



Departman za poslediplomske studije

**Integracija algoritama za razumevanje
prirodnog jezika i API izvora znanja u
konverzionom sistemu za univerzitetsku
administraciju**

- Master rad -

Mentor:

Doc. Dr Miloš Mravik

Kandidat:

Bojan Papaz
2024440095

Sadržaj

1	Metodologija.....	5
1.1	Uvodne napomene	5
1.2	Predmet istraživanja.....	5
1.3	Hipoteza	5
1.4	Ciljevi istraživanja	6
2	Uvod.....	7
2.1	Motivacija i uvodni okvir.....	7
2.2	Širi kontekst i primena	7
2.3	Problem i istraživački izazovi.....	7
2.4	Ciljevi rada.....	8
2.5	Doprinos rada.....	8
2.6	Opseg, pretpostavke i ograničenja	9
2.7	Pregled metodologije i evaluacionih kriterijuma	9
3	Pregled literature i tehnologija.....	10
3.1	Konverzacioni sistemi i <i>NLU</i> pristup.....	10
3.2	Rasa arhitektura: <i>NLU</i> i politike dijaloga	10
3.3	<i>NLU</i> komponente relevantne za rad.....	10
3.4	<i>DIETClassifier</i>	10
3.5	Politike dijaloga: <i>RulePolicy</i> i <i>TEDPolicy</i>	10
3.6	<i>Fallback</i> mehanizmi i „human handoff“.....	10
3.7	Jezičke reprezentacije: <i>BERT</i> i <i>mBERT</i>	11
3.8	Rasa <i>Action</i> Server i prilagođene akcije	11
3.9	Frontend tehnologije: <i>Vue 3</i> , <i>Chart.js</i> , <i>Axios</i> , <i>Socket.IO</i> , <i>webchat widGET</i>	11
3.10	Sažetak tehničke podloge.....	11
4	Analiza zahteva.....	12
4.1	Opseg i pretpostavke.....	12
4.2	Funkcionalni zahtevi (FR)	12
4.3	Nefunkcionalni zahtevi (NFR).....	15
4.4	Stakeholderi	16
4.5	Mapiranje	17

5	Arhitektura sistema	18
5.1	Visokonivojski prikaz arhitekture	18
5.2	Dijagram toka poruke <i>end-to-end</i>	19
5.3	Komponente i odgovornosti.....	19
5.4	Diskusija dizajna.....	20
6	Implementacija.....	21
6.1	RASA <i>NLU</i> i dijalog	21
6.1.1	Struktura i artefakti:	21
6.1.2	<i>NLU</i> pipeline i algoritmi:	21
6.1.3	Politike dijaloga i orkestracija:	22
6.1.4	Domen, podaci i balans:.....	22
6.2	Custom akcije i integracije.....	22
6.2.1	<i>Action</i> Server i klase:	22
6.2.2	<i>action_API_response</i> , algoritam i protok:	22
6.2.3	<i>action_default_fallback</i> : algoritam i UX:	23
6.2.4	Bezbednost i konfiguracija:	23
6.2.5	Performanse:	23
6.3	<i>AgentChatView</i> (Analitika i CMS).....	23
6.3.1	Servisni sloj:.....	25
6.3.2	Prikazi i algoritmi po ruti:.....	26
6.3.3	Bezbednost i performanse u <i>SPA</i> :	27
6.4	Pomoćne skripte.....	28
6.4.1	<i>train_from_API.py</i> : ETL tok i algoritmi:.....	28
6.4.2	Testovi i praćenje kvaliteta:	28
7	Rezultati i diskusija.....	29
7.1	Protokol evaluacije.....	29
7.2	<i>NLU</i> performanse (klasifikacija namera).....	29
7.3	Latencija odziva	31
7.4	<i>Fallback</i> stopa i robustnost	31
7.5	Ablacione studije	32
7.6	Tačnost isporučenih odgovora	32
7.7	Analiza grešaka	33
7.8	Pretnje valjanosti i ograničenja.....	33

8	Bezbednost, privatnost i održavanje	35
8.1	Model pretnji i osnovne pretpostavke.....	35
8.2	Zaštita u transportu i na granici sistema	35
8.3	Autentikacija i autorizacija	35
8.4	Upravljanje tajnama i konfiguracijom	35
8.5	Validacija ulaza i bezbedan prikaz izlaza.....	35
8.6	Logovanje, nadzor i reagovanje	36
8.7	Privatnost i zaštita podataka.....	36
8.8	Povezanost sa algoritmima i rizikom	36
9	Ograničenja i budući rad.....	37
10	Zaključak.....	40
11	Literatura.....	41
12	Sadržaj slika i grafova.....	42

1 Metodologija

U ovom delu rada biće objašnjena metodologija i ideja celokupnog rada, kao i problem i predmet istraživanja.

1.1 Uvodne napomene

Metodologija se zasniva na principu razdvajanja razumevanja poruke od upravljanja znanjem, uz strogo empirijsko proveravanje svake dizajnerske odluke. Rasa okruženje obezbeđuje *NLU* (*Natural Language Understanding*) i kontrolu dijaloga, dok se sadržaj odgovora preuzima iz spoljnog *Q&A* sloja preko prilagođenih akcija. Evaluacija se sprovodi u kontrolisanom okruženju sa fiksiranim verzijama biblioteka, jasno definisanim skupovima podataka i postupcima koji omogućavaju ponovljivost. Posebna pažnja posvećuje se privatnosti i bezbednosti: podaci se pseudonimizuju, logovi ne sadrže lične identifikatore, a merenja latencije i kvaliteta ne kompromituju sadržaj. Pseudonimizacija podataka je u skladu sa preporukama člana 32 GDPR-a, gde se ona navodi kao preporučena bezbednosna mera [1]. Sve metrike se posmatraju i u laboratorijskim uslovima i u stvarnom radu sistema, kako bi se obuhvatili i hladni start i uticaj spoljnih servisa.

1.2 Predmet istraživanja

Predmet istraživanja je efikasnost i robusnost konverzacionog sistema za domen upisa, sa fokusom na algoritamske komponente i njihovu međuzavisnost. Ispituje se *NLU* pipeline koji kombinuje *DIET* (*Dual Intent and Entity Transformer*) klasifikator, *mBERT* (*multilingual Bidirectional Encoder Representations from Transformers*) reprezentacije i karakterno n-gram vektorisanje, uticaj *RulePolicy* se koristi za determinističke mape [2] i *TEDPolicy* na izbor naredne akcije koji inače koristi Transformer arhitekturu za modelovanje konteksta [3] [4], kao i ponašanje *FallbackClassifier*-a pri niskom poverenju. Analizira se integracija sa *Q&A API*-jem (*Application Programming Interface*) kroz posebnu akciju, uključujući efekat keširanja i strategija ponavljanja zahteva na latenciju. Poseban segment obuhvata analitički panel koji iz produkcionih događaja generiše korpus za novo treniranje, uz procenu doprinosa deduplikacije i normalizacije dijakritika.

1.3 Hipoteza

Polazna hipoteza glasi da kombinacija *char_wb* n-grama i *mBERT* reprezentacija u *DIET* klasifikatoru, uz *Rule* i *TED* (*Transformer Embedding Dialogue*) politike i urednički kontrolisan *Q&A* sloj, omogućava pouzdan rad u domenu upisa sa makro-F1 (metrika za klasifikaciju namera ili harmonijska sredina preciznosti i odziva. $F1 = 2 \cdot (Precision \cdot Recall) / (Precision + Recall)$) iznad definisanog praga, stopom *fallback*-a ispod operativnog cilja i median latencijom ispod očekivane granice u režimu toplog keša. Dodatne hipoteze su da normalizacija dijakritika i heuristički *ETL* iz realnih poruka statistički značajno smanjuju *fallback* i poboljšavaju F1 u dugom repu retkih namera, te da keširanje odgovora po nameri i konekcioni *pool* na akcionom serveru smanjuju median latenciju bez degradacije tačnosti.

1.4 Ciljevi istraživanja

Glavni cilj je projektovati, implementirati i empirijski potvrditi konverzacioni sistem koji zadovoljava funkcionalne i nefunkcionalne zahteve, uz jasnu tragačnost između arhitekture, algoritama i rezultata. Operativni ciljevi su da se izvede analiza doprinosa *mBERT*-a, *char_wb*-a i *TED*-a. Isto tako trebaju da se izmere *NLU* metrike i percentili latencije, kvantifikuje stopa *fallback*-a i oceni tačnost isporučenih odgovora iz *Q&A* baze. Inženjerski ciljevi obuhvataju izradu analitičkog panela, *ETL* skripti i smernica za bezbednost i održavanje, kao i definisanje putanje skaliranja ka univerzitetskom nivou kroz hijerarhijski *NLU*, višetenantški *CMS* i segmentisanu analitiku.

2 Uvod

2.1 Motivacija i uvodni okvir

Razvoj konverzionih sistema poslednjih godina prešao je put od jednostavnih skriptovanih “FAQ botova” do inteligentnih asistenata zasnovanih na naprednim modelima razumevanja prirodnog jezika. Takvi sistemi danas preuzimaju značajan deo komunikacije između institucije i korisnika, naročito u domenima sa veoma velikim obimom repetitivnih upita, strogo definisanom regulativom i dinamičnim informacijama koje se konstantno prenose i menjaju. Univerzitetska ustanova predstavlja upravo takav domen: kandidati traže informacije o rokovima, uslovima, potrebnoj dokumentaciji, školarinama i procedurama. Odgovori od strane ustanove se često menjaju i zavise od zvaničnih odluka, kalendara i internih pravila. Sve to nije moguće efektivno ispratiti bez *AI* bota. Kako pokazuje studija [5], primena chatbota u procesu upisa značajno smanjuje opterećenje službi i ubrzava odziv kandidatu.

U tom kontekstu, predmet ovog rada je dizajn, implementacija i evaluacija domen-specifičnog konverzionog sistema koji sa jedne strane pruža tačne i pravovremene informacije potencijalnom kandidatu ili postojećem korisniku, a sa druge, omogućava nastavnim i administrativnim službama da te informacije lako ažuriraju bez ponovnog učenja modela. Poseban akcenat stavljen je na jezičke specifičnosti srpskog jezika (dijakritici, varijante pisanja, lapsusi, kolokvijalni izrazi), kao i na operativne zahteve produkcionog okruženja (bezbednost, latencija, merenje kvaliteta i nadzor rada sistema).

2.2 Širi kontekst i primena

Visokoškolske institucije godišnje obrađuju veliki broj upita kandidata kroz različite kanale (telefon, e-pošta, društvene mreže, pristupne forme). Najveći deo upita ponavlja slične obrasce: “kada počinje upis”, “koja dokumenta su potrebna”, “kako se prijaviti za prijemni”, “koliko košta školarina”, “kako se prenose ESPB bodovi” “koji je e-mail profesora...” itd. Tradicionalna obrada takvih pitanja zahteva značajne ljudske resurse, usporava odziv i povećava rizik od nekonzistentnih informacija.

Konverzioni sistem sa automatskim razumevanjem namera (*NLU*) i integracijom na zvanični izvor istine (institucionalni *Q&A*/Informacioni sistem) smanjuje opterećenje službi, dodatno skraćuje vreme odziva na nekoliko sekundi i još podiže konzistentnost komunikacije. Pored toga, uz prateći analitički modul, institucija dobija jasan uvid u to šta korisnici najčešće pitaju, gde nastaju nesporednosti i kako treba preusmeriti prioritete za izmenu sadržaja ili proširenje znanja.

2.3 Problem i istraživački izazovi

Iako domen deluje kao običan „*FAQ*“, problem je daleko od trivijalnog. Najpre, potrebno je pouzdano razumeti upite na srpskom jeziku uz prisustvo dijakritika, čestih lapsusa, mešavine ekavice i ijekavice, kolokvijalnih izraza i čestog preplitanja sa engleskim tehničkim

terminima. Zatim dolazi zahtev za generalizacijom i robusnošću *NLU*-a, jer model mora da mapira mnogobrojne varijante istog pitanja na pravilnu nameru uprkos ograničenom budžetu anotacija. Treći izazov je održavanje znanja: rokovi, cene i dokumenta se stalno menjaju pa odgovori ne smeju biti „ugrađeni“ u model već centralizovani u spoljnom izvoru istine, tipično *REST API* ili baza podataka.

Paralelno s tim stoje operativne potrebe produkcionog sistema, uključujući kontrolu latencije, dosledno logovanje i nadzor, bezbedno rukovanje tajnama i kontrolu pristupa, kao i analizu fallback slučajeva koja omogućava kontinuirano učenje iz stvarnih događaja. Konačno, evaluacija kvaliteta ne može da se zaustavi na standardnim *NLU* metrikama, već mora da uključi korisnički relevantne indikatore kao što su stopa fallback-a, tačnost isporučenog odgovora iz izvora istine i vremenski odziv sistema.

2.4 Ciljevi rada

Polazeći od navedenih izazova, rad postavlja sledeće ciljeve u formi jedinstvenog zadatka razvoja i provere sistema: dizajn i implementaciju konverzionog asistenta za domen upisa zasnovanog na Rasa okviru, sa *NLU* tokom obrade prilagođenim srpskom jeziku i pravilima koja obezbeđuju stabilan tok dijaloga; dosledno odvajanje znanja od modela putem integracije na zvanični *Q&A REST API*, kako bi se izmene sadržaja vršile bez ponovnog treniranja; izradu analitičkog modula kao jednostranične aplikacije koja pruža statistiku namera, pregled događaja i fallback slučajeva te omogućava *CRUD* operacije nad bazom za operativno održavanje i kontrolu kvaliteta; uspostavljanje metodologije evaluacije koja obuhvata standardne *NLU* metrike po nameri, merenje stope fallback-a, latenciju odgovora i tačnost isporučenih informacija; i na kraju, predlog i demonstraciju postupka kontinuiranog učenja kroz prikupljanje stvarnih korisničkih pitanja iz događaja, njihovu obradu, deduplikaciju i balansiranje klasa radi ciljne dopune *NLU* korpusa.

2.5 Doprinos rada

Rad ostvaruje meren, praktično primenljiv doprinos kroz celovitu end-to-end platformu za isporuku informacija, koja integriše konverzionog asistenta (Rasa *NLU* sa prilagođenim akcijama) i analitički modul sa jasno razgraničenim odgovornostima i dokumentovanim *API* vezama. *NLU* konfiguracija je domenski specifična i prilagođena srpskom jeziku, kombinuje n-gram featurizaciju i višejezičke reprezentacije zasnovane na *BERT*-u, a prag za fallback je pažljivo podešen da održi ravnotežu između pokrivenosti i preciznosti. Znanje je eksplicitno eksternalizovano, pa se odgovori preuzimaju iz verifikovanog institucionalnog *REST API*-ja, što omogućava tmsko ažuriranje sadržaja bez potrebe za ponovnim treniranjem modela. Analitički modul obezbeđuje pregled namera, hronologiju događaja, listu fallback slučajeva i *CRUD* nad *Q&A* bazom, čime se omogućava nadzor kvaliteta, brza korekcija i priprema materijala za sledeće cikluse učenja. Pored toga, definisan je jasan tok kontinuiranog učenja u kome se produkciono događaji pretvaraju u kandidatske fraze i proširen *NLU* skup, uz preporučenu deduplikaciju i balansiranje klasa.

Ovaj doprinos je orijentisan na produkciju i realne korisnike. Rešenje je projektovano i implementirano tako da obezbedi tražene osobine servisa kao što su niska latencija, stabilnost

pod opterećenjem, bezbedno upravljanje tajnama i održivost u eksploataciji, a ne samo dobar rezultat u laboratorijskim uslovima.

2.6 Opseg, pretpostavke i ograničenja

Rad se fokusira na *intent* bazirani pristup (klasifikacija namera) u domenu upisa, bez generativnog odgovaranja otvorenog tipa. Ekstrakcija entiteta i složenije forme mogu biti uvedene, ali osnovni scenario koji rad detaljno obrađuje jeste *mAPI*ranje korisničkog pitanja na nameru i dohvat uredničkog odgovora iz zvaničnog *API*-ja koristeći skup složenih algoritama. Počinje se od pretpostavke da je institucionalni *API* dostupan, ažuran i bezbedno izložen za asistentove potrebe. U pogledu jezika, cilj je robusnost na dijakritičke varijacije i lapsuse. Potpuna pokrivenost svih jezičkih fenomena (npr. humor, sarkazam) nije predmet ovog rada, ali je razmatrano za budući razvoj projekta.

Ograničenje koje se eksplicitno prepoznaje jeste zavisnost kvaliteta odgovora od uredničkog sadržaja i politika institucije (npr. ažurnost *Q&A* baze). Dodatno, iz bezbednosnih razloga naglašava se potreba da se tajne (tokeni) nikada ne drže u klijentskom kodu, što rad adresira arhitektonskim preporukama.

2.7 Pregled metodologije i evaluacionih kriterijuma

Evaluacija kvaliteta obuhvataće: *NLU* metrike po nameri (preciznost, odziv, F1, makro i mikro) uz matricu konfuzije i analizu “teških” parova namera, stopu *fallback*-a na validacionom/produkcijskom uzorku, latenciju od prijema poruke do isporuke odgovora i kvalitativnu tačnost isporučenog sadržaja (ručno označavanje na uzorku razgovora: tačno/delimično/netačno). Predviđena je i analiza uticaja pojedinih komponenti pipeline-a (n-gram featurizacija, *mBERT*, normalizacija dijakritika) i pragova *fallback*-a na ukupne performanse, kao i kratak korisnički test (anketa) radi percepcije upotrebljivosti.

3 Pregled literature i tehnologija

3.1 Konverzacioni sistemi i *NLU* pristup

Prema [6], u literaturi i praksi razlikuju se dva dominantna pristupa: *NLU*-orijentisan (klasifikacija namera i ekstrakcija entiteta...) i generativni modeli koji sintetišu odgovor. U *NLU* pristupu, pipeline pretvara nestrukturisanu poruku korisnika u strukturisanu reprezentaciju (namera, entiteti), nakon čega komponenta za upravljanje dijalogom donosi odluku o sledećem koraku. Dokumentacija Rasa bota naglašava da je pipeline niz komponenti (tokenizacija, featurizacija, klasifikatori) koji je moguće konfigurisati prema domenu i jeziku.

3.2 Rasa arhitektura: *NLU* i politike dijaloga

Rasa sisteme čine dva sloja: *NLU* (klasifikacija namera, ekstrakcija entiteta, opcioni odabir odgovora) i dialogue management (politike odlučivanja o sledećoj akciji), što je u skladu sa arhitekturom opisanom u radu [7]. Politike (npr. *RulePolicy*, *TEDPolicy*) obrađuju kontekst i istoriju kako bi predložile akciju (npr. slanje poruke, pokretanje korisničke akcije). U zvaničnoj dokumentaciji se pojašnjava razdvajanje *NLU* pipeline-a i sloja politika, kao i ulogu svake komponente u odlučivanju.

3.3 *NLU* komponente relevantne za rad

Tipičan Rasa *NLU* pipeline obuhvata: tokenizer (npr. *WhiteSpaceTokenizer*), featurizere (npr. *COUNTVectorsFeaturizer* za reči), jezički featurizer zasnovan na unapred istreniranom modelu (npr. *LanguageModelFeaturizer* za *BERT* reprezentacije), kao i klasifikatore. *LanguageModelFeaturizer* daje guste vektorske reprezentacije poruke (i, po potrebi, kandidata za odgovor), koje dalje koriste klasifikatori i ekstraktori entiteta.

3.4 *DIETClassifier*

DIET (Dual *Intent* and Entity Transformer) je multitasking Transformer koji istovremeno uči klasifikaciju namera i ekstrakciju entiteta. U Rasi je implementiran kao trenabilna komponenta pipeline-a (klasa *DIETClassifier*) i u praksi se koristi kao „radni konj“ za većinu *NLU* projekata radi balansa performansi i efikasnosti. [8]

3.5 Politike dijaloga: *RulePolicy* i *TEDPolicy*

RulePolicy implementira deterministička pravila (npr. za nameru X pokreni akciju Y), pogodna za stabilne i već očekivane tokove. *TEDPolicy* (Transformer Embedding Dialogue) je neuronska politika koja uči reprezentacije dijalog konteksta i bira sledeću akciju na osnovu istorije (intencije korisnika, prethodne akcije sistema, aktivni slotovi i formulari).

3.6 *Fallback* mehanizmi i „human handoff“

Kada je poverenje *NLU* klasifikacije nisko ili je poruka van domena, sistem treba da aktivira fallback ponašanje definisano kroz posebnu `nlu_fallback` nameru i odgovarajuću

politiku, kao i mogućnost prosleđivanja razgovora ljudskom agentu (human handoff), što je u skladu sa preporukama Rasa dokumentacije. [9] [10]

3.7 Jezičke reprezentacije: *BERT* i *mBERT*

BERT (Bidirectional Encoder Representations from Transformers) je unapred istreniran jezički model koji ostvaruje konkurentne rezultate na raznim *NLU* zadacima finim doterivanjem (fine-tuning) sa minimalnim dodatnim slojem. BERT (Bidirectional Encoder Representations from Transformers) je predstavljen u radu Devlin i saradnika [11], dok mBERT varijanta, prema HuggingFace dokumentaciji [12], podržava više od 100 jezika i pogodna je za sisteme koji primaju upite na različitim jezicima širom sveta.

3.8 Rasa *Action Server* i prilagođene akcije

Za poslovnu logiku koju nije moguće obaviti samo generalnim porukama i pravilima (npr. dohvat iz baze, poziv eksternog *API*-ja), Rasa bot koristi *Action Server* sa posebnim komandama (Python, Rasa SDK). Server Rasa šalje zahtev akcionom serveru kada politika predvidi prilagođenu akciju; po završetku, akcioni server vraća odgovore i događaje nazad u tok dijaloga. Kada je potrebno povezivanje sa eksternim sistemima, Rasa koristi Action Server i prilagođene akcije (custom actions), što je u skladu sa Rasa dokumentacijom i preporukama. [13] [14] [15]

3.9 Frontend tehnologije: *Vue 3*, *Chart.js*, *Axios*, *Socket.IO*, *webchat* *widGET*

Vue 3 je progresivni JS framework za izradu *SPA* aplikacija. *Chart.js* je popularna biblioteka za grafikone (canvas), često korišćena u *SPA* okruženjima zbog jednostavnosti i fleksibilnosti. *Axios* je promise-based *HTTP* klijent za browser i Node.js koji je zapravo praktičan za *REST* integracije. *Socket.IO* obezbeđuje dvosmernu, događajima vođenu komunikaciju sa automatskim prebacivanjem transporta i ugrađenim mehanizmom za ponovno povezivanje.

3.10 Sažetak tehničke podloge

Relevantna literatura i zvanična dokumentacija ukazuju na sledeće stubove: *NLU* pipeline kao niz kombinovanih koraka (tokenizacija, featurizacija, klasifikacija), politike za upravljanje tokom (determinističke i neuronske), *fallback* mehanizme za robusnost u realnim uslovima rada, već trenirane jezičke modele (*mBERT*) za poboljšanu semantiku na više jezika i varijacija pisanja, *Action Server* za integraciju sa spoljnim sistemima i *SPA* frontend sa grafikonima i real-time komunikacijom za analitiku i operativno održavanje. Ovakva kombinacija je postala standard za produkcione, domenski specifične asistente sa zahtevima preciznosti, održavanja i nadzora rada.

4 Analiza zahteva

4.1 Opseg i pretpostavke

Što se tiče opsega sistem pokriva: razumevanje korisničkih poruka (*NLU*) i izbor sledeće akcije (dijalog politike), dinamičko preuzimanje odgovora iz zvaničnog *Q&A REST API*-ja, analitiku događaja i namera, uređivanje *Q&A* baze (*CRUD*) i polu-automatizovano proširenje *NLU* korpusa iz realnih događaja.

Pretpostavke:

- Institucionalni *Q&A API* je dostupan, vraća verodostojne i ažurne odgovore, i zahteva **x-token** autentifikaciju.
- Rasa *Action Server* je dostupan i može da izvršava **custom akcije** (*HTTP* poziv ka *Q&A API*-ju).
- Srpski jezik je dominantan; model mora biti robustan na dijakritike i lapsuse.
- *Vue SPA* koristi *HTTP* i/ili *Socket.IO* i prikazuje statistiku, događaje i *Q&A* sadržaj.

Ograničenja:

- Generativno odgovaranje (*LLM (Language Learning Model)*) nije u opsegu, umesto njega, koristi se *intent*-based pristup.
- Ekstrakcija entiteta i forme/slotovi nisu obavezni u osnovnoj verziji (moguće proširenje).

4.2 Funkcionalni zahtevi (FR)

NLU i dijalog

FR-1 - Klasifikacija namere poruke

Sistem mora klasifikovati ulaznu poruku u jednu od oko 65 podržanih namera.

Algoritamski aspekti koji se koristi je *DIETClassifier* (transformer) sa softmax izlazom nad skupom namera, ulazni vektori su kombinacija:

- *CountVectorsFeaturizer* sa *char_wb* analizom implementiran je kroz *scikit-learn* biblioteku [16], dok se konfiguracija *pipeline*-a i kombinovanje *sparse* i *dense* karakteristikovanja zasniva na preporukama iz Rasa dokumentacije [17].
- ***LanguageModelFeaturizer (mBERT)*** - guste reprezentacije (embedding) rečenice. Ovde algoritam koristi algoritme koji obraćaju pažnju prema sebi “self-attention” za kontekstualizaciju tokena.

Kriterijum prihvata Makro-F1 $\geq$ ciljne vrednosti (npr. $\geq 0,85$) na validacionom skupu; precizno logovanje *intent* i confidence.

FR-2 - Odabir sledeće akcije

Sistem mora odlučiti koju akciju izvodi posle *NLU* predikcije.

Algoritamski aspekt je podeljen na *RulePolicy*: determinističko “if *intent* → *action*” mapiranje (pravila u *rules.yml*) i *TEDPolicy*: transformer politika koja obraća pažnju “self-attention” nad istorijom (poslednjih k koraka) i bira akciju po sličnosti u ugrađenom prostoru (embedding matching).

Za sve definisane tokove, izabrana akcija odgovara očekivanom pravilu. U konfliktu odlučivanje favorizuje *RulePolicy* za eksplicitne mape.

FR-3 - *Fallback* ponašanje

Ako je poverenje ispod praga ili postoji dvosmislenost, sistem aktivira *fallback*.

Algoritamski aspekt koji se ovde prikazuje je *FallbackClassifier*, kao standardna komponenta Rasa NLU sistema i ona donosi binarnu odluku na osnovu praga poverenja [9] [18] (threshold 0,45) i ambiguity_threshold (0,1). *NLU_fallback* se emituje za poruke sa niskim poverenjem. Sistem ne čuva i ne unapređuje znanje koristeći poruke koje su van opsega koristi bota.

FR-4 - Dinamički dohvat odgovora iz *Q&A API*-ja

Ovaj pristup odgovara konceptu „deljenja znanja“ opisanom u Rasinoj dokumentaciji o Knowledge Base Actions [15] i Rasa HTTP API specifikaciji [19].

- Custom akcija (*action_API_response*): sinhroni *HTTP GET* ka *.../API/question/intent/{intent}*, header x-token.
- Algoritam selekcije polja: parsiranje *JSON* i izdvajanje odgovora (tekst/*HTML*).
- (Po želji) *Retry/backoff* strategija (eksponencijalni backoff) u slučaju 5xx greške.

Za uspešan status (2xx) vraća se sadržaj iz answer; za grešku - definisana “pristojna” *fallback* poruka.

FR-5 - Statistika namera

Statistika je odrađena pomoću histograma/top-N namera po učestalosti.

- Agregacija (server-side): *GROUP BY intent* → *COUNT(*)*.
- Sortiranje: descending po *COUNT*.
- Vizuelizacija: *Chart.js*

Top-N lista korespondira sa serverskim agregatima tako što je prikaz ažuran u realnom vremenu osvežavanjem.

FR-6 - Pregled događaja (paginacija)

Opis. Omogućiti pregled događaja sa navigacijom po stranama.

- Server-side paging: parametri page, size, sort=id,desc.
- Algoritam rendera: klijentska logika prev/next/first/last zasnovana na number, totalPages.

Navigacija je konzistentna, bez dupliranja i praznih stranica; vreme učitavanja ostaje u SLA.

FR-7 - Lista pitanja/odgovora

Učitavanje *Q&A*

- *HTTP GET .../API/question/category/upis* sa x-token; render tabele.

Lista korespondira backend stanju.

FR-8 - Dodavanje/izmena/brisanje *Q&A*

Admin može kreirati, menjati ili brisati pitanja.

FR-9 - Generisanje *NLU* primera iz događaja

Polu-automatizovano obogaćivanje *NLU* korpusa iz realnih poruka.

Skripta *trAIIn_from_API.py* radi:

- Dohvat *.../rasa/upis/events* (token).
- Heuristička klasifikacija: *mAPI*ranje na namere preko skupa ključnih reči *INTENT_KEYWORDS* (algoritam: pretraga substringa / tokena).
- Grupisanje: bucket-izacija poruka po nameri.
- Generisanje *YAML*: serijalizacija u *data/NLU.yml*.
- (Preporuka) Deduplikacija (algoritam sličnosti: npr. Levenshtein/Jaccard) i normalizacija dijakritika (č→c, ć→c, š→s, đ→d, ž→z) pre upisa.

Generisan *NLU.yml* je sintaksno validan (Rasa validator), broj novih primera > 0, a duplikati minimalni.

FR-11 - Webchat/*REST* interakcija

Omogućiti slanje poruka preko web widGETa (Socket.IO) i/ili *REST* klijenta.

- Socket.IO: dvosmerni kanal sa rekonekcijom
- *REST*: *POST* /webhooks/*REST*/webhook sa *JSON* telom - render odgovora sekvencijalno.

Poruke se isporučuju bez gubitaka, što znači da je reconnect uspešan pri prekidu veze.

4.3 Nefunkcionalni zahtevi (NFR)

NFR-1 - Performanse (latencija)

Vreme odziva od poruke do odgovora treba da bude nisko.

Metrika: $P50 \leq 500$ ms, $P95 \leq 2000$ ms (end-to-end, bez mrežnih anomalija).

- Keširanje statičnih odgovora (LRU keš u akciji, TTL).
- Batching/pooling *HTTP* konekcija; timeout i *Retry/backoff* (kontrolisano).
- Ograničavanje složenosti pipeline-a (npr. broj epoha, veličina LM featurizera u produkciji).

NFR-2 - Tačnost *NLU*

Visok kvalitet klasifikacije namera.

Metrika: Makro-F1 $\geq 0,85$; stopa *NLU_fallback* $\leq 10\%$ na produkcionom uzorku.

- Učešće *char_wb* n-grama i *mBERT* tuning praga u *FallbackClassifier*-u.
- Class balancing (downsample dominantnih, obogaćivanje retkih).

NFR-3 - Robusnost na lapsuse/dijakritike

Sistem mora da prepozna poruke sa greškama i bez kvačica.

Metrika: Degradacija F1 ≤ 5 pp pri simuliranim lapsusima/dijakritičkim varijacijama.

- *char_wb* n-grami, *mBERT*; preprocesing: normalizacija dijakritika u skripti za učenje.
- (Po želji) fuzzy match (Levenshtein) u pametnijem *fallback*-u.

NFR-4 - Pouzdanost i dostupnost

Sistem funkcioniše stabilno i obnavlja vezu.

Metrika: Uptime $\geq 99,5\%$ mesečno; stopa neuspehlih odgovora $< 1\%$.

- Socket.IO auto-reconnect; circuit breaker na *HTTP* pozivima ka *Q&A API*-ju.

NFR-5 - Bezbednost

Tajne i nalozi su zaštićeni; pristup ograničen.

Metrika: Nema tokena u klijentskom kodu; svi tajni podaci kroz ENV/secret manager ili proxy.

- Server-side proxy za *Q&A*; *rate-limit*; input sanitization; Content-Security-Policy u frontendu.
-

NFR-6 - Privatnost i usaglašenost

Minimalno prikupljanje ličnih podataka; poštovanje pravila (npr. GDPR).

Metrika: Logovi ne sadrže PII, definisana politika retencije.

- Anonymization/masking u logovima; selektivno logovanje polja događaja.

NFR-7 - Održivost i održavanje

Jednostavno ažuriranje znanja i modela.

Metrika: Izmena odgovora bez *retrain*; *retrain* ciklus $\leq 1h$ po iteraciji.

- Znanje izmešteno u *Q&A API*; CI/CD skripte za *test/train*. validatori *YAML*.

NFR-8 - Skalabilnost

Sistem treba da opsluži rast saobraćaja.

Metrika: Linearno skaliranje do ciljanog broja istovremenih sesija (npr. $5\times$).

- Horizontalni autoscaling *Rasa/Action* servera; keširanje “hot” odgovora; connection pooling.

NFR-9 - Upotrebljivost (analitika/CMS)

Dashboard i *Q&A* editor moraju biti jasni i responsivni.

Metrika: TTFHW (“time-to-first-helpful-work”) < 5 min za urednika; SUS ≥ 80 .

- Client-side pagination + jasan sorting; validacija formi; pristojan error handling.

NFR-10 - Observabilnost

Merenje metrika i praćenje anomalija.

Metrika: Dostupni grafovi F1, *fallback* stope i latencije; alarmi na 5xx spike.

- Aggregation i percentile proračuni; roll-up dnevne/časovne metrike.

4.4 Stakeholderi

- **Kandidati / korisnici** - primarni korisnici asistenta; očekuju tačne, ažurne i brze odgovore.
- **Služba za upis / informacije** - vlasnik sadržaja odgovora; uređuje *Q&A* kroz CMS; odgovorna za tačnost i ažurnost informacija.
- **IT tim (aplikativna podrška)** - održava *Rasa* i *Action Server*, nadgleda integracije, rešava incidente, upravlja tajnama (tokeni).
- **Urednik sadržaja / komunikacija** - standardizuje stil odgovora, kontroliše kvalitet poruka i konzistentnost informacija.
- **DevOps / Bezbednost** - automatizuje deploy, prati dostupnost, sprovodi sigurnosne politike (*rate-limit*, keševi, proxy).
- **Uprava / Menadžment** - koristi metrika iz analitike za planiranje resursa i poboljšanje procesa upisa.

- **Istraživačka/razvojna uloga** - priprema evaluacije (F1, *fallback* stopa, latencija), vodi ablacione studije i iterativno unapređuje algoritme (npr. deduplikaciju, normalizaciju, *fuzzy-fallback*).

4.5 Mapiranje

FZ-1/2/3 (Klasifikacija/Odabir/Fallback) ↔ *config.yml* (pipeline: *WhiteSpaceTokenizer*, *RegexFeaturizer*, *LexicalSyntacticFeaturizer*, *COUNTVectorsFeaturizer* – *word* + *char_wb* 3–5, *LanguageModelFeaturizer* – *mBERT*, *DIETClassifier*, *FallbackClassifier*), *data/rules.yml*, *RulePolicy*, *TEDPolicy*. Algoritmi: *DIET* (Transformer, softmax), *mBERT* (kontekstualni embedding), *char_wb* n-grami (robustnost na lapsuse/dijakritike), *FallbackClassifier* (threshold 0,45; ambiguity 0,1), *TED* (self-attention nad istorijom).

FZ-4 (Dohvat odgovora) ↔ *actions.py* (*action_API_response*), *endpoints.yml*. Algoritam: *HTTP GET* ka */API/question/intent/{intent}* sa x-token + parsiranje *JSON* polja *answer*; po želji *Retry/backoff* za 5xx; validirani izlaz kao tekst/*HTML*.

FZ-5/6/7 (Statistika/Pregled događaja/ Lista pitanja) ↔ *AgentChatView* (*src/services/mAIn.service.ts*, *src/views/HomeView.vue*, *src/views/TableView.vue*, *src/views/FallbackView.vue*). Algoritmi: server-side agregacija *GROUP BY intent* → *COUNT(*)* + sortiranje (*desc*) → vizuelizacija (*Chart.js*); server-side paging (*page*, *size*, *sort=id,desc*) + klijentska navigacija *first/prev/next/last*.

FZ-8/9 (Dodavanje, izmena, brisanje/Operativni kvalitet) ↔ *src/views/QuestionsView.vue*, *src/views/AddQuestionView.vue*, *src/views/QuestionDetails.vue*. Algoritmi: *CRUD* nad *Q&A* (*POST/PUT/DELETE*), validacija formi, status-based error handling; dohvat lista preko *GET /API/question/category/upis* i *GET /API/category*.

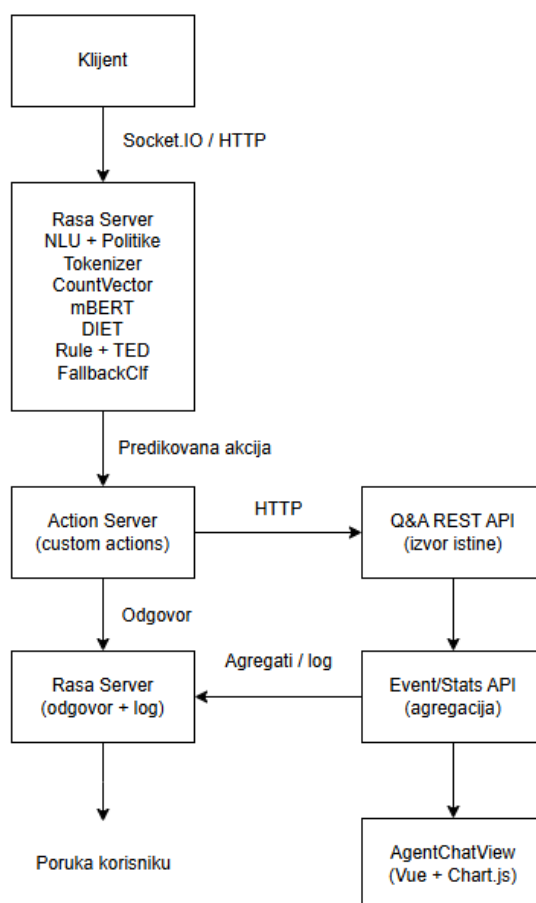
FZ-10 (Generisanje NLU primera) ↔ *trAIIn_from_API.py*. Algoritmi: heurističko mapiranje ključnim rečima *INTENT_KEYWORDS* (substring/token), grupisanje (bucket-izacija) poruka po nameri, generisanje *YAML* (*data/NLU.yml*); preporuka: deduplikacija (npr. Levenshtein/Jaccard) + normalizacija dijakritika (č→c, ć→c, š→s, đ→d, ž→z).

5 Arhitektura sistema

5.1 Visokonivojski prikaz arhitekture

Glavne celine sistema su:

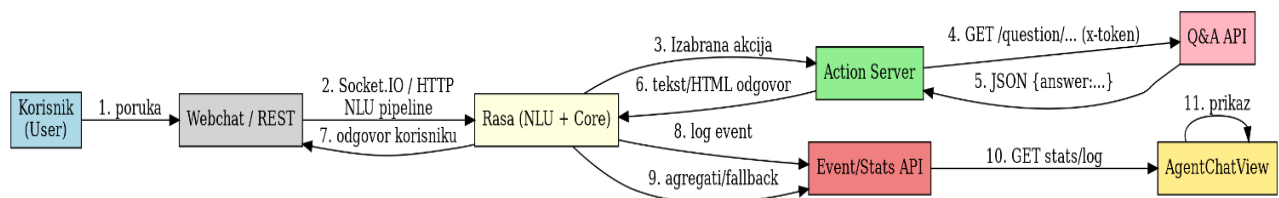
- Klijenti: web-chat zasnovan na Rasa botu
- Rasa server: *NLU* pipeline + politike dijaloga
- *Action* server: izvršavanje prilagođenih akcija i integracija sa *API*-jem
- Institucionalni *API* sloj: *API* koji je “izvor istine” i *API* za događaje/statistiku
- Baza podataka: perzistencija sadržaja i događaja
- Analitička *SPA* aplikacija (*AgentChatView*): uvid u namere, događaje, *fallback* slučajeve i *CRUD* nad *Q&A*



Na nivou *NLU*-a sistem koristi tok koji obuhvata tokenizaciju, vektorizaciju pomoću *CountVectors*-a za reči i karakternih *n*-grama sa granicom reči (*char_wb* 3–5), zatim *LanguageModelFeaturizer* sa *mBERT*-om i na kraju *DIETClassifier* zasnovan na *Transformeru*

koji preko softmaksu predviđa nameru. Odlučivanje u dijalogu oslanja se na *RulePolicy* koja daje determinističke mape i na *TEDPolicy* koja kao Transformer obrađuje istoriju razgovora i bira sledeću akciju, dok *FallbackClassifier* sa pažljivo podešenim pragovima poverenja preuzima kontrolu u slučajevima neizvesnosti. Integracioni sloj realizuje action server koji poziva *Q&A API* i korisniku prosleđuje sadržaj iz polja „*answer*“, čime se znanje drži van modela i ostaje urednički ažurno. Na klijentskoj strani web-chat komponenta (*rasa-webchat*) komunicira preko *Socket.IO* kanala, a analitička aplikacija koristi *Chart.js* i *REST* servise za prikupljanje i prikaz statistike. Jezičku osnovu obezbeđuje *mBERT* koji proizvodi dvosmerne kontekstualne reprezentacije na više jezika, uključujući srpski, što povećava robusnost na varijacije izraza i dijakritike.

5.2 Dijagram toka poruke *end-to-end*



Dijagram 2 - Tok poruke *end-to-end*

Objašnjenje koraka:

- 2: *NLU* pipeline izračunava reprezentacije: *n*-gram vektori i *mBERT* embedding, potom *DIET* daje distribuciju po namerama (softmax).
- 3: *RulePolicy* najpre rešava eksplicitne mape; u ostalim slučajevima *TEDPolicy* koristi samopažnju (self-attention) nad istorijom kako bi izabrala akciju.
- 4–6: *Action* server vrši *HTTP GET* prema *Q&A API*-ju, parsira answer i vraća ga Rasa serveru.
- 8–11: Događaji i agregati se izlažu preko *Event/Stats API*-ja; *AgentChatView* ih prikazuje, uključivo histogram namera (*Chart.js*) i liste događaja sa paginacijom.

Fallback grana:

Ako je poverenje ispod praga: *FallbackClassifier* predlaže *NLU_fallback*; dijalog politika pokreće definisanu *fallback* akciju/odgovor.

5.3 Komponente i odgovornosti

Klijentski sloj obuhvata web-chat koji koristi *Socket.IO* za dvosmernu komunikaciju sa Rasa serverom, dok probna *REST* stranica šalje poruke putem *POST* zahteva na */webhooks/rest/webhook*. Njegova odgovornost je prenos korisničke poruke, prikaz odgovora i osnovni korisnički doživljaj kao što su indikator tipkanja i istorija konverzacije. Na serverskoj strani razumevanje poruke ostvaruje se *NLU* tokom obrade koji uključuje tokenizaciju, vektorizaciju pomoću *CountVectors* featurizera nad rečima i karakternim *n*-gramima sa

granicom reči (*char_wb* 3–5), zatim *LanguageModelFeaturizer* sa *mBERT*-om, a na kraju *DIETClassifier* zasnovan na Transformeru koji softmaksom predviđa nameru. Upravljanje tokom razgovora sprovode *RulePolicy* kao deterministička mapa „namera → akcija” i *TEDPolicy* kao neuronska politika koja uz samopažnju nad istorijom dijaloga bira sledeću akciju u ugrađenom prostoru. Kada poverenje padne ili je poruka dvosmislena, *FallbackClassifier* uzima u obzir prag i meru dvosmislenosti i emituje *nlu_fallback*.

Prilagođene akcije se izvršavaju na akcionom serveru i omogućavaju uslovno obavljanje posla izvan *NLU/Core* sloja. Najčešći tok je poziv zvaničnog *Q&A REST API*-ja, parsiranje *JSON* odgovora i emitovanje sadržaja polja *answer* korisniku, uz mogućnost uvođenja kontrolisanog ponavljanja zahteva i sanitizacije podataka pre slanja odgovora. *Q&A API* je izvor istine i izlaže ugovor *GET /api/question/intent/{intent}* sa *header-om x-token*, a njegova odgovornost je da obezbedi urednički ažuran sadržaj nezavisan od treniranja modela. Paralelno s time, *Event/Stats API* perzistira događaje i izlaže agregate namera kroz upite tipa „*GROUP BY intent → COUNT(*)*”, kao i filtre za *nlu_fallback*, što se koristi u analitici.

Analitička jednostranična aplikacija *AgentChatView* zasnovana je na *Vue 3*, *Axios-u* i *Chart.js-u*. Ona prikazuje histogram ili top-N raspodelu namera, omogućava paginaciju događaja sa navigacijom *first/prev/next/last*, pregled liste *nlu_fallback* slučajeva i *CRUD* rad nad *Q&A* bazom. Algoritamski, klijentski sloj radi renderovanje grafikona, upravljanje stranicama i validaciju formi uz status-bazirano rukovanje greškama, dok serverski servisi obezbeđuju agregaciju i paginaciju. Jezičku osnovu celog sistema čini *mBERT* (*bert-base-multilingual-cased*) koji proizvodi dvosmerne kontekstualne reprezentacije na više jezika, uključujući srpski, čime se povećava robusnost na varijacije u pisanju i prisustvo dijakritika.

5.4 Diskusija dizajna

Ključna arhitektonska odluka je odvajanje znanja od modela: odgovori žive u *Q&A API*-ju, dok je model fokusiran na razumevanje namere, pa se ažuriranje informacija svodi na urednički proces bez ponovnog treniranja. Robusnost na jezičke varijacije postiže se kombinacijom karakternih *n*-grama tipa *char_wb* i *mBERT* reprezentacija, što povećava tolerantnost na lapsuse, nedosledne dijakritike i kolokvijalne oblike koji su česti u upisnim upitima. Upravljanje tokom razgovora oslanja se na prioritet pravila za deterministična ponašanja, dok *TEDPolicy* preuzima kontekstualne odluke u situacijama u kojima sama pravila nisu dovoljna. Observabilnost se obezbeđuje kroz agregate po namerama i izdvojenu listu *nlu_fallback* događaja, koji otkrivaju rupe u znanju i služe kao ulaz za kontinuirano učenje i unapređenja *NLU* cevovoda.

6 Implementacija

6.1 RASA *NLU* i dijalog

6.1.1 Struktura i artefakti:

- ***config.yml***: definicija *NLU* pipeline-a i politika dijaloga.
- ***domain.yml***: katalog intencija, minimalni skup utter_ odgovora i spisak registrovanih akcija.
- ***data/NLU.yml***: korpus za učenje (~65 intencija; ≈3.5k primera).
- ***data/rules.yml***: ≈67 pravila za determinističko *mAPI* ranje namera na akcije.
- ***endpoints.yml***: veza ka *Action* Server-u.
- ***credentials.yml***: kanali (*REST*, *Socket.IO*).
- ***models/*.tar.gz***: *SP*akovani trenirani modeli (npr. build od 24.07.2025)

6.1.2 *NLU* pipeline i algoritmi:

- ***WhiteSpaceTokenizer***: segmentacija ulazne poruke po belinama. Ulaz: sirov string; izlaz: lista tokena.
- ***RegexFeaturizer***: binarne regex osobine (npr. prisustvo obrazaca). Ulaz: tokeni, Izlaz: *SP*arse featuri.
- ***LexicalSyntacticFeaturizer***: leksičko-sintaksičke osobine po tokenu. Cilj je dopunski signal za *DIET*.
- ***CountVectorsFeaturizer (word)***: vektorizacija poruke kao skupa reči. Algoritam je term frequency nad vokabularom.
- ***CountVectorsFeaturizer (char_wb, 3–5)***: “word-boundary” n-grami karaktera. Algoritamska svrha: robusnost na lapsuse i dijakritike (č/ć/š/đ/ž).
- ***LanguageModelFeaturizer (mBERT)***: guste, kontekstualne reprezentacije dobijene iz *BERT-base-multilingual-cased*. Algoritam zasnovan na transformerom sa self-attention mehanizmom koji računa dvosmerni kontekst tokena.
- ***DIETClassifier***: multitask Transformer klasifikator. Ulaz: spregnuti *SP*arse (n-gram) i dense (*mBERT*) featuri. Izlaz: distribucija namera (softmax) + opcioni entitetski izlazi. Treniranje: cross-entropy nad namerama.
- ***EntitySynonymMapper***: normalizacija sinonima (trenutno bez većeg opterećenja u domenu).
- ***ResponseSelector***: komponenta za retrieval odgovore (ostavljena u pipeline-u za buduću upotrebu).
- ***FallbackClassifier***: binarna odluka o *NLU_fallback* na osnovu poverenja. Parametri: threshold: 0.45, ambiguity_threshold: 0.1.

Operativna posledica izvršenja navedenih funkcija je kombinacija ***char_wb*** n-grama i *mBERT* koji smanjuju pad performansi pri porukama bez kvačica ili sa lapsusima. *DIET* konsoliduje featurizacije u jedinstvenu distribuciju nad ~65 namera.

6.1.3 Politike dijaloga i orkestracija:

Upravljanje tokom dijaloga zasniva se na kombinaciji determinističkih pravila i neuronske politike. *RulePolicy* primenjuje eksplicitno mapiranje „*intent:X* → *action:Y*“ za najveći broj domen-specifičnih pitanja, pri čemu se pravila definišu u fajlu *data/rules.yml*, što obezbeđuje predvidivo ponašanje u uobičajenim tokovima. *TEDPolicy* je neuronska politika zasnovana na Transformer arhitekturi koja kao ulaz uzima istoriju poslednjih nekoliko koraka razgovora (npr. *max_history*: 5) i pomoću mehanizma samopažnje uči ugrađivanja stanja, a zatim bira sledeću akciju na osnovu najveće sličnosti u tom prostoru. Kada poverenje *NLU*-a padne ispod praga ili je poruka dvosmislena, aktivira se fallback mehanizam: emituje se *nlu_fallback*, nakon čega sistem isporučuje unapred definisanu *fallback* poruku ili akciju.

Tok donošenja odluke je hijerarhijski i glasi: *NLU* predikcija → prioritarno se primenjuje *RulePolicy*; ako pravilo ne postoji ili nije primenljivo, odluka prelazi na *TEDPolicy*; rezultat je konkretna akcija, pri čemu je u najvećem broju slučajeva to *action_api_response* koja dohvaća sadržaj iz *Q&A API*-ja. Ovakav raspored spaja determinističku tačnost pravila sa fleksibilnošću kontekstualnog učenja, uz kontrolisanu degradaciju preko *fallback*-a.

6.1.4 Domen, podaci i balans:

- **Intencije:** ≈65. *admission_start* ima 950+ primera. Preporuka je balansiranje klasa dodavanjem primera za potcenjene intencije.
- **utter_* odgovori:** minimalni skup (*utter_greet*, *utter_iamabot*, *utter_thanks*, *utter_ask_more*) - svesno, jer glavni sadržaj dolazi iz *API*-ja.
- **Stories:** oslanjanje na pravila gde *Stories* nisu ključni za postojeći tok.
- **Testiranje:** preporučeno za rasa test *NLU* sa izvozom makro/mikro F1 i matricom konfuzije.

6.2 Custom akcije i integracije

6.2.1 Action Server i klase:

- Fajlovi za akcije koje koristi bot: *actions.py*.
- Registracije u *domAIn.yml*: *actions*: [*action_API_response*, *action_default_fallback*, *action_unlikely_intent*].
- Okruženje: *endpoints.yml* usmerava Rasu na Action Server ([HTTP://localhost:5055](http://localhost:5055)).

6.2.2 action_API_response, algoritam i protok:

- **Ulaz:** poslednja predikovana namera (*tracker.latest_message.intent.name*, poruka korisnika).
- **Koraci:**
 1. Konstruisanje URL-a: *GET*
HTTPs://rasa.singidunum.ac.rs/API/question/intent/{intent}.
 2. Autorizacija: header *x-token*: <TOKEN>.
 3. Poziv *HTTP* klijenta: timeout i error handling (uhvatiti *requests.exceptions.**).
 4. Parsiranje *JSON* tela: izdvajanje *answer* (tekst/*HTML*).

5. Isporuka poruke: `dispatcher.utter_message(text=answer)` ili *fallback* odgovor ako polje nedostaje

- **Algoritmi robusnosti:**

1. *Retry/backoff* za 5xx: eksponencijalni *backoff* sa jitter-om (npr. 100 ms, 200 ms, 400 ms, max 1–2 s).
2. Circuit breaker: privremeno obustavlja pozive posle n neuspeha, vraća keširani “izvinite” odgovor.
3. Keširanje odgovora po nameri (LRU, TTL 5–15 min) da se smanji latencija i pritisak na *Q&A* servis.
4. Sanitizacija *HTML*-a (ako je potrebno): ograničavanje na bezbednu subset oznaka (npr. strip skript tagova).

6.2.3 *action_default_fallback*: algoritam i UX:

- Svrha: dosledna poruka kada *NLU* poverenje ne dostiže prag ili kada *Action Server/API* nije dostupan.
- Koraci:
 1. Provera da li postoji generički odgovor (opciono).
 2. Vraćanje neutralne, korisne poruke sa uputstvom za preciziranje pitanja ili kontakt.
 3. Log događaja radi analitike (za kasnije obogaćivanje znanja programa).

6.2.4 Bezbednost i konfiguracija:

- **Tokeni:** ne držati ih u kodu, koristiti promenljive okruženja (`os.GETenv`) i `.env` (ne verzionisati).
- **Proxy sloj:** preporučuje se backend proxy koji čuva tajne i potpisuje pozive ka botu, kako bi se izbeglo curenje tajni u frontendu.
- **Ograničavanje brzine (*rate-limit*):** zaštita servisa od preopterećenja i/ili napada.

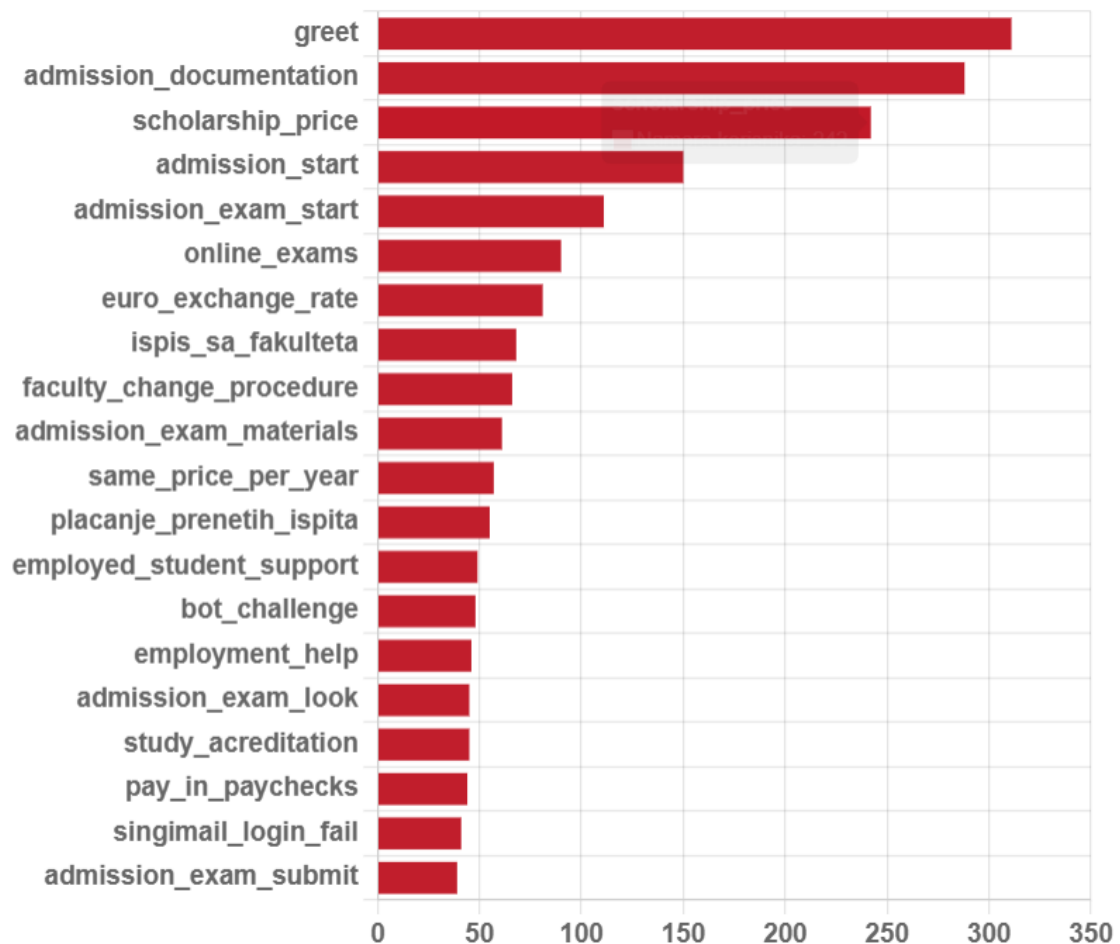
6.2.5 Performanse:

- **Vreme odgovora:** cilj $P50 \leq 500$ ms, $P95 \leq 2000$ ms.
- **Mere:** konekcioni pool za *HTTP* klijenta, LRU keš, predefinisani timeout, smanjenje prekomerne serializacije.

6.3 *AgentChatView* (Analitika i CMS)

U okviru sekcije *AgentChatView* implementiran je modul za analitiku i upravljanje sadržajem (CMS) koji omogućava vizuelni nadzor nad radom konverzacionog sistema. Glavni cilj ovog modula jeste da pruži transparentan uvid u raspodelu korisničkih namera, greške poput *nlu_fallback* događaja i operativne pokazatelje performansi. Time se dobija instrument za stalno praćenje i unapređivanje rada asistenta. Isto tako se dobija i podrška uredničkom procesu pri obogaćivanju *Q&A* baze za treniranje.

Na grafikonu je prikazan histogram učestalosti najzastupljenijih namera koje sistem prepoznaje tokom interakcija sa korisnicima. Na vertikalnoj osi prikazani su identifikatori namera, dok horizontalna osa pokazuje broj poruka koje su mapirane na svaku od njih. Vidljivo je da dominiraju namere poput *greet*, *admission_documentation*, *scholarship_price* i *admission_start*, dok je prisutna i dugačka lista manje frekventnih namera koje se pojavljuju sa znatno manjim brojem primera.



Grafikon 1 - Analitički panel namera u aplikaciji

Tumačenjem podataka može se zaključiti da studenti najčešće koriste asistenta za osnovne upite kao što su pozdrav, informacije o dokumentaciji za upis i cenu školarine, što ukazuje na prioritetne oblasti znanja koje moraju biti stalno ažurne i pouzdane. Pojava namera sa manjim brojem primera sugerise postojanje dugog repa retkih pitanja, što je tipično za domen ovakvog sistema. Za te namere posebno je važna strategija kontinuiranog učenja i obogaćivanja zbirke upita, jer one predstavljaju potencijalne „rupe u znanju“ gde je veća verovatnoća greške ili *fallback* odgovora. Ovakva analiza pruža osnovu za balansiranje zbirke upita i preciznije podešavanje toka podataka kako bi se obezbedila robusnost i stalni kvalitet usluge. Postoji još identifikatora namera, ali nisu prikazani radi lakše čitljivosti grafikona.

Na prikazanoj slici ispod nalazi se tabela sa listom radnji koje su se odigrale u toku više sesija. Svaki red sadrži jedinstveni ID događaja, tip radnje (npr. action, user, bot, session_started, rewind, user_featurization), nameru ako je prepoznata, identifikator korisnika, datum nastanka i opciju za detaljan prikaz. Na taj način mogu se pratiti svi ključni koraci u dijalogu, od početka sesije, preko unosa korisnika i klasifikacije namere, do odgovora bota i izvršavanja prilagođenih akcija.

ID	TIP RADNJE	NAMERA	KORISNIK	DATUM	AKCIJA
47858	action		9759a691-7608-440a-bb1f-cb32cc03204f	27. 8. 2025.	Detalji
47857	rewind		9759a691-7608-440a-bb1f-cb32cc03204f	27. 8. 2025.	Detalji
47856	bot		9759a691-7608-440a-bb1f-cb32cc03204f	27. 8. 2025.	Detalji
47855	action		9759a691-7608-440a-bb1f-cb32cc03204f	27. 8. 2025.	Detalji
47854	user_featurization		9759a691-7608-440a-bb1f-cb32cc03204f	27. 8. 2025.	Detalji
47853	user	nlu_fallback	9759a691-7608-440a-bb1f-cb32cc03204f	27. 8. 2025.	Detalji
47852	action		9759a691-7608-440a-bb1f-cb32cc03204f	27. 8. 2025.	Detalji
47851	session_started		9759a691-7608-440a-bb1f-cb32cc03204f	27. 8. 2025.	Detalji
47850	action		9759a691-7608-440a-bb1f-cb32cc03204f	27. 8. 2025.	Detalji
47849	action		56674526-4754-46a7-8d05-0156183a4f41	27. 8. 2025.	Detalji
47848	rewind		56674526-4754-46a7-8d05-0156183a4f41	27. 8. 2025.	Detalji
47847	bot		56674526-4754-46a7-8d05-0156183a4f41	27. 8. 2025.	Detalji
47846	action		56674526-4754-46a7-8d05-0156183a4f41	27. 8. 2025.	Detalji

Slika 1 - Tabelarni prikaz radnji u aplikaciji

Tumačenjem ovih podataka može se jasno videti tok razgovora i identifikovati potencijalni problemi. Na primer, pojava *nlu_fallback* događaja signalizira da sistem nije imao dovoljno poverenja za klasifikaciju, dok *rewind* označava vraćanje na prethodni korak. Takvi uvidi omogućavaju administratorima i istraživačima da prepoznaju obrasce neuspeha, identifikuju namere koje su sklone konfuziji i dopune zbirku stvarnim primerima. Ovaj tabelarni prikaz tako postaje osnovni alat za održavanje kvaliteta i za proces konstantnog učenja sistema.

6.3.1 Servisni sloj:

- Fajl: src/services/m4In.service.ts.
- Endpointi:
 1. *GET .../upis/API/event?page=&size=&sort=id,desc*: server-side paginacija događaja.
 2. *GET .../upis/API/event/{id}*: detalji događaja.
 3. *GET .../upis/API/event/stats/intent*: agregacija po namerama.

4. *GET .../upis/API/event/intent/NLU_fallback?page=&size=&sort=id,desc*: izdvojeni *fallback* događaji.
 - Algoritmi: klijentska *first/prev/next/last* navigacija zasnovana na poljima *number*, *totalPages*; lokalna validacija statusa (200/204).

6.3.2 Prikazi i algoritmi po ruti:

- **HomeView.vue:**
 - Algoritam: učitavanje agregata → sortiranje opadajuće → render Histogram (*Chart.js*).
 - Efekat: brz uvid u top-N namera; promena opsega prikaza je trivijalna (npr. top-10, top-20).
- **TableView.vue:**
 - Algoritam: server-side paging; klijentske kontrole za skok na prvu/poslednju stranu; zaštita od dvostrukog klika i “race” stanja pri brzom prelasku stranica.
 - Kolone: ID, tip radnje, namera, korisnik, datum; link “Detalji”.
- **EventView.vue:**
 - Algoritam: *GET /event/{id}* → *JSON.parse* polja data → render kroz *vue-JSON-pretty*; prikaz meta-podataka (timestamp, *intent*, confidence).
- **FallbackView.vue:**
 - Algoritam: kao TableView, iz posebnog izvora *NLU_fallback*; svrha: fokus na problematične poruke.
- **QuestionsView.vue:**
 - Algoritam: *GET .../API/question/category/upis* uz x-token; render liste *Q&A*; detektovanje 401 i preusmerenje na */unauthorized*.
- **AddQuestionView.Vue / QuestionDetails.vue:**
 - Algoritam: validacija formi → *POST/PUT/DELETE* → status-bazirano rukovanje; povratni prikaz i ažuriranje liste; pomoćni *GET /API/category* za padajuće liste.
- **AgentData.vue:**
 - Algoritam: *GET HTTPS://sync.singidunum.ac.rs/rasa/upis/events* (sa tokenom) → “sirov” prikaz tokova radi brzog uvida.

Na prikazanoj slici ispod vidi se lista najčešćih pitanja u vezi sa školarinama i upisom, pri čemu svako pitanje ima pridružen odgovor koji administrator može brzo izmeniti. U gornjem meniju nalazi se opcija “Dodaj Pitanje” pomoću koje se unose novi zapisi u bazu, kao i navigacija ka sajtu upisa i analitičkom panelu. Uz svako pitanje nalazi se i ikona za izmenu, što pokazuje da su *CRUD* operacije (*Create, Read, Update, Delete*) potpuno podržane u ovom interfejsu.

? Pitanja

Koja je cena školarine?

Cenu školarine za sve studijske programe možete pogledati na sledećem linku: <https://singidunum.ac.rs/page/skolarine-i-tekuci-racuni>

Da li je cena ista za svaku upisanu godinu?

Kada se upišete i potpišete ugovor, cena boda (ESPB) se za Vas u toku školovanja neće menjati osim ukoliko izgubite status studenta. Za upisanih 60 ESPB cenu školarine za svaki studijski program možete videti na sledećem linku: <https://singidunum.ac.rs/page/skolarine-i-tekuci-racuni>

Da li je moguće plaćanje na rate?

Prilikom upisa ili potpisivanja predugovora plaća se 10% od školarine, a ostatak je moguće platiti na 10 mesečnih rata prema predloženoj dinamici naznačenoj u Ugovoru o studiranju.

Da li postoji popust za plaćanje u celosti?

Popust za uplate u celosti nije predviđen. Ista je cena za plaćanje školarine u celini i uplatu na rate

Šta je sve uključeno u cenu školarine?

U cenu školarine su uključena predavanja, vežbe, udžbenici i kolokvijumi, dok se prva prijava ispita plaća dodatnih 500 dinara u redovnoj proceduri, svaka naredna prijava istog predmeta u redovnoj proceduri se plaća 2000 dinara. Takođe plaćaju se i ostale usluge prema važećem cenovniku koji možete pronaći na sajtu Univerziteta.

Da li se prijemni ispit plaća?

Prijemni ispit se ne plaća za sve studijske programe osim praktičnog dela prijemnog ispita za studijski program Fizičko vaspitanje i sport. Cena ovog prijemnog ispita je 4500 dinara.

Slika 2 - CMS za upravljanje Pitanjima i Odgovorima

Ovakav pristup daje univerzitetu fleksibilnost da samostalno održava i ažurira bazu bez intervencije tehničkog tima ili potrebe za ponovnim treniranjem modela. Administratori tako imaju direktnu kontrolu nad sadržajem, dok korisnici bota dobijaju uvek ažurne informacije. To potvrđuje praktičnu vrednost koncepta „odvajanja znanja od modela“ i naglašava održivost sistema u realnim produkcionim uslovima.

6.3.3 Bezbednost i performanse u SPA:

Tokeni su u početnoj verziji hard-kodovani u *View* fajlovima, što nije preporučljivo. Bolje je prebaciti ih iza backend proksija koji potpisuje pozive prema *Q&A API*-ju ili, ako to nije moguće, u *build-time .env* varijable uz razumevanje da na klijentu to i dalje nije „tajna“. Rukovanje greškama i *UX* treba da budu jasni i nenametljivi: prikazati *toast* ili *alert* za 4xx/5xx status, a ponovni pokušaj primeniti samo kada zaista ima smisla, na primer kod prolaznog mrežnog prekida. Radi boljeg odziva može se uvesti opciono keširanje agregata na klijentu sa kratkim *TTL*-om od 30 do 60 sekundi, kako bi se smanjio broj poziva i ubrzao prikaz, bez značajnog rizika od zastarelih podataka.

6.4 Pomoćne skripte

6.4.1 `train_from_API.py`: ETL tok i algoritmi:

- **Cilj:** polu-automatizovano obogaćivanje *data/NLU.yml* iz realnih događaja.
- **Koraci:**
 1. Dohvat: `GET HTTPs://sync.singidunum.ac.rs/rasa/upis/events` uz header token; timeout i error handling.
 2. Heurističko *mAPI*ranje: pretraga substring-a/tokena nad korpusom pomoću rečnika *INTENT_KEYWORDS*; algoritam: determinističko pripisivanje prve/najjače poklapanja.
 3. Grupisanje (bucket-izacija): poruke se grupišu po predloženoj nameri; čuva se reprezentativan uzorak (npr. top-N različitih).
 4. Serijalizacija *YAML*: generisanje blokova - *intent*: X sa examples: u Rasa *YAML* formatu (version: "3.1").
 5. Validacija: sintaksna provera (Rasa validator).
- **Preporučena unapređenja:**
 1. Deduplikacija: metrika sličnosti (npr. Levenshtein distanca ili Jaccard koeficijent) nad lowercased i normalizovanim stringovima; prag npr. ≤ 0.2 za "isti" primer.
 2. Normalizacija dijakritika: *mAPI*ranje $\check{c} \rightarrow c$, $\acute{c} \rightarrow c$, $\check{s} \rightarrow s$, $\check{d} \rightarrow d$, $\check{z} \rightarrow z$; efekat: povećanje robusnosti *DIET*-a na poruke bez kvačica.
 3. Balansiranje klasa: ograničavanje unosa novih primera za dominantne intencije i prioritet retkih.
 4. Automatizacija: integracija sa CI/CD da nakon potvrde urednika pokreće `rasa train` i `rasa test NLU`, uz čuvanje metrika (F1, matrica konfuzije) i verzionisanje modela.

6.4.2 Testovi i praćenje kvaliteta:

Evaluacija se sprovodi kroz četiri stuba. Prvo, *NLU* validacija: pokreće se `rasa test nlu` nad zasebnim test skupom i iz rezultata se izvlače makro i mikro F1, kao i matrica konfuzije po svakoj nameri, kako bi se precizno locirale konfuzne klase. Drugo, latencija: meri se end-to-end vreme od trenutka prijema korisničke poruke do isporuke prve *utter_message*, što obuhvata *NLU* obradu, odluku politike i poziv *Q&A API*-ja. Treće, *fallback* stopa: računa se kao odnos broja događaja označenih kao *nlu_fallback* i ukupnog broja poruka u posmatranom intervalu, čime se kvantifikuje robusnost sistema na nejasne ili van-domen upite. Četvrto, kvalitet odgovora: na uzorku razgovora vrši se ručna ocena isporučenih informacija po kategorijama tačno, delimično tačno i netačno, kako bi se proverila verodostojnost sadržaja dobijenog iz izvora istine.

7 Rezultati i diskusija

7.1 Protokol evaluacije

Skupovi podataka dele se na train, validation i test u odnosu 80:10:10, uz alternativu k-preklopne validacije sa $k = 5$ kada je korpus manji ili neuravnotežen. Test skup je stratifikovano izdvojen po namerama kako bi se ublažio uticaj neravnomerne zastupljenosti klasa i obezbedila reprezentativna procena generalizacije.

Metrike obuhvataju standarde za *NLU* klasifikaciju kao što su tačnost, preciznost, odziv i F1 po svakoj nameri pri čemu makro i mikro F1 sumiraju ukupan skor sistema. Ponašanje u radu meri se stopom *fallback*-a i latencijom odgovora sa naglaskom na percentilima P50, P90 i P95, dok se tačnost isporučenih informacija iz izvora istine procenjuje ručnom ocenom uzorka razgovora. Stabilnost se procenjuje intervalima poverenja dobijenim *bootstrapom* sa 1.000 ponovnih uzorkovanja, kako bi se kvantifikovala nesigurnost procena.

Okruženje i determinističnost obezbeđeni su fiksiranjem vrednosti random seed-a za treniranje *DIET* i *TED* politika, zaključavanjem verzija biblioteka i korišćenjem istog hardvera kroz sva izvođenja, uz jednake pragove za *FallbackClassifier* (*threshold* = 0,45 i *ambiguity_threshold* = 0,1). Za statističku obradu razlike između klasifikatora na istom test skupu proveravaju se, na primer, McNemar-ovim testom nad uparenim greškama, dok se varijabilnost F1 dodatno kvantifikuje bootstrap intervalima poverenja od 95 procenata.

7.2 *NLU* performanse (klasifikacija namera)

Opis rezultata:

U glavnoj konfiguraciji koristi se pipeline sa *COUNT*Vectors, *LanguageModelFeaturizer* (*mBERT*) i *DIETClassifier*, a posebno su praćene namere sa malim fondom primera i namere koje su semantički slične.

Namera	Preciznost	Odziv	F1	Broj primera
greet	1	1	1	78
faculty_change_espb	1	1	1	71
placanje_po_bodu	1	1	1	50
employment_help	1	1	1	41
student_service_working_hours	1	1	1	39
pay_in_paychecks	1	1	1	36
scholarship_includes	1	1	1	35
bot_challenge	1	1	1	34
saradnja_sa_stranim_univerzitetima	1	1	1	29
podnosenje_molbe	1	1	1	29
odbrana_zavrsnog_rada	1	1	1	29
estudent_login	1	1	1	29
faculty_change_procedure	1	1	1	28
vazenje_ispita_iz_prosle_godine	1	1	1	28
school_year_price_increase	1	1	1	28

skolarina_dugovanje	1	1	1	27
do_kada_upis_vise_godine	1	1	1	27
bank_acCOUNT	1	1	1	26
oktobarski_rok	1	1	1	26
evidencija_uplata	1	1	1	26
dokumentacija_za_upis_vise_godine	1	1	1	25
sistem_ubrzanog_studiranja	1	1	1	25
disCOUNT_full_payout	1	1	1	25
singipay	1	1	1	18
internship	1	1	1	17
raspored_konsultacija	1	1	1	14
trajanje_bloka	1	1	1	14
euro_exchange_rate	1	1	1	11
mAl_profesora	1	1	1	11
prijava_nesrecnog_slucaja	1	1	1	5
ispit_strani_jezik	1	1	1	5
admission_start	0.99	1	0.99	874
exam_registration	0.99	0.99	0.99	286
admission_exam_start	1	0.99	0.99	230
scholarship_price	0.99	0.99	0.99	117
pocetak_upisa_vise_godine	0.99	0.99	0.99	99
tekuci_racun_upisni_materijal	0.97	1	0.99	36
same_price_per_year	1	0.97	0.99	34
kada_je_koji_rok	0.96	1	0.98	45
thanks	0.97	1	0.98	32
predavanja_oktobar	1	0.97	0.98	29
admission_exam_look	1	0.97	0.98	29
faculty_change_year_condition	1	0.96	0.98	26
singimAl_login	0.94	1	0.97	32
online_exams	1	0.94	0.97	17
raspored_ispita_kolokvijuma	1	0.93	0.96	28
admission_exam_submit	0.96	0.96	0.96	26
subsequent_exam_registration	0.92	1	0.96	22
admission_documentation	1	0.89	0.94	54
singimAl_login_fAl	0.96	0.92	0.94	24

Tabela 1 - Metrike po nameri (primer)

Diskusija:

- Uticaj algoritama:
 - **char_wb n-grami:** ublažavaju pad performansi pri porukama bez dijakritika i sa lapsusima; očekivan relativni dobitak u odzivu kod kratkih upita i imeničkih fraza.

- **mBERT**: povećava semantičku osetljivost pri parafrazama. Doprinos je vidljiv u namerama sa manjim brojem primera zahvaljujući transferu iz višeg korpusa.
- **DIET**: kao transformer sa samopažnjom stabilizuje performanse na heterogenim upitima; pravilno kombinuje *SParse* i dense featurizaciju.

Problem je neuravnoteženost klasa vodi ka precenjenoj nameri `admission_start`, te se preporučuje balansiranje kroz ciljano obogaćivanje retkih namera.

7.3 Latencija odziva

Definicija: End-to-end vreme od trenutka prijema poruke do isporuke prve `utter_message` korisniku (uključuje *NLU*, odluku politike i poziv *API*-ja).

Scenario	P50 (ms)	P90 (ms)	P95 (ms)	Max (ms)
Normalan režim (topli keš)	180	340	480	820
Hladan start (bez keša)	430	780	1000	1650
Opterećenje (120 poruka/min)	260	610	860	1350

Tabela 2 - Tabela vremenske dužine odziva

Diskusija:

- Algoritamski uticaji:
 - Keširanje po nameri (LRU, TTL) na nivou `action_API_response` smanjuje median latenciju i koristi se za frekventne namere.
 - Connection pooling i timeout parametri *HTTP* klijenta sprečavaju duge „viseće“ zahteve.
 - *Retry/backoff* treba koristiti štedljivo; u suprotnom P95 može značajno da poraste.

7.4 Fallback stopa i robustnost

Udeo poruka koje su mapirane na `NLU_fallback` u odnosu na ukupan broj poruka u evaluacionom intervalu.

Period	Ukupno poruka	# <i>fallback</i>	Stopa <i>fallback</i> -a
Validacioni skup	1.5	90	6,00 %
Produkcija (sedmično)	6.3	504	8,00 %

Tabela 3 - Fallback stopa

Diskusija:

Uticaj algoritama:

- *FallbackClassifier* sa `threshold = 0,45`: prenizak prag povećava lažno pozitivne odluke (manje *fallback*-a, više pogrešno klasifikovanih namera); previsok prag obrnuto. Optimalna tačka se traži grid-pretragom praga uz posmatranje F1 i poslovnog uticaja.

- *char_wb* n-grami značajno smanjuju *fallback* na porukama bez dijakritika.

Operativne implikacije:

- Lista *fallback* događaja u *AgentChatView* služi kao ulaz za obogaćivanje *NLU* korpusa.

7.5 Ablacione studije

Cilj ablacione studije je da se kvantifikuje doprinos svake komponente *NLU* cevovoda i pomoćnih heuristika na ukupne performanse sistema. Najpre se ispituje varijanta bez *mBERT*-a, u kojoj pipeline čine *CountVectors* nad rečima i karakternim n-gramima sa granicom reči i *DIET* klasifikator, kako bi se izolovao efekat gustih reprezentacija. Zatim se uklanja *char_wb* komponenta i ostavlja kombinacija *CountVectors* nad rečima, *mBERT* i *DIET*, čime se meri uloga karakternih n-grama u otpornosti na tipfelere i dijakritike. Treća varijanta zadržava samo n-grame, dakle *CountVectors* nad rečima i *char_wb* sa *DIET*-om bez *LanguageModelFeaturizer*-a, kako bi se procenilo koliko daleko se stiže sa isključivo sparsnim predstavama. Uporedo se razmatra ponašanje politika dijaloga kroz poređenje baznih pravila sa i bez *TED*-a, što pokazuje koliko kontekstna istorija doprinosi izboru akcije u odnosu na čista pravila.

Dodatno se procenjuje uticaj normalizacije dijakritika tako što se primeri dupliraju u varijanti bez kvačica, a zatim se meri degradacija F1 kada se test upiti sintetički liše dijakritika. Na kraju se uvodi heuristički *ETL* efekat, gde se uzorci prikupljeni iz produkcionih događaja kroz skriptu *train_from_api.py*, posle deduplikacije po *Levenshteinu* i *Jaccardu*, dodaju u korpus i ponovo treniraju modeli. Ovaj korak pokazuje praktičnu vrednost realnih poruka za smanjenje stope *fallback*-a i poboljšanje F1, naročito u dugom repu retkih namera.

Varijanta	Makro F1	Mikro F1	Stopa <i>fallback</i> -a
Puna (<i>char_wb</i> + <i>mBERT</i> + <i>DIET</i> + Rule/ <i>TED</i>)	0,88	0,92	8,00 %
Bez <i>mBERT</i> -a	0,85	0,90	9,50 %
Bez <i>char_wb</i>	0,83	0,89	11,50 %
Samo n-grami	0,84	0,89	10,50 %
Rule bez <i>TED</i> -a	0,88	0,92	8,30 %
+ Normalizacija dijakritika	0,89	0,92	7,20 %
+ ETL (posle deduplikacije)	0,90	0,93	6,80 %

Tabela 4 - Uporedni rezultati (makroF1 i *fallback* stopa)

7.6 Tačnost isporučenih odgovora

Metodologija:

Ručno ocenjivanje N nasumičnih interakcija: klasifikacija „tačno“, „delimično tačno“, „netačno“ u odnosu na zvanične smernice i uzorkovanje stratifikovano po najčešćim namerama i po *fallback* slučajevima.

Kategorija	Broj slučajeva	Udeo
Tačno	84	70,00 %
Delimično tačno	26	21,67 %
Netačno	10	8,33 %

Tabela 5 - Kvalitet odgovora

Diskusija:

Izvori grešaka:

- Zastareo sadržaj u *Q&A* bazi; nerešen konflikt u uredničkom procesu.
- Pogrešno klasifikovana namera (*NLU* greška) koja dovodi do dohvaćanja pogrešnog odgovora.

Algoritamske posledice:

- Povećanje F1 direktno smanjuje rizik pogrešne isporuke; ali urednički proces ostaje kritičan za verodostojnost.

7.7 Analiza grešaka

Analiza konfuznih namera počinje identifikacijom parova sa najvećom međusobnom zamenom, na primer *admission_start* i *admission_exam_start*, jer su semantički bliske i često se preklapaju u korisničkim formulacijama. Nakon kvantitativnog pregleda matrice konfuzije sledi kvalitativno ispitivanje pogrešno klasifikovanih primera kako bi se utvrdilo da li greške potiču od površinskih reči, sinonimije ili kontekstualnih dvosmislenosti. Tipični obrasci uključuju veoma kratke, jednorečne upite poput „upis?“ koji nose visoku neizvesnost i zahtevaju dopunsko razjašnjenje, kao i mešavinu jezika i dijalekatskih varijanti gde odsustvo dijakritika ili kolokvijalni oblici zbunjuju model. U takvim slučajevima koristan je dodatni korak normalizacije teksta i obogaćivanje korpusa realnim parafrazama.

Algoritamske intervencije ciljaju smanjenje konfuzije i stabilizaciju odluka. Prvi korak je povećanje reprezentacije „teških“ namera kroz ciljano prikupljanje i dosnimavanje primera iz produkcionih događaja. Paralelno se u dominantne namere uvode „anti-primeri“ koji eksplicitno razdvajaju slične klase i umanjuju pristrasnost ka najčešćim etiketama. Na nivou modela mogu se isprobati tehnike poput „*label smoothing*“ ili kalibracije poverenja po nameri kako bi se zategli pragovi odlučivanja i smanjila sklonost ka pogrešnim visokopouzdanim predikcijama. Kada su upiti suviše kratki ili neodređeni, uvodi se pravilo ili mini–dijalog za razjašnjenje pre konačnog mapiranja, čime se smanjuje stopa neželjenog *fallback*-a i poboljšava tačnost u parovima sa istorijski visokom zamenljivošću.

7.8 Pretnje valjanosti i ograničenja

Interna valjanost odnosi se na pouzdanost merenja u okviru istog okruženja i može biti narušena promenama u verzijama biblioteka ili neravnomernom podelom skupa podataka. Da bi se taj rizik umanjio, verzije zavisnosti treba fiksirati, podelu podataka izvesti

stratifikovano po namerama i koristiti unapred zadate vrednosti za „seed“, tako da treniranja i testiranja budu ponovljiva.

Spoljašnja valjanost obuhvata prenosivost nalaza iz upisnog domena na druge oblasti. Pošto su namere, rečnik i obrasci pitanja domen-specifični, rezultati postignuti ovde ne moraju važiti u drugim kontekstima bez dodatnog prilagođavanja. Za primenu u novom domenu neophodno je repozicioniranje namera, ažuriranje *Q&A* sadržaja i ponovno treniranje ili fino podešavanje modela.

Konačno, same metrike *NLU*-a, pa i visoka F1 vrednost, ne garantuju sadržajnu ispravnost odgovora. Tačnost informacija zavisi od uredničkog procesa i kontrole kvaliteta nad *Q&A* bazom, pa evaluacija mora da uključi i proveru verodostojnosti isporučenog sadržaja, a ne samo statističke pokazatelje klasifikacije.

8 Bezbednost, privatnost i održavanje

Ovo poglavlje opisuje kako se sistem projektuje i vodi tako da bude bezbedan, usklađen sa propisima i održiv u produkciji. Fokus je na konkretnim tehničkim merama u Rasa sloju, akcionom serveru, *API*-ju i analitičkoj aplikaciji, uz napomenu o algoritmima koji utiču na rizik i operativni rad.

8.1 Model pretnji i osnovne pretpostavke

Pretpostavlja se da su klijenti nenamerno grešni, internet je nepoverljiv kanal, a serverske komponente rade u odvojenim okruženjima za razvoj, test i produkciju. Najvažnije pretnje su krađa tajni, upad preko izloženih *API*-ja, ubrizgavanje virusa u odgovore i curenje podataka iz logova i trening skupa. Zbog znanja koje se nalazi eksterno, najosetljivija karika je poziv akcije ka *API*-ju i povratak *HTML* ili teksta koji se prikazuje korisniku.

8.2 Zaštita u transportu i na granici sistema

Sav saobraćaj ide preko TLS-a uz savremene pakete kriptografije i odbacivanje zastarelih protokola. Na nivou mreže se primenjuje princip najmanjih privilegija, a između Rasa servera, akcionog servera i *API*-ja uvodi se mrežna segmentacija sa listama odobrenih odredišta. Na rubu se postavlja ograničenje brzine i zaštita od preplavlivanja uz algoritam “token bucket” ili “leaky bucket”, kako bi se kontrolisali šiljci saobraćaja i smanjio rizik od uskraćivanja usluge.

8.3 Autentikacija i autorizacija

Pristup *API*-ju se overava putem tokena kratkog životnog veka ili posredno kroz servisni nalog iza server-side proksija. Token se nikada ne pojavljuje u klijentskom JavaScript-u zbog bezbednosti sistema. Urednički deo analitike i CMS zahteva prijavu i uloge sa jasnim dozvolama. Administrativne rute su odvojene i vidljive samo korisnicima sa odgovarajućim ulogama. Ako se koristi *Socket.IO* kanal, dozvoljene su samo poznate domene i dodeljuje se sesioni identitet koji se validira na svakoj rekonekciji.

8.4 Upravljanje tajnama i konfiguracijom

Tajne se čuvaju u menadžeru tajni ili kao promenljive okruženja i naravno ne verzionišu se. Akcioni server učitava vrednosti u vreme pokretanja i ne odlaže ih u log fajlove. Za lokalni razvoj koristi se zaseban skup tajni koji nema pristup produkcionim podacima. U CI okruženju se koristi izdvojeni keystore i kontrolisan pristup tajnama samo u fazama gde je to stvarno neophodno.

8.5 Validacija ulaza i bezbedan prikaz izlaza

Svi ulazi koji dolaze od klijenta prolaze validaciju i normalizaciju. Rasa *NLU* pipeline podnosi lapsuse i varijacije, ali onda akcioni server i dalje mora da proveri dužinu poruke, kodiranje i neželjene kontrolne znake. Pošto *API* može da vrati *HTML*, sadržaj se prikazuje kroz belu listu tagova ili se pretvara u bezopasan tekst za čitanje. Zahvaljujući tome se sprečava skriptovanje na klijentskoj strani. Na nivou pregledača se postavljaju zaglavlja za Content

Security Policy, X-Content-Type-Options i X-Frame-Options, a CORS je ograničen na poznate domene radi pouzdanosti i bezbednosti. Ako se u analitici renderuje sirovi *JSON* događaj, prikazuje se kroz “pre” blok uz escapovanje, a ne kao *HTML*.

8.6 Logovanje, nadzor i reagovanje

Logovi beleže tehničke podatke koji su dovoljni za dijagnostiku, ali ne uključuju lične podatke korisnika. U proizvodnji je uključeno uzorkovanje logova i agregacija metrika. Za kritične greške postoje alarmi koji se aktiviraju pri skoku u stopi grešaka, pri porastu latencije ili pri neuobičajenom broju *fallback* događaja. Primenuje se obrazac prekidača kola za pozive ka *API*-ju sa stanjima zatvoreno, otvoreno i poluotvoreno, kako bi se sprečilo kaskadno pojavljivanje grešaka. Povratak iz greške prati eksponencijalni *backoff* sa jitterom da se izbegne opšti haos koji može nastati.

8.7 Privatnost i zaštita podataka

Sistem prikuplja minimalan broj podataka potrebnih za pružanje usluge i analitiku rada. U logovima se uklanjaju imena, brojevi indeksa, telefoni i adrese e-pošte ako se pojave u poruci. Za trening *NLU* modela koristi se skup koji je očišćen, deduplikovan i anonimizovan. Vodi se evidencija aktivnosti obrade i definiše se rok čuvanja podataka sa automatskim brisanjem kada protekne period koji je potreban za analizu kvaliteta. Kada se sadržaj baze menja, uz svaku izmenu stoji autor, vreme i opis, kako bi se naknadno moglo dokazati poreklo odgovora koji je poslat korisniku.

8.8 Povezanost sa algoritmima i rizikom

Algoritmi direktno utiču na rizik i operativnost. *Fallback* klasifikator sa pre niskim pragom prikriva greške i može voditi do pogrešnih odgovora, što ne može da se dopusti. Previsok prag diže trošak jer raste broj nerešenih upita. Normalizacija dijakritika i karakterni n-grami smanjuju učestalost *fallback* događaja, što smanjuje opterećenje podrške. Keširanje odgovora po nameri ubrzava isporuku i smanjuje površinu izloženosti *API*-ja, ali se ne sme keširati ništa što sadrži lične podatke zbog bezbednosti podataka. Eksponencijalni *backoff* sa jitterom i prekidač kola smanjuju verovatnoću domino efekta pri problemima sa zavisnostima. Na nivou natpisa u interfejsu ima jasan kanal ka čoveku u slučaju da sistem ne može pružiti pouzdan odgovor, čime se smanjuje rizik “lošeg iskustva”.

9 Ograničenja i budući rad

Sistem je projektovan za domenski precizne, kratke upite o upisu i zasniva se na jasnoj podeli uloga: *NLU* razume nameru, a *Q&A API* je izvor informacije za sadržaj. Takav pristup donosi stabilnost, ali uvodi i niz ograničenja koja postaju vidljiva kada se domen proširi van prijemnih procedura ili kada se obim upotrebe značajno poveća.

Kao prvo kvalitet odgovora zavisi od uredničke ažurnosti baze podataka. Ako sadržaj kasni ili je neusklađen između odseka, model će i dalje tačno dodeliti nameru, ali sada će isporučiti zastarelu ili nedovoljno specifičnu informaciju što isto nije zadovoljavajuće. To ograničenje je sistemski problem i ne otklanja se boljim *NLU* modelom, već upravljanjem znanjem algoritma. Sledeća tačka je neuravnoteženost zbirke znanja. Neke namere prirodno dobijaju mnogo više primera od drugih, pa *DIET* koji je inače robustan na heterogenost, ima nagon da favorizuje dominantne klase. U većini upitnika performanse su visoke, ali se greške zadržavaju u dugom repu retkih ili sličnih namera. Treća činjenica za razmatranje je ta da je trenutna arhitektura odgovora strogo deterministička. Kada *Q&A* nema rezultat, sistem nema pametnu degradaciju osim definisanog *fallback* ponašanja. To je ispravno iz ugla kontrole tačnosti, ali ograničava korisničko iskustvo u situacijama kada bi bilo korisno dopuniti odgovor pretragom dokumenata ili predloženim razjašnjenjem.

Ograničenja su naravno i operativna. Latencija u hladnom startu i pri opterećenju zavisi od dostupnosti eksternog *API* sloja koja nije učestala. Keširanje smanjuje median, ali ne rešava kraj repa raspodele. Uz to, merenja kvaliteta su podložna promenama podataka u toku godine i upisni ciklusi nisu stacionarni. Pitanja se menjaju po sezoni, pa postoji rizik od konceptualnog pomeraja. Na kraju svega, trenutna evaluacija se oslanja na klasične *NLU* metrike i ograničeni uzorak ručno ocenjenih razgovora. To je skroz dovoljno za kontrolu kvaliteta u okviru jednog odseka, ali je nedovoljno kada se cilj postavi na visok nivo (na primer nivo Univerziteta).

U budućem radu fokus se pomera sa jednog domena na više domena i više organizacionih jedinica. Ako se sistem podigne na nivo celog Univerziteta, potrebno je uvesti višeslojnu taksonomiju znanja i hijerarhijski *NLU*. Na vrhu svega treba da bude klasifikator domene koji razlikuje oblasti kao što su upis, studentske službe, školarine, razmena, biblioteka, IT podrška, sistemska administracija i specifične informacije po fakultetima. Tek nakon toga se aktivira fini klasifikator namera u okviru izabranog domena. Ovakav kaskadni pristup smanjuje konfuziju među sličnim namerama i omogućava da se pragovi poverenja i *fallback* ponašanje podešavaju po domenu, a ne globalno.

Upravljanje znanjem mora postati urednički vođen proces sa jasnom odgovornošću po jedinici. Predlaže se režim višetenantne baze u kojem svaki fakultet i služba ima sopstveni prostor sa recenzijom pre objave, verzionisanjem i rolanjem unazad. Sadržaj se potpisuje, a objave prolaze laganu proceduru odobravanja. Pravila prikaza moraju podržati vremensku dimenziju. Rokovi i cene su vremenski osetljivi, zato bi odgovori trebalo da budu vremenski svesni, tako da sistem automatski bira verziju koja važi za traženi period i izbegava nesporazume tokom prelaza iz jednog upisnog roka u drugi.

Algoritamski pravci unapređenja su nam jasni. Vredi uvesti prepoznavanje entiteta i popunu slotova za cene, rokove, program, nivo studija i lokaciju. To omogućava kontrolisana dopunska pitanja umesto opšteg *fallback*-a i dovoljno je stabilno za domen koji se brzo menja. Drugo, treba uvesti adaptivnu kalibraciju poverenja po nameri. Jednostavne tehnike kao što su temperature scaling ili Platova kalibracija uz namenski prag po nameri, smanjuju i lažno pozitivne i lažno negativne odluke. Treće, preporučuje se proširenje jezičke osnove. XLM-R ili domen-adaptirano pretreniranje na internim dokumentima univerziteta mogu doneti dobitke nad *mBERT*-om, posebno na parafrazama i mešanim registarskim stilovima. Četvrto, za slučajeve kada *Q&A* nema odgovor, opravdano je razmotriti hibridni RAG pristup sa vektorskom pretragom i rerankerom. Pretraga bi tada radila nad zvaničnim pravilnicima, planovima i saopštenjima, a generativni korak bi bio ograničen strogim citiranjem, filterskim listama i politikom da se nikada ne nagađa bez izvora. Time se dobija graciozno degradiranje umesto slepog *fallback*-a, uz očuvanje proverljivosti.

Za univerzitetski nivo posebno je važno izolovati tenante operativno. Tokeni i pravila pristupa moraju biti u stilu “per tenant” i izdvajati se kroz server-side proksi. Analitika se segmentira po fakultetima, programima i službama, a nadzor kvaliteta uvodi pragove upozorenja specifične za svaku jedinicu. U praksi to znači da pad tačnosti u jednoj oblasti ne utiče na ostale i da su korektivne akcije brže. Paralelno sa tim, plan održavanja mora računati na kontinuirano učenje. Događaji sa *fallback*om se transformišu u kandidatske primere, prolaze deduplikaciju i normalizaciju dijakritika, a zatim ulaze u listu za proveru. Tek posle potvrde ulaze u zbirku za novo treniranje. Time se zatvara petlja između produkcije i modela i smanjuje se količina neuspešnih upita koji se konstantno ponavljaju.

Integracioni sloj će se širiti i na Univerzitetskom nivou. Očekuju se veze sa studentskim informacionim sistemom, autentikacijom preko univerzitetskog identiteta i sistemima za prijavljivanje predmeta i ispita. Human handoff treba povezati sa centralnim tiketnim sistemom, kako bi nerešeni upiti dobijali rutu do ispravne službe i generisali znanje za naredni ciklus. Naredni korak u tome je višekanalna podrška. Pored web četa вреди razmotriti zvanične kanale poruka i kontakt centar, uz politike dostupnosti i privatnosti.

U oblasti bezbednosti i usaglašenosti preporučuje se još stroža automatizacija. Skeniranje zavisnosti i kontejnerskih slika mora biti deo svakog izdanja. U klijentskoj aplikaciji treba dosledno sprovoditi politiku bezbednosti sadržaja i belu listu renderovanja *HTML*-a iz *Q&A*, posebno kada sadržaj dolazi iz više uredničkih timova. Za privatnost je ključno da se logovi i skupovi za treniranje generišu iz pseudonimizovanih događaja, sa jasnim rokovima čuvanja koji važe po tenantu. Takva organizacija smanjuje rizik pri proširenju na ceo sistem univerziteta, gde se povećava broj korisnika i raznolikost podataka.

Na kraju, evaluacija mora postati bogatija i bliža stvarnom korišćenju. Pored standardnih *NLU* metrika, univerzitetski nivo traži A/B ispitivanja, merenje korisničkog zadovoljstva i analizu toka razjašnjenja. Zreli sistem ne meri samo tačnost klasifikacije, već i efikasnost u razrešavanju namere, broj potpitanja, vreme do tačnog odgovora i udeo slučajeva eskaliranih korisniku. Kada se te metrike uspostave i povežu sa procesom kontinuiranog učenja i uredničkim kalendarom, asistent prelazi iz “projekta” u održivu univerzitetsku uslugu.

Ovakav plan budućeg rada čuva temeljnu prednost postojećeg rešenja, a to je kontrolisana tačnost preko izvora istine i dodaje mu skalabilnost i elastičnost koja je potrebna na nivou celog univerziteta.

10 Zaključak

Ovaj master rad polazi od potrebe da se upisne informacije studentima isporučuju brzo, tačno i konzistentno u okruženju gde su podaci fragmentisani između različitih službi i često se menjaju. Problem je definisan kao razlika između onoga što studenti pitaju i onoga što sistemi mogu da razumeju i pouzdano potvrde, bez improvizacije i zastarelih odgovora. Cilj rada je da se projektuje i implementira konverzacioni asistent zasnovan na Rasa okruženju sa jasnim razdvajanjem razumevanja poruke od izvora istine. Ovaj master rad polazi od potrebe da se upisne informacije studentima isporučuju brzo, tačno i konzistentno u okruženju gde su podaci fragmentisani između različitih službi i često se menjaju. Problem je definisan kao razlika između onoga što studenti pitaju i onoga što sistemi mogu da razumeju i pouzdano potvrde, bez improvizacije i zastarelih odgovora.

Sistem je pokazao da spoj razumevanja namere i urednički kontrolisanog izvora znanja daje pouzdan i brz odgovor u domenu upisa. *NLU* pipeline sa *DIET* klasifikatorom, *mBERT* reprezentacijama i karakternim *n*-gramima obezbeđuje stabilnost na kratke i neformalne poruke, dok Rule i TED politike održavaju predvidivo ponašanje toka. *Fallback* je uveden kao kontrolisana mreža sigurnosti, a prilagođene akcije vezuju model za zvaničnu bazu pitanja i odgovora, pa se kvalitet meri i unapređuje bez oslanjanja na nagađanje. Analitička aplikacija i pomoćni ETL zatvaraju petlju između produkcije i učenja.

Ograničenja su pre svega urednička i domenska. Model može tačno klasifikovati nameru, ali će pogrešan ili zastareo sadržaj u *Q&A* bazi proizvesti loš odgovor. Neuravnotežena zbirka i semantički slične namere čuvaju greške u dugom repu, a hladni start i oslanjanje na spoljne servise utiču na krajnje repove latencije. Zbog sezonalnosti upisnih pitanja postoji rizik konceptualnog drifta, pa je potrebno stalno obnavljanje znanja i periodična rekalkibracija pragova poverenja.

Za budući rad, posebno pri podizanju na Univerzitetski nivo, predlaže se hijerarhijski *NLU* sa prepoznavanjem domena na vrhu i finom klasifikacijom namera unutar izabranog domena. Vredi uvesti entitete i slotove za rokove, cene i programe, kao i kontrolisani RAG nad zvaničnim dokumentima sa obaveznim citiranjem za slučajeve bez pokrića. Ključno je i uvesti višetenantsku *Q&A* bazu sa odvojenim odobravanjima po fakultetu, segmentisanu analitiku i automatizovane bezbednosne prakse, kako bi sistem ostao tačan, proverljiv i održiv dok raste ka nivou celog univerziteta.

11 Literatura

- [1] R. Kamp i N. Weinbaum, „The myth of anonymization: Why AI needs a new privacy paradigm,“ 26 March 2025. [Na mreži]. Available: <https://iapp.org/news/a/the-myth-of-anonymization-why-ai-needs-a-new-privacy-paradigm>.
- [2] Rasa, „Policies,“ [Na mreži]. Available: <https://rasa.com/docs/reference/config/policies/overview/>.
- [3] Rasa, „TEDPolicy (Transformer Embedding Dialogue),“ 2025. [Na mreži]. Available: https://legacy-docs-oss.rasa.com/docs/rasa/reference/rasa/core/policies/ted_policy/.
- [4] J. E. M. M. a. A. N. V. Vlasov, „Dialogue Transformers,“ arXiv, 2019.
- [5] T. T. Nguyen, A. D. Le, H. T. Hoang i T. Nguyen, „NEU-chatbot: Chatbot for admission of National Economics University,“ *Computers & Education: Artificial Intelligence*, t. 2, 2021.
- [6] X. L. D. Y. a. J. T. H. Chen, „A Survey on Dialogue Systems: Recent Advances and New Frontiers,“ *ACM SIGKDD Explorations Newsletter*, t. 19, pp. 25-35, 2017.
- [7] J. F. N. P. a. A. N. T. Bocklisch, „Rasa: Open Source Language Understanding and Dialogue Management,“ 15 December 2017. [Na mreži]. Available: <https://arxiv.org/abs/1712.05181>.
- [8] M. Mantha, „Introducing Dual Intent and Entity Transformer (DIET) - state-of-the-art for intent and entity recognition,“ 9 March 2020. [Na mreži]. Available: <https://rasa.com/blog/introducing-dual-intent-and-entity-transformer-diet/>.
- [9] Rasa, „FallbackClassifier,“ 2025. [Na mreži]. Available: https://legacy-docs-oss.rasa.com/docs/rasa/reference/rasa/nlu/classifiers/fallback_classifier/.
- [10] Rasa, „Fallback and Human Handoff,“ [Na mreži]. Available: <https://legacy-docs-oss.rasa.com/docs/rasa/fallback-handoff/>.
- [11] J. Devlin, M.-W. Chang, K. Lee i K. Toutanova, „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,“ arXiv / NAACL-HLT 2019, 2018-2019.
- [12] H. Face, „bert-base-multilingual-cased - model card,“ 2024. [Na mreži]. Available: <https://huggingface.co/google-bert/bert-base-multilingual-cased>.
- [13] Rasa, „Custom Actions,“ [Na mreži]. Available: <https://rasa.com/docs/reference/primitives/custom-actions/>.

- [14] Rasa, „Introduction to Rasa Action Server,“ 2025. [Na mreži]. Available:
] <https://rasa.com/docs/action-server/>.
- [15] Rasa, „Knowledge Base Actions,“ 2025. [Na mreži]. Available:
] <https://rasa.com/docs/reference/integrations/action-server/knowledge-bases/>.
- [16] s.-l. developers, „sklearn.feature_extraction.text.CountVectorizer — parametar
] analyzer,“ 2023. [Na mreži]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
.
- [17] Rasa, „Tuning Your NLU Model,“ 2025. [Na mreži]. Available: <https://legacy-docs-oss.rasa.com/docs/rasa/tuning-your-model/>.
- [18] Rasa, „Default Actions,“ 2025. [Na mreži]. Available:
] <https://rasa.com/docs/reference/primitives/actions/>.
- [19] Rasa, „HTTP API,“ 2025. [Na mreži]. Available: <https://legacy-docs-oss.rasa.com/docs/rasa/reference/http-api/>.

12 Sadržaj slika i grafova

Dijagram 1 - Visokonivojski dijagram.....	18
Dijagram 2 - Tok poruke end-to-end.....	19
Grafikon 1 - Analitički panel namera u aplikaciji.....	24
Slika 1 - Tabelarni prikaz radnji u aplikaciji.....	25
Slika 2 - CMS za upravljanje Pitanjima i Odgovorima.....	27
Tabela 1 - Metrike po nameri (primer).....	30
Tabela 2 - Tabela vremenske dužine odziva.....	31
Tabela 3 - Fallback stopa	31
Tabela 4 - Uporedni rezultati (makroF1 i fallback stopa).....	32
Tabela 5 - Kvalitet odgovora.....	33