od optimizovanih (čistih) upita, koji su dovoljno uredni da posluže kao osnova za kontrolisano zašumljivanje.

Cilj optimizatora je da popravi upravo onakve upite kakve stvarni korisnici pišu u praksi: nesređene, sa suvišnim tekstom, lošim redosledom i ponavljanjima. Da bi se efekat optimizacije uopšte mogao izmeriti, potreban je par u kome se zna kako izgleda „idealan" upit i kakav odgovor on daje, a jedino što se menja jeste forma upita. Zato se od svakog čistog upita namerno pravi njegova zašumljena verzija: čist upit daje ciljni (referentni) odgovor, a zašumljeni oponaša realan korisnički unos koji optimizator treba da očisti. Drugim rečima, upite ne kvarimo da bismo dobili loše odgovore, već da bismo dobili kontrolisane uslove za merenje. Pošto se između čistog i zašumljenog upita menja samo forma, dok sadržaj ostaje isti, svaku razliku u kvalitetu odgovora možemo pripisati isključivo formi upita.

Okvir merenja oporavka pretpostavlja da postoji jedan dobro definisan ciljni odgovor. Zbog toga je skup ograničen na upite čiji je tačan odgovor u suštini fiksiran, tako da se menja samo formulacija, a ne i sadržaj odgovora. Upiti otvorenog tipa (kreativno pisanje, preporuke, mišljenja, generisanje koda kod kojih je moguće više podjednako ispravnih odgovora) uklonjeni su i zamenjeni kvalifikujućim upitima iz *train* dela BPO skupa. Konačni skup ima ukupno 500 primera podeljenih u dve ciljne grupe - grupu od 300 upita (oznaka *validation*) i grupu od 200 upita (oznaka *test*); oznake ovde označavaju samo veličine grupa, a ne izvorne podele BPO skupa, pošto zamenski upiti potiču iz *train* dela BPO (u konačnom skupu je oko 291 od 500 primera poreklom iz *train* dela). Ovo ne utiče na valjanost evaluacije jer je predloženi optimizator deterministučan i ne uči iz podataka (nad ovim primerima se ništa ne trenira niti podešava), nema klasične podele na skup za podešavanje i skup za testiranje, niti rizika od curenja informacija iz test skupa. Zato se sve mere izveštavaju nad celim skupom od 500 upita.

3.3 Konstrukcija zašumljenih upita

Budući da BPO ne sadrži realistične, zašumljene korisničke verzije upita, one su konstruisane namerno. Samo zašumljivanje izveo je veliki jezički model, vođen unapred definisanom specifikacijom: za svaki čist upit model je generisao verziju kakvu bi mogao da otkuca nepažljiv korisnik. Ovaj izbor je nameran – upit koji „pokvari" drugi model deluje prirodnije i raznovrsnije nego šum dobijen ručno pisanim pravilima, a specifikacija i naknadna provera obezbeđuju da nijedan podatak ne bude izgubljen ni izmenjen. Dozvoljene transformacije obuhvataju:

- dodavanje uzrečica i suvišnosti (pozdravi, ponovljeni zahtevi, isto uputstvo iskazano na više načina);
- narušavanje logičkog redosleda instrukcija (na primer, traženje zaključka pre navođenja ulaznih podataka);
- blage digresije i nebitan kontekst koji ne menja zadatak;
- labaviju formulaciju (žargon, greške u kucanju, nedosledna interpunkcija i pisanje velikih slova);
- stvarna instrukcija zakopana u nebitnom kontekstu;
- dupliranje pojedinih ograničenja ili primera.

Pri tome se poštuju stroga ograničenja: zadatak se ne sme promeniti, nijedna suštinska informacija (činjenica, ograničenje, ulazna vrednost) ne sme biti izostavljena, ne smeju se uvoditi novi zahtevi, i referentni odgovor mora ostati ispravan odgovor i na zašumljeni upit. Na taj način svaka razlika u kvalitetu odgovora potiče isključivo od forme upita, a ne od promene sadržaja, što je preduslov za pošteno merenje oporavka.

Primer para čistog i zašumljenog upita:

> **Čist:** Sort the following list of numbers in ascending order: [8, 3, 10, 4, 7, 6].
>
> **Zašumljen:** hey so i've got these numbers [8, 3, 10, 4, 7, 6] and i kinda need help. can you put them in order for me, from smallest to largest? just sort them please. the numbers again are [8, 3, 10, 4, 7, 6]. thanks!

Glava 4

Optimizator upita

Optimizator je jezgro sistema koji ovaj rad gradi. On zašumljen upit (`degraded_prompt`) pretvara u čist i strukturiran upit (`system_prompt`) potpuno algoritamski, bez pozivanja jezičkog modela. Postupak se sastoji iz jednog koraka zaštite podataka i tri komplementarne tehnike, koje se primenjuju redom. Svaka tehnika može nezavisno da se uključi ili isključi, tako da arhitektura omogućava i ablaciono ispitivanje pojedinačnih doprinosa.

Postupak se sastoji iz pet faza koje se primenjuju redom: zaštite podataka (mask), lingvističkog čišćenja (clean), deduplikacije (dedup), uređivanja redosleda (order) i vraćanja podataka (restore). Svaka faza opisana je u zasebnom odeljku u nastavku, a njihova kompozicija sažeta je na kraju poglavlja (odeljak 4.5).

4.1 Korak 0: zaštita podataka u upitu

Pre bilo kakve izmene teksta, optimizator pronalazi sve delove upita koji moraju ostati netaknuti i privremeno ih zamenjuje oznakama (engl. *placeholders*), da bi ih na kraju vratio u izvornom obliku. Štite se: niske pod navodnicima (stringovi, na primer `„good morning"`), delovi nalik kodu (blokovi u vitičastim zagradama `{...}`, definicije funkcija i klasa, uvučeni redovi), jednačine ($E = mc^2$) i liste u uglastim zagradama (`[3, 1, 2]`), kao i argumenti poziva funkcija (`count_words(„one two three")`). Tako optimizator ni u jednom koraku ne može da izmeni same podatke zadatka. Pošto ove oznake ne sadrže interpunkciju, one ne ometaju kasnije deljenje teksta na rečenice ni promenu redosleda.

Zaštićeni delovi ponekad su ugnježđeni jedan u drugom (na primer niz pod navodnicima unutar poziva funkcije, ili lista u uglastim zagradama koja i sama

sadrži navodnike), pa se vraćanje oznaka ponavlja sve dok u tekstu ne ostane nijedna, čime se i takvi, ugnježđeni podaci vraćaju doslovno.

Ilustracije radi, u nastavku je prikazano kako upit izgleda dok su podaci zaštićeni. Lista i poziv funkcije privremeno se zamenjuju oznakama `[D1]` i `[D2]` (ovde su prikazane šematski; u implementaciji su to interne oznake). Optimizator zatim menja samo tekst oko njih, bez rizika da promeni podatke, a na kraju ih vraća doslovno:

> **Ulaz:** Sort the list `[3, 1, 2]` and then call `count_words(„one two three")`.
>
> **Sa oznakama:** Sort the list `[D1]` and then call `[D2]`.
>
> **Značenje oznaka:** `[D1]` → `[3, 1, 2]`, `[D2]` → `count_words(„one two three")`.

4.2 Korak 1: lingvističke heuristike (čišćenje)

Nad rečenicama koje nisu maskirane primenjuju se lingvističke heuristike:

- uklanjaju se rečenice ćaskanja i digresija, to jest one koje nemaju imperativni glagol, upitnu reč niti zaštićene podatke, a zvuče kao uobičajene društvene fraze (na primer pozdravi, ljubazne fraze i lične digresije koje ne nose zadatak);
- uklanja se i uvodno-završni okvir razgovora: pozdravi na početku, izvinjenja, lične najave (*„hoping you could help"*), meta-najave o dostavljanju sadržaja (*„here's the article"*, *„below is..."*) i završni pozdravi (*„thanks in advance"*) – ali samo kada stvarna instrukcija ili podatak postoje na drugom mestu u upitu, da se pravi kontekst nikada ne izgubi;
- unutar preostalih rečenica uklanjaju se uzrečice i izrazi učtivosti (pozdravi, *please*, *could you / can you*, *kinda*, *like*, *i guess*, *you know*), a sa njihovih krajeva skidaju se nagomilane uvodne fraze (*„hey, so, quick question, ..."*) i razgovorni dodaci na kraju (*„...lol"*, *„...thanks"*, *„...or nah?"*);
- normalizuju se velika slova (veliko slovo na početku rečenice; samostalno englesko *i* postaje *I*), interpunkcija i višak beline.

Rečenica se prepoznaje kao instrukcija ako počinje upitnom reči, sadrži znak pitanja ili sadrži neki od imperativnih glagola iz unapred definisanog skupa; takve rečenice se čuvaju, dok se čisto društvene rečenice bez instrukcije uklanjaju. Pri svemu ovome, rečenice koje sadrže zaštićene podatke i nabrajani ulazi ili ponuđene opcije (na primer *Sentence 1 / Sentence 2*) nikada se ne uklanjaju, čime se čuva sav sadržaj zadatka.

4.3 Korak 2: deduplikacija zasnovana na vektorskim reprezentacijama

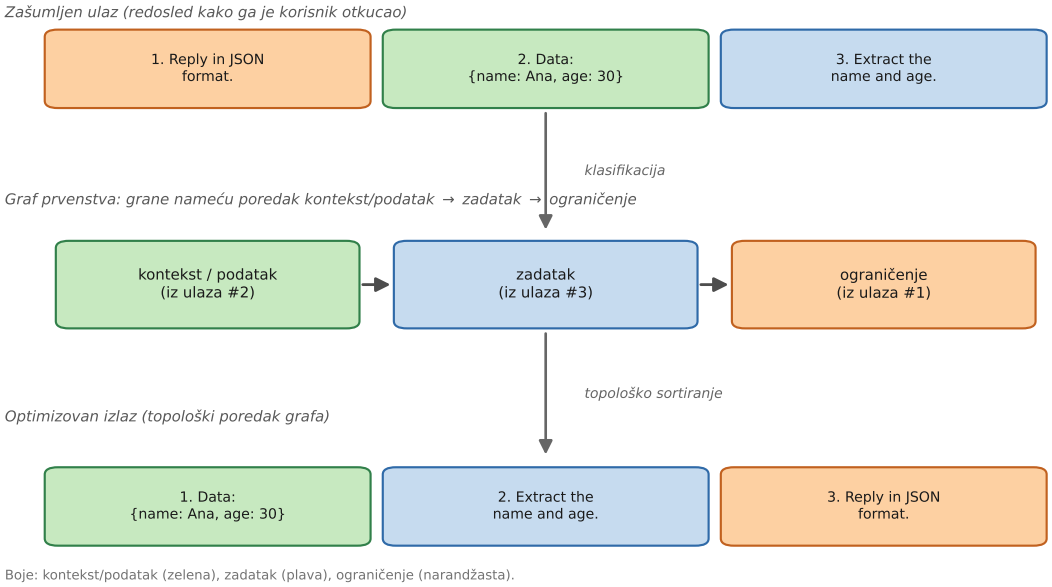
Zašumljeni upiti tipično ponavljaju istu instrukciju na više načina. Preostale rečenice se vektorizuju kompaktnim modelom `all-MiniLM-L6-v2` [15, 19], a zatim se svaki par čija je kosinusna sličnost veća ili jednaka pragu (podrazumevano 0,85) tretira kao ponavljanje. Slika 2.2 (glava 2) ilustruje zašto ovo radi: parafraze istog zahteva prelaze taj prag, dok nepovezani zahtevi ostaju znatno ispod njega. Od takvih rečenica zadržava se samo jedan predstavnik, i to najpotpuniji, uz očuvanje redosleda prvog pojavljivanja. Da se ne bi izgubili zaista različiti ulazi, iz deduplikacije su izuzete maskirane rečenice i nabrajani ulazi (na primer *Sentence 1 / Sentence 2*, *Sample 1 / Sample 2*), koji se nikada ne sažimaju. Ovaj izuzetak je nužan zato što su takvi ulazi po formi često vrlo slični, pa bi ih mera sličnosti lako pogrešno proglasila ponavljanjem. Neka je, na primer, dat sledeći upit za poređenje rečenica (prikazan radi jasnoće u čistom obliku):

> Compare the following two sentences and explain whether they describe
> the same event.
> Sentence 1: The cat sat on the mat.
> Sentence 2: The cat sits on a mat.

Ovde dve ulazne rečenice imaju kosinusnu sličnost iznad praga, pa bi ih deduplikacija spojila u jednu – a time bi sam zadatak (poređenje *dve* rečenice) izgubio smisao, jer bi ostala samo jedna. Slično, dva reda podataka poput *Sample 1: {visina: 170, težina: 65}* i *Sample 2: {visina: 180, težina: 90}* razlikuju se samo u zaštićenim (maskiranim) vrednostima, pa bi ih deduplikacija, koja poredi okolni tekst, mogla spojiti i tako obrisati jedan uzorak. Izuzimanjem maskiranih i eksplicitno nabrajanih ulaza ovakvi slučajevi se pouzdano izbegavaju.

4.4 Korak 3: grafovsko uređivanje redosleda

Svaka preostala jedinica (rečenica ili zaštićeni blok podataka) razvrstava se pomoću pravila nad samim tekstom, bez pozivanja modela: gleda se da li jedinica počinje upitnom rečju ili sadrži znak pitanja (znak zadatka), da li sadrži zaštićene (maskirane) podatke (znak konteksta/podatka) i da li se poklapa sa nekim od unapred zadatih izraza koji označavaju ograničenje formata ili obima (na primer *„in JSON format”*, *„one sentence”*, *„yes or no”*, *„no more than”*, *„step by step”*). Prema tome, jedinica se svrstava u kontekst/podatak, glavni zadatak ili ograničenje. Poredak se ne određuje ručnim premeštanjem, već se zadaje kao skup pravila prvenstva: formira se usmereni graf poretka (engl. *precedence graph*) pomoću biblioteke `networkx` [4], u kome grane kodiraju ciljni redosled kontekst/podatak → zadatak → ograničenje, a upit se zatim ispisuje u topološkom poretku tog grafa, uz očuvanje originalnog redosleda unutar iste klase. Prednost ovakvog, grafovskog formulisanja jeste što je opšte i lako proširivo: dodavanje novog pravila prvenstva svodi se na dodavanje grane, a topološko sortiranje uvek daje konzistentan izlaz. Ciljna struktura ka kojoj graf teži jeste ona kakvu ima čist upit – podaci i kontekst napred, zahtev u sredini, a ograničenja na kraju. Slika 4.1 prikazuje ovaj postupak na konkretnom upitu.



Slika 4.1: Uređivanje redosleda na konkretnom upitu sa tri jedinice. Zašumljen ulaz navodi ograničenje pre podataka i zadatka; svaka jedinica se klasifikuje kao kontekst/podatak, zadatak ili ograničenje, grane grafa nameću poredak kontekst/podatak → zadatak → ograničenje, a upit se ispisuje u topološkom poretku tako da podaci dođu prvi, zatim zahtev, a ograničenje formata na kraj.

Ovakav poredak opravdavaju dva razloga. Najpre, pregledom čistih (optimizovanih) upita iz skupa BPO [3] uočeno je da oni prvo navode ulazne podatke i kontekst, potom sam zahtev, a ograničenja formata i obima na kraju; pošto je cilj optimizatora da zašumljen upit približi baš takvom obliku, usvojen je isti redosled. Uz to, u istom pravcu idu i pojedine smernice za pisanje upita - Anthropic, na primer, savetuje da se ulazni podaci i kontekst postave na vrh upita, a zahtev i instrukcije na kraj [1].

Iako graf može da izrazi puno razdvajanje na sve tri klase, u glavnoj konfiguraciji optimizatora klasifikator je namerno postavljen oprezno: pouzdano izdvaja samo ona pravila prvenstva koja se tiču ograničenja, dok kontekst, podatke i glavni zahtev tretira kao jednu klasu. Praktična posledica je da se na kraj pomeraju isključivo rečenice koje su čisto ograničenje formata ili obima (na primer *„answer in one sentence”* ili *„reply in JSON”*), a sve ostale rečenice zadržavaju prvobitni međusobni redosled. Drugim rečima, u ovoj konfiguraciji graf se svodi na stabilno razdvajanje „ograničenja na kraj, ostalo nepromenjeno”; puno troklasno preuređivanje okvir podržava, ali je ostavljeno neaktivnim.

Razlog za takvu opreznost je dvostruk. Prvo, mnogi upiti su već smisleno poređani, pa bi grubo premeštanje moglo samo da ih pogorša – što se i pokazalo u ablaciji, gde je agresivnije preuređivanje smanjivalo oporavak. Drugo, agresivno pomeranje moglo bi da razdvoji glavni zahtev od podataka koji dolaze odmah iza njega – na primer, da odvoji *„Summarize the following text:"* od samog teksta koji treba sažeti. Time što dira samo jasno obeležena ograničenja, optimizator dobija urednu strukturu na kraju bez rizika po ostatak upita.

4.5 Sastavljanje izlaza

Sada kada su sve faze opisane, ceo postupak možemo sažeti jednom kompozicijom. Ako sa mask, clean, dedup, order i restore označimo pojedine faze, izlaz optimizatora je

$$\texttt{system\_prompt} = \mathrm{restore}\big(\mathrm{order}(\mathrm{dedup}(\mathrm{clean}(\mathrm{mask}(\texttt{degraded\_prompt}))))\big).$$

Nakon uređivanja, rečenice se spajaju u konačan tekst, maskirani blokovi se vraćaju u originalni oblik, a delovi koji su isključivo zaštićeni blok (na primer kod) izdvajaju se u zaseban red. Za svaki upit beleže se dužine pre i posle optimizacije. Glavna mera efikasnosti je broj tokena po tokenizatoru samog generativnog modela (Qwen2.5), jer se upiti modelu i naplaćuju u tokenima; dodatno se beleže broj reči i broj karaktera kao grublje aproksimacije. Tako se ušteda može izmeriti i onda kada se kvalitet odgovora gotovo ne menja.

Slike 4.2 i 4.3 prikazuju ceo postupak na dva stvarna upita iz testnog skupa. U oba slučaja gore je zašumljen upit kakav bi korisnik otkucao, a ispod rezultat optimizatora; precrtani delovi su suvišne fraze u razgovoru koje korak čišćenja uklanja. U prvom primeru (Slika 4.2) uz čišćenje se vidi i zaštita podataka: matematički izraz je prepoznat kao zaštićeni blok i prenet doslovno, bez ikakve izmene. U drugom primeru (Slika 4.3) do izražaja dolazi i uređivanje redosleda - u zašumljenom upitu zahtev („convert ...into a list of bullet points") stoji ispred samog članka, a optimizator ga pomera iza konteksta, tako da podaci dođu prvi, a zahtev sa format-ograničenjem na kraj.

Zašumljen upit (54 tokena)

~~quick q for you if you have a sec, no rush at all. feel free to keep it short, i don't need an essay or anything.~~ **determine the coefficient of x in the expression** `(2x^2+x+1)` . ~~thanks a ton in advance!~~

⇓ *optimizacija (≈ 67% manje tokena)*

Optimizovan izlaz *(18 tokena)*

Determine the coefficient of x in the expression `(2x^2+x+1)` .

zadatak `zaštićeni podatak` ~~*suvišne fraze*~~

Slika 4.2: *Optimizacija upita od početka do kraja. Precrtani delovi su suvišne fraze u razgovoru koje korak čišćenja uklanja, a istaknuti matematički izraz je zaštićeni podatak koji se prenosi doslovno. Rezultat je kraći i uredniji upit uz uštedu od oko 67% tokena (sa 54 na 18).*

Zašumljen upit (84 tokena)

~~so i was just sitting around the other night and got kinda curious.~~ **convert this article into a list of bullet points:** Article: The global pandemic is causing uncertainty and stress for many people. Besides the health risks, the economic impact has been significant and the effects will last longer than the initial disruption. ~~so yeah, that's pretty much the gist of what i'm asking. cheers, you're a lifesaver!~~

⇓ *optimizacija (≈ 45% manje tokena)*

Optimizovan izlaz *(46 tokena)*

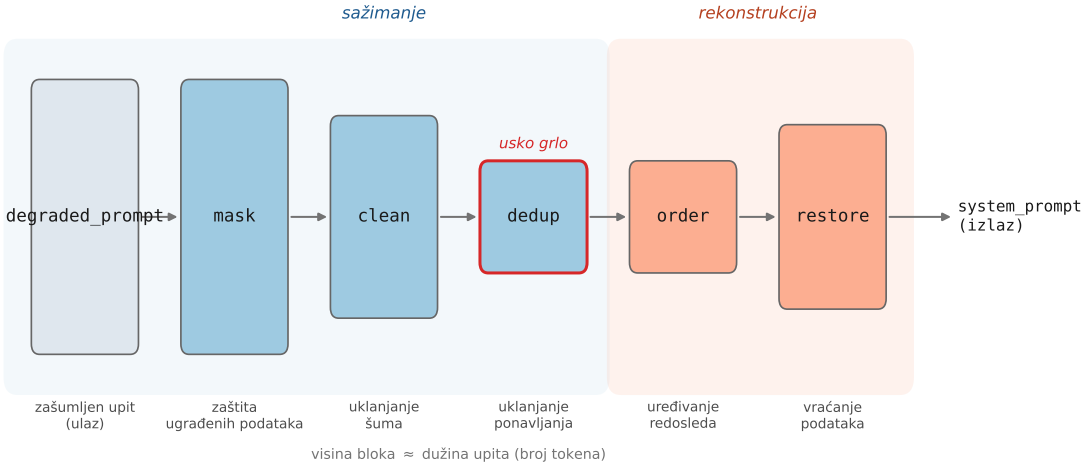
Article: The global pandemic is causing uncertainty and stress for many people. Besides the health risks, the economic impact has been significant and the effects will last longer than the initial disruption. ***Convert this article into a list of bullet points:***

zadatak i format-ograničenje *kontekst/podaci* ~~*suvišne fraze*~~

Slika 4.3: *Optimizacija upita kod koje se, pored čišćenja, aktivira i uređivanje redosleda. U zašumljenom upitu zahtev stoji pre članka; optimizator premešta zahtev sa format--ograničenjem iza konteksta, čime se dobija struktura kontekst → zadatak. Ušteda je oko 45% tokena (sa 84 na 46).*

Sada kada je ceo postupak opisan, vredi primetiti da on veoma podseća na kompresiju podataka i na autoenkodere. Autoenkoder je neuronska mreža koja ulaz najpre sažima u kraću, gušću reprezentaciju, a zatim iz nje rekonstruiše polazni ulaz; da bi rekonstrukcija uspela iz tako sažetog oblika, mreža mora da odbaci redundansu i zadrži samo bitne osobine [5]. Slično tome, optimizator uklanja jezički višak (ponavljanja, uzrečice, digresije), a čuva svaku informaciju zadatka.

Još je bliža analogija sa *denoising* autoenkoderom [18], koji se obučava da iz namerno pokvarene (zašumljene) verzije ulaza rekonstruiše čist original. Po postavci je to blisko eksperimentu u ovom radu: čist upit ima ulogu originala, degradacija odgovara procesu kvarenja, zašumljen upit je pokvarena verzija, a zadatak optimizatora je da je vrati ka čistom obliku, pri čemu je čist upit referenca. Slika 4.4 prikazuje ovu srodnost: faze optimizatora poređane su u oblik peščanog sata svojstven autoenkoderu, gde se zašumljen upit najpre sažima do „uskog grla" (očišćen i deduplikovan sadržaj), a zatim mu se vraćaju redosled i zaštićeni podaci; kada u polaznom upitu ima suvišnog teksta, rezultat je pri tome i kraći od njega.



Slika 4.4: Faze optimizatora (`mask`, `clean`, `dedup`, `order`, `restore`) poređane u oblik peščanog sata karakterističan za autoenkoder; visina svakog bloka odgovara dužini upita u toj fazi. Leva (plava) strana sažima upit uklanjanjem šuma i ponavljanja do „uskog grla", a desna (crvena) ga rekonstruiše uređivanjem redosleda i vraćanjem zaštićenih podataka.

Analogiju, ipak, ne treba shvatiti doslovno; između predloženog postupka i pravog autoenkodera postoji nekoliko suštinskih razlika. Autoenkoder se uči na podacima i u „uskom grlu" gradi apstraktnu, naučenu reprezentaciju, dok je predloženi optimizator deterministčan i definisan pravilima, a njegovo „usko grlo" nije

naučeni kod nego prosto kraći tekst na prirodnom jeziku. Dalje, kod autoenkodera sve informacije prolaze kroz to usko grlo i rekonstruišu se iz njega (uz neizbežan gubitak), dok se u predloženom postupku zaštićeni podaci uopšte ne sažimaju: oni zaobilaze usko grlo, čuvaju se sa strane i na kraju se vraćaju doslovno. Zato je i „sažimanje" bez gubitaka po sadržaju zadatka – odbacuje se samo redundansa u formulaciji, nikada sama informacija. Najzad, kod denoising autoenkodera ulaz se najčešće kvari nasumičnim šumom, dok su upiti u ovom radu pokvareni smisleno, jezičkim sredstvima. Analogija je zato korisna kao intuicija o obliku postupka (sažimanje pa rekonstrukcija), a ne kao tvrdnja o istovetnom mehanizmu.

Glava 5

Metodologija evaluacije

5.1 Modeli i okruženje

Generisanje odgovora i ocenjivanje sprovedeni su lokalno, pomoću sistema Ollama [10], nad porodicom modela Qwen2.5 [21]. Za generisanje su korišćene instrukcione varijante veličina 0.5B, 1.5B, 3B i 7B, kao i bazna varijanta veličine 3B. Pošto Ollama ne nudi baznu (instrukciono nepodešenu) varijantu modela Qwen2.5, ona je uvezena iz GGUF formata i konfigurisana odgovarajućom šablonskom datotekom. Dekodiranje je determinističko (temperatura 0, fiksirana početna vrednost generatora, ograničenje na 1024 nova tokena), čime je obezbeđena ponovljivost.

Oznake 0.5B-7B odnose se na broj parametara modela izražen u milijardama: tako 7B označava model sa oko sedam milijardi parametara, a 0.5B sa oko pola milijarde. Broj parametara je gruba mera kapaciteta modela – više parametara po pravilu znači šire naučeno znanje i bolju sposobnost uopštavanja (pa i veću robusnost na loše formulisane upite), ali i srazmerno veće zahteve za memorijom i dužim izvršavanjem. Namerno je korišćen širok raspon veličina kako bi se videlo da li i koliko veličina modela utiče na prostor za optimizaciju (glava 6).

5.2 Model sudija

Kao sudija zasnovan na LLM-u (engl. *LLM-as-a-judge*) korišćen je model `qwen2.5:7b-instruct`, i to uvek isti, nezavisno od modela koji generiše odgovore, kako bi sudija bio stabilan merni instrument. Sudija je referentnog tipa i uvek dobija tri stvari: opis zadatka (uvek čist, referentni upit, nezavisno od toga koja se varijanta ocenjuje), referentni odgovor (odgovor modela na taj čist upit) i,

konačno, odgovor koji se ocenjuje. Zatim procenjuje koliko se ocenjivani odgovor poklapa sa referencom po tačnosti i potpunosti, oslanjajući se na suštinu, a ne na formulaciju. Bitno je da sudija ne vidi sam upit-varijantu koja je proizvela odgovor, već samo njen odgovor; time je za svaki odgovor zadatak isti, pa su ocene svih varijanti međusobno uporedive.

Ovakav izbor zadatka najlakše je razumeti na primeru. Neka je čist upit *„Are avocados considered fruits or vegetables?"*, a referentni odgovor – dobijen upravo na tom upitu – kaže da je avokado voće, specifičnije bobičasto voće. Isti zadatak potom rešava i zašumljena verzija upita (*„ok, silly question but bear with me...settle a debate, avocados, are those a fruit or a vegetable?...thanks!"*) i verzija koju proizvede optimizator. Kada sudija ocenjuje odgovor na zašumljeni, a zatim odgovor na optimizovani upit, u oba slučaja kao zadatak dobija *isti* čist upit i *istu* referencu – menja se samo odgovor koji se ocenjuje.

Da se sudiji umesto čistog upita prosledi baš ona (zašumljena ili optimizovana) verzija koja je proizvela odgovor, poređenje bi izgubilo smisao iz dva razloga. Prvo, svaka varijanta imala bi svoj zadatak, pa ocene ne bi bile na istoj skali i oporavak se ne bi mogao pošteno izmeriti. Drugo, čitajući loš i dvosmislen upit, sudija bi lako opravdao i mlak odgovor („pa i pitanje je bilo nejasno"), umesto da odgovor meri prema stvarnoj nameri, koja je najjasnije izražena u čistom upitu. Zato je čist upit zajednički etalon: istovremeno polazište za referentni odgovor i zadatak prema kome se ocenjuju sve varijante.

Prvobitno korišćena skala od 1 do 5 dovela je do izraženog grupisanja ocena (u polaznom eksperimentu sa modelom 7B čak 74% odgovora dobilo je ocenu 4), pa je uvedena skala od 1 do 10 sa jasno opisanih pet nivoa, koja sudiji daje više prostora za razlikovanje.

Skala je namerno celobrojna, a ne neposredno kontinualna ocena iz [0,1]: jezički modeli nepouzdano dodeljuju fine decimalne vrednosti (sklone su grupisanju i nekonzistentne su između poziva), dok mali diskretan skup nivoa sa jasno opisanim značenjem svakog opsega daje stabilnije i ponovljivije procene. Dobijena celobrojna ocena se tek naknadno linearno preslikava na [0,1] obrascem $(\text{ocena} - 1)/9$, radi lakšeg poređenja sa merama zasnovanim na vektorskim reprezentacijama. Pet nivoa odgovara opsezima: 1–2 (pogrešno ili nebitno), 3–4 (uglavnom netačno, ali na temu), 5–6 (delimično tačno), 7–8 (tačno uz sitne propuste) i 9–10 (potpuno tačno i kompletno).

Konkretno, za zadatak *„What is the sum of the first five prime numbers?"* i

referentni odgovor *„The first five primes are 2, 3, 5, 7, 11; their sum is 28."*, odgovor *„2 + 3 + 5 + 7 + 11 = 28."* suštinski se poklapa sa referencom, pa dobija ocenu iz najvišeg opsega (na primer 9, odnosno 0,89 posle normalizacije); odgovor *„The sum is 26."* daje pogrešan rezultat i pada u najniži opseg (ocena 2, odnosno 0,11). Sudija pri tome gleda isključivo suštinsku tačnost u odnosu na referencu, a ne stil ili formulaciju – zato dva različito sročena, ali tačna odgovora dobijaju sličnu ocenu.

5.3 Mere kvaliteta

Za svaki odgovor računaju se referentne mere u odnosu na referentni odgovor – BERTScore-F1 [22] i ROUGE-L [7] u opsegu [0,1] i BLEU [12, 14] u opsegu [0,100] – kao i ocena sudije na skali od 1 do 10. U svim slučajevima veća vrednost znači bliže referenci, odnosno bolji odgovor. Detaljan opis ovih mera i njihovih ograničenja dat je u odeljku 2.5. Referentne mere hvataju koliko se *formulacija* odgovora približila idealnoj, dok ocena sudije hvata koliko je odgovor *suštinski tačan*. Kombinovanjem obe vrste mera moguće je razdvojiti promene u formi od promena u tačnosti, što se pokazalo ključnim za tumačenje rezultata.

Glava 6

Rezultati

Pre prikaza rezultata korisno je unapred razdvojiti dve vrste oporavka koje se u nastavku prate. Referentne mere (BERTScore, BLEU, ROUGE-L) porede odgovor sa referentnim po formi i formulaciji, dok ocena sudije meri činjeničnu tačnost. Te dve ose ne moraju da se kreću zajedno: ako model i na zašumljen upit odgovara suštinski tačno, oporavak će se videti pre svega u formi, dok će ocena tačnosti ostati skoro nepromenjena. Zato mali oporavak ocene, uz osetan oporavak referentnih mera, nije nedostatak metoda nego očekivan ishod na zadacima sa određenim odgovorom - i, kako će se videti, jedan od centralnih nalaza ovog rada. Veći pomak u samoj tačnosti javlja se tek pod jakim šumom (odeljak 6.4).

6.1 Koliko zašumljivanje zaista šteti odgovoru?

Pre izgradnje optimizatora izmereno je koliko zašumljen upit uopšte kvari odgovor, jer to određuje koliko prostora optimizator ima za popravku. Ispostavilo se da instrukcioni model veličine 7B odgovara na zašumljene upite gotovo jednako dobro kao na čiste, pa je prostor za popravku tačnosti mali. Da bi se taj prostor proširio, odgovori su generisani sve manjim modelima iste (instrukcione) porodice, uz očekivanje da manji modeli budu osetljiviji na šum u upitu. Rezultati su prikazani u tabeli 6.1.

Tabela 6.1 otkriva zanimljiv raskorak između dve vrste mera. Referentne mere (BERTScore, BLEU, ROUGE-L) opadaju gotovo pravilno kako model postaje manji, pa BERTScore pada sa 0,50 na 0,37; forma odgovora se, dakle, sve više udaljava od one koju isti model daje na čist upit. Ocena sudije, međutim, taj pad ne prati. Ona se ne smanjuje sistematski sa veličinom modela, nego ostaje

Model	BERTScore-F1	BLEU	ROUGE-L	Ocena (1-10)
qwen2.5:7b-instruct	0,504	37,6	0,478	8,1
qwen2.5:3b-instruct	0,452	33,0	0,435	8,43
qwen2.5:1.5b-instruct	0,422	29,9	0,417	8,24
qwen2.5:0.5b-instruct	0,374	25,8	0,393	7,64
qwen2.5:3b-base	0,330	23,8	0,378	8,17

Tabela 6.1: Zašumljen odgovor u odnosu na referentni, po veličini instrukcionog modela; u poslednjem redu je i bazni model 3B radi neposredne uporedivosti sa istoimenim instrukcionim. Niže vrednosti znače da je zašumljen odgovor dalje od idealnog, tj. veći prostor za optimizaciju.

praktično nepromenjena: model od 3B (8,43) i model od 1.5B (8,24) dobijaju čak i višu ocenu od najvećeg, 7B modela (8,1), a osetniji pad javlja se tek kod najmanjeg, 0.5B modela (7,64). Iako je model smanjen za ceo red veličine, tačnost odgovora ostaje u uskom pojasu oko 8, bez jasnog trenda.

Ovaj raskorak je upravo suština nalaza: smanjenje modela (kao i zašumljivanje upita) menja *kako* model odgovara mnogo više nego *da li* je u pravu. Manji modeli formulišu odgovor drugačije - što referentne mere registruju kao udaljavanje od ciljne forme - ali suštinski i dalje pogađaju tačan sadržaj, pa ih sudija ocenjuje slično.

Poslednji red tabele daje i bazni model iste veličine (3B). Njegov zašumljen odgovor je i po referentnim merama najdalji od idealnog (BERTScore 0,330 naspram 0,452 kod istoimenog instrukcionog); ovo dodatno razrađujemo u odeljku 6.3. Bazni model uključen je samo u veličini 3B (videti odeljak 5.1), pa služi za poređenje sa istoimenim instrukcionim modelom pri fiksnoj veličini, a ne za praćenje trenda po veličinama.

6.2 Da li jači šum više kvari odgovor?

Provereno je i da li agresivniji stil zašumljivanja više pogoršava odgovor (a time ostavlja više prostora za popravku), na manjem uzorku od 40 upita (model fiksiran na 3B, ista referenca). Takav stil čuva svaku činjenicu, broj i entitet, kao i sam zadatak, ali zahtev zakopava u opširno ćaskanje, prepliće instrukciju sa podacima i dodaje lažna potpitanja i protivrečne signale o obimu. Prosečna ocena jedva se pomerila (sa 8,23 na 7,98): model je ignorisao šum i odgovorio na stvarni zadatak,

uz sitne propuste. Suština je ostala tačna.

Zaključak je da savremeni instrukcioni modeli već sami sprovode neku vrstu *implicitnog čišćenja* upita: i iz veoma zašumljenog upita samostalno rekonstruišu pravu nameru korisnika. Zato se na zadacima sa određenim odgovorom tačnost ne može bitno popraviti – ni smanjivanjem modela ni jačim (ali poštenim) zašumljivanjem. Iako na prvi pogled deluje obeshrabrujuće, ovo je **najvažniji nalaz polazne faze**: pošto kod instrukcionih modela nema mnogo prostora za popravku tačnosti, glavnu vrednost optimizacije treba tražiti drugde – u uštedi tokena i urednijoj strukturi upita, kao i kod baznih modela, gde struktura upita više dolazi do izražaja.

6.3 Uticaj optimizatora u zavisnosti od tipa modela

Zbog prethodnog nalaza očekivan je mali dobitak u tačnosti na instrukcionom modelu, pa je sprovedeno poređenje sa baznim modelom, gde struktura upita ima veći značaj. Isti optimizator, isti upiti i isti sudija primenjeni su na instrukcioni i bazni model veličine 3B; sve ostalo je, dakle, isto, a razlikuje se samo to da li je model instrukciono podešen ili nije. Zahvaljujući tome, svaka razlika u rezultatima može se pripisati upravo instrukcionom podešavanju. Rezultati su dati u tabeli 6.2, gde je oporavak razlika između vrednosti za optimizovan i za zašumljen odgovor.

Mera	Instrukcioni 3B	Bazni 3B
BERTScore-F1	0,452 → 0,488 (+0,036)	0,330 → 0,409 (+**0,079**)
BLEU	33,0 → 37,2 (+4,1)	23,8 → 30,6 (+**6,8**)
ROUGE-L	0,435 → 0,473 (+0,038)	0,378 → 0,436 (+0,058)
Ocena (1-10)	8,43 → 8,51 (+0,08)	8,17 → 8,17 (+0,00)

Tabela 6.2: Oporavak na modelu 3B: instrukcioni naspram baznog. Za svaku meru dat je par (zašumljen → optimizovan odgovor), a u zagradi oporavak. Veći oporavak znači da je optimizator više približio odgovor idealnom.

Iz tabele 6.2 slede dva zapažanja. Prvo, bazni model je zaista osetljiviji na formulaciju: njegov zašumljen odgovor je znatno dalje od idealnog (BERTScore 0,330 naspram 0,452 kod instrukcionog), pa isti optimizator ostvaruje približno dvostruko veći oporavak na referentnim merama (+0,079 naspram +0,036 za BERTScore; +6,8 naspram +4,1 za BLEU). Drugim rečima, **ista optimizacija donosi korist**

na oba tipa modela, ali u različitom obimu - veću kod baznog, a manju kod instrukcionog, jer instrukcioni model svojom robusnošću sam nadoknađuje deo onoga što optimizator inače popravlja. Drugo, ocena sudije ostaje praktično nepromenjena kod oba modela (u granicama šuma): čak i bazni model, kada se ustali u odgovoru, na zadacima sa određenim odgovorom uglavnom je suštinski tačan, pa zašumljivanje menja formulaciju i strukturno preklapanje mnogo više nego činjeničnu tačnost.

Moglo bi se očekivati da optimizacija, kad već približava odgovor referentnom po formi, osetno podigne i ocenu tačnosti; ona to, međutim, ne čini - i upravo je taj izostanak jedan od ključnih nalaza. Zato vrednost optimizacije pod blagim šumom ne treba tražiti u skoku tačnosti, već u uštedi tokena i urednijoj strukturi upita, kao i u osetno većem oporavku forme kod baznog modela. Jasan porast same tačnosti javlja se tek pod jakim šumom, o čemu govori naredni odeljak.

6.4 Jačina degradacije: blag naspram jakog šuma

Korisnički upiti se veoma razlikuju po količini šuma: neki su kratki i relativno uredni, a drugi vrlo opširni i razgovorni - sa pozdravima, pričama o sebi, digresijama, više puta ponovljenim istim zahtevom i ljubaznim porukama na kraju. Oba tipa su realistična. Blago zašumljeni upiti iz prethodnih eksperimenata predstavljaju uredniji kraj tog raspona, pa optimizator na njima ima malo suvišnog teksta za uklanjanje (oko 4% tokena) i mali prostor za oporavak. Da bi se optimizator izmerio i na opširnijem kraju raspona, napravljena je zašumljenija verzija istih 500 upita: svaki blago zašumljen upit dodatno je obavijen suvišnim, ali *uklonjivim* tekstom, i to tako da svaka činjenica i svaki podatak ostanu sačuvani. Čist upit, koji služi kao referenca pri merenju oporavka, pri tome ostaje nepromenjen - duži postaje samo ulaz. Prosečna dužina raste sa oko 29 na oko 66 reči. Za ovaj režim optimizator je proširen korakom koji uklanja upravo taj razgovorni okvir (pozdrave, izvinjenja, priče o sebi i poruke na kraju), a zadržava sve što pripada samom zadatku; na blagom skupu ovaj korak ništa ne menja, jer takvi upiti i nemaju takav okvir.

Isti optimizator, isti modeli i isti sudija primenjeni su na oba režima; jedina promena je koliko je ulaz zašumljen. Tabela 6.3 sažima oporavak po režimu.

Režim	Ušteda tokena	Oporavak instr. (BERT / ocena)	Oporavak bazni (BERT / ocena)
Blag	$\sim 4\%$	$+0{,}036$ / $+0{,}08$	$+0{,}079$ / $+0{,}00$
Jak	$\sim \mathbf{54\%}$	$\mathbf{+0{,}095}$ / $\mathbf{+0{,}17}$	$\mathbf{+0{,}115}$ / $\mathbf{+0{,}27}$

Tabela 6.3: Uticaj jačine šuma na uštedu tokena i oporavak (model 3B). Oporavak je promena posle optimizacije (vrednost za optimizovan minus za zašumljen odgovor), prikazan za BERTScore-F1 i ocenu sudije (1-10), posebno za instrukcioni i bazni model. Veće vrednosti znače veći oporavak.

Pod jakim šumom menjaju se dve stvari. Prvo, efikasnost naglo raste - ušteda tokena skače sa oko 4% na oko 54% (detaljna analiza dužina i uštede data je u odeljku 6.5). Drugo, i važnije, sada se pomera i ocena sudije: oporavak tačnosti je jasno pozitivan na oba modela (+0,17 za instrukcioni, +0,27 za bazni), dok je pod blagim šumom bio zanemarljiv.

Razlog je taj što jači, ali i dalje veran šum udaljava odgovor na zašumljen upit od odgovora na čist upit više nego blag šum: BERTScore zašumljenog odgovora pada sa 0,452 na 0,362 kod instrukcionog, odnosno sa 0,330 na 0,267 kod baznog modela. Optimizator tako dobija stvaran prostor za oporavak – i ostvaruje ga, po svim merama, najviše kod baznog modela, koji je i najosetljiviji na formulaciju. Ovo je najjasniji izraz teze rada: na opširnim korisničkim upitima optimizacija istovremeno donosi veliku uštedu tokena i merljiv porast kvaliteta odgovora, a *korist je najveća upravo tamo gde je model najmanje robustan.*

6.5 Ušteda tokena

Optimizator, nezavisno od promene tačnosti, merljivo skraćuje upit. Ušteda se meri tokenizatorom samog generativnog modela (Qwen2.5), jer je broj tokena stvarna mera efikasnosti - broj reči i karaktera su samo grube aproksimacije. Pošto su izlazni tokeni (sam odgovor) obično skuplji od ulaznih, korisno je odvojeno posmatrati koliko optimizacija štedi na *ulazu* i da li pri tome menja dužinu *izlaza.*

Ulazni tokeni

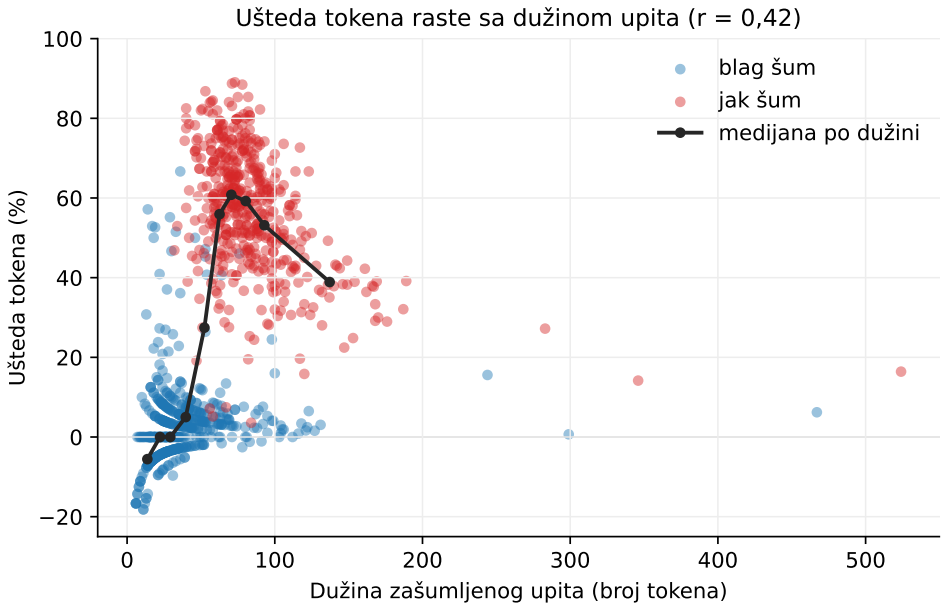
Na celom skupu od 500 blago zašumljenih upita prosečan broj tokena smanjen je sa 38,1 na 36,3, odnosno za 4,6% (poređenja radi, 6,1% po broju reči i 4,7% po

broju karaktera), i to bez gubitka informacija, jer su svi podaci zaštićeni tokom optimizacije.

Da bi se ova ušteda pravilno protumačila, korisno ju je uporediti sa teorijskim maksimumom. Prirodna gornja granica sažimanja je dužina samog čistog (optimalnog) upita, jer ispod nje optimizator ne može ići a da ne izgubi sadržaj. Čist upit ima u proseku 30,8 tokena, pa je blago zašumljen upit (38,1 tokena) svega oko 1,24 puta duži od njega. Kada bi optimizator zašumljen upit sveo tačno na dužinu čistog, ušteda bi iznosila oko 19%; ostvarenih 4,6% znači da je uklonjena približno četvrtina ionako male količine viška. Ušteda je na ovom skupu skromna prosto zato što blago zašumljeni upiti već jesu prilično sažeti - nema mnogo suvišnog teksta za uklanjanje.

Pod jakim šumom (odeljak 6.4) slika je sasvim drugačija. Prosečan broj tokena pada sa 83,6 na 38,4, odnosno za oko 54% (oko 55% po broju reči i 52% po broju karaktera): optimizator uklanja većinu šuma i vraća upit koji je po dužini blizak čistom, a pri tome se nijedan od 500 upita nije udaljio od čistog, pa nema gubitka podataka.

Odnos dužine upita i uštede vidi se i na nivou pojedinačnih primera (slika 6.1). Kod blago zašumljenih (kratkih) upita ušteda je mala i grupisana oko nule, dok kod jako zašumljenih (dugih) skače na 50–75%; ukupna korelacija dužine i procenta uštede je pozitivna (Pearsonov koeficijent korelacije $r = 0{,}42$). Taj koeficijent meri koliko su dve veličine linearno povezane, na skali od -1 do 1: vrednost 0 znači da veze nema, a što je bliža jedinici, veza je jača; $r = 0{,}42$ je stoga umerena pozitivna veza - duži upiti po pravilu donose veću uštedu, ali ne uvek. U poređenju sa teorijskim maksimumom, jako zašumljen upit je u proseku 2,72 puta duži od čistog, pa je gornja granica uštede oko 63%; ostvarenih 54% znači da optimizator ukloni oko 85% šuma i time upit po dužini gotovo izjednači sa čistim (38,4 naspram 30,8 tokena u proseku). Zanimljiv je i desni rep krive: nekoliko najdužih upita ima malu uštedu jer su dugi upravo zbog obimnog zaštićenog sadržaja (kod, podaci), koji se po pravilu ne sme uklanjati.



Slika 6.1: Ušteda tokena u odnosu na dužinu zašumljenog upita, za sve upite iz blagog (plavo) i jakog (crveno) režima. Crna linija je medijana uštede po opsezima dužine. Duži upiti po pravilu donose veću uštedu; najduži upiti sa malom uštedom dugi su zbog zaštićenog sadržaja koji se ne uklanja.

Izlazni tokeni

Dosadašnja ušteda odnosi se na *ulaz* - upit koji se šalje modelu. Ostaje pitanje da li optimizacija menja i dužinu *izlaza*, tj. samog odgovora. Tabela 6.4 daje prosečan broj izlaznih tokena po varijanti i režimu, mereno istim tokenizatorom.

Šum	Model	Zašumljen	Optimizovan	Referentni
Blag	instrukcioni	356	355	339
Blag	bazni	311	270	258
Jak	instrukcioni	340	365	339
Jak	bazni	258	267	258

Tabela 6.4: Prosečan broj izlaznih tokena odgovora po varijanti upita (zašumljen, optimizovan, referentni) i režimu šuma, na celom skupu od 500 primera.

Ključno je zapaziti da dužina odgovora zavisi pre svega od samog zadatka, a ne od količine šuma u upitu. Optimizator uklanja suvišan tekst iz ulaza, ali ne menja ono što se traži, pa model na optimizovan i na zašumljen upit odgovara u suštini istom sadržinom, slične dužine. Zbog toga su razlike u dužini izlaza male.

Kao što se vidi u tabeli, razlika u broju izlaznih tokena između zašumljenih, optimizovanih i referentnih odgovora nije značajna ni u jednom režimu - dužina odgovora niti osetno raste niti se osetno smanjuje. Ušteda se, dakle, ostvaruje na ulazu.

6.6 Koliko agresivno optimizator treba da menja upit?

Okvir merenja oporavka nagrađuje optimizator koji odgovor dovede blizu odgovora na čist upit, pa nije unapred jasno koliko *agresivno* optimizator treba da menja upit: da li je bolje samo ukloniti očigledan šum, ili upit dodatno sažimati i preformulisati. Da bi se to proverilo, ista procedura sa *blago* zašumljenim skupom (isti kao u odeljku 6.4 pre uvođenja jakog šuma; oba modela veličine 3B, uz nepromenjene ostale postavke) ponovljena je sa tri sve agresivnija podešavanja optimizatora, pa je oporavak upoređen. Tri podešavanja razlikuju se po tome koliko duboko zadiru u tekst:

- *Lako čišćenje* (glavna konfiguracija) uklanja pozdrave, uzrečice, ponavljanja i očigledan šum i blago normalizuje interpunkciju i velika slova, ali inače čuva korisnikovu formulaciju.
- *Agresivno čišćenje* dodatno jače uklanja uvodne fraze iz učtivosti, prevodi neformalne izraze u standardne, deduplikuje na nivou klauza (a ne samo celih rečenica) i premešta instrukcije; ušteda raste na oko 13% reči – skoro dvostruko više nego kod lakog čišćenja (oko 6%) – jer je tekst osetno više prepravljen.
- *Telegrafska kompresija* (naziv po telegrafskom stilu starih telegrama, u kojima su se izbacivale sve reči bez kojih poruka ostaje razumljiva) povrh svega izbacuje i reči male informativnosti (članove, glagol „biti” u ulozi spone, pomoćne glagole), pa broj tokena spušta za oko 31%, po cenu gramatičnosti.

Razlika se najlakše vidi na primeru. Za zašumljen upit *„hey there! i was just wondering, could you please summarize the article below in three sentences? like just three sentences would be perfect, thanks!”* tri podešavanja daju redom:

Lako: Summarize the article below in three sentences.
Agresivno: Summarize the article in three sentences.
Telegrafski: Summarize article three sentences.

Sva tri zadržavaju suštinu zahteva, ali telegrafska verzija više ne liči na prirodan jezik.

Podešavanje	Model	ΔBERT	ΔBLEU	Δocena
Lako čišćenje (glavno)	instr.	**+0,036**	**+4,1**	+0,08
	bazni	**+0,079**	**+6,8**	+0,00
Agresivno čišćenje	instr.	+0,010	+1,8	−0,17
	bazni	+0,057	+4,7	−0,23
Telegrafska kompresija	instr.	−0,063	−6,7	−0,37
(∼ 31% uštede)	bazni	−0,045	−4,6	−0,52

Tabela 6.5: Oporavak (vrednost za optimizovan minus za zašumljen odgovor) za tri podešavanja optimizatora. Što je optimizator agresivniji, oporavak je manji, a ocena tačnosti niža.

Trend je dosledan na oba modela: što optimizator jače prepravlja upit, oporavak je manji. Agresivno čišćenje već osetno smanji dobitak i obara ocenu, a telegrafska kompresija oporavak učini negativnim – optimizovan odgovor završi *dalje* od odgovora na čist upit nego netaknut zašumljeni. Razlozi su dvojaki: agresivni prolazi povremeno uklone sadržajnu reč uz šum ili premeste zahtev. Na primer, iz zašumljenog upita *„random thought but jaws, the movie, is that based on a true story?"* agresivno čišćenje prepozna *„random thought but"* kao šum i uz njega izbaci i naziv filma *Jaws*, pa se izgubi sam predmet pitanja (koji film). Praktičan zaključak je da na zadacima sa određenim odgovorom optimizator treba da bude konzervativan: da ukloni šum, sačuva sadržaj i formulaciju, a uštedu tokena tretira kao dodatnu korist, a ne kao cilj koji se maksimizuje. Ovo opravdava izbor lakog čišćenja kao glavne konfiguracije optimizatora.

6.7 Doprinos pojedinačnih tehnika

Prethodna analiza je gledala *koliko duboko* optimizator treba da menja upit. Zasebno pitanje jeste *koliko svaki od tri koraka* – čišćenje, uklanjanje ponavljanja i uređivanje redosleda – sam za sebe doprinosi, i šta se dobija njihovim kombinovanjem. Da bi se to razdvojilo, koraci su uključivani i isključivani u svih osam

kombinacija (zaštita podataka maskiranjem je pri tome uvek uključena), a ceo postupak je sproveden na skupu sa jakim šumom – dakle tamo gde efekat optimizacije najviše dolazi do izražaja – i to na *oba* modela veličine 3B, baznom i instrukcionom. Radi preglednosti najpre se prikazuje bazni model (tabela 6.6), a zatim instrukcioni (tabela 6.7). Oporavak je i ovde meren u odnosu na netaknut zašumljen upit (na baznom modelu: BERTScore 0,267, BLEU 16,5, ROUGE-L 0,309, ocena 7,88).

Uključeni koraci	ΔBERT	ΔBLEU	ΔROUGE-L	Δocena
nijedan	+0,017	+0,0	+0,004	+0,06
čišćenje	**+0,115**	**+10,1**	**+0,096**	+0,27
ponavljanja	+0,018	+0,0	+0,005	+0,06
redosled	+0,018	+0,0	+0,004	+0,07
čišćenje + ponavljanja	**+0,120**	**+10,8**	**+0,101**	+0,26
čišćenje + redosled	+0,111	+9,9	+0,094	+0,28
ponavljanja + redosled	+0,018	+0,1	+0,005	+0,07
sva tri (glavno)	+0,115	+10,6	+0,099	+0,27

Tabela 6.6: Doprinos pojedinačnih koraka oporavku. Oporavak je razlika u odnosu na netaknut zašumljen upit; maskiranje podataka je uvek uključeno. Čišćenje nosi najveći deo oporavka, a dodavanje uklanjanja ponavljanja donosi mali, ali dosledan plus.

Slika je jasna: *čišćenje nosi gotovo sav oporavak*. Bez njega – bilo da je uključeno samo uklanjanje ponavljanja, samo uređivanje redosleda ili oba – rezultat ostaje na nivou netaknutog zašumljenog upita. Dodavanje uklanjanja ponavljanja na čišćenje donosi mali, ali dosledan dobitak (BERTScore sa +0,115 na +0,120, BLEU sa +10,1 na +10,8), dok uređivanje redosleda oporavak praktično ne menja. Sitna razlika između konfiguracije sa sva tri koraka i one sa samo čišćenjem i uklanjanjem ponavljanja (BERTScore −0,005, BLEU −0,2, uz +0,01 na oceni) u granicama je šuma: uređivanje redosleda menja samo raspored rečenica na malom broju upita, pa ni ne može bitno da pomeri ove mere.

Da bi se proverilo da li isti obrazac važi i van baznog modela, ista ablacija ponovljena je i na *instrukcionom* modelu veličine 3B (i dalje pri jakom šumu; oporavak u odnosu na netaknut zašumljen upit, sa vrednostima za zašumljen odgovor BERTScore 0,362, BLEU 24,0, ROUGE-L 0,357 i ocenom 8,30). Rezultati su dati u tabeli 6.7.

Obrazac je istovetan kao kod baznog modela: čišćenje nosi gotovo sav oporavak, dodavanje uklanjanja ponavljanja donosi mali, ali dosledan plus (BERTScore

Uključeni koraci	ΔBERT	ΔBLEU	ΔROUGE-L	Δocena
nijedan	+0,013	+1,4	+0,013	+0,06
čišćenje	**+0,089**	**+8,6**	**+0,077**	+0,16
ponavljanja	+0,012	+1,3	+0,012	+0,06
redosled	+0,012	+1,3	+0,011	+0,05
čišćenje + ponavljanja	**+0,095**	**+9,3**	**+0,084**	+0,18
čišćenje + redosled	+0,088	+8,8	+0,077	+0,15
ponavljanja + redosled	+0,012	+1,2	+0,011	+0,04
sva tri (glavno)	+0,095	+9,5	+0,083	+0,17

Tabela 6.7: Doprinos pojedinačnih koraka oporavku na *instrukcionom* modelu 3B (jak šum). Oporavak je razlika u odnosu na netaknut zašumljen upit; maskiranje podataka je uvek uključeno. Obrazac je istovetan kao kod baznog modela.

sa +0,089 na +0,095, BLEU sa +8,6 na +9,3), a uređivanje redosleda ove mere praktično ne menja. Vrednosti za sva tri koraka poklapaju se sa ranije prikazanim oporavkom instrukcionog modela pri jakom šumu (tabela 6.3). Zaključak o podeli uloga među koracima, dakle, ne zavisi od toga da li je model instrukciono podešen.

Ovo, međutim, ne znači da su preostala dva koraka suvišna, već da mera oporavka opisuje samo jednu stranu kvaliteta. Oporavak poredi pojedinačan odgovor sa odgovorom na čist upit, i to merama koje se oslanjaju na preklapanje reči (BERTScore, BLEU, ROUGE); takve mere su gotovo neosetljive na redosled rečenica. Zato uređivanje redosleda po prirodi stvari i ne može da pomeri ove brojke – ono ne dira *koje* su reči u upitu, nego samo njihov raspored. Slično tome, uklanjanje ponavljanja ima efekta samo kada u upitu zaista ima ponavljanja.

Svaki korak zapravo popravlja drugu stranu kvaliteta, pa se i njegov učinak najbolje vidi drugom merom (Tabela 6.8). Čišćenje diže tačnost odgovora i tu je njegov doprinos očigledan. Uklanjanje ponavljanja skraćuje upit kada ima šta da se skrati, pa se pre vidi kroz uštedu tokena (odeljak 6.9) nego kroz tačnost. Uređivanje redosleda sređuje strukturu upita i doprinosi tome da model dosledno odgovara na iste zahteve formulisane na razne načine, što se meri zasebno (odeljak 6.8).

Drugim rečima, tri koraka se dopunjuju: čišćenje diže tačnost, uklanjanje ponavljanja skraćuje upit kada ima šta da se skrati, a uređivanje redosleda sređuje strukturu i pomaže ujednačenosti odgovora. To što se baš na zadacima sa određenim odgovorom najviše ističe čišćenje očekivano je – upravo zato što su druga dva koraka usmerena na osobine koje ova, jednoodgovorna mera ne hvata u potpunosti.

Korak	Uloga u postupku	Kada najviše vredi
čišćenje	uklanja šum i suvišne reči	na svakom konverzacijskom, zašumljenom upitu – nosi najveći deo oporavka
uklanjanje ponavljanja	sažima ponovljene ili preformulisane zahteve	kada korisnik istu stvar traži više puta; tada skraćuje upit i uklanja protivrečnosti
uređivanje redosleda	dovodi upit u oblik kontekst → zadatak → ograničenja	na strukturno neurednim upitima i za ujednačenost odgovora

Tabela 6.8: Uloga svakog koraka i uslovi u kojima najviše doprinosi. Koraci se dopunjuju, a ne ponavljaju isti posao – usmereni su na različite osobine upita, pa se njihov učinak vidi različitim merama.

6.8 Robusnost na parafraze istog zahteva

Sve dosadašnje mere procenjuju kvalitet jednog odgovora. Zasebno, ali podjednako važno pitanje jeste koliko je model *dosledan* kada isti zahtev stigne formulisan na različite načine – a upravo to optimizator treba da stabilizuje. Da bi se to izmerilo, uzeto je 100 zadataka i za svaki napravljeno 5 različitih varijanti: isti zahtev obavijen različitim šumom, tako da su jezgro instrukcije i svaki podatak identični, a menjaju se opširnost i redosled propratnog razgovora. Porede se dva režima: *sirovi* (svaka varijanta se prosleđuje modelu kao takva) i *optimizovani* (svaka varijanta se prvo optimizuje). Kvalitet je uobičajeni oporavak (BERTScore-F1 odgovora u odnosu na odgovor na čist upit). Doslednost se sažima na dva načina po zadatku: *varijabilnost* (engl. *spread*, standardna devijacija kvaliteta preko 5 varijanti; manje je robusnije) i *saglasnost* (prosečan uzajamni BERTScore između 5 odgovora; više je robusnije). Prati se i kvalitet u *najgorem slučaju* (minimum preko varijanti), jer robustan optimizator treba najviše da podigne baš one nesrećne formulacije.

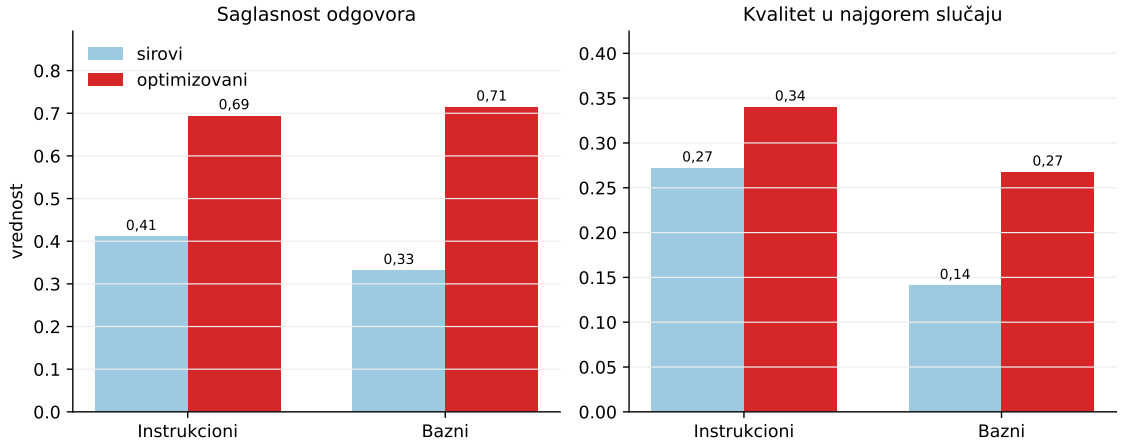
Da ideja bude konkretnija: isti zahtev *„list the first five prime numbers"* može stići kao *„hey, quick one - first five primes?"*, kao *„i need the first 5 prime numbers pls"*, itd. U sirovom režimu, gde model dobija svaku varijantu neizmenjenu, svaka od tih formulacija može dati pomalo drugačiji odgovor, pa je slaganje između njih nisko. Pošto optimizator sve varijante svodi na skoro isti čist upit, odgovori posle optimizacije postaju gotovo identični – otuda i nagli skok saglasnosti u tabeli 6.9.

Efekat je velik i na svim merama pokazuje u istom smeru (tabela 6.9). Optimizacija otprilike prepolovi varijabilnost i, najupečatljivije, podigne saglasnost

Mera (smer)	Instrukcioni (sirovi → opt.)	Bazni (sirovi → opt.)
Kvalitet, prosek (↑)	0,355 → 0,387 (+0,032)	0,275 → 0,333 (+**0,058**)
Najgori slučaj, prosek (↑)	0,272 → 0,340 (+0,068)	0,141 → 0,267 (+**0,127**)
Varijabilnost, std (↓)	0,058 → 0,035 (−0,023)	0,092 → 0,048 (−**0,044**)
Saglasnost odgovora (↑)	0,411 → 0,694 (+0,283)	0,331 → 0,714 (+**0,383**)

Tabela 6.9: Robusnost preko 5 parafraza istog zahteva (100 zadataka), pre (sirovi) i posle (optimizovani) optimizacije, za instrukcioni i bazni model 3B.

odgovora sa oko 0,41 na 0,69 (instrukcioni), odnosno sa 0,33 na 0,71 (bazni): posle optimizacije model daje gotovo isti odgovor bez obzira na to kako je zahtev formulisan, jer optimizator svodi zašumljene varijante na skoro isti čist upit. Kvalitet u najgorem slučaju raste najviše (kod baznog modela se skoro udvostruči, 0,141 → 0,267), što je i praktična poenta – optimizator štiti od nesrećnih formulacija. Kao i u svim ranijim rezultatima, dobitak je veći kod baznog modela osetljivog na formulaciju nego kod instrukcionog koji se sam čisti. Doslednost je time konkretna, merljiva korist optimizacije i onda kada se tačnost pojedinačnog odgovora jedva menja.



Slika 6.2: Robusnost na parafraze pre (sirovi) i posle (optimizovani) optimizacije, za instrukcioni i bazni model 3B. Levo: saglasnost odgovora (prosečna uzajamna sličnost); desno: kvalitet u najgorem slučaju. Optimizacija podiže obe mere kod oba modela, najviše kod baznog.

6.9 Strukturne i efikasnosne metrike upita

Prethodne mere prolaze kroz odgovor modela. Korisno je izmeriti i šta optimizator radi sa *samim upitom*, bez ijednog generisanja. Tabela 6.10 daje tri mere na nivou upita, preko celog skupa od 500 primera, za blag i jak šum.

Mera	Blag šum	Jak šum
Ušteda tokena	4,6%	**54,0**%
Semantička bliskost čistom (kosinus)	0,952 → 0,954	0,892 → **0,951**
Leksička bliskost čistom (unigram-F1)	0,788 → 0,806	0,431 → **0,776**
Očuvanje zaštićenih podataka	**100**%	**100**%

Tabela 6.10: Strukturne i efikasnosne mere na nivou upita (ceo skup, 500 primera). Kod mera bliskosti prikazana je vrednost pre i posle optimizacije (šum → opt.).

Semantička bliskost računa se kao kosinus sličnost vektorskih reprezentacija rečenica, pa hvata podudarnost značenja i kada se reči razlikuju. *Leksička* bliskost meri se unigram-F1 merom: posmatraju se skupovi pojedinačnih reči (unigrama) dva upita, pa se računa harmonijska sredina preciznosti (udeo reči optimizovanog upita koje postoje i u čistom) i odziva (udeo reči čistog upita koje su pokrivene); vrednost je u [0,1] i meri doslovno preklapanje reči, nezavisno od njihovog redosleda. Obe mere strukturno pokazuju koliko optimizator vraća zašumljen ulaz ka čistom obliku: neznatno kod blagog šuma , a snažno kod jakog šuma (unigram-F1 sa 0,431 na 0,776, kosinus sa 0,892 na 0,951). *Očuvanje podataka* je udeo upita kod kojih svaki zaštićeni deo (kod, brojevi, navodnici, liste, jednačine, URL) preživi doslovno u izlazu; na oba skupa iznosi 100%. Do te vrednosti nismo stigli odmah – rad na njoj otkrio je stvarnu grešku u koraku zaštite, koju vredi zabeležiti. Zaštićeni delovi mogu biti *ugnježdeni* jedan unutar drugog: na primer string pod navodnicima unutar poziva funkcije, JSON unutar liste u uglastim zagradama, ili string unutar bloka koda. U takvim slučajevima zaštita napravi ugnježdene oznake, a vraćanje u jednom prolazu razreši samo spoljašnju oznaku i ostavi unutrašnju nerazrešenu. Konkretno, u pozivu `count_words('Loving living life')` bi umesto samog stringa u izlazu ostala nerazrešena oznaka, pa bi rezultat izgledao kao `count_words(<id>)` – čime se gubio upravo argument `'Loving living life'`, tekst čije reči treba prebrojati, a bez njega zadatak nema smisla. Iterativno vraćanje (ponavljanje dok ne nestane nijedna oznaka) ispravlja sve takve slučajeve i pruža upravo garanciju koju korak zaštite i treba da ima.

6.10 Statistička značajnost oporavka

Brojevi u tabelama 6.2 i 6.3 su proseci oporavka preko svih 500 upita. Sam prosek, međutim, ne govori da li je efekat stvaran ili je slučajno ispao pozitivan baš na ovom skupu upita. Zato je uz svaki prosek dodata mera pouzdanosti, i to na dva načina.

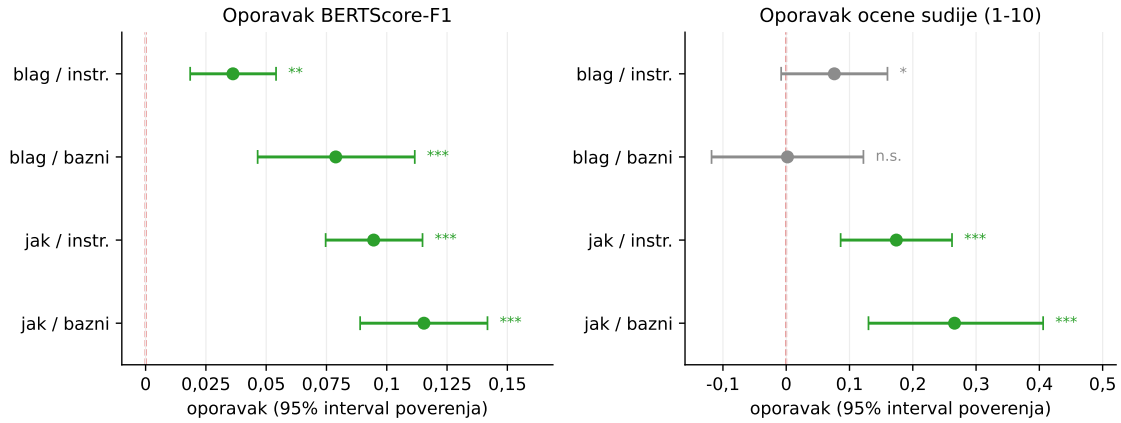
Prvi je *interval poverenja*: raspon u kome se, sa 95% sigurnosti, nalazi pravi prosečni oporavak. Ključno pravilo je jednostavno – ako taj raspon *ne dodiruje nulu*, efekat je stvaran (nula bi značila da optimizacija ništa ne menja); ako raspon *prelazi preko nule*, efekat se ne može pouzdano tvrditi. Interval je dobijen postupkom uzorkovanja sa ponavljanjem (engl. *bootstrap*): iz izmerenih oporavaka se mnogo puta (10 000) nasumično izvuče novih 500 vrednosti (uz vraćanje) i izračuna njihov prosek, pa se posmatra raspon tako dobijenih proseka.

Drugi je *p-vrednost*: verovatnoća da bi se ovako jak (ili jači) oporavak pojavio i kada efekta zapravo nema. Mala p-vrednost (po uobičajenom dogovoru manja od 0,05) znači da je malo verovatno da je rezultat slučajan, pa se efekat smatra *statistički značajnim*. p-vrednost je izračunata Vilkoksonovim testom znaka i ranga (engl. *Wilcoxon signed-rank*), koji za svaki upit poredi ista dva odgovora (na zašumljen i na optimizovan upit) i ne pretpostavlja poseban oblik raspodele podataka. Obe provere sprovedene su za svaku meru i svaki režim, nad celim skupom ($n = 500$).

Režim	Model	Oporavak BERTScore (95% interval poverenja)	Oporavak ocene (95% interval poverenja)
Blag	instrukcioni	+0,036 [+0,018; +0,054]	+0,08 [−0,01; +0,16]
Blag	bazni	+0,079 [+0,046; +0,112]	+0,00 [−0,12; +0,12]
Jak	instrukcioni	+0,095 [+0,075; +0,115]	+0,17 [+0,09; +0,26]
Jak	bazni	+0,115 [+0,089; +0,142]	+0,27 [+0,13; +0,41]

Tabela 6.11: Srednji oporavak sa 95% intervalom poverenja (bootstrap) za BERTScore-F1 i ocenu sudije, po režimu šuma i tipu modela. Interval koji ne obuhvata nulu odgovara značajnom oporavku.

Nalazi su sledeći (slika 6.3). Oporavak na referentnim merama – BERTScore, BLEU i ROUGE-L – statistički je značajan u *sva četiri* režima ($p < 0{,}05$ u svakoj kombinaciji mere i režima, u većini slučajeva $p < 0{,}01$ ili $p < 0{,}001$; jedini izuzetak od $p < 0{,}01$ je BLEU pod blagim šumom na instrukcionom modelu, gde je $p \approx 0{,}014$), pa interval poverenja ni u jednom ne obuhvata nulu. Drugim re-



Slika 6.3: Oporavak sa 95% intervalom poverenja za BERTScore-F1 (levo) i ocenu sudije (desno), po režimu šuma i tipu modela. Isprekidana linija označava nulu (nema efekta): intervali koji je ne dodiruju odgovaraju značajnom oporavku (zeleno), a oni koji je presecaju nisu značajni (sivo, *n.s.*). Zvezdice označavaju nivo značajnosti ($*$ za $p < 0{,}05$, $**$ za $p < 0{,}01$, $***$ za $p < 0{,}001$).

čima, približavanje odgovora idealnom – glavni empirijski nalaz rada – nije puka posledica izbora baš ovih upita, već stvaran efekat.

Oporavak ocene sudije (tačnost) značajan je pod jakim šumom na oba modela ($p < 0{,}001$) i granično kod instrukcionog modela pod blagim šumom ($p \approx 0{,}03$), ali *nije* značajan kod baznog modela pod blagim šumom (interval $[-0{,}12; +0{,}12]$ obuhvata nulu). To se poklapa sa ranijim nalazom: pod blagim šumom tačnost se jedva menja, a merljiv dobitak u tačnosti javlja se tek kada ulaz sadrži više uklonjivog šuma (opširniji upiti).

Konačno, razlika u oporavku između baznog i instrukcionog modela – koliko bazni model više dobija od optimizacije – značajna je pod blagim šumom na merama BERTScore i BLEU ($+0{,}043$, odnosno $+2{,}7$; $p < 0{,}05$), čime se statistički potkrepljuje tvrdnja da je korist od optimizacije *veća kod baznih nego kod instrukcionih* modela. Pod jakim šumom ta razlika prestaje da bude značajna, jer tada oba modela snažno dobijaju od čišćenja upita. Zbog većeg broja poređenja, najsitnije razlike (na primer doprinos koraka uređivanja redosleda) i dalje treba tumačiti kao granične.

6.11 Provera sudije modelom iz druge porodice

Sudija (`qwen2.5:7b-instruct`) pripada istoj porodici kao generativni modeli, pa bi u načelu mogao pristrasno da ocenjuje odgovore iz sopstvene porodice (engl. *self-preference bias*). Da bi se proverilo da zaključak o oporavku tačnosti ne zavisi od izbora sudije, isti odgovori na zašumljene i optimizovane upite (ceo skup od 500 pri jakom šumu) ponovo su ocenjeni sudijom iz *druge* porodice, `llama3.1:8b`, uz identičan upit i istu rubriku od 1 do 10.

Rezultati se slažu u onome što je najvažnije. Oba sudije ocenjuju odgovor na optimizovan upit više nego odgovor na zašumljen – dakle oba beleže pozitivan oporavak tačnosti, i oba su statistički značajna (Vilkoksonov test na oporavku ocena: $p \approx 0{,}0001$ za Qwen, $p \approx 0{,}006$ za Llamu, dakle oba ispod 0,01). Sudija iz porodice Qwen podiže prosečnu ocenu sa 8,30 na 8,48, a sudija iz porodice Llama sa 7,81 na 7,95 (isti oporavak instrukcionog modela pri jakom šumu prikazan je u tabeli 6.11).

Postoji samo jedna sistematska razlika: Llama sve odgovore ocenjuje strože, za oko pola poena niže. To je upravo pristrasnost na koju se sumnjalo – da sudija iz porodice Qwen blago favorizuje odgovore iz sopstvene porodice – i ona zaista postoji. Ali ta pristrasnost pomera samo *apsolutni nivo* ocene, i to podjednako kod zašumljenih i kod optimizovanih odgovora, pa se poništava čim se gleda *razlika* između njih, a upravo ta razlika jeste oporavak.

Zaključak je zato jasan: koji god sudija da se koristi, optimizacija pri jakom šumu podiže kvalitet odgovora za praktično isti iznos, a ocene sudija dobro se slažu i na nivou pojedinačnog odgovora (Spirmanova korelacija ranga 0,73; ta mera pokazuje u kojoj meri sudije slično poređaju odgovore po kvalitetu – to jest, da li je ono što je jednom sudiji bolje po pravilu bolje i drugom – pri čemu vrednost 1 znači savršeno slaganje poretka, a 0 nikakvo; za razliku od obične korelacije, gleda samo redosled ocena, a ne njihov apsolutni nivo). Drugim rečima, apsolutna ocena zavisi od izbora sudije, ali *zaključci rada ne zavise*.

Glava 7

Zaključak

U radu je predstavljen algoritamski sistem za automatsku optimizaciju i strukturiranje upita za velike jezičke modele, zasnovan na lingvističkim heuristikama, deduplikaciji preko vektorskih reprezentacija i grafovskom uređivanju redosleda, uz zaštitu podataka u upitu. Za evaluaciju je konstruisan namenski testni skup i uveden okvir merenja oporavka u odnosu na odgovor na čist upit.

Glavni nalaz je da moderni instrukcioni modeli sami implicitno čiste zašumljen upit. Zbog toga je na blago zašumljenim zadacima sa određenim odgovorom prostor za popravku tačnosti mali, što je potvrđeno na dva nezavisna načina – smanjivanjem modela i agresivnijim zašumljivanjem, pri čemu ni jedno ni drugo nije značajno oborilo tačnost odgovora na zašumljene upite. Taj nalaz, međutim, ne obesmišljava optimizaciju upita, već pomera težište pitanja. Kontrast sa baznim modelom pokazuje da ista optimizacija donosi korist na oba tipa modela, ali u različitom obimu: na baznom modelu, koji nema instrukciono podešavanje da samostalno nadoknadi loše formulisan upit, isti optimizator na istim upitima donosi približno dvostruko veći oporavak na referentnim merama. Drugim rečima, optimizacija upita najviše vredi upravo tamo gde je model najmanje robustan, pa bi bazni modeli mogli biti prirodnije okruženje za demonstraciju njenog efekta.

Razdvajanje referentnih mera od ocene sudije pokazalo se ključnim za tumačenje. Na blago zašumljenim upitima BERTScore, BLEU i ROUGE-L osetno rastu nakon optimizacije, dok ocena tačnosti ostaje gotovo nepromenjena – optimizacija tu pre svega približava *formu i strukturu* odgovora idealnoj, jer je sadržaj i pre optimizacije uglavnom bio tačan. Slika se, međutim, menja sa jačinom šuma: na opširnim, jako zašumljenim upitima (poglavlje 6.4) i ocena tačnosti jasno raste (+0,17 za instrukcioni, +0,27 za bazni model), a ušteda tokena skače sa oko 4%

na oko 54% bez gubitka informacija. Što upit sadrži više uklonjivog šuma, to je korist od optimizacije veća – i po efikasnosti i po tačnosti. Uz to, optimizacija znatno povećava doslednost: odgovori na različite parafraze istog zahteva postaju međusobno mnogo sličniji (saglasnost raste sa oko 0,41 na 0,69 kod instrukcionog i sa 0,33 na 0,71 kod baznog modela), uz podignut kvalitet u najgorem slučaju.

Ovi nalazi imaju i metodološku posledicu. Ako se efekat optimizacije upita meri samo tačnošću odgovora na instrukcionom modelu, taj efekat lako ostaje skriven. Poštenija slika dobija se kombinovanjem tri ugla: efikasnosti (ušteda tokena i uklanjanje ponavljanja), strukture odgovora (referentne mere sličnosti) i kontrasta baznog i instrukcionog modela.

Pouzdanost ovih nalaza je i formalno provedena. Glavni efekti oporavka praćeni su 95% bootstrap intervalima poverenja i parnim Vilkoksonovim testom (odeljak 6.10), koji pokazuju da približavanje odgovora idealnom nije slučajno svojstvo baš ovog skupa upita, već statistički značajan efekat. Dodatno, da bi se isključila mogućnost da nalaz o oporavku tačnosti potiče od pristrasnosti LLM sudije prema odgovorima iz sopstvene porodice modela, isti odgovori pri jakom šumu ponovo su ocenjeni sudijom iz druge porodice (odeljak 6.11); zaključak o postojanju i veličini oporavka ostaje isti, čime je nalaz dodatno potkrepljen.

Treba imati u vidu i ograničenja rada.

- **Sintetički testni skup.** Iako izveden iz stvarnih upita i konstruisan po strogoj specifikaciji koja čuva sadržaj, zašumljivanje je kontrolisano i ne mora u potpunosti odražavati raznolikost stvarnih korisničkih upita.

- **Vezanost heuristika za jezik i domen.** Lingvističke heuristike iz koraka čišćenja (odeljak 4.2) prepoznaju šum pomoću unapred zadatih obrazaca koji su pisani za engleski jezik i podešeni prema tipovima šuma zastupljenim u skupu BPO. Takav, na jezik i domen vezan optimizator dobro radi u uslovima ovog eksperimenta, ali se ne prenosi automatski na drugi jezik ili na upite bitno drugačijeg stila; za njih bi obrasce trebalo iznova prilagoditi ili ih zameniti pristupom koji se sam uči iz podataka.

- **Sudija iz iste porodice modela.** LLM sudija (`qwen2.5:7b-instruct`) pripada istoj porodici modela čiji se odgovori ocenjuju; poznato je da LLM sudije mogu pristrasno ocenjivati stil modela iz sopstvene porodice (engl. *self-preference bias*). Provera sudijom iz druge porodice (odeljak 6.11) po-

kazuje da ta pristrasnost postoji, ali je blaga i utiče samo na apsolutni nivo ocene, dok smer i veličina oporavka ostaju isti.

- **Ograničena statistička analiza.** Za glavne nalaze o oporavku prikazani su 95% intervali poverenja i sproveden je paran Vilkoksonov test (odeljak 6.10), ali nije primenjena korekcija za višestruko poređenje, pa najsitnije razlike (na primer doprinos uređivanja redosleda) treba tumačiti kao granične.
- **Obim ablacije komponenti.** Doprinos pojedinačnih tehnika (odeljak 6.7) ispitan je na baznom i na instrukcionom modelu 3B pri jakom šumu, gde se obrazac pokazao istovetnim; nije, međutim, provereno da li se isti obrazac održava i pod blagim šumom.

Dalji rad kreće se u nekoliko pravaca. Prvi je unapređenje samog optimizatora, koji je za sada napravljen u osnovnoj verziji: bolje lingvističke heuristike, preciznija deduplikacija, snažniji klasifikator za uređivanje redosleda i nove tehnike za sažimanje i strukturiranje (na primer eksplicitne sekcije zadatak/kontekst/ograničenja), i podešavanje parametara sa ciljem veće uštede tokena i većeg oporavka. Drugi pravac je dalje proširenje strukturnih i efikasnosnih mera uvedenih u ovom radu, na primer njihova analiza po tipu zadatka, kako bi se doprinos optimizacije preciznije opisao i onda kada tačnost ostaje visoka. Konačno, korisno bi bilo proširiti evaluaciju i na neki stvarni (a ne samo sintetički) skup upita, kako bi se nalazi potvrdili u realnijim uslovima.

Literatura

[1] Anthropic. Prompt engineering: long context tips. `https://docs.anthropic.com/en/docs/build-with-claude/prompt-engineering/long-context-tips`, 2024. Pristupljeno avgusta 2026.

[2] Tom B. Brown et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[3] Jiale Cheng, Xiao Liu, Kehan Zheng, Pei Ke, Hongning Wang, Yuxiao Dong, Jie Tang, and Minlie Huang. Black-box prompt optimization: Aligning large language models without model training. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 3201--3219, 2024.

[4] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy)*, pages 11--15, 2008.

[5] Geoffrey E. Hinton and Ruslan R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504--507, 2006.

[6] Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 13358--13376, 2023.

[7] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74--81, 2004.

[8] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Ro-

BERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[9] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 8086--8098, 2022.

[10] Ollama. Ollama: Get up and running with large language models locally. `https://ollama.com`, 2023.

[11] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[12] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311--318, 2002.

[13] PopoDameron. High-level overview of reinforcement learning from human feedback. Wikimedia Commons, CC BY-SA 4.0, `https://commons.wikimedia.org/wiki/File:RLHF_diagram.svg`, 2024.

[14] Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation (WMT)*, pages 186--191, 2018.

[15] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

[16] Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design. In *International Conference on Learning Representations (ICLR)*, 2024.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you

need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[18] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 1096--1103, 2008.

[19] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep self-attention distillation for task-agnostic compression of pre--trained transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[20] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations (ICLR)*, 2022.

[21] An Yang et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[22] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. BERTScore: Evaluating text generation with BERT. In *International Conference on Learning Representations (ICLR)*, 2020.

[23] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[24] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *International Conference on Learning Representations (ICLR)*, 2023.