Roboţii şi Societatea: Sisteme Cognitive pentru Roboţi Personali şi Vehicule Autonome

Număr contract 72PCCDI **din** 01/03/2018

Etapa II

Raport ştiinţific şi tehnic

Consorțiu

Coordonator proiect: UNIVERSITATEA POLITEHNICA DIN BUCURESTI
P1: INSTITUTUL DE CERCETARI PENTRU INTELIGENTA ARTIFICIALA "MIHAI DRĂGĂNESCU"
P2: INSTITUTUL DE MATEMATICA "SIMION STOILOW" AL ACADEMIEI ROMANE
P3: UNIVERSITATEA TEHNICA DIN CLUJ - NAPOCA
P4: UNIVERSITATEA "DUNAREA DE JOS"

Rezumat

ROBIN este un proiect centrat pe utilizator care proiectează sisteme şi servicii pentru utilizarea roboților într-o societate digitală interconectată și permite companiilor realizarea de produse și servicii complexe, inteligente și performante destinate utilizatorilor și societății în ansamblu. Proiectul se referă la o gamă diversă de roboţi: roboţi asistivi pentru sprijinul persoanelor cu nevoi speciale, roboţi de interacţiune cu clienţii şi roboţi software care pot fi instalaţi pe vehicule în scopul realizării unei conduceri autonome sau semi-autonome. Proiectul combină tehnici și tehnologii avansate de inteligență artificială, interacțiune om-robot, interacțiune cu un mediu pervasiv și prelucrări în Cloud.

Proiectul ROBIN este alcătuit din cinci proiecte componente:

ROBIN-Social: ***Roboți sociali cognitivi pentru o societate digitală a viitorului, centrată pe utilizator*.** Scopul proiectului este realizarea unei soluții integrate și ușor configurabile pentru personalizarea roboților asistivi (pt. persoane cu nevoi speciale) și sociali (pt. clienți), soluție bazată pe tehnici avansate de inteligență artificială care pot oferi roboților un caracter cognitiv și autonom.

ROBIN-Car: ***Sisteme cognitive autoinstruibile pentru vehicule autonome*.** Scopul proiectului constă în dezvoltarea de metode de vedere computaţion- ală care să rezolve o gamă mai largă și mai sofisticată de sarcini de asistenţă în pilotaj, realizarea unor module inteligente pentru „Hands-off driving” și „Automated driving” și un sistem prototip care va fi testat pe un autovehicul electric semi-autonom.

ROBIN-Context: ***Servicii inteligente, dependente de context, pentru personalizarea roboţilor şi conducere autonomă*.** Scopul proiectului este crearea unei platforme suport pentru definirea / reprezentarea semantică și gestiunea facilă și eficientă a datelor ce devin context în scenarii de asistență robotică personalizată și sisteme ADAS (Advanced Driving Assistance Systems).

ROBIN-Dialog: ***Înțelegerea și sinteza limbajului natural pentru roboți asistivi și interacțiunea cu mediul*.** Scopul proiectului presupune dezvoltarea unei serii de scenarii pentru câteva micro-lumi și tehnologia de prelucrare a limbii române pentru dialoguri situaționale în aceste micro-lumi.

ROBIN-Cloud: ***Platformă informatică distribuită pentru sisteme pervasive, în contextul sistemelor Cloud-Edge***. Scopul proiectului îl constituie crearea unei platforme suport pentru colectarea de date provenind de la senzorii unor sisteme suport pentru roboți (ex. IoT), oferirea de mecanisme de procesare și învățare combinând modele Cloud cu dispozitive aflate aproape de sursa de colectare, oferirea de biblioteci suport pentru procesare inteligentă / semantică a datelor, și în final dezvoltarea de servicii Web centrate spre agenți economici interesați de folosirea datelor stocate la nivelul platformei.

Site proiect http://aimas.cs.pub.ro/robin/

CUPRINS

1. Introducere

ROBIN este un proiect centrat pe utilizator care proiectează sisteme şi servicii pentru utilizarea roboților într-o societate digitală interconectată și permite companiilor realizarea de produse și servicii complexe, inteligente și performante destinate utilizatorilor și societății în ansamblu.

Proiectul se referă la o gamă diversă de roboţi: roboţi asistivi care vin în sprijinul persoanelor cu nevoi speciale, roboţi de interacţiune cu clienţii şi roboţi software care pot fi instalaţi pe vehicule în scopul realizării unei conduceri autonome sau semi-autonome.

Proiectul combină tehnici și tehnologii avansate de inteligență artificială, interacțiune om-robot, interacțiune cu un mediu pervasiv, arhitecturi și prelucrări în Cloud, arhitecturi Cloud-Edge si Cloud-Robotics.

Figura 1.1 prezintă viziunea generală a proiectului complex din punct de vedere al aplicațiilor dezvoltate și al tehnologiilor utilizate. Se pune astfel în evidență interacțiunea cu roboții de tip umanoid sau semiumanoid pentru realizarea roboților asistivi, dar și cu roboții instalați pe autovehicule în scopul conducerii autonome sau semi-autonome, și se reliefează principalele tehnologii utilizate, de exemplu tehnologii de computer vision bazate pe rețele de convoluție, tehnologii de tip Cloud și Cloud-Edge, și tehnologii de integrare a datelor provenite de la sensori.

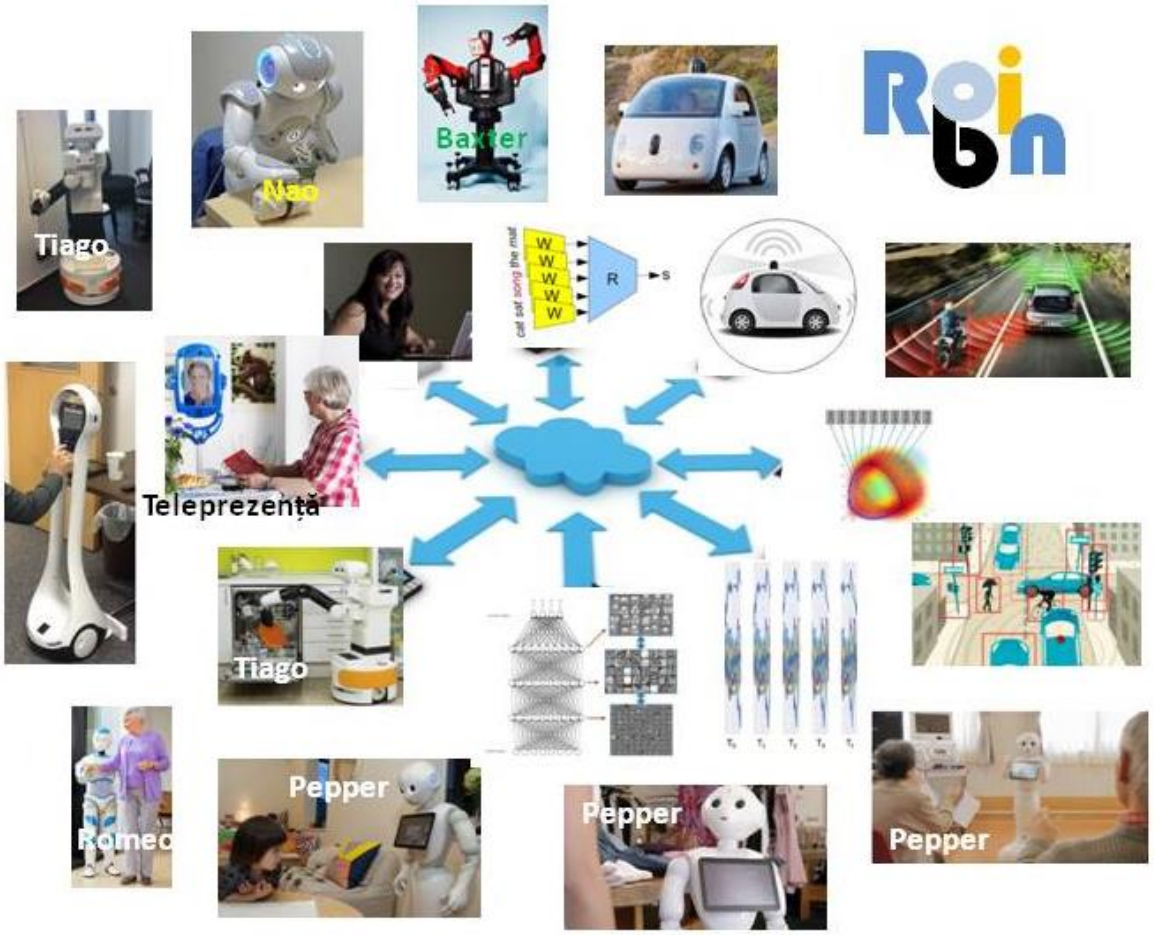


Figura 1.1.Roboții și societatea

Figura 1.2 prezintă conceptul general al proiectului, atât funcțional cât și arhitectural, prin care avem în vedere dezvoltarea de sisteme și servicii științifice și tehnologice performante, construite peste platforme interoperabile, cu capacități de structurare și procesare inteligentă a datelor și oferite comunității științifice și agenților economici.

Tot în Figura 1.2 se indică de asemenea cele 5 proiecte componente ale proiectului complex ROBIN, și partenerii din consorțiu, cât și componentele principale ale viitoarei platforme integrate în care se vor regăsi serviciile dezvoltate în proiectele componente.

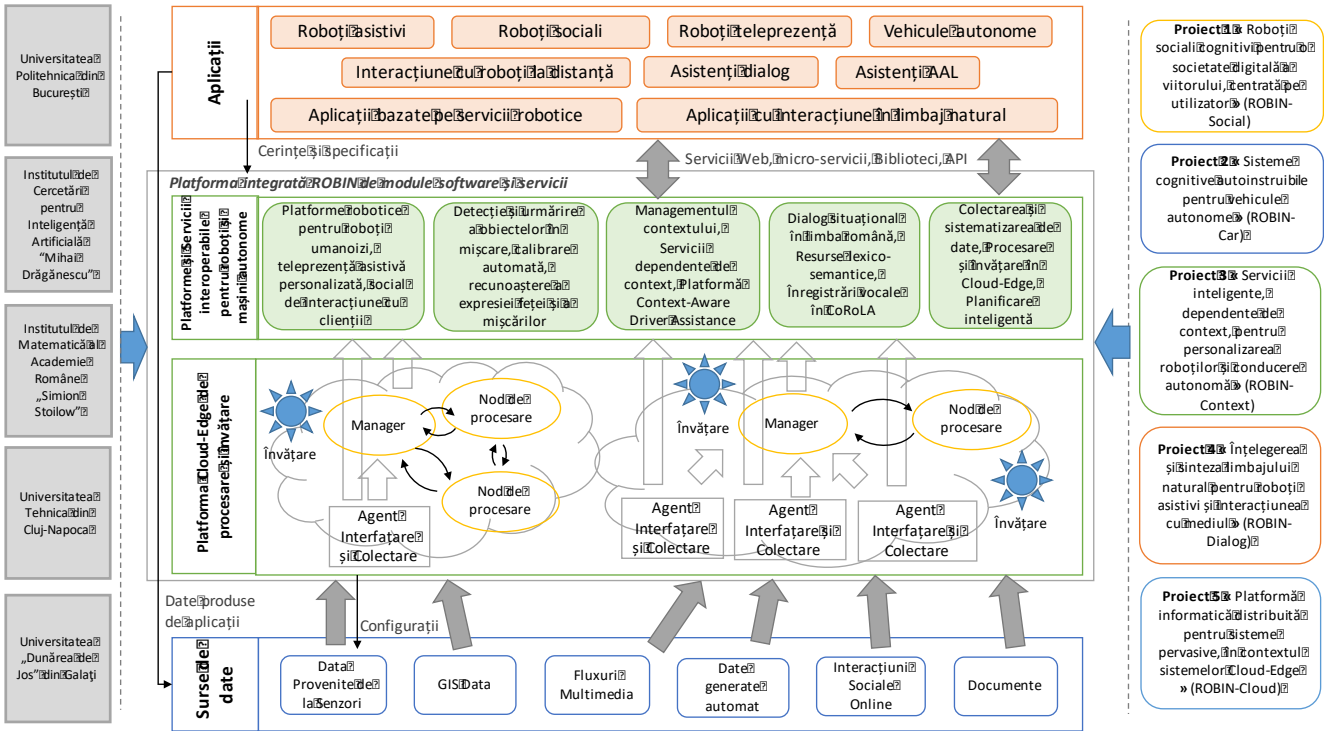


Figura 1.2. Prezentarea conceptului integrat a proiectului ROBIN

2. Proiectul component ROBIN-Social

Parteneri ai proiectului ROBIN-Social

UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI (CO)

INSTITUTUL DE CERCETĂRI PENTRU INTELIGENȚĂ ARTIFICIALĂ "MIHAI DRĂGĂNESCU"

INSTITUTUL DE MATEMATICĂ "SIMION STOILOW" AL ACADEMIEI ROMÂNE

2.1. Prezentare generală

Scopul proiectului ROBIN-Social este realizarea unei soluții integrate și ușor configurabile pentru personalizarea roboților asistivi (pt. persoane cu nevoi speciale) și sociali (pt. clienți), soluție bazată pe tehnici avansate de inteligență artificială care pot oferi roboților un caracter cognitiv și autonom. Prin utilizarea unor algoritmi robuști de învățare automată, vedere computerizată și prelucrare a limbajului natural, inclusiv cel vorbit, cât și prin combinarea algoritmilor de pe robot (de ex. SLAM, recunoaștere persoană) cu date provenite de la dispozitive IoT, dar și prin utilizarea unor soluții de tip Cloud-Robotics[1], proiectul are ca obiectiv depășirea limitărilor existente în roboții comerciali (umanoizi sau semi-umanoizi) și personalizarea acestora pentru aplicații specifice.

În acest context, ***obiectivele proiectului ROBIN-Social*** sunt:

- Realizarea unei platforme de dezvoltare a aplicațiilor robotice pentru diferite modele de roboți umanoizi sau de tip teleprezență, de ex. asistarea persoanelor în vârstă sau cu afecțiuni la domiciliu, asistența clienților, platformă care să permită integrarea dispozitivelor IoT pentru creșterea acurateței de operare și utilizarea resurselor din Cloud pentru calcule intensive.
- Dezvoltarea unor algoritmi performanți de SLAM, 3D-mapping și fuziune a datelor (data fusion) cu datele de la senzori care să permită autonomie pe termen lung într-un mediu dinamic (oameni și obiecte în mișcare).
- Dezvoltarea unor algoritmi de planificare a mișcărilor robotului, bazați pe tehnici de inteligență artificială, considerând candidați cum ar fi rețele neurale adânci sau învățarea prin recompensă.

[1] http://roboearth.ethz.ch/cloud_robotics/index.html

- Utilizarea și adaptarea algoritmilor de vedere computațională pentru detecția și recunoașterea persoanelor.
- Utilizarea și adaptarea sistemului de dialog în limba română din ROBIN-Dialog pentru interacțiunea cu roboții în limbaj natural.
- Realizarea unui produs de tip teleprezență asistivă autonomă, cu un hardware comercial.
- Realizarea unui produs de tip robot asistiv pentru persoane cu nevoi speciale, cu un robot comercial.
- Realizarea unui produs de tip robot social de interacțiune cu clienții într-un supermarket, cu un robot comercial.

2.2. Platforma AMIRO

In cadrul etapei a 2-a am continuat dezvoltarea platformei AMIRO, initiata in etapa a 1-a, si am realizat soluțiile de asistenta cognitiva si interactiune sociala cu platforma robotica. Platforma AMIRO este organizată sub forma unei arhitecturi modulare, cu componente independente conectate prin intermediul unității de planificare. Modularitatea sistemului oferă atât posibilitatea de a rula individual fiecare modul, cat și posibilitatea de a le înlocui cu ușurință. Fiecare componenta trebuie sa implementeze metode specifice corespunzătoare sarcinilor pe care trebuie sa le execute. Figura 2.1 prezintă arhitectura și conexiunea intre componentele principale ale sistemului: navigare, vedere computațională, vorbire și planificare.

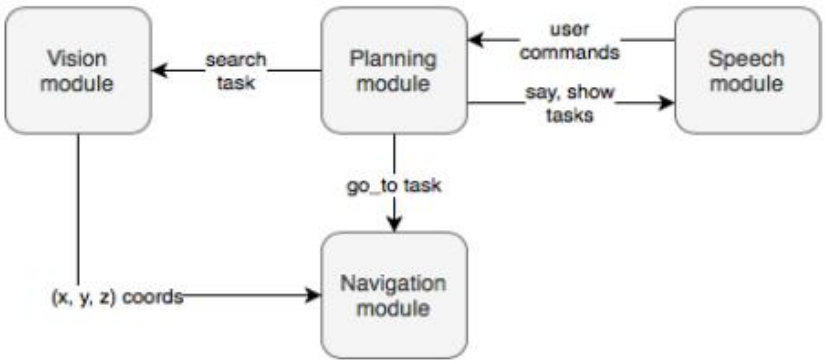


Figura 2.1. Platforma AMIRO

Modulul de navigare se ocupa sarcinile de mișcare a robotului prin mediu. Componenta de vedere computațională este responsabila cu interpretarea vizuala a mediului. Componenta de vorbire interpretează limbaj natural și reproduce text în vorbire. Modulul de planificare preia comenzi care vin de la utilizator prin mai multe interfețe grafice, și le atribuie celorlalte module. Comportamentul robotului este o combinație intre aceste sarcini de baza.

In cadrul primei etape ne-am concentrat pe dezvoltarea modulului de Navigare si pe cel de Vedere artifiială, iar in etapa a 2-a am dezvoltat în continuare modulul de Planificare și am configurat acest modul pentru interacțiune socială, și ne-am concentrat pe dezvoltarea modului de Interacțiune Vocală.

2.3. Descrierea modulului de planificare pentru interactiuni sociale

Pentru a putea avea interactiuni semnificative intre oameni si robot, a fost nevoie de implementarea unei componente de interactiune sociala. Acesta componenta integreaza in platforma AMIRO un modul de dialog si un modul de planificare. Modulul de dialog are ca functionalitati înțelegerea limbajului natural și redarea textelor predefinite, in timp de modulul de planificare este responsabil de planificarea secventelor de actiuni pe care robotul le executa.

Planificarea comportamentului robotului este realizata in cadrul modulului de planifcare pe baza informatiilor obtinute de la celelalte module cu care interactioneaza. Acesta permite robotului să aibă un comportament inteligent, astfel incat acțiunile pe care acesta le execută să fie cât mai aproape de cele umane. Modulul de planificare permite robotului un control al activităților în execuție.

Comenzile pe care modulul de Planificare le primeste pot proveni din surse diferite: de la un sistem extern, de la utilizator sau din mediu. Modulul interpreteaza comenzile in acelasi mod, indiferent de sursa.

Elementul de baza al modulului de planificare este o coada de prioritati. Fiecare modul din arhitectura expune un set de funcționalități de bază, care sunt compuse intr-un arbore executat secvențial de către modulul de planificare.

Comenzile pe care sistemul le primeste sunt adaugate într-o coadă cu priorități din care se extrage mereu prima comanda. Atunci cand este primita o noua comanda iar sistemul are deja un comportament in executie, prioritatea noii comenzi este comparata cu functionalitatea care se executa, pentru ca robotul să decidă ce execută în continuare.

Acesta poate decide sa intrerupa comportamentul curent si sa execute noua comanda, urmand sa reia vechea comanda dupa ce o executa pe cea noua, sau poate decide ca functionalitatea in curs este mai prioritara decat comanda nou primita, si isi va continua comportamentul. Acest flux al modului de planificare este prezentat sub forma grafica in Figura 2.2.

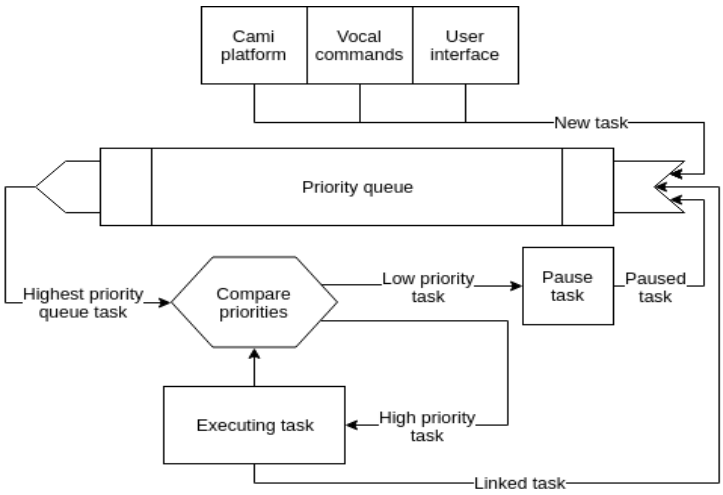


Figura 2.2. Fluxul componentei de planificare

In cadrul platformei AMIRO am implementat o serie de comenzi de baza care pot fi combinate intr-un comportament mai complex. Acest lucru se datoreaza modului in care sunt definite functionalitati primare, mai precis includ evenimentele ce vor fi rulate când aceasta se termină cu succes și când nu. Cele două evenimente pot fi legate mai departe de alte funcționalități de baza, astfel incat in urma definirii unei secvente logice sa obtinem un comportament complet.

Pentru a procesa comenzile de la utilizatori, modulul preia datele de la topicul ROS \audio pe care robotul publică fluxul audio neprocesat de la microfoane. Acesta este transmis către un serviciu extern de recunoaștere vocală care returnează textul recunoscut. Textul este apoi transmis către un modul de procesare a vorbirii pe topicul \speech_text, care returneaza entitățile cheie din frază, precum și o clasificare a acestuia într-un tip predefinit. Comenzile de baza pe care sistemul le interpreteaza si pe care le adauga in coada de prioritati sunt:

- Spune - Comanda conform careia robotul reda un text.
- MutaLa - Comanda preluata de modulul de navigare pentru a muta robotul la o anumită poziție in mediu.
- Caută - Comanda preluata de modulul de vedere computațională pentru a detecta in imagini un obiect sau o persoana.
- Arată - Comanda conform careia robotul afiseaza informatii pe un display dacă este disponibil.
- Ascultă - Comanda executata de modulul de vorbire prin care robotul asteapta ca input anumite informatii in limbaj natural. Deși modulul de vorbire este mereu activ și realizează recunoașterea vorbirii pentru interpretarea de comenzi, in cazul comenzii asculta acesta interpreteaza doar texte cu o structura specifica.
- Actioneaza - Comanda care actioneaza un senzor din mediu sau dispozitive inteligente, cum ar fi jaluzele sau lumini.
- Modul in care este implementat modulul de planificare permite extinderea facila a acestuia cu alte comenzi de baza.

2.4. Descrierea modului de dialog

In cadrul componentei de planificare, punctul principal al modulului este preluarea comenzilor vocale de la utilizatori in doua limbi diferite, respectiv limba engleza si limba romana. Astfel ca a fost necesara o implementare si evaluare a componentei de recunostere si interpretare a vorbirii. Pentru a analiza performanta modulului in cadrul platformei AMIRO, am folosit atat servicii externe, cat si o componenta proprie, dezvoltata in cadrul proiectului ROBIN-Dialog.

2.4.1. Implementare si evaluare folosind servicii externe

Comunicarea verbală este cea mai de succes formă de comunicare între oameni [1]. Aceasta, permite oamenilor să-și transmită gândurile într-un mod ușor și rapid. Prin urmare, posibilitatea de a interacționa prin comunicări verbale cu orice robot în special în cazul roboților sociali.

Pentru a permite comunicăriile verbale între utilizatori și Pepper, a fost dezvoltat un modul de voce. Acest modul este capabil să proceseze comenzi vocale multilingve și este format din cinci componente principale așa cum este ilustrat în Figura 2.3.

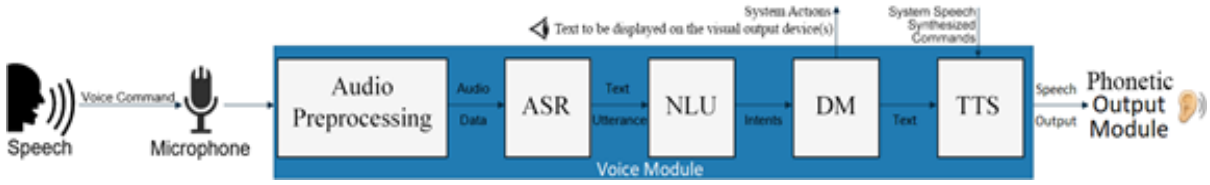


Figura 2.3. Modulul de Voce

In timpul cercetarii au fost identificate mai multe solutii relevante pentru modulul de dialog, dintre care: Alexa Voice Service, Google Cloud Speech Services, Azure Speech Services, Windows Desktop Speech Technology, Speech services of IBM Watson, Dragon NaturallySpeaking, Lexix, Cmusphinx, Julius, Jasper, ResponsiveVoice.JS, eSpeak, Wit.ai, RASA, LUIS, DialogFlow… O comparatie intre diferitele solutii identificate este prezentate in Tabelul 2.1.

Modulul de dialog are cinci componente principale: Preprocesare Audio, Recunoașterea Automată a Vorbirii (ASR), Înțelegerea Limbajului Natural (NLU), Gestionarea Dialogului (DM) și Redarea textului (TTS). Acest modul suporta limba română și limba engleza.

Preprocesare Audio: Componenta de preprocesare audio extrage date care sunt utile pentru modulul vocal din profilul utilizatorului și din setarea sistemului, de exemplu: limba care este extrasă din profilul utilizatorului, frecvența și numărul de canale audio sunt extrase din setările sistemului. Pe lângă datele anterioare, mai sunt extrase preferințele de punctuație automată, codarea audio precum și alte opțiuni.

Recunoaşterea Automată a Vorbirii (ASR): Serviciul Google Speech-to-Text este utilizat ca soluție ASR. Sistemul surprinde comanda vocală a utilizatorului și trimite fișierul audio în Google cloud, care returnează transcrierea textului în timp real. Următorul exemplu este dat pentru a ilustra modul în care modulul vocal comunică cu serviciul cloud ASR. În acest exemplu, modulul vocal trimite fișierul audio care conține comanda vocală înregistrată a utilizatorului către serviciul Google Speech-to-Text folosind un fișier JSON.

Natural Language Understanding (NLU) & Dialog Management (DM): Platforma wit.ai este utilizată ca soluție NLU. Aceasta permite sistemului să înțeleagă ce vor utilizatorii atunci când se exprimă cu propriile lor cuvinte. Platforma extrage intențiile și entității din comanda utilizatorului. De asemenea platforma wit.ai oferă un sprijin de bază pentru componentul de DM, de aceea, aceasta platformă este utilizată și ca soluție DM. După extragerea intențiile și entității din comanda utilizatorului și utilizarea acestei comenzi, platforma va transmite sistemului un fișier JSON ce contine informații despre intențiile și entității extrase dar și răspunsul sistemului la comanda utilizatorului.

Text-to-Speech (TTS) Synthesis: Serviciu Google Text-to-speech este utilizat ca soluție TTS pentru limba engleză în timp ce ResponsiveVoice.JS este utilizat ca soluție TTS pentru limba română. Textul primit de la componentul DM este trimis în cloud-ul serviciului utilizat printr-un fișier JSON, ambele servicii returnează fișierul audio care conține textul vorbit artificial în timp real. În cazul în care cererea utilizatorului nu necesită ca răspuns de la sistem o ieșire audio (ce va conține textul vorbit artificial), componentul TTS nu va fi solicitat iar dacă cererea utilizatorului necesita o acțiune care trebuie luată de sistem, DM cere sistemului de a executa comanda de exemplu la comanda utilizatorului “ridică jaluzelele”, DM va trimite ordin la sistem să ridice jaluzelele.

Soluții	Serv.	Serviciu		Limbi								Comentarii
		Gratuit	Platit	EN	FR	RO	SW	DA	PL	IT	Adi.	
IBM Watson	S-t-T		✓	✓	✓						✓	Dependent de Internet
	T-t-S									✓		
Dragon NaturallySpeaking	S-t-T		✓	✓	✓	✓	✓	✓	✓	✓	✓	Dependent de Internet
	T-t-S											
Lexix	S-t-T		✓	✓								
	T-t-S											
Windows Desktop Speech Technology	S-t-T	✓		✓	✓					✓	✓	Dependent de Microsoft Windows
	T-t-S					✓	✓	✓	✓			
Azure Speech Services	S-t-T		✓*	✓	✓		✓	✓	✓	✓	✓	Dependent de Internet
	T-t-S					✓						
Google Cloud Speech Services	S-t-T		✓*	✓	✓	✓	✓	✓	✓	✓	✓	Dependent de Internet
	T-t-S											
Alexa Voice Service	Toate Serv.	✓		✓							✓	Dependent de Internet si dispozitiv
Julius	S-t-T	✓		✓°							✓°	Orice limbă poate fi suportată
Cmusphinx	S-t-T	✓		✓°	✓°						✓°	Orice limbă poate fi suportată
HTK	S-t-T	✓									✓	Orice limbă poate fi suportată
	T-t-S											
Kaldi	S-t-T			✓°							✓	Orice limbă poate fi suportată
Jasper	S-t-T			✓°							✓°	Dependent de board RPi Orice limbă poate fi suportată
Apple Speech Framework	S-t-T	✓		✓	✓		✓	✓		✓	✓	Dependent de iOS & internet
Ispeech	S-t-T		✓	✓	✓		✓	✓	✓	✓	✓	Dependent de Internet
	T-t-S											
eSpeak	T-t-S	✓		✓°	✓°	✓°	✓°	✓°	✓°	✓°	✓°	
ResponsiveVoice.JS	T-t-S	✓‡	✓‡	✓	✓	✓	✓	✓	✓	✓	✓	Dependent de Internet
LUIS	NLP		✓*	✓	✓					✓	✓	Dependent de Internet
DialogFlow	NLP	✓‡	✓‡	✓	✓		✓	✓		✓	✓	Dependent de Internet
RASA NLU	NLP	✓										Orice limbă poate fi suportată
Wit.ai	NLP	✓		✓	✓	✓	✓	✓	✓	✓	✓	Dependent de Internet
Azure Bot Service	DM		✓*	N/A								Dependent de Internet
Bot Builder SDK	DM	✓		N/A								Dependent de Microsoft Windows

*Tabel 2.1. Compararea soluțiilor identificate. *: perioadă de testare gratuită și anumite tranzacții lunare gratuite, ‡: unele funcții sunt gratuite, altele necesită plata, °: limba predefinita,S-t-T: serviciul Speech-to-Text și T-t-S: serviciul Text-to-Speech*

Evaluare

Modulul de voce a fost evaluat de un grup de 14 persoane în laborator. Toți utilizatorii au testat modulul de voce folosind limbile română și engleză. Pentru limba română, fiecare utilizator a testat 220 de comenzi vocale cu Robotul Pepper, în timp ce pentru limba engleză a testat 190 de comenzi. Prin urmare, testele au fost efectuate pe un set de 5740 de interacțiuni (3080 în română și 2660 în engleză).

Procedura de evaluare a folosit distanțele Levenshtein pentru a calcula diferențele dintre rezultatele recunoașterii comunicărilor verbale și fraza textului inițial. După calcularea distanțelor Levenshtein pentru diferitele interacțiuni dintre Pepper și utilizatori, Formula (2.1) este utilizată pentru a calcula procentul mediu pentru recunoașterea fiecărei interacțiuni.

$$\text{Procentaj mediu de recunoaștere} = \left(\sum_{i=1}^{r}(1 - l / n) \times 100\right) / r \qquad (2.1)$$

unde i este numărul utilizatorului, r este numărul total de utilizatori, l este distanța Levenshtein a interacțiunii utilizator mașină curentă și n este lungimea celei mai lungi fraze dintre recunoașterea curentă și fraza text originală. O parte din rezultatele testelor sunt ilustrate în Tabelul 2.2 pentru limba română și în Tabelul 2.3 pentru limba engleză.

Comanda utilizatorilor	ASR Media Recunoașterii	Intenție extrasă	Entitate extrasă / Entități extrase	Ieșire
Du-te la Ștefania	65.25%	find_person	person: Ștefania	Pepper căuta pe Ștefania.
Spune-mi mai multe despre Alex	68.50%	talk_about	Subject: Alex	Pepper vorbește despre Alex.
Cum va fi vremea mâine in Paris	74.25%	get_weather	location: Paris datetime: mâine	Pepper vorbește cum va fi vremea maine: *Maine, vremea va fi rece și ploioasă în Paris.*
găsește un scaun	54.25%	find_object	object:scaun	Pepper căuta un scaun.
Afișează-mi tensiunea arterială	74.50%	display_health_ param	health_param: tensiunea arterială	Pepper afișează un grafic pentru valorile tensiunii arteriale în ultimele 7 zile.

Tabel 2.2. O parte din rezultatele testelor pentru limba română

Comanda utilizatorilor	ASR Media Recunoașterii	Intenție extrasă	Entitate extrasă / Entități extrase	Ieșire
Go to Stefania	59.25%	find_person	person: Stefania	Pepper searches for Stefania.
Tell me more about Alex	82.75%	talk_about	Subject: Alex	Pepper talks about Alex.
How will be the weather tomorrow in Paris	84.50%	get_weather	location: Paris datetime: tomorrow	Pepper tells how the weather will be tomorrow: *Tomorrow, the weather will be cold and rainy in Paris.*
Find me a chair	61.25%	find_object	object:chair	Pepper searches for a chair.
Display my blood pressure	85.50%	display_health _param	health_param: blood pressure	Pepper displays the blood pressure chart for the measurements obtained during the last 7 days.

Tabel 2.3. O parte din rezultatele testelor pentru limba engleză

Referitor la componentul TTS, utilizatori au umplut la sfârșitul fiecărui sesiuni un chestionar despre gradul de claritate și ușurința de înțelegere a vocei generate. Rezultatele obținute sunt satisfăcătoare.

2.4.2. Dezvoltarea unei componente proprii pentru interacțiune vocală în limba română

Au fost create mai multe resurselingvistice pentru limba română care permit implementarea sistemelor de dialog om-robot. Astfel, corpusul MONERO[2] a fost creat și pus la dispoziția comunității științifice. Acesta este primul corpus ”gold standard” biomedical in limba română adnotat la nivel morfologic cu un set de 714 etichete și cu patru clase de entitățăți denumite (ANATomy, CHEMical substances, DISOrders și PROCedures). MONERO este o component a unui corpus mai mare (BioRo) dezvoltat în teza de doctorat a dnei Maria Mitrofan. Spre deosebire de corpusul mare, MONERO a fost foarte atent validat de doi adnotatori experți. Evaluările gradului de concordanță între evaluatori (s-a folosit coeficientul Cohen-Kappa pe un eșantion 1% din corpus) de 92.8% arată faptul că cei doi adnotatori au identificat aproximativ aceleași entitățăți denumite. În urma procesului de adnotare a rezultat un corpus de 154,825 de unitățăți de analiză distribuite în 4.989 de fraze adnotate cu entitățăți denumite, toate acestea avand MSD-ul (eticheta morfo-lexicală) corectat. Lungimea medie a unei fraze este destul de mare și anume 31.04 de unitățăți de analiză. Au fost identificate și

[2] http://www.racai.ro/en/tools/text/

correct etichetate 14.133 de termeni medicali (ANAT-1964, CHEM-4156, DISO-6611, PROC-1402). Forma curentă a acestui corpus și modul în care s-a atins calitatea deosebită a sa este descrisă în lucrarea [2]. Corpusul MONERO a fost folosit pentru antrenarea unui system de identificare și recunoaștere a entităților numite dar și pentru extragerea unui lexicon medical.

Pentru interacțiunea vocală în limba română a fost creat, pe baza scenariilor din proiectul ROBIN-Dialog, un lexicon amplu, cu 32627 intrări bogate informațional. Structura unei intrări din acest lexicon (ce urmează a fi îmbogățit și cu termenii medicali extrași din corpusul MONERO) este următoarea:

<formă>tab<lemă>tab<etichetă-MSD>tab<silabație>tab<accent>tab<transcriere_fonetică>

<formă> este o formă ocurență a lemei <lemă>. Câmpul <etichetă-MSD> este completat cu o etichetă morfo-sintactică din specificație morfo-lexicală a limbii române, conformă cu standardul MULTEXT-EAST. Informația de silabificare este furnizată în câmpul <silabație> limita unei silabe fiind indicate cu ajutorul unui punct (.). Marcarea accentului se specifică în câmpul <accent> prin folosirea apostrofului (') în fața vocalei accentuate. Ultimul câmp al unei intări conține transcrierea fonetică a formei ocurență (convenția SAMPA). Exemplu:

zicând zice Vmg zi.când zic'ând [z i k 1 n d]

Interfața de dialog în limba română este dezvoltată în proiectul component ROBIN-Dialog și are structura prezentată in Figura 2.4:

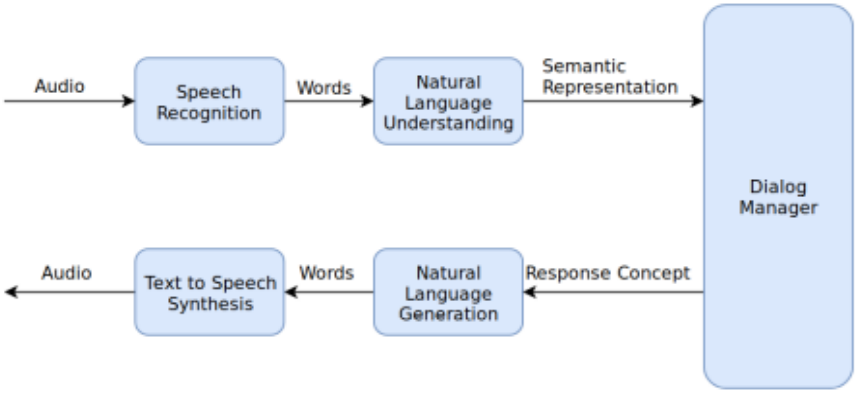


Figura 2.4. Structura interfetei de dialog

Experimente pentru recunoașterea vorbirii în limba română s-au desfășurat atât în colectivul de la ICIA cât și în cel de la UPB iar sinteza din text a vorbirii în limba română s-a bazat pe rezultatele obținute de ICIA în anul precedent (v. raportul pe 2019 pentru proiectul component. În cadrul acestui proiect component au fost create mai multe resurse lingvistice pentru limba română care permit implementarea sistemelor de dialog om-robot. Astfel, corpusul MONERO a fost creat și ROBIN-Dialog). Aceste module folosesc resursele prezentate anterior. Managerul de Dialog este descris pe larg în raportul pe 2019 pentru proiectul component ROBIN-Dialog și a fost deja făcut public in Gitlab-ul:

https://gitlab.com/raduion/robindialog.git https://docs.google.com/document/d/ și mai multe amanunte au fost distribuite pe pagina 1ZQV0N6mlUkOUU0fgoUEAK6_92qXj_PICm-ZUu1MZxgc/edit.

2.4.3. Recunoașterea intențiilor vocale

S-au elaborat componentele de Intent Detection and Slot Filling și Keyword spotting. În cadrul componentei de Intent Detection and Slot Filling - home assistant pentru vorbitorii de limba romana, am definit o metodologie de generare sistematica de date adnotate; am aplicat metodologia pentru 3 tipuri de intent-uri (de lumina, temperatura, media), toate comenzi - in total 13 intent-uri diferite. Analizand problema, am identificat o serie de provocari ce ar putea aparea in invatare si am definit diferite dimensiuni de complexitati (datele pentru invatare si testare)generandu-le pentru un sistem de baza - fara complexitati, si adaugand incremental cate o complexitate, dupa cum urmeaza:
C1 - sinonimia (acelasi inteles poate fi exprimat in mai multe moduri)
C2 - valori noi/lipsa la parametri
C3 - dezechilibrul claselor.
Bazandu-ne pe aceste complexitati, am generat seturi de date specifice, pentru diferitele scenarii de dificultati considerate.

Am evaluat performanta a 2 tool-uri pe fiecare din seturile de date generate - wit.ai si RASA NLU (lb Eng + Ro), precum si 2 solutii propri (identificate, proiectate, implementate, testate si evaluate) bazate pe ideea de Capsule Neural Networks (CapsNetS2I si CapsNetI2S). Am descoperit o serie de dificultati la evaluare (facand o analiza calitativa a erorilor ce apar la invatare). Performantele pentru solutia originala sunt satisfacatoare (de la aproape 100% pentru cazul de baza - fara complexitate, diferite performante, la complexitate maxima ~43%. În prezent se lucrează la o imbunatatire, dar rezultatele sunt încurajatoare deoarece sunt 13 clase, si 3 tipuri de complexitati - sinonimie, zgomot si dezechilibru in clase)

În componenta de Keyword spotting am construit un pipeline un pipeline pe care momentan l-am evaluat doar pe limba engleza (in absenta datelor pentru limba romana). Solutia transforma semnalul vocal in imagini, in mai multi pasi: se extrag trasaturi MFCC din semnalul vocal, se masoara matrici de similaritati intre semnacul curent si sabloanele pentru cuvinte cheie (folosind Dynamic Time Warping), matricile fiind apoi transformate in imagini grayscale, ce sunt apoi segmentate si transmise unui clasificator convolutional-recurent. Metoda a fost evaluata pe setul de date TIMIT.

2.5. Cercetări pentru recunoașterea acțiunilor

Pentru implementarea soluției de asistență de urgență a utilizatorilor cu nevoi speciale, am continuat dezvoltarea componentei de recunoaștere a acțiunilor care are ca scop atât detectarea situațiilor de urgență, cum ar fi de exemplu căderea, cât și a altor acțiuni care au pot avea ca scop o intervenție a celui care are grijă de persoana în vârstă, fie o asisetnță de urgență fie de alt tip.

Pentru recunoașterea acțiunilor, am plecat de la conceptul de reprezentare pe bază de schelet prezentat de Johanson in 1973, care a demonstrat că un număr mic de poziții comune pot reprezenta în mod eficient comportamente umane.

Am definit mai multe variante de arhitetură de rețea neuronală convoluțională pentru recunoașterea acțiunilor. Am utilizat o repezentare pe bază de scheleți a corpului uman. Spre deosebire de prima etapă în care scheleții erau predefiniți și obținuți printr-o cameră Kinect, am implementat propria variantă de obținere a scheleților. Arhitectura folosită este formată din 2 rețele FC Densely Connected Convolutional Networks [3] pentru recunoașterea poziției, urmate de mai multe rețele LSTM pentru prelucrarea dimensiunii temporale a acțiunii. Jointurile sscheletului sunt aranjate într-un layout 2D ca o matrice de 5 x 5 x3, care este apoi transmisă către straturi convoluționale 3D asa cum este ilustrat in Figura 2.5.

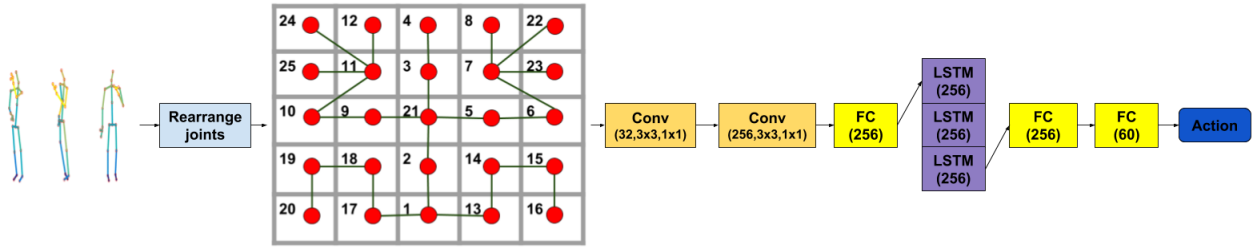


Figura 2.5. Arhitectura rețelei

Pentru antrenare și experimente, am utilizat setul de date NTU RGB + D[3]. Setul de date NTU RGB + D conține în total 60 de clase de acțiuni, care sunt împărțite în trei grupe majore: 40 acțiuni zilnice (băut, mănâncă, citit etc.), 9 acțiuni de sănătate (strănut, împiedicare, cădere etc.) și 11 acțiuni reciproce (punching, kicking, îmbrățișare, etc.). În prezent, acesta este cel mai mare set de date de recunoaștere a acțiunii bazate pe adâncime, furnizând coordonate 3D a 25 articulații colectate de Kinect v2. Acest set de date conține peste 56 000 de secvențe și 4 milioane de cadre, capturate în diferite condiții de fond.

În plus, în această etapă am început dezvoltarea propriului set de date în laborator, care este axat în special pe detectarea acțiunilor utilizatorilor cu nevoi speciale și a situațiilor de urgență.

2.6. Continuarea cercetărilor

Planurile de viitor includ realizarea aplicatiilor prototip, extinderea modulului de voce și integrarea unui modul de recunoaștere de emoții a utilizatorilor ceea ce va face interacțiunea dintre utilizatori și robotul Pepper mai naturală și mai interesantă. În plus, sunt planificate testări intensive pentru softare dezvoltat.

[3] https://github.com/shahroudy/NTURGB-D

3. Proiectul component ROBIN-Car

Parteneri ai proiectului ROBIN-Car

INSTITUTUL DE MATEMATICĂ "SIMION STOILOW" AL ACADEMIEI ROMANE (CO)

UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI

INSTITUTUL DE CERCETĂRI PENTRU INTELIGENȚĂ ARTIFICIALĂ "MIHAI DRĂGĂNESCU"

UNIVERSITATEA "DUNAREA DE JOS" DIN GALAȚI

3.1. Prezentare generală

Scopul proiectului ROBIN-Car constă în dezvoltarea de metode de vedere computaţională care să rezolve o gamă mai largă şi mai sofisticată de sarcini de asistenţă în pilotaj, realizarea unor module inteligente pentru „Hands-off driving" și „Automated driving" și un sistem prototip care va fi testat pe un autovehicul electric semi-autonom pus la dispoziția consorțiului de compania PRIME Motors Industry, pe durata derulării proiectului. Sistemul va fi capabil să observe, recunoasca si monitorizeze scena, drumul, obiectele si persoanele din mediul exterior precum si expresia șoferului, oferindu-i informatiile necesare intr-un mod cat mai non-invaziv (inclusiv interacține vocală prin comenzi simple), capacitate crescuta de pilotaj si de luarea deciziilor.

Obiectivele proiectului ROBIN-Car sunt:

- Realizarea unui modul de înţelegere semantică a obiectelor din mediul înconjurător autovehiculului, pentru detecția și urmărirea obiectelor aflate în mișcare (alte mașini din trafic, pietoni), recunoașterea obiectelor fixe (obstacole, benzi, semne de circulaţie) prin fuziunea datelor de la camere 2D dar și 2D și 3D laser, bazat pe algoritmi noi de computer vision.
- Realizarea unui modul de gestionare eficientă a geometriei scenei 3D, în vederea estimării 3D a zonelor cu gropi, denivelari, etc.
- Realizarea unui modul de recunoaștere a expresiei feței șoferului și a direcţiei de privire, pentru atenționare și asistare în timpul condusului, in vederea identificării gradului de oboseală sau a direcţiei privirii.
- Realizarea unui modul de atenţionare a conducătorului autovehiculului pe baza unor servicii sensibile la context oferite de subproiectul P3 – ROBIN-Context.
- Preluarea de comenzi vocale în limbaj natural și translatarea informațiilor vizuale in limbaj natural (în limba română) pentru realizarea interacțiunii între conducătorul auto și mașină.
- Accelerarea algoritmilor dezvoltaţi bazaţi pe reţele convoluţionale, pe sisteme reconfigurabile (FPGA-uri).

Realizarea unui sistem prototip care să integreze facilitățile descrise anterior, instalarea acestuia pe un vehicul electric semi-autonom și testare intensivă.

3.2. Setul de date UPB si analiza comparativa a algoritmilor de detectie a pietonilor

Pentru implementarea unui modul bun de detectie si urmarire a obiectelor din mediul inconjurator a fost nevoie de colectarea unui set de date care sa prezinte o provocare prin faptul ca este reprezentativ pentru o zona rezidentiala, unde pot fi intalniti multi pietoni si multe autovehicule parcate care impart zona de drum navigabil cu masina pentru care se dezvolta sistemul ADAS inteligent din ROBIN-Car.

Ca atare, folosind un setup constand intr-o singura camera, in perioada 20-30 iulie a fost strans un set suplimentar de date. Echipa de proiect a fost apoi activa in adnotarea semantica a setului de date, de data aceasta distingand intre trei clase conceptuale: drum navigabil, pieton, masina. Un exemplu de adnotare este prezentat in Figura 3.1, unde sunt evidentiate cele trei clase mentionate.

Acest set de date ajuta echipa de proiect in doua aspecte. In primul rand, in urma segmentarii mai complexe a drumului navigabil, setul de date poate fi utilizat pentru a testa in mod mai robust performantele algoritmilor de segmentare semantica (de a distinge drumul navigabil, pietonii si alte autovehicule).

In al doilea rand, setul poate fi folosit pentru a studia in mod comparativ algoritmi bazati pe retele neurale care fac detectia pietonilor.

Figura 3.1. Exemplu din utilitarul CVAT de adnotare semantica a celor 3 clase urmarite a fi clasificate in noua iteratie a setului de date UPB: sosea navigabila (maro deschis), pietoni (rosu), masini (verde)

3.2.1. Analiza algoritmilor de detectie a pietonilor

Pentru a obtine un termen de referinta in privinta eficacitatii algoritmilor de detectie de pietoni, s-a efectuat o evaluare out-of-the-box a unor algoritmi bazati pe retele neurale, care sunt antrenati pe sarcini generice de detectie de obiecte (printre care si persoane, autovehicule, biciclisti sau unele categorii de semne de circulatie).

Pentru aceasta evaluare s-au ales ~13000 cadre care contin obiecte de interes: ~60000 obiecte dintre care ~41000 masini si ~14500 pietoni. De mentionat este faptul ca multe dintre aceste obiecte sunt aceleasi, surprinse insa la momente diferite de timp (si deci avand diferite dimensiuni si orientari in camera).

Pentru a defini metricile de referinta s-au calculat precision si recall, precum si eroarea medie de predictie (Mean Average Precision). In domeniul de conducere autonoma, metrica de recall (cati pietoni din totalul celor existenti in imagine sunt detectati de algoritm) este la fel de importanta cu cea de precizie (cati dintre pietonii detectati sunt intr-adevar pietoni si sunt detectati in locul corespunzator din imagine).

Un True Positive se numara atunci cand pietonul este identificat prin clasa corespunzatoare (cea de pieton) si cand bounding box-ul care defineste pozitia lui in imagine are o suprapunere cu pozitia reala (adnotata manual) de peste 50% (metrica de Intersection over Union - IoU).

Au fost evaluati 4 algoritmi des intalniti pentru detectie de obiecte: YOLOv3 [4], SSD [5], RetinaNet [6] si Faster-RCNN [7].

Tabelul 3.1 prezinta rezultatele evaluarii algoritmilor out-of-the-box, care serveste drept referinta. Rezultatele complete pot fi vizualizate in [8].

	AP@0.50IOU	MAP	AR@0.50IOU	MAR
Yolo	0.69	0.55	0.59	0.47
SSD	0.79	0.65	0.12	0.10
Faster R-CNN	0.68	0.54	0.16	0.13
RetinaNet	0.17	0.26	0.06	0.09
Yolo (car and person)	0.71	0.57	0.62	0.50
SSD (car and person)	0.90	0.74	0.13	0.10
Faster R-CNN (car and person)	0.80	0.63	0.17	0.13
RetinaNet (car and person)	0.20	0.30	0.06	0.10

Tabel 3.1 Rezultate ale algoritmilor de detectie out-of-the-box pentru setul de date UPB. Valorile mai mari sunt mai bune

Dezvoltarile urmatoare trebuie sa fie capabile sa intreaca aceasta referinta. Din algoritmii analizati, luand in considerare si criteriile enuntate, modelul YOLOv3 este cel care asigura un compromis bun intre precizie medie si recall mediu si va fie cel folosit drept referinta. Motivul este ca precizia poate fi imbunatita mult mai usor, plecand chiar de la acesti algoritmi existenti, prin proceduri de fine-tuning.

3.2.2. Augmentare setului de date cu instante sintetice de pietoni

Setul de date UPB a fost colectat in doua iteratii majore: toamna 2018 si vara 2019. In sesiunea setului de date din toamna 2018, numarul total de pietoni prezenti pe strazile campusului a fost scazut, fiindca testele s-au efectuat dupa-amiaza pentru a surprinde condus atat pe timp de zi, cat si pe timp de seara (pentru a varia gradul de luminozitate).

Ca atare, robustetea reala a solutiilor de segmentare semantica si detectie de pietoni este insuficient de bine evaluata. O solutie pentru aceasta problema o constituie ideea de a introduce in mod fictiv instante de pietoni pe video-urile deja colectate de catre masina, astfel incat algoritmii de segmentare si detectie sa fie evaluati pe cazuri *mult mai variate.*

Problema principala se reduce la a reusi *insertia* pietonilor in ipostaze cat mai naturale, astfel incat alipirea intre imaginea unui pieton fictiv si fundalul existent sa se faca in mod imperceptibil, fara artefacte de *lipire* notabile, pe care un algoritm de detectie ar invata sa le exploateze in mod eronat (si pe care, in mod evident, nu le-ar regasi pe cazuri reale).

In plus fata de modul de alipire, *locul* alipirii trebuie sa poata fi identificat in mod corect, astfel incat sa nu apara cazuri in care pietonii sunt introdusi pe zone de fundal din imaginea originala, reprezentand cerul.

Metoda aleasa pentru implementarea insertiei de pietoni se bazeaza pe *retele generative adversariale* (eng. Generative Adversarial Networks - GAN).

In problema noastra, ele sunt folosite pentru a genera diferite *alipiri* ale unor sabloane de pietoni pe o zona anume aleasa dintr-o imagine surprinsa de camerele masinii autonome in campusul UPB. Factorul de variatiune se da prin diferitele texturi si culori, pe care le poate lua forma pietonului, astfel incat insertia sa in imagine sa fie cat mai naturala.

Pentru a realiza procedura propusa sunt necesare doua etape: (i) identificarea zonelor dintr-o imagine de trafic existenta in care sa fie introdus un pieton sau un grup de pietoni; (ii) *alipirea* in sine a unuia sau mai multor sabloane de pietoni in zona aleasa.

3.2.3. Identificarea zonei de insertie

Pentru alegerea zonelor posibile pentru insertie se face uz de segmentarea semantica a drumului navigabil. Procesul este descris in Figura 3.2.

Zona este identificata prin intermediul unui cadru de delimitare (eng. bounding box), definit prin coordonata coltului stanga-sus, precum si lungimea si latimea acestuia in pixeli. Luand in calcul parametrii intrinseci ai camerei cu care s-au facut inregistrarile, se poate face o estimare a distantei fata de bordul masinii a unui pixel din imagine. Echivalarea distantei din spatiul pixelilor cu distanta in metri permite determinarea dimensiunii scalate a unei siluete tipice de pieton. Raportul este apoi transformat din nou in spatiu pixelilor, astfel incat sa calculeze dimensiunea cadrului de delimitare. O data calculata dimensiunea, punctul de ancorare se determina prin analiza segmentarii semantice a drumului navigabil, astfel incat chenarul de incadrare sa interesecteze atat drum navigabil, cat si marginea drumului (pentru a plasa pietonii spre marginea carosabilului).

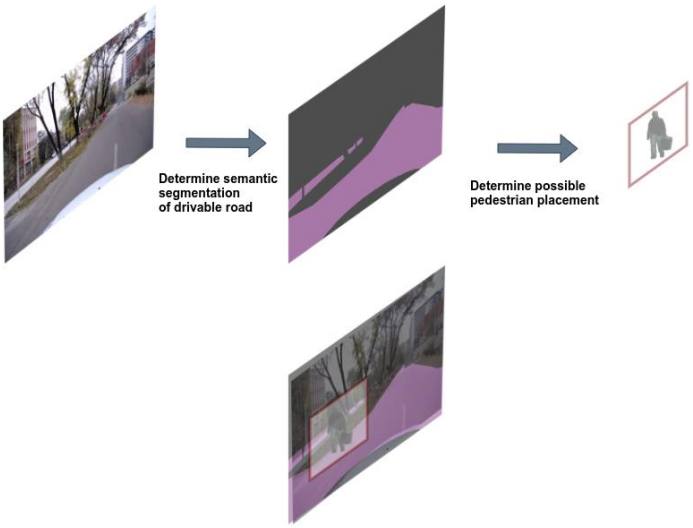


Figura 3.2. Descriere a procesului prin care se determina zona din imagine unde trebuie inserat pietonul / grupul de pietoni. Segmentare semantica a drumului ajuta la identificarea marginilor drumului. Impreuna cu estimarile de adancime in spatiul pixelilor (obtinute din proprietatile intrinseci ale camerei) se poate obtine coordonata si dimensiunea unui cadru de delimitare (eng. bounding box) care sa marcheze zona din imagine in care va fi alipita o silueta a pietonului

3.2.4. Reteaua generativa pentru insertia de pietoni

Dupa identificarea zonei de insertiei, se alege in mod aleator o silueta de pieton/pietoni care sa "umple" cat mai bine chenarul de delimitare calculat la pasul anterior. Siluetele sunt alese din segmentarile semantice ale pietonilor, existente atat in setul de date UPB, cat si in alte seturi mari de date.

Arhitectura retelei generative dezvoltate pentru aceasta sarcina, cat si metoda de antrenare a acesteia sunt bazate pe modelul Pix2Pix [9].

Figura 3.3 prezinta principiul de antrenare. La intrare se pleaca cu zona de imagine, extrasa prin procesul explicat in Sectiunea 4.1, in care se doreste includerea pietonului. In acelasi timp, tot la intrare se introduce forma pietonului (data sub forma de segmentare semantica) care se doreste *sintetizat* in imagine. *Generatorul* este reteaua care primeste cele doua conditionari (fundalul si silueta pietonului) si creeaza o imagine cu pietonul adaugat.

Imaginea rezultata, impreuna cu silueta de pieton care a conditionat generarea, sunt trimise apoi catre reteaua de tip *Discriminator*, al carei rol este de a distinge intre generari false si extrase (eng. crop-uri) reale de pietoni.

Astfel, modul in care se antreneaza o retea generativa este similar unui joc adversarial intre doi competitori. Discriminatorul trebuie sa invete sa distinga intre imagini fictive (cu pietoni sintetizati) si unele reale (cu regiuni extrase din imagini cu pietoni adevarati).

La randul sau, generatorul trebuie sa invete sa "pacaleasca" discriminatorul, prin generarea de imagini cu pietoni pe care cel din urma sa nu le poata distinge de unele reale.

Rezultatele acestei proceduri sunt inca in lucru si constituie un aspect de cercetari viitoare.

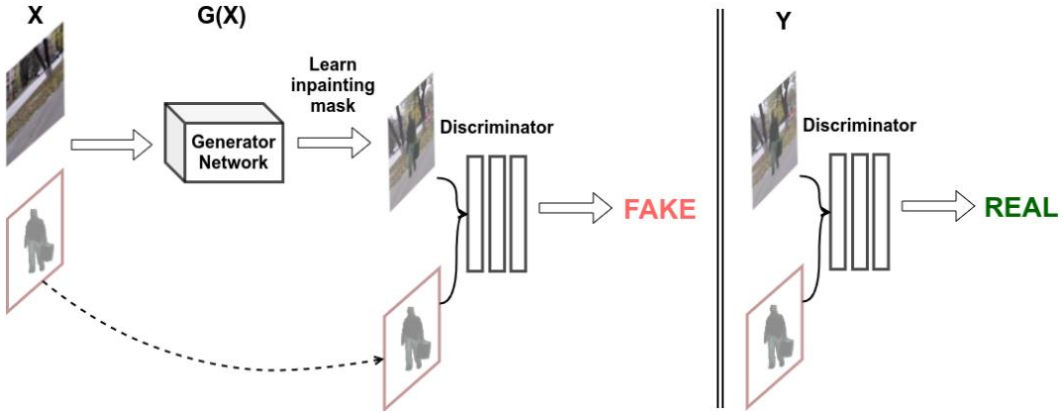


Figura 3.3. Descriere a procesului de antrenare pentru reteaua de generare a pietonilor sintetici.

3.2.5. Continuarea cercetărilor

In etapa imediat urmatoare, eforturile de cercetare si dezvoltare se vor concentra pe urmatoarele:

- Definitivarea procedurii de insertie a pietonilor sintetici, care poate actiona ca metoda generalizabila de augmentare a seturilor de date de conducere autonoma.
- Implementarea algoritmiilor particularizati de detectie a pietonilor, prin fine-tuning al modelelor out-of-the-box si utilizarea augmentarii datelor mentionata in punctul anterior.
- Studiul si implementarea algoritmilor de predictie a intentiei pietonilor aflati la marginea drumului.

3.3. Invatand navigare prin localizare vizuala și prezicerea traiectoriei

Cand conduc, oamenii iau decizii pe baza traficului curent, precum și a traseului dorit. Au o hartă mentală a rutelor cunoscute și sunt adesea capabili să navigheze fără a avea nevoie de indicații GPS. Modelele actuale de conducere automată își îmbunătățesc performanțele folosind informații GPS suplimentare. În acest proiect ne propunem să extindem cercetarea existenta și să realizăm planificarea rutelor chiar și în absența GPS-ului.

Sistemul nostru învață să prezică în timp real locația actuală a vehiculului și traiectoria viitoare în funcție de timp, pe o hartă cunoscută, folosind numai date video și destinația dorită. Semnalul GPS este

disponibil numai în timpul antrenamentului pentru adnotarea datelor în mod automat. Spre deosebire de alte metode publicate, noi prezicem traiectoria vehiculului cu până la șapte secunde in avans, din care se pot obține informații complete despre direcție și viteză pe întreaga durată de timp. Traiectoriile captează informații de navigație pe mai multe niveluri, de la comenzi instantanee care depind de traficul actual și obstacolele din față, până la decizii de navigare pe termen mai lung, spre o destinație specifică.

Colectăm setul de date cu o mașină obișnuită și un smartphone care înregistrează date video și GPS. Sistemul nostru depășește metode publicate pentru localizare vizuală și conducere automată, oferind asistență de navigare precisă între oricare două locații cunoscute. Această lucrare [10] a fost trimisă recent la conferința International Conference on Robotics Automation (ICRA) 2020, una dintre cele două conferințe de top în robotică din lume. În Figura 3.4 prezentăm o imagine de ansamblu asupra sistemului nostru propus.

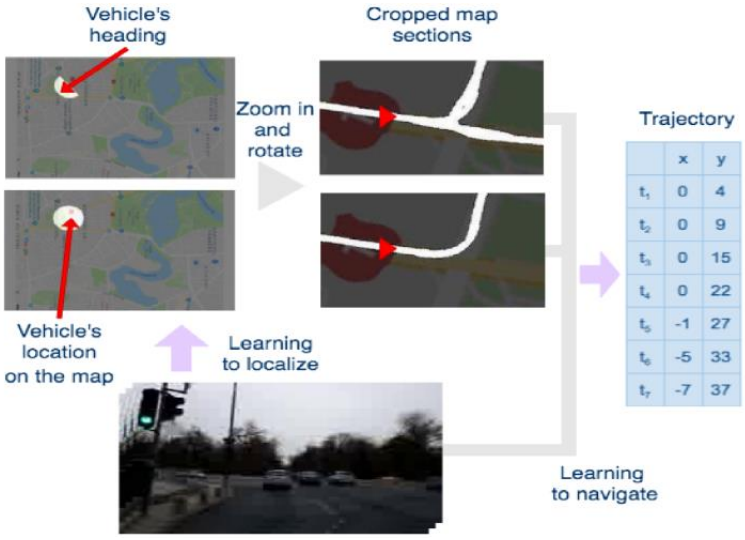


Figura 3.4. Structura la nivel înalt a sistemului. Prima rețea învață să prezică locația pe hartă prin segmentare. Secțiunile de hartă rutieră sunt decupate în jurul locației, unul afișând toate direcțiile, iar celălalt doar ruta prevăzută. Din secțiunile de hartă rutieră împreună cu imaginile de intrare, a doua rețea prezice traiectoria de navigație.

Pe scurt, contribuțiile noastre sunt următoarele:

1) Propunem primul sistem bazat pe deep-learning, care învață simultan să se auto localizeze și să navigheze spre o destinație planificată numai din informație video.

2) Sistemul poate fi scalat la costuri minime și poate fi ușor implementat pentru a învăța într-un întreg oraș prin utilizarea lui de către mai mulți șoferi. De asemenea, introducem setul de date Urban Eastern European Driving Dataset (UEEDD), pe care îl vom face public.

3) Alte contribuții includ: 1) extindem și îmbunătățim un model anterior de localizare din imagini și îl adaptăm pentru a învăța să localizeze cu precizie în trafic. 2) modelăm traiectorii, ca funcții de timp ale spațiului, care cuprind informații de direcție și viteză pentru șapte secunde în viitor. 3) harta rutieră este creată analitic și în mod automat din datele GPS colectate.

3.3.1. Localizare vizuala prin segmentare

Ideea de a învăța locația prin segmentare a fost introdusă în [11] pentru imaginile din satelit cu geolocatie asociată. Modelul lor în două etape prezice masca unui cerc pe o hartă de ieșire (reprezentând harta geografică), astfel încât centrul (x, y) al punctului reprezintă coordonatele locației imaginii raportat la harta geografică. Prin tratarea localizării ca problemă de segmentare se exploatează relația dintre „ce” și „unde”, adică între relațiile semantice și geometrice din spațiul de ieșire. De asemenea, în cazul distribuțiilor complexe de ieșire, o rețea de segmentare poate produce mai multe locații posibile, în timp ce regresia este obligată direct să producă un singur răspuns.

Abordarea de segmentare a localizării a fost cu mult superioară celor publicate bazate pe regresie. Am îmbunătățit și mai mult generalizarea și acuratețea localizării, printr-o nouă tehnică inteligentă de augmentare a datelor utilizată la antrenare. Aplicăm propagarea inversa ghidată pentru a găsi ce regiuni din imagine sunt cele mai relevante pentru localizare, apoi le mascam pentru a face rețeaua de localizare mai robustă.

3.3.2. Invatand navigare prin prezicerea de traiectorii

Etapa finală a metodei noastre folosește harta analitică și rețeaua de localizare vizuală, pentru a învăța cum să navigheze în trafic între destinatii văzute anterior. Prin prezicerea traiectorilor modelul este mai capabil să învețe sa urmeze un traseu. Acesta produce semnificativ mai puține oscilații de înaltă frecvență și erori locale decât modelele care învață comenzi instantanee. Arhitectura rețelei de navigație este o adaptare a modelului descris în [12]. Modelul produce 7 perechi de valori reale (x, y) corespunzătoare coordonatelor punctelor care definesc traiectoria, una pe secundă.

3.3.3. Evaluare a sistemului de conducere automată

În Tabelul 3.2, prezentăm comparații cantitative cu mai mulți algoritmi state-of-the-art [11, 13 si 14] pentru predicția de poziție (localizare), care au fost antrenați și testați pe setul nostru de date. Abordarea noastră depășește metodele concurente cu o marjă semnificativă.

În Figura 3.5 arătăm traiectoria prezisă de sistem în diverse situații de trafic. În Figura 3.6 comparam metoda noastra de prezicere a traiectoriei cu abordări state-of-the-art [12 si 15], care se bazează pe predicția de comenzi instantanee. Algoritmul nostru demonstrează că predicțiile pe termen mai lung cresc performanțele cu privire la menținerea traseului.

	Position Error (m)			Orientation Error (deg)		
Method	Response	Mean	Median	Response	Mean	Median
[13]	**100%**	58.09	17.75	**100%**	7.81	2.41
[14]	**100%**	50.84	15.39	**100%**	7.33	1.88
[11]	91.00%	27.36	11.44	-	-	-
Ours1	94.72%	17.31	11.55	-	-	-
Ours2	96.35%	16.89	11.18	96.08%	**3.65**	1.43
Ours3	**100%**	**16.05**	**10.90**	**100%**	3.73	**0.67**

Tabel 3.2. Resultatele pentru predicția de poziție

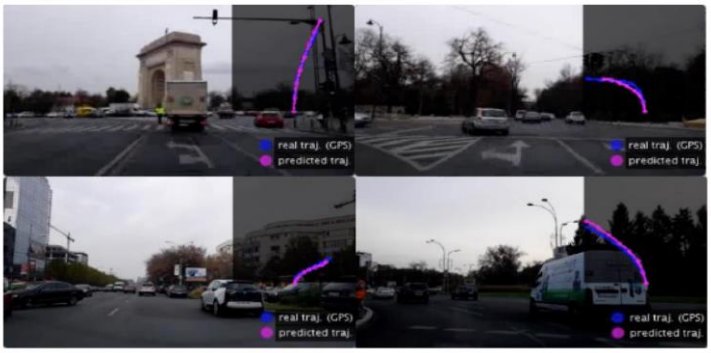


Figura 3.5. Traiectoria prezisă de sistem în diverse situații de trafic: înaintea intrării în intersecție, la începerea virajului la stanga, la începerea virajului la dreapta și virand stanga în sensul giratoriu

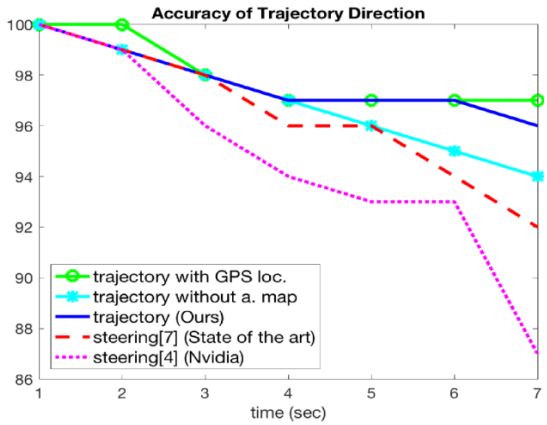


Figura 3.6. Precizia direcției în intersecții pentru punctele traiectoriei. Procentul de puncte ale traiectoriei prezise pe traseul corect, este mai mare pentru modelele noastre, decât pentru cele concurente. Precizia punctelor de frontieră este mai greu de stabilit din cauza faptului că nu sunt disponibile informații precise despre structura și amplasarea drumurilor. Cu toate acestea, punctele de traiectorie care sunt clar în afara drumului (prezente la intervale de timp mai largi din cauza erorilor acumulate) sunt ușor reperate

3.4. Detectarea obstacolelor în 3D din imagini monoculare pentru conducere autonomă

Un alt aspect important în dezvoltarea unui asistent de conducere autonom este capacitatea de a prezice structura 3D a scenei și obstacolele care se mișcă în scenă. Informațiile 3D ar putea ajuta semnificativ navigarea în siguranță, îmbunătățirea localizării (structura 3D a scenei ar putea funcționa ca o amprentă prin care acea anumită locație ar putea fi mai bine recunoscută) și pentru a avea o mai bună înțelegere semantică a scenei. Recent, am introdus noul sistem deep-learning Shift R-CNN, un model hibrid pentru detectarea obiectelor 3D din imagini monoculare, care combină învățarea automată cu principiile geometrice. Am publicat recent această abordare [16] la conferinta International Conference on Image Processing (ICPR), Taipei, 2019.

Adaptăm o rețea Faster R-CNN pentru regresia proprietăților inițiale ale obiectelor 2D și 3D și o combinăm cu o soluție least squares pentru problema de mapare geometrică inversă din 2D în 3D, folosind matricea de proiecție a camerei. Soluția în formă închisă a sistemului matematic, împreună cu ieșirea inițială a Faster R-CNN sunt apoi trecute printr-o rețea ShiftNet finală care rafinează rezultatul folosind noua noastră propunere de dislocare a volumului. Abordarea noastra obține rezultate de top pe KITTI 3D Detection Object Benchmark, fiind cea mai bună dintre toate metodele monoculare care nu utilizează nicio rețea pre-antrenanta pentru estimarea adâncimii.

În Figura 3.7, prezentăm arhitectura completă a sistemului nostru de detectare a obiectelor 3D dintr-o singură imagine, iar în Figura 3.8 prezentăm o compara a rezultatelor in etapiile 2 și 3.

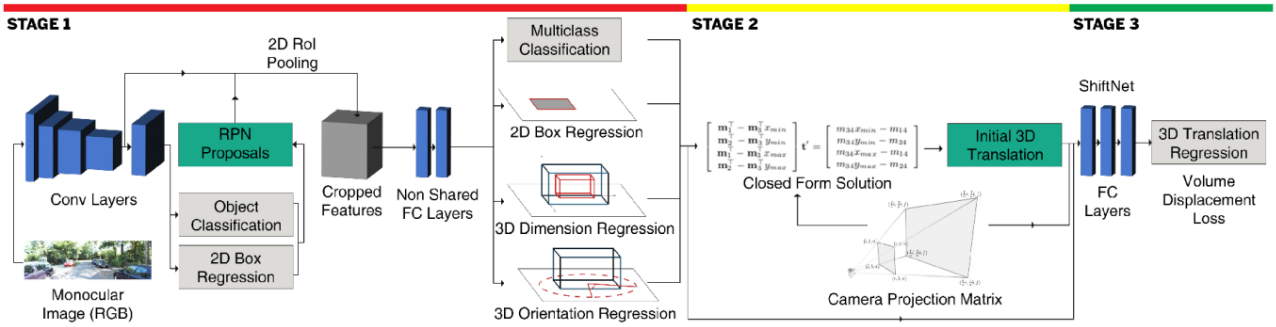


Figura 3.7. Structura globală a sistemului hibrid Shift R-CNN

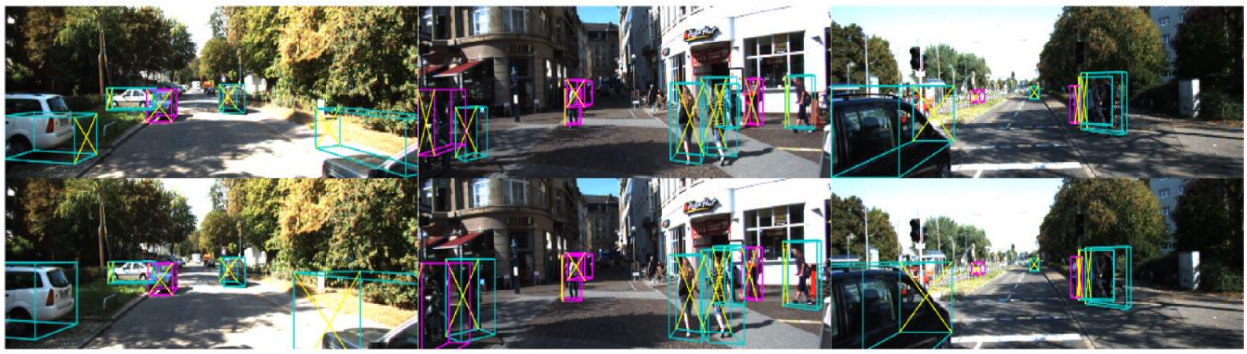


Figura 3.8. Compararea rezultatelor din etapa 2 (sus) și etapa 3 (jos). Etapa 3 îmbunătățește estimarea 3D datorită robusteții sale. Secțiunile turcoaz indica obiecte cu aceeași orientare și culoarea magenta orientarea opusă

Sistemul de detectare a obiectelor 3D ar putea fi ușor combinat cu cel prezentat în secțiunea 3.3 pentru localizarea automată și predicția traiectoriei pentru conducere autonomă. Modulul de evitare a obstacolelor ar putea ajuta la îmbunătățirea traiectoriilor prezise în special atunci când există obstacole în fața vehiculului. În timp ce sistemul din secțiunea 3.3 [10] învață indirect să evite obstacolele, este de așteptat ca o reprezentare explicită a obstacolelor să ajute în mod semnificativ toate aspectele controlului la un siguranță sporit.

4. Proiectul component ROBIN-Context

Parteneri ai proiectului ROBIN-Context

UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI (CO)

UNIVERSITATEA TEHNICĂ DIN CLUJ – NAPOCA

4.1. Prezentare generală

Scopul proiectului ROBIN-Context este crearea unei platforme suport pentru definirea/reprezentarea semantică și gestiunea facilă și eficientă a datelor ce devin context în scenarii de asistență robotică personalizată și sisteme ADAS. Platforma va defini un flux bine stabilit al datelor de context (de la achiziția lor, la diseminare/consum) și va oferi biblioteci suport pentru inferarea situațiilor de nivel semantic înalt prin combinarea de tehnici bazate pe cunoștințe și tehnici bazate pe date.

Obiectivele proiectului ROBIN-Context sunt:

- Definirea unui format de reprezentare a datelor care să fie expresiv și sa poată fi utilizat cu ușurință în inferențe de tip semantic, în sisteme de procesare de evenimente, precum și în modele de inferență pe bază de machine learning.
- Proiectarea și realizarea platformei suport pentru implementarea fluxului de achiziție – inferență – diseminare necesar în procesarea și gestiunea datelor de context pentru aplicații centralizate (de exemplu ADAS) și servicii web.
- Dezvoltarea unei componente de procesare semantică a datelor pentru a defini constrângerile de detecție a unor situații.
- Dezvoltarea unei componente pentru efectuarea inferențelor sub formă de procesare rapidă a evenimentelor.
- Dezvoltarea unei componente de inferență a contextului folosind biblioteci de algoritmi pe bază de modele grafice probabilistice.
- Proiectarea și dezvoltarea unei componente ce asigură capabilitatea de explicare a procesului de inferență contextuală.
- Proiectarea, realizarea și testarea de instrumente și servicii destinate agenților economici.

4.2. Platforma ROBIN-Context ca suport pentru Context-as-a-Service

Platforma ROBIN-Context dezvolta un sistem de suport pentru procesarea informatiei de context utilizata pentru scenarii cu cerinte functionale si non-functionale diferite: unul aferent asistarii conducerii de autovehicule (eng. Advanced Driving Assistance System - ADAS) si unul aferent roboticii sociale asistive. In primul tip (cel de ADAS) informatia contextuala necesara provine din senzori proprii masinii, prin care aceasta intelege scena de condus. In cel de robotica asistiva, insa, informatia contextuala pe care o exploateaza un robot social este in foarte multe cazuri aferenta mediului in care se situeaza robotul (e.g. senzori/acutatori instalati intr-o sala de curs, in casa unei persoane, intr-o incinta comerciala).

Pentru a putea generaliza, modul in care contextul ajunge sa imbunatatesca functionarea unei aplicatii poate fi definit ca un serviciu furnizat aplicatiilor. Se pune atunci intrebarea despre cum se stabileste interactiunea intre partile interesate (eng. stakeholders) intr-un ecosistem de furnizare si consum al informatiei contextuale.

Paradigma sensing-as-a-service [17] defineste un model de productie si consum al informatiilor de context care separa entitatiile care detin senzori (care furnizeaza informatii), de cele care le grupeaza si analizeaza (procesatori intermediari care aduc plus valoare peste informatia primara) si, la final, de cele care consuma informatia fie de la producatorii de date primare, fie la cei intermediari. Aceste entitati si relatiile dintre ele sunt ilustrate in Figura 4.1.

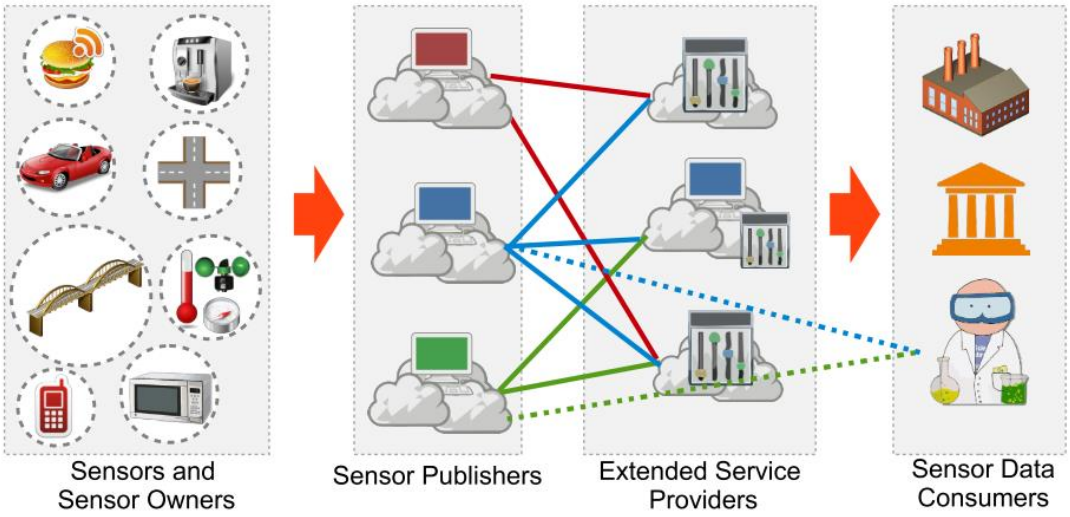


Figura 4.1. Paradigma de sensing-as-a-service [17] in care sunt evidentiate partile interesate (stakeholders) care compun un ecosistem de productie, analiza si consum de informatie contextuala

Platforma ROBIN-Context doreste generalizarea conceptului de sensing-as-a-service la context-as-a-service si propune o metoda prin care entitatile implicate in ecosistemul de producere si consum de context isi pot exprima in mod implicit sau explicit interestul pentru anumite informatii contextuale (de la date ale senzorilor, pana la capabilitati ale actuatorilor intelgenti - a se vedea Sectiunea 4.4).

Unitatile de procesare care compun platforma ROBIN-Context (e.g. CtxSensor, CtxCoordinator, CtxUser) au fost prezentate in raportul anterior. Tot in raportul anterior au fost explicate notiunile de Dimensiune si Domeniu Contextual (ContextDimension si ContextDomain), care fac o impartire logica a informatiilor partajate intr-un sistem de tip context-as-a-service.

Figura 4.2 arata modul in care se face acum explicit interesul pentru un informatiile partajate intr-un anumit ContextDomain, anume prin apartenenta la un ContextDomainGroup, care are ca obiectiv regruparea informatiilor conexe unei Dimensiuni Contextuale (e.g. prezenta intr-o anumita sala, implicarea intr-o anumita activitate).

Apartenenta la un anumit ContextDomainGroup se poate face (i) explicit: prin cerere de aderare, (ii) implicit: considerand ca anumite evenimente din mediu conteaza ca o cerere de aderare acceptata (e.g. un senzor de miscare si o camera RGB detecteaza prezenta unei anumite persoane sau a robotului asistiv intr-o sala de laborator -> prin urmare persoana/robotul sunt inclusi in ContextDomainGroup-ul aferenta dimensiunii de prezenta intr-un laborator si a domeniului care denota instanta laboratorul in cauza).

Ideea de acces la date contextuale pe masura interesului cere ca acest acces sa fie protejat, atunci cand apartenenta la grupul contextual nu este nici implicit asigurat, nici explicit acordat. Pentru aceasta, natura ierarhica posibila peste ContextDimensions si ContextDomains (a se vedea partea din dreapta a Figurii 4.2) este folosita pentru a defini un mecanism de criptare prin chei partiale, astfel incat unitatea de tip CtxCoord, care gestioneaza accesul la un ContextDomainGroup sa poata decripta toate mesajele venite dinspre CtxSensors sau CtxUsers care fac parte din grup, dar CtxSensors intre ei sa nu poata decripta mesajele unul altuia.

Toate aceste conceptualizari si proiectari arhitecturale sunt prezentate mai pe larg in doua articole publicate de catre echipa de proiect a ROBIN-Context [18 si 19].

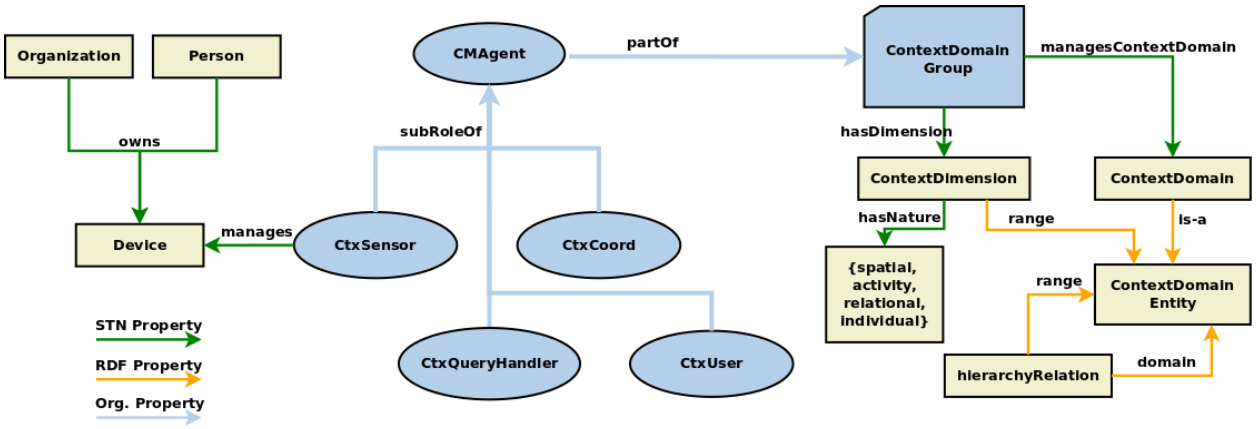


Figura 4.2. Organizarea conceptuala a unitatilor de procesare de context din platform ROBIN-Context pe Grupuri Contextuale care definesc interesul/focusul lor curent in termen de informatie contextuala pe care doresc sa o produca/consume

4.3. Inglobarea rationamentelor probabilistice in platforma ROBIN-Context

Platforma ROBIN-Context pleaca de la dezvoltarile facute in directia motorului CONSERT [20]. Motorul de inferența CONSERT este o componenta de raționament asupra informației contextuale, a carei functionalitate este similara celei unui motor de recunoaștere de evenimente complexe (semantic complex event processing - SCEP). Pentru a integra rationament de tip probabilistic, asa cum a fost descris in raportul anterior, a fost necesara introducerea unui nou tip de structura de date asupra careia sa se poata aplica tehnici de invatare automata.

In motorul CONSERT a fost introdusa *fereastra de evenimente (EventWindow)*, care cumuleaza evenimente individuale care definesc *istoria* unei activitati.

Un *EventWindow* in motorul CONSERT poate fi construit avand doua tipuri de limite:

- *temporala:* in fereastra de evenimente intra evenimente care au avut loc in utlimele *t* unitati de timp (e.g. secunde).
- *numerica:* fereastra de timp contine un anumit numar maxim *N* de evenimente.

O particularitate anume a motorului CONSERT este aceea ca o fereastra de evenimente poate fi asociata unui tip *probabil* de asertiune contextuala compusa, iar evenimentele individuale pot fi incluse in fereastra in urma unor presupuneri a-priori umane.

In acest fel se face integrarea intre cunostinte de bun simt (e.g. o activitate didactica nu se poate executa la toaleta), cu metode de inferenta din date.

Cu alte cuvinte, in motorul de inferenta CONSERT sunt introduse reguli de procesare prin care se stabileste *in ce conditii* se introduc evenimente intr-o fereastra de evenimente aferenta unei activitati.

Regulile introduse sunt de trei tipuri:

- reguli care stabilesc crearea unei noi instante de fereastra de evenimente pentru un anumit tip de asertiune contextuala.
- reguli care adauga evenimente noi la o instanta existenta de fereastra de evenimente.
- reguli care determina inchiderea explicita a unei ferestre de evenimente (acestea sunt reguli suplimentare celor de timp/numar care fac inchiderea in mod implicit).

Pentru a explora acest mecanism de lucru, platforma ROBIN-Context si motorul CONSERT, in particular, au fost folosite pentru a detecta activitati cotidiene (eng. Activities of Daily Living) din setul de date CASAS Kyoto [21]. Figura 4.3 ilustreaza modul in care sunt amplasati senzori cu activare binara (senzor declansat versus nedeclansat) intr-un apartament de test. Exemple de tipuri de senzori folositi sunt cei de miscare, de inchidere sau deschidere a unor usi/dulapuri, de utilizare a unor obiecte de uz casnic.

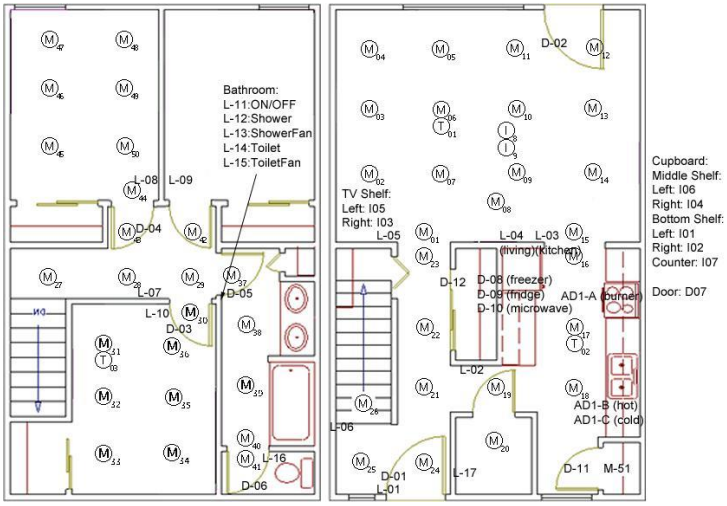


Figura 4.3. Amplasare a senzorilor cu evenimente binare intr-un apartament de test din proiectul CASAS [21]

Activarile senzorilor au fost asadar adnotate manual, in sensul ca in dreptul fiecarei activari de senzor e trecuta si activitat(ea)/atile corespunzatoare. Pe toata durata experimentelor apartamentul a fost ocupat de o singura persoana, iar in total au participat intre 20 - 25 de persoane la teste, in functie de tipul acestora. Experimentele inregistrate au urmat doua scenarii. Intr-o prima instanta persoanele sunt rugate sa execute o secventa predefinita de activitati, fara intreruperea acestora (ADL normal). In al doilea scenariu, activitatile pot fi intrepatrunse - o persoana poate incepe sa faca o activitate, apoi o incepe pe a doua, o termina pe prima, apoi o termina pe a doua (ADL Interwoven). Aceste scenarii precum si modul lor de evaluare sunt prezentate in sectiunile urmatoare.

Atunci cand se folosesc ferestre de evenimente pentru a prezice tipul de activitate probabil, este nevoie de calculul unor atribute care sa surprinda un fel de "rezumat" al secventei de evenimente inclusa in fereastra.

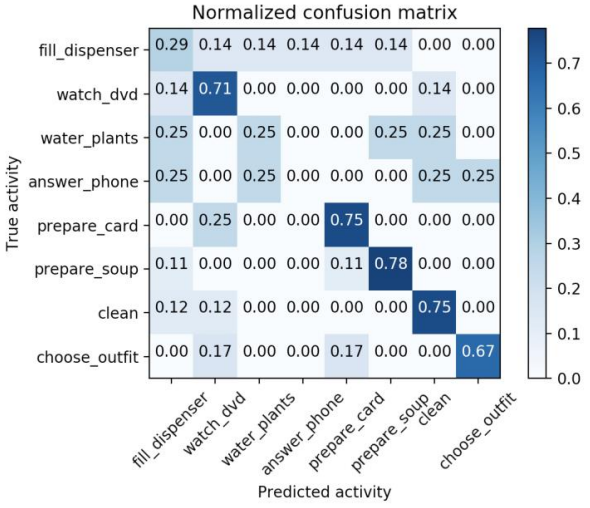
Pentru a analiza puterea de predictie a unei ferestre ideale pe setul de date CASAS, ferestrele evaluate initial sunt construite pe baza adnotarilor manuale (i.e. ele cuprind fix acele evenimente care fac parte dintr-o instanta al unui anumit tip de activitate - e.g. curatenie, udat plante, gatit, imbracat haine).

Pe aceste ferestre se considera mai multe moduri de rezumare a evenimentelor:

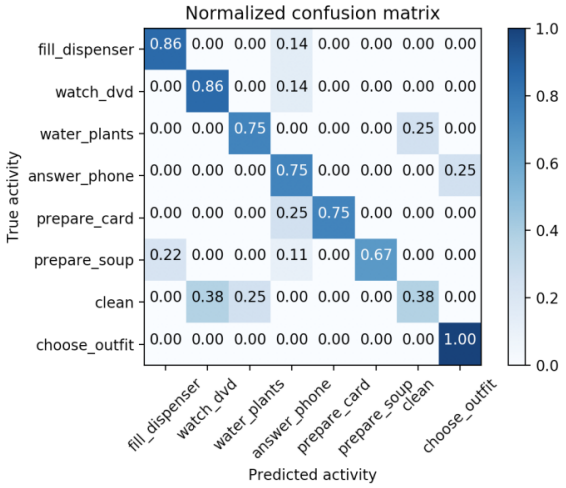
- tipul senzorilor.
- contor per tip de activari.
- contor per tip de activari si dezactivari.

Peste aceste atribute ale unei ferestre se foloseste un algoritm de tip Support Vector Classifier, pentru a clasifica fereastra intr-una din 8 tipuri de activitati. Rezultatele clasificarii se observa in Figura 4.4 unde sunt indicate matricile de confuzie ce arata gradul de acuratete in predictia fiecarui tip de activitate in parte.

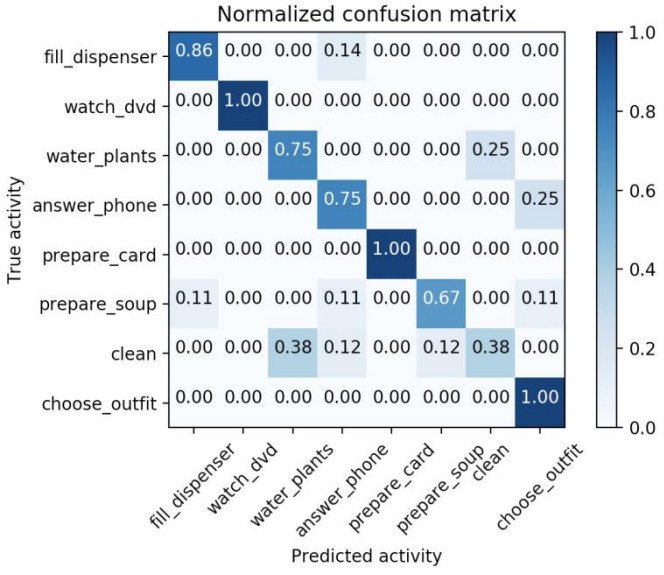
Analizand, se poate observa ca atributele care tin cont atat de un contor de activari, cat si de unul de dezactivari per tip de senzor din fereastra, sunt cele care dau rezultatele cele mai bune. Acest rezultat este unul promitator, dar metoda trebuie studiata in continuare, intrucat pe cazuri reale, includerea perfecta a evenimentelor individuale in ferestre perfecte aferente activitatilor nu poate fie garantata. Ferestrele vor fi incomplete sau vor contine evenimente care apartin altor activitati. Astfel, in dezvoltari urmatoare se cauta metode de calcul al unor atribute suplimentare, precum si metode care sa poata performa in conditiile unor ferestre de evenimente incomplete.



a) Matrice de confuzie pentru tipuri de senzor



b) Matrice de confuzie pentru contor de activare per tip de senzor



c) Matrice de confuzie pentru contor de activare si dezactivare per tip de senzor

Figura 4.4. Rezultatele clasificarii de ferestre de evenimente prin cele 3 metode de calcul a atributelor pe fereastra: a) tipuri de senzori, b) contor de activari per tip de senzor, c) contor de activari si dezactivari per tip de senzor

4.4. Interfatarea platformei ROBIN-Context cu sisteme externe

Un aspect important este acela ca intr-un sistem ADAS comunicarea se face in mediu inchis, pe cand in robotica social asistiva, schimbul de informatie este majoritar cu sisteme externe (i.e. alti senzori/actuatori sau aplicatii externe). Multe dintre platformele de perceptie senzoriala si control instalate pe masini cu sisteme ADAS sau cu autonomie se folosesc de un framework de comunicare des utilizat si in robotica, pe nume ROS (Robot Operating System). Ca atare, pentru obtinerea si furnizarea informatiilor de context in astfel de sisteme, platforma ROBIN-Context, si motorul CONSERT in particular, au fost extinse pentru a putea comunica peste ROS. Precum s-a aratat in Sectiunea 4.2, interactiunea in scenariile de robotica sociala asistiva implica accesul la sisteme contextuale externe si se incadreaza in ceea ce poate fi numit *context-as-a-service.* Pentru ca aceste sisteme externe sunt deseori eterogene, deschise si cu durata medie spre lunga de viata, o propunere foarte sustinuta de interactiune este cea prin *servicii web RESTful.*

4.4.1. Reprezentarea Asertiunilor Contextuale in mesaje ROS

Pentru comunicarea peste ROS, motorului CONSERT i-a fost adaugata o noua interfata care sa-i permita sa actioneze ca un nod ROS[4]. In ROS, nodurile isi comunica folosind canale aferente unui *topic* de discutie. Nodurile pot actiona drept emitator sau receptor al mesajelor transmise pe un canal.

Motorul CONSERT defineste trei canale/topicuri de comunicare:

- *consert/engine/contextAssertions:* Topic pe care sunt publicate catre exterior asertiunile contextuale inferate de motorul CONSERT
- *consert/engine/insertedAssertions:* Topic pe care sunt motorul CONSERT asculta dupa asertiuni introduse de aplicatii externe (incluzandu-le pe cele deduse prin analiza de ferestre de evenimente, publicate de submotorul de inferenta probabilistica)
- *consert/engine/eventWindows:* Topic pe care sunt publicate catre sub-motorul de inferenta probabilistica ferestrele de evenimente nou create sau actualizate cu evenimente individuale

```
# The URI ID of this ContextAssertion instance
string id
# The class type of this ContextAssertion instance
string type
# Arity of the ContextAssertion: 1, 2 or 3 (for n-ary ContextAssertions)
uint8 arity
# The acquisition type of the ContextAssertion
string acquisitionType
# The key-value dictionary of entities and their roles in this ContextAssertion instance
EntitiyRole[] entities
# The annotations of this ContextAssertion instance
ContextAnnotation[] annotations
```

Listarea 4.1. Reprezentarea unui Mesaj de Asertiune Contextualai in format ROSMsg

In Listarea 4.1 se prezinta campurile pe care le contine un mesaj aferent unei asertiuni contextuale in format ROSMsg. Sunt serializate elementele care definesc asertiunea in mod unic: id-ul acesteaia, tipul (numele calificat al clasei Java aferenta asertiunii, prin care se poate face instantierea acesteia in motorul CONSERT), aritatea asertiunii, modul in care a fost obtinuta (de la senzori, in mod explicit de la utilizator sau in mod derivat), precum si entitatile contextuale peste care este definita asertiunea. Nu in utlimul rand, este cuprinsa lista de adnotari care caracterizeaza suplimentar asertiunea (e.g. timestamp, durata temporala, coeficient de incredere in acuratetea asertiunii). Toate aceste modificari sunt public accesibile pe contul de Github aferent proiectului CONSERT[5].

4.4.2. Interactiunea platformei ROBIN-Context cu actuatori inteligenti

Platforma ROBIN-Context pleaca de la conceptul arhitectural din CONSERT [22]. In raportul anterior s-a mentionat ca varianta actualizata a framework-ului CONSERT se bazeaza pe comunicare prin protocol web RESTful, avand ca suport fie HTTP, fie Websockets. In iteratia curenta, s-a propus extinderea aplicabilitatii platformei ROBIN-Context de la furnizor al informatiilor percepute (eng. sensed) de catre senzori, la una in

4 http://wiki.ros.org/rosnode

5 https://github.com/ami-lab/CONSERT/tree/63-possible-event-for-mechanism

care platforma poate gestiona interactiunea contextuala (e.g. descoperirea, cautarea, actionarea directa) cu actuatori inteligenti (eng. smart devices - becuri inteligente, jaluzele automatizate, termostate inteligente...).

In mod particular, platforma ROBIN-Context trebuie sa furnizeze o descriere a *capabilitatilor* dispozitivelor inteligente (care este, in esenta, informatie de context), cat si a *modului de interactiune* cu aceste dispozitive (i.e. o descriere semantica a API-ului de serviciu web prin care se exploateaza capabilitatile dispozitivelor - e.g. aprinderea/stingerea/schimbarea culoarii unui bec inteligent, ridicare/coborarea jaluzelelor).

Pentru a ramane in sfera comunicarii peste web, platforma ROBIN-Context se inscrie in trendul puternic sustinut de comunitatea W3C de a modela *capabilitile* si *modul de interactiune* cu acestea folosind tehnici web. Modelul propus de W3C se numeste ThingDescription[6] si este o specificatie cu titlul de Candidate Recommendation.

Web-of-Things ThingDescription se ocupa cu descrierea interactiunii cu un dispozitiv web-enabled in termen de:

- proprietati observabile: Caracteristici care descriu starea curenta a dispozitivelor sau a perceptiei lor asupra mediului in care sunt amplasate, e.g. culoarea curenta a unui bec, pozitia curenta a jaluzelelor, temperatura inregistrata de termostat.
- evenimente: notificari asupra schimbarilor de stare ale dispozitivelor, e.g. aprinderea/stingerea becului, supraincalzirea becului, scaderea temperaturii sub un prag stabilit.
- actiuni: interactiuni cu un dispozitiv prin care se poate schimba starea acestuia, e.g. aprinderea luminii, comanda de ridicare sau coborare a jaluzelei, schimbarea pragului de temperatura al termostatului pentru o anumita camera.

ThingDescription este, in esenta, un model prin care se specifica API-ul RESTful prin care se:

acceseaza proprietatile observabile

- se face inscrierea pentru a primi notificari de evenimente.
- se trimit POST-uri HTTP cu argumentele necesare schimabrii de stare prin actiuni.

Descrierea semantica a actiunilor se face, ca si pana acum, folosind o descriere semantica ce utilizeaza meta-modelul CONSERT.

In cadrul proiectului ROBIN-Context s-a prototipat o modelare, dupa specificatiile ThingDescription, a API-ului de acces la o lampa inteligenta (pe sistem Philips Hue[7]) si la un sistem custom de control al jaluzelelor, ambele instalate in laboratorul de cercetare al echipei din UPB. Prototipul este public accesibil in repository-ul de cod aferent proiectului[8].

4.5. Continuarea cercetărilor

In etapa imediat urmatoare, eforturile de cercetare si dezvoltare se vor concentra pe urmatoarele:

- Extinderea capabilitatii de tip context-as-a-service a platformei ROBIN-Context de la prototipul pentru lumini inteligente si jaluzele la o platforma generala
  - Implementarea specificatiilor WoT ThingDescription pentru modelarea API-ului de comunicare cu actuatori inteligenti exploateatii in ciclul de viata al unui robot social.
  - Implementarea conceptului de ContextDomainGroups la nivel general, unde acestea vor actiona ca mijloc expres de modelare a interesului pentru informatii de context
  - Implementarea unor mecanisme de cautare si descoperire a actuatorilor si senzorilor pe baza apartanentei la ContextDomainGroups
- Extinderea tipului de algoritmi probabilistici care exploateaza ferestre de evenimente si studiul compartiv al perfomantei acestora. Aici se include si studiul utilizarii algoritmilor de tip deep learning pentru analiza de secvente.

6 https://www.w3.org/TR/wot-thing-description/

7 https://www2.meethue.com/en-au

8 https://github.com/ami-lab/CONSERT-CaaS/tree/10-manager-node

5. Proiectul component ROBIN-Dialog

Parteneri ai proiectului ROBIN-Dialog

INSTITUTUL DE CERCETĂRI PENTRU INTELIGENȚĂ ARTIFICIALĂ "MIHAI DRĂGĂNESCU" (CO)

UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI

UNIVERSITATEA TEHNICĂ DIN CLUJ – NAPOCA

5.1. Prezentare generală

Obiectivele proiectului ROBIN-Dialog: Presupune dezvoltarea unei serii de scenarii pentru câteva micro-lumi și tehnologia de prelucrare a limbii române pentru dialoguri situaționale în aceste micro-lumi. Această tehnologie va fi validată pe scenariile și micro-lumile cercetate, dar va fi dezvoltată în așa fel încât să poată fi aplicată cu ușurință și pe alte scenarii și/sau micro-lumi. Caracterul de generalitate va fi asigurat de metodele de învățare automată de tip ”deep learning” și de specificarea resurselor (e.g. baze de cunoștințe) în limbaje standard (e.g. XML/RDF) care vor asigura funcționarea sistemului în orice micro-lume și/sau scenariu, atâta timp cât datele de antrenare și resursele specifice vor fi disponibile pentru acestea.

- Proiectarea acestor scenarii și a sistemului de dialog situațional în limba română presupune următoarele activități:
  a) Construirea unui lexicon de cuvinte și expresii reprezentative pentru micro-lumea țintă. Exemple de micro-lumi sunt: i. o casa inteligentă; ii. Un robot acționând într-un mediu/spațiu specificat
  b) Extensia automată utilizând metode semantice moderne („continuous vector spaces”) a lexiconului creat manual la pasul 1a
  c) Crearea universului de discurs pentru micro-lumea/scenariul selectat. Acest pas implică identificarea relațiilor semantice care se stabilesc între cuvinte și care astfel devin predicate care vor fi validate (adevărat/fals) în contextul dialogului.
- Intrările resursei lexico-semantice creată în pasul 1b vor fi transcrise fonetic și aliniate cu semnalul vocal corespunzător în cazul în care aceste înregistrări există în CoRoLa.
- Sistemele de antrenare ASR și TTS vor fi alimentate cu rezultatele pasului 2. Sistemele ASR si TTS vor fi testate si validate.
- Implementarea sistemului de dialog cooperant pentru micro-lumile selectate.

5.2. Transcrierea fonetică a cuvintelor din lexiconul validat

În etapa anterioară, descriam procedura de construcție a unui lexicon pe baza descrierilor micro-lumilor țintă. Vorbeam despre extragerea tuturor lemelor din aceste descrieri într-o listă inițială și menționam două strategii de extindere a acestei liste cu cuvinte similare sau aflate în relație semantică: utilizarea reprezentărilor vectoriale învățate automat (cunoscute și ca “word embeddings”) pentru a identifica cuvinte similare și utilizarea wordnetului românesc (RoWordnet) pentru a extrage hiperonime și sinonime. De asemenea, menționam utilizarea resursei interne tbl.wordfom.ro (peste 1.150.000 de intrări) pentru a genera familia de cuvinte a fiecărei leme și pentru a completa lexiconul ROBIN cu etichete morfo-sintactice. La sfârșitul etapei 2018, lexiconul (validat la nivel de lemă și etichetă morfosintactică) conținea 99150 înregistrări de forma:

<formă ocurentă>tab<lemă>tab<etichetă morfo-sintactică>

În cadrul acestei etape, ne-am propus, pe de o parte, (1) să rafinăm prin metode semantice lexiconul pentru a ne asigura ca nu există intrări care să fie în afara universului de discurs și pe de alta parte (2) să completăm lexiconul cu informația de silabificare, accent și transcriere fonetică, utile în aplicațiile de ASR și TTS. Rafinarea lexiconului a avut loc într-o cercetare descrisă în [23], care își propunea să prezinte crearea lexiconului în contextul mai larg al proiectului ROBIN și să evalueze utilitatea vectorilor semantici și a wordnetului românesc în extinderea lexiconului. În acest context, am experimentat cu dezambiguizarea semantică în contextul micro-lumilor a cuvintelor din lexicon (prin asocierea de sensuri din wordnet) înainte de expandarea lexiconului pe relațiile de hiperonimie și sinonimie. Astfel, am reușit să restrângem lista de leme din lexiconul expandat la 1827, iar lista de intrări din lexicon (conținând variantele morfologice ale acestor leme) s-a redus de la 99150 intrări la 27559 intrări. Considerăm că versiunea rafinată a lexiconului este mai utilă sistemului de dialog ROBIN, deoarece elimină ambiguități semantice și evită supraîncărcarea acestuia cu informație inutilă.

Completarea lexiconului cu informațiile de silabificare, accent și transcriere fonetică s-a făcut în două etape (descrise în raportul în extenso), intrările lexiconului având forma:

<formă>tab<lemă>tab<etichetă_morfo-sintactică>tab<silabificare>tab<accent>tab<transcriere_fonetică>

Exemplu:

cercetăm cerceta Vmip1p cer.ce.tăm cercet'ăm [tS e r tS e t @ m]

Împărțirea în silabe este marcată prin simbolul ".", accentul este marcat printr-un apostrof în poziție anterioară vocalei accentuate iar transcrierea fonetică este afișată între paranteze drepte. Alfabetul folosit pentru transcrierea fonetică este SAMPA[9].

Corectarea erorilor de silabificare, accent și transcriere fonetică (de așteptat în urma generării automate) a avut loc în ordinea silabificare -> accent -> transcriere fonetică. Etapizarea corecturii permite automatizarea anumitor pași, deoarece în unele cazuri transcrierea fonetică este dependentă, în mod determinist, de silabificare și accent (v. regulile de mai jos).

Corectarea silabificării s-a făcut integral manual, concentrându-ne pe: cuvintele care nu se găsesc în RoSyllabiDict; cuvinte care conțin silabe mai lungi de patru litere; cuvinte care conțin secvențe de vocale + semivocale, etc. În corectarea manuală a accentului, am acordat atenție specială cuvintelor cu două variante de accent (omonimii), forme care pot fi substantive sau verbe (ex.: „data"), forme verbale diferite ale aceleiași leme (ex.: „atribui"), forme verbale diferite pentru leme identice (ex. „alungi", cu lemele „alunga" și „alungi"). Raportul în extenso detaliază principalele probleme și rezolvarea lor.

Obiectivul a fost integral îndeplinit, s-au transcris fonetic toate intrările lexiconului construit, s-au completat intrările cu informație suplimentară, necesară sistemelor de prelucrare a vorbirii (silabificare, marcarea accentului).

5.3. Aliniere text-voce și încărcarea în componenta de vorbire a corpusului CoRoLa

Având în vedere că înregistrările realizate în format .WAV sunt redări fidele ale fișierelor text, a fost realizată alinierea automată a acestora, utilizând un proces similar celui utilizat la alinierea fișierelor text-voce din componenta audio a corpusului CoRoLa. Acest proces, descris în [24], constă în realizarea unui format intermediar de text fără semne de punctuație sau alte elemente în afară de cuvintele pronunțate, cu toate literele mici și urmat de alinierea propriu-zisă a cuvintelor cu sunetele înregistrate. Fișierele realizate în urma aplicării acestei proceduri sunt salvate cu extensia ".lab". Pentru alinierea propriu-zisă, a fost utilizată unealta software HTK [25], fiind produse la final fișiere .phs conținând momentul de start și momentul de sfârșit asociat fiecărui fonem prezent în text. Pe baza acestora putându-se reconstitui momentele de început și sfârșit asociate fiecărui cuvânt din text.

În urma alinierii, fișierele au fost indexate în componenta audio a corpusului CoRoLa, fiind apoi disponibile pentru căutare în interfața acestuia, disponibilă online la adresa: http://89.38.230.23/corola_sound_search/. În urma realizării unei interogări, cuvântul găsit poate fi ascultat în fiecare fișier audio (pe baza alinierilor realizate) sau întreaga frază poate fi ascultată.

Obiectivul a fost integral îndeplinit, toate înregistrările noi au fost aliniate cu transcrierile lor. În plus, ele au fost indezate și introduce in componenta de vorbire a corpusului de referință al limbii române CoRoLa.

5.4. Crearea înregistrărilor vocale pentru cuvinte care nu au înregistrări în corpus

Au fost identificate în lexiconul construit anul trecut aproape 300 de cuvinte pentru care nu există inregistrari vocale in corpusul CoRoLa. Din corpusul textual s-au extras propoziții ce conțineau cuvintele respective și s-au inregistrat propozițiile alese în rostiri de către doi bărbați și două femei. Au fost construite fișierele txt pentru care au fost create fișierele wav corespunzătoare cu frecvența de eșantionare 48khz și 44khz. Echipamentul de înregistrare a fost reprezentat de casti cu microfon (Huawei P20 Pro). Microfonul avea atat functie de "noise reduction" cat si de "echo cancellation". O parte din înregistrari au fost efectuate în camera izolată fonic și restul în camere obișnuite. Programul software care s-a folosit a fost Audacity, cu setarile default. Fișierele cu extensia .lab au fost pregătite corespunzător. În final, noile înregistrări au fost încărcate în secțiunea de voce a corpusului public CoRoLa.

Obiectivul a fost integral indeplinit, s-au realizat înregistrările de foarte bună calitate ale frazelor context pentru cuvintele țintă.

[9] https://www.phon.ucl.ac.uk/home/sampa/romanian.htm

5.5. Sistemele de antrenare ASR și TTS vor fi alimentate cu noile date. Testarea și evaluarea sistemelor ASR si TTS.

Pentru antrenarea modulului de TTS au fost urmati pasii de la https://github.com/tiberiu44/TTS-Cube/blob/master/TRAINING.md. In bazele de date de care dispunea RACAI, existau deja fisiere pregatite pentru acest sistem. Cele mai multe fisiere, erau pentru vocea Ancai, motiv pentru care calitatea ei este cea mai buna. Calitatea vocii este bună, se poate întelege clar mesajul transmis. Principalul dezavantaj al sistemului este timpul mare de prelucrare a textului. Chiar și pe echipamente hardware puternice, timpul de sinteza depinde de lungimea textului, astfel incat pentru o propozitie mai lungă poate ajunge la câteva secunde.

Pentru antrenarea modulului ASR S-a pornit de la toate bazele de date de care dispunea ICIA. Fisierele .txt care contineau caractere invalide (cifre, accente din alte limbi) au fost eliminate impreuna cu inregistrarile. S-a obținut astfel o serie de înregistrări cu transcrierea aferentă, transcriere care este curată și poate fi folosită pentru antrenat. Am dezvoltat un ASR in Kaldi cu ajutorul resurselor din Corola. Demo-ul este disponibil la adresa http://relate.racai.ro/index.php?path=robin/asr.

Există limitare pentru fisiere input să fie doar .wav, Dimensiunea maximă nu reprezintă o limită dar serverul va da timeout cu un fișier mai lung de 10 minute pentru că durează mult procesarea. (>10 min). Inregistrările sunt din proiectul Robin iar dictionarul folosit la antrenare este 3-gram pruned cu rescoring pe 4 gram. Este folosit Kaldi cu modele GMM-HMM.

Performanta modelului este WER = 30%.

Demo

Text rostit:

ÎN ROMÂNIA DOAR OPTSPREZECE LA SUTĂ DIN POPULAȚIE CONȘTIENTIZEAZĂ IMPORTANȚA SALVĂRII UNEI VIEȚI OMENEȘTI PRIN DONAREA DE ORGANE IAR UN PROCENT FOARTE MIC ESTE ÎNREGISTRAT ÎN RÂNDUL TINERILOR ÎNTRE ȘAISPREZECE ȘI DOUĂZECI ȘI CINCI DE ANI TRANSMITE CORESPONDENTUL RADIO ROMÂNIA ACTUALITĂȚI IOAN SUCIU CITÂND DATELE PREZENTATE ASTĂZI LA ARAD

Predicție:

ÎN ROMÂNIA DOAR OPTSPREZECE LA SUTĂ DIN POPULAȚIE CONȘTIENTIZEAZĂ IMPORTANȚA SALVĂRII UNEI VIEȚI OMENEȘTI PRIN DONAREA DE ORGANE IAR UN PROCENT FOARTE MIC ESTE ÎNREGISTRATĂ ÎN RÂNDUL TINERILOR ÎNTRE ȘAISPREZECE ȘI DOUĂZECI ȘI CINCI DE ANI TRANSMITE CORESPONDENTUL RADIO ROMÂNIA ACTUALITĂȚI IOAN SUCIU CITÂND DATELE PREZENTATE ASTĂZI LA ARAT

Un alt experiment de ASR a fost realizat în grupul de la UPB folosind alte metode (DeepSearch) decât cele utilizate la ICIA (GMM-HMM). Prezentarea și evaluarea acestui experiment sunt furnizate în continuare.

Setul de date SWARA

Seturile de date pentru limba română sunt limitate, având un număr mic de vorbitori. De obicei, se pot găsi înregistrări audio specifice pentru fragmente mici de text, care au fost create pentru antrenarea unor modele adaptate unor nevoie foarte specifice. De asemenea, nu există seturi de date mai dificile, cu zgomot de fundal sau vorbitori simultan. În contextul antrenării modelului DeepSpeech, aceasta este o limitare pentru obținerea unei performanțe bune în ceea ce privește WER.

În ciuda faptului că seturile de date în limba română sunt limitate ca dimensiune, corpusul SWARA [26] s-a dovedit a fi un bun candidat pentru antrenarea rețelei neurale de la DeepSpeech. Acesta conține aproape 21 de ore de conținut vorbit, înregistrat în condiții de studio de către 17 vorbitori, bărbați și femei. Aceste statistici arată o varietate bună, cu toate că rețeaua nu ar putea învăța să ignore zgomotul de fundal și alte probleme din lumea reală. De asemenea, setul de date este curățat, aliniat și conține transcrierile fonetice ale tuturor cuvintelor care apar în el. Acest lucru este foarte util, pentru că permite testarea unor metode variate, printre care și modele bazate pe HMM sau DeepSpeech.

Antrenare modelului DeepSpeech

Primul pas în antrenarea arhitecturii DeepSearch [27] este segmentarea corpusului SWARA. O intrare a setului de date inițial conține transcrierea unei propoziții și un fișier .wav cu înregistrarea unui vorbitor care rostește propoziția curentă. Colecția acestor intrări conține aproape 20000 de elemente. Durata medie a unei

înregistrări este de 30 secunde, iar propozițiile transcrise au în medie 10 cuvinte. Aceste intrări au fost împărțite în partiții de aproximativ 60%

Rezultate

Așa cum a fost menționat anterior, rețeaua DeepSpeech fost antrenată pe setul de date SWARA, modificând hiper-parametrii pentru a obține cel mai bun model în ceea ce privește WER. În timpul experimentelor, dimensiunea celulelor a fost variată de la 512 la 2048, pentru a micșora modelul, păstrând același număr de straturi. WER a variat între 58% și 63% pentru diferite dimensiuni ale rețelei folosind modelul de limbă predefinit (vezi Tabelul 5.1). Deși WER nu a fost considerabil mai bun de la 1024 la 2048, antrenarea modelului din urmă dura aproape 4 ore. Prin comparație, procesul dura numai 1.5 ore pe modelele mai mici. Aceleași experimente au fost efectuate și cu modelul de limbă pe română KenLM [28], iar WER s-a îmbunătățit substanțial cu aproximativ 20%, variind între 39% și 42%. Toate rezultatele din Tabelul 5.1 au fost obținute folosind o rată de antrenare de 0.0001, 20 de epoci, iar dimensiunea unui batch fiind 24. De asemenea, au fost făcute experimente și cu rate de antrenare mai mari, dar rezultatele au fost mai slabe. De asemenea, numărul de epoci a fost variat între 15 și 20, dar rezultatele au fost similare.

Dimensiunea celulei	WER folosind modelul predefinit	WER cu modelul în română
2048	58 %	39%
1024	60 %	40%
512	63%	42%

Tabelul 5.1. WER pentru diferite configurații ale modelului (rata de antrenare 0.0001 și 20 de epoci)

Tabelul 5.2 conține exemple de propoziții din setul de test care au fost recunoscute folosind modelul de dimensiune 1024 și modelul de limba română. Din pricina faptului că există diferențe între formele cuvintelor, iar unele elemente au fost recunoscute drept pauze în loc de foneme, WER este influențat. Acesta este calculat drept numărul de înlocuiri, adăugări și ștergeri de cuvinte, împărțit la numărul de cuvinte din propoziția corectă. În acest caz, rezultatul a avut mai multe cuvinte, influențând WER. Acest lucru arată că un model specific de limbă poate îmbunătăți recunoașterea și compensa cu un corpus mai mic de antrenare. O altă metrică măsurată a fost eroarea la nivel de caracter (eng. „Character Error Rate” – CER), care a fost în jur de 15% pentru variantele cu model specific de limbă.

Propoziția originală	Propoziția recunoscută
“erorile ne-au costat”	“erori le ne au costat”
“apreciază punctualitatea”	“aprecia să punctualitate”
“directorii nu organizaseră niciun concurs”	“direct primul ma nica să rând cum concurs”
“a precizat antrenorul sorin cârțu”	“apreciat an trenul ori încă u”

Tabelul 5.2. Exemple de propoziții din setul de test folosind modelul cu dimensiunea 1024

Obiectivul a fost integral îndeplinit, modulele de TTS si ASR au fost antrenate, testate și evaluate.

5.6. Implementarea prototipului generic de sistem de dialog cooperant

Sistemul de dialog cooperant dezvoltat pentru proiectul ROBIN-Dialog este un sistem de dialog care funcționează pentru o „microlume” definită de utilizator. Așa cum am descris în rapoartele anterioare, o microlume este o mulțime de concepte împreună cu relațiile care se stabilesc între ele despre care poate fi vorba într-un schimb de interacțiuni dintre utilizator și robot.

Sistemul de dialog a fost implementat în Java 1.8 și conține următoarele componente:

- **Definiția unei microlumi** care se face într-un fișier text dar care, datorită nivelurilor de abstractizare oferite, se poate face și în clase derivate, direct în Java;
- **Analiza interogării în limba română** care se face automat cu RELATE [29] și în urma căreia sistemul de dialog recunoaște concepte și predicate prezente în cerere. De exemplu, folosind dialogul de mai sus, în întrebarea „În ce sală se ține laboratorul de robotică?”, analizorul de limbaj natural extrage variabila „sala” fiind cea căreia trebuie să i se găsească o valoare în microlumea dată și „laboratorul de robotică” care este un eveniment (laborator în cazul de față) ancorat în microlumea dată (adică definit în microlumea dată de către inginerul de cunoștințe);

- Algoritmul de unificare care caută cel mai apropiat predicat din universul de discurs (microlume) care poate răspunde cererii utilizatorului prin legarea uneia sau mai multor variabile lăsate în suspensie. Pentru exemplificare, folosind de asemenea dialogul de mai sus, întrebarea „Unde se află sala în care se ține laboratorul de robotică?" se traduce automat în predicatul ține(laboratorul de robotică, S, P), predicat care unifică cu ține(laboratorul de robotică, sala 412, Popa Eugenia) prin atribuirile S = sala 412 și P = Popa Eugenia;
- **Managerul de dialog** care este bucla (infinită) de I/O pentru sistemul de dialog: așteaptă întrebări de la utilizator, cere informații suplimentare dacă este necesar și verbalizează informațiile solicitate de utilizator sau clasifică așteptările utilizatorului pentru sistemul de planificare al robotului (cel care decide ce acțiuni poate efectua Pepper).

Sistemul de dialog ROBIN-Dialog Este disponibil la https://gitlab.com/raduion/robindialog și este scris în Java 1.8. Fiecare clasă care implementează componente ale sistemului are o versiune abstractă (clasă sau interfață) care poate fi rescrisă dacă implementările existente nu sunt suficiente pentru un scop sau altul și, de asemenea, este comentată în stilul JavaDoc pentru a facilita utilizarea.

Pachetul ro.racai.robin.dialog conține clasele care implementează **managerul de dialog** și **algoritmul de unificare**[10]. Așa cum am mai menționat, din întrebarea utilizatorului, analizorul de limbaj natural extrage un predicat pe care algoritmul de unificare încearcă să-l „potrivească" cât mai bine cu un predicat din universul de discurs. Predicatele au un număr variabil de argumente iar fiecare argument are un tip. De exemplu, folosind scenariul de orientare a studenților în facultate, există argumente de tip „sală" sau de tip „curs". Pentru a facilita potrivirea, fiecare concept are o mulțime de sinonime[11] cum ar fi de exemplu „sală, cameră, încăpere" sau „curs, laborator, seminar". Algoritmul de unificare va putea potrivi un predicat cu un număr mai mic de argumente (sub-specificat) cu un predicat din microlume care are mai multe argumente, cu condiția ca variabilele legate (cele care sunt instanțiate) să fie identice și de același tip.

Pachetul ro.racai.robin.mw conține clasele care construiesc **universul de discurs al sistemului de dialog** dintr-o microlume. În implementarea actuală, o microlume se construiește automat dintr-un fișier .mw în care inginerul de cunoștințe descrie conceptele și predicatele care sunt „adevărate" (există) în microlumea respectivă.

Pachetul ro.racai.robin.nlp conține clasele **analizorului de interogări în limba română**, algoritm care extrage un predicat cu argumente parțial instanțiate din întrebarea utilizatorului. Acesta folosește platforma RELATE pentru a preprocesa întrebarea (segmentare lexicală, adnotare cu etichete morfo-sintactice, lematizare și analiza cu relații de dependență sintactică).

Un exemplu de fișier .mw se află în GitLab[12] și definește o microlume care facilitează orientarea unui student în facultate (în cazul nostru, în Facultatea de Automatică și Calculatoare a Universității POLITEHNICA din București). Inginerul de cunoștințe poate defini următoarele:

- **Concepte**: entități *despre care poate fi vorba* în dialogul dintre om și mașină;
- **Obiecte** referite de fiecare concept: instanțe ale fiecărui concept;
- **Obiecte de diverse tipuri** despre care poate fi vorba;
- **Predicate**: relații care se stabilesc între diverse instanțe de concepte sau obiecte de diverse tipuri și care au valoarea de adevăr „adevărat" în această microlume.

În raportul în extenso sunt exemplificate toate aceste entități inclusiv sunt indicate scheme noi de dialog.

5.7. Concluzii

Prototipul avansat al sistemului ROBIN-Dialog se află la https://gitlab.com/raduion/robindialog. Clasele sunt documentate (în limba engleză) cu comentarii în stilul JavaDoc. În stadiul actual, ROBIN-Dialog poate răspunde la întrebări factuale și poate continua conversația pe marginea subiectului început, cu vocabularul constrâns al microlumii active.

[10] https://gitlab.com/raduion/robindialog/blob/master/src/main/java/ro/racai/robin/dialog/RDUniverse.java
[11] Termen folosit aici într-un sens mai larg. Poate fi vorba și de hiperonime sau hiponime.
[12] https://gitlab.com/raduion/robindialog/blob/master/src/main/resources/precis.mw

spaCy și RASA

O soluție alternativă pentru implementarea dialogului om-robot în limba română a fost investigate de grupul de cercetare de la UPB. Ea se bazează pe instrumente open-source spaCy și RASA. Prezentăm mai jos această abordare.

spaCy

spaCy este o bibliotecă în Python gândită să ușureze munca dezvoltatorilor de aplicații în care aceștia au nevoie de procesare de limbaj natural. Este ușor de instalat, asemănător cu orice altă bibliotecă de Python, este ușor de folosit datorită documentației clare. Din punct de vedere al performanțelor măsurate pe probleme specifice domeniului de procesare a limbajului natural, spaCy obține rezultate extrem de bune. Un alt beneficiu al acestei biblioteci îl constituie interfața unificată pentru 51 de limbi, printre care se numără și engleza, franceza, româna și germana. Acest lucru simplifică munca dezvoltatorilor care nu sunt nevoiți să învețe diferite moduri de a lucra cu biblioteca în funcție de limbă. spaCy oferă modele preantrenate de dimensiuni diferite (mic, mare) pentru limbile cu un grad de utilizare mai mare cum ar fi engleza și franceza. Pentru celelalte limbi suportul este minimal și modelul trebuie antrenat de către programatorii care vor să-l folosească pentru limbile respective. În acest caz se află și limba română. După antrenarea pe limba română (v. amănunte în raportul detaliat) s-au obținut următoarele rezultate:

- Part-of-speech tag accuracy (Tags_acc):97.288%;
- Unlabeled Attachment Score (UAS): 88.589%;
- Labeled Attachment Score (LAS): 81.172%;
- Named entity accuracy - Precision (ents_p): 75.514%;
- Named entity accuracy – Recall (ents_r): 78.102%;
- Named entity accuracy – F score (ents_f): 76.786%.

Este util să menționăm că la adresa http://relate.racai.ro este disponibil un lant de prelucrări pentru limba română, gata antrenat cu mai multe facilități și performanțe mai bune decât spaCy. Acesul este liber pentru prelucrări mono-fișier și pe bază de credențiale (user&password) pentru prelucrări masive de texte.

RASA

Rasa (https://rasa.com/, 2019) este o soluție open source de unelte de învățare automată și procesare de limbaj natural, ce are scopul de facilita implementarea de agenți inteligenți care să fie capabili de a purta conversații fluente și coerente cu utilizatorii. Rasa este distribuit sub forma unei versiuni gratuite ce poate fi modificată dupa propriile nevoi și scopuri, dar se poate opta și pentru o versiune enterprise ce oferă capabilități sporite. Pentru a face posibilă integrarea bibliotecii RASA în limba română este de dorit utilizarea unui model spaCy pe limba română, lucrul dezvoltat în paralel în cadrul acestui proiect. De asemenea, este nevoie de un corpus de antrenare în care să fie adnotate atât intenții cât și entități. Pentru acest proiect a fost generat un corpus, manual, disponibil în anexă. Odată având aceste componente, pentru instalare este nevoie să se urmeze doar pașii de instalare prezenți pe site-ul RASA care explică integrarea unei noi limbi.

Obiectivul a fost integral realizat. Toate lucrările menționează cu multumiri, finanțarea cercetărilor de către proiectul ROBIN-Dialog. Site-ul proiectului ROBIN-Dialog a fost actualizat cu rapoartele integrale și lucrările științifice realizate.

Toate obiectivele asumate de proiectul component ROBIN-Dialog au fost îndeplinite complet.

6. Proiectul component ROBIN-Cloud

Parteneri ai proiectului ROBIN-Cloud

UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI (CO)

UNIVERSITATEA TEHNICĂ DIN CLUJ – NAPOCA

UNIVERSITATEA "DUNAREA DE JOS" DIN GALAȚI

6.1. Prezentare generală

***Obiectivele proiectului ROBIN-Cloudi*:** îl constituie crearea unei platforme suport pentru colectarea de date provenind de la senzorii unor sisteme suport pentru roboți (ex. IoT), oferirea de mecanisme de procesare și învățare combinând modele Cloud cu dispozitive aflate aproape de sursa de colectare, oferirea de biblioteci suport pentru procesare inteligentă / semantică a datelor, și în final dezvoltarea de servicii Web centrate spre agenții economici interesați de folosirea datelor stocate la nivelul platformei. *Obiective specifice sunt*:

- Proiectarea și realizarea unei platforme suport pentru colectarea și sistematizarea de date captate de la nivelul unor sisteme IoT și roboți, peste arhitectura IoT-A ARM;
- Proiectarea și realizarea unor mecanisme de procesare și învățare distribuită pe baza datelor de la senzori, în tehnologie hibridă Cloud / Edge Computing (componentele Complex Event Processing și Context Broker);
- Dezvoltarea unei biblioteci de algoritmi de procesare semantică a datelor pe platforme Cloud;
- Proiectarea și realizarea de tehnici, metode și algoritmi pentru căutarea și extragerea cunoștințelor din Depozitul de Date la nivelul platformei ROBIN-Cloud;
- Dezvoltarea în Cloud a unor componente de suport pentru Machine Learning (algoritmi deep-learning, deep-reinfocement learning, support vector machine, etc.) specifice pentru prelucrarea limbajului natural, mașini autonome, dar și pentru colaborarea roboților în mediul social.
- Planificare inteligentă și adaptivă prin crearea unui set de algoritmi pentru: procesarea în Cloud, cuplarea cu SLAM, planificarea rutelor mașinilor autonome, analiza limbajului natural, etc.
- Proiectarea, realizarea și testarea de instrumente și servicii web destinate agenților economici.

Relația Robin-Cloud cu celelalte proiecte componente

Suport pentru ROBIN-Social. Integrarea algoritmilor de detecție și recunoaștere a persoanelor și a activităților cu datele provenite de la senzori pentru utilizarea roboților în medii reale, afectate de zgomot și în prezența mai multor utilizatori.

- Deoarece Robin-Social necesită capabilități de stocare și procesare ridicate pentru atingerea tuturor obiectivelor propuse, este necesar distribuirea acestora într-o arhitectură Cloud-Fog-Edge. În funcție de timpul de răspuns asociat unei acțiuni și complexitatea de prelucrare a acesteia, resursele asociate vor fi alocate mai aproape de robot (nivel Edge) sau la nivelele superioare Fog, respectiv Cloud. Pe măsură ce se urcă în ierarhie, viteza de răspuns scade, dar crește complexitatea task-urilor care pot fi procesate.
- *Modulul de vedere computațională* este o componentă importantă a proiectului Robin-Social, permițând detecția spațiului în care se află robotul, obiectelor din jur dar și a fețelor asociate persoanelor din încăpere. Prin distribuirea modulului între cele trei nivele ale arhitecturii Cloud-Fog-Edge se poate optimiza acest proces. La nivel Edge/Fog se pot prelucra recunoașterea spațială respectiv identificarea obiectelor.
- *Componenta de vorbire* este o altă componentă importantă a proiectului Robin-Social, permițând o interacțiune vocală între platformă și utilizator. Deoarece este necesar ca robotul să poată da de înțeles utilizatorului ca a înțeles mesajul transmis de acesta, acest modul trebuie partajat între nivelul Edge și respectiv Fog. Mesajele mai des întâlnite și comenzile acestora vor fi stocate mai aproape de platformă și anume la nivel Edge. Celelalte mesaje vor fi stocate la nivel Fog.
- *Planificarea acțiunilor de navigare* presupune cartografierea mediului înconjurător. Deoarece conexiunea la rețea poate să afecteze comunicarea cu bazele de date de la nivelul Cloud, este necesar ca algoritmii SLAM să fie prelucrați la nivelele de mai jos ale arhitecturii Cloud-Fog-Edge. În cazul unei conexiuni bune, harta mediului poate fi descărcată de pe Cloud unde se stochează o hartă detaliată/îmbunătățită pe baza cartografierilor anterioare. În momentul descărcării hărții algoritmii SLAM pot să beneficieze de informații detaliate.

Suport pentru ROBIN-Car. Premisa: modelele necesare conducerii autonome sunt dependente de amplasamentul geografic, configurație senzori, condiții de drum (de ex. luminozitate). Propunere:

- Faza experimentală: Upload/download la cerere și gestiune versiuni de modele; Upload/download la cerere și gestiune test-runs per model; Rulare la cerere de algoritmi de analiză eficacitate model;

- Faza deployment: Download de versiune de model către vehicul; la cerere versiune specificată sau; update funcție de amplasament geografic, configurație senzori; Gestiune modele și vehicule gazdă;
- Alte servicii

Suport pentru ROBIN-Context. În cadrul proiectului ROBIN Context, arhitectura sistemului cuprinde întreaga gamă de senzori care monitorizează activitatea umană, dar și sistemul de tip cloud al cărui rol este de a stoca date (database module), de a procesa informații (processing module) și de a interacționa cu dispozitivele inteligente (network module). Luând în considerare aspecte precum evitarea supraîncărcării rețelei, reducerea cheltuielilor pentru conectarea la internet a tuturor senzorilor și altele, multitudinea de senzori ce monitorizează activitatea umană se conectează strict la un dispozitiv inteligent, spre exemplu un smartwatch. Smartwatch-ul are rol de gateway sau interfață între senzori și sistemul de tip cloud. Totodată aceasta este echipat la rândul său cu o serie de senzori (spre exemplu GPS, accelerometru, etc.). Datele furnizate de senzorii embedded, cat și cei externi sunt corelate, iar rezultatul este transmis către cloud așa cum este ilustrat in Figura 6.1.

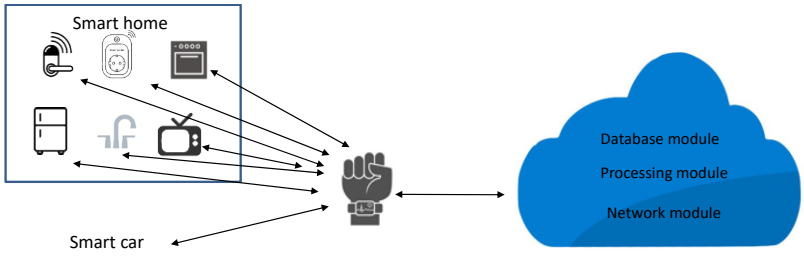


Figura 6.1. Suport pentru proiectul ROBIN-Context

Comunicația între senzori, smartwatch și cloud este criptată, pentru a se asigura confidențialitatea datelor personale. Agregarea datelor de la mai multe servere cu posibilități de corelare. Registru cu datele importante (date care nu afectează intimitatea).

Suport pentru ROBIN-Dialog. În cadrul proiectului ROBIN Dialog, arhitectura sistemului cuprinde atât roboții programați pentru interacțiune cu oamenii în micro-lumi, dar și sistemul de tip Cloud al cărui rol este de a stoca date (database module), de a procesa informații (processing module) și de a interacționa cu roboții din rețea (network module) așa cum este ilustrat in Figura 6.2.

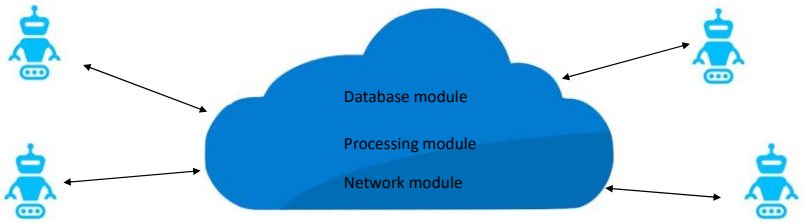


Figura 6.2. Suport pentru proiectul ROBIN-Dialog

În cazul în care sistemul de tip Cloud este reprezentat doar de o bază de date partajată de către toți roboții din rețea, aceștia furnizează către Cloud informații noi captate cu ajutorul senzorilor, dar totodată au și posibilitatea de a descărca noi informații. La acest punct, se poate evidenția un serviciu prototip, Sensing-as-a-Service, unde roboții cu capabilități reduse (senzori mai ieftini și mai puțin performanți, etc.) pot apela la servicii Cloud contra cost pentru a spori calitatea serviciilor oferite.

În cazul în care sistemul de tip Cloud este reprezentat ca bază de informații partajată, dar și ca tool de procesare, capabilitățile roboților pot fi reduse, ținând cont de serviciile oferite de Cloud. Astfel, toate procesările pe care roboții ar trebui să le realizeze se reduc strict la un schimb de mesaje cu sistemul Cloud. Comunicația între sistemul de tip Cloud și fiecare robot trebuie criptată cu o cheie de criptate unică, astfel încât să se asigure confidențialitatea și autenticitatea informațiilor transmise.

6.2. Plaforma Cloud/Edge

Drept rezultat al progreselor tehnologice realizate în ultimii ani, domeniul științei calculatoarelor a avut un progres remarcabil, atât din punct de vedere software, cât și din punct de vedere hardware. Această stare a generat posibilitatea abordărilor inovatoare în ceea ce privește proiectarea și implementarea sistemelor și

arhitecturilor distribuite. Având în vedere aspectele anterior menționate, propunem o platforma de tip Cloud-Edge bazată pe tehnologie software de ultimă oră de containerizare și orchestrare, precum și pe dispozitive moderne care posedă arhitecturi de sistem și capabilități eterogene. Scopul este acela de a utiliza avantajele oferite de sistemele de tip IoT, precum și de puterea de calcul, încrederea și ușurința accesului oferite de paradigma Cloud computing, acest deziderat fiind realizat prin crearea unei platforme eterogene, autoscalabile, care posedă un grad mare de disponibilitate. Pe baza unei analize de performanță, prezentăm comportamentul acestui tip de platformă, dar și cel al elementelor sale constituente, în diferite scenarii.

Platforma se bazează la fiecare nivel (Edge, Fog, Cloud) pe microservicii, create și gestionate cu ajutorul lui Docker și Kubernetes, ca instrumente principale de containerizare și planificare. În plus, soluția ilustrează modul în care sistemele încorporate precum Raspberry Pi 3 pot fi integrate într-un astfel de mediu și rulate alături de calculatoare obișnuite care sunt de obicei folosite în cloud computing. Propunem o arhitectură pe trei nivele care contribuie la crearea contextului pentru construirea, implementarea și utilizarea microserviciilor în mod eficient pentru a obține rezultatul dorit precum integrarea dispozitivelor Edge, care au capacități și structuri eterogene, cu medii Cloud într-o singură platformă, astfel încât avantajele ambelor sectoare pot fi utilizate în diferite cazuri relevante de utilizare. Structura arhitecturală prezentată permite diferențierea corespunzătoare între facilitățile pe care fiecare nivel trebuie să le aibă. De exemplu, generarea și colectarea de date este vizată să se realizeze în nivelul Edge și apoi trimise la nivelul Cloud pentru procesare intensivă de calcul prin intermediul nodurilor din nivelul fog, unde activități suplimentare, cum ar fi preprocesarea, memorarea în cache sau programarea sarcinilor în timp real pot avea loc.

În plus, arhitectura oferă consumatorului un grad ridicat de flexibilitate, în special atunci când utilizează microservicii, deoarece noile elemente pot fi ușor integrate în sistem și, în același timp, componentele vechi pot fi înlocuite, actualizate sau personalizate, fără a compromite funcționalitatea întregului sistem. Acest lucru se întâmplă atât în ceea ce privește software-ul, cât și în ceea ce privește hardware-ul, deoarece redundanța este oferită la toate nivelurile pentru a obține o disponibilitate ridicată a sistemului final.

Nivelul Edge constă în dispozitive care permit utilizarea avantajelor IoT în sistem. Pentru a realiza acest deziderat, propunem folosirea roboților sau altor dispozitive cu senzori, cum ar fi boxe inteligente, care au ca rol central interacțiunea cu mediul în care sunt plasate. Din această interacțiune apar diferite tipuri de date (multimedia, GIS, datele senzorilor, etc.) care pot fi captate, procesate, ambalate și trimise în nivelele superioare pentru a face obiectul altor activități pe care le vom detalia în continuare în teză. Cu toate acestea, în unele cazuri, această interacțiune poate reprezenta accentul principal al întregului sistem, nivelurile superioare oferind mijloacele de bună funcționare pentru întreaga platformă.

Pe fiecare nivel, câteva microservicii pot fi definite pentru a integra capabilitățile oferite de noile tipuri de tehnologii în sistem: **servicii audio** (cuprinde facilități precum înregistrarea și reluarea fluxurilor audio), **servicii de mutare** (prezente în cazul roboților sau a altor tipuri de sisteme similare care au capacitatea de a-și schimba poziția în mod automat), **servicii de date** (colectare și agregare date de la celelalte servicii disponibile pe dispozitiv și **servicii video**.

Nivelul Fog cuprinde dispozitive, numite și puncte de acces, care facilitează crearea unei punți de comunicare între nivelurile Edge și Cloud și, de asemenea, ajută la buna funcționare a activităților prezente în nivelul Edge. Entitatea pe care o propunem la acest nivel se numește **Supervizor**. Cel mai important rol al Supervizorului este supravegherea stării sistemului pentru fiecare nod din nivelul Edge, astfel încât procesele precum planificarea sarcinilor să poată fi realizate. Pentru Supervizor, a avea cea mai recentă imagine a nivelului pe care îl gestionează este crucial pentru atingerea scopurilor pentru care este utilizat.

Interacțiunea dintre un nod care face parte din nivelul Edge și un Supervizor este ilustrat în Figura 6.3.

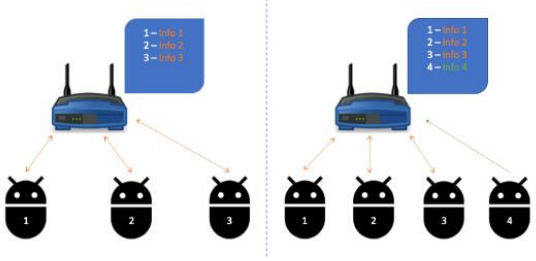


Figura 6.3. Interacțiunea Fog-Edge.

În primul rând, pe măsură ce un nou dispozitiv din nivelul Edge intră în sistem și în apropierea unui supervizor, acesta începe să trimită mesaje, la un interval de timp, cu detalii privind identitatea sa și starea

dispozitivului. În al doilea rând, supervizorul primește aceste mesaje și actualizează constant starea sistemului de care este conștient. Fiecare dintre aceste intrări au o stare valabilă pentru o anumită perioadă de timp, astfel nodurile care au scăzut sau nu sunt disponibile în acest moment sunt eliminate, temporar, din schema de comunicare pentru a păstra resurse precum lățimea de bandă sau timpul procesorului.

Nivelul Cloud oferă suportul principal pentru procesarea și stocarea avansată a datelor. Aici, datele generate în nivelul Edge sunt trimise pentru a face obiectul unor algoritmi costisitori de calcul, cum ar fi Algoritmi de Procesare Big Data sau Machine Learning.

6.3. Metrici de performanță

Analizăm modul în care hardware-ul folosit în sistem funcționează atunci când este folosit cu tehnologii precum Kubernetes și Docker, deoarece acestea nu sunt construite neapărat pentru un astfel de mediu. Metricile importante sunt reprezentate de încărcarea procesorului, memoria utilizată, încărcarea rețelei și numărul de Pod-uri alocate pe fiecare nod - în ceea ce privește cantitatea maximă de astfel de entități care pot fi expediate fiecărui membru al clusterului. În plus, este important să vedem dimensiunea finală a fiecărei imagini Docker pe care am creat-o, deoarece un alt obiectiv pe care l-am avut în acest context a fost să facem imaginile cât mai mici, având în construcția lor doar elementele necesare pentru a funcționa corect, astfel încât fiecare nod să le poată descărca mai rapid (implementări rapide) și să economisească cât mai mult spațiu de stocare spațiu posibil pentru alte nevoi prezente sau viitoare (de exemplu, Capsule de Date).

În primul rând, am măsurat starea fiecărui nod din sistem, Kubernetes rulând doar Pod-urile sale administrative (comunicare în rețea și programare), deoarece este important să se stabilească o bază de bază pentru sistem. Tabelul 6.1 ne oferă detalii despre starea inactivă.

Nod	Procesor	Memorie	Pod-uri alocate	Încărcare rețea (ieșire, intrare)
Kubernetes Master	13.37 %	643 MB	8/110	225.26, 35.45 kBit/s
Worker 1	8.02 %	318 MB	2/110	21.62, 60.25 kBit/s
Worker 2	4.65 %	294 MB	2/110	20.57, 51.98 kBit/s
Supervisor	5.61 %	283 MB	2/110	21.58, 52.17 kBit/s
Cloud	0.95 %	707 MB	3/110	19.79, 52.88 kBit/s

Tabel 6.3. Consumul de resurse, Kubernetes fiind inactiv

În al doilea rând, am derulat în cluster Implementările și Serviciile prezentate în ultimul capitol, păstrându-le într-o stare inactivă, astfel încât să putem vedea cu un grad ridicat de granularitate modul în care sistemul este afectat de noile entități. Tabelul 6.2 ne oferă detalii despre consumul de resurse cu microservicii implementate în stare de repaus.

Nod	Procesor	Memorie	Pod-uri alocate	Încărcare rețea (ieșire, intrare)
Kubernetes Master	13.37 %	643 MB	8/110	225.26, 35.45 kBit/s
Worker 1	8.02 %	318 MB	2/110	21.62, 60.25 kBit/s
Worker 2	4.65 %	294 MB	2/110	20.57, 51.98 kBit/s
Supervisor	5.61 %	283 MB	2/110	21.58, 52.17 kBit/s
Cloud	0.95 %	707 MB	3/110	19.79, 52.88 kBit/s

Tabel 6.4. Consumul de resurse cu microservicii implementate, starea fiind de repaus

Am măsurat cum se comportă sistemul în caz de utilizare normală care cuprinde etapele prezentate în secțiunea dedicată fluxurilor de comunicare. Cazul analizat include acțiunile de descărcare a sarcinilor efectuate de Supervizor, astfel încât să putem înțelege mai bine interacțiunea completă dintre nodurile sistemului. Tabelul 6.3 ne oferă detalii despre consumul de resurse cu microservicii implementate in stare activă.

Nod	Procesor	Memorie	Pod-uri alocate	Încărcare rețea (ieșire, intrare)
Kubernetes Master	13.37 %	643 MB	8/110	225.26, 35.45 kBit/s
Worker 1	8.02 %	318 MB	2/110	21.62, 60.25 kBit/s
Worker 2	4.65 %	294 MB	2/110	20.57, 51.98 kBit/s
Supervisor	5.61 %	283 MB	2/110	21.58, 52.17 kBit/s
Cloud	0.95 %	707 MB	3/110	19.79, 52.88 kBit/s

Tabel 6.5. Consumul de resurse cu microservicii implementate, starea fiind activă

6.4. Sistem de stocare descentralizat pentru Edge Computing

Soluția noastră este o aplicație de stocare distribuită, descentralizată, care va rula pe mai multe tipuri de noduri edge, cu sisteme de operare proprii, arhitecturi și sisteme de fișiere. Nodurile edge vor partaja fișiere de la diferiți senzori, iar utilizând un mediu de comunicare fără fir, va găzdui date fragmentate cu backup Cloud constant. Aplicația va acționa ca o platformă, oferind un API, pe care s-ar putea construi soluții de agregare și analize. Stocarea va respecta caracteristicile edge ale scalabilității, localizarea datelor și redundanța datelor

Localizarea datelor și suportul pentru mobilitatea utilizatorilor sunt principalii factori care influențează motivația edge computing. Dacă un client se deplasează de la o stație de bază în alta, sistemul de stocare trebuie să răspundă la schimbarea locației și să acționeze în mod transparent. În scenarii precum vehicule inteligente și orașe inteligente, clienții sunt în permanență în mișcare. Interacțiunea cu mobile edge computing ar putea fi îmbunătățită prin prezicerea vitezei și a direcției utilizatorilor.

Nodurile ar trebui să fie responsabile pentru descoperirea și aderarea la rețea. Avem un spațiu de stocare descentralizat pe care dorim să construim o topologie de la zero. Primul nod aderă la o rețea nepopulată și realizează o scanare completă pentru alte gazde, dar nu poate identifica pe nimeni, așa că rămâne la dispoziție. De acum poate răspunde și interoga fișierele operațiunilor PUT sau GET, dar este doar o mașină de stocare cu un singur nod care poate gestiona fișierele, deci nu există nimic disponibil în cadrul acesteia. Când cel de-al doilea nod intră în rețea, descoperă primul nod și decide că ar trebui să se asocieze cu el și să fie vecini. Când cel de-al treilea nod intră în rețea, acesta alege la întâmplare un nod dintre primii doi cu care ar trebui să se asocieze și așa mai departe. Astfel, alegând un nod aleatoriu, ne garantează că topologia în orice moment va fi un arbore care are rădăcina egală cu primul nod care a intrat în rețea. O ilustrare a acestui algoritm este reprezentată în Figura 6.4.

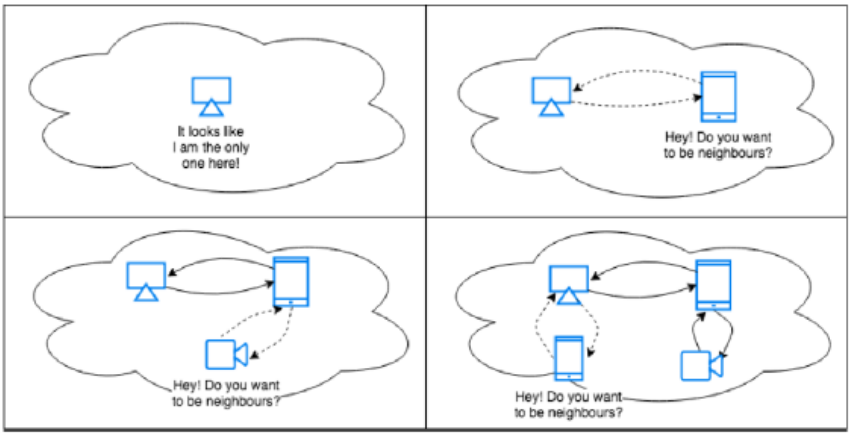


Figura 6.4. Algoritmul de descoperire.

Un serviciu de descoperire personalizat interoghează fiecare IP din interval pentru fiecare port dintr-un interval de porturi bine cunoscut. De exemplu, dacă IP-ul 192.168.1.138 a fost singurul care a rămas atunci când s-a realizat descoperirea cu ajutorul comenzii ping, atunci serviciul va porni cereri paralele pentru fiecare port între 8500 și 8700 și va verifica care dintre acestea sunt deschise. Pentru a face acest lucru, am introdus un interval de timp în care, dacă gazdele nu răspund, atunci se va presupune că un anumit set de date (adresă IP, număr de port) nu a fost disponibil în faza de descoperire. Acest serviciu de descoperire personalizat folosește ruta /alive a vecinului pentru a identifica dacă că este disponibil sau nu.

La descoperirea tuturor gazdele posibile din rețea, cererea /alive poate răspunde cu detalii despre resursele nodului, precum spațiul liber pe disc și câtă memorie RAM este liberă. Pe baza acestor informații, factorul decizional aleator poate favoriza vecinii cu resurse abundente. De asemenea, atunci când alege un vecin cu resurse disponibile, acesta poate rezerva un procent din spațiul liber și din memoria RAM pentru acel vecin specific și data viitoare îl raportează, va raporta o valoare mai mică, astfel încât să nu fie copleșit de toate nodurile din topologie.

Pentru a măsura capacitățile de echilibrare, am extras dimensiunea fiecărui nod după încărcarea unui fișier de 60 MB pe o rețea personalizată generată. Putem vedea că la început nodul care a gestionat cererea va avea toată încărcarea. Fragmentele vor începe să se propage în rețea având acest nod ca epicentru (a se vedea Figura 6.5).

Al doilea experiment a măsurat durata de preluare a unui fișier de 20MB din sistem cu 10 până la 50 de noduri și un factor de replică de 2, respectiv 4. Cu numărul tot mai mare de noduri, procesul de recuperare

devine mai lent, deoarece colectarea trebuie să traverseze mai multe noduri. Deși, dacă creștem numărul de replici, creștem șansele de descoperire a fișierului mai aproape de nodul sursă (a se vedea Figura 6.6).

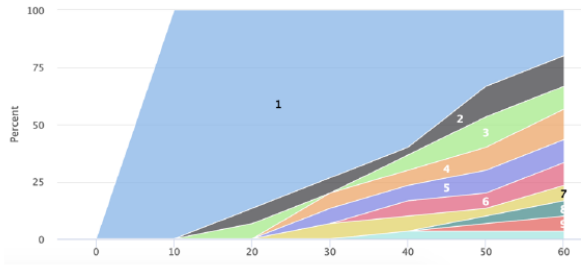


Figura 6.5. Evoluția stocării după plasarea unui fișier

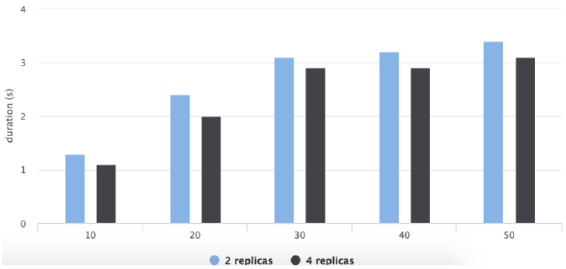


Figura 6.6. Timpul de recuperare influențat de factorul replică

Deoarece algoritmul de descoperire trebuie să caute în întreaga rețea, acesta nu se scalează foarte bine. Partea interesantă a acestei abordări este că, în primul rând, elimină orice gazdă din gama de IP-uri care nu este disponibilă la momentul scanării și având lista filtrată de IP-uri, știm sigur că va scana în continuare doar gazdele care ne interesează (cele disponibile). De asemenea, intervalul de timp trebuie ales foarte bine: dacă este ales prea scăzut, s-ar putea să nu descopere gazde care sunt disponibile, dar sunt prea lente ca răspuns; dacă este ales prea înalt, timpul de descoperire poate dura ceva și, prin urmare, operațiunile PUT și GET ar putea fi întârziate, de asemenea, deoarece nodul va fi ocupat să descopere vecini. Figura 6.7 prezintă modalitatea în care această diferență poate avea sens.

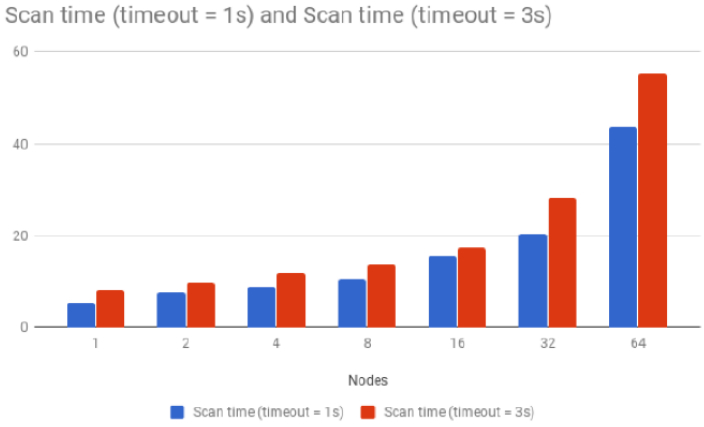


Figura 6.7. Timpul de scanare pentru diferite praguri temporale

6.5. Algoritmi de planificare și descărcare în sistemele Cloud-Edge

Procesul de alegere a unui nod în care trebuie descărcată o sarcină este un factor foarte important în obținerea unei performanțe mai bune pentru sistem. Un alt aspect demn de menționat este posibilitatea utilizării mai multor noduri. În acest fel, unele aplicații ar putea profita de o execuție paralelă pe mai multe dispozitive Fog, astfel încât timpul de răspuns ar putea fi chiar mai scurt decât orice altă abordare. În al doilea rând, descărcarea se poate face de la Cloud la Edge. Aceasta este o abordare foarte comună în care serviciul oferit de Cloud este fie replicat complet, fie partiționat pe dispozitive suficient de puternice din Edge. Rezultatul direct al acestei tehnici este o experiență mai bună a utilizatorului (datorită timpilor de răspuns mai scurti), obținută prin faptul că utilizatorii se conectează mai bine la centrele de date mai mici, la fel cum s-ar conecta la cloud.

Există numeroși algoritmi care iau în considerare diferite aspecte, cum ar fi puterea de calcul a dispozitivelor sau cerințele de procesare ale sarcinilor sau latența conexiunii de rețea și își propun să ofere soluții, cum ar fi descărcarea de sarcini numai în cloud, crearea de imagini clonate ale dispozitivelor care solicită putere de procesare mai aproape de nodurile edge, rezolvând problema ca o sub-problemă care optimizează consumul total de energie sau îmbunătățește timpul de răspuns. Deși aplicațiile lor au arhitecturi diferite decât cea prezentată, considerarea avantajelor și blocajelor multor implementări este un factor important în crearea unui model eficient de procesare Cloud-Edge. În primul rând, adunăm tipurile de mesaje care sunt transferate între dispozitive din nivelele adiacente (Edge-Fog și Fog-Cloud) în Figura 6-8.

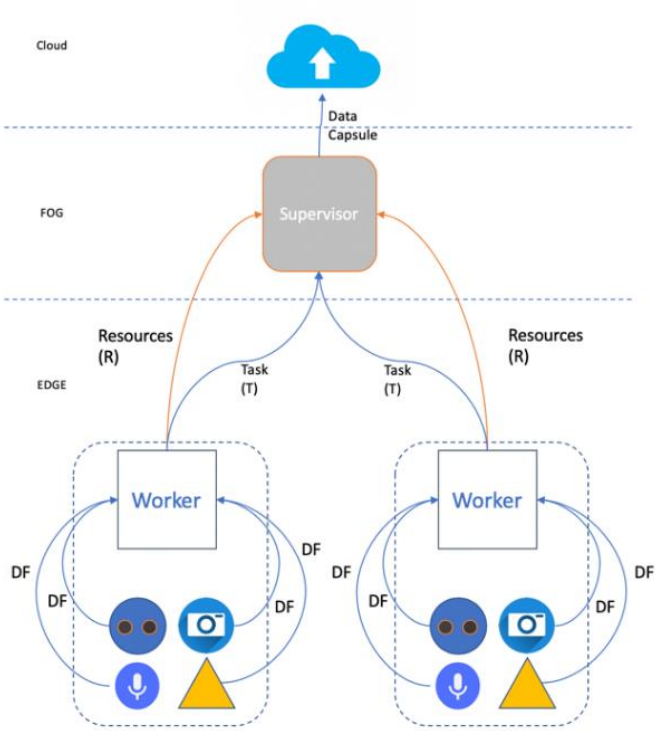


Figura 6-8. Flux de date

Atunci când un dispozitiv edge este inițializat, acesta trimite un *mesaj tip resursă* R către *supervizorul* S și apoi un mesaj de actualizare periodic. Apoi, acest dispozitiv recent activat, echipat cu senzor primește o comandă de înregistrare de la utilizator; în consecință, fiecare microserviciu disponibil pe dispozitiv începe să realizeze sarcina, ceea ce va avea ca rezultat un Fragment de Date creat de fiecare serviciu. După cum s-a descris anterior, aceste fragmente au aceleași id și marcă temporală, pentru a fi contopite într-o Capsulă de Date la un moment dat după finalizarea procesării.

7. Concluzii

Prezentul raport reprezintă descrierea activităților de cerectare și a rezultatelor obținute de consorțiul proiectului în cadrul etapei a 2-a a proiectului ROBIN. Este de remarcat faptul că realizările proiectului au cosntat atât în contribuții teoretice (studii ale dezvoltărilor în domeniu, noi arhitecturi, noi algoritmi sau algoritmi îmbunătățiți) iar fiecare propunere de arhitectură sau algoritm a fost validată print-o implementare, chiar și preliminară, așa cum se poate observa din evaluarea performanțelor atinse, din filmulețele demnstrative de pe youtube, sau din lucrările științifice publicate în conferințe și reviste de prestigiu în lumea științifică.

Este de asemenea de subliniat faptul că cele 5 proiecte componente reprezintă o viziune unitară a conceptului de roboți autonomi, inteligenți, fie că aceștia sunt utilizați pentrua sistenâă socială sau interacțiune cu clienții, fie că sunt utilizați pentru pilotaj automat. Proiectele componente ROBIN-Context și ROBIN-Cloud, pe lângă contribuțiile teoretice intrinsece, au rolul de a aduce și crea tehnologii moderne, inovatoare care să susțină și să eficientizeze realizarea roboților mai sus menționați.

Bibliografie

[1] H.L. Lohmeier and J.D Hoit, "Ventilator-supported communication: a survey of ventilator users". In: Journal of medical speech-language pathology, Vol. 11, pp. 61–72, March 2003.

[2] I. Marinescu, V. Mititelu and M. Mitrofan, "Polishing MONERO, the Morphologically and Medical Named Entities Annotated Corpus of Romanian". In: Proceedings of the 14th International Conference Linguistic Resources and Tools for Natural Language Processing, Cluj-Napoca, pp. 3-13, 18-20 November 2019.

[3] G. Huang, Z. Liu, L. Van Der Maaten and K.Q. Weinberger, "Densely Connected Convolutional Networks". In: CVPR, Vol. 1, pp. 3, 2017.

[4] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement" arXiv preprint arXiv:1804.02767, 2018.

[5] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.Y. Fu and A.C. Berg, "Ssd: Single shot multibox detector". In: European conference on computer vision, pp. 21-37, Springer, Cham, October 2016.

[6] T.Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, "Focal loss for dense object detection". In: Proceedings of the IEEE international conference on computer vision, pp. 2980-2988, 2017.

[7] S. Ren, K. He, R. Girshick and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks". In: Advances in neural information processing systems, pp. 91-99, 2015.

[8] D. Iancu, A. Sorici and A.M. Florea, "Object detection in autonomous driving - from large to small datasets". In: 11th International Conference on Electronics, Computers and Artificial Intelligence (ECAI 2019), June 2019.

[9] P. Isola, J-Y Zhu, T. Zhou, and A.A. Efros, "Image-to-image translation with conditional adversarial networks". In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1125-1134, 2017.

[10] I. Paraicu and M. Leordeanu, "Learning Navigation by Visual Localization and Trajectory Prediction". In: arXiv preprint arXiv:1910.02818 (2019), submitted to International Conference on Robotics and Automation (ICRA), 2020.

[11] A. Marcu, D. Costea, E. Slusanschi, and M. Leordeanu, "A multi-stage multi-task neural network for aerial scene interpretation and geolocalization". In: arXiv preprint arXiv:1804.01322, 2018.

[12] I. Croitoru, S.V Bogolin and M. Leordeanu, "Unsupervised Learning of Foreground Object Segmentation". In: International Journal of Computer Vision, Vol. 127, Iss. 9, September 2019.

[13] A. Kendall, M. Grimes and R. Cipolla, "Posenet: A convolutional networkfor real-time 6-dof camera relocalization". In: Proceedings of the IEEE international conference on computer vision, pp. 2938-2946, 2015.

[14] A. Kendall and R. Cipolla, "Geometric loss functions for camera poseregression with deep learning". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5974-5983, 2017.

[15] A. Nicolicioiu, I. Duta and M. Leordeanu, "Recurrent Space-time Graph Neural Networks". In: Advances in Neural Information Processing Systems, pp. 12818-12830, 2019.

[16] A. Naiden, V. Paunescu, G. Kim, B. Jeon and M. Leordeanu, "Shift R-CNN: Deep Monocular 3D Object Detection with Closed-Form Geometric Constraints". In: International Conference on Image Processing (ICPR), Taipei, 2019.

[17] C. Perera, A. Zaslavsky, P. Christen and D. Georgakopoulos, "Sensing as a service model for smart cities supported by internet of things". In: Transactions on emerging telecommunications technologies, Vol. 25, Iss. 1, pp.81-93, 2014.

[18] A. Sorici, A. Olaru and A.M. Florea, "Towards Enabling Internet-Scale Context-as-a-Service: A Position Paper". In: Companion Proceedings of the 2019 World Wide Web Conference, pp. 668-671, ACM, May 2019.

[19] A. Sorici, A. Olaru, S. Flonta, E. Szilard and A.M. Florea, "Towards Enabling Context-as-a-Service for Social Assistive Robotics". In: 2019 22nd International Conference on Control Systems and Computer Science (CSCS), pp. 228-235, IEEE, 2019.

[20] M. Trăscău, A. Sorici and A.M. Florea, "Detecting Activities of Daily Living Using the CONSERT Engine". In: International Symposium on Ambient Intelligence, pp. 94-102, Springer, Cham, June 2018.

[21] D.J. Cook, A.S. Crandall, B.L. Thomas and N.C. Krishnan "CASAS: A smart home in a box". In: Computer, Vol. 46, Iss. 7, pp.62-69, 2012.

[22] A. Sorici, G. Picard, O. Boissier and A.M. Florea, "Multi-agent based flexible deployment of context management in ambient intelligence applications". In: International Conference on Practical Applications of Agents and Multi-Agent Systems, pp. 225-239, Springer, Cham, 2015.

[23] E. Irimia, M. Mitrofan and V.B. Mititelu, "Evaluating the Wordnet and CoRoLa-based Word Embedding Vectors for Romanian as Resources in the Task of Microworlds Lexicon Expansion". In: Wordnet Conference, p. 176, 2019.

[24] T. Boros, S.D. Dumitrescu and V. Pais, “Tools and resources for Romanian text-to-speech and speech-to-text applications”. In: Proceedings of the International Conference on Human-Computer Interaction - RoCHI pp. 46-53, 2018.
[25] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, et al. In: “The HTK book”, Cambridge University Engineering Department 3, pp.175, 2002.
[26] A. Stan, F. Dinescu, C. Ţiple, Ş. Meza, B. Orza, M. Chirilă and M. Giurgiu, “The SWARA speech corpus: A large parallel Romanian read speech dataset”. In: 2017 International Conference on Speech Technology and Human-Computer Dialogue (SpeD), pp. 1-6, IEEE, 2017.
[27] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng and G. Chen, “Deep speech 2: End-to-end speech recognition in english and mandarin”. In: 33rd International Conference on Machine Learning, pp. 173–182, New York, NY, USA, JMLR: W&CP, Vol. 48, 2016.
[28] K. Heafield, “KenLM: Faster and smaller language model queries”. In: Proceedings of the sixth workshop on statistical machine translation, pp. 187-197, Association for Computational Linguistics, pp. 187–197, July 2011.
[29] V. Păiș, D. Tufiș and I. Radu, “Integration of Romanian NLP tools into the RELATE platform”. In: International Conference on Linguistic Resources and Tools for Natural Language Processing, November 2019.

Director de proiect
Prof. Adina Magda Florea