

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	3
1 Introducere	5
1.1 Motivația	5
1.2 Descrierea problemei	6
1.3 Progres precedent	6
1.4 Rezumatul capitolelor	7
2 Dialog	9
2.1 Conversația naturală	9
2.1.1 Acte de vorbire	10
2.2 Sistem de dialog	10
3 Sistem de dialog	12
3.1 Înțelegerea limbajului natural	13
3.1.1 Abordări anterioare	13
3.1.2 Model propus	14
3.2 Administrator de dialog	15
3.2.1 Abordări anterioare	15

3.2.2	Slot filling	15
3.3	Metode de evaluare	15
3.4	Rezultate	15
4	Noțiuni teoretice	17
4.1	Rețele Neurale Recurente	17
4.1.1	Problema dispariției sau a explodării valorilor gradientului . . .	19
4.1.2	Soluții pentru dispariția/explodarea gradientului	21
4.1.3	Seq2Seq	21
4.1.4	GRU	22
4.1.5	LSTM	23
5	Expunerea tehnologiei	25
5.0.1	Serviciul NLU	25
5.1	Tehnologii folosite	26
5.1.1	Python	26
5.1.2	Numpy	27
5.1.3	PyTorch	27
6	Concluzii	29
	Bibliografie	30
	Anexe	30

Listă de figuri

2.1	Sistem de Dialog Modular	11
3.1	Seq2Seq-x	14
3.2	Bidirectional Encoder	15
3.3	Attention Decoder	16
4.1	Arhitectura RNN [1]	18
5.1	Arhitectura serviciului de NLU	25

Lista porțiunilor de cod

Capitolul 1

Introducere

Dincolo de utilitatea practică de a transmite informații, limbajul este cel care intensifică interacțiunea și menține legăturile interumane. Făcând o paralelă cu sintagma ”o imagine valorează cât o mie de cuvinte” la fel un proverb românesc surprinde esența cu: ”*Vorba dulce mult aduce*”, reliefând astfel caracterul artistic și contemplativ al limbajului, trăsătură pusă într-o antiteză chiar de caracterul pragmatic din debut. Cu alte cuvinte limbajul este uneori paradoxal, iar cuvintele ar putea curge la nesfârșit spre deslușirea acestei capacități,

În cartea sa *Speech Processing*, profesorul Dan Jurafski 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. Această remarcă arată de fapt longevitatea și neoboseala dorinței creatoare concretizată chiar în această ramură a lingvisticii matematice care ne permite cu adevărat să dăm naștere unui astfel de agent conversațional.

1.1 Motivația

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 să 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 spre cunoaștere.

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 să îmi aduc contribuția într-un domeniu atât de important în drumul nostru spre o viziune clară.

1.2 Descrierea problemei

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 simplu dar care să ducă la bun sfârșit sarcinile dorite spre a fi executate de care agentul virtual rezultat.

În viziunea mea, această chestiune reprezintă mai degrabă o nevoie decât o problemă. O nevoie ce survine în urma stilului nostru de viață dinamic și învelit în straturi de informație.

Cum limbajul natural este cel mai la îndemână instrument de comunicare, consider ca prin intermediul său vom putea satisface nevoia unei interfețe capabile să ușureze interacțiunea dintre noi și tehnologie.

1.3 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 să 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 [?]. 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 degrabă unui scop subiectiv decât unui 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 [2]

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 neuronale 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.

Privind comunicarea ca o nevoie de bază ne ajută să vedem mai clar de ce procesarea limbajului natural este un element esențial în drumul nostru spre cunoaștere. Datorită actualului progres în acest domeniu ne putem bucura de ușurința cu care informațiile circulă între noi, făcând realizabil acest avânt tehnologic de care nu bucurăm cu toții.

1.4 Rezumatul capitolelor

Introducere -

Capitolul întâi vorbește în 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 cunoștințelor care se bazează unele pe altele. Precum această imagine implementarea acestei tehnologii de dialog impune un anumit progres precedent, așa că în capitolul doi vor fi prezentate aceste instrumente care fac posibilă aceasta tehnologie.

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

În partea a patra se prezintă modulul de înțelegerea limbajului natural

Al cincelea capitol descrie modulul care ține contextul unei conversații.

Iar în partea de final concluziile referitoare la studiul elaborat în 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 [3]. 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 acțiunile 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 plănuși, 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 și cum sunt ele văzute în 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 rândul-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.

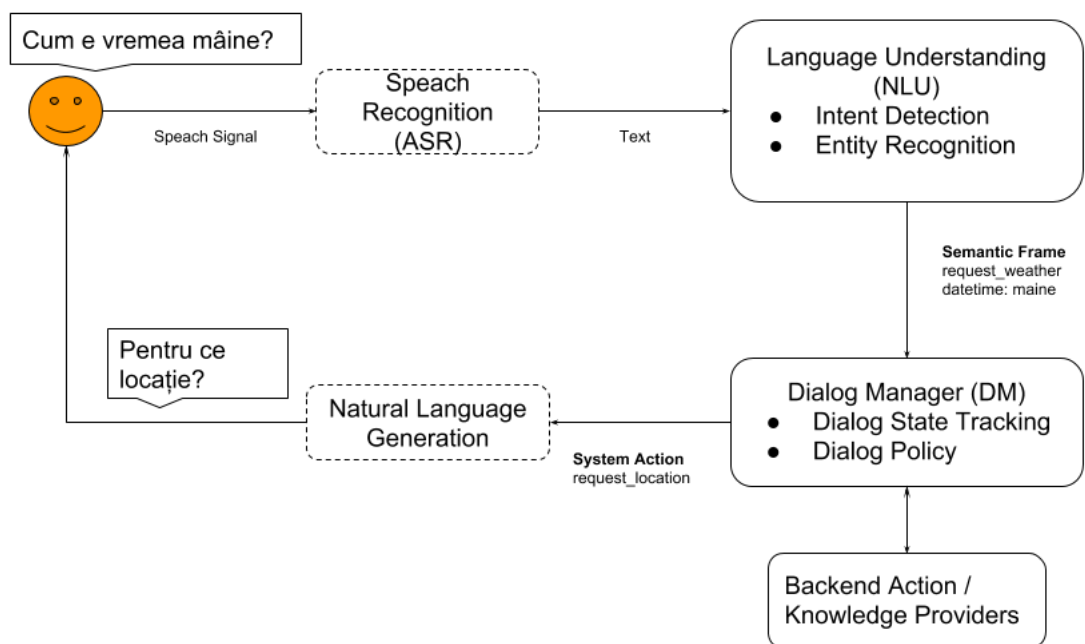


Figura 2.1: Sistem de Dialog Modular

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 până 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 dezvoltati 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, în terminologia sistemelor de dialog acesta joacă rolul componentei care detectează intenția vorbitorului și extrage constituenți semantici din limbajul natural exprimat. Detecția intenției poate fi tratată ca o problemă 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 și anume SVM (Haffner et al., 2003), dar și 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 Ricciardi, 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 concomitentă a acestor două probleme, întrucât combinarea celor două modele ajută la învățarea unei reprezentări cât mai precise a textului. [?]

3.1.2 Model propus

Arhitectura matematică ce a devenit foarte populară în procesarea limbajului natural este rețeaua neurală recurentă, datorită formelor sale (LSTM/GRU) optimizate pentru învățarea automată ea se bucură de un succes răsunător. 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.

Ambele probleme sunt rezolvate simultan folosind un codicator pentru reprezentarea înțelesului unei propoziții și două decodificatoare câte unul pentru fiecare chestiune.

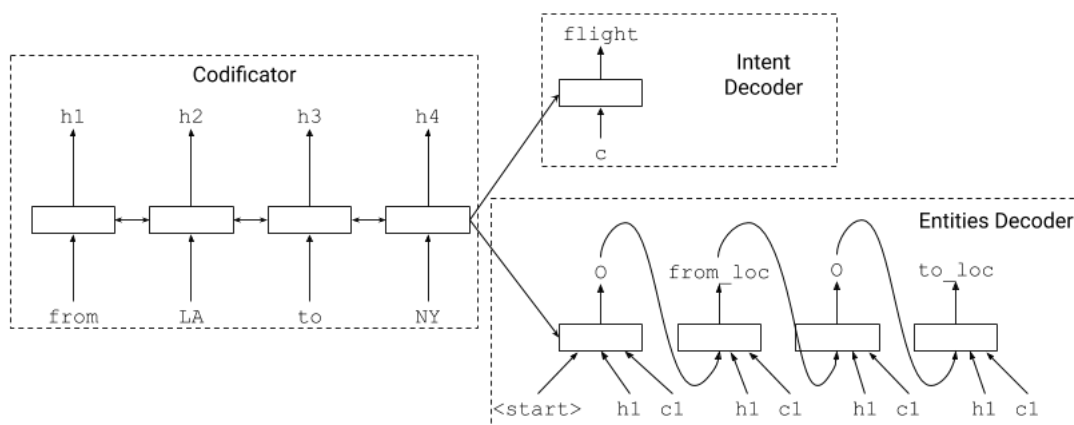


Figura 3.1: Seq2Seq-x

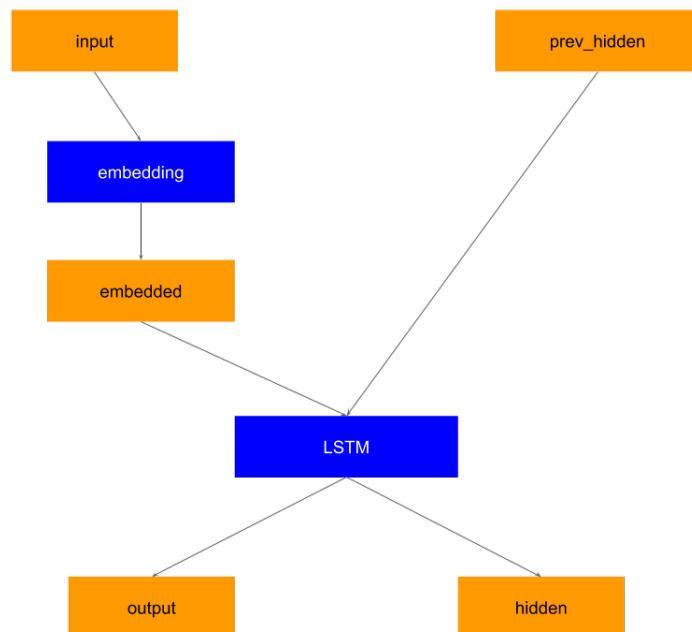


Figura 3.2: Bidirectional Encoder

3.2 Administrator de dialog

3.2.1 Abordări anterioare

3.2.2 Slot filling

3.3 Metode de evaluare

3.4 Rezultate

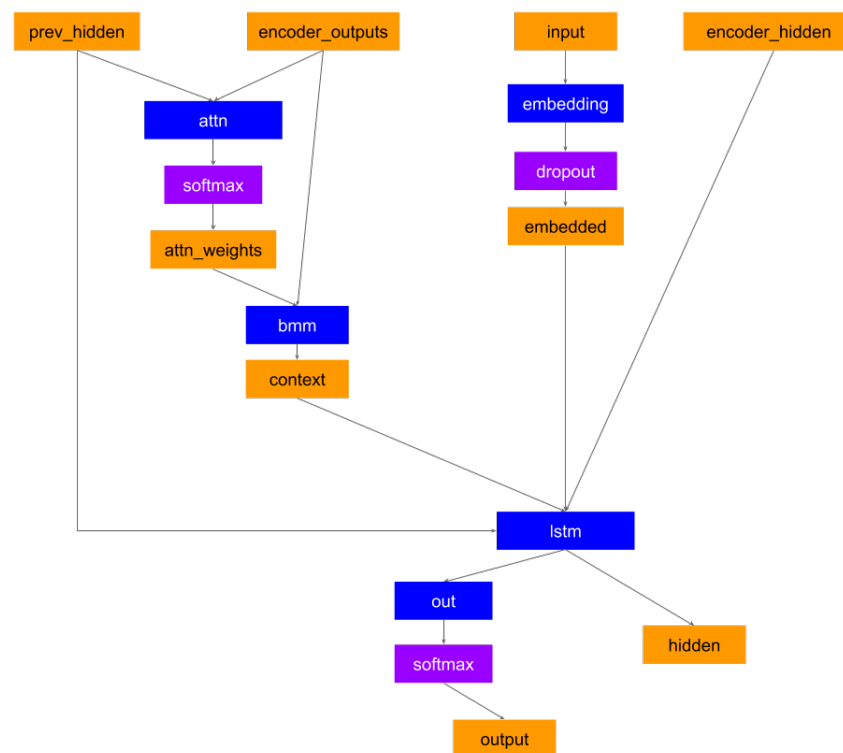


Figura 3.3: Attention Decoder

Capitolul 4

Noțiuni teoretice

4.1 Rețele Neuronale Recurente

Rețelele neuronale recurente (RNN - Recurent Neural Networks), sunt o arhitectura aparte de rețele neuronale, ce le face atât de speciale este faptul că ele reușesc să capteze secvențialitatea datelor. Ele sunt folosite în special în procesarea limbajului natural, dar și în procesarea imaginilor, a seriilor de timp, a recomandărilor de produse. Cu alte cuvinte oricând vine vorba de succesiunea anumitor evenimente, ele reprezintă un candidat bun în captarea acestor modele în 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 înmulțire între parametrii și input, urmată de o operație nonlinară (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 .

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

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

Descrierea parametrilor din rețea:

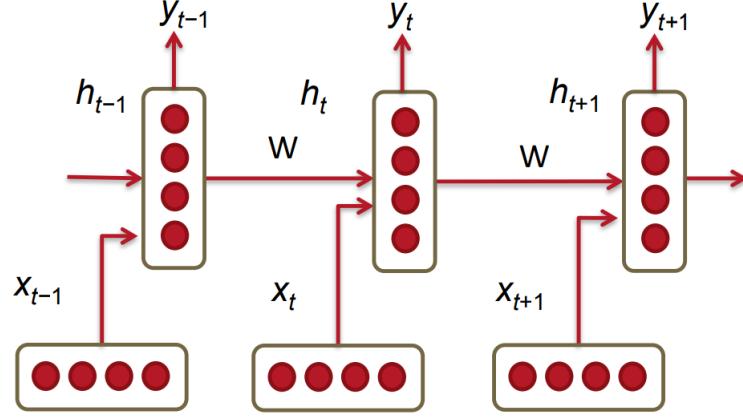


Figura 4.1: Arhitectura RNN [1]

- x_1, x_2, \dots, 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 ponderilor utilizată la condiționarea vectorului de intrare, x_t
 - $W^{hh} \in \mathbb{R}^{D_h \times D_h}$ matricea ponderilor 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 nonlinară (sigmoid în acest exemplu)
- $y_t = \text{softmax}(W^{(S)}h_t)$ 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 în rețelele neuronale recurente este de obicei CE (cross entropy error).

Pentru pasul t :

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

Pentru toată secvența de lungime T funcția de cost devine:

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

[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.

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

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

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

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

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

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

$$\frac{\partial h_j}{\partial h_{j-1}} = \left[\frac{\partial h_j}{\partial h_{j-1,1}} \dots \frac{\partial h_j}{\partial h_{j-1,D_n}} \right] = \begin{bmatrix} \frac{\partial h_{j,1}}{\partial h_{j-1,1}} & \dots & \frac{\partial h_{j,1}}{\partial h_{j-1,D_n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_{j,D_n}}{\partial h_{j-1,1}} & \dots & \frac{\partial h_{j,D_n}}{\partial h_{j-1,D_n}} \end{bmatrix} \quad (4.6)$$

Din 4.4, 4.5, 4.6 avem următoarea relație:

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

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.

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

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

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

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 în 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 valoarea gradientului crește foarte mult, poate cauza erori de memorie (valoarea nu încapă 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 rezolva 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 parametrilor 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.

Ecuatia 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 .

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

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

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

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

Pentru a 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 dependențele 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 dependențele de lungă durată.

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

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

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

$$\tilde{h}_t = \tanh(r_t \circ U h_{t-1} + W x_t) \quad (4.16)$$

$$h_t = (1 - z_t) \circ \tilde{h}_t + z_t \circ h_{t-1} \quad (4.17)$$

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 \tilde{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 \tilde{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 tip de celulă recurentă, care folosește aceeași filosofie a porților, dar într-un mod puțin mai diferit față de GRU.

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

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

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

$$\tilde{c}_t = \tanh(W^{(c)}x_t + U^{(c)}h_{t-1}) \quad (4.21)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \quad (4.22)$$

$$h_t = o_t \circ \tanh(c_t) \quad (4.23)$$

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 două porți, de uitare și de intrare cu privire la memoria precedentă $c_{t-1}r$, respectiv noua memorie \tilde{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

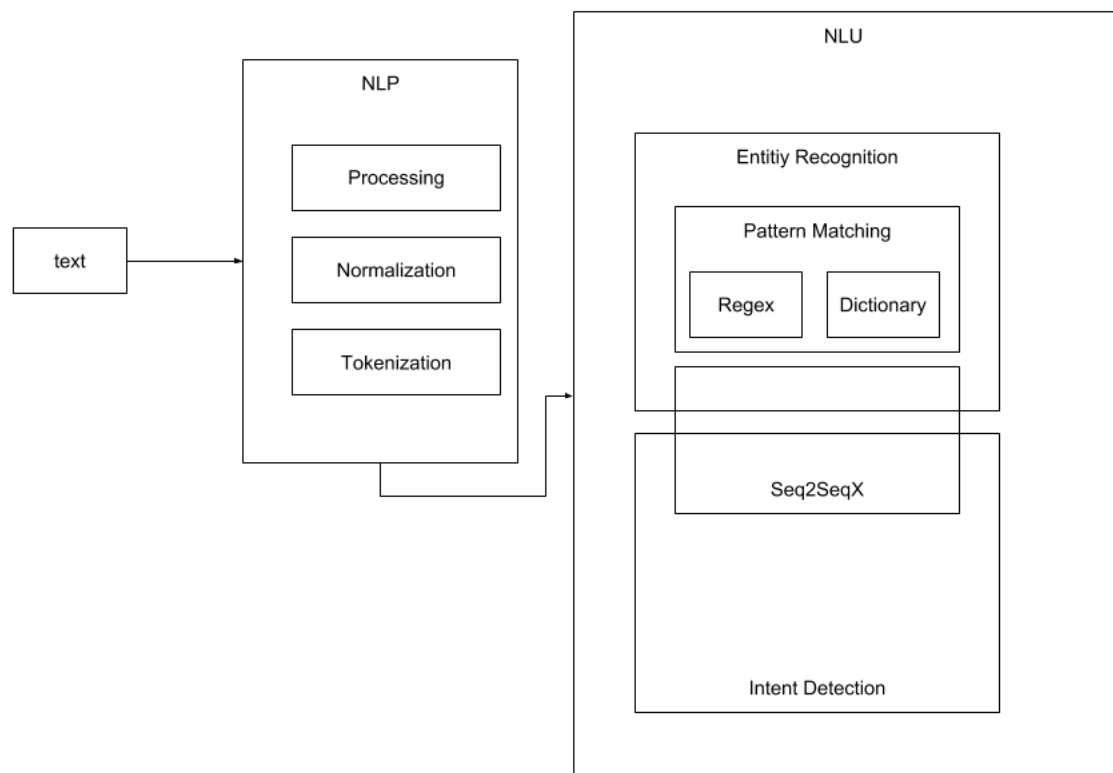


Figura 5.1: Arhitectura serviciului de NLU

5.1 Tehnologii folosite

Desemnarea tehnologiilor open source folosite ca dependențe 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 simplă, 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. [5]

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 dactilografie 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.

Interpreterul 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.

Interpreterul 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. [6]

5.1.2 Numpy

Numpy este un acronim pentru "Numeric Python" sau "Numerical Python". El este un pachet fundamental pentru calculul științific în Python, ce furnizează funcții pre-compilate 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 și matrice, implementarea sa suportând chiar și dimensiuni uriase.

SciPy (Scientific Python) este adesea menționat atunci când vine vorba de NumPy. SciPy extinde capacitățile 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 învățare 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 împiedică urmărirea viitoarelor calcule.

Mai există încă o clasă care este foarte importantă pentru implementarea auto-diferențierii - 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 în 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 persoane 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 de pe 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] J Weizenbaum. Eliza. *Communications of the ACM*, 9(1), 36–45, 1966.
- [3] Ludwig Wittgenstein. *Philosophical Investigations*. Translated by Anscombe, G.E.M. Blackwell, 1953.
- [4] Jimmy Lei Ba Diederik P. Kingma. Adam: A method for stochastic optimization. *Published as a conference paper at ICLR 2015*, 2015. URL <http://arxiv.org/pdf/1412.6980v8.pdf>.
- [5] Python Software Foundation. History and license, . URL <https://docs.python.org/3/license.html>. Online, accesat la: 11.06.2016.
- [6] Python Software Foundation. The python tutorial, . URL <https://docs.python.org/3/tutorial/index.html>. Online, accesat la: 11.06.2016.

Anexe