RAPORTARE ŞTIINŢIFICĂ

FAZA DE EXECUŢIE NR. 2

CU TITLUL Adnotarea datelor si proiectarea modulelor de detectare automata a subiectelor si categoriilor.

Avizat,

Coordonator	**Agent Economic**
Denumire: Universitatea din Bucuresti	**Denumire:** Pluriva SRL
Reprezentant Legal: prof. dr. Mircea Dumitru	**Reprezentant Legal:** Marius Pascu
Semnătură:	**Semnătură:**
Ștampilă:	**Ștampilă:**

Director Proiect: prof.dr. Liviu P. Dinu

Semnătură:

Responsabil de proiect: Marius Pascu

Semnătură:

Raport stiintific
privind implementarea proiectului in perioada ianuarie – decembrie 2017

Titlu Proiect: *Sistem Inteligent de Generare Automată a Răspunsurilor (SIGAR).*

SIGAR: Project 53BG/2016, funded by Romanian National Authority for Scientific Research and Innovation, CNCS/CCCDI – UEFISCDI, PNCDI III (Programul 2 - Creşterea competitivităţii economiei româneşti prin cercetare, dezvoltare şi inovare. Transfer de cunoaştere la agentul economic "Bridge Grant")

Durata Proiectului: 1 octombrie 2016- 30 septembrie 2018

Director: Prof. univ. dr. Liviu P. Dinu
Contractor: Universitatea din Bucuresti
Etapa: 20 decembrie 2017 (unica)

Obiectivul principal al proiectului:

Principalul obiectiv al proiectului constă în implementarea unei soluții bazate pe învățare automată, care va parsa și analiza baza de date de tichete în vederea construirii unor răspunsuri automate la întrebări sau probleme frecvent întâlnite.

Rezumat Faza 2017:

Obiectivul proiectului pentru anul 2017 este:

1. **Prelucrarea datelor**. **Adnotarea datelor si proiectarea modulelor de detectare automata a subiectelor si categoriilor**
   Pentru îndeplinirea acestui obiectiv, au fost planificate si executate următoarele activități:

   Activitate 2.1***: Adnotarea datelor***. În scopul realizarii acestei activități următoarele acțiuni au fost planificate și realizate:
   - Curatare si editare date Pluriva;
   - Adnotare date pe categorii;
   - Determinarea raspunsurilor prototip.

   Activitate 2.2: ***Activitati diseminare***. Principalele acțiuni desfășurate în cadrul acestei activități au fost participarea la conferinte, efectuarea unor stagii de cercetare, diseminarea rezultatelor in intalniri formale si informale, in organizarea unui seminar de cercetare, etc.

Activitate 2.3: ***Clasificare automata a subiectelor pe categorii (classification by topic).*** În scopul realizarii acestei activități următoarele acțiuni au fost planificate și realizate:

- Proiectare algoritm;
- Antrenare, testare, evaluare.

Activitate 2.4: ***Proiectare modul 1.*** Principalele acțiuni desfășurate în cadrul acestei activități au fost analiza, testarea, si evaluarea subclasificatorilor.

Activitate 2.5***: Dezvoltare agregator.*** Principalele acțiuni desfășurate în cadrul acestei activități au fost analiza, testarea, si evaluarea agregatorului.

Activitate 2.6: ***Proiectare si dezvoltare retea neuronala.*** Principalele acțiuni desfășurate în cadrul acestei activități au fost analiza, testarea, evaluare retelei precum si integrarea retelei in agregator.

Pe parcursul acestei etape, una din principalele activități a constat în extragerea, prelucrarea și filtrarea datelor agentului economic. S-au executat operațiuni de extragere din baza de date în format text pentru a fi cât mai ușor procesate folosind algoritmi de învățare automată. În baza de date sunt salvate întrebări și răspunsuri în format HTML brut, elemente extrase din fie mailuri, fie din soft-ul produs Pluriva. Întrebările și răspunsurile sunt catalogate în tabele de task-uri și activități. Un task reprezintă crearea unui tichet de lucru de către un client sau de către un angajat intern, task-urile de cele mai multe ori sunt definite de întrebări propriu-zise. Activitățile sunt date de alocarea task-urilor de către angajați și înregistrarea activității corespunzătoare taskurilor.

Pentru clasificarea automată a subiectelor pe categorii, am ales un număr restrâns de categorii (topics) extrase printr-un proces hibrid care presupune executarea unui algoritm de topic modelling și clusterizarea task-urilor și selectarea manuală a categoriilor. O inovare pe care am adus-o în modelarea categoriilor (topic modelling) a constat în utilizarea reprezentărilor vectoriale ale cuvintelor (word embeddings) în extragerea categoriilor

Proiectarea modulului 1 s-a bazat nu doar pe colectarea unui set de răspunsuri prototip, dar și o bună parte din cunoștințele dobândite la clasificarea de task-uri au contribuit la deciziile luate cu privire la clasificatori.

Metodele de scoring sunt agregate folosind un optimizator cu stol de particule. Regresorul și metodele de scoring prezic pentru un text dat la intrare o listă de ranguri corespunzătoare cu răspunsurile prototip. Agregarea se face prin ponderarea vectorilor prezișt astfel încât rata erorii să fie minimizată.

Ca rețea neuronală am folosit o arhitectură tip encoder-decoder, care pentru o secvență de cuvinte dintr-un task, construiește o reprezentare vectorială care mai apoi este folosită pentru a genera cuvinte din răspuns.

Rezultatele activitatii in cadrul proiectului in perioada 1 ianuarie – 20 decembrie 2017

Descrierea ştiinţifică şi tehnică, cu punerea în evidenţă a rezultatelor fazei și gradul de realizare a obiectivelor (se vor indica rezultatele)

În cadrul acestui proiect ne-am propus să eficientizăm procesul de răspuns al angajaților pentru task-urile primite. Așadar, din task-urile primite, a fost nevoie să implementăm un proces semi-automat de detecție și filtrare a task-urilor care conțin un mesaj relevant, fie el acesta o întrebare formulată, descrierea unei probleme sau trimiterea unor erori de sistem. Totalul de task-uri finalizate din baza de date însumează aproximativ 100,000 de intrări care au înregistrate activități. Din acest total, un număr semnificativ de task-uri este dat de mesaje automate de eroare înregistrate în sistem prin API. Pentru a elimina zgomotul, am implementat un proces semi-automat de filtrare care clasifică între emailuri sau întrebări formulate și mesaje automate de eroare. Pentru clasificare am folosit trăsături booleene precum existența unui antet / subsol, prezența unui destinatar, mesaje de întâmpinare specifice, prezența elementelor de cerere, rugăminte, formule de politețe etc. Mai mult decât atât, am detectat și prezența unui număr semnificativ de task-uri duplicate, probabil introduse din greșeală, pe care le-am elminat folosind un detector de duplictate text bazat pe o similaritate folosind n-grame de caractere. În urma acestor procese am putut elimina pe cât de mult zgomotul, obținând task-uri care conțin doar informații relevante pentru a fi introduse în modelele de procesare automata de text. O parte semnificativă de task-uri au fost eliminate și manual în momentul în care personalul responsabil a inspectat calitatea corpusului. Numărul total de task-uri a fost restrâns la 52,776, fiecare salvat în câte un fișier individual. Cât despre activități, am ales să selectăm doar acele activiăți care corespund listei finale de task-uri filtrate. Activitățile au fost salvate în fișiere individual sub format idTask_idActivitate. Activitatea 2.1, de extragere și curățare a datelor a fost una extrem de dificilă datorită tipurilor de date variate și nestructurate existente în baza de date.

Pentru determinarea răspunsurilor prototip, a fost necesară o analiză detaliată a tipurilor de text disponibile în corpus. Au fost analizate întregul set de răspunsuri și activități înregistrate pentru fiecare task. Fiecare task are cel puțin o activitate înregistrată, ceea ce face ca numărul total de activități să fie aproape dublu față de cel al task-urilor, ajungând undeva la 100,000 de activități înregistrate. În primă fază am ales să grupăm activitățile / răspunsurile după similarite în 300 de grupuri folosind un algoritm de clustering bazat pe frecvențele cuvintelor. Analiza pe care am efectuat-o a fost una preponderent manuală, trecând prin fiecare cluster manual și inspectând datele prezente acolo. La fiecare parcurgere au fost marcate răspunsuri recurente care se regăsesc în date. Analiza noastă a arătat că numărul total de răspunsuri prototipice pe care le putem întâlni este constrâns de varietatea problemelor pe care le întâlnesc agenții. Mai exact, marea majoritate a task-urilor conțin o cerință de rezolvare a unei probleme, fiind de cele mai multe ori specifică doar acelui client care înregistrează tichetul. Așadar soluția transmisă unui client nu este neapărat aplicabilă în general. Apoi de cele mai multe ori, un client înregistrează un task care necesită execuția unor pași, linii de cod, implementare, ștergere sau ajustare manuală de către agent, iar răspunsul la astfel de cereri este de tipul ”am rezolvat, s-a rezolvat / am corectat / s-a remediat etc.”. Corpusul de răspunsuri este acoperit în proporție de 86% de răspunsuri de acest tip, unele conținând nu doar confirmarea rezolvării, ci și pașii efectuați. Celelalte răspunsuri prototip includ: acordare de drepturi utilizatori, resetare parolă, preluare cerere, cerere de detalii, operații (update, ștergere), faptul că s-a răspuns prin email, task închis, cerere de verificare, formule de politețe sau altceva. În timpul acestui proces am depistat existența unui vocabular care nu se regăsește în normele limbii române făcând întreaga analiză

cu atât mai dificilă cu cât resursele existente pentru limba română (e.g. corpora sau dex) au aplicabilitate restrânsă în analiza textelor din baza de date Pluriva. Această constatare s-a dovedit extrem de utilă în luarea unor decizii pentru proiectarea și dezvoltarea activităților ulterioare.

Pentru clasificarea automată a subiectelor pe categorii, am ales un număr restrâns de categorii (topics) extrase printr-un proces hibrid care presupune executarea unui algoritm de topic modelling (Latent Dirichlet Allocation conform Activității 1.2 din etapa 1 a proiectului) și clusterizarea task-urilor și selectarea manuală a categoriilor. O inovare pe care am adus-o în modelarea categoriilor (topic modelling) a constat în utilizarea reprezentărilor vectoriale ale cuvintelor (word embeddings) în extragerea categoriilor. Am creat un spațiu de reprezentări ierarhice pentru cuvinte, task-uri și categorii pe care le-am utilizat pentru dezvoltarea clasificatorului. Astfel că am antrenat reprezentări vectoriale ale cuvintelor (word embeddings) pe corpusul de text conținând doar task-uri. Pentru acestea am folosit algoritmul CBOW (contextual bag of words), cu sampling negativ, o fereastră de 10 cuvinte și o dimensiune a reprezentărilor de 200 de cuvinte. Pentru fiecare cuvânt dintr-un task, extragem reprezentările, iar centroidul acestor reprezentări este considerat a fi reprezentarea vectorială a task-ului. Acestea pot fi introduse într-un algoritm de clusterizare, folosind centroizii, cel mai la îndemână este algoritmul k-means cu inițializare k-means++. După aplicare algoritmului obținem taskuri clasificate pe categorii. Reprezentarea unui task este dată de centroidul reprezentărilor cuvintelor, iar categoriile sunt date de centroidele reprezentărilor clusterelor de task-urilor. Obținem un proces ierarhic cu care nu doar că putem clasifica documentele în funcție de categorii, dar poate fi folosit și în mod constructiv pentru a extrage categoriile și cuvintele reprezentative din categorii. Avantajul reprezentărilor vectoriale constau în faptul că acestea depind doar de corpusul dat, incluzând forme lexical, expresii sau simboluri care sunt specifice doar în corpusul de antrenare. Pentru ca reprezentările să se poată antrena / construi, este necesar ca elementele lexicale să se regăsească într-un context din corpus de un număr minim de ori (în cazul nostru am ales minim 5). Vocabularul total atât pentru task-uri cât și pentru activități este redat în tabelul următor:

	Taskuri	**Activități**
Nr. de cuvinte	7,297,400	11,370,417
Nr. de tipuri	4,425,651	4,439,299
Varietate lexicală	0.6065	0.3904

În comparație cu algoritmul LDA utilizat în etapa 1, această metodă permite extragerea categoriilor din corpus și a documenteler ce aparțin fiecărei categorii, totul printr-o singură execuție. Un alt avantaj este faptul că reprezentările vectoriale se pot extinde și prin alte tipuri de trăsături numerice. În cazul nostru am ales să extindem reprezentările cu unigrame și bigrame de cuvinte, procesate prin ignorarea diacriticelor. De asemenea, frecvența minimă permisă în cazul acesta a fost de 2 bigrame, iar numărul maxim de trăsături nu a fost restricționat. Formulele

pentru tf-idf folosite includ adaugarea unui smoothing de 1 și logaritmarea frecvențelor n-gramelor prin: idf(d, t) = log [(1 + n) / (1 + df(d, t))] + 1 și tf(t) = 1 + log(tf).

Evaluarea și testarea metodei a implicat un proces manual îndelung prin care s-a căutat un număr optim de categorii de selecție (în cazul nostru, 300) pentru ca fiecare categorie să conțină cât mai multe documente cu grad mare de similaritate. La inspectarea mai atentă a clusterelor și a elementelor / cuvintelor cheie reprezentative din fiecare categorie, putem vedea că similaritatea cosine dintre centroidul task-urilor și cuvintele din vocabular redau exact problemele sau cerințele expuse de către clienți în cereri:

Cuvinte: "factura", 0.7177397012710571, "imaginea", 0.7047235369682312, "varianta", 0.6804873943328857, "facture", 0.6714362502098083, "sistem", 0.6646356582641602, "inregistrare", 0.6633201241493225, "plaja", 0.6615076065063477, "tot", 0.6604551076889038, "facturii", 0.6583608388900757, "informatia", 0.6564807891845703

Cerinte: „Va rog sa modificati tipizatul nota de comanda pe factura externa client", „va rog sa verificati la prima coloana din factura la NR.CRT", „va rog sa verificati facture cu nr 1743 furnizor", „Va rog sa ma ajutati prin configurarea modulului de intrari facturi in acest sens", etc

Toate cerințele care sunt despre operații și procedee pe facturi au fost acoperite de această categorie. O altă categorie de cerințe sunt acelea care invocă crearea de noi utilizatori, resetare parolă și alte operațiuni similare:

Cuvinte: "parola", 0.7102556228637695, "mine", 0.6828228235244751, "accesul", 0.6566786766052246, "dumneavoastra", 0.648627758026123, "nesoldatele", 0.6417368650436401, "dvs", 0.6382433176040649, "colegilor", 0.6268476843833923, "cererile", 0.6255913376808167, "Dvs", 0.6226714849472046, "cumva", 0.6205050945281982

Cerinte: „te rog sa creezi urmatorul user:", „Va rog sa ii faceti cont de email pentru", „sa mi se comunice parola pe mailul personal", „Va rog sa alocati user nou pentru acces retea si internet", etc.

La fel, și în acest caz categoria aceasta a acoperit task-urile care conțin cerințe cu privire la crearea de utilizatori, resetare parolă etc.

Metoda de detectare categorii este nouă prin felul în care am implementat-o și prezintă destul potențial pentru a putea fi implementată în producție. Pe măsură ce task-uri noi sunt introduse în sistem, acestea pot fi alocate cu o categorie în mod automat astfel încât repartizarea lor să fie făcută în funcție și de capacitatea echipei care trebuie să le rezolve. Adăugarea câtorva potențiale etichete pe fiecare task, ar putea duce la timpi mai scurți de rezolvare a unor taskuri.

Proiectarea modulului 1 s-a bazat nu doar pe colectarea unui set de răspunsuri prototip, dar și o bună parte din cunoștințele dobândite la clasificarea de task-uri au contribuit la deciziile luate cu privire la clasificatori. Scopul unui clasificator este să prezică care dintre cele 18 răspunsuri prototip este cel mai plauzibil pentru un task dat sau pentru o întrebare dată. În cadrul acestei activități am construit o serie de clasificatori care returnează o listă ordonată după rang pentru fiecare răspuns. Logistic regression, deși este un regresor mai mult decât un clasificator, s-a dovedit o alegere bună pentru multe cazuri de clasificare text, de la identificarea autorului sau a limbii mamă a unui autor non-nativ până la clasificarea limbii unui text. Această metodă de scoring, deși simplă ca principiu, poate fi antrenată pentru rezultatele cele mai bune prin alegerea unor trăsături numerice optime extrase din texte. Dată fiind eficiența dovedită la clasificarea de taskuri pe categorii a ponderilor tf-idf, am decis să le extragem și la acest pas pentru a le folosi ca trăsături. Aceste ponderi sunt utile când avem texte de lungime variabilă cu cuvinte cheie foarte specifice cum este cazul nostru. Inspirați de rezultatelel bune obținute cu word embeddings pentru activitatea 2.3, am experimentat și cu regresori care folosesc reprezentări vectoriale ale cuvintelor combinate cu tf-idf. Avantajul regresorului este că ne poate oferi un scor de încredere corespunzător fiecărei alegeri pe care o face, astfel obținând o listă de ranguri pentru fiecare răspuns prototipic potențial. Metoda folosită este one-vs-the rest, cu penalizare l2 și

implementare liblinear. De asemenea, am ales să echilibrăm numărul de exemple disponibile din fiecare clasă pentru a nu introduce bias către clasa majoritară, care oferă răspunsul ”am rezolvat” de fiecare dată. Echilibrarea se face prin random sampling din fiecare clasă la dimensiunea celei mai mici clase. Un alt pas pe care îl facem în acest caz este să extrage mai multe seturi de date (10) cu random sampling și să antrenăm 10 regresori iar predictia finală fiind dată de centroidul predicțiilor acestora.

Un alt clasificator pe care l-am folosit este XGBoost, clasificator bazat pe gradient boosting trees, s-a dovedit extrem de performant în multe competiții de machine learning. Funcția obiectiv este multi-class softmax cu o adâncime a pădurii maxime de 12 și ponderea minimă pe fiu 3. Acest clasificator returnează clase în format numeric reprezentând răspunsurile prototip și este de asemenea antrenat pe date echilibrate, doar că sampling-ul este aplicat numai o singură dată.

Un al treilea clasificator folosit este bazat pe nuclee de șiruri de caractere (string kernel). Acest clasificator s-a dovedit extrem de eficient în foarte multe competiții de clasificare a textelor scurte sau de identificare a nuanțelor prezente în texte. Clasificatorul se bazează pe mașini cu vector suport (SVM) și acest kernel pre-antrenat pe corpus. Kernel-ul este calculat ca suma n-gramelor de caractere comune dintre două texte, unde n-gramele acoperă un spectru de la bigrame la 4-grame. Dezavantajul principal este lipsa de eficiență a memoriei și a timpului, pentru fiecare exemplu de test trebuie re-calculat kernelul. Cu toate acestea, este una dintre metodele cele mai eficiente pentru predicții. Datele de antrenare, din nou, sunt extrase cu un singur random sampling echilibrat iar predicțiile sunt făcute la nivel de clasă, nu de probabilități asignate fiecărei clase.

Următoarele metode nu sunt clasificatori per-se, ci mai mult niște metrici care pot fi folosite pentru a extrage o listă de ranguri din predicții. Acestea merg pe presupunerea că textul de la input trebuie să conțină o informație similară cu textul de la output, în special când este vorba de descrierea unei probleme. Am ales 3 metrici - una bazată pe suma de n-grame comune dintre input și output, refolosită de la calcularea kernelului. Aceasta poate fi utilă prin faptul că nu depinde de cuvinte, ci de n-grame de caractere, afixe, punctuație, separare sufix + prefix, rădăcini ale cuvintelor și alte secvențe similare. În același timp poate găsi similarități și acolo unde nu sunt, dat fiind că cuantifică subsecvențe de cuvinte. A doua metrică este bazată pe scorul BLEU folosit pentru evaluarea traducerilor automate. În cazul nostru, folosim calculul modificat al preciziei pentru unigrame, unigrame unice și bigrame între textul de la intrare și setul de răspunsuri prototip. Considerăm că metrica aceasta poate fi utilă în calculul similarităților pentru a acoperi suprapuneri de cuvinte și pentru a balansa efectul primei metrici. O ultimă metrică pe care am implementat-o este insiprată din rezultatele activității 2.3 cu care comparăm la nivel semantic input și output urmărind cosine similarity dintre centroidele reprezentării vectoriale ale task-ului și centroidele reprezentărilor vectoriale ale răspunsurilor prototip. Folosind proprietatea reprezentărilor de a îngloba elemente de semantică distribuțională, metrica ne oferă o aproximație a categoriilor aflate cerință față de cele din răspunsuri. Fiecare clasificator prezice o listă cu scoruri pentru cel mai similar răspuns prototip. Răspunsul este apoi comparat cu ce a zis agentul și se verifică dacă răspunsul prototip este derivat din cel al agentului. Evaluarea individuală a subclasificatorilor se face prin acuratețea acestuia dar și prin evaluarea manuală a preciziei primelor 3 răspunsuri prototip sugerat. Evaluarea clasificatorilor și a metodelor de scoring s-a făcut individual, dar va fi prezentată în secțiunile următoare care acoperă agregatorul.

Cele 6 metode de scoring sunt apoi agregate folosind un optimizator cu stol de particule. Regresorul și metodele de scoring prezic pentru un text dat la intrare o listă de ranguri corespunzătoare cu răspunsurile prototip. Clasificatoarele în schimb returnează o clasă fixă, așa că am ales să convertim predicția într-un vector one-hot unde 1 se află pe poziția corespunzătoare clasei. Practic rangul 1 este obținut de clasa prezisă iar celelalte valori au rangul 0. Agregarea se face prin ponderarear vectorilor prezișt astfel încât rata erorii să fie minimizată. Optimizatorul cu stoluri de particule funcționează pornind de la un spațiu dat de soluții, de preferat uniform distribuite iar fiecare soluție / particulă se ajustează atât individual pentru găsirea minimului cât și ținând seama de minimul pe care îl găsesc particulele vecine. Așadar, pornim cu un stol de 200 de elemente, timp de 1000 de epoci maxim care încearcă să găsească niște ponderi pentru fiecare clasificator și metodă de scoring astfel încât rata erorii să fie minimă. Algoritmul converge relativ repede și putem observa ce ponderi primesc metodele la fiecare rulare pe datele de dezvoltare. Datele de test, reprezintă un sample complet nou de perechi de task-uri și răspunsuri prototip, acest set nefiind folosit în antrenarea clasificatorilor. În plus, pe parcursul epocilor de antrenare am observat diferențe considerabile în acuratețea clasificatorilor în funcție de numărul de clase ales. Pentru a obține o calitate optimă, am ales să comasăm răspunsurile prototip pe categorii – soluție oferită, răspuns dat prin mail sau telefonic, procesare drepturi / utilizator / parola, testat, politețe. În felul acesta, dacă este necesar, putem antrena sub-clasificatori care să distingă într-un mod ierarhic între posibile răspunsuri din categoria soluție oferită, spre exemplu: *S-a rezolvat / Am rezolvat / Am corectat / Eroarea a fost rezolvată / S-a remediat / Problemă rezolvată / Importul a fost facut / Am făcut setările necesare.*
Acuratețea la nivelul categoriilor este încurajatoare, fiind redată în tabelul următor:

Metoda	**Acuratețe 10-fold cv**	**Acuratețe date de test**
Logistic Reg	79.89	74.15
XGBoost	77.0	73.04
SVM + Kernel	78.25	73.33
n-gram metric	-	37.5
BLEU	-	41.0
Cosine emb.	-	44.75
Agregator	84.07	81.2

Inspecția manuală a datelor de test a arătat că dacă agregatorul prezice greșit (pe prima poziție ca rang) răspunsul care trebuie dat, atunci răspunsul corect se va găsi în primele 5 poziții. Ceea ce poate fi interpretat ca la o predicție de 5 posibile răspunsuri oferite unui agent, acesta să-și poată

alege cu o șansă ridicată un răspuns pe care să-l trimită mai departe. Conform tabelului, observăm că metricile de scoring nu au o acuratețe foarte mare, cum este și normal, dar conform optimizatorului, acestea ajută la îmbunătățirea rezultatelor per-total. De asemenea, merită menționat și faptul că metoda nu ține cont de trăsături sau factori specifici limbii române, deoarece aceste aspecte urmează să fie implementate în etapa 3. Credem că acuratețea totală poate fi îmbunătățită cu resurse specifice limbii. Conform rezultatelor, această metodă reprezintă un punct bun de pornire pentru eficientizarea procesului de răspuns pentru tipul de date cu care am lucrat.

Ca rețea neuronală am folosit implementarea din (Nisioi et al., 2017) publicată pentru un task de simplificare text. Ideea din spatele acestei rețele este de a construi o arhitectură tip encoder-decoder, dotată cu atenție globală și input feeding, care pentru o secvență de cuvinte dintr-un task, construiește o reprezentare vectorială care mai apoi este folosită pentru a genera cuvinte din răspuns. În cazul acesta, răspunsurile prototip nu sunt necesare, deoarece sperăm ca rețeua să poată genera răspunsul potrivit din datele de antrenare. În plus rețelele encoder-decoder au capacitatea de a da răspunsuri generale, lucru care nu este întotdeauna de folos pentru situații specifice. În schimb, în cazul nostru, considerăm că răspunsurile sau sugestiile generale de răspuns sunt singurele plauzibile, ținând cont de varietatea lexicală, stilistică și de subiecte care se regăsește în datele Pluriva. În plus, am arătat că rețeaua poate fi utilizată cu succes pentru simplificarea propozițiilor și extragerea unui conținut relevant, implicit un astfel de proces ar putea îmbunătăți calitea sistemeleor de procesare text existente. Antrenarea rețelei necesită resurse de calcul intensive și nu se poate face cross-validare, cu atât mai puțin optimizarea cu stoluri de particule, de aceea, pentru evaluare, am preferat să intrgrăm rețeaua în agregator la final și să comparăm rezultatele doare pe datele de test. Integrarea presupune ca decoderul să ofere un scor în plus pentru predicțiile agregatorului. Pentru datele de testare, nu am putut observa o îmbunătățire considerabilă a agregatorului prin utilizarea rețelei.

Evaluarea rețelelor de dialog sau pentru întrebări răspunsuri este încă un subiect larg dezbătut în comunitatea academică. Rețeaua a fost evaluată manual prin verificarea output-ului de către echipa de lingviști. Verdictul acestora a fost că rețeaua este capabilă să dea răspunsuri parțial gramaticale, de multe ori având dificultăți din cauza formei nestandardizate a corpusului. Astfel că rețeaua întâmpină dificultăți la antrenare atunci când în același context întâlnește cuvinte scrise greșit, cuvinte cu diacritice sau cu diacritice parțiale, și cuvinte lipsite de diacritice. Mai mult, utilizarea prescurtărlior și a formelor nestandardizate ale unor cuvinte din limba română face cu atât mai dificilă antrenarea în parametri optimi a unei rețele sequence to sequence. De aceea a fost imperios necesar să analizăm structurile lexicale prezente în corpus în diverse contexte prin antrenarea unor rețele de CBOW pentru a urmări reprezentările vectoriale ale cuvintelor și cantitatea de cuvinte care se regăsesc în dicționarul standard al limbii române. Tabelul următor prezintă gradul de suprapunere între lexiconurile folosite în task-uri, activități și un corpus conținând și text în limba română – Common Crawl.

	Task-uri	Activități	
DEX diacr. suprap	41.75	40.65	
DEX n. diacr. suprap	55.51	52.96	suprapuneri fără diacritice
Activități suprap	67.87	-	
Activități dif. En	4.83	-	nu apar în DEX, dar sunt în Engleză
Task-uri suprap	-	58.34	
Task-uri dif. En	-	20.95	nu apar în DEX, dar sunt în Engleză

Observăm că atât task-urile cât și activitățile au un vocabular din care maxim 55% din cuvinte se regăsesc în dicționarul explicativ. O analiză mai atentă arată că de fapt un număr considerabil de cuvinte sunt folosite ca prescurtări. Un rezultat preliminar, așadar, folosind anumite tipuri de word embeddings constă în faptul că putem identifica, după context, cuvântul sursă din care este derivată prescurtarea.

word	Q/A model	score
pentru (for)	pt	0.85
	Pentr	0.74
	ptr	0.64
ex (for example)	exemplu (example)	0.77
	Ex	0.65
	Exemplu	0.65
	adica (which means)	0.6
banca (bank):	registru (register)	0.79
	numerar (cash)	0.78
	casa (cash desk)	0.73
	plati (payments)	0.73
factura (invoice)	facture	0.84
	comanda (order)	0.77
	fact	0.76
	fct	0.76
	fc	0.72

Activități administrative și de organizare

Membrii echipei s-au întâlnit periodic pentru a prezenta și discuta aspecte legate de organizare, de repartizarea temelor de cercetare, de colaborare între membrii echipei, de stabilirea conferințelor și jurnalelor în care se trimit spre publicare rezultatele cercetării și de convenirea asupra unui calendar comun cu respectarea termenelor de atingere a obiectivelor proiectului și de livrare a rezultatelor. Intre membrii proiectului si partenerul economic a fost o continua comunicare. Studentii membri ai proiectului au facut practica la sediul agentului economic si s-au deplasat la sediul acestuia ori de cate ori a fost nevoie.

Au fost îndeplinite astfel toate activitățile administrative necesare bunei desfășurări a proiectului.

Activități de diseminare

În aceasta etapa, membrii proiectului au reușit să publice 5 lucrări la unele din cele mai importante conferințe dedicate procesării limbajului natural și lingvisticii computaționale (1 articol in conferinta ACL2017, conferinta A*, cea mai importanta din domeniu, doua articole in conferinta CICLING, categorie B, 1 articol intr-un workshop asociat unei conferinte A (EMNLP) si un articol intr-o conferinta importanta din domeniu cu larga audienta, CLEF, categorie C), au participat la stagii de cercetare, au sustinut mai multe conferinte in tara si strainatate, au organizat prima conferinta internationala RAAI 2017 (Recent Advances in Artificial Intelligence). De asemenea, în acest an a fost continuat în cadrul proiectului seminarul bilunar de lingvistică matematică și computațională, pe parcursul căruia a continuat seria de prezentari inceputa in 2016. În cadrul prezentărilor participă cu regularitate atât membrii proiectului, cât și masteranzi, doctoranzi sau postdoctoranzi din Departamentul de Informatică, din alte departamente ale Universității din București, din cadrul Facultății de Automatică a Universității Politehnica din București, membri ai partenerului Pluriva, dar și alți colegi din industrie interesați de tematica abordată. Considerăm că prin această activitate aducem un beneficiu proiectului prin activitatea de diseminare, si, în același timp realizam noi punți de comunicare

între mediul academic și industrie. Alte articole au fost trimise spre publicare, articole care se afla in diverse faze de evaluare. De asemenea, a fost întreținută și actualizată pagina oficială a proiectului (http://nlp.unibuc.ro/projects/sigar.html), astfel încât toate rezultatele așteptate la finalul acestei etape (pagina web, articol trimis spre publicare, raport, analiza datelor) au fost realizate.

Concluzii

- Obiectivele etapei au fost substanțial realizate.
- Este necesară continuarea cercetării fără modificări în planul curent de realizare.

Publicații și conferințe susținute de către membrii proiectului în anul 2017:

1. Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto and Liviu P. Dinu, 2017. *Exploring Neural Text Simplification Models.* In Proc. of the 55th Annual Meeting of the Association for Computationa Linguistics, ACL 2017, p 85-91, Vancouver, Canada, July 30 - August 4.

2. Alina Ciobanu, Liviu P. Dinu, and Andrea Sgarro, 2017. *Towards a Map of the Syntactic Similarity of Languages*. In Proc18th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing 2017), Budapest, Hungary, Aprile 17-23, 2017(proceedings in curs de aparitie la Springer LNCS)

3. Alina Maria Ciobanu, Liviu P. Dinu, 2017. *Romanian Word Production: an Orthographic Approach Based on Sequence Labeling*. In Proc18th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing 2017), Budapest, Hungary, Aprile 17-23, 2017(proceedings in curs de aparitie la Springer LNCS)

4. Marcos Zampieri, Alina Maria Ciobanu, Liviu P. Dinu. *Native Language Identification on Text and Speech*. Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications (co-located with EMNLP 2017), pages 398–404 Copenhagen, Denmark, September 8, 2017.

5. Alina Maria Ciobanu, Marcos Zampieri, Shervin Malmasi, Liviu P. Dinu. *Including Dialects and Language Varieties in Author Profiling.*Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum, Dublin, Ireland, September 11-14, 2017, {CEUR} Workshop Proceedings, vol. 1866

Director Proiect,
Prof. dr. Liviu P. Dinu