MINISTERUL EDUCAȚIEI NAȚIONALE

UEFISCDI

Raport științific și tehnic pentru proiectul ROBIN

ROBIN-Dialog

REZUMAT

În cursul anului doi al proiectului ROBIN-Dialog au fost urmărite și realizate toate obiectivele incluse în Agenda Comună și Planul de realizare pe anul 2019 și anume:

1) Transcrierea fonetică a cuvintelor din lexiconul validat
2) Alinierea cu semnalul vocal corespunzător;
3) crearea înregistrărilor vocale pentru cuvinte pentru care nu există înregistrări în corpus
4) Sistemele de antrenare ASR și TTS vor fi alimentate cu rezultatele activității precedente. Sistemele ASR si TTS vor fi testate și evaluate.
5) Implementarea și descrierea prototipului generic de sistem de dialog cooperant.

În cele ce urmează vor fi prezentate activitățile desfășurate pentru realizarea acestor obiective cu precizarea că toate sarcinile asumate au fost îndeplinite integral.

Capitolul 5.

Descrierea științifică și tehnică

Autori: Dan Tufiș, Radu Ion, Elena Irimia, Vasile Păiș, Maria Mitrofan (Carp), Verginica Mititelu, Eric Curea, Valentin Badea, George Cioroiu – Institutul de Cercetări pentru Inteligență Artificială ”Mihai Drăgănescu”

Mihai Dascălu, Ștefan Trăușan-Matu, Dragoș Corlătescu – Universitatea ”Politehnica” București

5.1 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 (Irimia et al., 2019), 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:

- suprapunerea intrărilor din lexiconul ROBIN peste două resurse disponibile: i.) – RoSyllabiDict (Barbu, 2018), un dicționar de silabificare și accent cu 525.534 forme flexionare corespunzătoare a 65.000 de leme; ii.) – MaRePhor (Toma et al., 2017), care conține transcrierea fonetică pentru 72.375 leme.
- generarea (pentru lemele din ROBIN care nu se găsesc în cele două resurse) acestor informații cu instrumentul Romanian TTS.

Lexiconul rezultat este o resursă cu 27559 intrări de 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[1].

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: vezi cazul hiat versus diftong, triftong de mai jos;

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

cuvinte la vocativ; cuvinte cu silabe care conțin mai mult de o vocală (o silabă poate conține o singură vocală și mai multe semivocale; silabele care conțin două dintre literele "a", „â", „î, "ă" (care nu pot fi niciodată semivocale) sunt întotdeauna greșite). În corectarea manuală a accentului, am acordat atenție specială cuvintelor cu două variante de accent (omonimii): substantive cu două leme posibile (exemplu: „fotograf/fotografie" sunt leme posibile pentru „fotografii"); 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").

În procesul de validare și corectare a transcrierilor fonetice ne-am concentrat pe situații pe care le-am identificat ca potențial problematice:

1. cuvinte care conțin două sau mai multe vocale: aici am implementat reguli speciale pentru transcrierea diftongilor, triftongilor și hiat-ului;
    - identificarea hiat-ului se bazează pe silabificare și depinde de corectitudinea acesteia: vocalele în hiat sunt vocale aflate de o parte și de alta a unor silabe, care nu sunt implicate la rândul lor în diftongi sau triftongi;
    - cei mai mulți diftongi pot fi transcriși fonetic prin reguli, deoarece sunt clasificați în ascendenți (prima literă se transcrie ca semivocală, a doua se transcrie ca vocală; în această categorie intră: „ea", „eo", „ia", „ie", „io", „oa", „ua", „uă", „uo") sau descendenți (prima literă se transcrie ca vocală, a doua ca semivocală; în această categorie intră „ai", „au", „ei", „eu", „ii", „oi", „ou", „ui", „ăi", „ău", „âi", „âu"); diftongul "iu" este excepția acestei clasificări deterministe, el putând fi atât descendent (de exemplu în "fiu") cât și ascendent (de exemplu în "iubit"), și necesitând dezambiguizare și corectare manuală;
    - triftongii pot clasificați determinist drept centrați (transcriși ca semivocală + vocală + semivocală, ex.: „eai", „eau", „iai", „iau", „iei", „oai", „ioi") sau ascendenți (formați din semivocală + semivocală + vocală, ex.: „ioa", „eoa", „uea").
2. cuvinte care conțin grupurile de sunete „ce", „ci", „ge", „gi", „che", „chi", „ghe", „ghi": au fost implementate următoarele reguli:

    I. pentru grupuri la final de cuvânt:

    a. atunci când toată silaba e reprezentată de unul dintre grupurile vizate, litera finală a grupului ("e"/"i") are valoare vocalică și se transcrie ca "e", respectiv „i"; exemplu: *trece [t r e tS e]*

    b. atunci când grupurile se află la final de cuvânt, dar nu formează singure silabă (alături de ele mai există, în aceeași silabă, cel puțin o vocală;), litera finală ("e"/"i") are valoare „zero" (nu se transcrie): i final asilabic este absorbit în articulația consoanei precedente; exemplu: *mici [m i tS]*

    II. pentru grupuri la final de silabă în interiorul cuvântului:

    a. indiferent dacă este urmat de semn vocalic sau de semn consonantic în următoarea silabă, litera finală ("e"/"i") are valoare vocalică și se transcrie "e"/„i": exemplu: *erbacee [e r b a tS e e]*

    III. pentru grupuri în interiorul silabei:

    a. atunci când grupul este urmat de consoană, litera finală ("e"/"i") are valoare vocalică și se transcrie ca e sau i; exemplu: *certa [tS e r t a]*

b. atunci când grupul este urmat de vocală neaccentuată și care nu formează silabă (este semivocală), litera finală (“e”/“i”) are valoare vocalică: în cazul acesta e sau i din grup se transcriu cu e, respectiv cu i, iar cealaltă vocală (cea neaccentuată care urmează) se transcrie ca semivocala corespunzătoare; (vezi regulile pentru diftongi pentru a identifica dacă litera care urmează grupul este vocală sau semivocală); exemple: *cercei [tS e r tS e j], cretaceu [k r e t a tS e w]*
c. atunci când grupul este urmat de o vocală care formează silabă, litera finală (“e”/“i”) nu are valoare vocalică (sau are valoare „zero”) și nu se transcrie (este absorbită în articulația consoanei precedente); exemple: *ceaţă [tS a ts @]ciumă [tS u m @], chiar [k_j a r], gheaţă [g_j a ts @];*

3. cuvinte care conțin consoana „x”, care se poate transcrie fie drept „ks”, fie drept „gz”: aceste cuvinte au fost corectate manual;
4. conțin cratimă în interiorul cuvântului (sunt cuvinte compuse): aceste cuvinte au fost corectate manual;
5. cuvinte care se încheie cu o silabă ce se termină în *"(orice sau nimic) vocală consoană i"*: aceasta este regula pentru i final asurzit, care se transcrie „i_0” (fac excepție grupurile ce/ci/ge/gi/che/chi/ghe/ghi, care respectă regula Ia); exemplu: *cercetări [tS e r tS e t @ r i_0]*;

Bibliografie

[Barbu, 2008] Barbu, Ana-Maria. "Romanian Lexical Data Bases: Inflected and Syllabic Forms Dictionaries." LREC (2008)

[Irimia et al., 2019] Irimia, E., Mitrofan, M., & Mititelu, V. B. (2019). 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).

[Stan et al., 2011] Stan, Adriana, Junichi Yamagishi, Simon King, and Matthew Aylett. „The Romanian Speech Synthesis (RSS) corpus: building a high quality HMM-based speech synthesis system using a high sampling rate” In Speech Communication vol.53 442-450. (2011)

[Toma et al., 2017] Toma, Ştefan-Adrian, et al. "MaRePhoR—An open access machine-readable phonetic dictionary for Romanian." 2017 International Conference on Speech Technology and Human-Computer Dialogue (SpeD). IEEE. (2017)

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).

5.2 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 (Boroș et al., 2018), constă în realizarea unui format intermediar de text fără semne

de punctuație sau alte elemente în afară de cuvintele pronunțate, urmat de alinierea propriu-zisă a cuvintelor cu sunetele înregistrate.

În scopul creării formatului intermediar a fost realizat un script care transformă toate literele românești în formatul cu litere mici și apoi păstrează doar simbolurile care corespund următoarei expresii regulate (1):

(1) "/[^a-zăîâșț]/"

Fișierele realizate în urma aplicării acestei proceduri sunt salvate cu extensia ”.lab”. Un exemplu este prezentat în figura următoare, împreună cu fișierul text original:

Te uiți cam câș, îmi reproșase clătinând capul, dar nu vă mai reținde cât pentru a vă spune o epigramă care mi s-a ițit acușica sub frunte.	*te uiți cam câș îmi reproșase clătinând capul dar nu vă mai reținde cât pentru a vă spune o epigramă care mi s a ițit acușica sub frunte*
a)	b)

Figura 1. Exemplu fișier .txt a) și fișierul corespunzător .lab b)

Pentru alinierea propriu-zisă , a fost utilizată unealta software HTK (Young et al., 2002), fiind produse la final fișiere .phs conținând momentul de start și momentul de sfârșit asociat fiecărei foneme prezentă în text. Pe baza acestora putându-se reconstitui momentele de început și sfârșit asociate fiecărui cuvânt din text. Un exemplu este prezentat în figura următoare:

0 7500000 pau pau
7500000 8000000 t te
8000000 9700000 e
9700000 10000000 u ui\310\233i
10000000 10800000 j
10800000 11500000 ts
11500000 11800000 ij
11800000 12200000 k cam
12200000 14400000 a
14400000 14700000 m
14700000 15300000 k c\303\242\310\231
15300000 15700000 a@
15700000 18200000 sh
18200000 20300000 sp
20300000 21100000 a@ \303\256mi
21100000 21900000 m
21900000 22200000 i
22200000 22500000 r repro\310\231ase
22500000 23100000 e
23100000 23800000 p
23800000 24100000 r
24100000 24700000 o
24700000 26000000 sh
26000000 26800000 a
26800000 28000000 s
28000000 28700000 e

28700000 29100000 k cl\304\203tin\303\242nd

Figura 2. Exemplu fișier aliniere .phs corespunzător exemplului anterior de fișiere .txt și .lab

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 acest scop, au fost adnotate cu ajutorul serviciului web TTL (Tufiș et al., 2008 ; Ion, 2007) obținându-se etichetele de tip parte de vorbire aferente fiecărui cuvânt. Un exemplu al interfeței de căutare este prezentat în Figura 3.

Figura 3. Interfața de căutare în componenta audio a corpusului CoRoLa

Căutarea se poate realiza pe baza unui cuvânt sau a unei leme, putându-se specifica opțional și tag-urile de tip part-of-speech dorite. De asemenea, se poate specifica și formatarea la afișare, utilizatorul putând cere afișarea doar a cuvintelor sau/și a lemelor, etichetelor asociate cuvintelor. Adițional se poate cere afișarea unui context de 5 cuvinte în jurul cuvântului căutat sau a întregii fraze în care a fost găsit cuvântul. În urma realizării unei interogări, cuvântul găsit poate fi ascultat în fiecare fișier audio (pe baza alinierilor realizate) sau întreaga frază poate fi ascultată.

Un exemplu de căutare, împreună cu rezultatele obținute este prezentat în Figura 4. În acest caz a fost realizată căutarea cuvântului ”*câș”* prezent în exemplul prezentat anterior. Au fost identificate 3 propoziții conținând acest cuvânt, provenind de la vorbitori diferiți.

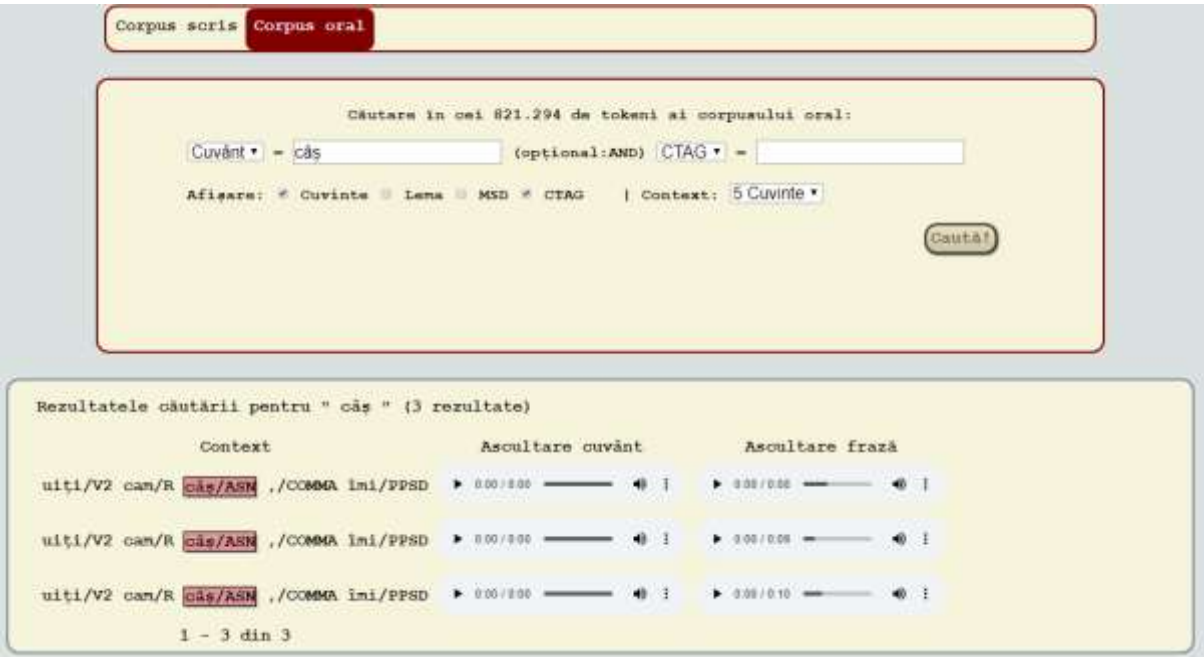

Figura 4. Exemplu căutare cuvânt ”câș”

Referințe

T. Boros, S.D. Dumitrescu, 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 2018*, pp 46-53.

Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, et al. 2002. The HTK book. Cambridge university engineering department 3 (2002), 175.

D. Tufiş, R. Ion, A. Ceauşu, D. Ştefănescu, “RACAI’s Linguistic Web Services”, in Proceedings of the 6th Language Resources and Evaluation Conference – LREC’08, Marrakech, Morocco, 2008.

R. Ion, “Word Sense Disambiguation Methods Applied to English and Romanian” (in Romanian). PhD Thesis, Romanian Academy, 2007.

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.

5.3 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 înregistrari vocale in corpusul CoRoLa. Din corpusul textual s-au extras propoziții ce conțineau cuvintele respective și s-au înregistrat 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 (v. secțiunea 5.1.2). 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.

5.4 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.

La modul general, calitatea vocii este bună, se poate înțelege 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%

Descrierea modelului.
ASR-ul are scopul de a transforma vorbirea de orice fel in text. Asta inseamna sa faca identificarea celei mai probabile secvente pentru semnalul vocal. Pentru a gasi cea mai probabila transcriere trebuie calculata probabilitatea semnalului stiind secventa de cuvinte care ii corespunde si apoi trebuie calculata probabilitatea unei secvente de cuvinte . Prima se obtine din modelul acustic iar a doua din modelul de limba.

Sunt 2 etape.:
Antrenare in care se face o modelare acustica si modelare lingvistica. Practic este nevoie de o resursa mare de inregistrari transcrise si de un dictionar mare de cuvinte posibile. In aceasta etapa sistemul invata corespondenta dintre vorbire si text.

- **Modelul acustic** este antrenat din inregistrarile transcrise. Se parametrizeaza mesajul vocal (se imparte semnalul vocal in ferestre fixe 25-50 ms , se accentueaza frecventele inalte pentru ca energia scade odata cu frecventa si se aplica un filtru

trece jos. se trece in domeniul frecventa prin fft pentru ca practic sunetul este o rezultanta dintre aer si raspunsul in timp a corzilor vocale (daca privim ca un filtru). Ce ne intereseaza la recunoasterea vocala este raspunsul filtrului fara semnalul de intrare adica aerul din plamani. pentru a putea separa aceste componente se trece in domeniul frecventa se aplica un filtru logaritmic si se ajunge la o operatie liniara de tip suma in care se pot separa foarte usor cei doi termeni. Apoi se aplica un banc de filtre MEL care practic grupeaza parametrii vocali in jurul frecventelor mai joase pentru a imita urechia umana. Si apoi se trece din nou in domeniul timp prin o transformata cosinus discreta pentru a reduce dimensionalitatea.Asa se obtin parametrii mel-cepstrali (MFCC)) si se face o corespondenta intre valorile vectorilor acustici si fonemul (litera) pe care o reprezinta. Asa se ajunge la o interpretare statistica.

- **Modelul de limba** este antrenat cu ajutorul unei baze mari de texte pentru a invata cum se succed cuvintele intr-o fraza. Practic invata ca dupa cuvantul "pentru" cel mai probabil cuvant este "ca".
- **Modelul fonetic** este componenta care "lipeste" modelul de limba si modelul acustic. Practic pentru fiecare litera face transcrierea in fonema. Este un dictionar fonetic.

Decodarea reprezinta folosirea modelelor pentru a putea estima vorbirea.
Exista decodare online si offline.

- Cea online reprezinta decodarea in timp real pe segmente prestabilite din semnalul vocal.
- Cea offline se foloseste cand intreg semnalul vocal este deja inregistrat.. Pentru fraze scurte sau comenzi se pot folosi ambele.

Pentru dialog cu fraze mai complexe sau servicii speech to text(grafiere, dictare, etc) cea online este preferata. In demo-ul mentionat se foloseste decodare offline.

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

Predictie:

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 (Stan et al., 2017) 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 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% pentru antrenare, 20% validare și 20% pentru testare.

Cel mai mic dintre modelele inițiale de la DeepSpeech a fost antrenat folosind peste 300 de ore de înregistrări. Datorită faptului că SWARA conține doar 21 de ore, modelul a fost micșorat pentru a fi mai adecvat pentru corpusul în limba română. Experimentele au fost efectuate modificând dimensiunea celulei din fiecare dintre cele 5 straturi, valorile încercate fiind 512, 1024 și 2048. De asemenea, numărul de epoci de antrenare a fost fixat la 20 pentru fiecare variantă de rețea. Deși au fost încercate între 15 și 20 de epoci cu dimensiuni diferite ale straturilor, acuratețea nu varia semnificativ. Rata de învățare a fost și ea variată de la 0.0001 la 0.001, cele mai bune rezultate fiind obținute cu valori mai mici.

Mai mult decât atât, alfabetul a fost actualizat cu diacriticele din limba română („ă”, „ș”, „ț” etc.), prezente în toate textele de intrare. În urma analizei primelor antrenări, s-a observat că diacriticele nu erau recunoscute, ci doar literele corespunzătoare lor. Optimizarea recomandată pentru această problemă era introducerea unor date specifice limbii, cum ar fi, un alfabet actualizat, un model de limbă binar compilat cu KenLM (Heafield, 2011), precum și un model bazat pe trie pentru îmbunătățirea vitezei de recunoaștere. În secțiunile de rezultate și discuții sunt prezentate diferențele între antrenarea modelului predefinit, comparat cu adăugarea modelul de limbă.

Din nefericire, niciun set de date cu zgomot de fundal nu este disponibil pentru a antrena rețeaua să recunoască înregistrări mai apropiate de condițiile reale, precum sunt antrenate modelele pe engleză. Cum DeepSpeech folosește Tensorflow pentru definirea modelului, antrenarea s-a făcut direct cu suport GPU.

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, 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.

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

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

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.

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

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"

Discuții

O provocare majoră întâmpinată în timpul antrenării a fost micșorarea modelului și căutarea unor hiper-parametri potriviți pentru rețea, având în vedere că setul de date SWARA este de mai mult de 10 ori mai mic decât corpusul de engleză pe care a fost antrenat modelul original. Rezultatele arată că rețeaua încă nu reușește să recunoască cuvinte complete sau entități cu nume, deși am modificat alfabetul și am folosit un model pentru limba română. Acest lucru arată că modelul de recunoaștere a vorbirii are nevoie de un set de date mai mare pentru antrenare. De asemenea, s-a încercat reducerea numărului de straturi ale rețelei pentru a fi mai potrivit pentru setul de date folosit, dar WER nu s-a îmbunătățit.

O altă abordare încercată a fost folosirea unui model de limbă antrenat specific pe fiecare partiție din setul de date. De exemplu, Cucu, Besacier, Burileanu, and Buzo (2011) au raportat un WER între 40% și 20%, în funcție de modelul de limbă folosit, antrenat pe un corpus de 54 de ore și 17 vorbitori. Cu toate acestea, folosind această metodă s-a observat o variație mare în timpul antrenării, care nu ajută rețeaua DeepSpeech. Din această cauză, cea

mai bună abordare este folosirea unui model de limbă general și actualizarea sa continuă cu seturi de date adiționale.

Concluzii și dezvoltări ulterioare

În cadrul acestui raport au fost descriși pașii parcurși pentru antrenarea unei rețele DeepSpeech pe un corpus mult mai mic, în încercarea de a depăși nevoia unor modele acustice, dicționare fonetice și modele de limbă pentru corpusul SWARA. Cu toate că au fost încercați hiper-parametri diferiți pentru rețeaua neurală, problema dimensiunii corpusului nu a putut fi rezolvată.

Există mai multe soluții potențiale pentru a obține rezultate mai bune și pentru a construi un model stabil de recunoaștere a vorbirii în limba română folosind arhitectura DeepSpeech. Una dintre ele ar fi combinarea intrării audio cu parametrii unui model specific de limbă, fie antrenat pe transcrierea propozițiilor, fie folosirea unui model general. Această optimizare s-a dovedit, în experimentele noastre, a fi o soluție potrivită pentru reducerea rezultatelor pozitive false. Cu toate acestea, per total, această soluție nu a avut un impact major asupra acurateței.

O altă soluție este folosirea mai multor colecții de înregistrări diferite pentru antrenarea modelului DeepSpeech cu mai multe date. Mai mult decât atât, înregistrări audio conținând zgomot de fundal și mai mulți vorbitori ar putea ajuta rețeaua să învețe aceste caracteristici și pentru limba română. Cu toate acestea, problema nu este una simplă, deoarece seturile de date de recunoaștere a vorbirii necesită curățarea și alinierea transcrierilor, care sunt greu de obținut pentru un volum considerabil mai mare de date.

O ultimă alternativă ar fi antrenarea pe SWARA pornind de la un model pre-antrenat pentru engleză, cu rezultate bune. Cu toate că există puține asemănări între cele două limbi (română și engleză), ar fi interesant de încercat îmbogățirea modelului pe engleză cu date în limba română. Mai mult, aceeași metodă ar putea fi folosită pentru îmbogățirea modelului de limbă și evaluarea acurateței sistemului pe corpusul SWARA.

Referințe

[Amodei et al., 2016] Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., & Chen, G. (2016). Deep speech 2: End-to-end speech recognition in english and mandarin. In *33rd Int. Conf. on Machine Learning* (pp. 173–182). New York, NY, USA: JMLR: W&CP volume 48.

[Cucu et al., 2011] Cucu, H., Besacier, L., Burileanu, C., & Buzo, A. (2011). Enhancing automatic speech recognition for Romanian by using machine translated and web-based text corpora. *SPECOM 2011*, 81-88.

[Hannun et al., 2014] Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., & Coates, A. (2014). Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

[Heafield, 2011] Heafield K. KenLM: Faster and smaller language model queries. In *Proceedings of the sixth workshop on statistical machine translation* (pp. 187-197): Association for Computational Linguistics.

[Juang and Rabiner, 1991] Juang, B. H., & Rabiner, L. R. Hidden Markov models for speech recognition. *Technometrics, 33*(3), 251-272.

[Jurafsky and Martin, 2008] Jurafsky, D., & Martin, J. H. (2008). *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics* (2nd ed.). London: Pearson Prentice Hall.

[Lamere, 2003] Lamere, P., Kwok, P., Walker, W., Gouvea, E., Singh, R., Raj, B., & Wolf, P. Design of the CMU Sphinx-4 decoder. In *Eighth European Conference on Speech Communication and Technology*.

[Lane and Tranel, 1971] Lane, H., & Tranel, B. The Lombard sign and the role of hearing in speech. *Journal of Speech and Hearing Research, 14*(4), 677–709.

[Rao et al., 2015] Rao, K., Peng, F., Sak, H., & Beaufays, F. (2015). Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 4225-4229): IEEE.

[Stan et al., 2017] Stan, A., Dinescu, F., Ţiple, C., Meza, Ş., Orza, B., Chirilă, M., & Giurgiu, M. (2017). 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.

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

5.5 Implementarea prototipului generic de sistem de dialog cooperant, configurarea pentru robot asistiv

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. De exemplu, într-o microlume în care robotul trebuie să asiste vizitatorii în a se orienta într-o clădire (de ex. în facultate) și de a le oferi acestora informații despre evenimentele care au loc în diverse săli (de ex. cursuri sau laboratoare), următorul dialog este plauzibil între un student și Pepper:

- **Student**: În ce sală se ține laboratorul de robotică?
- **Pepper**: Cu cine aveți laboratorul: cu d-na Popa sau cu d-l Georgescu?
    - **Student**: Nu știu.
        - **Pepper**: În ce grupă sunteți?
            - **Student**: În grupa 3C.
            - **Pepper**: Laboratorul dvs. se desfășoară în sala 412.

Astfel, microlumea va conține, de exemplu, mulțimea de săli pe care robotul le „cunoaște”, ce cursuri sau laboratoare ține fiecare profesor în fiecare sală, etc. Pe lângă aceste lucruri, microlumea va conține și restricțiile de limbaj folosit astfel încât Pepper să poată recunoaște cererile care i se adresează.

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 (Păiș et al., 2019) 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).

Exemple de modelare a dialogului om-mașină în microlumi, în literatura de specialitate

Fluxul de lucru „tradițional” în **sistemele de dialog care operează în microlumi** (engl. „task oriented dialogue systems”) este (Chen et al., 2017):

1. „Înțelegerea” cererii utilizatorului care înseamnă transformarea algoritmică a întrebării formulate în limbaj natural într-o reprezentare de tip predicat;
2. Determinarea stării în care se află dialogul, privit ca un automat finit care este deja definit în microlumea dată;
3. Învățarea stării în care se trece din starea curentă dacă există mulțimi de dialoguri deja purtate de utilizatori și mașină (nu e cazul proiectului ROBIN);
4. Verbalizarea stării curente prin generarea unei fraze în limbaj natural care să o reflecte.

Înțelegerea cererii (întrebării) utilizatorului se face prin identificarea de „locașuri” (engl. „slot”), anume acele grupuri nominale din întrebare care se regăsesc în microlumea sistemului de dialog iar sarcina sistemului de dialog ar fi să „umple” acele locașuri care nu conțin informație cu informația relevantă. Cele mai multe sisteme de dialog au acces la mulțimi de dialoguri care au fost purtate de oameni în microlumea respectivă (de ex. discuții într-un magazin de electronice) și folosesc rețele neuronale complexe (de ex. convoluționale și/sau recurente) pentru a deduce automat locașurile de interes pentru microlumea dată.

Starea în care se află sistemul de dialog se referă la identificarea locașurilor care trebuie umplute cu informație, la momentul conversației. Lucrarea citată descrie sisteme de învățare automată care realizează și acest lucru, în contextul conversației, pe baza corpusurilor de dialog existente.

Smith et al. (1995) descriu un sistem de dialog cu microlumi, bazat pe limbajul de programare logică Prolog, care asistă tehnicienii electricieni la reparații de echipamente. Este similar cu sistemul nostru în ce privește noțiunea de unificare în Prolog. Acest sistem poate susține dialoguri de următorul tip:

- Calculator: Rotește potențiometrul la 0.
- Tehnician: În regulă.
- Calculator: Câți volți sunt între conectorul 1 și 2?

- Tehnician: Nu știu.
    - Calculator: Localizează voltmetrul.
    - Tehnician: L-am găsit.
    - Calculator: Conectează un fir între borna neagră a voltmetrului și conectorul 1.
    - Tehnician: Gata.
    - Calculator: Conectează un fir între borna roșie a voltmetrului și conectorul 2.
    - Tehnician: Gata.
- Calculator: Câți volți sunt între conectorul 1 și 2?
- Tehnician: 12V.

Pentru a putea susține dialoguri de tipul exemplificat mai sus, pentru fiecare echipament, clauza de „reparație efectuată” depinde de alte clauze care trebuie satisfăcute pe rând. De exemplu, pentru determinarea poziției unui comutator, o regulă de inferență poate arăta așa:

```
observeposition(sw1, X) <- find(sw1), reportposition(sw1, X)
```

în care poziția X raportată (reportposition(sw1, X)) aparține comutatorului care trebuie găsit mai întâi (find(sw1)).

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**[2] și **algoritmul de unificare**[3]. 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[4] 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).

Descrierea fișierului de definire a unei microlumi

[2] https://gitlab.com/raduion/robindialog/blob/master/src/main/java/ro/racai/robin/dialog/RDManager.java
[3] https://gitlab.com/raduion/robindialog/blob/master/src/main/java/ro/racai/robin/dialog/RDUniverse.java
[4] Termen folosit aici într-un sens mai larg. Poate fi vorba și de hiperonime sau hiponime.

Un exemplu de fișier .mw se află în GitLab[5] 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.

Conceptele sunt definite astfel: CONCEPT sală, laborator, cameră -> LOCATION, unde cuvântul cheie CONCEPT introduce o listă de sinonime din care primul reprezintă *denumirea standard* a conceptului, adică o cheie a conceptului în sistem. Cuvântul care urmează după simbolul săgeată este tipul conceptului: fiecare concept împreună cu instanțierile sale are un tip iar identificarea predicatelor similare se face (și) după tipurile argumentelor.

Pentru fiecare concept, o serie de instanțieri se pot defini astfel: REFERENCE sală sala de consiliu = S3, unde cuvântul cheie REFERENCE introduce, pentru o denumire standard de concept, o descriere în limbaj natural a unui obiect din universul de discurs care este în extensiunea conceptului. După simbolul egal urmează un alias (o prescurtare) al descrierii care va fi folosit în definirea predicatelor (pentru a ușura scrierea acestora).

Microlumea poate conține și obiecte de diverse tipuri despre care poate fi vorba. De exemplu, putem defini persoane cu PERSON Adriana Vlad = P1 sau momente de tip cu TIME marți, 8:00 = T1. Dacă se dorește adăugarea unui tip nou, enumerarea ro.racai.robin.dialog.CType trebuie actualizată cu tipul nou. Mecanismul de alias este identic cu cel al definirii instanțelor.

Predicatele acceptă un număr variabil de argumente cu tipuri precizate și reprezintă ce „există" (sau are valoarea de adevăr „adevărat") în microlumea definită. Scopul managerului de dialog este să găsească cel mai apropiat predicat definit în fișierul .mw de cel extras din interogarea în limba română primită la intrare și să răspundă cu valoarea/valorile variabilelor unificate în procesul de potrivire. Ordinea argumentelor nu este importantă în procesul de potrivire. Un predicat se introduce astfel: PREDICATE ține, desfășura -> SAY_SOMETHING, unde cuvântul cheie PREDICATE introduce o listă de sinonime în care primul element este denumirea standard a predicatului (similar cu definiția conceptelor prezentată mai sus). După simbolul săgeată urmează un identificator de comportament al robotului care va fi folosit de algoritmul de planificare al robotului (extern sistemului de dialog). Dacă se adaugă un comportament nou, acesta trebuie adăugat și în enumerarea ro.racai.robin.dialog.UIntentType (engl. „user intent type"). Un predicat cu valoarea de adevăr „adevărat" se definește ca: TRUE ține C1 S1 T1, unde cuvântul cheie TRUE arată că predicatul este adevărat și este urmat de numele standard al predicatului. Argumentele sunt precizate cu aliasurile conceptelor instanțiate definite anterior.

Un exemplu de dialog

Să presupunem că în microlumea cu care operăm, avem următoarele definiții:

CONCEPT sală, cameră -> LOCATION
CONCEPT curs, materie, seminar, laborator -> WORD
CONCEPT unde -> LOCATION

[5] https://gitlab.com/raduion/robindialog/blob/master/src/main/resources/precis.mw

REFERENCE sală sala 209 = S1
REFERENCE curs laboratorul de informatică = C1
TIME marți, ora 8:00 = T1
PREDICATE ține, desfășura -> SAY_SOMETHING
TRUE ține C1 S1 T1

Sistemul de dialog primește următoarea întrebare: „Unde se desfășoară laboratorul de informatică?" În acest caz, analizorul de limbaj, după ce analizează sintactic întrebarea, extrage predicatul „desfășura(X/LOCATION, laboratorul de informatică/WORD)". Algoritmul de potrivire identifică predicatul „ține(C1, S1, T1)" ca fiind cea mai bună potrivire din universul de discurs pentru că:

- „(a) ține" este sinonim cu „(a se) desfășura";
- „laboratorul de informatică" este o instanțiere a conceptului „curs" pentru că „laborator" și „curs" sunt sinonime; aliasul acestei instanțieri este C1;
- predicatul „ține" are un argument de tip LOCATION (aliasul S1) care unifică cu variabila nelegată X.

În urma acestei deducții, sistemul de dialog poate răspunde (X = S1 = 209): „sala 209". Dacă imediat după acest răspuns, primește întrebarea „Și când se ține?", pentru că are în memorie ultima potrivire și pentru că observă că „ține(C1, S1, T1)" are un argument de tip TIME, răspunde imediat cu (Y = T1 = marți, ora 8:00): „marți, ora 8:00".

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. Versiunile viitoare vor avea în vedere extensii pentru:

- rezolvarea ambiguității întrebării prin clarificarea intenției utilizatorului. De exemplu, la întrebarea de deschidere a conversației (foarte vagă) „Cine ține cursul?", ar trebui să răspundă cu „Ce curs vă interesează?"/„La cine vă referiți?" în cazul în care mai mult de un predicat din universul de discurs se potrivește întrebării primite;
- detectarea întrebărilor de tip „adevărat/fals": „Prof. Ciurescu ține cursul de programare în sala 105?" la care ar trebui să poată răspunde cu „Da/Nu/Nu știu.";
- facilitățile de ASR (engl. „Automatic Speech Recognition") și TTS (engl. „Text To Speech") în ROBIN-Dialog astfel încât să putem rosti răspunsurile și să putem recunoaște forma textuală a întrebării din rostirea ei.

Referințe bibliografice

[Chen et al., 2017] Hongshen Chen, Xiaorui Liu, Dawei Yin și Jiliang Tang. A Survey on Dialogue Systems: Recent Advances and New Frontiers. În CoRR abs/1711.01731. http://arxiv.org/abs/1711.01731

[Păiș et al., 2019] Păiș Vasile, Tufiș Dan și Ion Radu. (2019). Integration of Romanian NLP tools into the RELATE platform. În International Conference on Linguistic Resources and Tools for Natural Language Processing. Noiembrie 2019.

[Smith et al., 1995] Ronnie W. Smith, D. Richard Hipp și Alan W. Biermann. (1995). An Architecture for Voice Dialog Systems Based on Prolog-Style Theorem Proving. Computational Linguistics, Volume 21, Number 3, September 1995

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.

În Figura 1 este prezentată arhitectura generală pentru spaCy care conține toate operațiile de procesare de text care vor fi detaliate în secțiunile următoare. Se poate observa interfața unică pentru limbi menționată anterior. Operațiile de extragere a cuvintelor sau grupărilor de cuvinte („token”), determinarea părților de vorbire, stabilirea arborelui de dependențe și determinarea entităților dintr-un text, sunt acțiunile principale pe care biblioteca spaCy le face în momentul în care primește un text. Mai departe, spaCy structurează informațiile într-o ierarhie de clase: document – propoziție – token (care poate fi și o grupare de cuvinte) – cuvânt. Pe baza acestei ierarhi se pot executa operații mai complexe de extragere de sens din textul dat la intrare.

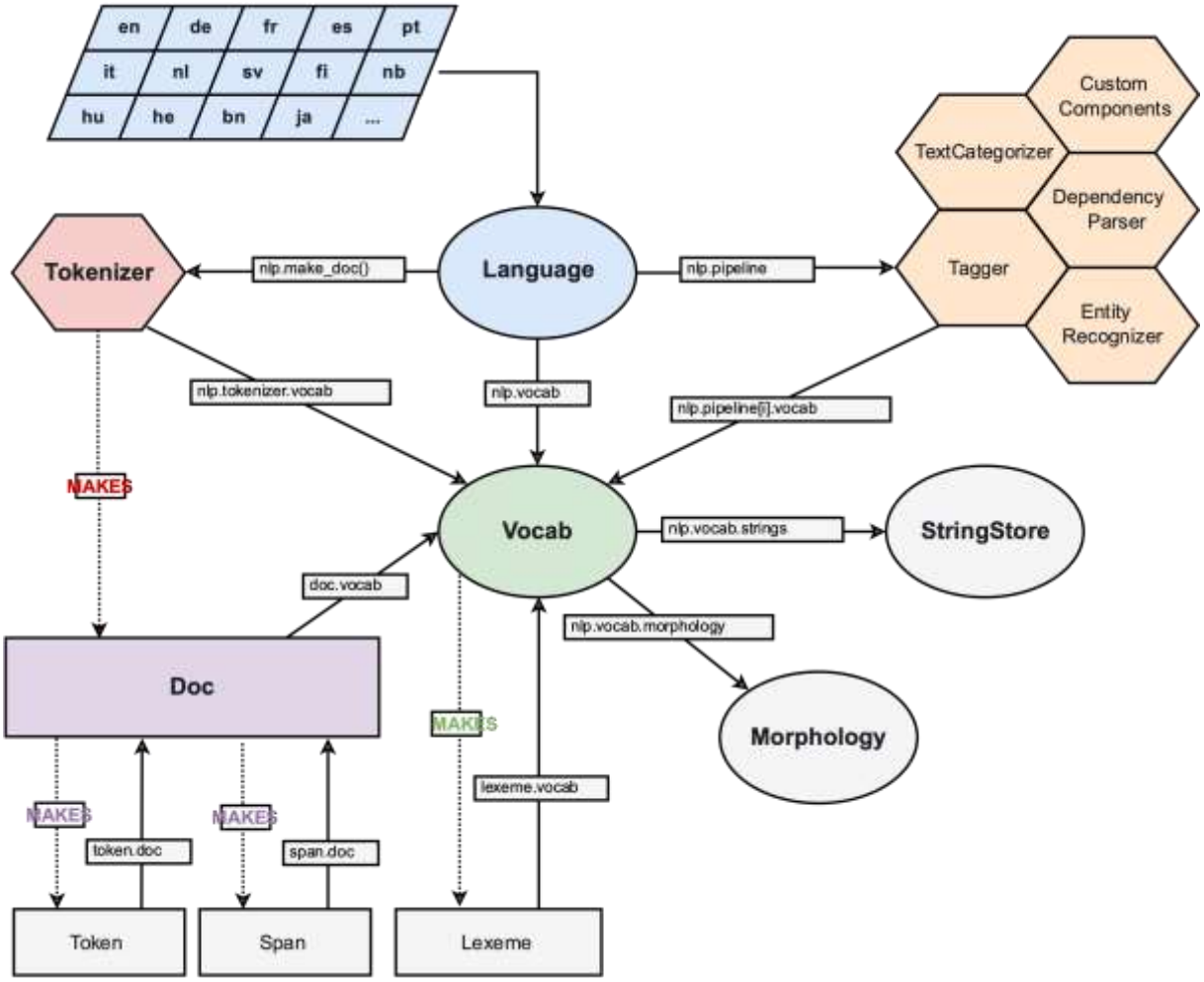

Figura 1. Arhitectura spaCy[sursa: https://spacy.io].

Prelucrări preliminare ale unui text

Prelucrarea complexă a unui text presupune anumite operații preliminare, care fac parte dintr-un așa numit „pipeline” („conductă”). După cum se poate observa în Figura 2, rezultatul fiecărei etape reprezintă intrarea etapei următoare.

Un text neprelucrat trebuie trecut printr-un „pipeline” („conductă”) înainte de a putea fi folosit în metode mai complexe. Etapele acestea preliminare sunt prezentate în Figura 2 alături de ordinea în care acestea sunt efectuate:

Figura 2. Secvența („pipeline”) de pre-procesare a textului.

Împărțire în cuvinte (Tokenizare)

Pentru partea de tokenizare trebuie identificate token-urile corecte. Cea mai simplă și ușor de implementat variantă ar fi să se creeze token-urile prin segmentarea textului pe baza oricărui

caracter care nu este alfanumeric. Această abordare are dezavantaje evidente, precum în situația cuvântului Cluj-Napoca care va genera două token-uri deși această grupare de cuvinte reprezintă o singură entitate. Metoda propusă și utilizată de spaCy (https://spacy.io) implică împărțirea repetată a textului doar după spații până când această operație nu mai este posibilă. Astfel, procesul constă în următoarele două etape:

1. Fiecare cuvânt este verificat dacă face parte din o listă de excepții (cuvinte care nu conțin spații, dar ar trebui sparte în mai multe token-uri) dependente de limbă.
2. Având în vedere sufixe și prefixe, se încearcă împărțirea cuvintelor. De exemplu, semnele de punctuație de la finalul unei fraze sunt considerate prefixe. Această abordare implică îndeosebi aceste segmentări în funcție de semnele de punctuație, și sunt aplicabile în general pentru fiecare limbă nefiind nevoie de modele specifice limbii.

În Figura 3 este prezentată metoda descrisă mai sus pentru limba engleză, exemplu oferit de SpCy:

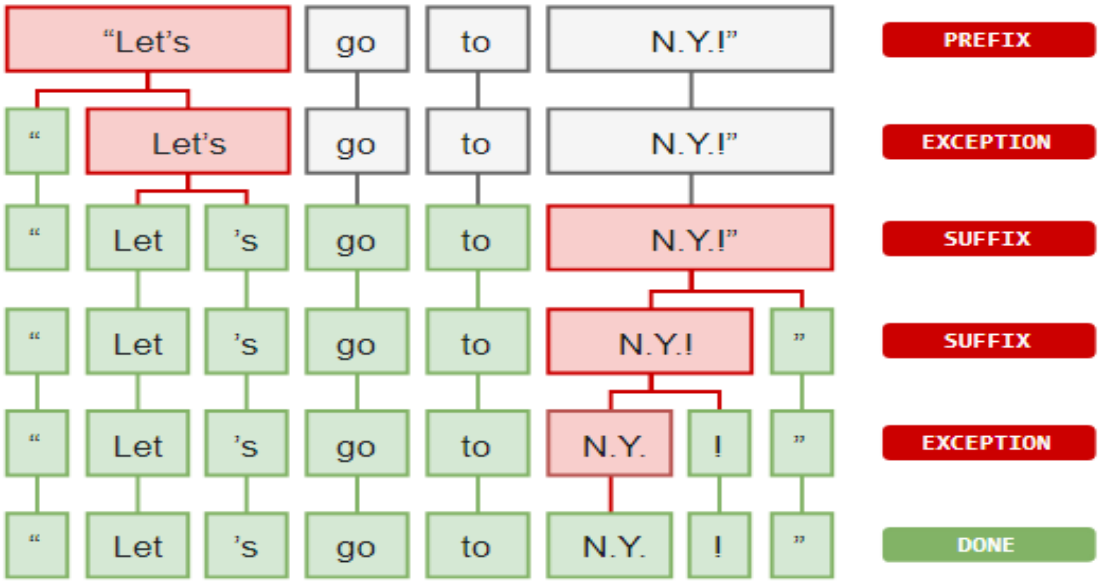

Figura 3. Segmentare cuvinte.

(https://spacy.io/usage/spacy-101)

Adnotarea cu părți de vorbire

Adnotarea cu părți de vorbire reprezintă etapa următoare împărțirii în cuvinte. Această adnotare se realizează în Spacy, folosind un model statistic preantrenat care îi permite să îi atribuie fiecărui cuvânt partea de vorbire corespunzătoare. Modelul are nevoie de foarte multe exemple de text adnotate pentru antrenare, cu scopul de a învăța să facă aceste predicții. De exemplu, modelul poate să învețe că cel mai probabil după un adjectiv urmează un substantiv.

Modelele Markov ascunse vor fi prezentate în continuare ca fiind unul dintre algoritmii preferați în antrenarea de modele care determină partea de vorbire a unui cuvânt, existând însă și alte variante la fel de consistente.

Antrenarea pleacă de la un corpus care conține un număr semnificativ de texte unde fiecare cuvânt este adnotat. Se calculează două tipuri de probabilități: probabilitatea unei părți de vorbire fiind dată probabilitatea părții de vorbire anterioare, și probabilitatea unui cuvânt fiind dată o parte de vorbire. Plecând de la aceste informații se realizează modelul Markov, cu stările fiind părți de vorbire și tranzițiile fiind probabilitățile anterior calculate. Modelul este prezentat în Figura 4 („NN", „VB" sunt părți de vorbire folosite în standarde internaționale):

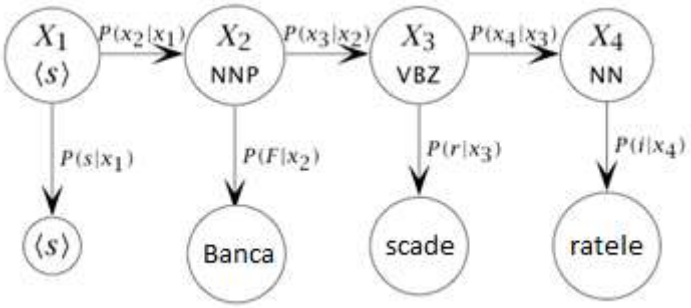

Figura 4. Modelul Markov pentru părți de vorbire.

(https://people.cs.umass.edu/~mccallum/courses/inlp2004/lect10-tagginghmm1.pdf)

După realizarea modelului trebuie calculată cea mai probabilă tranziție de la starea inițială (0) la starea finală (5), ținând cont pentru fiecare parte de vorbire de probabilitatea ei, dându-se cuvântul. O metodă des utilizată este algoritmul lui Viterbi (Forney, 1973).

Alte variante populare pentru adnotare au la bază rețele neurale. Rețeaua va fi alcătuită din 4 straturi: unul de intrare, unul de ieșire și două ascunse între cele două straturi specificate anterior. Neuronii vor fi conectați fiecare cu fiecare (în engleză „fully-connected"), iar ca funcție de activare este folosită în general ReLU (Rectified Linear Units) (Nair & Hinton, 2010). Stratul de ieșire va fi de fapt un strat de tip „softmax" (Bouchard, 2007) care va afișa pentru fiecare parte de vorbire probabilitatea ca acel cuvânt să facă parte din acea clasă. Eroarea va fi calculată conform funcției de „sparse-cross-entropy" (Shore & Johnson, 1980) care măsoară entropia rezultatelor. Ca algoritm de optimizare se poate folosi Adam (Kingma & Ba, 2014), și pentru prevenirea overfitting-ului se pot adăuga straturi de „dropout" (Gal & Ghahramani, 2016) ignorându-se astfel o parte din neuroni. De asemenea, se pot ajusta parametrii de antrenare pentru rezultate cât mai bune.

Analiza sintactică bazată pe dependențe

Pentru prelucrările ulterioare este nevoie de efectuarea analizei sintactice a propoziției („parsing" în engleză). Acest lucru este realizat cu spaCy construindu-se un arbore de dependențe. De exemplu, pentru o propoziție se va determina cine este atributul sau predicatul unui substantiv. Parserele probabiliste construiesc modele plecând de la propoziții parsate „de mână", pe baza lor furnizându-se cea mai probabilă analiză. O ieșire parserului Spacy poate fi vizualizată în Figura 5.

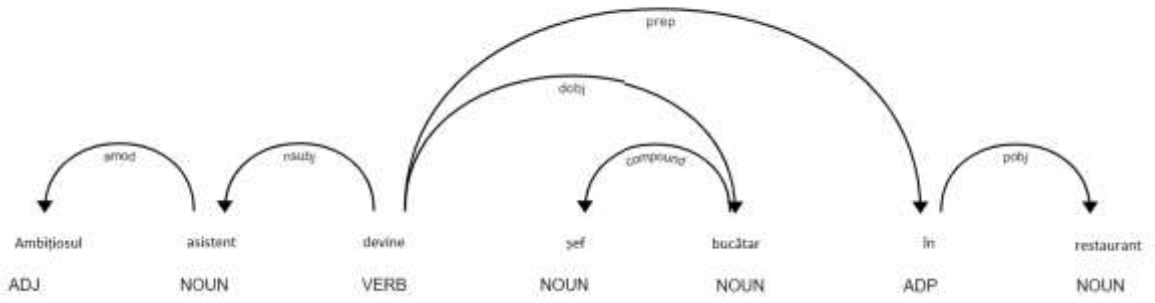

Figura 5. Arborele de dependențe.

(https://nlp.stanford.edu/software/nndep.html)

În Figura 5 se observă că se specifică și tipul de relații dintre cuvintele, nu doar dependențele lor. De asemenea, sunt prezente și părțile de vorbire care ajută în construirea arborelui. Pentru antrenarea modelului se va folosi corpusul (texte în limba română deja adnotate din care modelul învață) de la http://universaldependencies.org/. Pentru engleză Spacy pune la dispoziție un model preantrenat.

În continuare, este prezentată o metodă pentru determinarea acestor dependențe bazată pe tranziții (spacy.io). Algoritmul construiește secvențial arborele de dependențe luând în ordire fiecare propoziție. La fiecare pas, se păstrează o stivă de cuvinte care sunt încă în procesare, și o listă de cuvinte care urmează a fi procesate. Lista împreună cu stiva definesc starea curentă. În starea inițială toate cuvintele se află în listă nefiind încă explorate, iar stiva conține un singur nod și anume nodul „Root”. În orice stare se pot realiza trei tranziții:

1. arc-stânga: marchează al doilea element de pe stivă ca dependent al primului și îl elimină de pe stivă (se poate aplica doar dacă stiva conține minimum două elemente);
2. arc-dreapta: marchează primul element de pe stivă ca dependent al celui de-al doilea și îl elimină (se poate aplica doar dacă stiva conține minimum două elemente);
3. deplasare (shift): elimină un cuvânt din listă și îl pune pe stivă.

Parserul decide ce tranziție să aleagă folosind un clasificator bazat pe o rețea neurală care va avea ca intrare cuvintele din propoziție reprezentate ca vectori.

Antrenare model spaCy pentru limba română

Pentru limbi de circulație internațională cum ar fi engleza, franceza sau germana, spaCy oferă deja modele preantrenate gata să fie folosite. Pentru multe alte limbi, printre care se numără și româna, nu există ceva disponibil. În continuare sunt prezentați pașii pentru antrenarea modelului pentru limba română.

Corpusul în română adnotat cu părți de vorbire și dependențe sintactice a fost descărcat de pe siteul de Universal Dependecies (https://universaldependencies.org/, 2019). Acesta a trebuit transformat din formatul descărcat și anume conllu într-un format JSON specific permis de spaCy pentru antrenare.

Antrenarea a constat în rularea comenzii „spacy train" care suportă următorii parametri:

```
$ python -m spacy train [lang] [output_path] [train_path] [dev_path]
[--base-model] [--pipeline] [--vectors] [--n-iter] [--n-early-stopping] [--n-examples] [--use-gpu]
[--version] [--meta-path] [--init-tok2vec] [--parser-multitasks] [--entity-multitasks]
[--gold-preproc] [--noise-level] [--learn-tokens] [--verbose]
```

În tabelul de mai jos sunt prezentați toți parametrii care pot fi dați comenzii de antrenare. Parametrul „lang" trebuie să fie specificat în formatul spaCy, de exemplu pentru română fiind „ro". Căile directoarelor unde se află seturile de antrenare și de validare împreună cu directorul unde se vrea a fi salvat modelul sunt obligatorii. O placă grafică a fost folosită pentru antrenare, prin urmare parametrul „—use-gpu" a fost adăugat. S-a observat că numărul optim de pași se află în intervalul 150-200.

Parametrul „vectors" a dat posibilitatea de a folosi vectori de cuvinte preantrenați disponibili online. Astfel, FastText (https://fasttext.cc/, 2019) a fost utilizat datorită performanțelor și a integrării ușoare.

Argument	Tip	Descriere
lang	positional	Modelul de limbă.
output_path	positional	Directorul unde se va salva modelul. Va fi creat dacă nu există.
train_path	positional	Locația către directorul/fișierele de antrenare.
dev_path	positional	Locația către directorul/fișierele de validare.
--base-model, -b	optional	Numele unui model de bază spacy care va fi actualizat.
--pipeline, -p	optional	Numele componentelor „pipeline"-ului care vor fi antrenate. Predefinit este 'tagger,parser,ner'.
--vectors, -v	optional	Model de vectori de cuvinte.
--n-iter, -n	optional	Număr de iterații (predefinit:30).
--n-early-stopping, -ne	optional	Numărul maxim de interații fără ca acuratețea de validare să se îmbunătățească.
--n-examples, -ns	optional	Numărul de exemple folosite (predefinit: 0).
--use-gpu, -g	optional	Dacă să se folosească GPU.
--version, -V	optional	Versiunea modelului.
--meta-path, -m	optional	Cale opțională de salvare a fișierului de metadate meta.json.
--init-tok2vec, -t2v	optional	Calea către vectorii de cuvinte preantrenați. Experimental.

Argument	Tip	Descriere
--parser-multitasks, -pt	optional	Alte obiective pentru parserul CNN.
--entity-multitasks, -et	optional	Alte obiective pentru NER CNN.
--noise-level, -nl	optional	Număr care indică cât zgomot există în date.
--gold-preproc, -G	indicator	Folosirea preprocesării de aur („gold”).
--learn-tokens, -T	indicator	Dacă modelul să învețe tokenizare. Util pentru limba Chineză.
--verbose, -VV	indicator	Arată mai multe informații în loguri.
--help, -h	indicator	Afișează mesaje ajutătoare.

Pe lângă acești parametri în linie de comandă, spaCy oferă posibilitatea configurării antrenării prin variabile de mediu. Acestea sunt prezentate în tabelul de mai jos. Pentru a evita overfitting-ul a fost micșorată rata de antrenare („learning rate”) și a fost mărit L2_penalty.

Nume	Descriere	Predefinit
dropout_from	Valoare inițială de droprate.	0.2
dropout_to	Valoare finală de droprate.	0.2
dropout_decay	Rata cu care se schimbă dropout-ul.	0.0
batch_from	Mărimea lotului inițială.	1
batch_to	Mărimea lotului finală.	64
batch_compound	Rata de accelerare a mărimii lotului.	1.001
token_vector_width	Lățimea tabelelor de embeddings și a straturilor convoluționale.	128
embed_size	Numărul de linii în tabelele de embeddins.	7500
hidden_width	Mărirea straturilor de parsare și recunoaștere de entități.	128
learn_rate	Rata de învățare.	0.001
optimizer_B1	Valoarea momentului pentru Adam.	0.9
optimizer_B2	Adagrad-momentum pentru Adam.	0.999
optimizer_eps	Valoarea lui epsilon pentru Adam.	1e-08
L2_penalty	Penalizarea la L2 regularization.	1e-06
grad_norm_clip	Gradientul L2.	1.0

Pentru partea de recunoaștere de entități în limba română a fost folosit corpusul RONEC (Dumitrescu et all, 2019) (https://github.com/dumitrescustefan/ronec, 2019) care are 5127 de propoziții, 16 clase și 26376 de entități adnotate. Această parte a fost antrenată separat de celelate.

Rezultatele pentru modelul de limba română sunt prezentate mai jos, unde UAS reprezintă Unlabeled Attachement Score, LAS reprezintă Labeled Attachement Score, Ents_p,r,f reprezintă scorurile de precision, recall și f1 pentru entități, iar Tags_acc reprezintă acuratețea la nivel de taguri:

- 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. Rasa se bazează pe înțelegerea limbajului natural pentru a extrage entități și intenții din texte, astfel eliminând nevoia agenților inteligenți care sunt bazați pe un set extins de reguli ce la un moment dat devine greu de întreținut și de extins. Astfel, se poate structura limbajul și se pot folosi tehnici precum monitorizarea evoluției conversației pentru a prevedea posibilele acțiuni ce derivă din comportamentul utilizatorului. Acesta poate fi privit ca o soluție eficientă deoarece nu este nevoie de transmiterea datelor la entități auxiliare în vederea procesării, ceea ce scade spre zero numărul de cereri HTTP care ar trebui execute, și în același timp se poate ajusta modelul creat după propriile nevoi. Rasa oferă și posibilitatea de a implementa acțiuni ca răspuns la mesajele utilizatorului, cum ar fi: răspunderea cu un mesaj, modificarea unei baze de date, apelarea unui API, sau chiar transmiterea legăturii către un operator uman.

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.

Folosirea acestui instrument este exemplificată în 5.1. Utilizatorul pune o întrebare în interfață, iar mesajul este trimis către aplicație. Acest mesaj este interpretat de modulul care determină intențiile și entitățile din mesaj, și care trimite mai departe aceste informații către modulul inteligent capabil să ofere un răspuns la întrebare. Acesta din urmă se folosește de surse de date, în cazul de față orarul facultății, pentru a putea oferi informațiile necesare modulului de dialog din RASA și răspunde utilizatorului.

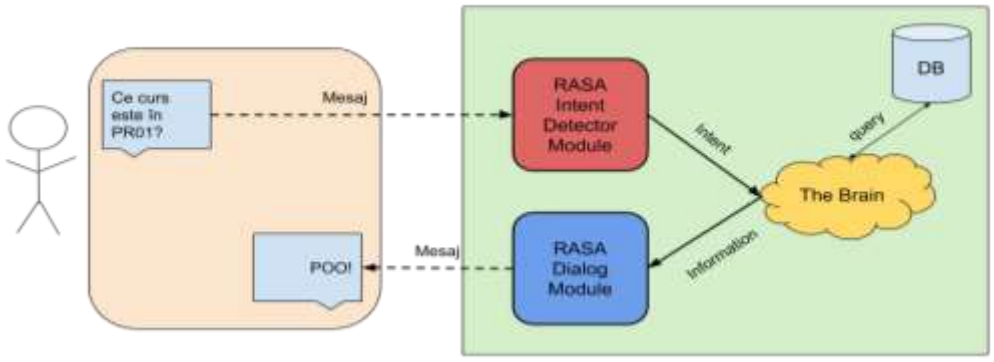

Figura 5.1. Exemplu de utilizare și arhitectura RASA.

Referințe

[Bouchard , 2007] Bouchard, G. Efficient bounds for the softmax function, applications to inference in hybrid models.

[Forney, 1973] Forney, G. D. The viterbi algorithm. *Proceedings of the IEEE, 61*(3), 268-278.

[Gal and Ghahramani , 2016] Gal, Y., & Ghahramani, Z. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems* (pp. 1019-1027).

[Kingma and Ba, 2014] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Nair and Hinton, 2010] Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)* (pp. 807-814).

[Shore and Johnson, 1980] Shore, J., & Johnson, R. (1980). Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy. *IEEE Transactions on information theory, 26*(1), 26-37.

5.6 Diseminare

Dan Tufiș, Verginica Barbu Mititelu, Elena Irimia, Maria Mitrofan, Radu Ion, and George Cioroiu (2019). Making Pepper Understand and Respond in Romanian. 22ndINTERNATIONAL CONFERENCE ON CONTROL SYSTEMS AND COMPUTER SCIENCE, Bucuresti, 28-30 mai 2019

Elena Irimia, Maria Mitrofan, Verginica Barbu Mititelu (2019). Evaluating the Wordnet and CoRoLa-based Word Embedding Vectors for Romanian as Resources in the Task of Microworlds Lexicon Expansion. The 10th Global WordNet Conference, Wroclaw, Polonia, 22-27 iulie 2019

Maria Mitrofan, Verginica Barbu Mititelu, Grigorina Mitrofan (2019). MoNERo: a Biomedical Gold Standard Corpus for the Romanian Language, The 18th BioNLP Workshop and Shared Task, Florenta, Italia, 1 August 2019

Panaite, M., Ruseti, S., Dascalu, M., Balyan, R., McNamara, D. S., & Trausan-Matu, S. (2019). Automated Scoring of Self-explanations Using Recurrent Neural Networks. In M. Scheffel, J. Broisin, V. Pammer-Schindler, A. Ioannou & J. Schneider (Eds.), *14th*

European Conference on Technology Enhanced Learning (EC-TEL 2019) (pp. 659–663). Delft, Netherlands: Springer.

Panaite, M., Ruseti, S., Dascalu, M., & Trausan-Matu, S. (2019). Towards a Deep Speech Model for Romanian Language. In *4th Int. Workshop on Design and Spontaneity in Computer-Supported Collaborative Learning (DS-CSCL-2019), in conjunction with the 22nd Int. Conf. on Control Systems and Computer Science (CSCS22)* (pp. 416–419). Bucharest, Romania: IEEE.

Boboc, I. G., Dascalu, M., & Trausan-Matu, S. (2019). Image Style Transfer using Text Descriptions. In *International Conference on Human-Computer Interaction (RoCHI2019)* (pp. 22–29). Bucharest, Romania: MatrixRom.

Nenciu, B., Ruseti, S., & Dascalu, M. (2018). Extracting Actions from Romanian Instructions for IoT Devices. In V. Pais, D. Gifu, D. Trandabat, D. Cristea & D. Tufis (Eds.), *13th Int. Conf. on Linguistic Resources and Tools for Processing Romanian Language (ConsILR 2018)* (pp. 168–176). Iasi, Romania.

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 indeplinite complet.

5.7 Servicii de cercetare şi tehnologice oferite de Institutul de Cercetări pentru Inteligență Artificială (https://erris.gov.ro/RACAI-ICIA):

Servicii: Interogare corpus de referință al limbii române scisă și vorbită corola.racai.ro; TTL http://ws.racai.ro/ttlws.wsdl, Modular Language Processing for Lightweight Applications (MPLA) cu prelucrări pentru mai mult de 40 de limbi http://slp.racai.ro/index.php/mlplanew/; Sistemul ROBIN-dialog de prelucrare a dialogurilor în micro-lumile țintă; Lexicon extins pentru aplicații de prelucrare a vorbirii;

5.8. Locuri susținute de acest proiect

6 cercetători cu vechime în ICIA (D. Tufis, V. Mititelu, E. Irimia, M. Mitrofan, R. Ion, E. Curea) plus 2 tineri cercetători angajați pe proiect (G. Cioroiu, V. Badea).

5.9. CEC-uri

Sumele alocate cec-urilor nu au fost valorificate, în principal pentru că nu s-au identificat oportunități conforme cu reglementările de acordare.

Anexa – Exemple de Intenții

// Pepper poate conduce persoana până la Sx
// Pepper poate afișa o hartă a etajului cu sala Sx reprezentată
// Pepper poate explica unde se află sala Sx (de ex. „înainte, pe al doilea culoar la dreapta, a treia ușă”)
intent:locateClassroom

- Unde se află sala [Sx] (classroom:classroom)?
- Îmi poți explica unde se află sala [Sx] (classroom:classroom), te rog?
- Spune-mi unde găsesc sala [Sx] (classroom:classroom)!
- Condu-mă la sala [Sx] (classroom:classroom)!
- Explică-mi cum ajung la sala [Sx] (classroom:classroom)!
- Cum pot ajunge la sala [Sx] (classroom:classroom)?
- Există vreo hartă pentru a vizualiza localizarea sălii [Sx] (classroom:classroom)?
- Unde găsesc sala [Sx] (classroom:classroom)?
- Arată-mi harta cu sala [Sx] (classroom:classroom) reprezentată.
- Îmi poți arăta o hartă cu sala [Sx] (classroom:classroom) reprezentată.
- Unde pot găsi sala [Sx] (classroom:classroom)?
- Care este drumul către sala [Sx] (classroom:classroom)?
- Poti să mă conduci la sala [Sx] (classroom:classroom), te rog?
- Ajută-mă să găsesc sala [Sx] (classroom:classroom)!
- Mă poți îndruma către sala [Sx] (classroom:classroom)?
- În ce direcție se găsește sala [Sx] (classroom:classroom)?
- Mă poți ghida către sala [Sx] (classroom:classroom)?
- În ce sens se află sala [Sx] (classroom:classroom)?
- Îmi poți arăta calea către sala [Sx] (classroom:classroom), te rog?
- Pe unde să o iau ca să ajung la sala [Sx] (classroom:classroom)?
- Unde e sala [Sx] (classroom:classroom)?
- În ce loc se află sala [Sx] (classroom:classroom)?
- Care este direcția către sala [Sx] (classroom:classroom)?
- Încotro să o iau către sala [Sx] (classroom:classroom)?
- Pe unde se află sala [Sx] (classroom:classroom)?
- Arată-mi calea către sala [Sx] (classroom:classroom)!
- Poți să îmi indici drumul către sala [Sx] (classroom:classroom)?
- Zi-mi unde găsesc sala [Sx] (classroom:classroom)!
- Vreau să văd harta cu sala [Sx] (classroom:classroom) reprezentată.
- La ce etaj se află sala [Sx] (classroom:classroom)?
- Știi unde se află sala [Sx] (classroom:classroom)?
- Mă poți îndrepta către sala [Sx] (classroom:classroom)?
- Mă conduci până la sala [Sx] (classroom:classroom)?
- Afișează-mi harta etajului corespunzător sălii [Sx] (classroom:classroom)!
- Unde este localizată sala [Sx] (classroom:classroom)?
- Ajută-mă să găsesc sala [Sx] (classroom:classroom)!
- Indică-mi drumul către sala [Sx] (classroom:classroom)!
- Am nevoie de ajutor să găsesc drumul către sala [Sx] (classroom:classroom)!
- Care este drumul spre sala [Sx] (classroom:classroom)?

// Pepper consultă orarul sălii Sx și răspunde
intent:searchTeacher

- Ce profesor are curs în sala [Sx] (classroom:classroom)?
- Ce profesor are laborator în sala [Sx] (classroom:classroom)?
- Cine ține cursul în sala [Sx] (classroom:classroom)?
- Cine ține laboratorul în sala [Sx] (classroom:classroom)?
- Caută-mi numele profesorului ce are curs în sala [Sx] (classroom:classroom)!
- Caută-mi numele profesorului ce ține laboratorul din sala [Sx] (classroom:classroom)!
- Spune-mi, te rog, numele profesorului ce are curs în sala [Sx] (classroom:classroom)!
- Spune-mi, te rog, numele profesului ce are laborator în sala [Sx] (classroom:classroom) !
- Îmi poți spune, te rog, numele profesului ce are laborator în sala [Sx] (classroom:classroom)?
- Îmi poți spune, te rog, numele profesului ce are curs în sala [Sx] (classroom:classroom)?
- Cine predă cursul în sala [Sx] (classroom:classroom)?
- Cine predă laboratorul în sala [Sx] (classroom:classroom)?

// Pepper consultă orarul tuturor sălilor și răspunde
// Dacă sunt mai multe evenimente de același fel (de ex. mai multe laboratoare de sisteme de operare), Pepper trebuie să întrebe la rândul său despre grupa studentului/profesorul care ține cursul, etc.
// Ex - curs, laborator sau workshop
##intent:findClassroom

- Unde se desfășoară [Ex] (event:event)?
- În ce sală are loc [Ex] (event:event)?
- Unde va avea loc [Ex] (event:event)?
- Îmi poți spune, te rog, unde se va ține [Ex] (event:event)?
- Știi unde se va ține [Ex] (event:event)?
- Unde trebuie să mă duc pentru a participa la [Ex] (event:event)?
- Unde se predă [Ex] (event:event)?
- În ce loc se ține [Ex] (event:event)?
- Spune-mi, te rog, în ce sală pot participa la [Ex] (event:event)?
- Ajută-mă să găsesc sala unde se ține [Ex] (event:event)!
- Unde are loc [Ex] (event:event)?
- Mă poți ajuta să găsesc unde are loc [Ex] (event:event)?
- Știi unde se desfășoară [Ex] (event:event)?
- Arată-mi sala în care se desfășoară [Ex] (event:event)!

// Pepper consultă orarul tuturor sălilor și răspunde

##intent:getHourEvent

- Când începe [Ex] (event:event)?
- La ce oră are loc [Ex] (event:event)?
- Poți să îmi spui la ce oră se ține [Ex] (event:event)?
- Îmi poți spune, te rog, la ce oră începe [Ex] (event:event)?
- Știi la ce oră se desfășoară [Ex] (event:event)?
- La ce oră se desfășoară [Ex] (event:event)?
- La ce oră începe [Ex] (event:event)?
- Îmi poți indica momentul de timp la care începe [Ex] (event:event)?
- Îmi poți preciza ora la care se ține [Ex] (event:event)?
- Ai idee la ce oră are loc [Ex] (event:event)?
- Precizează-mi ora la care începe [Ex] (event:event)!
- Spune-mi ora la care începe [Ex] (event:event)!

- Vreau să aflu ora la care începe [Ex] (event:event)!

// // Pepper consultă orarul tuturor sălilor și răspunde
##intent:getDateEvent

- În ce dată are loc [Ex] (event:event)?
- Îmi poți spune, te rog, în ce dată se ține [Ex] (event:event)?
- Îmi poți specifica în ce dată se desfășoară [Ex] (event:event)?
- Spune-mi, te rog, în ce dată are loc [Ex] (event:event)?
- Ai idee în ce dată se ține [Ex] (event:event)?
- Spune-mi în ce dată are loc [Ex] (event:event)!
- Precizează-mi dată când se ține [Ex] (event:event)
- Precizează-mi dată când se ține [Ex] (event:event), te rog!
- Poți afla data când se ține [Ex] (event:event)?
- Vreau să știu data în care se desfășoară [Ex] (event:event)!
- Știi în ce dată se ține [Ex] (event:event)?

// Pepper consultă orarul sălii în care se află și răspunde
##intent:confirmDateEvent

- Aici se desfășoară [Ex] (event:event)?
- Se desfășoară aici azi [Ex] (event:event)?
- Se desfășoară aici, în acest moment, [Ex] (event:event)?
- În acest moment, se ține aici [Ex] (event:event)?
- Aici are loc [Ex] (event:event)?
- Acum se ține aici [Ex] (event:event)?
- Poți să îmi spui, te rog, dacă aici se ține acum [Ex] (event:event)?
- Poți să îmi spui dacă aici se ține acum [Ex] (event:event)?
- Îmi poți spune dacă aici are loc acum [Ex] (event:event)?
- Îmi poți spune, te rog, dacă aici are loc acum [Ex] (event:event)?
- Știi cumva dacă aici se desfășoară acum [Ex] (event:event)?
- Spune-mi, dacă acum se ține [Ex] (event:event)!
- Spune-mi, te rog, dacă acum se ține [Ex] (event:event)!
- Îmi poți preciza dacă aici are loc acum [Ex] (event:event)?
- Îmi poți confirma dacă astăzi se ține aici [Ex] (event:event)?

// Pepper răspunde cu indicații suplimentare unde se desfășoară Ex
##intent:startTimeEvent

- Când începe [Ex] (event:event)?
- A început [Ex] (event:event)?
- Îmi poți spune, te rog, dacă a început [Ex] (event:event)?
- Îmi poți spune dacă a început [Ex] (event:event)?
- Spune-mi, când a început [Ex] (event:event)!
- Spune-mi, te rog, când a început [Ex] (event:event)!
- Știi cumva când a început [Ex] (event:event)?
- Știi dacă a început [Ex] (event:event)?
- Când a început [Ex] (event:event)?
- La cât a început [Ex] (event:event)?
- Poți să îmi zici la ce oră a început [Ex] (event:event)?
- Poți să îmi zici, te rog, la ce oră a început [Ex] (event:event)?
- Poți să îmi spui dacă începe [Ex] (event:event)?

// Pepper răspunde cu indicații suplimentare unde se desfășoară Ex
##intent:endTimeEvent

- Când se termină [Ex] (event:event)?
- Îmi poți spune, te rog, când se termină [Ex] (event:event)?
- Îmi poți spune când se termină [Ex] (event:event)?
- Poți să îmi spui dacă s-a terminat [Ex] (event:event)?
- Poți să îmi spui, te rog, dacă s-a terminat [Ex] (event:event)?
- Spune-mi, te rog frumos, dacă s-a terminat [Ex] (event:event)!
- Spune-mi, când se termină [Ex] (event:event)!
- Ai idee când se termină [Ex] (event:event)?
- Știi când se termină [Ex] (event:event)?
- Pe la ce oră se termină [Ex] (event:event)?

// Pepper răspunde cu programul tuturor examenelor (ora + dată)
// Pepper să întrebe grupa/seria sau întrebarea să conțină grupa/seria
##: getExamTimetable

- Când au loc examenele în acest semestru?
- În ce date sunt programate examenele?
- Când se vor desfășura examenele în acest semestru?
- S-a postat programul examenelor?
- Există o listă cu examenele de pe acest semestru?
- Îmi poți spune, te rog, când vor avea loc examenele?
- Îmi poți spune când vor avea loc examanele?
- Care este programul examenelor?
- Spune-mi, te rog, care este programul examenelor!
- Spune-mi care este programul examenelor!
- Afișează-mi programul examenelor!
- Arată-mi, te rog, programul examenelor?

// Pepper răspunde cu data și ora examenului cerut (Vx variabilă de tip “examen”)
##intent:getDateExam

- În ce dată are loc [Vx] (exam:exam)?
- Când se va desfășura [Vx] (exam:exam)?
- Când are loc [Vx] (exam:exam)?
- Spune-mi, data când se va desfășura [Vx] (exam:exam)!
- Spune-mi, te rog, data când se va desfășura [Vx] (exam:exam)!
- Poți să îmi spui când se ține [Vx] (exam:exam)?
- Poți să îmi spui, te rog, când se ține [Vx] (exam:exam)?
- S-a postat când are loc [Vx] (exam:exam)?
- Știi când se va desfășura [Vx] (exam:exam)?

// Pepper consultă programarea examenelor
##intent:locateExam

- Unde se desfășoară [Vx] (exam:exam)?
- În ce sală are loc [Vx] (exam:exam)?
- Unde va avea loc [Vx] (exam:exam)?
- Îmi poți spune, te rog, unde se va ține [Vx] (exam:exam)?

- Știi unde se va ține [Vx] (exam:exam)?
- Unde trebuie să mă duc pentru a participa la [Vx] (exam:exam)?
- În ce loc se ține [Vx] (exam:exam)?
- Spune-mi, te rog, în ce sală pot participa la [Vx] (exam:exam)?
- Ajută-mă să găsesc sala unde se ține [Vx] (exam:exam)!
- Unde are loc [Vx] (exam:exam)?
- Mă poți ajuta să găsesc unde are loc [Vx] (exam:exam)?
- Știi unde se desfășoară [Vx] (exam:exam)?
- Arată-mi sala în care se desfășoară [Vx] (exam:exam)!

// Pepper consultă orarul sălilor și răspunde cu numele sălii și numele laborantului
// Pepper ia in calcul alternativele din aceeasi saptamana
##intent: changeLab

- Mă pot duce la laboratorul [Ex] (event:event) în altă zi?
- Se mai face acest laborator [Ex] (event:event) în altă zi?
- Știi dacă se mai face în altă zi laboratorul [Ex] (event:event)?
- Am o altă alternativă pentru a participa la laboratorul [Ex] (event:event)?
- Poți să îmi spui când mai apare în orar laboratorul [Ex] (event:event)?
- Poți să îmi spui când mai apare în orar laboratorul [Ex] (event:event), te rog?
- Ai idee când se mai face laboratorul [Ex] (event:event)?
- Există alt interval orar pentru laboratorul [Ex] (event:event)?
- Când se mai ține laboratorul [Ex] (event:event)?
- Când se mai desfășoară laboratorul [Ex] (event:event)?
- Spune-mi, te rog, când se mai desfășoară laboratorul [Ex] (event:event)!
- Spune-mi, când se mai ține laboratorul [Ex] (event:event)!
- În ce zi mai pot participa la laboratorul [Ex] (event:event)?
- În ce dată se mai face laboratorul [Ex] (event:event)?
- Se mai ține acest laborator săptămâna aceasta?

// Pepper răspunde cu email-ul profesorului
// Tx variabilă de tip “profesor”
##intent: getEmailTeacher

- Ce email are profesorul [Tx] (teacher:teacher)
- Îmi poți spune, te rog, email-ul profesorului [Tx] (teacher:teacher)?
- Știi adresa de email a profesorului [Tx] (teacher:teacher)?
- Spune-mi, te rog, adresa de email a profesorului [Tx] (teacher:teacher)!
- Care este adresa de email a profesorului [Tx] (teacher:teacher)?
- Poți să îmi spui adresa de email a profesorului [Tx] (teacher:teacher)?
- Arată-mi adresa de email a profesorului [Tx] (teacher:teacher), te rog!
- Ai cumva adresa de email a profesorului [Tx] (teacher:teacher)?

// Pepper răspunde cu email-ul șefului de serie
// Bx variabilă de tip “sef serie”
##intent: getEmailBossStudent

- Ce email are șeful de serie [Bx] (boss:boss)?
- Îmi poți spune, te rog, email-ul șefului de serie [Bx] (boss:boss)?
- Știi adresa de email a șefului de serie [Bx] (boss:boss)?
- Spune-mi, te rog, adresa de email a șefului de serie [Bx] (boss:boss)!
- Care este adresa de email a șefului de serie [Bx] (boss:boss)?

- Poți să îmi spui adresa de email a șefului de serie [Bx] (boss:boss)?
- Arată-mi adresa de email a șefului de serie [Bx] (boss:boss), te rog!
- Ai cumva adresa de email a șefului de serie [Bx] (boss:boss)?

// Pepper răspunde cu numele șefului de serie
// Dx variabilă de tip serie/directie
##intent: getNameBossStudent

- Cum îl cheamă pe șeful seriei de la [Dx] (direction:direction)?
- Cum se numește șeful seriei de la [Dx] (direction:direction)?
- Știi cumva numele șefului de serie de la [Dx] (direction:direction)?
- Care este numele șefului de serie de la [Dx] (direction:direction)?
- Spune-mi, te rog, numele șefului de serie de la [Dx] (direction:direction)!
- Poți să îmi zici numele șefului de serie de la [Dx] (direction:direction)?
- Îmi poți spune numele șefului de serie de la [Dx] (direction:direction)?

// Pepper răspunde cu intervalul orar asociat secretariatului
##intent: getScheduleSecretariat

- Care este programul secretariatului?
- În ce interval orar pot trece pe la secretariat?
- Când pot trece pe la secretariat?
- Mă poți ajuta cu orele când o găsesc pe secretară?
- Când o pot găsi pe secretară?
- Între ce ore este program cu studenții la secretariat?
- În ce interval o pot găsi pe secretară?
- Spune-mi, te rog, programul secretariatului!
- Poți să îmi spui, te rog, programul secretariatului?
- Știi între ce ore este deschis secretariatul?

// Pepper explică formatul orarului (săptămână pară/săptămână impară), oferind și un exemplu
// Utilizatorul solicită o explicație pentru orarul de la o serie (de exemplu: CB)
##intent: explainSchedule

- Poți să mă ajuți să citesc orarul de la [Dx] (direction:direction)?
- Poți să îmi explici orarul de la [Dx] (direction:direction)?
- Ai idee cum pot să citesc orarul de la [Dx] (direction:direction)?
- Spune-mi, te rog, cum să citesc orarul de al [Dx] (direction:direction)?
- Știi să citești orarul de al [Dx] (direction:direction)?
- Explică-mi orarul de al [Dx] (direction:direction), te rog!
- Ajută-mă să înțeleg orarul de la [Dx] (direction:direction)!

// Pepper oferă o descriere a unui Ex (materie, curs, workshop..etc)
##intent: getDescriptionEvent

- Poți să îmi descrii [Ex] (event:event)?
- Ce detalii cunoști despre [Ex] (event:event)?
- Poți să îmi oferi o descriere despre [Ex] (event:event)?
- Ce știi despre [Ex] (event:event)?
- Descrie-mi [Ex] (event:event), te rog!
- Ce presupune [Ex] (event:event)?

- Despre ce este [Ex] (event:event)?
- Poți să îmi prezinți [Ex] (event:event)?
- Ai putea să îmi oferi o scurtă descriere pentru [Ex] (event:event)?

// Pepper răspunde cu o listă cu deadline-uri pentru grupa respectivă
// Gx variabilă utilizată pentru grupa unui student
##intent:getHomeworkTimetable

- Poți să îmi spui când trebuie să predau temele la grupa [Gx] (group:group)?
- Știi care este programarea temelor pentru grupa [Gx] (group:group)?
- Când sunt programate deadline-urile pentru grupa [Gx] (group:group)?
- În ce zile sunt programate deadline-urile pentru grupa [Gx] (group:group)?
- Poți să îmi faci o listă cu deadline-urile grupei mele [Gx] (group:group)?
- Știi când am deadline la grupa [Gx] (group:group)?
- Ai idee când este următorul deadline pentru grupa [Gx] (group:group)?
- Poți să îmi spui când am următoarele deadline-uri, fiind la grupa [Gx] (group:group)?
- Vreau o listă cu deadline-urile de la grupa [Gx] (group:group)!
- Spune-mi, te rog, când am următoarele deadline-uri pentru grupa [Gx] (group:group)!
- Vreau să știi când trebuie să trimit temele pentru grupa [Gx] (group:group)!
- Știi când trebuie să trimit temele la grupa [Gx] (group:group)?
- Ai idee când trebuie trimise temele la grupa [Gx] (group:group)?
- Când trebuie trimise temele la grupa [Gx] (group:group)?

// Pepper răspunde cu deadline-ul pentru tema respectivă
// Hx variabilă utilizată pentru temă
##intent:getHomeworkDeadline

- Când este deadline-ul pentru [Hx] (homework:homework)?
- Când trebuie postată tema [Hx] (homework:homework)?
- Cânt trebuie trimisă tema [Hx] (homework:homework)?
- Ai idee când ar trebui să trimit tema [Hx] (homework:homework)?
- Vreau să știu când trebuie să trimit tema [Hx] (homework:homework)!
- Spune-mi, te rog, când să postez tema [Hx] (homework:homework)!
- Spune-mi când să trimit tema [Hx] (homework:homework)!
- Poți să îmi spui când să trimit tema [Hx] (homework:homework)?
- Când aș putea trimite tema [Hx] (homework:homework)?
- În ce dată trebuie trimisă tema [Hx] (homework:homework)?
- În ce zi ar trebui postată tema [Hx] (homework:homework)?

// Pepper va confirma dacă deadline-ul pentru o temă este astăzi
##intent:confirmHomeworkDeadline

- Astăzi trebuie să trimit tema [Hx] (homework:homework)?
- Azi este deadline-ul pentru tema [Hx] (homework:homework)?
- Spune-mi, te rog, dacă astăzi trebuie să postez tema [Hx] (homework:homework)!
- Poți să îmi spui dacă astăzi trebuie să trimit tema [Hx] (homework:homework)?
- Îmi poți spune dacă azi ar trebui să postez [Hx] (homework:homework)?
- Știi cumva dacă azi ar trebui să trimit tema [Hx] (homework:homework)?
- Ai idee dacă astăzi ar trebui să trimit tema [Hx] (homework:homework)?

// Pepper răspunde cu media peste care se oferă burse
##intent: findGradeForScholarship

- Care este media de bursă pentru anul acesta?
- Ce medie de bursă este anul acesta?
- Ce medie trebuie să am pentru a obține bursă?
- Peste ce medie pot să obțin bursă?
- Care este media minimă de la cât pot primi bursă?
- Îmi poți spune, te rog, care este media de la care pot primi bursă în acest an?
- Poți să îmi spui ce medie îmi trebuie pentru a obține bursă în acest an universitar?
- De la ce medie pot primi bursă?
- Spune-mi, te rog, de ce medie am nevoie pentru a obține bursă!
- Vreau să aflu medie de bursă pentru acest an!
- Ce medie este necesară pentru a obține bursă?
- Știi cumva media de bursă pentru acest an?

// Pepper răspunde cu condițiile minime pentru a trece o materie
##intent:getMinConditions

- Care sunt condițiile minime pentru a trece [Ex] (event:event)?
- Care sunt condițiile pentru a promova [Ex] (event:event)?
- Poți să îmi spui, te rog, condițiile pentru a promova [Ex] (event:event)?
- Ce condiții trebuie să îndeplinesc pentru a promova [Ex] (event:event)?
- Spune-mi, te rog, care sunt condițiile pentru a promova [Ex] (event:event)!
- Știi cumva care sunt condițiile pentru a trece [Ex] (event:event)?
- Poți să îmi specifici condițiile pentru a promova [Ex] (event:event)?
- Ai idee care sunt condițiile pentru a promova [Ex] (event:event)?

// Pepper răspunde cu punctajul aferent fiecărui eveniment (curs, laborator..)
##intent:getScoreEvent

- Care este punctajul lui [Ex] (event:event)?
- Ce punctaj are [Ex] (event:event)?
- Cum este împărțit punctajul pentru [Ex] (event:event)?
- Știi cumva cum este împărțit punctajul pentru [Ex] (event:event)?
- Ai idee care este punctajul pentru [Ex] (event:event)?
- Poți să îmi spui ce punctaj are [Ex] (event:event)?
- Îmi poți spune ce punctaj are [Ex] (event:event)?
- Ajută-mă să aflu ce punctaj are [Ex] (event:event), te rog!
- Spune-mi ce punctaj are [Ex] (event:event)!
- Vreau să aflu punctajul pentru [Ex] (event:event)!

// Pepper va răspunde cu o listă cu colegii de grupă
// Îl va întreba pe utlizator numele lui, pentru a-l exclude
##intent:getListStudents

- Poți să îmi faci o listă cu colegii de la grupa [Gx] (group:group)?
- Știi ce studenți sunt în grupa [Gx] (group:group)?
- Ai idee ce studenți sunt la grupa [Gx] (group:group)?
- Ce studenți fac parte din grupa [Gx] (group:group)?
- Ce studenți sunt înmatriculați la grupa [Gx] (group:group)?
- Spune-mi, te rog, ce studenți fac parte din grupa [Gx] (group:group)?
- Poți să îmi spui numele studenților care fac parte din grupa [Gx] (group:group)?
- Vreau să aflu numele studenților care învață la grupa [Gx] (group:group)!

- Arată-mi numele studenților care fac parte din grupa [Gx] (group:group), te rog!
- Ai putea să mă ajuți cu o listă cu studenții care fac parte din grupa [Gx] (group:group)?

// Pepper va răspunde cu lista materiilor corespunzătoare grupei studentului, deci va solicita grupa
##intent:getListCourses

- Ce cursuri am în acest an?
- La ce cursuri sunt înscris/înscrisă?
- Mă poți ajuta să aflu cursurile la care sunt înscrisă?
- Ai idee la ce cursuri sunt înscris/înscrisă?
- Spune-mi, te rog, la ce cursuri sunt înscris/înscrisă.
- Ai putea să îmi specifici cursurile la care sunt înscris/înscrisă?
- Știi cumva la ce cursuri sunt înscris/înscrisă?
- Îmi poți arâta cursurile la care trebuie să particip în acest an?
- La ce cursuri trebuie să particip în acest an?

// Pepper va răspunde cu o listă a profesorilor (voi utiliza variabila Ex, chiar dacă se referă, în mod special, la un laborator)
// În cazul în care utilizatorul specifică un Px, acesta va fi eliminat din lista expusă acestuia
##intent:getListTeachers

- Ce laboranți țin [Ex] (event:event)?
- Care este numele tuturor laboranților care țin laboratorul [Ex] (event:event)?
- Care sunt profesori care desfășoară laboratorul [Ex] (event:event)?
- Ai idee ce laboranți predau [Ex] (event:event)?
- Știi cumva care sunt numele laboranților care predau [Ex] (event:event)?
- Spune-mi, te rog, numele laboranților, care țin laboratul de [Ex] (event:event)?
- Vreau numele laboranților care fac [Ex] (event:event)!
- Cu cine se mai poate face [Ex] (event:event)?
- Mai există și alt laborant în afară de [Px] (teacher:teacher) pentru [Ex] (event:event)?
- Ce alți laboranți, excluzând profesorul [Px] (teacher:teacher), țin laboratorul [Ex] (event:event)?
- Ce alți laboranți, în afară de profesorul [Px] (teacher:teacher), țin laboratorul [Ex] (event:event)?
- Mă poți ajuta să aflu numele tuturor laboranților care țin [Ex] (event:event)?
- Ai putea să îmi faci o listă cu toți laboranții care țin [Ex] (event:event)?

// Pepper va răspunde cu un model de cerere (Exemplu: utilizatorul vrea să depună o cerere la secretariat pentru a obține o adeverință de student și are nevoie de un model)
// Dacă nu este specificat tipul de cerere, Pepper va întreba ce tip de cerere
// Cx variabilă de tipul cerere
##intent:getModelForRequest

- Ai vreun model de cerere pentru [Cx] (request:request)?
- Ai vreun model de cerere?
- Mă poți ajuta, te rog frumos, cu un model de cerere?
- Mă poți ajuta, te rog frumos, cu un model de cerere pentru [Cx] (request:request)?
- Poți să îmi arăți un model de cerere pentru [Cx] (request:request)?
- Poți să îmi arăți un model de cerere?
- Arată-mi, te rog, un model de cerere pentru [Cx] (request:request)!

- Ai putea să îmi arăți, te rog, un model de cerere?
- Îmi poți arăta un model de cerere pentru [Cx] (request:request)?
- Îmi poți arăta un model de cerere?
- Ai vreun exemplu de model de cerere pentru [Cx] (request:request)?
- Ai vreun exemplu de model de cerere?
- Știi cum se formulează o cerere pentru [Cx] (request:request)?
- Știi cum se formulează o cerere?
- Ai idee cum se formulează o cerere pentru [Cx] (request:request)?
- Ai idee cum se formulează o cerere?
- Mă poți ajuta să formulez o cerere pentru a o depune la secretariat?
- Mă poți ajuta să formulez o cerere pentru [Cx] (request:request) pentru a o depune la secretariat?