UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

LUCRARE DE DISERTAȚIE

Agent conversațional

Coordonator științific:

Prof. Dr. Liviu Dinu

Student:

Ionuț Ciocoiu

FEBRUARIE 2019

Cuprins

Listă de figuri

Lista porțiunilor de cod

Capitolul 1

Introducere

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

1.1 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 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 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 sa îmi aduc contribuția într-un domeniu atât de important în drumul nostru spre o viziune clara.

1.2 Descrierea problemei

Lucrarea își propune descoperirea unui model îndeajuns de puternic încât să înțeleagă limbajul natural exprimat în contextul 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 este mai degrabă ca o nevoie decăt 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 îndemâna instrument de comunicare, consider ca prin intermediul său vom putea satisface nevoia unei interfețe capabile să ușureze interacțiunea dintre noi și tehnologie.

1.2.1 Privire de ansamblu

Avem nevoie de un modul de NLU și un modul de DM pentru a pune bazele unui sistem de dialog

1.3 Rezumatul capitolelor

- 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 îndraznețe.
- Atunci cand se rostește ”progres” am în minte o spirală a cunoștiintelor 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

Tehnologii folosite în implementare

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).

2.1 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. [1]

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. [2]

2.2 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.

2.3 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.

2.3.1 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.

2.3.2 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 3

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.

3.1 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.

3.1.1 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.

3.2 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 3.1. 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.

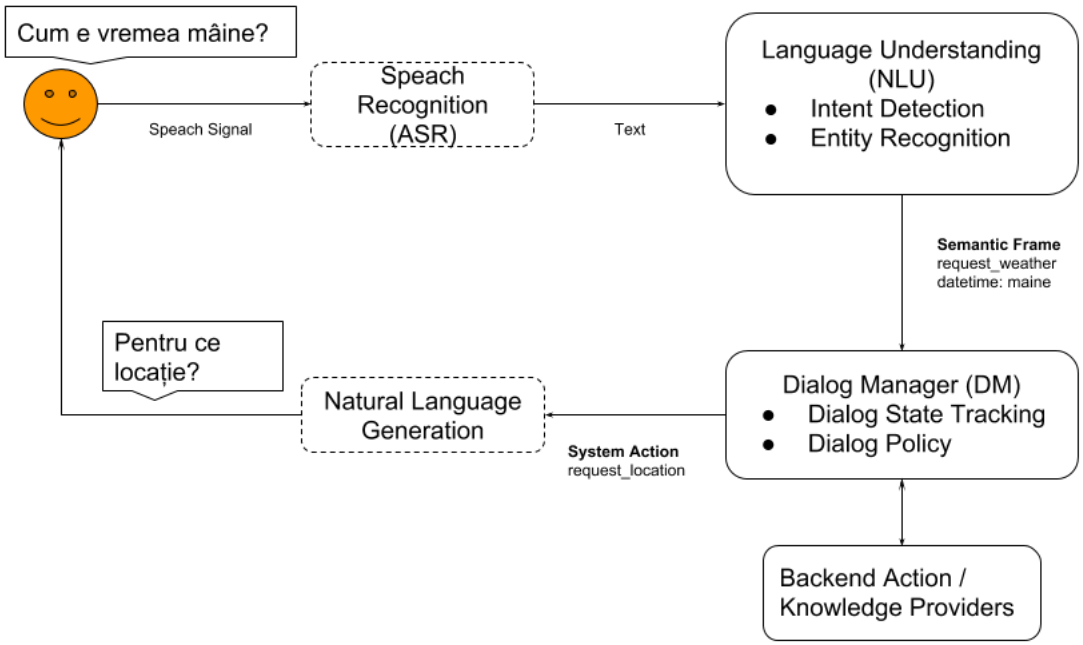

Figura 3.1: Sistem de Dialog Modular

3.2.1 Înțelegerea limbajului natural (NLU)

3.2.2 Administrator de dialog (DM)

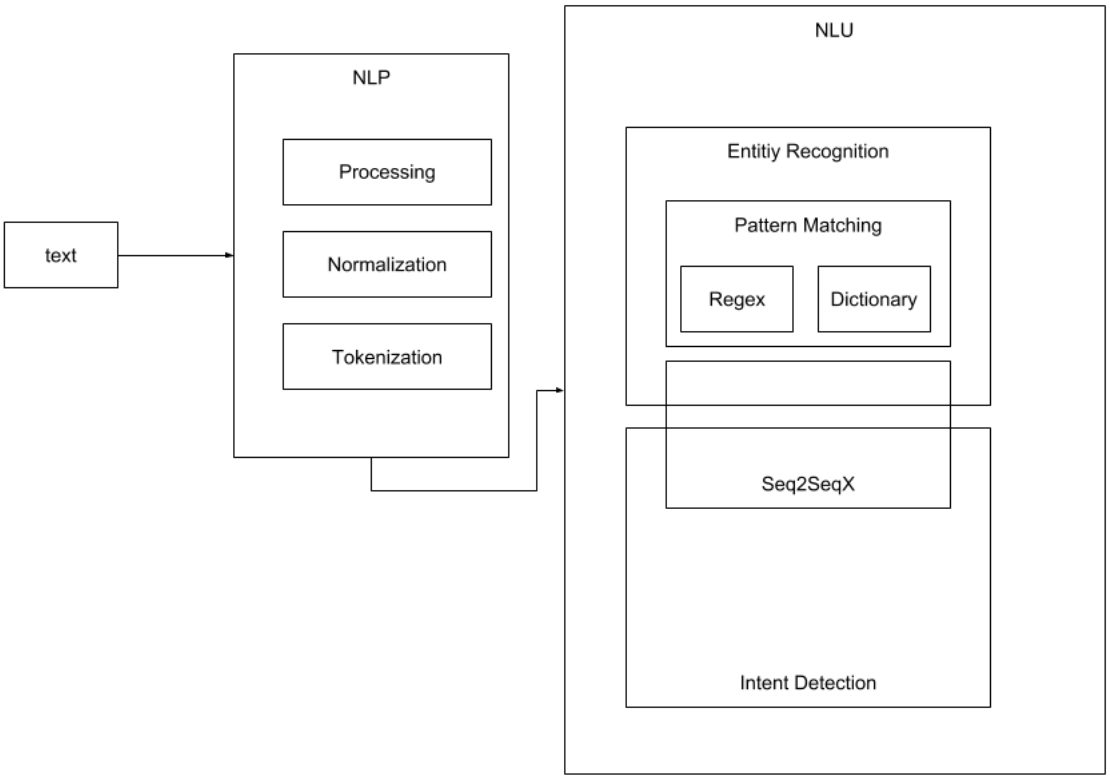

Figura 3.2: Arhitectura serviciului de NLU

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.

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

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

Descrierea parametrilor din rețea:

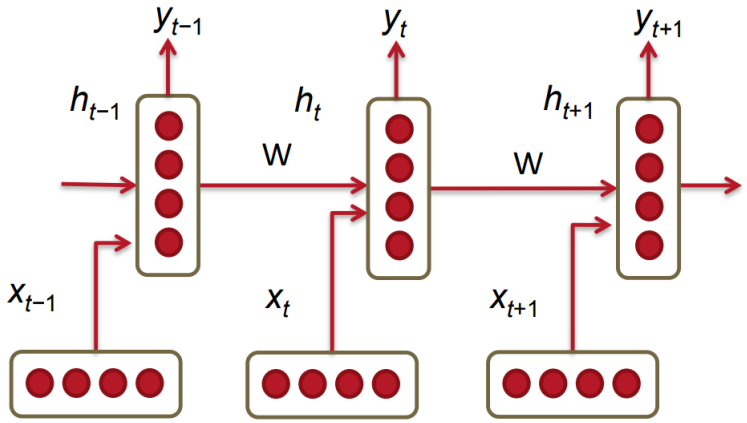

Figura 4.1: Arhitectura RNN [4]

- $x_1, x_2, ..., x_T$ 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:

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

[4]

4.1.1 LSTM

4.1.2 Seq2Seq

Capitolul 5

Sistem de dialog

5.1 Înțelegerea limbajului natural

5.1.1 Abordări anterioare

5.1.2 Seq2Seq

5.2 Administrator de dialog

5.2.1 Abordări anterioare

5.2.2 Slot filling

Capitolul 6

Evaluare

6.1 Metode de evaluare

6.2 Rezultate

Capitolul 7

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

[2] Python Software Fundation. The python tutorial, . URL `https://docs.python.org/3/tutorial/index.html`. Online, accesat la: 11.06.2016.

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

[4] 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].

Anexe