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 **3**

1 Introducere **5**

1.1 Motivația 5

1.2 Descrierea problemei 6

1.2.1 Privire de ansamblu 6

1.3 Rezumatul capitolelor 6

2 Tehnologii folosite în implementare **7**

2.1 Python 7

2.2 Numpy 8

2.3 PyTorch 9

2.3.1 Diferențiere Automată 9

2.3.2 Tensor 9

3 Dialog **10**

3.1 Conversație naturală 10

3.1.1 Acte de vorbire 11

3.2 Sistem de dialog 11

3.2.1 Componente 11

4 Noțiuni teoretice **12**

4.1 Rețele Neurale Recurente 12

4.1.1 LSTM 12

5 Sistem de dialog **13**

5.1 Înțelegerea limbajului natural 13

5.1.1 Abordări anterioare 13

5.1.2 Seq2Seq 13

5.2 Administrator de dialog 13

5.2.1 Abordări anterioare 13

5.2.2 Slot filling 13

6 Evaluare **14**

6.1 Metode de evaluare 14

6.2 Rezultate 14

7 Concluzii **15**

Bibliografie **16**

Anexe 16

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 se contureaza tot mai bine gandul ca este nevoie de o întreaga armata de oameni de știință care să întoarca pe toate parțile un subiect pentru ca mai apoi cei mai de seamă dintre aceștia sa poată veni cu o sclipire. Fac această remarcă având în minte imaginea omului care obosit in a cauta raspuns la intrebarile existentiale, dar care se concentrează acum pe creație și descoperiri care sa facă viața mai ușoară și mai placută, lucru care dă naștere și mai multor mistere decât elucidări.

Dorința de a crea cu scopul de a aduce un aport de liniște în viața oamenilor este imboldul intrinsec ce ghidează acțiunile mele, așadar evaluând cunoștiințele mele, am decis sa imi aduc contribuția într-un domeniu atât important în drumul nostru spre o viziune clara precum un râu liniștit ce curge fară zgomot, dar limpede.

1.2 Descrierea problemei

Nu aș privi această chestiune precum o problemă, ci mai degrabă ca o nevoie. 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 percepției de decizie.
- 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 Conversație 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]

3.1.1 Acte de vorbire

3.2 Sistem de dialog

3.2.1 Componente

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.

4.1.1 LSTM

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

Cele profunde concluzii

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.

Anexe