UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

LUCRARE DE DISERTAȚIE

Agent conversațional

Student:

Ionuț Ciocoiu

Coordonator științific:

Prof. Dr. Liviu P. Dinu

IUNIE 2019

Cuprins

Listă de figuri

Lista porțiunilor de cod

Capitolul 1

Introducere

Privind dincolo de utilitatea practică de a transmite informații, limbajul reprezintă principala noastră unealtă în evoluție și nu numai, el este cel care intensifică interacțiunea și menține legăturile interumane.

"Vorba dulce mult aduce" surprinde în esență trăsături definitorii precum puterea cuvântului, capacitatea afectivă și în principal caracterul contemplativ al limbajului. El în sine reprezintă o formă de artă, paradoxală și recurentă, poate din cauza faptului ca ne reprezintă ajunge să se confunde cu noi.

În cartea sa Speach and Language Proccessing [2], profesorul Dan Jurafsky observă faptul că există numeroase exemple în care omul încearcă să comunice cu creația sa, acest comportament este surprins adesea și în literatură unde personificarea nu se oprește numai la natură. În lumina acestor observații se remarcă longevitatea și neoboseala dorinței creatoare concretizată în această ramură a lingvisticii matematice unde puterea de a înțelege limbajul este folosită la implementarea agenților conversaționali.

1.1 Nevoia de a comunica

Privind comunicarea ca o nevoie de bază și limbajul ca un prim instrument al său, vedem de ce domenii științifice precum procesarea limbajului natural și lingvistica matematică reprezintă elemente cheie ale accesului nostru la informație, mai presus de atât curiozitatea stârnită aici ne poartă de la înțeles la autocunoaștere.

Stilul nostru de viață se schimbă odată cu tehnologia, acest dinamism ne învăluie în straturi de informații ce dau naștere unor probleme de căutare, structurare și reprezentare a informației. Nevoia de a comunica cu un astfel de sistem se dorește a fi cât mai naturală, dat fiind rapiditatea cu care aceste inovații intră în contact cu oamenii. Gradul de eficiență și acces la informație cerut de generațiile tinere este în continuă creștere, iar acest lucru nu face doar să sporească nevoia de înțelegere a limbajului natural și menținerea unui dialog cât mai firesc între om și mașină.

Actualele modele de a structura informația se dovedesc a fi instrumente bune la stocarea și procesarea acesteia. Pe când reprezentarea și inferența sunt două concepte aduse în prim plan de inteligența artificială care cere noi moduri de a rezona cu informațiile stocate.

Lucrarea își propune cercetarea unui model îndeajuns de puternic încât să înțeleagă limbajul natural exprimat într-un anumit context (general sau specific unui domeniu). Și punerea în aplicare a unui administrator de dialog, suficient de complex pentru a ține contextul unor sarcini bine definite.

NLU este un concept care poate îngloba o mulțime de probleme precum analiza de sentiment, clasificarea de text, rezumarea și multe alte chestiuni legate de semantică. În literatura de specialitate a sistemelor de dialog, NLU ține locul componentei ce este responsabilă cu detectarea intenției vorbitorului și recunoașterea entităților, aceste grupuri de cuvinte care au o însemnătate în cadrul unui domeniu. De exemplu: ”aș dori să îmi resetez parola, numele meu de utilizator este cioionut”, avem ca intenție resetarea parolei, iar ca entități numele de utilizator.

În administratorul de dialog, aceste intenții sunt văzute ca reprezentări ale obiectivelor utilizatorului. Odată ce intenția este stabilită, pentru a duce la bun sfârșit un obiectiv s-ar putea să mai avem nevoie de anumite informații așa cum în exemplu anterior am avut nevoie de numele de utilizator. Ceea ce face ca entitățile să fie văzute drept argumente ale unui obiectiv..

1.1.1 Contribuția noastră

Luând contact tot mai des cu mediul de cercetare, încerc să observ modul în care această comunitate reușește să aducă contribuții în societate. Un lucru care m-a făcut sa prețuiesc fiecare efort, este acela că în general o descoperire se bazează pe cercetări anterioare și ca orice gând exprimat în scopul de a descoperi, poate fi valoros și dus mai departe.

Dorința de a crea cu scopul de face viața oamenilor mai ușoară este imboldul intrinsec ce ghidează acțiunile mele, așadar evaluând cunoștințele mele, am decis sa îmi aduc contribuția într-un domeniu atât de important în drumul nostru spre o viziune clara.

Ideea de a cerceta un model pentru înțelegerea limbajului, vine în egală măsură din curiozitate și din conștientizarea nevoii. Abordarea propusă conține două module, unul pentru nlu și altul pentru dm, decizia care a stat la baza acestei arhitecturi este legată în primul rând de rigoarea impusă de un administrator de dialog separat în favoarea unei soluții end-to-end ce impune mulțimi de antrenare mult mai rafinate. S-a urmărit și posibilitatea de a pune în producție un astfel de sistem, dar și ușurința de a crea noi agenți doar adăugând date specifice domeniului.

Pentru înțelegerea limbajului natural

1.1.2 Progres precedent

Mergând pe acest drum al construirii unui sistem de dialog în literatura de specialitate se disting două ramuri: cea a asistenților digitali care rezolvă diferite cerințe (taskbots), în general conversația nu este menită dă dureze mult, iar rolul agentului este de a capta informații din conversația cu utilizatorul și de a-l ajuta să ducă la bun sfârșit o operație [3, 4]. În cea de-a doua direcție este vorba despre agenți conversaționali (chatbots), acest termen face referire la un agent orientat spre discuție la nivel general, servind mai de grabă unui scop subiectiv decât unuia obiectiv soluțional. În această situație dialogul poate dura mai mult, existând și aplicații practice ale acestuia cum ar fi cea de a testa anumite teorii psihologice [5]

Pornind de la dorința de a crea cu scopul de a aduce valoare în societate, am ales ca direcție de studiu sistemul de dialog orientat pe rezolvarea cerințelor. Tot mai des se încearcă metode end-to-end pentru a descoperi un singur model care să țină loc unui astfel de sistem [**?**]. În industrie lucrurile sunt abordate mai modular, antrenându-se modele diferite pentru fiecare componentă (ASR, SLU/NLU, DM, NLG). Un minimum viabil al acestei arhitecturi modulare îl reprezintă componenta de înțelegerea limbajului natural(NLU) și cea de administrare a dialogului (DM) acestea două făcând obiectul lucrării de față.

Componenta de înțelegerea limbajului natural se ocupă cu detectarea intenției și a entităților aflate într-o replică venită din partea utilizatorului. De multe ori cele două chestiuni sunt văzute separat, ca o problemă de clasificare respectiv ca o problemă de etichetare a unei secvențe. Pentru detectarea intenției se folosesc abordări precum [**?**]. Iar pentru recunoașterea entităților sunt folosite tehnici de etichetare ca [**?**]. Dar luând în considerare puterea rețelelor neuronal recurente și a arhitecturii de codificare-decodificare utilizată cu succes în traducere și recunoașterea vorbirii [**?**]. Aceste două probleme sunt tratate de multe ori concomitent, în esență se folosește capacitatea de a capta înțelesul unei secvențe de text cu ajutorul unui encoder iar apoi se folosesc diferite decodere pentru a traduce cuvinte sau întregul înțeles în clase de intenții și entități.

1.2 Rezumatul capitolelor

Introducere -

Capitolul întâi vorbește in principal despre modul în care această lucrare își propune să rezolve nevoia de interacțiune cu tehnologia, dar și despre un capitol istoric privit prin ochii unei motivații îndrăznețe.

Atunci când se rostește ”progres” am în minte o spirală a cunoștintelor care se bazează unele pe altele. Precum această imagine implementarea acestei tehnologi de dialog impune un anumit progres precedent, așa că în capitolul doi vor fi prezentate aceste instrumente care fac posibilă aceasta tehnologie.

In capitolul al treilea vor fi explicate modelele matematice care stau în spatele modelelor decizionale.

In partea a patra se prezintă modulul de înțelegerea limbajului natural

Al cincelea capitol descrie modulul care ține contextul unei conversații.

Iar in partea de final concluziile referitoare la studiul elaborat in această teză.

Capitolul 2

Dialog

Conform definiției din limba română, dialogul este modul de expunere care prezintă succesiunea replicilor dintr-o conversație care are loc între două sau mai multe persoane.

Această lucrare își propune să dea formă înțelegerii limbajului natural dintr-o perspectivă matematică, prezentând sub forma unei soluții programabile un întreg sistem de micro servicii toate funcționând sub umbrela aceluiași scop, comunicarea.

Pe parcursul lucrării se va face referire la dialog ca o secvență de replici între un om și un calculator pentru a transmite informații. Referitor la componentele unui dialog, se va prezenta doar o abordare bazată pe componenta *verbală*, celelalte componente *nonverbală* (gesturi, mimică, poziția corpului) și *paraverbală* (accentul, ritmul și intensitatea vorbirii) făcând obiectul altor lucrări viitoare.

2.1 Conversația naturală

Un factor cheie într-o conversație este acela că fiecare replică dintr-un dialog este o formă de **acțiune** venită din partea vorbitorului [6]. În literatura de specialitate **actele de vorbire** sunt cele ce dau tipul acestor acțiuni.

De-a lungul timpului s-au propus numeroase moduri de a grupa acte de vorbire. Iar modul de clasificare potrivit abordării de față conține doar 4 categorii și se con-

centrează pe intenția comunicată, conținutul propozițional și contextul de producere.

2.1.1 Acte de vorbire

1. **Reprezentative (asertive)** - sunt acele acte care definesc o constatare, cu ajutorul lor se comunică informații se impun constrângeri, iar la nivel de conținut putem spune că descriu realitatea. Se descriu anumite evenimente (”Coletul a fost livrat”), sau se impun anumite constrângeri (”Voi pleca maine”). Mărcile acestui act de vorbire sunt date de verbe precum: *a afirma, a admite, a anunța, a avertiza, a declara, a insista, a înștiința, a zice, etc.*

2. **Directive** - sunt actele ce urmăresc impunerea unei acțiuni. Scopul lor este de a determina participanții la dialog să execute o sarcină. Verbele marcante sunt *a întreba, a interzice, a ordona, a cere*

3. **Expresive** - reprezintă o manifestare în plan verbal a sentimentelor, emoțiilor și a atitudinilor vorbitorului cu referire la acțunile întreprinse sau nu de către interlocutor. Verbele reprezentative ale acestui act sunt: *a mulțumi, a lăuda, a felicita, a regreta, a reproșa, a critica, a acuza, etc*

4. **Comisive (promisive)** - vizează angajamentul locutorului de a da curs unor acțiuni/convingeri *a promite, a plănui, a jura, a paria, a se opune*

Pentru stabilirea intenției, actul de vorbire joacă un rol foarte important, iar sistemele de dialog extind aceste clasificări în intenții specifice domeniului cu scopul de a caracteriza cât mai bine conversația.

Exemplu de intenții si cum sunt ele văzute in menagerul de dialog.

2.2 Sistem de dialog

Doar componenta de înțelegere a limbajului și cea de administrare a dialogului vor fi analizate în cele ce urmează, întrunindu-se astfel minimul necesar unei conversații scrise.

Întregul proces este descris de diagrama din figura 2.1. Totul începe cu utilizatorul care formulează o cerere către sistem, vocea acestuia este procesată de către modulul de recunoaștere vocală care întoarce textul rostit de către vorbitor. Cererea în format text este apoi trimisă la modulul de înțelegere a limbajului care detectează la rându-i, intenția utilizatorului dar și entitățile menționate. Acest cadru semantic extras este apoi trimis modulului care ține firul dialogului și în funcție de definiția sarcinii pe care utilizatorul o dorește a fi îndeplinită se iau acțiunile în consecință și se trimite răspunsul către utilizator.

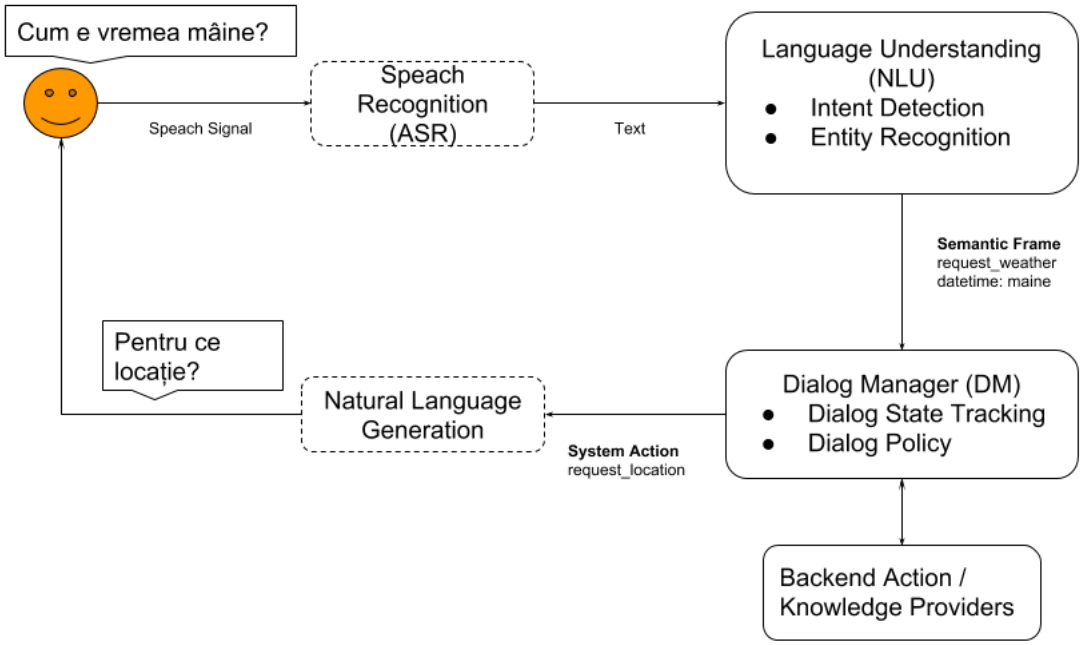

Figura 2.1: Sistem de Dialog Modular

Capitolul 3

Sistem de dialog

Construcția unui agent virtual menit să întrețină cursul unei conversații a fost întotdeauna un etalon al performanței de cercetare, de aceea testul ce poartă numele cercetătorului britanic Allan Turing **??** a fost pană de curând un criteriu în această direcție.

Au fost propuse diferite arhitecturi și moduri de a schița matematic o conversație, printre ele și ELIZA compus dintr-un set de reguli elaborate pe baza mai multor studii având la bază logica. Bineînțeles industria cere și abordări mai modulare, mai robuste dar care să nu iasă din tipare, în această speță făcându-și apariția prima arhitectură bazată pe umplerea de sloturi (GUS), adică pentru fiecare replică din dialog se extrag constituenți semantici specifici domeniului care mai apoi sunt completați într-o structură tabelară (frame) urmând să servească drept parametrii în interogările cu sistemul.

Dacă privim la scopul final al unei conversații distingem două clase și anume: agenți orientați pe rezolvarea de cerințe și agenți orientați pe discuție la nivel general.

Cum majoritatea asistenților virtuali sunt dezvoltați în scopuri comerciale, rezolvarea cerințelor primează în funcționalitatea unei astfel de aplicații, așadar industria se concentrează pe arhitecturi modulare, care să necesite cât mai puține date de antrenare, RASA, SNIPS, WATSON, DialogFlow sunt platforme care pun la dispoziție instrumente pentru a crea genul acesta de agenți.

Prin natura sa, lumea academică este întotdeauna mai în față în ceea ce privește tehnica sa de a concepe aceste sisteme, mai exact se folosesc rețele neurale recurente pentru a capta contextul unei replici și pentru a genera răspunsuri, iar pentru contextul conversației și generarea unor politici de răspuns se folosește învățarea prin recompensă. [**?**]

În cadrul studiului curent accentul este pus pe structura ce îmbină mai multe modele, drept pentru care vor fi detaliate experimentele pentru NLU și DM.

3.1 Înțelegerea limbajului natural

NLU este un concept destul de vast, in terminologia sistemelor de dialog acesta joacă rolul componentei care detectează intenția vorbitorului si extrage constituenți semantici din limbajul natural exprimat. Detecția intenției poate fi tratata ca o problema de clasificare a unei replici din punct de vedere semantic, iar recunoașterea entităților poate fi văzută ca o etichetare de secvențe.

3.1.1 Abordări anterioare

Pentru detectarea intenției se pot folosi tehnici standard de clasificare a unui text si anume SVM (Haffner et al., 2003), dar si rețele neurale convoluționale (CNNs) (Xu and Sarikaya, 2013) întâlnite adesea în procesarea imaginilor.

Pentru recunoașterea entităților se folosesc abordări precum (MEMMs) (McCallum et al., 2000), conditional random fields (CRFs) (Raymond and Riccardi, 2007), și rețele neurale recurente (RNNs) (Yao et al., 2014; Mesnil et al., 2015)

Având la baza înțelegerea limbajului, majoritatea lucrărilor actuale se concentrează pe rezolvarea concomitenta a acestor doua probleme, întrucât combinarea celor doua modele ajuta la învățarea unei reprezentări cat mai precise a textului. [**?**]

3.1.2 Model propus

Rețelele neuronale recurente (RNN) sunt unele dintre cele mai populare modele matematice folosite în procesarea limbajului natural, avantajul pe care acestea îl oferă se datorează într-o mare măsură formelor sale LSTM, GRU, dar și prin caracterul lor de a învăța din date cu caracter secvențial. Aceste tipuri de rețele sunt folosite adesea ca structuri de bază pentru a crea alte modele de calcul, un exemplu este rețeaua de codificare-decodificare (seq2seq), formată din două rețele recurente, una pentru captarea (codarea) înțelesului unei secvențe iar cealaltă pentru operația de decodificare într-o secvență de ieșire.

Modelul propus pentru înțelegere a limbajului rezolvă ambele probleme simultan folosind un codificator pentru reprezentarea înțelesului unei propoziții și două decodificatoare câte unul pentru fiecare chestiune. Figura 3.1 sintetizează procesele implicate în acest model.

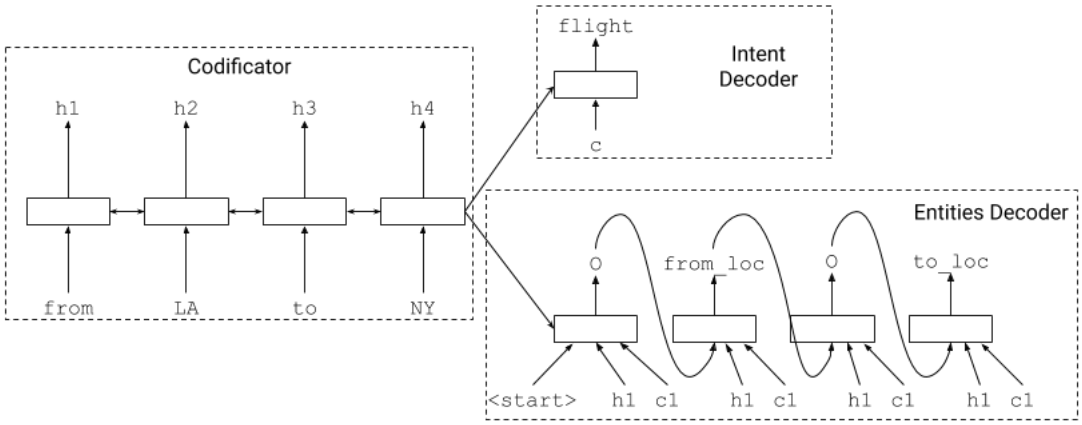

Figura 3.1: Seq2Seq-x

Pentru recunoașterea entităților se dorește etichetarea unei secvențe de cuvinte $x = (x_1, ..., x_T)$ într-o altă secvență de nume de entități corespunzătoare $y = (y_1, ..., y_T)$. Operația de codificare se realizează prin citirea întregii secvențe de către o rețea recurentă bidirecțională, înțelesul secvenței de intrare este reprezentat de suma celor două stări ascunse finale, adică pentru citirea de la stânga la dreapta obținem stările ascunse $(fh_1, ..., fh_T)$ și invers $(bh_T, ..., bh_1)$, apoi suma celor două $h_i = [fh_i + bh_i]$ este folosită drept codificare a înțelesului secvenței de intrare. Alte lucrări [3] folosesc

ca stare ascunsă finală h_i o concatenare a celor două stări. Alegerea de a aduna cele două reprezentări în schimbul concatenării, ține de faptul că dimensiunea reprezentării finale crește în cazul concatenării, fiind mai dificil de antrenat, dublând în acest caz și jumătate din parametrii rețelei. Figura 3.2 descrie procesul de codificare.

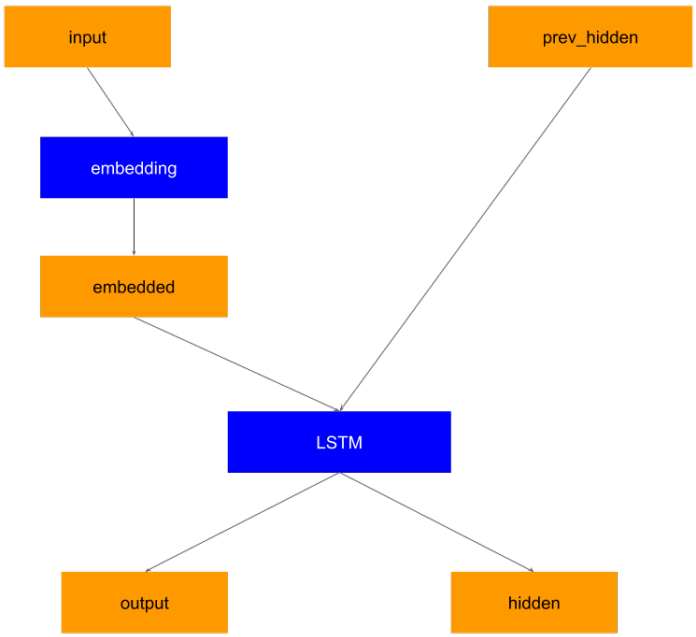

Figura 3.2: Bidirectional Encoder

Vom folosi reprezentarea captată de codificator pentru a inițializa operația de decodificare, aceasta este produsă de către o rețea recurentă unidirecțională, văzută ca o funcție $s_i = f(s_{i-1}, y_{i-1}, h_i, c_i)$ care la fiecare pas i, calculează o nouă stare s_i, bazată pe: starea anterioară a decodificatorului s_{i-1}, eticheta anterior prezisă, y_{i-1}, starea ascunsă din codificator corespunzătoare pasului curent, h_i și un vector de context c_i, calculat ca o sumă ponderată peste stările ascunse din codificator. Inspirat din [7] am experimentat mai multe modele de atenție.

$$c_i = \sum_{j=1}^{T} \alpha_{i,j} h_j$$

$$\alpha_{i,j} = \frac{\exp(score_{i,j})}{\sum_{k=1}^{T} \exp(score_{i,k})}$$

$$score(i,k) = g(s_{i-1}, h_k) = \begin{cases} s_{i-1}^\top h_k & \text{dot} \\ s_{i-1}^\top W_a h_k & \text{general} \\ v_a^\top \tanh(W_a[s_{i-1}; h_k]) & \text{concat} \end{cases}$$

Modelul de atenție poate fi văzut ca o rețea de tip feed-forward $g(s_{i-1}, h_k)$, care la fiecare pas din operația de decodare calculează ponderile pentru stările ascunse ale codificatorului, folosindu-se de ultima stare ascunsă din decodor, s_{i-1} și stările ascunse $(h_1, ..., h_T)$. Mai jos în figura 3.3 este descris întreg fluxul de date și operații realizate de decodorul responsabil de etichetarea entităților.

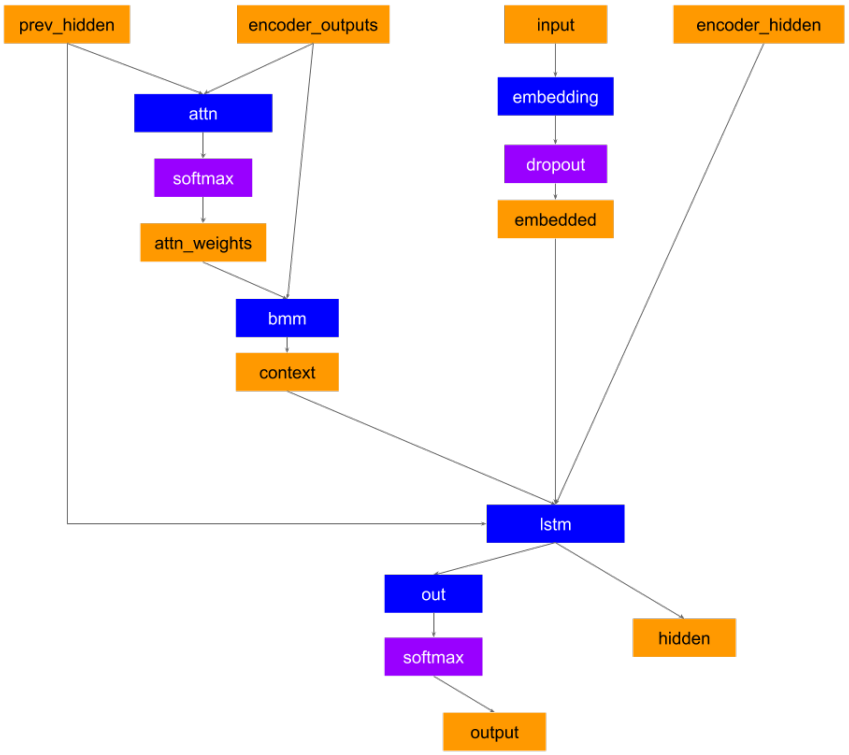

Figura 3.3: Attention Decoder

3.1.3 Mulțimea de antrenare

ATIS (Airline Travel Information Systems) [8] este o colecție de cereri venite din partea clienților referitor la zborurile aeriene de la începutul anilor 90. DARPA este instituția americana ce s-a ocupat de colectarea acestor date. Mulțimea de antrenare conține un număr de aproape 5000 de propoziții etichetate cu intenții si entități. Formatul

Tabela 3.1: Exemplu din mulțimea de antrenare

Propoziție	flights	from	New	York	to	Cicago	on	wednesday	morning
Entități	O	O	B-from.loc	I-from.loc	O	B-to.loc	O	B-depart.day-name	B-depart.day-mood
Intenție	flight								

Tabela 3.2: Statistici ATIS

Multime de antrenare	#Train	#Dev	#Test	$\|V\|$	#Intenții	#Entități
ATIS	4978	-	893	900	18	80

de etichetare utilizat este IOB (Inside–outside–beginning), care ajuta la etichetarea entitatilor formate din mai multe cuvinte, astfel încât fiecare etichetă este precedata de B-<nume-eticheta> pentru început, iar cu I-<nume-eticheta> pentru următoarele cuvinte care fac parte din entitate. Exemplele sunt date în limba engleză, iar lungimea medie a unei prepoziții este de 15 cuvinte.

3.1.4 Rezultate

Model	**Scor F1 (entități)**	**Scor F1 (intenții)**
Seq2SeqX (dot - attn)	97,5	96,5

3.1.5 Analiza erorilor

3.2 Administrator de dialog

3.2.1 Abordări anterioare

Capitolul 4

Noțiuni teoretice

4.1 Rețele Neurale Recurente

Rețelele neuronale recurente (RNN - Recurent Neural Networks), sunt o arhitectura aparte de rețele neuronale, ce le face atat de speciale este faptul ca ele reușesc sa capteze secvențialitatea datelor. Ele sunt folosite in special în procesarea limbajului natural, dar și în procesarea imaginilor, a seriilor de timp, a recomandarilor de produse. Cu alte cuvinte oricând vine vorba de succesiunea anumitor evenimente, ele reprezintă un candidat bun în captarea acestor modele in date.

Figura 4.1 prezintă arhitectura unei rețele recurente, unde fiecare dreptunghi ține locul stratului ascuns de la pasul, t. Fiecare strat ascuns este format din perceptroni care execută operația de inmulțire intre parametrii și input, urmată de o operație nonlineară (ex.$tanh$). La fiecare pas din timp, ieșirea de la pasul anterior, împreună cu vectorul următorului cuvânt, x_t, sunt intrări în stratul ascuns care produce pentru pasul următor, ieșirea y, și vectorul de caracteristici al stratului ascuns, h.

$$h_t = \sigma(W^{(hh)}h_{t-1} + W^{(hx)}x_{[t]}) \tag{4.1}$$

$$y_t = softmax(W^{(S)}h_t) \tag{4.2}$$

Descrierea parametrilor din rețea:

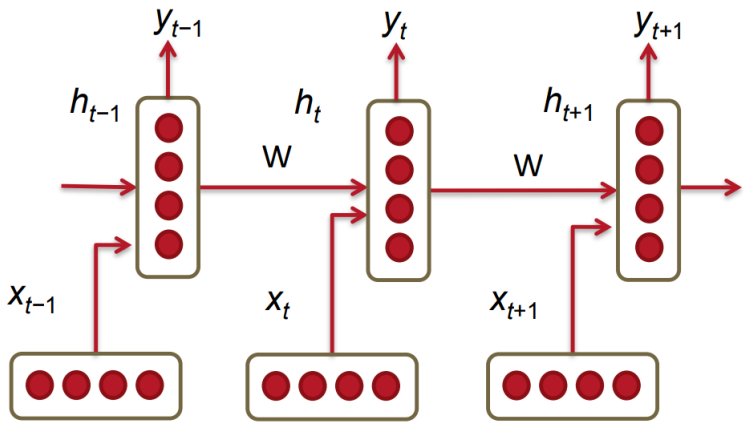

Figura 4.1: Arhitectura RNN [1]

- $x_1, x_2, ..., x_T$ vectorii cuvintelor dintr-o secvență de lungime T
- $h_t = \sigma(W^{(hh)}h_{t-1} + W^{(hx)}x_{[t]})$: formula care descrie calculul vectorului de caracteristici, h_t, la fiecare pas t:
  - $x_t \in \mathbb{R}^d$ vectorul cuvântului t
  - $W^{hx} \in \mathbb{R}^{D_h \times d}$ matricea proderilor utilizată la condiționarea vectorului de intrare, x_t
  - $W^{hh} \in \mathbb{R}^{D_h \times D_h}$ matricea proderilor utilizată la condiționarea ieșirii de la pasul anterior, h_{t-1}
  - $h_{t-1} \in \mathbb{R}^{D_h}$ ieșirea funcției nonlineare de la pasul anterior, $t-1$, iar $h_0 \in \mathbb{R}^{D_h}$ este vectorul de inițializare pentru stratul ascuns, la pasul $t = 0$
  - $\sigma()$ funcția nonlineară (sigmoid in acest exemplu)
- $y_t = softmax(W^{(S)}h_t)$ ieșirea care reprezintă o distribuție de probabilitate peste vocabular la fiecare pas t. În esență, y_t este următorul cuvânt prezis, dându-se contextul de până acum (h_{t-1}) și ultimul cuvânt observat (x_t). Avem $W^S \in \mathbb{R}^{|V| \times D_h}$ și $y \in \mathbb{R}^{|V|}$, unde $|V|$ reprezintă cardinalitatea mulțimii V adică a vocabularului de cuvinte.

Funcția de cost utilizată in rețelele neuronale recurente este de obicei CE (cross entropy error).

Pentru pasul t:

$$J_{(t)}(\theta) = -\sum_{j=1}^{|V|} ytrue_{t,j} \times \log(y_{t,j})$$

Pentru toată secvența de lungime T funcția de cost devine:

$$J = -\frac{1}{T}\sum_{t=1}^{T} J_{(t)}(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{|V|} ytrue_{t,j} \times \log(y_{t,j})$$

[1]

4.1.1 Problema dispariției sau a explodării valorilor gradientului

O RNN împarte aceeași matrice a parametrilor pentru toată secvența de intrare. Țelul unei astfel de arhitecturi este acela de a capta contextul și în cazul secvențelor de lungimi mai mari.

Considerăm ecuațiile 4.1 și 4.2 de mai sus la pasul t. Pentru a calcula eroarea unei RNN, dE/dW, vom însuma erorile de la fiecare pas de timp.

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W} \tag{4.3}$$

Eroarea la fiecare pas de timp este calculată prin diferențierea ecuațiilor 4.1 și 4.2:

$$\frac{\partial E_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t}\frac{\partial y_t}{\partial h_t}\frac{\partial h_t}{\partial h_k}\frac{\partial h_k}{\partial W} \tag{4.4}$$

Unde $\frac{dh_t}{dh_k}$ este derivata parțială a stării ascunse h_t în raport cu toți k pași de timp anteriori.

$$\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} = \prod_{j=k+1}^{t} W^T \times diag[f'(j_{j-1})] \tag{4.5}$$

Deoarece $h \in \mathbb{R}^{D_n}$, fiecare $\partial h_j/\partial h_{j-1}$ este matricea Jacobian pentru h:

$$\frac{\partial h_j}{\partial h_{j-1}} = [\frac{\partial h_j}{\partial h_{j-1,1}} ... \frac{\partial h_j}{\partial h_{j-1,D_n}}] = \begin{bmatrix} \frac{\partial h_j,1}{\partial h_{j-1,1}} & \cdots & \frac{\partial h_j,1}{\partial h_{j-1,D_n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_j,D_n}{\partial h_{j-1,1}} & \cdots & \frac{\partial h_j,D_n}{\partial h_{j-1,D_n}} \end{bmatrix} \tag{4.6}$$

Din 4.4, 4.5, 4.6 avem următoarea relație:

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} (\prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}}) \frac{\partial h_k}{\partial W} \tag{4.7}$$

Inecuația 4.8 arată norma matricei Jacobian, unde β_w și β_h reprezintă limitele superioare ale celor doua norme ale matricelor. Norma gradientului de la fiecare pas t este calculată în funcție de inecuația de mai jos.

$$||\frac{\partial h_j}{\partial h_{j-1}}|| \leq ||W^T|| ||diag[f'(h_{j-1})]|| \leq \beta_w \beta_h \tag{4.8}$$

Norma L2 este folosită în aceste inecuații. Iar norma derivatei $f'(h_{j-1})$ nu poate depăși valoarea 1, deoarece $f = sigmoid$.

$$||\frac{\partial h_j}{\partial h_k}|| = || \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}}|| \leq (\beta_w \beta_h)^{t-k} \tag{4.9}$$

Termenul $(\beta_w \beta_h)^{t-k}$ poate foarte ușor să devină foarte mic sau foarte mare, atunci când $\beta_w \beta_h$ este mai mare ca 1 și $t - k$ suficient de mare. Obținem valori mari pentru $t-k$ în cazul in care se dorește evaluarea funcției de criteriu pentru cuvintele (intrările) mai îndepărtate. Așadar contribuția intrărilor de la pașii inițiali se diminuează cu cât ne îndepărtăm în timp.

În timpul experimentelor s-a observat că odată ce valoare gradientului crește foarte mult, poate cauză erori de memorie (valoarea nu încape în tipul de date specificat) ceea ce rezultă în valori nedefinite (exemplu NaN (not a number)), lucru ușor detectabil la timpul rulării. Această problemă adresează explodarea valorilor gradientului(

Iar cea de-a doua chestiune, cea a diminuării valorii gradientului privește cazul în care valorile pentru gradient devin apropiate de zero, de multe ori acest comportament trece nedetectat, rezultând astfel o calitate mai slabă a învățării cuvintelor îndepărtate.

4.1.2 Soluții pentru dispariția/explodarea gradientului

Pentru a problema explodării gradientului, Thomas Mikolov, introduce pentru prima dată o euristică simplă - când gradientul tinde să aibă valori exagerate, ele se vor reseta la valori mici.

Referitor la problema dispariției gradientului, se folosesc de regulă două metode. Una din ele abordează modul de inițializare al parametriilor astfel că în locul unei inițializări aleatoare se folosește matricea identitate. Iar cea de-a doua tehnică propune folosirea funcției de activare ReLu (Rectified Linear Units) în detrimentul celei sigmoid. Motivul este acela că derivata funcției relu este fie 1 fie 0, iar la propagarea erorii înapoi vor fi afectați doar neuronii a căror derivată este 1.

4.1.3 Seq2Seq

Modelul de arhitectură seq2seq se referă la translatarea unei secvențe în alta, de aceea ele sunt cu succes folosite în machine translation. Acest model are două componente importante: cea care face codificarea (captează înțelesul secvenței de intrare) și cea de decodificare înțelesului într-o secvență de ieșire. În funcție de specificul problemei, vocabularul de intrare și de ieșire pot să difere.

Ecuația 4.10 descrie operațiile necesare pentru a codifica în h_t înțelesul secvenței de intrare. Apoi decodificare se realizează cu ajutorul ecuațiilor 4.11 și 4.12 pentru a scoate o distribuție de probabilitate peste cuvintele din vocabularul de ieșire la fiecare pas t.

$$h_t = \phi(h_{t-1}, x_t) = f(W^{(hh)}h_{t-1} + W^{(hx)}x_t) \tag{4.10}$$

$$h_t = \phi(h_{t-1}) = f(W^{(hh)}h_{t-1}) \tag{4.11}$$

$$y_t = softmax(W^{(S)}h_t) \tag{4.12}$$

$$max_\theta \frac{1}{N} \sum_{n=1}^{N} \times log(p_\theta(y^{(n)}|x^{(n)})) \tag{4.13}$$

Pentru ajunge la o performanță crescută, pe lângă funcția de cost 4.13 se adaugă mai multe extensii.

- Extensia 1: se vor folosi ponderi diferite pentru fiecare rețea în parte, adică în ecuația 4.11 în loc de $W^{(hh)}$ vom avea o matrice diferită.
- Extensia 2: fiecare stare ascunsă din rețeaua care decodifică va fi compusă din trei intrări diferite:
  - starea anterioară h_{t-1}
  - ultima stare din encoder ($c = h_T$)
  - ultima ieșire prezisă, y_{t-1}.
- Extensia 3: creșterea numărului de straturi ascunse din encoder și decoder
- Extensia 4: antrenarea unui encoder bidirecțional
- Extensia 5: antrenarea folosind la secvența de intrare în ordine inversă

4.1.4 GRU

Pe lângă extensiile aduse arhitecturii de codare-decodare, o îmbunătățire majoră are loc atunci când modelului matematic din interiorul celulei RNN i se adaugă mai multe porți ce simulează comportamentul unei memorii. (numim celulă a unei rețele neuronale recurente, partea recurentă a modelului care folosește aceleași ponderi de-a lungul secvenței).

Bineînțeles în teorie RNN captează dependințele de lungă durată de-a lungul unei secvențe, dar un asemenea model devine foarte greu de antrenat când vine vorba de cazuri practice. Așadar prin introducerea acestor unități recurente cu porți, rețeaua neuronală recurentă va căpăta o memorie mult mai persistentă fiind mult mai ușor de captat dependințele de lungă durată.

Prima arhitectură de acest gen este Gated Recurrent Units (GRU)

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1}) \tag{4.14}$$

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1}) \tag{4.15}$$

$$\widetilde{h_t} = \tanh(r_t \circ Uh_{t-1} + Wx_t) \tag{4.16}$$

$$h_t = (1 - z_t) \circ \widetilde{h_t} + z_t \circ h_{t-1} \tag{4.17}$$

1. Generarea unei noi memorii: ecuația 4.16 surprinde consolidarea intrării x_t cu starea anterioară. Acest pas este răspunzător de îmbogățirea contextului cu noua intrare venită.

2. Poarta de resetare: semnalul r_t din 4.15 determină cât de important este h_{t-1} pentru a fi luat în calcul de $\widetilde{h_t}$ care calculează noua memorie.

3. Poarta de actualizare: semnalul z_t din 4.14 este responsabil de propagarea contextului surprins anterior de h_{t-1}. De exemplu dacă $z_t \approx 1$ atunci h_{t-1} este aproape copiat în întregime în h_t, altfel dacă $z_t \approx 0$ noua memorie $\widetilde{h_t}$ este transmisă mai departe în h_t

4. Starea ascunsă: h_t este generată folosind ca intrare starea de la pasul $t-1$ și noua memorie în concordanță cu semnalul de actualizare.

4.1.5 LSTM

Long-Short-Term-Memories este un alt timp de celulă recurentă, care folosește aceeași filosofie a porților, dar într-un mod puțin mai diferit față de GRU.

$$i_t = \sigma(W^{(i)} x_t + U^{(i)} h_{t-1}) \tag{4.18}$$

$$f_t = \sigma(W^{(f)} x_t + U^{(f)} h_{t-1}) \tag{4.19}$$

$$o_t = \sigma(W^{(o)} x_t + U^{(o)} h_{t-1}) \tag{4.20}$$

$$\widetilde{c_t} = \tanh(W^{(c)} x_t + U^{(c)} h_{t-1}) \tag{4.21}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c_t} \tag{4.22}$$

$$h_t = o_t \circ \tanh(c_t) \tag{4.23}$$

1. Generarea unei noi memorii: ecuația 4.21 captează noi aspecte introduse de intrarea x_t

2. Poarta de intrare: se asigură că intrarea de la pasul t merită să fie luată în calcul pentru calculul memoriei curente: ecuația 4.18

3. Poarta de uitare: această poartă este similară cu cea de intrare, diferența este că ea determină utilitatea memoriei precedente în calculul celei curente.

4. Generarea memoriei finale: acest pas este descris de ecuația 4.22 care agregă sub semnul operației de adunare deciziile celor doua porți, de uitare și de intrare cu privire la memoria precedentă $c_{t-1}r$, respectiv noua memorie $\widetilde{c_t}$.

5. Poarta de ieșire/expunere: ecuațiile 4.20, 4.23. Rolul acestei porți este de a separa memoria finală c_t de starea ascunsă, h_t. c_t conține informații care nu sunt neapărat esențiale pentru calculul stării ascunse. Semnalul produs de această poartă, o_t este folosit la ponderarea utilizării memoriei finale în calculul stării, h_t.

Capitolul 5

Expunerea tehnologiei

5.0.1 Serviciul NLU

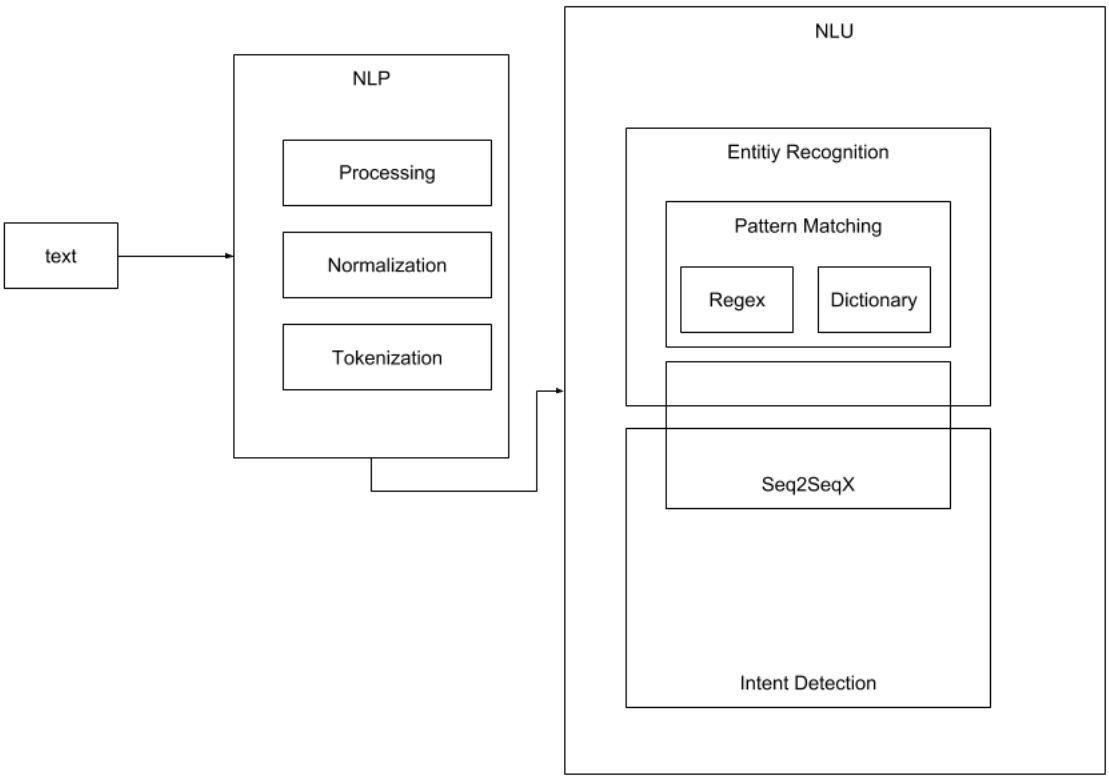

Figura 5.1: Arhitectura serviciului de NLU

5.1 Tehnologii folosite

Desemnarea tehnologiilor open source folosite ca dependințe poate fi văzută la prima vedere o alegere ușoară, însă este nevoie de o analiză mult mai amănunțită în ceea ce privește specificul proiectului. Pentru această lucrare am luat în considerare următorii factori: complexitatea de a descrie o rețea neuronală să fie cat mai simpla, dar să reflecte cat mai bine tot procesul matematic din spate, flexibilitatea de a putea jongla cu diferite arhitecturi de rețele. Evident viteza și eficiența cu care aceste biblioteci rulează, dar și dispozitivele pe care ele rulează (CPU/GPU).

5.1.1 Python

Python a fost creat la începutul anilor 1990 de Guido van Rossum la Stichting Mathematisch Centrum (CWI) în Olanda ca un succesor al limbajului, ABC. [9]

Python este un limbaj de programare puternic și ușor de învățat. El are structuri de date implementate la un nivel înalt și reprezintă o abordare simplă, dar eficientă a programării orientate pe obiecte. Python are o sintaxa elegantă, ce impune o dactilografiere dinamică, împreună cu natura sa de limbaj interpretat, reprezintă un instrument ideal pentru scripting și dezvoltarea rapidă a aplicațiilor în multe domenii, pe majoritatea platformelor.

Interpretorul Python și biblioteca standard extinsă sunt disponibile gratuit în format sursă sau binar pentru toate platformele majore pe site-ul Web Python. În același site sunt conținute, de asemenea, distribuții și indicii pentru mai multe module, programe, instrumente și documentație suplimentară.

Interpretorul Python este ușor de extins cu noi funcții și tipuri de date implementate în C sau C++ (sau alte limbaje apelabile din C). Python este de asemenea potrivit ca o extensie pentru aplicații personalizate. [**?**]

5.1.2 Numpy

Numpy este un acronim pentru ”Numeric Python” sau ”Numerical Python”. El este un pachet fundamental pentru calculul științific in Python, ce furnizează funcții precompilate care se execută rapid cu scopul de a efectua operațiile matematice de rutină. Mai mult decât atât, NumPy îmbogățește limbajul de programare cu structuri puternice de date pentru calculul eficient de vectori si matrice, implementarea sa suportând chiar și dimensiuni uriașe.

SciPy (Scientific Python) este adesea menționat atunci când vine vorba de NumPy. SciPy extinde capabilitățile NumPy cu alte funcții utile pentru minimizare, regresie, transformate Fourier și multe altele.

Atât NumPy și SciPy nu sunt de obicei instalate în mod implicit. NumPy trebuie să fie instalat înainte de a instala SciPy.

NumPy se bazează pe două module anterioare Python care se ocupă cu matrice. Unul dintre acestea este Numeric. Numeric este ca NumPy un modul Python pentru înaltă performanță de calcul numeric, dar este învechit în zilele noastre. Un alt predecesor al NumPy este Numarray, care este o rescriere completă a modulului Numeric, dar este învechit de asemenea. NumPy este o fuziune a celor două, adică este construit pe codul lui Numeric dar cu caracteristicile lui Numarray.

5.1.3 PyTorch

PyTorch este o bibliotecă software ce oferă un cadru de lucru cu algoritmi de invățare automată. Se prezintă ca o variantă de Numpy care poate rula pe placa video, având tot odată și capacitatea de autodiferențiere atunci când este nevoie să antrenăm, spre exemplu folosind metoda gradientului descendent.

Diferențiere Automată

Componenta cheie a rețelelor neuronale din PyTorch este pachetul *autograd*. El oferă diferențierea automată pentru toate operațiile cu *tensori*. Este un cadru de definire a operațiilor (forward dar și backward) la momentul execuției, ceea ce înseamnă

că pasul de backpropagation este definit de modul în care este rulat codul.

Tensor

torch.Tensor este clasa centrală a pachetului. Dacă se setează atributul `.requires_grad` ca `True`, se va începe urmărirea tuturor operațiilor în care acesta intervine. După ce se termină calculul, se poate apela `backward()` pentru a calcula automat toate derivatele, iar gradientul pentru acest tensor va fi acumulat în atributul `.grad`.

Pentru a opri tensorul din istoricul de urmărire, se apelează `.detach()` care detașează tensorul de istoricul de calcul și care împiedica urmărirea viitoarelor calcule.

Mai există încă o clasă care este foarte importantă pentru implementarea autodiferențierii - și anume *Function*.

Tensorul și funcția sunt interconectate și construiesc un graf aciclic, care codifică un istoric complet al calculelor. Fiecare tensor are un atribut `.grad_fn` care se referă la o funcție care a creat tensorul (cu excepția tensorurilor creați de utilizator unde - `.grad_fn = None`).

Capitolul 6

Concluzii

Vorbind dintr-o perspectivă academică, rezultatele obținute reușesc să convingă de adevărata putere a acestei arhitecturi. Datorită rețelelor recurente se reușește captarea contextului, fapt ce duce la o putere mare de generalizare. Capacitatea modelului de a prezice crește atunci când in joc intervine și atenția, acest strat care ajută estimatorul să se concentreze pe anumite cuvinte atunci când decide asupra unei etichete.

Un punct slab al acestei abordări din punct de vedere al comercializării este că pentru faza de antrenare este nevoie de un număr mare de exemple. Un alt factor care stă în calea scalării numărului de clienți este nevoia unui expert în domeniu care să concentreze în mulțimea de antrenare intenții și exemple relevante. Construirea unei astfel de mulțimi de antrenare necesită și implicarea programatorului (sau a unei persoanei care cunoaște cum funcționează tehnologia - seq2seq), astfel încât să se asigure că exemplele construite de expertul în domeniu, au o oarecare consistență, spre exemplu: dacă s-a dat propoziția: ”vreau să imi resetez parola din aplicația Saturn” etichetat cu entitatea: app: Saturn, să existe și contra exemple în care să se specifice și celalalt sens al cuvântului ”Saturn”, astfel modulul de NLU să poată identifica cu succes atunci când este vorba de aplicație sau despre planetă.

Legat de componenta care ține contextul întregii conversații, este foarte ușor de urmărit întregul fir al discuției, datorită modelului bazat pe umplerea de sloturi (entități necesare în cadrul unei sarcini).

Bibliografie

[1] Richard Socher Milad Mohammadi, Rohit Mundra. Cs 224d: Deep learning for nlp, lecture notes: Part iv, 2015. URL http://cs224d.stanford.edu/lecture_notes/notes4.pdf. [Online, accesat la: 27.03.2019].

[2] Dan Jurafsky and James H. Martin. *Speech and Language Processing*. 2018.

[3] Bing Liu and Ian Lane. Joint online spoken language understanding and language modeling with recurrent neural networks, 2016.

[4] Bing Liu and Ian Lane. Attention-based recurrent neural network models for joint intent detection and slot filling, 2016.

[5] J Weizenbaum. Eliza. *Communications of the ACM, 9(1), 36–45*, 1966.

[6] Ludwig Wittgenstein. *Philosophical Investigations*. Translated by Anscombe, G.E.M. Blackwell, 1953.

[7] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation, 2015.

[8] J. J. Godfrey C. T. Hemphill and G. R. Doddington. The atis spoken language systems pilot corpus. *Proceedings, DARPA speech and natural language workshop*, pages 96–101, 1990.

[9] Python Software Fundation. History and license. URL https://docs.python.org/3/license.html. Online, accesat la: 11.06.2016.

Anexe