Universitatea Politehnica București

Facultatea de Automatică și Calculatoare

București, 2021

AGENT CONVERSAȚIONAL ÎN LIMBA ROMÂNĂ

Absolvent:

Silviu-Gabriel Gavriluță

Profesor îndrumător:

Prof. dr. ing. Mihai Dascălu

As. drd. ing. Dragoș Corlătescu

În timpul anilor de studiu fiecare student are nevoie de informații administrative sau legate de programul de la facultate. Obținerea acestora poate fi uneori laborioasă, iar pentru a putea fi simplificată, furnizarea de informații ar trebui să fie făcută într-un mod automatizat. Datorită progreselor din ultimul timp, făcute de către tehnologiile din domeniul înțelegerii limbajului natural, acest lucru poate fi făcut cu ajutorul unui agent conversațional. Odată cu admiterea în mediul universitar, studentul se poate confrunta cu mai multe nelămuriri: „*Ce programe trebuie să îmi instalez pentru a-mi desfășura activitatea online?”, „Cum pot să îmi aleg colegul de cameră?”, „Există vreo modalitate să îmi schimb grupa?”, „În ce sală are loc cursul de Programarea Calculatoarelor?”.* Agentul implementat va fi capabil să le răspundă studenților la aceste întrebări. Pentru a putea fi folosit de cât mai mulți studenți, agentul va fi integrat în aplicația facultății, ACS UPB Mobile.

Această lucrare prezintă etapele din implementarea agentului conversațional, cercetarea făcută pentru realizarea acestora și rezultatele obținute. În urma testării s-a constatat că modelul rezultat este capabil să răspundă corect la întrebările primite de la utilizator. Acest lucru este demonstrat în urma vizualizărilor graficelor, care arată că intențiile din întrebări sunt identificate corespunzător. Totodată, în urma discuțiilor cu viitorii utilizatori, am ajuns la concluzia că interfața grafică realizată în aplicație este ușor de folosit iar interacțiunea cu acesta poate fi de folos unui student în viața de zi cu zi.

Cuprins

CAPITOLUL 1. Introducere

1.1. Motivație

Față de liceu, viața unui student se schimbă radical: cursurile se fac fiecare în săli diferite, majoritatea nu mai stau acasă cu părinții, organizarea este diferită (grupe, semigrupe, serii), cu o parte administrativă diferită. Aceste schimbări au nevoie de clarificări, creează multe întrebări. De obicei răspunsurile se află cel mai des de la colegi sau de la cadrele didactice. Odată cu trecerea în regim online a facultății, pentru o perioadă nedeterminată, acestea au fost mai greu de găsit, răspunsurile colegilor venind mai rar, deoarece se rezumau la rețelele de socializare.

În zilele de astăzi, concepte precum inteligență artificială, învățare automată, procesarea limbajului natural nu mai sunt o țintă greu de atins pentru oamenii care lucrează în dezvoltarea de software și în știința calculatoarelor. Există documentații pentru fiecare domeniu în parte, s-au dezvoltat framework-uri ușor de folosit, toate acestea ducând la o creștere semnificativă a folosirii inteligenței artificiale în aplicațiile de zi cu zi.

Pentru a eficientiza aflarea informației, și ținând cont de ascensiunea domeniului de procesare a limbajului natural, am decis că cea mai bună metodă este implementarea unui agent conversațional, unde utilizatorul va putea adresa întrebări acestuia folosind o pagină de chat. Răspunsurile întrebărilor vor veni mult mai rapid (în jur de o secundă) în comparație cu cele primite pe aplicațiile de socializare, iar gradul de încredere a celor furnizate de agent este mai mare, nemaiavând problema erorii umane.

1.2. Context

Agentul conversațional va fi integrat în aplicația facultății de Automatică și Calculatoare, ACS UPB Mobile, unde va putea fi folosită de orice student.

1.2.1. Aplicația ACS-UPB-mobile

ACS UPB Mobile[1] este o aplicație începută inițial de Ioana Alexandru[2], în jurul căreia s-a format o comunitate de studenți, care ajută la dezvoltarea de noi funcționalități, numită StudentHub[3]. Aceasta are ca scop principal ușurarea modului de organizare a informațiilor necesare unui student, cu privire la viața academică, prin punerea acestora într-un singur loc.

[1] https://github.com/student-hub/acs-upb-mobile
[2] https://ioana-alexandru.com/about/
[3] https://github.com/student-hub

Folosindu-ne de faptul ca framework-ul flutter este cross-platform, aplicația funcționează pe web[4], Android[5] și iOS.

În cadrul ACS UPB Mobile, studentul își poate selecta seria și grupa din care face parte, pentru ca informațiile să fie filtrate în funcție de acestea. Odată intrat în aplicație, utilizatorul are acces la evenimentele viitoare din cadrul facultății, anunțurile oficiale preluate de pe site-ul facultății și la site-urile pe care le-a accesat cel mai des în aplicației. Una dintre funcționalitățile cele mai folosite este orarul, care este filtrat în funcție de datele studentului, existând posibilitatea de completare a materiilor suplimentare – opționale, restanțe. Totodată, sunt disponibile datele despre profesori care sunt publice, luate de pe site-ul departamentului de calculatoare[6], implementate într-o pagină dedicată din aplicație, unde se poate vizualiza numele, biroul, numărul de telefon și poziția ocupată în cadrul facultății. O altă pagină importantă este cea a site-urilor folosite în procesul didactic, cum ar fi cele folosite la cursuri sau laboratoare, cele necesare unui student pentru realizarea temelor (programe necesare, documentații), dar și site-uri care ajută studentul să se integreze într-un mediu social, precum asociațiile studențești sau canale de comunicare pentru jocuri sau alte activități.

Datele din aplicație se bazează pe munca unei comunități de studenți, care adaugă orarele la începutul fiecărui semestru. Studenții care doresc să contribuie cu date, completează un formular în care solicită acest drept. Contribuțiile sunt monitorizate, fiecare eveniment, având asociat și utilizatorul care l-a adăugat. În cazul unor greșeli repetate, utilizatorul își poate pierde dreptul de a contribui. Evenimentele, punctajele de la materii sunt actualizate în fiecare an universitar, sau când este semnalat de către cineva.

1.2.2. Integrarea agentului conversațional

Agentul va fi integrat în pagina de căutare a aplicației. Pentru o eficientizare a experienței utilizatorului, acesta va fi apelat doar dacă utilizatorul nu va găsi răspunsul la ceea ce caută prin intermediul rezultatelor oferite de algoritmii implementați deja. Utilizatorul va putea adresa întrebări în pagina de chat, urmând ca agentul să o proceseze și să îi ofere răspunsul dorit.

[4] https://acs-upb-mobile.web.app/
[5] https://play.google.com/store/apps/details?id=ro.pub.acs.acs_upb_mobile
[6] https://cs.pub.ro/index.php/?option=com_comprofiler&task=userslist&listid=2

1.3. Funcționalități cheie

Prin intermediul agentului conversațional, studentul va putea obține informații despre facultate:

- Locația sălilor de curs, a evenimentelor
- Data de desfășurare a evenimentelor
- Punctajul de la materii, numărul minim de prezențe
- Informații referitoare la șeful de grupă/serie
- Cursurile profesorilor
- Informații administrative: costuri de cazare la cămin, alegerea colegilor de cameră, programul la secretariat, modele de cereri, atribuțiile unui șef de grupă/serie, burse etc

Folosind pagina de căutare, utilizatorul va putea afla date despre cadrele didactice: birou, numărul de telefon, profesia (datele care sunt făcute publice de către aceștia); Totodată, căutarea îi va returna și o serie de materii, în funcție de cuvintele căutate, având la dispoziție, prin apăsarea pe aceste rezultate, să îi fie prezentate informații extra (locul de desfășurare, punctaj, materiale suplimentare.

1.4. Obiective

Prin intermediul chatbot-ului vrem ca utilizatorului să i se răspundă corect și rapid la întrebări. Interacțiunea cu acesta trebuie să fie una ușoară și intuitivă, răspunsurile oferite de agent să nu fie repetitive iar informația oferită să fie cât mai exactă în conformitate cu întrebarea pusă.

Dorința asociației în curs de formare, StudentHub, din cadrul căreia face parte aplicația, este de a se extinde și la alte universități. Astfel, ne dorim ca modul de antrenare al agentului să fie ușor de adaptat la alte informații, majoritatea datelor să fie extrase automat din baze de date, urmând să fie scrise manual doar informațiile specifice unei anumite universități, cum ar fi cele administrative.

1.5. Structura tezei

Prima etapă este cea de cercetare, unde ne m-am documentat despre tehnologiile si framework-urile existente în acest domeniu, și modul lor de implementare.

În a doua etapă am făcut cercetare pe tehnologiile folosite pentru crearea interfeței cu utilizatorul.

Etapa a treia o reprezintă implementarea interfeței cu utilizatorul, în aplicația ACS UPB Mobile, și a corpus-ului agentului conversațional (intenții, acțiuni), alegerea arhitecturii și antrenarea acesteia.

În etapa a patra am interpretat rezultatele obținute, am testat agentul și am analizat performanțele

Etapa a cincea este reprezentată de discuțiile cu viitorii utilizatori referitor la părerea acestora despre modul de interacțiune cu agentul, ușurința obținerii rezultatelor dorite de la acesta și corectitudinea lor.

Ultima etapa, a șasea, conține concluziile desprinse din urma implementării acestora, prezentarea metricilor obținute și strategii de îmbunătățire a implementării.

CAPITOLUL 2. State of the art

2.1 Prezentare generală

Interesul pentru cercetarea în domeniul procesării limbajului natural a crescut foarte mult în ultimii ani datorită dorinței de automatizare a cât mai multor domenii. Prin folosirea procesării limbajului natural s-a început automatizarea sarcinilor care implică discuția cu o persoană, cum ar fi cele din relațiile cu clienții, e-commerce și mai nou, în domeniul medical și al sănătății publice.

2.2 Agenți conversaționali

Pentru a putea implementa agenți conversaționali este necesar ca prima dată să înțelegem ce presupune un dialog între oameni, cât este de complex. Într-un dialog, interlocutorii își dau seama în funcție de context sau replici, dacă dialogul s-a încheiat, la fel și un agent, trebuie să fie capabil să prezică momentul când un dialog se termină. Oamenii își transmit unii altora mesaje, în momentul discuției, pentru a se asigura că se înțeleg unii cu alții, cum ar fi „Ok! Am înțeles!” sau „Poți să repeți te rog întrebarea?”. Agenții conversaționali moderni trebuie să fie capabili să adreseze sau să răspundă la astfel de întrebări, pentru a fi siguri că s-a înțeles contextul dialogului, subiectul și informația transmisă.

2.2.1 Agenți conversaționali – Statistici 2021

Agenții conversaționali au evoluat foarte mult, de a lungul anilor, la fel ca părerea consumatorilor asupra capacităților acestora de procesare. Primul chatbot a fost Eliza, realizat în anul 1966 de către Joseph Weizenbaum[7], care, deși a picat majoritatea testelor, a fost o piatră de temelie pentru această tehnologie. În vremea implementării primului chatbot oamenii erau destul de reticenți la faptul că mașinăriile vor fi capabile să le răspundă prompt la întrebări. Cu timpul, părerea oamenilor s-a mai schimbat, 27% dintre utilizatori sunt interesați de echipamentele care utilizează inteligența artificială.

[7] https://en.wikipedia.org/wiki/Joseph_Weizenbaum

Astăzi, deoarece 80% dintre agenții conversaționali de pe piață pot răspunde la întrebări standard din domeniul în care activează, numărul persoanelor care îi folosesc a ajuns la aproximativ 1.4 miliarde, iar în fiecare domeniu, numărul acestora se preconizează să crească foarte mult, statistică prezentată în Fig. 1. Interesul față de aceștia a crescut cel mai mult în anul 2018, aproximativ cu 160%. Cei mai mulți dintre utilizatori (aproximativ 64%) consideră că cel mai folositor loc de utilizare al unui chatbot este în domeniul de relații cu clienții. Se preconizează că sistemul bancar va automatiza peste 90% din interacțiunile cu clienții folosind agenți conversaționali până în anul 2022.

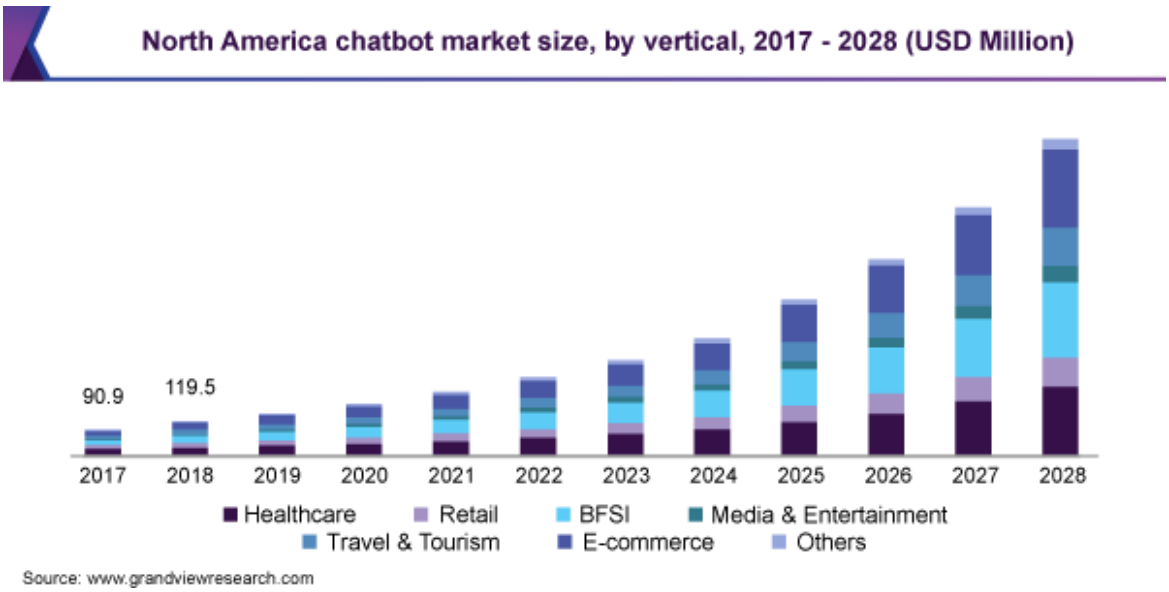


Fig. 1 Statistică privind evoluția chatboților pe piață din 2017 până în 2028

2.2.2 Chatboți corpus-based

Pentru a construi chatboți corpus-based avem nevoie de foarte multe date, peste sute de mii de cuvinte pentru etapa de învățare. De-a lungul timpului s-au folosit la antrenare seturi de date din conversații la telefon, subtitrări de filme, relatări de dialoguri între persoane etc.

Aceștia produc răspunsurile la întrebări în contextul dialogului, folosind metode de regăsire (caută răspunsul potrivit în corpus) sau de generare (generează răspunsul folosind un model de limbă sau un codificator-decodificator). Ei se numesc sisteme generatoare de răspuns, deoarece sistemele generează un singur răspuns, în oricare dintre cele două cazuri. Metoda de generare folosind un model de limbă se poate implementa antrenând un model pe un set de date care conține conversații iar acesta să faca direct predicții asupra răspunsului. Pentru a crește experiența utilizatorilor cu acești chatboți, aceștia pot fi antrenați și folosind

alte surse de text, existând încercări de-a lungul vremii de antrenare cu reviste, articole sau site-uri de informații, precum Wikipedia[8].

Există și arhitecturi hibride, bazate pe combinarea unor elemente din celelalte arhitecturi. Spre Exemplu, sistemul Chirpy Cardinal, unde erau combinate legarea entităților cu Wikipedia (specifică entitatea care este adusă în discuție), clasificarea intențiilor utilizatorului (când se încearcă schimbarea subiectului) și clasificarea actului de dialog (pentru a afla tipul de enunț – întrebare/răspuns și tipul răspunsului – negativ/pozitiv)

2.2.3 Agent conversațional folosind transfer learning

Agentul conversațional va avea o bază de cunoștințe pentru a putea descrie (persona) și un istoric al dialogului, cum ar fi în Fig. 2. Atunci când primește o întrebare, acesta va genera răspunsul folosindu-se de enunțul primit și a cunoștințelor de bază.

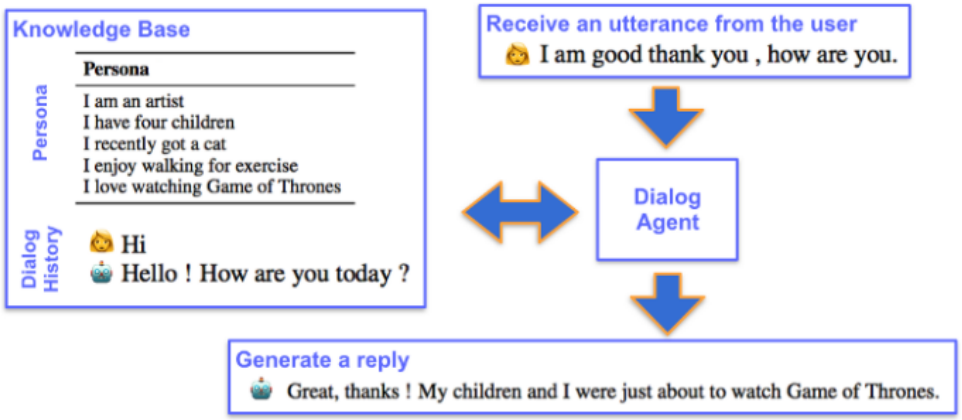


Fig. 2 Exemplu Agent Conversațional-Persona-Enunț-Replică

Din cauză ca seturile de date sunt mici, este greu ca un agent să învețe destul pentru a putea funcționa corespunzător. Pentru a rezolva această problemă se poate folosi transfer learning. Pentru început se pre-antrenează un model pe un set mare de date, astfel încât să poată fi capabil să genereze mult text. Apoi, acest model este ajustat pentru a putea fi folosit într-un act de dialog.

Pentru a ușura implementarea se pot alege modele deja pre-antrenate, de preferat cu cât mai mult text, care sunt capabile să genereze o distribuție a probabilităților peste vocabular, a tokenilor primiți ca input și să prezică următorul token. Spre exemplu, se pot folosi modelele GPT(Generative Petrained Transformer) sau GPT2, care sunt implementate folosind Transformer. Acestea folosesc contextul din stânga, pentru a prezice următorul

[8] https://ro.wikipedia.org/wiki/Pagina_principal%C4%83

cuvânt. Deoarece pre-antrenarea este costisitoare, se poate opta pentru model și tokenizer pre-antrenate de OpenAI.

Inputul pe care îl primește modelul este o secvență de cuvinte, dar acesta trebuie să țină cont de mai multe contexte: persona, istoricul dialogului și cuvintele care au fost deja generate. Soluția pentru aceasta este concatenarea segmentelor(persona + istoric + replici). Acest lucru aduce două probleme - delimitările segmentelor sunt greu de aflat și avem nevoie să știm poziția tokenilor pentru calcularea atenției. Pentru a rezolva aceste probleme se introduc doua noi secvențe, paralel, una care să conțină pozițiile tokenilor și una pentru delimitarea segmentelor.

Implementarea de bazează pe un model „Double-Head”: într-o parte se vor calcula predicțiile de modelare a limbajului iar celalat va prezice etichetele următorului enunț. Pentru interacțiune, avem nevoie de un decodificator, care v-a construi secvențele pe baza predicțiilor făcute. Greddy-decoding este un decodificator care la fiecare pas selectează tokenul cu probabilitatea cea mai mare, pană când se ajunge la tokenuri de sfârșit de secvență. Prin aceasta metodă, există riscul să pierdem un token cu probabilitate mare, daca aceste se află lângă un token cu probabilitate mică. Beam-search atenuează această problemă, deoarece construiește mai multe secvențe de cuvinte, iar la sfârșitul procesului se alegea secvența cea mai buna. În schimb, beam-search prezintă probleme la folosirea în dialog, deoarece enunțuri de diferite lungimi pot avea același înțeles. Astfel, s-a introdus k-sampling, unde decodificatorul alege numai tokenurile cu probabilitatea ai mare de pragul k, care este dat ca hiper-parametru.

2.3 RASA

Rasa este un framework open-source folosit pentru a crea boți pentru diverse taskuri; se bazează pe înțelegerea mesajului utilizatorului, oferirea răspunsului așteptat de acesta și întreținerea unui dialog coerent. Acesta este împărțit în doua framework-uri distincte, Rasa NLU, care se ocupă de înțelegerea textului și Rasa Dialogue Management, care procesează viitoarele acțiuni pe care le va efectua agentul în urma analizei textului.

2.3.1 Rasa NLU (Natural Language Understanding)

Rasa NLU se ocupă cu înțelegerea limbajului, identifică ce informație dorește utilizatorul să obțină, folosindu-se de intenții și entități. Intențiile reprezintă tipul de informație cerută de utilizator. Spre exemplu, pentru mesajul „Unde se află sala EC105?”, intenția este de localizare a unei săli. Entitățile sunt cuvinte cheie extrase din mesajul utilizatorului, pentru a putea prelucra răspunsul corect. În exemplul anterior, entitatea era „EC105”, care reprezintă o sală de curs. Acestea pot reprezenta mult mai multe alte lucruri: persoane, lucruri, evenimente, totul depinde de modul în care au fost configurate de către programator. Intențiile

sunt scrise pe baza unor clasificări, în funcție de tipul de informație cerută, iar la fiecare trebuie date cât mai multe exemple, pentru a crește gradul de corectitudine al răspunsului.

Când se trimite un mesaj, acesta va trece printr-un pipeline, Fig. 3, pentru prelucrarea acestuia. Un pipeline predefinit și recomandat este spacy_sklearn, care este alcătuit din mai multe etape. Prima etapă este cea de tokenizare, în care mesajul este împărțit în tokeni. În a doua etapă se verifică dacă există entități in mesaj și se extrag. După, se va face o reprezentare vectorială a mesajului și se va antrena in estimator pentru setul de date, de către scikit-learn.

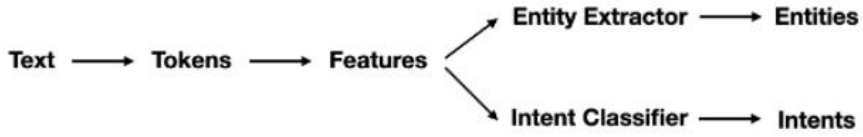


Fig. 3 RASA NLU pipeline

2.3.2 Rasa dialogue management

După ce s-a interpretat mesajul utilizatorului, următorul pas este de creare a unui răspuns, acest lucru fiind realizat de Rasa dialogue management. Acesta răspunde la mesaj folosindu-se de acțiuni, povești, template-uri si slot-uri. Acțiunile reprezintă răspunsurile la intenții, fiecare intenție trebuie sa aibă asociată o acțiune. Acestea pot fi sub forma de text, link sau o funcție care trebuie executată. Poveștile reprezintă o serie de intenții și acțiuni, folosite în interacțiunea cu utilizatorul, prin legarea acestora unele de celelate, atunci când există dependențe între ele. În template-uri sunt definite toate acțiunile care pot fi folosite. Datele necesare pentru purtarea conversației sunt stocate în slot-uri, acestea fiind efectiv memoria chatbotului.

Starea dialogului este salvată într-un obiect de tip tracker, existând câte unul pentru fiecare sesiune. Acolo se stochează sloturile si un log al evenimentelor petrecute.

Politicile au responsabilitatea de a selecta următoarea acțiune, folosindu-se de datele salvate in tracker. Acestea au un obiect, numit featurizer, folosit pentru a reține starea dată de către tracker, cum ar fi ultima acțiune folosită, ultima intenție, entitățile din ultimul mesaj și informațiile din sloturi. În practică se pot stoca mai multe stări anterioare, acest lucru depinde de modul de implementare.

Odată ce este interpretat mesajul de intrare de catre Rasa NLU, se trimite trackerului informația ca un nou mesaj a fost primit. Politicile primesc starea curenta a trackerului și aleg acțiunea de răspuns. Acțiunea este înregistrată de către tracker și este executată. Dacă nu se primește acțiunea de așteptare a noi mesaje de la utilizator, se trimite acțiunea către politici. Aceste etape, pe care le parcurge un mesaj, sunt descrise în Fig. 4.

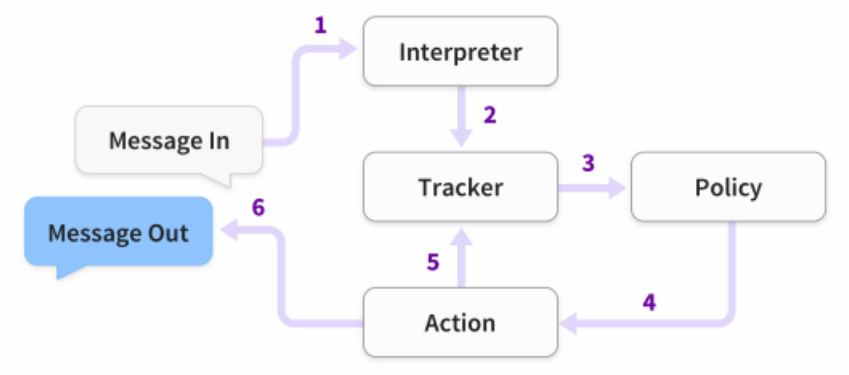


Fig. 4 Flux de execuție

2.4. Procesarea limbajului natural

2.4.1. Arhitectura de tip Transformer

Un Transformer este o arhitectură folosită pentru a rezolva sarcini secvență în secvență (sequence-to-sequence) ținând cont de dependențele pe distanțe lungi. Aceasta tehnica nu folosește rețele neuronale sau convoluție, ci se bazează pe atenție de sine(self-attention).

Atenția înseamna în termeni de NLP concentrarea pe anumite părți din input atunci când prezicem output-ul. Self attention reprezintă un mecanism care leagă diferite poziții dintr-o secvență, pentru a putea fi reprezentată. Atenția poate fi de mai multe tipuri: atenție encoder-decoder(între secvența de input și cea de output), self attention din input (atenție asupra input-ului) și self attention din output (atenție asuprea output-ului).

Arhitectură

Codificatorul și decodificatorul sunt de fapt mai multe, ele fiind stivuite unul peste celălalt, având același număr de unități. Astfel, inputul este trimis de la un codificator la altul, de a lungul stivei, iar ultimul va trimite outputul către toți decoderii din stivă. Self-attention este calculată de mai multe ori folosind această arhitectură, în paralele și independent, purtând numele de Multi-Head Attention, prezentată în Fig. 5. Output-urile sunt concatenate si transformate liniar .

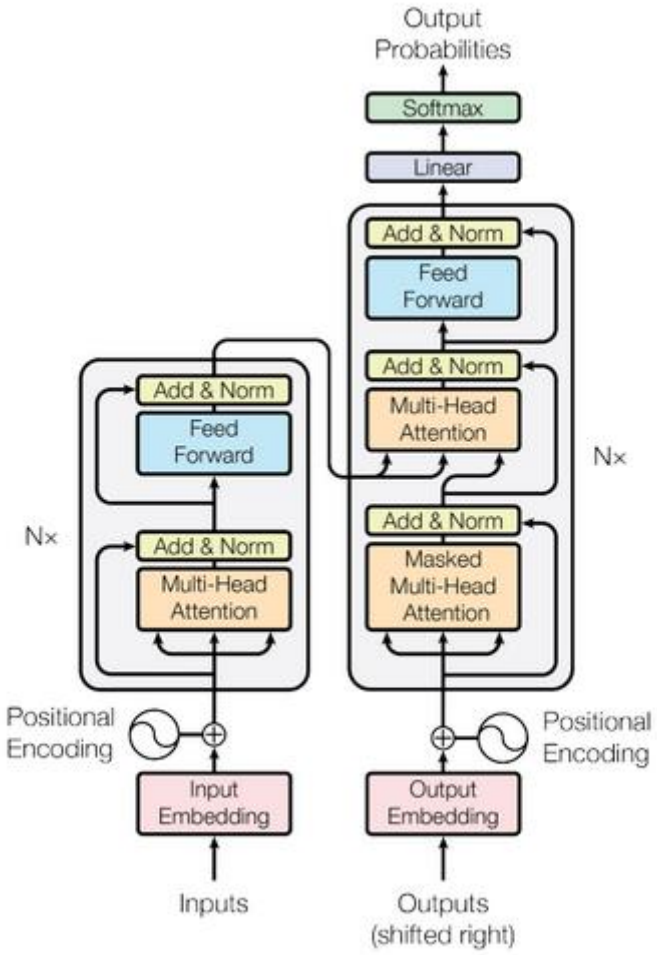


Fig. 5 Arhitectura unui Transformer

Limitări

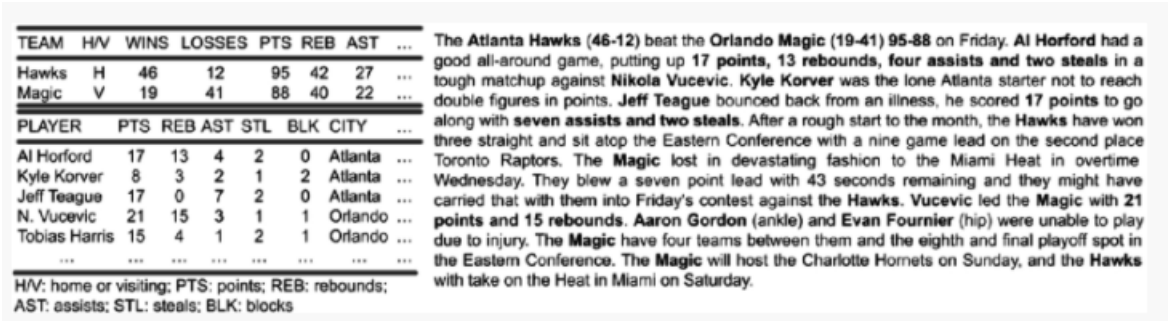
Transformerii prezintă totuși limitări la nivel de text. Aceștia pot prelucra text de dimensiune fixă, deoarece acesta trebuie împărțit în bucăți înainte de a fi prelucrat. Împărțire textului poate duce la pierderi în context, deoarece textul este împărțit fără a respecta regulile semantice.

Transformer-XL

Pentru a trece peste limitări, s-a introdus un mecanism de recurență în arhitectura transformerului - Segment-Level Recurrence with State Reuse. În timpul antrenării se salvează secvența prelucrată anterior pentru a fi utilizată drept un context extins atunci când se prelucrează următoarea, această intrare suplimentară permițând rețelei să aibă acces la informații anterioare și să reușească să prelucreze dependența pe termen lung. Acest lucru face ca pericolul fragmentării contextului să fie evitat.

2.4.2 Generarea de text folosind date (Data-to-Text Generation)

Există o mulțime de data, despre orice, stocate în baze de date sub forma de tabele, perechi cheie-valoare. Acestea pot fi de dimensiuni mari, deseori greu de citit/interpretat. Ele sunt folosite în majoritatea domeniilor, precum prognoza meteo, mass-media, financiar sau medical. Pentru a fi mai ușor de interpretat, se încearcă generarea de text coerent pe baza acestor date - Fig. 6.



TEAM	H/V	WINS	LOSSES	PTS	REB	AST	...
Hawks	H	46	12	95	42	27	...
Magic	V	19	41	88	40	22	...

PLAYER	PTS	REB	AST	STL	BLK	CITY	...
Al Horford	17	13	4	2	0	Atlanta	...
Kyle Korver	8	3	2	1	2	Atlanta	...
Jeff Teague	17	0	7	2	0	Atlanta	...
N. Vucevic	21	15	3	1	1	Orlando	...
Tobias Harris	15	4	1	2	1	Orlando	...
...	...	...	...	...	...	...	...

H/V: home or visiting; PTS: points; REB: rebounds; AST: assists; STL: steals; BLK: blocks

The **Atlanta Hawks** (**46-12**) beat the **Orlando Magic** (**19-41**) **95-88** on Friday. **Al Horford** had a good all-around game, putting up **17 points, 13 rebounds, four assists and two steals** in a tough matchup against **Nikola Vucevic**. **Kyle Korver** was the lone Atlanta starter not to reach double figures in points. **Jeff Teague** bounced back from an illness, he scored **17 points** to go along with **seven assists and two steals**. After a rough start to the month, the **Hawks** have won three straight and sit atop the Eastern Conference with a nine game lead on the second place Toronto Raptors. The **Magic** lost in devastating fashion to the Miami Heat in overtime Wednesday. They blew a seven point lead with 43 seconds remaining and they might have carried that with them into Friday's contest against the **Hawks**. **Vucevic** led the **Magic** with **21 points and 15 rebounds**. **Aaron Gordon** (ankle) and **Evan Fournier** (hip) were unable to play due to injury. The **Magic** have four teams between them and the eighth and final playoff spot in the Eastern Conference. The **Magic** will host the Charlotte Hornets on Sunday, and the **Hawks** with take on the Heat in Miami on Saturday.

Fig. 6 Exemplu de generare de text folosind date

Pentru ca un model să poată genera un text, prima dată acesta trebuie sa poată înțelege datele și să fie capabil să genereze descrieri pentru fiecare în parte. Acest lucru se poate face folosind o arhitectură encoder-decoder, unde datele care sunt codificate intr-o reprezentare vectorială iar un decodificator generează cuvinte pe baza acestor date. Contextul descrierii se realizează pe baza elementelor detectate în etapa de codificare, ca fiind importante(attention mechanism), iar cuvintele necunoscute sau rare sunt tratate folosind cuvinte din același context sau mediu(copy mechanism). Aceste mecanisme se confruntă cu două probleme, care le diminuează precizia: linearizarea structurii de date(ex: prin linearizarea datelor dintr-un tabel se pierd entitățile, deoarece se pierd liniile tabelelor) și ordonarea arbitrară a colecțiilor neordonate din rețelele recurente, care poate avea un impact semnificativ asupra performanței învățării.

Pentru a evita aceste probleme s-a schimbat modul de codificare a datelor. Prima dată se codifică entitățile pe baza elementelor lor iar apoi se codifică datele pe baza entităților. Pentru a asigura o codificare cât mai bună, în concordanță cu celelate elemente, fără a conta

poziționarea lor inițială s-a introdus un codificator de tip Transformer. De asemenea, pentru a calcula contextul ierarhic, s-a introdus un mecanism de atenție ierarhic.

Pentru implementarea mecanismului de atenție ierarhică sunt doua metode. Prima metodă, Traditional Hierarchical Attention, se bazează pe alegerea entităților și apoi a înregistrărilor acestor entități. În implementarea acesteia se folosește un set de scoruri de atenție (attention scores) care este atribuit entităților, și un set de scoruri de atenție pentru inregistrările unei entități, calculate la fiecare pas din etapa de decodificare. Entitățile sunt reprezentate apoi ca suma scorurilor înregistrărilor iar structura de date ca suma reprezentării entităților. A doua metodă este Key-guided Hierachical Attention, care se bazează pe faptul că atunci cand o entitate este aleasă, ajutându-se de scorul de atenție, numai înregistrările acesteia sunt relevante pentru a construi descrierea, necontând valoarea entității.

Prin utilizarea acestor tehnici rezultă un codificator ierarhic care eficientizează modul în care sunt prelucrate datele de intrare, care ține cont de o organizare ierarhică a datelor și care poate ajuta la o mai buna translatare a datelor în text.

2.4.3 Clasificarea actelor de dialog folosind contextul și auto-atenția (self-attention)

Actul de dialog este unitatea de bază a interacțiunii umane, a comunicării lingvistice. Clasificarea actului de dialog este un subiect îndelung cercetat în domeniul procesării limbajului natural, fiind folosit în implementarea de agenți conversaționali, recunoaștere vocală etc.

Studiile recente au reușit rezultate bune folosind modele deep learning pentru clasificarea actului de dialog. Majoritatea acestor metode erau implementate folosind interpretarea fiecărei replici în parte, izolată de restul dialogului. Dacă se cunoaște textul sau etichetelor anterioare se poate prezice starea dialogului. Pentru a putea îmbunătăți aceste modele s-au folosit concepte noi din procesarea limbajului natural, cum ar fi self-attentive, modele de învățare folosind deep learning, și dependințele legate de contextul dialogului.

Un act de dialog este format din mai multe enunțuri, care l-a rândul lor sunt formate din mai multe cuvinte iar fiecare cuvânt are asociată o etichetă. Fiecare conversație are asociată o secvență de etichete, care reprezintă actul de dialog asociat acelei conversații - Fig. 7. Cuvântul este încorporat în sub-cuvinte pentru a putea determina partea lexicală si entitățile din cuvinte mai bine. Această încorporare face parte din etapa de procesare a enunțurilor folosind rețele neuronale recurente (Recurrent Neural Networks - RNNs), care formează inputul pentru etapa de identificarea a contextului și a self-attentive (Context-aware Self-attention).

Speaker	Utterance	DA label
A	Okay.	Other
A	Um, what did you do this weekend?	Question
B	Well, uh, pretty much spent most of my time in the yard.	Statement
B	[Throat Clearing]	Non Verbal
A	Uh-Huh.	Backchannel
A	What do you have planned for your yard?	Question
B	Well, we're in the process of, revitalizing it.	Statement

Fig. 7 Exemplu act de dialog cu etichetele corespunzătoare acțiunilor

Starea rezultată la etapa precedentă, care oferă contextul dialogului se combină cu stările cuvintelor din enunț într-un codificator self-attentive. Reprezentarea enunțurilor este prelucrată într-o etapă de procesare la nivel de conversație, folosind rețele neuronale recurente, care apoi este trimisă catre nivelul CRF(Constant Rate Factor), unde se evaluează relațiile dintre etichete în context și decodează împreună secvența optima de etichete ale enunțului. În urma acestor etape se realizează clasificarea dialogului ținand cont de context și de replicile anterioare, acest lucru ajutând la cresterea preciziei predicției de replici dintr-o conversație.

2.5 spaCy

spaCy este o bibliotecă open-source pentru procesarea limbajului natural, dezvoltată in Python. Acesta este folosită pentru prelucrarea textelor de mari dimensiuni: indentifică subiectul din text și cuvintele cheie (companii, persoane, produse etc.). Principalele caracterisitici ale acestei bibilioteci sunt: împărțirea textului in cuvinte (tokenizare), indentificarea părților de vorbire, identificarea relațiilor dintre tokeni, împărțirea frazelor în propoziții, găsirea entităților dintr-un text și stabilirea cuvintelor/frazelor care au un grad de similaritate ridicat.

2.5.1 Etapele de procesare ale unui text

În urma procesării unui text rezultă un obiect de tip Doc. Pentru crearea acestui obiect, textul este împărțit în tokeni (tokenizer), se identifică părțile de vorbire (tagger), dependențele dintre tokeni(parser) și entitățile (ner) - Fig. 8. După aceste etape se identifică forma de bază a cuvintelor(lemmatizer), se adaugă etichete documentului și atribute/proprietăți/metode personalizate.

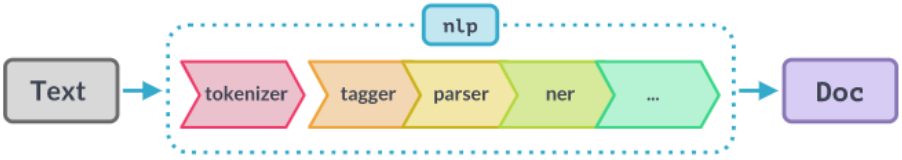


Fig. 8 spaCy - Etapele de procesare ale unui text

2.5.2 Antrenare

Antrenarea unui model se face folosind date pentru antrenare. Componentele folosite la procesarea unui text sunt modele statice, care decid în funcție de exemplele date în timpul antrenării - Fig. 9. Aceste exemple pot fi sub forma unui text, în care sunt cuvinte pe care modelul ar trebui sa le gasească, cum ar fi o entitate, o parte de vorbire etc. Exemplele trebuie să fie formulate astfel încât modelul să poată generaliza și pe alte date; aceastea trebuie alcătuite astfel încât modelul să memoreze contextul în care a apărut acea entitate/parte de vorbire sau orice alte cuvinte cheie pe care vream sa le identifice.

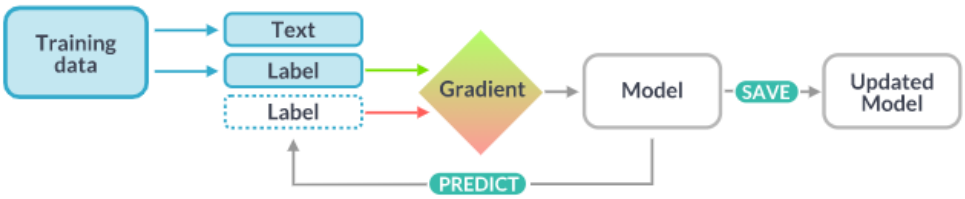


Fig. 9 spaCy – Antrenarea unui model

Pentru a vedea dacă antrenarea a decurs corespunzător, este nevoie de un set de date pentru testare, pentru a vedea cât de bine a învățat modelul să generalizeze pe baza exemplelor antrenate și pentru a verifica faptul că nu face overfitting pe datele de antrenare.

CAPITOLUL 3. Metoda propusă

Agentul conversațional este implementat folosind Rasa NLU, Rasa Core si Spacy. Pentru stabilirea caracteristicilor agentului, s-a ținut cont de cerințele pe care le doresc viitorii utilizatori, studenții de la Facultatea de Automatică și Calculatoare din București. Integrarea în aplicația ACS UPB Mobile este făcută folosind framework-ul Flutter, care folosește la rândul său limbajul de programare Dart.

3.1 Corpus

3.1.1. Intenții

O intenție reflectă scopul enunțului scris de utilizator, ce tip de informație vrea să obțină acesta de la agent. Astfel, pentru implementare, au fost definite doua categori de intenții, cele de tip custom, folosite pentru diferite informații dinamice,unde este necesară interogaarea unei baze de date, și intenții faqs (frequently asked questions) pentru întrebări frecvente.

Intențiile custom, în număr de 9, sunt grupate în mai multe categorii, în funcție de informația cerută:

- localizare a sălilor de curs (o intenție), ex: Unde se află sala EC105?
- date despre cursuri (5 intenții): dată, oră, notare ex: Spune-mi, te rog, în ce sală se va ține cursul de POO?
- date despre studenți: șefi de grupă/serie - Fig. 10 (2 intenții); ex: Cum îl cheamă pe șeful de serie de la seria CD?
- date despre profesori (o intenție), ex: Ce profesor are curs în sala PR001?

Intențiile faqs, în număr de 30, se referă la informațiile de care ar putea avea nevoie un student pe partea administrativă:

- cazare la cămin, ex: Îmi pot alege colegul de cameră?
- depunere acte secretariat, ex: Poate să facă cineva cerere în locul meu la secretariat?
- program secretariat, ex: Știi intervalul orar când mă pot duce la secretariat?
- schimbare grupă/serie, ex: Există vreo modalitate să îmi schimb seria?
- Schimbare specializare - Fig. 11, ex: Pot să îmi schimb specializare?
- facultate online, ex: Ce programe trebuie să îmi instalez pentru a-mi desfășura activitatea online?
- burse, ex: Cum pot să aflu care sunt criteriile pentru obținerea burselor?
- exemple de cereri, ex: Ai un model de cerere pentru secretariat?
- Eduroam, ex: Cum pot să ma conectez la rețelele Wi-Fi Eduroam?
- contractul de studii, ex: Îmi poți explica ce este contractul de studii?

- ticket pentru suport[9], ex: Mă poți ajuta să deschid un tichet?
- taxa școlarizare/cămin, ex: Nu știu cât este taxa de școlarizare.
- adeverințe, ex: Cum pot obține o adeverință de student?

  Acestea sunt scrise in fisiere de tip yml.

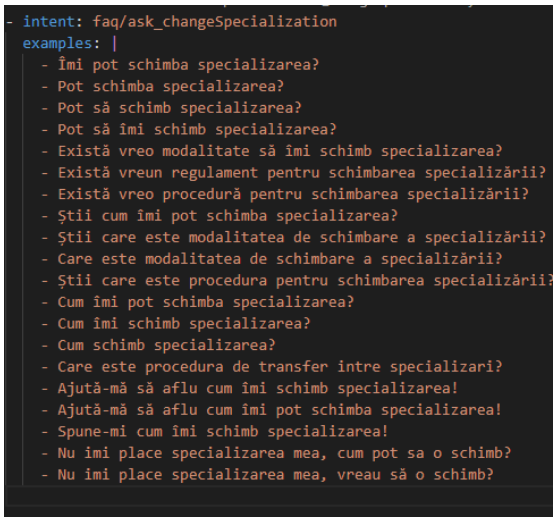


Fig. 11 Formulări intenție – schimbare specializare

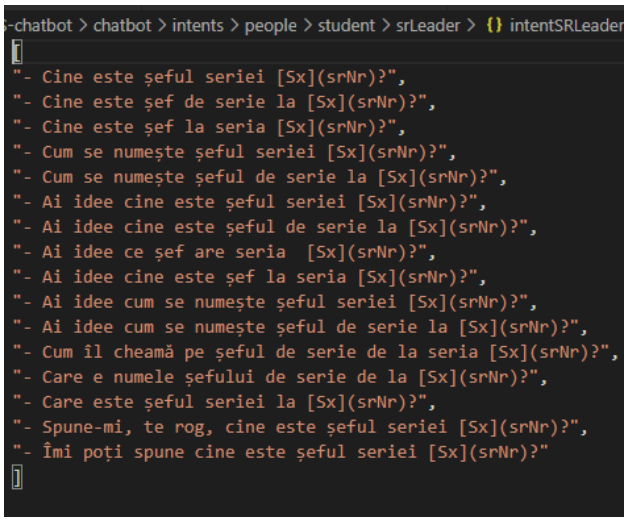


Fig. 10 Formulări intenție – șeful seriei

3.1.2. Entități

Entitățile folosite în antrenarea agentului conversațional și care ar trebui identificate de acesta în urma etapei de tokenizare, sunt:

- eveniment (nume de curs/laborator/seminar/conferință), ex: Învățare Automată
- tip eveniment (evenimentul este curs/laborator/seminar)
- sală, ex: PR001
- oameni (profesori, asistenți), ex: Mihai Dascălu
- birou (biroul profesorilor), ex: ED405
- grupa (numărul grupei din care face parte un student), ex: 313
- seria (seria din care face parte un student), ex: CD

[9] https://ticketing.upb.ro/

3.2. *Stabilirea caracteristicilor agentului conversațional*

Pentru a afla nevoile studenților, aceștia au completat un formular unde li s-au prezentat exemple de informații pe care le-ar afla folosind un chatbot in aplicatia ACS UPB mobile. Pe baza acestui formular s-a început schema de proiectare a corpusului agentului conversațional.

Din vizualizarea răspunsurilor, prezentată în Fig. 12, s-a remarcat un interes major pentru a afla punctajele de la materii, condițiile minime de promovare, modele de cereri sau locația sălilor din campusul studențesc. Cei mai mulți au optat pentru comunicarea prin text (70,8%). S-a observat că 33,6% dintre persoanele care au completat au fost studenți de anul 1, drept urmare, pentru ai ajuta cu eventualele nelămuriri au fost adăugate în corpusul agentului întrebări cu care studenții de anul 1 s-ar putea confrunta, cum ar fi cele referitoare la colegii de cameră din cămin, schimbarea grupei sau locația facultății.

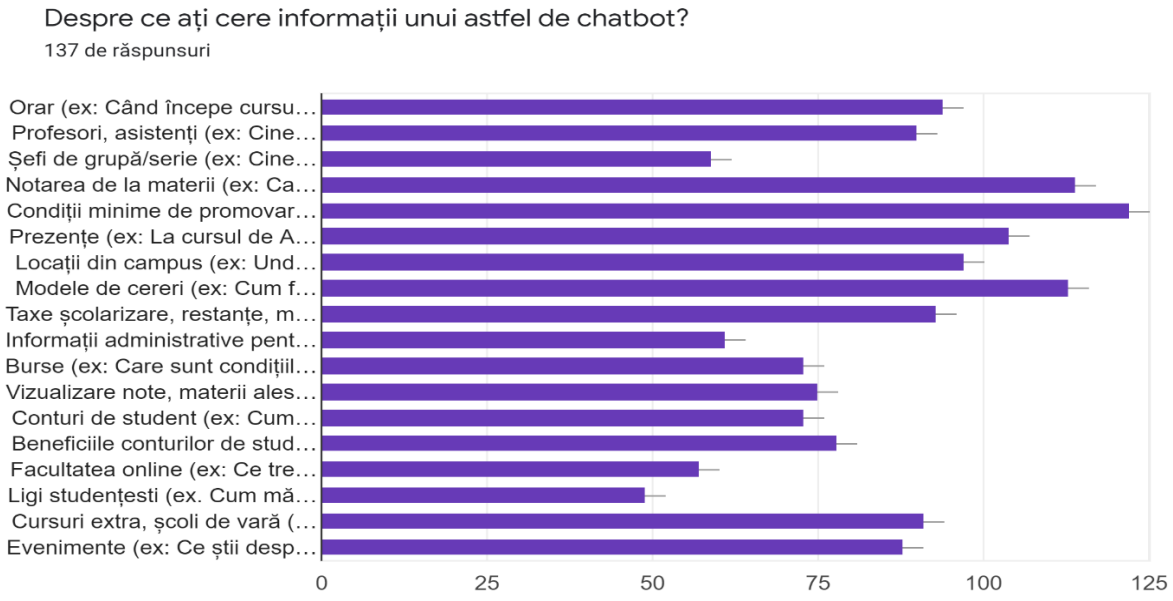


Fig. 12 Răspunsurile studenților la formular

3.3 Agent Conversațional

3.3.1 Reguli

Pentru început, există doar o regulă pentru întrebările frecvente, care are ca pași identificarea intenției de faq, din cele prezente în fișierul nlu.yml (întelegerea limbajului natural), iar apoi identificarea răspunsului aflat in domain.yml.

Răspunsuri

Răspunsurile sunt mesajele întoarse de agent către utilizator, acestea sunt, de asemea, definite în fișiere de tip .yml, denumirea lor ar trebui să înceapă cu „utter_”, spre exemplu „utter_faq_ask_becomeGroupLeader”. În implementarea lor s-au folosit mai multe formulari, pentru nu face comunicarea cu utilizatorul, drept una monotonă.

Răspunsurile întoarse ca acțiune la intențiile pentru întrebările frecvente au o formulare statică, spre exemplu pentru intenția de aflare a modului de schimbare a seriei, unul dintre răspunsuri este :„ *Pentru a schimba seria trebuie facută o cerere la început de an, la decanat. Aceasta poate fi respinsă, depinde în mare parte de locurile disponibile. Șansele cele mai mari de acceptare le au studenții care își gasesc persoane cu care să facă schimb de serie.”.*

Acestea sunt implementate folosind *Response Selector,* care supervizează întrebările și selectează din setul de antrenarea și pe baza similarității dintre inputul dat de utilizator și a întrebărilor din setul de antrenare, oferă un răspuns oferit la întrebarea din setul de antrenare care a obținut rangul cel mai mare de asemănare. El este introdus în pipeline-ul NLU, din fișierul de configurare, iar pentru ca modelul sa poată folosi aceste răspunsuri, întoarse de Response Selector, trebuie sa configureze o regulă, care să aibă ca intenție, întrebări frecvente și ca acțiune, răspuns la întrebări frecvente, urmând ca intențiile și răspunsurile să fie configurate separat, pentru fiecar în parte, în fișierele nlu.yml (exemplele de întrebări) si domain.yml (răspunsurile).

3.3.2 Povești

Poveștile sunt o reprezentare a dialogului dintre agentul conversațional și utilizator, în care repricile utilizatorului sunt reprezentate de intenții și entități, iar cele ale agentului, de acțiuni unde se răspunde la acestea; definirea poveștilor folosite la antrenarea modelului este prezentată în Fig. 13. Spre exemplu, povestea de localizare a unei săli de curs, are drept input

intenția locate_classroom, în care se identifică entitatea de tip classroom iar ca acțiune este action_locate_classroom, care va întoarce un răspuns în funcție de sala extrasă din intenție

```
stories:
- story: locate the classroom in the faculty
  steps:
  - intent: locate_classroom
  - action: action_locate_classroom
- story: get the date of the event
  steps:
  - intent: get_date_event
  - action: action_get_date_event
- story: get the end time of the event
  steps:
  - intent: get_end_time_event
  - action: action_get_end_time_event
- story: locate the event
  steps:
  - intent: locate_event
  - action: action_locate_event
```

Fig. 13 Configurarea poveștilor din aplicație

Acțiuni

În urma fiecărui mesaj al utilizatorului, este necesară o acțiune. Acestea sunt întoarse în funcție de informațiile extrase din dialog și în urma indentificării intențiilor. Acțiunile custom folosite în implementarea modelului rulează un cod în care se interoghează baza de date, in funcție de id-ul utilizatorului, pentru a vedea datele la care are acesta acces. Acestea sunt folosite pentru a oferi răspunsuri la intențiile care au nevoie de informații suplimentare pentru prelucrarea răspunsurilor.

Răspunsurile întoarse in urma unei acțiuni custom, se formează în funcție de entitățile extrase din enunțul intenției, introduse în răspuns sub forma unor variabile, și în urma interogării bazei de date, spre exemplu, pentru intenția de localizare a unui eveniment, un răspuns întors este: „***Cursul*** *de* ***Proiectare Orientată pe Obiect*** *are loc vineri, la ora 14”.* Baza de date interogată este cea a aplicației ACS UPB Mobile, de unde se iau date despre utilizatorul care dialoghează cu agentul conversațional sau alte informații despre relevante, de care este nevoie în convorbire, cum ar fi date despre materii, profesori sau orar.

Acțiunile custom au fost implementate pentru a avea acces la baza de date sau la diferite fișiere din care să se poată lua informația necesară. Toate acțiunile implementate conțin la începutul metodei extragerea entităților identificate din mesajul utilizatorului. Cele care privesc evenimentele de la facultate mai conțin construirea denumirii evenimentului, pe baza datelor utilizatorului

Localizarea unei săli – ActionLocateClassroom

În urma extragerii entității din enunț trebuia să rezulte o denumire de sală de curs. În funcție de sala identificată, se caută locația într-un fișier populat cu amplasarea acestora. Fișierul a fost realizat folosind planurile clădirilor de la Facultatea de Automatică și Calculatoare. Pentru ca partea caligrafică a denumirii să nu încurce căutarea sălii, denumirilor identificate li se scot spațiile goale iar toate literele sunt salvate cu majuscule. Dacă locația nu este găsită, se returnează un mesaj în care se specifică acest lucru.

Identificare a timpului de desfășurare al unui eveniment

Aceste acțiuni sunt definite în implementare sub forma ActionGetDateEvent, ActionGetStartTimeEvent și ActionGetEndTimeEvent Entitățile extrase trebuie să fie de tip event sau typeEvent. Se verifică dacă a fost identificată o enitate typeEvent cu un grad de încredere mare, altfel se consideră că utilizatorul se referă la un curs. Se interoghează baza de date pentru a afla datele pe care le are aplicația despre eveniment. În momentul în care se găsește evenimentul în baza de date, are loc prelucrarea datelor acestuia:

- verifică dacă tipul evenimentului găsit în baza de date coincide cu entitatea typeEvent, dacă nu se caută în continuare un eveniment
- se identifică dacă evenimentul are relevanță pentru utilizator. Ex: Cursul de Analiza Algoritmilor de la seria CC, are relevanță doar pentru studenții care sunt în acea serie
- extrage ziua săptămânii din regula de repetare a evenimentului (un șir de caractere care conține datele privnd repetarea evenimentelor, majoritatea din ele repetându-se săptămânal; sunt definite perioadele de început și de sfârșit) și o prelucrează astfel încât să poată fi introdusă în mesaj ex: MO -> Luni.
- extrage ora de începere din timestamp
- compune mesajul de returnare, folosindu-se de numele evenimentului identificat, tipul acestuia și data la care are loc

În implementarea acțiunii ActionGetEndTimeEvent este calculată și ora de sfârșit a evenimentului, folosind durata evenimentului, regăsită în informațiile despre eveniment din baza de date. Pentru sunt sanșe ca un eveniment să se desfășoare de mai multe ori pe săptămână, se vor returna toate evenimentele care au trecut de verificările denumirii, tipului și ale relevanței. Ex:„ Cursul de Matematică 1 se termină marți, la ora 12 și vineri la ora 16”.

Localizarea unui eveniment

Odată aflat numele și tipul evenimentului din entități se verifică gradul de încredere al acestora. Daca tipul evenimentului nu este identificat, la fel ca la acțiunile care gestionează timpul de desfășurare, tipul evenimentului este setat implicit ca fiind curs. După ce se identifică evenimentul căutat în baza de date, verificând numele și relevanța acestuia, se poate identifica locația unde se desfășoară, care este și aceasta stocată acolo, drept câmp. Deoarece, la fel ca la celelate acțiuni, sunt evenimente care au loc de mai multe ori în aceeași săptămână, se identifică și ziua în care se desfășoară, extragând-o la fel ca la celelalte acțiuni, din regula de repetare, și se returnează împreună cu locația, pentru a ști la care eveniment se referă. Ex: Cursul de Paradigme de Programare are loc marți, în PR001 și joi, în EC105.

Obținere notare de la curs

Notarea unui curs reprezintă punctele repartizate pe fiecare activitate în parte (curs, laborator, seminar, teme) din cadrul acestuia. După ce a fost identificat cursul căutat, se caută în baza de date și se verifică dacă sunt prezente detaliile legate de notare. Dacă acestea există, sunt adăugate mesajului, specificând pentru fiecare la ce activitate se referă.

Aflarea a șefului de grupă/serie

Un student are nevoie de informații despre șefii de grupă/serie. În urma identificării grupei/seriei despre care se dorește aflarea șefului, se interoghează un fișier json, unde sunt stocate aceste date, și se adaugă informația găsită la mesajul de răspuns

Aflarea profesorului care predă într-o sală

Atunci când este identificat numele sălii, prin extragerea entităților din enunț, se va verifica dacă avem date despre ce profesor predă în acea sală. În caz că există stocate informații relevante despre profesorii care predau în sala identificatăm acestea vor fi adăugate la mesajul de răspuns.

3.3.3 *Pipeline*

Componentele pipeline-ului prelucrează secvențial inputul primit, pentru a putea fi interpretat. Configurarea acestuia este prezentată în Fig. 14

E: > ACS-chatbot > chatbot > src > ! config.yml

```yaml
language: "ro"

pipeline:
  - name: SpacyNLP
    model: "ro_core_news_md"
  - name: SpacyTokenizer
  - name: SpacyFeaturizer
  - name: RegexFeaturizer
  - name: LexicalSyntacticFeaturizer
  - name: CountVectorsFeaturizer
  - name: CountVectorsFeaturizer
    analyzer: "char_wb"
    min_ngram: 1
    max_ngram: 4
  - name: DIETClassifier
    epochs: 100
  - name: EntitySynonymMapper
  - name: ResponseSelector
    epochs: 100
    retrieval_intent: faq


policies:
  - name: MemoizationPolicy
  - name: RulePolicy
  - name: TEDPolicy
    max_history: 5
    epochs: 100
```

Fig. 14 Pipeline-ul și politicile folosite în implementare

Spacy NLP

Această componentă inițializează structurile folosite de spaCy, fiecare componentă depinde de ea, de aceea, trebuie se află la începutl pipeline-ului. Pentru a putea funcționa, trebuie să primească un model de limbă; agentul conversațional fiind în limba română, pipeline-ul conține modelul ro_core_news_md, bazat pe română.

SpacyTokenizer

SpacyTokenizer este folosit pentru împarțirea textului (mesajel primite de la utilizator,răspunsurile și intențiile) în tokeni, cu ajutorul cărora se vor identific mai apoi intențiile și entitățile.

Featurizers

Aceste componente sunt folosite pentru a găsi caracteristicile textului (entitățile, intențiile), reprezentând textul sub diverse forme:

- *SpacyFeaturizer* creează o reprezentare vectorială a textului

- *RegexFeaturizer* reprezintă mesajele utilizatorului sub formă de expresii regulate
- *CountVectorsFeaturizer*, care identifică si reprezintă cuvintele în funcție de numărul de apariții
- *LexicalSyntacticFeaturizer*, folosește o fereastră glisantă care parcurge fiecare token extras și ii reține anumite caracteristici, folosite pentru identificarea entităților:
    - Se află la începutul sau la sfârșitul unei propoziții
    - Conține litere mari sau mici
    - Începe cu literă mare, iar celelate litere sunt mci
    - Identifică partea de vorbire
    - Verifică dacă este format doar din cifre
    - Reține primele/ultimele caractere

DIETClassifier

DIET (Dual Intent Entity Transformer) este o arhitectură de indentificare a intențiilor și extragere a entităților bazată pe transformer. Acesta, atunci cand returnează entitățile și intențiile, returnează si gradul de încredere pe care o are predicția făcută, pentru a putea ști cât de mari sunt șansele ca acestea să fie identificate corect.

EntitySynonimMapper

EntitySynonimMapper, după cum îi spune și numele, este folosit pentru a mapa entitățile care au valori identice, dar scrise sub o altă formă. Spre exemplu, entitatea Analiza Algoritmilor, care este un nume de materie, trebuie să fie echivalentă cu AA, care este abrevierea acesteia.

Response Selector

Response Selector prezice răspunsul modelului dintr-un set de răspunsuri date pentru antrenare. Numărul de parcurgeri ale setului de antrenare este dat de parametrul epochs. În cazul agentului implementat, valoarea dată parametrului epochs este 100, iar Response Selector este folosit pentru identificarea răspunsurilor întrebărilor frecvente.

3.3.4 Politici

Politicile sunt utilizate pentru a decide care acțiune va fi efectuată. Aceastea sunt implementate folosind învățare automată sau folosind reguli. Politicile folosite pentru modelul de față sunt prezentate în Fig. 14

Memoization Policy

Această politică verifică dacă momentul conversației se găsește în poveștile definite în datele de antrenare. Dacă găsește o poveste a cărei acțiune are un grad de încredere de 1.0, o va folosi pe aceasta la următorul pas. Atunci când se caută o poveste care se potrivește cu conversația curentă, se verifică ultimele max_history replici (emițător-receptor), care în configurația aleasă pentru agent, este egală cu 5.

TEDPolicy

Politica TED (Transformer Embedding Dialogue) are la bază recunoașterea entităților și prezicerea următoarei acțiuni, folosind transformer. Pentru această politică au fost setați ca parametri max_history : 5 și epochs : 100.

Rule Policy

În implementarea acestei politici sunt folosite reguli care să gestioneze conversația, care să prezică următoarea acțiune pe baza unui șablon definit drept o regulă. Această politică este folosită în realizarea chatbot-ului la gestionarea întrebărilor frecvente, la care, dacă se identifică o intenție de acest tip, se acționează alegereau unui răspuns din setul de date de pentru întrebările frecvente.

3.3.5 Explicarea fișierelor auxiliare

În afară de fișierele actions.yml, în care sunt prezente acțiunile custom implementate și stories.yml, unde se configurează poveștile unui dialog, există mai multe fișiere care sunt folosite în implementarea modelului.

domain.yml

În acest fișier se configurează cea mai mare parte e modelului. Aici sunt definite intențiile, entitățile, acțiunile și modelele de răspuns. În acest caz sunt trecute răspunsurile la întrebările frecvente. Tot aici se află versiunea de RASA, (versiunea 2.0) și detalii legate de sesiune, cum ar fi timpul de expirare al unei sesiuni (60).

config.yml

În fișierul config.yml este descris pipeline-ul folosit pentru prelucrarea mesajelor, politicile prezente și limba folosită

nlu.yml

Acest fișier conține toate exemplele folosite pentru definirea intențiilor, aici se află toate datele de input cu ajutorul cărora este antrenat modelul

3.4 Antrenare

Setul de date de antrenare este alcătuit din exemple de enunțuri, grupate în funcție de intențiile pe care le exprimă. Un set de antrenare este bun, cu cât este mai mare si mai diversificat. Astfel, pentru partea de antrenare a modelului, s-au folosit diverse formulări ale intențiilor, unele care conțineau elemente de limbaj formal, iar unele, informal, pentru a putea înțelege întrebările mai multor tipuri de utilizatori.

În formulările realizate au fost inserate variabile de entități - Fig. 15, pentru a putea fi mai apoi multiplicate cu mai multe exemple de entități de antrenare. Acest exemple ajută la

```
"- Îmi poți spune, te rog, unde se desfășoară [Ex](event)?",
"- Îmi poți spune, te rog, unde se desfășoară [cursul](typeEvent) de [Ex](event)?",
"- Îmi poți spune, te rog, unde se desfășoară [laboratorul](typeEvent) de [Ex](event)?",
"- Îmi poți spune, te rog, unde se desfășoară [seminarul](typeEvent) de [Ex](event)?",
```

Fig. 15 Model de intenție cu variabile pentru entități

identificarea și extragerea entităților din enunțuri. Astfel, am implementat scripturi în Python care parcurgeau exemplele de intenții și înlocuiau variabilele cu exemple - Fig. 16.

Pentru a găsi exemple de entități s-au folosit informațiile din baza de date. Acestea au fost luate folosind utilitarul firestore-export, care extrage datele dintr-un document din Firebase și le stochează local, într-un fișier JSON. De aici au fost extrase exemple de săli de curs, denumirea materiilor, a evenimentelor și tipurile de evenimente prezente, folosind tot scripturi Python, care la rândul lor creau fișiere JSON, de unde informația putea fi procesată de celelalte scripturi.

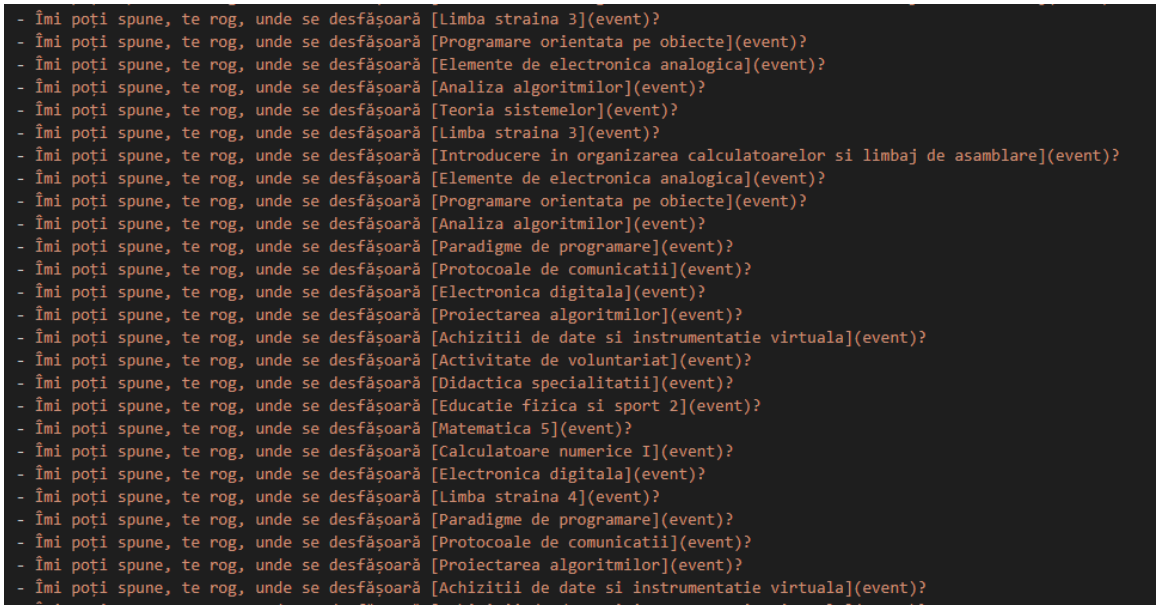


Fig. 16 Exemplu de date generate în urma înlocuirii variabilelor cu exemple de entități

3.5 Interogarea bazei de date Firebase

Aplicația cu care este integrat agentul conversațional, ACS UPB Mobile, folosește o bază de date Firebase. Comunicarea dintre aceasta și serverul pe care rulează modelul, se face folosind cereri HTTP. Pentru a putea avea acces la datele utilizatorului, odată cu mesajul de input din aplicație, este atașată cererii și id-ul utilizatorului. Cu ajutorul id-ului se identifică datele relevante ale unui utilizatori: anul de studiu, seria, grupa, materii, nume, prenume.

Pentru datele despre evenimentele de la facultate, se folosește documentul pentru evenimente, care conține ora de început a acestuia, data, locația, denumirea, tipul si grupa/seria pentru care acesta are o relevanță (ex: laboratorul de POO de la 8:00, are relevanță pentru grupa care este înscrisă pentru acest interval, nu trebuie să apară pentru restul grupelor). Evenimentele sunt căutate în funcție de denumirea unică a fiecăruia, formată prin concatenarea literelor specifice tipului de învățământ (licență sau master), anul universitar, semestrul, abrevierea denumirii evenimentului și seria pentru care are relevanță (ex: L-A1-S1-PC-CA indică materia Programarea Calculatoarelor, care este în anul 1 de licență, semestrul I, abrevierea este PC și este cea dedicată seriei CA). Aceste date sunt

extrase atunci când se apelează acțiunile custom, pentru a putea formula răspunsurile pentru intențiile legate de timpul de desfășurare al evenimentelor și locația acestora.

Datele referitoare la punctajul materiilor sunt extrase din documentul dedicat acestora, unde este sunt scrise, de către studenți, punctajele pe fiecare parte din materie (examen final, teme, seminar/laborator), prezentate de către cadrele didactice la început de semestru.

Fiind un proiect în desfășurare, în momentul de față nu există date în aplicație despre șefii de serie/grupă, condiții minime de trecere la materii, numărul minim de prezențe pe care trebuie să le aibă un student pentru a putea promova o materie sau informații legate despre ce cadre didactice predau într-o anumită sală, sau locația sălilor (fiind în desfășurare un proiect de navigație prin campus pentru rezolvarea acestui task). Astfel, pentru implementarea răspunsurilor sub formă de prototip, am implementat fișiere json în care am stocat date într-un mod asemănător cu structura pe care o vor aveam documentele din baza de date.

3.6. Integrarea cu aplicația ACS UPB Mobile

3.6.1 Pagina de căutare

Utilizatorul aplicației va avea la dispoziție în aplicație o pagină de căutare - Fig. 17. Rezultatele unei căutări returnează sugestii de profesori si materii în funcție de datele de intrare. Dacă rezultatele nu sunt satisfăcătoare, utilizatorul va avea opțiunea să vorbească cu agentul conversațional.

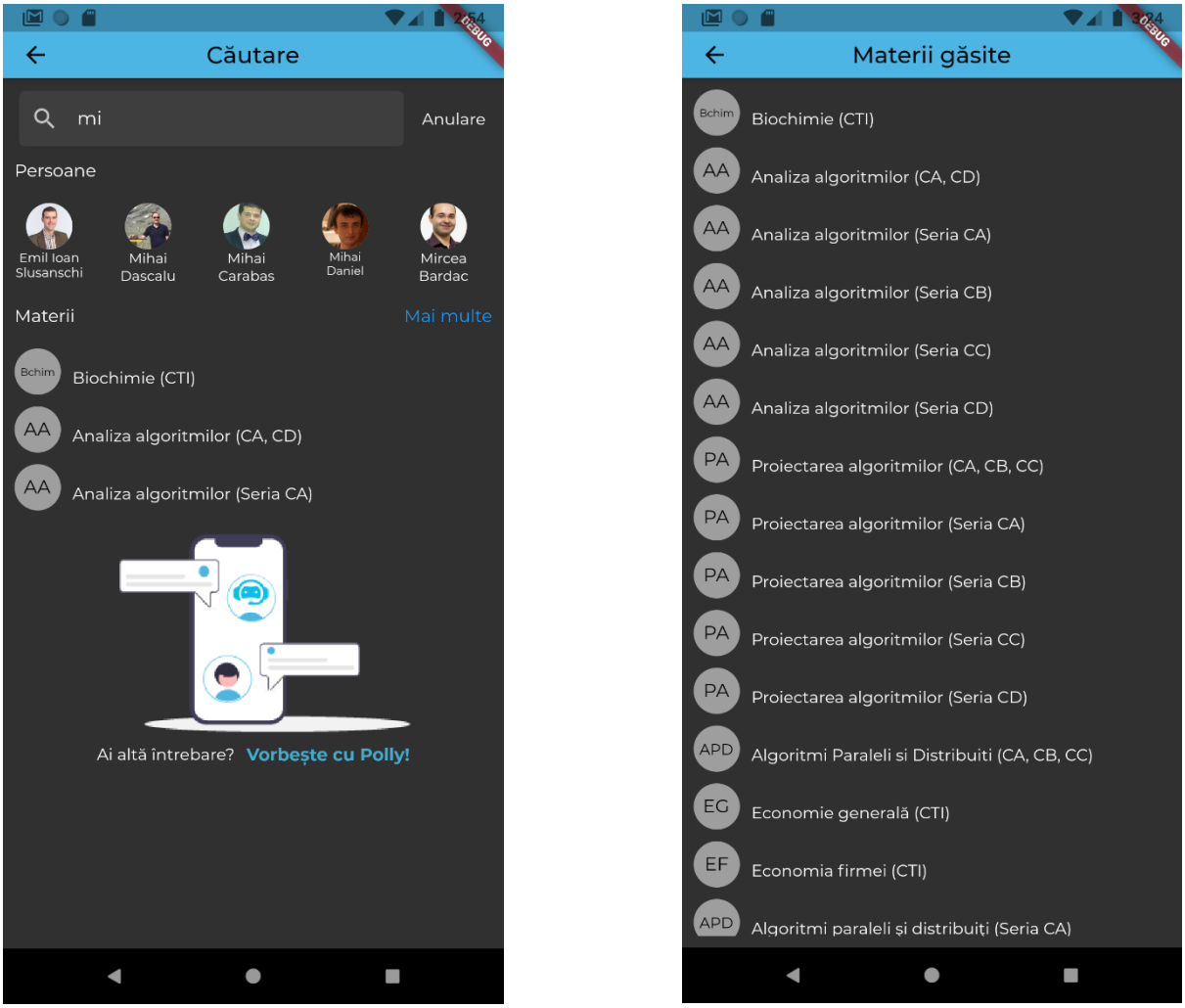


Fig. 17 Captură de ecran din pagina de căutare a aplicației

Aceste rezultate sunt implementate în două etape. În prima etapă de căutare se caută profesorii prezenți în aplicație. Astfel, se interoghează baza de date unde se află profesorii extrași de pe site-ul facultății și se verifică similaritatea numelui acestora cu datele introduse în câmpul de text din widget-ul de căutare, SearchWidget. Persoanele identificate ca fiind relevante pentru căutare sunt adăugate intr-o listă și afișate în aplicație, sub formă de poză și nume. Acest lucru se face folosind funcții asincrone, pentru a nu se bloca aplicația până se face procesarea datelor. Afișarea imaginilor se face cu widget-ul NetworkImage, care afișează imaginea de la URL-ul primit ca parametru, urmând ca forma circulară să fie configurată folosind un CircleAvatar. Elementele listei au atașate câte un GestureDetector, configurat ca

atunci când se apasă pe una din ele, să fie afișate datele din aplicație referitoare cadrului didactic pe care îl conțin.

În a doua etapă se identifică materiile a căror denumire seamănă cu inputul primit la căutare. Materiile prezente în baza de date au fost importate de pe Moodle[10], pentru a putea fi toate cele prezente la facultate și să conțină denumirea și abrevierea oficială. Pentru rezultate cât mai bune, comparația din timpul căutării se face și pe baza abrevierii materiei. Lista rezultată în urma căutării este afișată sub formă de abreviere și nume materie. Inițial se afișează doar primele 3 rezultate, în caz că sunt mai multe, se poate apăsa pe „Mai multe”, și se vor afișa toate într-o pagină nouă. La fel ca la căutarea de persone, dacă se apasă pe un rezultat, se vor afișa informații suplimentare despre acea materie, utilizatorul fiind redirectat pe pagina acesteia. Acolo este afișată notarea, evenimentele viitoare și eventualele link-uri către un drive al materiei sau site-uri pentru laboratoare/seminare.

Sub aceste rezultate, este introdusă posibilitatea de a vorbi cu chatbot-ul, al cărui nume este Polly. Numele a fost ales împreună cu membrii echipei StudentHub și reflectă intrinsec denumirea universității din care face parte Facultatea de Automatică și Calculatoare din București.

3.6.2 Pagina de chat

În urma apăsării pe mesajul „Vorbește cu Polly!”, utilizatorul este redirectat către pagina de chat, unde poate convorbi cu modelul - Fig. 18. Modelul, în momentul de față rulează pe serverul localhost, urmând ca apoi, după ce este migrat pe varianta de producție a aplicației, să ruleze pe un server dedicat.

[10] https://moodle.org/?lang=ro

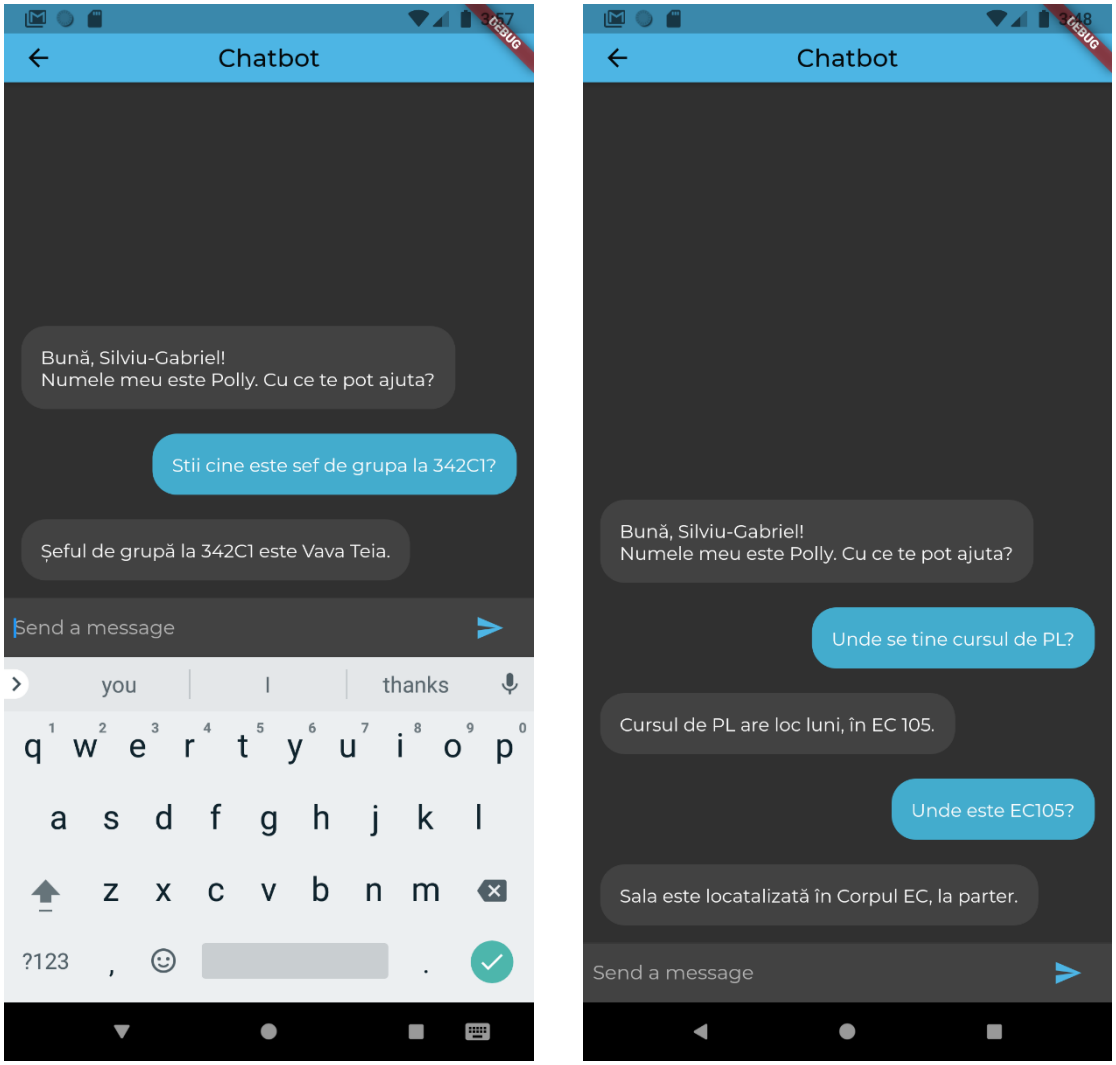


Fig. 18 Interfață grafică – pagina de chat

Pagina este implementată construind un widget cu stare, ChatPage. Acesta este format din doua containere, unul pentru afișarea mesajelor și unul pentru adăugarea de noi input-uri. Când se introduce un nou mesaj de input, se adaugă în lista de mesaje, cu tipul sender, și se apelează o metodă care face o cerere HTTP de tip POST către URL-ul unde rulează serverul modelului. In corpul cererii HTTP se pune id-ul utilizatorului, pentru a putea afla informațiile relevante pentru acesta, și mesajul introdus. Când se primește răspunsul, se adaugă în listă textul răspunsului, cu tipul receiver.

Mesajele din listă se afișează în funcție de tipul fiecăruia:

- Sender: afișate în partea dreaptă a ecranului, cu fundal albastru
- Receiver: afișate în partea stângă a ecranului, cu fundal gri sau alb, în funcție de tema aleasă din aplicație

3.7 Tehnologii integrate

3.7.1 Flutter

Flutter este un framework modern, open-source, cross-platform, cu ajutorul căruia se pot implementa aplicații pe mai multe sisteme de operare, folosind același cod. Dezvoltarea acestuia a fost începută de Google, iar prima versiune, „Sky”, a fost lansată în anul 2015.

Interfața cu utilizatorul

Interfața cu utilizatorul se bazează, în mare parte, pe componente de tip Widget. Un widget este un obiect ce reprezintă o parte a interfeței cu utilizatorul; aceștia sunt organizați sub forma unui arbore - Fig. 19, în care părinții gestionează comportamentul copiilor săi. Widget-urile pot avea stare (Stateful Widget), sau nu (Stateless Widget). Cele care au stare, își pot modifica starea în funcție în timpul rulării aplicație, în funcție de modificarea elementelor pe care le conține. Cele fără stare nu își schimbă configurația, dar consumă și mai puține resurse. Aceste widget-uri sunt asemănătoare cu tag-urile din HTML, iar structura codului este asemănătoare celei din React Native.

```
class ChatBotIntro extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container(
      child: Column(
      children: [
        const Padding(
          padding: EdgeInsets.only(left: 20),
          child: Image(
            image: AssetImage('assets/illustrations/undraw_chat_image.png'),
          ), // Image
        ), // Padding
        Padding(
          padding: const EdgeInsets.fromLTRB(10, 10, 10, 10),
          child: Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Padding(
                  padding: const EdgeInsets.only(right: 10),
                  child: Text(S.current.messageAnotherQuestion)), // Padding
              GestureDetector(
                  child: Text(
                    S.current.messageTalkToChatbot,
                    style: TextStyle(
                        color: Theme.of(context).accentColor,
                        fontWeight: FontWeight.bold,
                        fontSize: 15), // TextStyle
                  ), // Text
                  onTap: () =>
                      Navigator.of(context).push(MaterialPageRoute<ChatPage>(
                    builder: (_) => ChatPage(),
                  )), // MaterialPageRoute
                ) // GestureDetector
            ],
          ), // Row
        ) // Padding
      ],
    )); // Column, Container
  }
}
```

Fig. 19 Codul folosit pentru a randa Fig. 20

Pentru a crea interfața din Fig. 20 s-a folosit un widget fără stare, deoarece sunt prezente elemente care nu se modifică în timpul rulării aplicației. Widgetul părinte este unul de tip *Container*, care combină mai multe widget-uri pentru a forma o parte din layout, în

figură conține widgetul *Column*. Acesta este format la rândul lui din widget-uri de tip *Padding*, folosit pentru a contura marginile obiectelor, *Row*, care aranjează cele doua texte intr-o linie orizontală, *Text*, pentru a afișa textul celor doua mesaje, *TextStyle*, personalizează textul și *GestureDetector*, folosit pentru a detecta când se apasă pe textul „Vorbește cu Polly”.

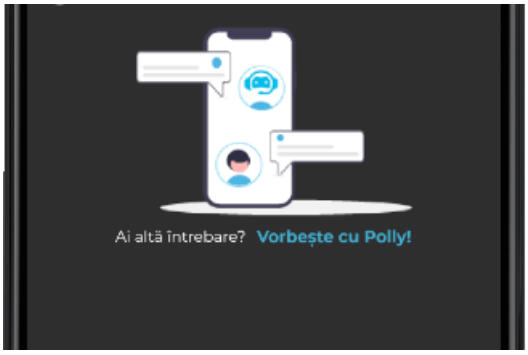


Fig. 20 Introducerea chatborului în aplicație

Backend

Backend-ul este implementat folosind limbajul Dart, asemănător cu Kotlin, din Android. Acesta este un limbaj orientat obiect, dezvoltat de Google; folosește pentru eliberarea memoriei un garbage collector. Pe lângă scrierea de cod, Dart oferă si tool-uri de formatare, analiză și testare a codului.

În Fig. 21 avem drept exemplu de cod Dart, implementarea funcției de căutare a persoanelor din aplicația ACS UPB mobile, care primește lista de persoane din baza de date și le caută pe acelea care conțineau cuvintele din query-ul de căutare.

```
78  Future<List<Person>> search(String query) async {
79    if (query.isEmpty) {
80      return <Person>[];
81    }
82    final List<Person> people = await fetchPeople();
83    final List<String> searchedWords = query
84        .toLowerCase()
85        .split(' ')
86        .where((element) => element != '')
87        .toList();
88    return people
89            .where((person) => searchedWords.fold(
90                true,
91                (previousValue, filter) =>
92                    previousValue &&
93                    person.name.toLowerCase().contains(filter.toLowerCase())))
94            .toList() ??
95        <Person>[];
96  }
97  }
```

Fig. 21 Backend căutare persoane

3.7.2 Firebase

Firebase este o bază de date NoSQL, care stochează datele în format JSON[11]. Acesta are o configurare ușoară și o interfață cu utilizatorul intuitivă. Firebase oferă clienților baze de date pentru autentificare sau pentru stocarea de informații (Firestore, Realtime Database, Storage). Totodată, oferă și servicii de hosting, învățare automată și funcții de cloud. În aplicația ACS UPB Mobile au fost folosite de la crearea aplicației serviciile de autentificare de utilizatori și stocare de date. Ulterior, s-au implementat funcții de cloud pentru backup-ul datelor stocate.

Firebase pune la dispoziția clienților servicii de analiză asupra bazelor date – Firebase Analytics. Acestea conțin grafice despre utilizatori: numărul de utilizatori, țara de unde s-au autentificat, timpul mediu petrecut în aplicație, dispozitivele de pe care au folosit aplicația etc. Aceste informații sunt foarte folositoare pentru a manageria interacțiunea utilizatorului cu aplicația, spre exemplu, timpul mediu de folosire a aplicației ACS UPB Mobile este de aproximativ un minut, ceea ce ne indică faptul că utilizatorul o folosește când are de o informație, o găsește rapid și ulterior, părăsește aplicația - Fig. 22.

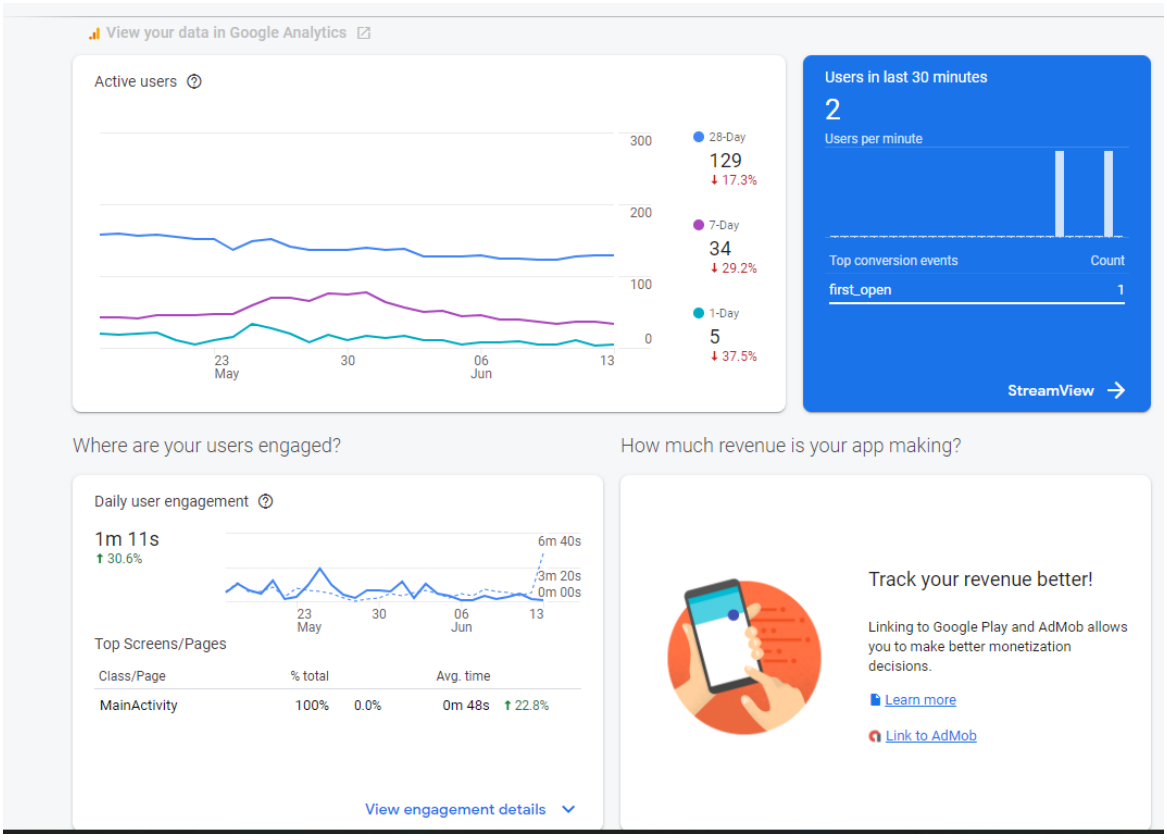


Fig. 22 Firebse Analytics

[11] https://www.json.org/json-en.html

CAPITOLUL 4. Rezultate

4.1 Testare

Testarea s-a realizat folosind rasa test, care împarte setul de date în 80% date de antrenare și 20% date de test. Printre rezultatele scriptului de test, s-a aflat și matricea de confuzie pentru clasificarea intențiilor, prezentă în Fig. 23, care dovedește faptul că predicțiile pe care le face modelul antrenat la clasificarea intențiilor, sunt corecte pentru toate tipurile de intenții definite în setul de date, atât pentru cele care definesc întrebări frecvente, cât și pentru cele care definesc acțiuni custom.

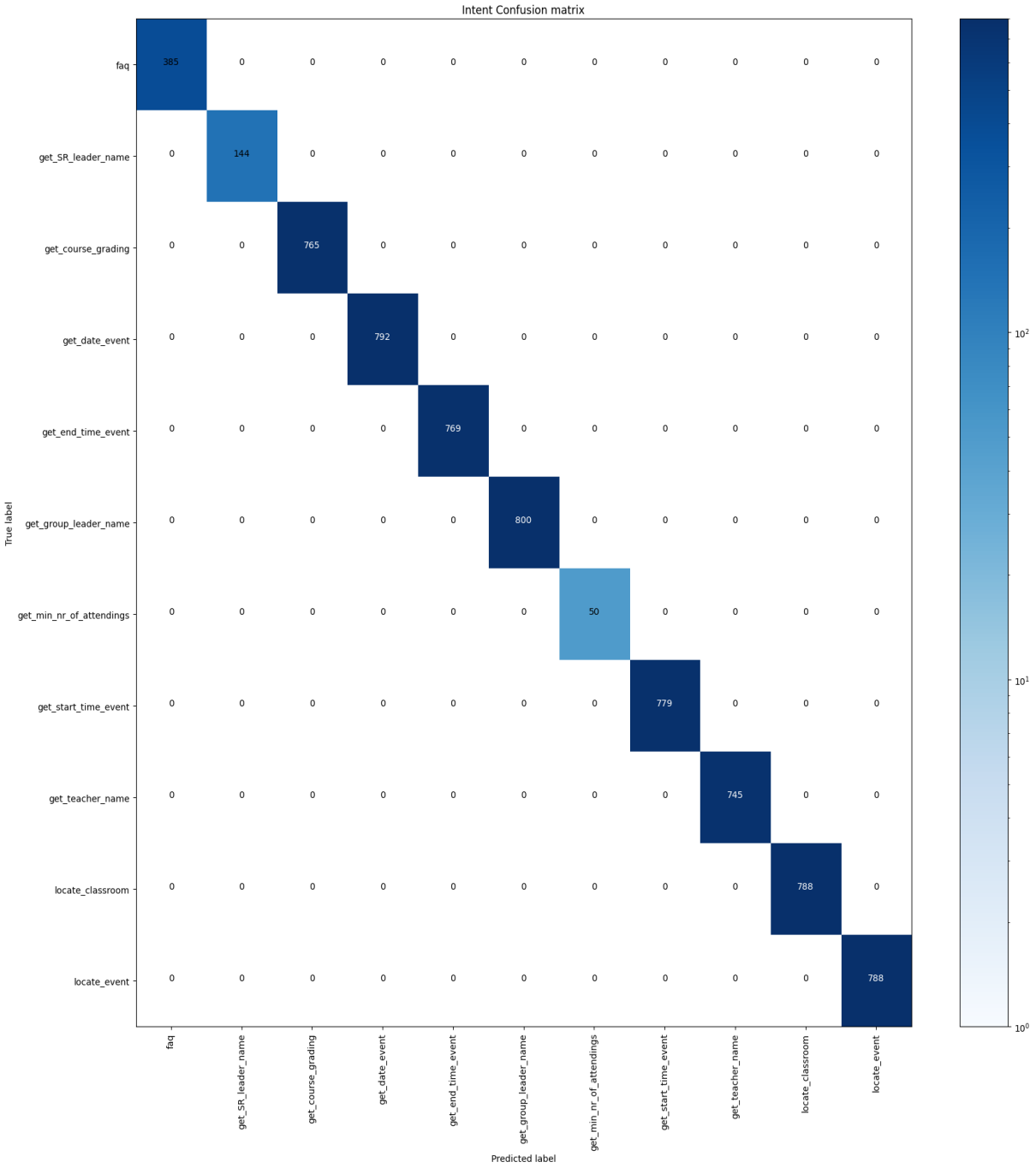


Fig. 23 Clasificarea intențiilor – matricea de confuzie

La fel ca predicția intențiilor, și răspunsul selectat de Response Selector este unul corect, de cele mai multe ori cu un grad de încredere foarte mare - Fig. 24. Histograma arată că acest grad de încredere, etichetat în figură drept „confidence”, variază între 0.94 până aproape de 1, cele mai multe valori fiind cele cu cuprinse în intervalul (0.99, 1). Acest fapt duce la garantarea unui răspuns satisfăcător la întrebările de tip faq adresate agentului, care folosesc Response Selector pentru a selecta un răspuns din datele de antrenare.

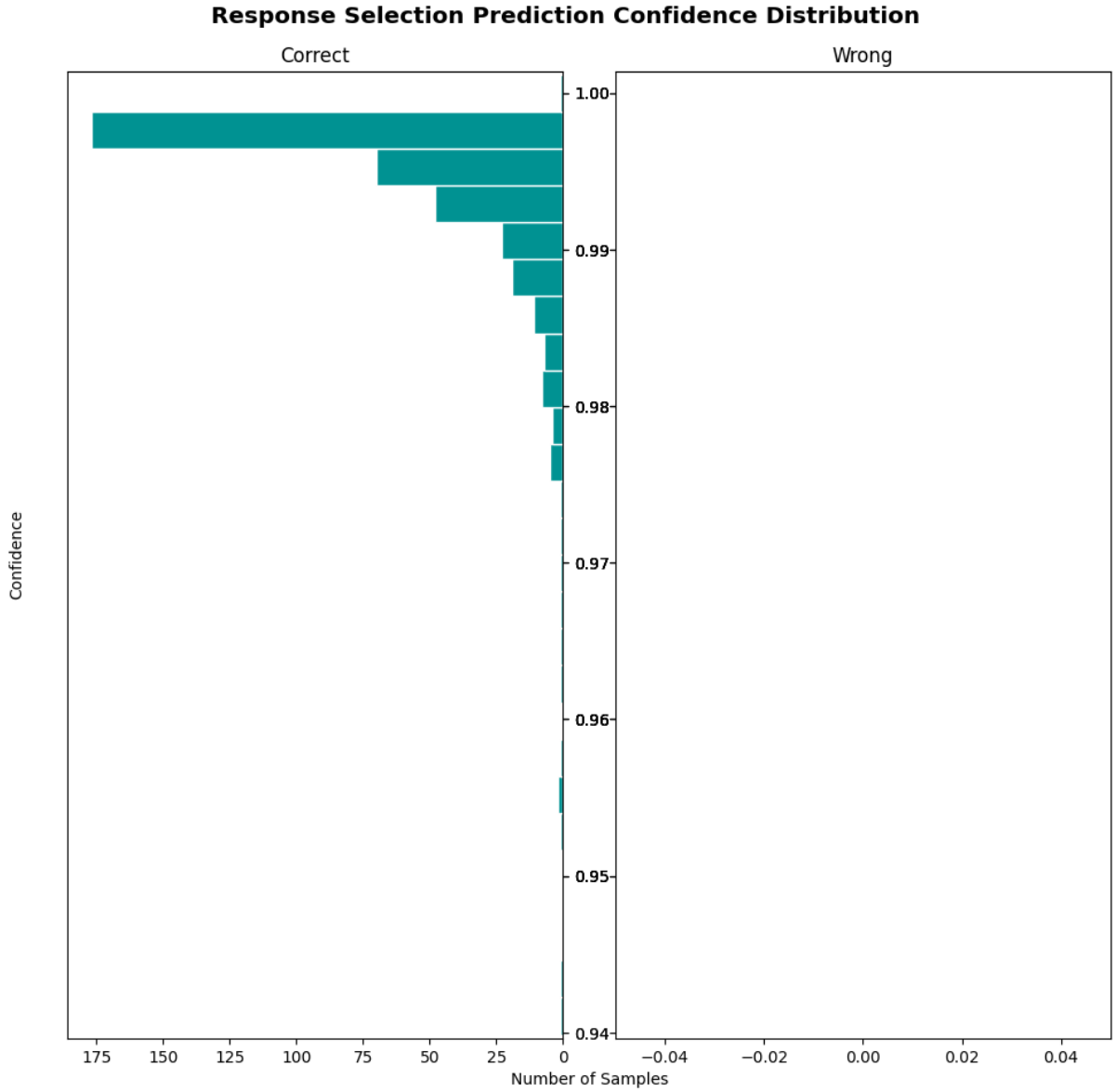


Fig. 24 Histograma răspunsurilor selectate de Response Selector

4.2 Performanțe

O mică problemă pe care o întâmpină modelul are loc la extragerea entităților, atunci cand, de câteva ori, nu este recunoscută o entitate de tip classroom sau de tip event. Tot la fel se întâmplă și in sens invers, când într-un enunț fară entități se identifică entități de tip classroom - Fig. 25. Această problemă este posibil să apară datorită asemănării gramatice dintre abrevierea numelui unui eveniment și denumirea unei săli; spre exemplu, există corpul EG, care face partea din denumirea multor săli de curs/laborator, dar există și materiile EG (Economie generală) sau EGC (Elemente de grafică pe calculator)

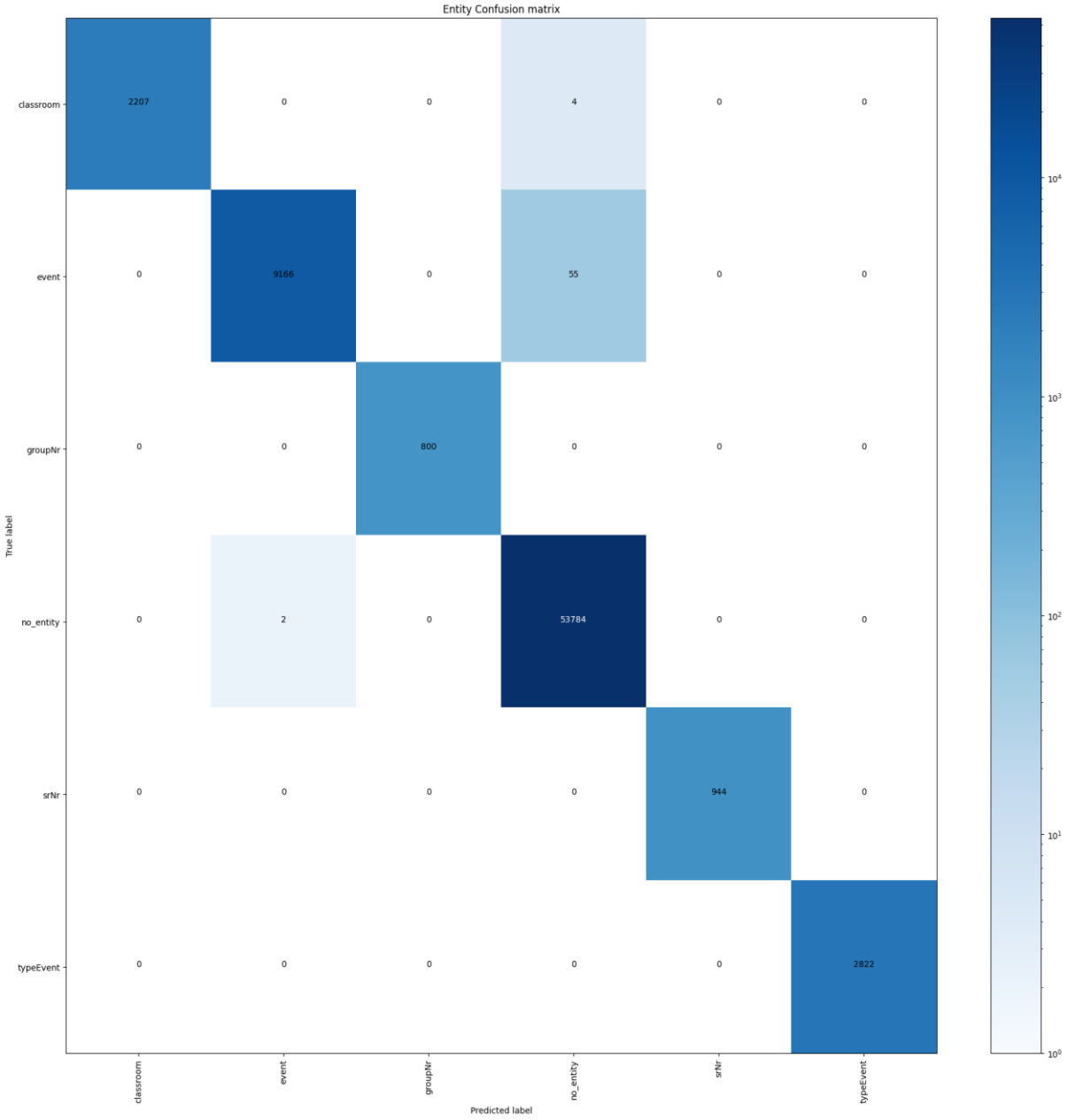


Fig. 25 Clasificarea entităților – matricea de confuzie

CAPITOLUL 5. Discuții

După finalizarea implementării agentului conversațional, au avut loc o serie de discuții cu membrii StudentHub, asupra caracteristicilor care au fost realizate, și despre eficiența acestora.

5.1 Întrebările frecvente

Deși unele întrebări frecvente prezente în corpus-ul chatbot-ului existau în aplicația ACS UPB Mobile în secțiunea dedicată acestora, completarea lor a fost necesară pentru ca toate informațiile să poată fi obținute dintr-un singur loc. În urma discuțiilor asupra acestora, am stabilit o serie de intenții categorisite drept cele mai folositoare, pe care le voi discuta mai jos:

- conectare la rețelele wifi Eduroam; Majoritatea studenților văd că există o rețea wifi disponibilă numită Eduroam, dar nu sunt mulți cei care știu că se pot conecta la acesta folosind contul de student. Răspunsul oferit în urma acestei intenții oferă pașii folosiți pentru conectare, împreună cu credențialele care trebuie folosite
- schimbarea unei grupe/serii; De obicei, în primul an de facultate și în ultimul (când se împart seriile pe specializări) se efectuează cele mai multe schimbări de acest gen. Este important ca un student să știe că acest lucru se poate face doar în primul semestru, pentru a se gandi de la început ce să facă. Totodată, a fost considerată util faptul că un student este anunțat faptul că sanșele ca o cerere de schimbare are șanse mai mari să se aprobe dacă există persoane cu care să facă schimb
- deschidere de ticket pentru suport; Personal, nu am știut de existența acestei posibilități până în anul 4 de studiu. De asemea, și membrilor de echipă li s-a părut utilă promovarea acestei modalitate de suport pentru studenți, pentru a putea ușura discuțiile cu partea administrativă a facultății
- devenire șef de serie/grupă; La această intenție am căzut de acord că este foarte folositoare studenților de anul 1, care doresc să se implice pe această latură. Fiind la început, încă nu au apucat să se cunoască, așa că, studentul, aflând că alegerea se face pe baza de vot, va fi stimulat să încerce să se afirme și să se facă plăcut printre colegii săi
- responsabilități șef de grupă/serie; Un student, pe lângă dorința de a deveni șef de serie/grupă, acesta trebuie să știe și ce atribuțiuni îi revin odată cu punerea în funcție. Pentru aceasta, studentului îi sunt prezentate principalele sarcini pe care le-ar putea avea
- model de cerere; Nu mulți studenți știu să redacteze o cerere, așa că răspunsul la această intenție oferă un model de cerere pentru eliberarea unei adeverințe de

student, cea mai comună cerere dintre cele primite de către personalul de la secretariat. Cu ajutorul acesteia se poate vedea cum se adresează într-un mod formal unei persoane din cadrul administrativ, în acest model adresarea se face către domnul decan al Facultății de Automatică și Calculatoare

5.2 Întrebările despre evenimente

Dintre aceste întrebări, am ajuns la concluzia că cea mai utilă intenție este cea de localizare a unui eveniment. Am hotărât acest lucru deoarece majoritatea dintre studenți pleacă la curs, reținând ora la care începe, dar, până să ajungă în campus, uită sala unde se ținea. Astfel, acesta va putea afla locația prin adresarea unei simple întrebări, în care să se specifice denumirea evenimentului și tipul acestuia (curs, seminar sau laborator).

Întențiile care privesc timpul de desfășurare al evenimentelor, la fel ca cele de localizare au primit aprecierea că tratează și cazul în care un eveniment se poate desfășura de mai multe ori în decursul unei săptămâni, și se returnează toate rezultatele (cum ar fi cursurile, care pot avea loc de doua ori pe săptămână, în funcție de paritatea acesteia).

5.3 Interfața cu utilizatorul

Pagina de căutare am stabilit că arată bine, rezultatele sunt calculate corect iar modul de interacțiune este sugestiv. Materiile returnate ca rezultate sunt puțin cam multe; acestea ar trebui sortate în funcție de relevanța utilizatorului (să apară prima dată cele de la seria din care face parte).

Pagina de chat este ușor de folosit, fiind construită folosind elementele comune prezente într-o pagină obișnuită de mesagerie. Ar putea fi introduse posibile animații atunci când se așteaptă un mesaj de la chatbot.

5.4 Interacțiunea cu agentul conversațional

În urma discuțiilor am decis că modul de interacțiune este unul satisfăcător. Agentul știe să răspundă, în mare parte, și la întrebările scrise greșit gramatical. Am primit sugestia ca dialogul să poata avea mai multe replici interactive, provenite de la chatbot, cum ar fi: mai multe moduri de salut, formulări de încheiere ale conversației, adăugarea de documente unde ar putea fi utile (cum ar fi un model de cerere.

5.7 Îmbunătățiri

5.7.1 Secțiunea de sugestii

Atât pagina de căutare cât și pagina se sugestii ar putea avea o rubrică de sugestii de întrebări pe care le-ar putea adresa chatbotului. În pagina de căutare, sugestiile ar putea fi filtrate în funcție de cuvintele căutate, iar în pagina de chat, ar fi util ca sugestiile să fie din categoria de întrebări frecvente, pentru a putea ști utilizatorul că poate să ii adreseze și astfel de întrebări. Selectarea unei sugestii va putea fi introdusă direct în dialog, drept un mesaj primit de la utilizator.

5.7.2 Adăugare de mai multe intenții de tip întrebări frecvente

Chatbotul ar trebui să știe să răspundă și la întrebări legate de aplicație:

- Cum văd orarul în aplicație?
- Cum pot să-mi adaug materii care nu fac parte din seria mea? (materii opționale, din anii trecuți)
- Cum pot să contribui la dezvolatea aplicației?
- Care este modalitatea prin care pot contribui la aplicație cu date?
- Cum pot să fac o recenzie la o materie?

Totodată, ar putea fi adăugate întrebări legate de ligi studențești, asociații, stagii de practic, cum ar fi:

- Ce este LSAC? Cum pot deveni membru?
- Ce este StudentHub? Cum pot să imi fac stagiul de practică la StudentHub?
- Ce site-uri există unde îmi pot găsi stagii de practică?

5.7.3 Comportamentul agentului conversațional

Modul de interacțiune al agentului ar putea fi îmbunătățit, acțiunile ar putea fi împărțite pe mai multe bucăți (ex: dacă utilizatorul nu specifică tipul de eveniment într-o întrebare, acesta ar putea fi ulterior despre acest lucru). Totodată, comportamentul ar putea fi îmbunătățit prin reținerea contextului dialogului și prin folosirea istoricului conversației în momentul interpretării unui input, cum ar modelul de conversație din Fig. 26.

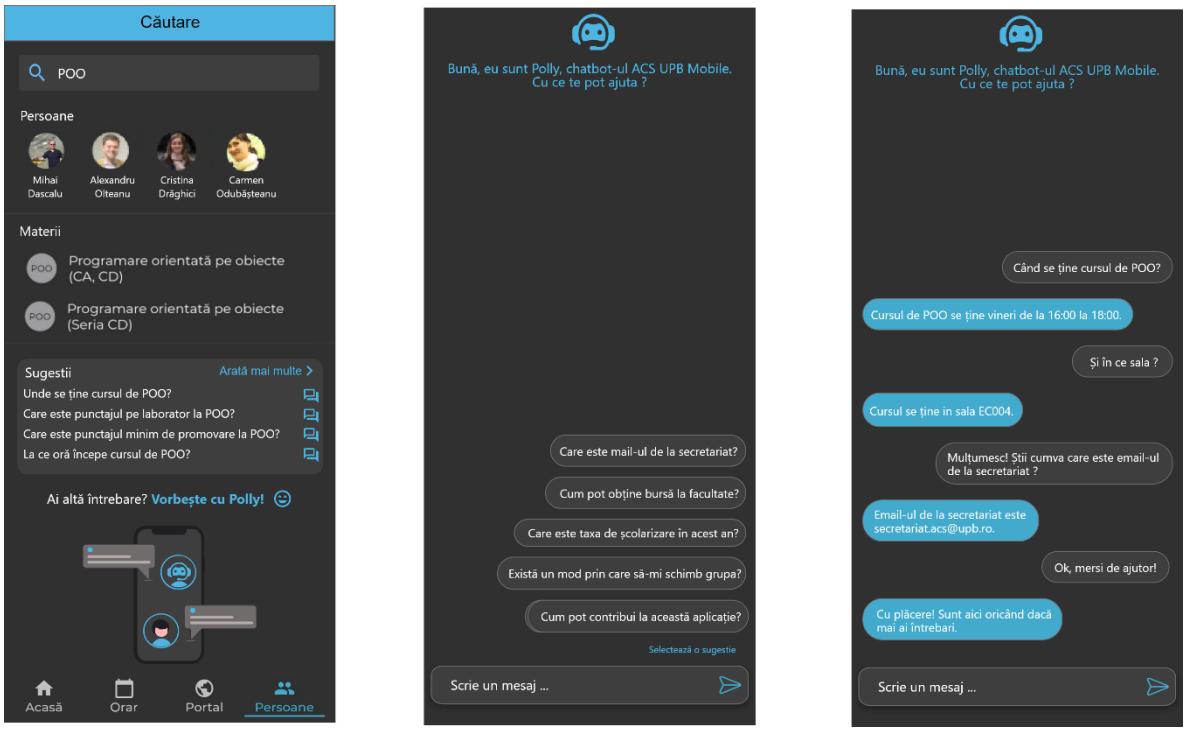


Fig. 26 Mock-up îmbunătățiri

CAPITOLUL 6. Concluzii

Implementarea agentului s-a realizat pe mai multe etape, fiecare depinzând în mod direct una de cealaltă. O contribuție în dezvoltare a avut-o și programul Innovation Labs 2021[12], aplicația ACS UPB Mobile, în care este integrat agentul conversațional, fiind unul dintre proiectele care fac parte din acest program, din partea asociației în curs de formare, StudentHub. Mentorii din cadrul Innovation Labs au contribuit atât la identificarea funcționalităților pe care ar trebui să le aibă chatbot-ul cât și pe partea de experiență a utilizatorului cu aceste pagini din aplicație, dându-ne sfaturi despre cum ar trebui sa îmbunătățim rezultatele întoarse de pagina de căutare, cum ar trebui să fie introdusă pagina de chat în aplicație și sfaturi despre restilizarea interfeței cu utilizatorul. S-a reușit implementarea unui model care reușește să răspundă la majoritatea întrebărilor pe care le-ar putea avea un student de a lungul anilor universitari; Cele mai importante îmbunătățiri are ar putea fi aduse ar fi prelucrarea enunțurilor primite de la utilizator ținând cont de contextul dialogului și adăugarea de noi întrebări frecvente la care să poată răspunde.

Fiind un proiect în desfășurare, agentul conversațional va primi o serie de noi funcționalități odată cu trecerea timpului. Pe lângă îmbunătățirile relatate în capitolul de discuții, avem în vedere să creștem numărul de funcționalități.

Integrarea de componente de navigare prin campus

Odată cu implementarea unei serviciu prin campusul studențesc, care preconizăm că va fi gata în vara anului 2021, aceasta ar putea fi introdusă și in pagina de chat, prin intenții de forma „Cum ajung în EC301?” care vor redirecta utilizatorul în modul de navigare, cu destinația aleasă prin extragerea entității de tip sală de curs.

Integrarea cu recenziile date de utilizatori materiilor de curs

După apariția versiunii care oferă posibilitatea utilizatorilor să ofere recenzii la materiile pe care le-au avut în decursul unui an universitar, studentul ar putea beneficia de informații extrase din acestea; folosind dialogul cu chatbot-ul, acesta ar putea cere informații despre materiile de la facultate, bazate pe recenziile studenților din anii trecuți, având posibilitatea de a vizualiza grafice cu timpul mediu de rezolvare a temelor, dificultatea acestora, cât de greu le-a fost să strângă punctajul de intrat în examen sau cu notele pe care le-a primit cadrul didactic la anumite capitole, precum: modul de relaționare cu studenții, dificultatea informațiilor livrate sau cantitatea de informații pe care s-a pus accent.

[12] https://www.innovationlabs.ro/

Integrarea cu planificatorul de sarcini

În aplicație este în desfășurarea un proiect care constă într-un planificator pentru sarcinile primite la facultate (teme, proiecte), unde își pot seta o țintă a punctajului pe care doresc să o atingă, și pot vedea progresul pe care îl au pentru atingerea acesteia. Prin intermediul chatului, se va face o intenție de planificare a unei sarcini, care v-a face acest lucru în mod automat (ex: Seteaza-mi , te rog, o țintă de 2/3 puncte pentru tema 1 la proiectul de la Analiza Algoritmilor), prin interogarea bazei de date și setarea acesteia în funcție de entitățile extrase.

Bibliografie

[1] MobileMonkey. 2021. „*8 Interactive Websites with Chatbots: Web Chat Examples for Marketing, Sales & Customer Support”*. [online] Disponibil la: https://mobilemonkey.com/blog/websites-with-chatbots [Accesat 28. 01. 2021].

[2] Netomi. 2021. „*11 Best AI Chatbot Softwares for 2021 [Key Features]* ”- Netomi. [online] Disponibil la: https://www.netomi.com/best-ai-chatbot [Accesat 28. 01 2021].

[3] SmallBizGenius. 2021. „*The Future Is Now - 37 Fascinating Chatbot Statistics”*. [online] Disponibil la: https://www.smallbizgenius.net/by-the-numbers/chatbot-statistics/#gref [Accesat 02. 02. 2021].

[4] Machinelearningplus.com. 2021. [online] Disponibil la: https://www.machinelearningplus.com/nlp/chatbot-with-rasa-and-spacy [Accesat 02. 02. 2021].

[5] Bocklisch, T., Faulkner, J., Pawlowski, N. and Nichol, A., 2017. „*Rasa: Open Source Language Understanding and Dialogue Management”*. [online] arXiv.org. Available at: https://arxiv.org/abs/1712.05181 [Accesat 02. 02. 2021].

[6] Rakesh Kumar Sharma, Manoj Joshi, 2020, „*An Analytical Study and Review of Open Source Chatbot framework”, RASA*, International Journal of Engineering Research, V9(06).

[7] Documentație Spacy, (n.d) Disponibila la https://spacy.io/usage/spacy-101 [Accesat 03.02.2021].

[8] Models, H., 2021. „*Transformers In NLP | State-Of-The-Art-Models”*. [online] Analytics Vidhya. Disponibil la: https://www.analyticsvidhya.com/blog/2019/06/understanding-transformers-nlp-state-of-the-art-models [Accesat 03. 02. 2021].

[9] Ria Kulshrestha, 2020, „*Transformers”*. [online] Disponibil la: https://towardsdatascience.com/transformers-89034557de14 [Accesat 10. 02. 2021].

[10] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V.Le, Ruslan Salakhutdinov, 2019, „*Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context”*, Carnegie Mellon University, Google Brain

[11] Rebuffel, C., Soulier, L., Scoutheeten, G. and Gallinari, P., 2020. „*A Hierarchical Model for Data-to-Text Generation”.* Lecture Notes in Computer Science, pp.65-80.

[12] Ashish Vaswni, Naom Shazeer, Niki Parmar, Jakob Uskoreit, Llion Jones, Aidan N.Gomez, Lukasz Kaiser, Illia Polosukhin, 2017*, „Attention is All You Need”*, Advances in Neural Information Processing Systems 30 (NIPS 2017)

freeCodeCamp.org. 2021. „*How to Get Started with Firebase Using Python”*, [online] Disponibil la: https://www.freecodecamp.org/news/how-to-get-started-with-firebase-using-python/ [Accesat 10. 02. 2021]

[13] Vipul Raheja, Joel Tetreault, 2019, „ *Dialogue Act Classification with Context-Aware Self-Attention”*, Association for Computational Linguistics

[14] Thomas Wolf, 2019, Medium. 2021. „*How to build a State-of-the-Art Conversational AI with Transfer-Learning”*. [online] Disponibil la https://medium.com/huggingface/how-to-build-a-state-of-the-art-conversational-ai-with-transfer-learning-2d818ac26313 [Accesat 11. 02. 2021]

[15] Chatcompose.com. (n.d.). „*FAQ chatbot: How to design a chatbot for frequent questions”*, [online] Disponibil la: https://www.chatcompose.com/faqchatbot.html [Accesat 30. 03. 2021].

[16] Paranjape, A., See, A., Kenealy, K., Li, H., Hardy, A., Qi, P., Sadagopan, K. R., Phu, N. M., Soylu, D., and Manning, C. D. (2020). „*Neural generation meets real people: Towards emotionally engaging mixed-initiative conversations”*. 3rd Proceedings of Alexa Prize.

[17] Documentație Rasa (n.d.), Disponibilă la: https://rasa.com/docs/rasa/ [Accesat 02. 04. 2021]

[18] Daniel Juradsky, James H. Martin, 2020, „*Speech and Language Processing*,” Capitolul 24: „*Chatbots & Dialogue Systems”*

[19] Git-scm.com. (n.d.). *Git – „gittutorial Documentation”*, [online] Disponibil la: https://git-scm.com/docs/gittutorial [Accesat 03. 04. 2021]

[20] Ioana Alexandru, Vlad Posea, 2020, „*Design and Implementation of a Cross-Platform Mobile Application That Facilitates Student Collaboration”*, Disponibil la https://github.com/student-hub/paper/blob/master/Design_and_implementation_of_a_cross_platform_mobile_application_that_facilitates_student_collaboration.pdf

[21] Documentație Flutter (n.d.), https://flutter.dev/docs [Accesat 07. 04. 2021].

[22] Mockitt.wondershare.com. 2021. „*How to Create Mockup with Adobe XD Step-by-Step*”, [online] Disponibil la: https://mockitt.wondershare.com/adobe-xd/adobe-xd-mockup.html [Accesat 11. 04. 2021].

[23] Villanueva Jr., G., 2020. „*Design Architecture of FAQ Chatbot for Higher Education Institution"*, Journal of Advanced Research in Dynamical and Control Systems, 12(01-Special Issue), pp.189-196.

Ali, F. and Aydah, S., 2012. *„Development of Prototype Chat System Using Mobile Platform for Disable People"*, Procedia - Social and Behavioral Sciences, 57, pp.33-39.

[24] Andrie Asmara, R., Al Huda, F. and Nur Handayani, A., 2018. „*Design and Implementation of Blind Runner Guide Android Mobile Application for the Visual Impairment User Experience"*, International Journal of Engineering & Technology, 7(4.44), p.65.

[25] Dyna.mo. 2021. [online] Disponibil la: https://dyna.mo/en/blog/flutter-building-apps-beyond-ui/ [Accesat 20. 04. 2021]

[26] Medium. 2021. „*Building beautiful, flexible user interfaces with Flutter, Material Theming, and official Material…."* [online] Disponibil la: https://medium.com/flutter/building-beautiful-flexible-user-interfaces-with-flutter-material-theming-and-official-material-13ae9279ef19 [Accesat 28. 04. 2021]

[27] Codecademy. 2021. „*HTTP Requests* | *Codecademy*", [online] Disponibil la: https://www.codecademy.com/articles/http-requests [Accesat 10. 05. 2021].

[28] Toptal Engineering Blog. (n.d.). „*Dart 2 for C# and Java Developers*" [online] Disponibil la: https://www.toptal.com/dart/dartlang-guide-for-csharp-java-devs [Accesat 12. 05. 2021]

[29] Pipedream. 2021. „*Integrate the HTTP / Webhook API with the Rasa API – Pipedream",* [online] Disponibil la: https://pipedream.com/apps/http/integrations/rasa [Accesat 02. 06. 2021]

[30] Data School. (n.d.). „*Simple guide to confusion matrix terminology"*, [online] Disponibil la: https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/ [Accesat 25. 06. 2021].

Anexă

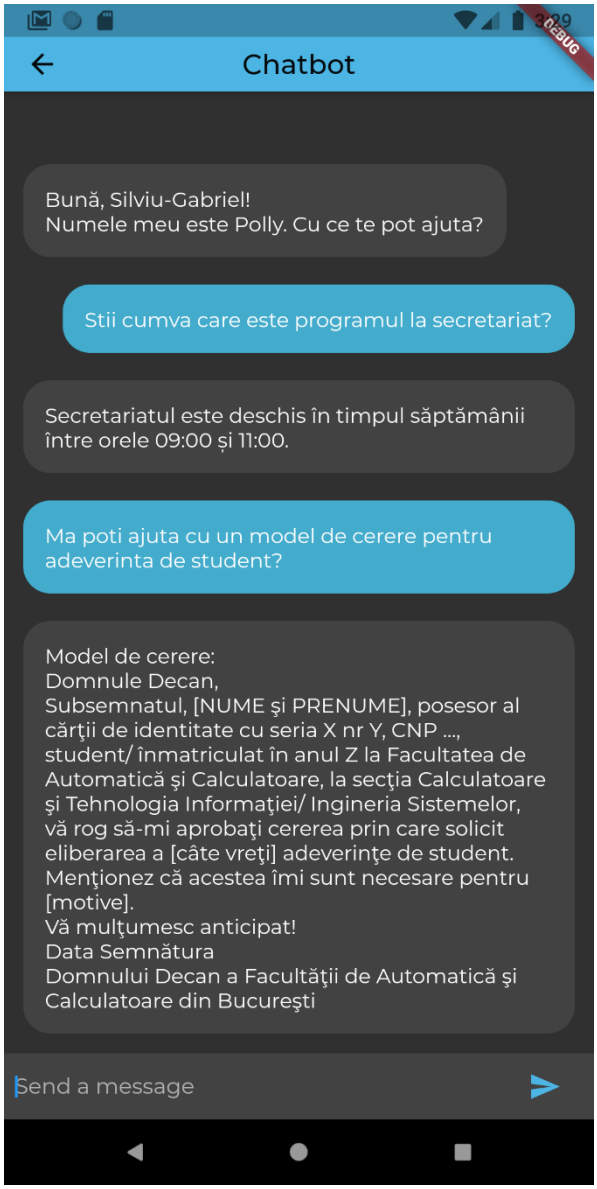


Fig. 27 Exemplu de dialog folosind întrebări frecvente

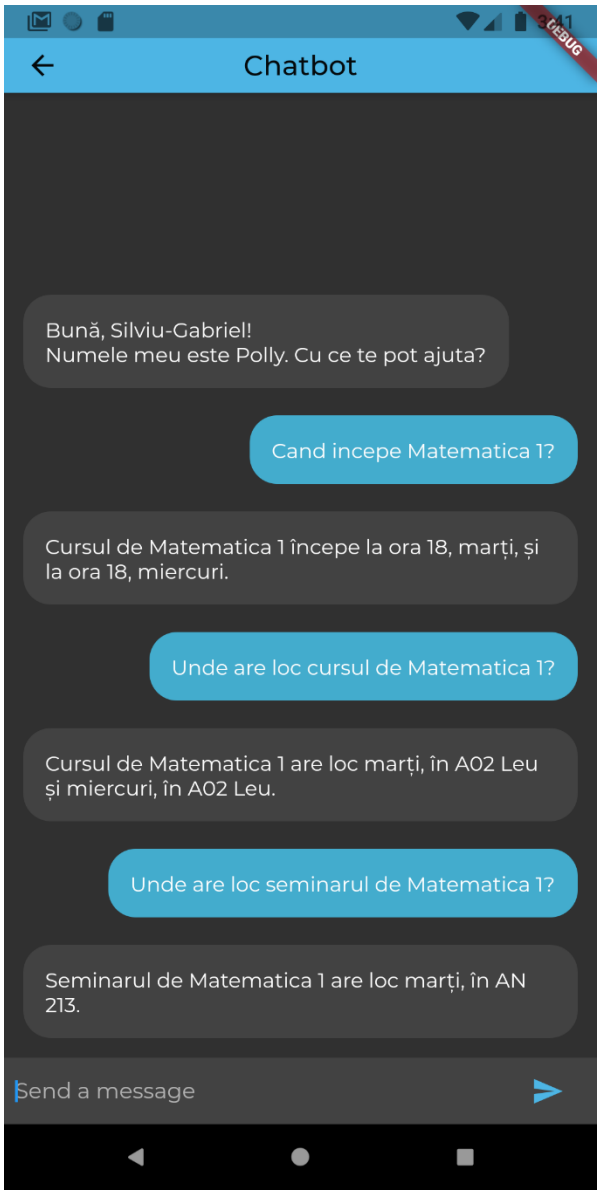


Fig. 28 Localizare evenimente

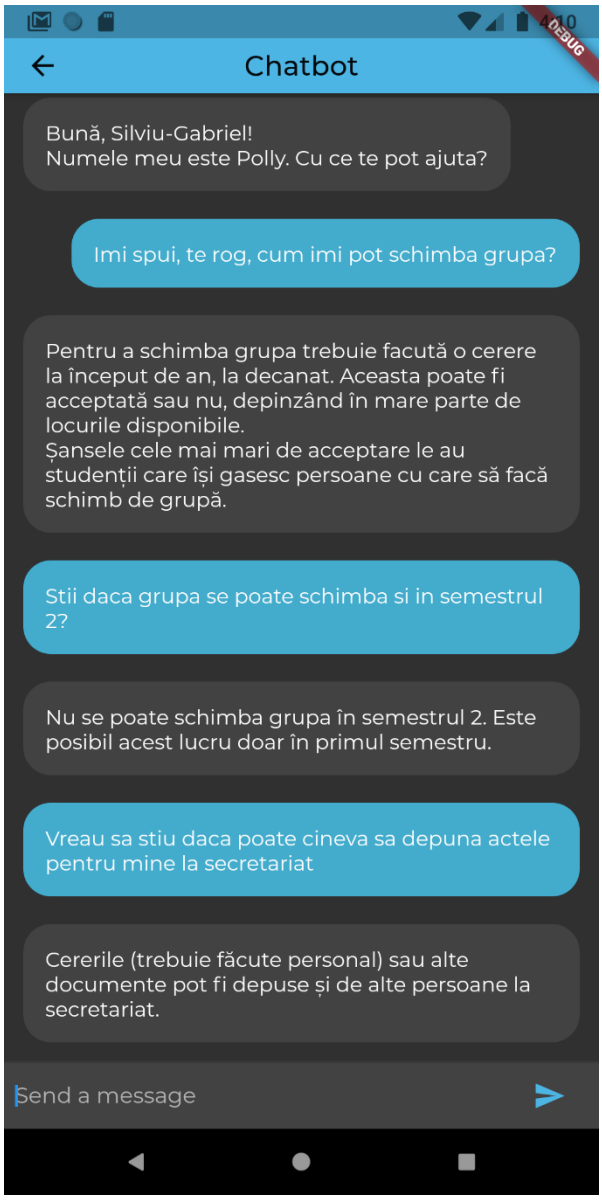


Fig. 29 Întrebări frecvente – Schimb de grupă

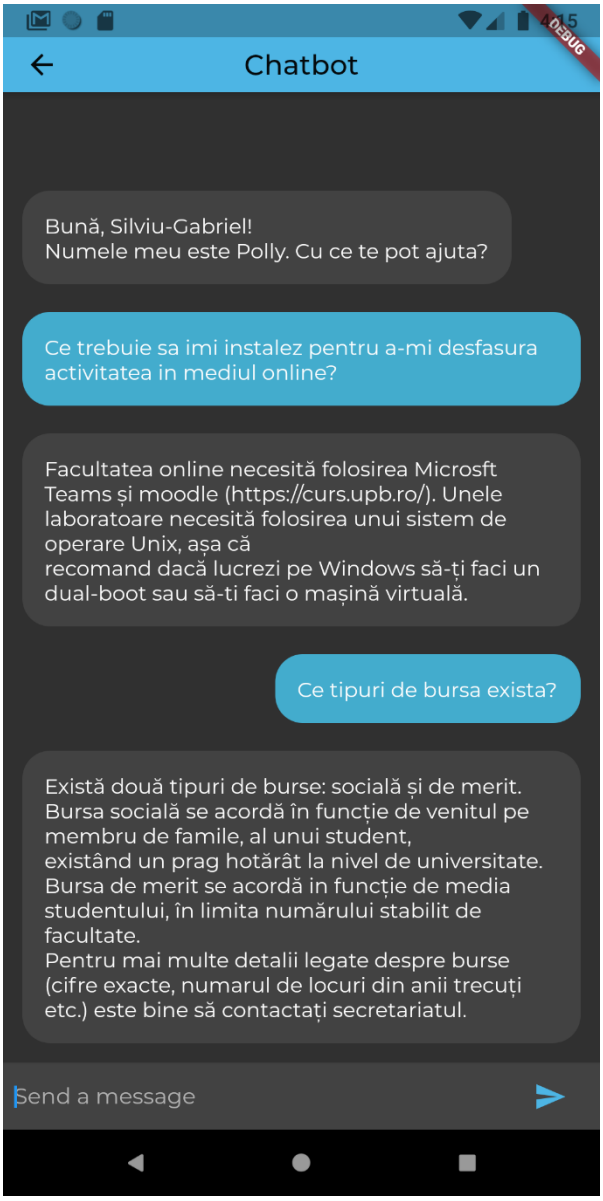


Fig. 30 Întrebări frecvente facultate online și bursă

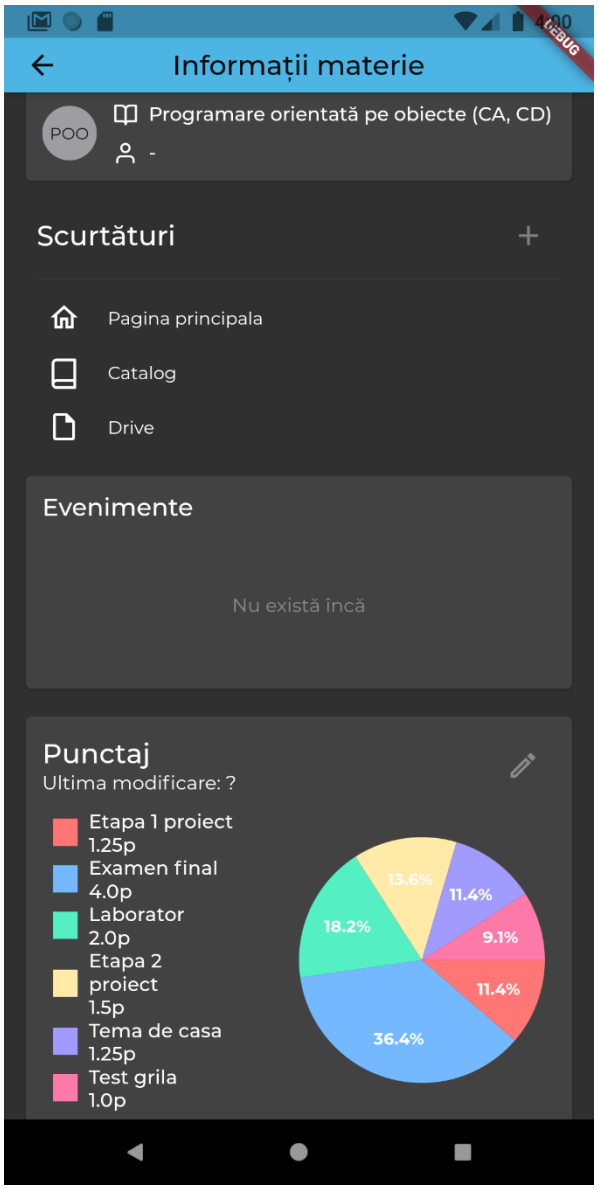


Fig. 31 Informațiile deținute de aplicație despre materii

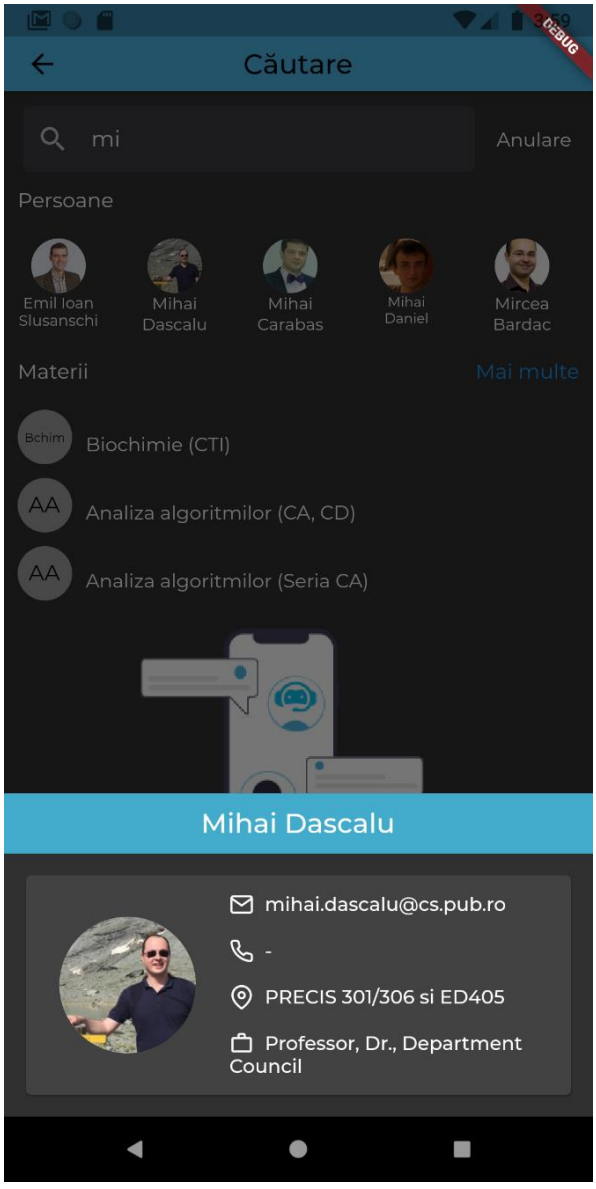


Fig. 32 Informații deținute în aplicație despre cadrele didactice

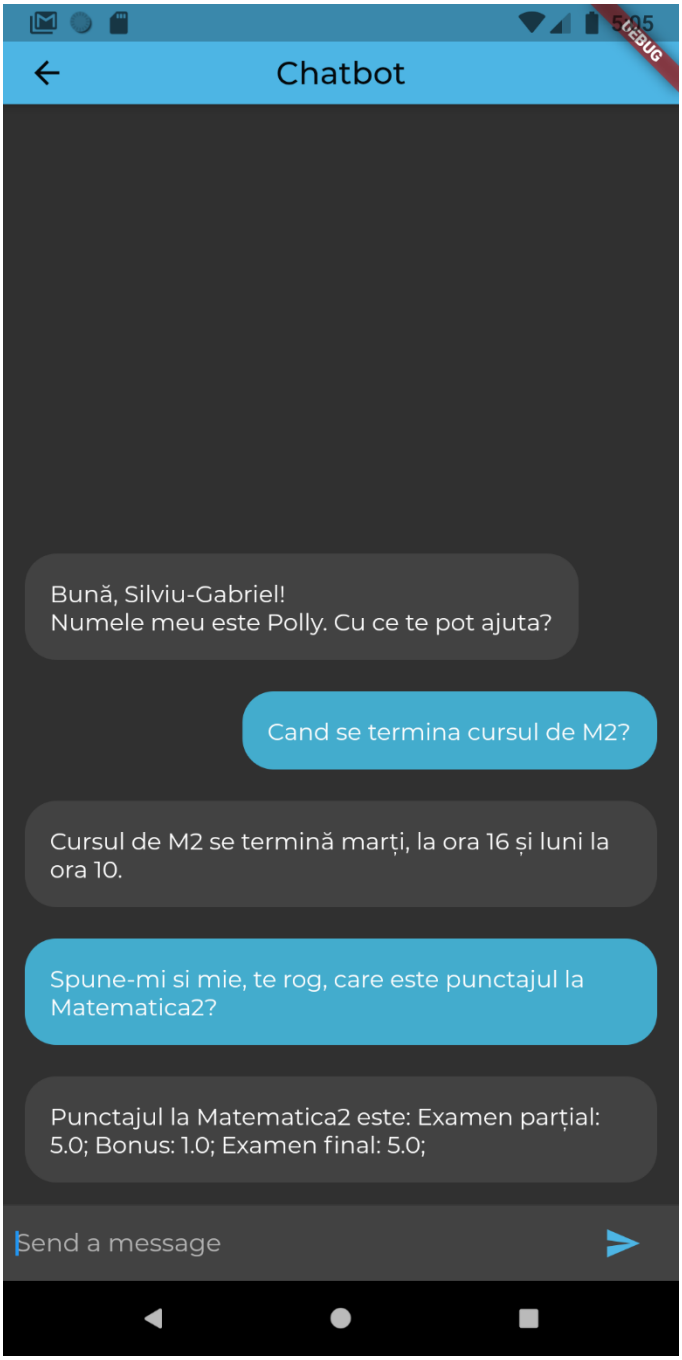


Fig. 33 Finalizare eveniment și informații punctaj