Universitatea Transilvania din Braşov
Facultatea de Matematică şi Informatică
Specializarea Tehnologii Moderne
în Ingineria Sistemelor Soft

LUCRARE DE DISERTAŢIE

Autor: Muşat Bogdan-Adrian
Coordonator: Conf. dr. Lucian-Mircea Sasu

Braşov
2017

Cuprins

Capitolul 1

Introducere

În zilele noastre, societatea se confruntă tot mai mult cu diverse probleme tehnice în mediul online, industrial, cotidian etc. Până de curând, aceste probleme puteau fi rezolvate doar cu ajutorul unui specialist în domeniul respectiv. După cum ştim, omul reprezintă o resursă limitată când vine vorba de suport tehnic, deoarece cererile pot fi numeroase, în consecinţă raspunsul primit poate fi întarziat. De asemenea, trebuie luat in considerare costul prohibitiv de întreţinere a unor persoane responsabile cu acest tip de suport.

Recent, diverse companii de succes precum Google [1], Facebook[2], Microsoft [3], Apple [4] şi alţii au făcut un demers spre adoptarea unor sisteme inteligente de comunicaţie numite chatbots. În cazul chatbots-ilor orientaţi pe suport tehnic, scopul lor este tocmai de a imita sprijinul pe care o persoană reală îl poate oferi. Un sistem computaţional inteligent poate fi folosit pentru a răspunde multor cereri simultane, iar costurile de intreţinere sunt mici practic, odată ce sistemul este dezvoltat, este necesară doar expunerea lui, de exemplu ca serviciu web. Cu cât tehnologia şi munca de cercetare în această arie progresează, cu atât sistemele de suport automat devin din ce în ce mai inteligente şi mai apropiate de ce un om este capabil să ofere [19-21]. Se preconizează că în următorii zece ani, aceste sisteme vor fi capabile să înlocuiască cu succes o mulţime de sarcini. Tot aşa cum automatizarea industrială de producere a vehiculelor reprezintă un standard în era contemporană, aşa vor reprezenta şi aceşti asistenţi conversaţionali un standard în anii ce vor urma.

Majoritatea modelelor actuale se bazează pe un tip de oferire a unor răspunsuri predefinite. Aceste tipuri de sisteme pot oferi doar soluţii existente, nefiind capabile de a genera nimic nou. Pe de altă parte, chatbots-ii

bazaţi pe inteligenţă artificială reprezintă cea mai scalabilă modalitate de comunicare între clienţi şi mediile de afaceri care le sunt dedicate [5].

Vorbim deci de evoluţia unor modele de chatbots bazate pe pattern matching către unele bazate pe modele generative, care, precum omul, pot emite răspunsuri bazate pe experienţe anterioare. În ultimii ani, ramura inteligenţei artificiale numită machine learning (învăţare automată) a luat amploare în acestă direcţie prin curentul numit deep learning [6-8]. Acest curent a produs o mulţime de rezultate spectaculoase atât în direcţia procesării naturale de limbaj cât şi a procesării de imagini.

Problema asistenţilor conversaţionali este una de procesare a limbajului natural. Sistemul trebuie să fie capabil să "înţeleagă" informaţia primită de la o persoană şi să producă un raspuns cât mai coerent şi util. Însă cum înţelege un calculator o limbă? Pentru a răspunde la această întrebare, vom face o analogie la cum învaţă un om o limbă. Porneşte de la anumite cuvinte de bază iar apoi pe baza acestora, învaţă cuvinte tot mai complexe. Începe să creeze fraze prin care leagă aceste cuvinte precum şi gramatica specifică limbajului. Practic, totul se bazează pe o anumită experienţă trecută. Conversaţia este o modalitate ce facilitează şi impulsionează deprinderea limbajului: omul este pus în faţa unor contexte de utilizare, fenomen ce întăreşte deprinderea de utilizare a cuvintelor sau expresiior individuale. Fiecare cuvânt nou învăţat reprezintă o experienţă spre învăţarea în continuare a limbajului. Încă un lucru la care omul este bun este capabilitatea de a întreţine o conversaţie lungă cu o altă persoană şi să reţină tot felul de informaţii noi pe parcurs.

În abordările de chatbot actuale se doreşte imitarea acestei modalităţi de învăţare pentru un sistem computaţional. Modelarea cea mai puternică în ziua de azi pentru o astfel de problemă este oferită de deep learning [6-8]. Folosind un număr mai mare de neuroni faţă de arhitecturile shallow folosite până la începutul anilor 2000 şi plecând de la seturi de instruire masive - în cazul de faţă corpusuri de text cu cât mai multe fraze în limba pe care o dorim a fi învăţată se formează modele generative care pot produce continuarea unor propoziţii, fraze etc. Ideea din spatele abordării deep learning este că sistemul devine mai inteligent pe măsură ce avem tot mai multe date - în acest caz, speţe de conversaţii. Pentru a procesa o asemenea cantitate de date de instruire este nevoie de putere computaţională pe măsură. Asta presupune pe scurt, capacitate hardware. Calculele matematice efectuate pentru învăţarea unui limbaj de către un sistem sunt complexe şi numeroase (de ordinul sutelor de milioane). Folosirea unui procesor, chiar multicore de

ultimă generaţie, deşi o idee posibilă, este considerată a duce la sugrumarea procesului de instruire.

În locul microprocesorarelor se preferă folosirea de plăci grafice (Graphical Processing Units, GPU). Iniţial acestea au fost dezvoltate pentru rulare rapidă de jocuri, dar potenţialul lor a fost rapid intuit şi exploatat prin programare paralelă. Deoarece placa video conţine mult mai multe nuclee de procesare (câteva mii, comparate cu cele 4-8 nuclee tradiţionale dintr-un microprocesor actual), este preferată programarea şi rularea modelelor computaţionale pe GPU. Sunt dezvoltate biblioteci care facilitează unui programator dezvoltarea de aplicaţii de machine learning pe GPU: Tensorflow [9], Theano [10], Caffe [11], Keras [12] etc.

1.1 Motivaţia alegerii temei

O potenţială utilizare este augmentarea sistemului de creare a tichetelor: Universitatea Transilvania foloseşte în mod curent un sistem de suport tehnic bazat pe tichete, care apoi sunt procesate de persoane reale pentru rezolvare (Figura 1.1, Figura 1.2). Un tichet presupune primirea de la solicitant a cât mai multor detalii legate de problemă. De regulă, din lipsă de experienţă, detaliile furnizate sunt insuficiente pentru o soluţionare eficientă, motiv pentru care, după completarea iniţială a tichetului se poartă un dialog între suportul IT şi solicitant pentru aflarea de informaţii suplimentare legate de problema raportată. Acest lucru consumă timp; se poate îmbunătăţi procesul prin demararea cât mai rapidă a unui dialog solicitant - chatbot prin care să se completeze tichetul cât mai fidel.

Figura 1.1: Sistem pentru suport IT pe portalul Universităţii

Figura 1.2: Pagina de sesizări pentru biroul IT

Un sistem artificial de chatbot, pe baza dialogurilor anterioare înregistrate şi a unei similitudini a cererilor, ar putea fie să solicite mai multe detalii, fie să sugereze paşi de rezolvare. În cazul în care există un corpus de cunoştinţe (knowledge-base) dat de experienţele anterioare (dialoguri, soluţii date), e posibil ca el să fie exploatat în mod automat [19-21].

1.2 Structura lucrării

TODO

1.3 Acknowledgement

TODO

Capitolul 2

Lucrări similare

În anul 1950, matematicianul Alan Turing a propus un test care pune la încercare abilitatea unei maşini computaţionale de a manifesta inteligenţă echivalentă cu cea umană. Un evaluator uman judecă o conversaţie între un om şi o maşină desemnată să genereze răspunsuri cât mai naturale. Evaluatorul este conştient că unul dintre partenerii angrenaţi in discuţie este o maşină. Conversaţia este limitată doar la text astfel încât rezultatul să nu depindă de abilitatea calculatorului de a genera sunete. Dacă evaluatorul nu poate diferenţia maşina de om, putem spune că aceasta a trecut testul. Până în prezent, acest test rămâne încă în picioare.

Există două mari abordări la ora actuală de a genera limbaj cât mai natural. Avem în primul rând modele de tip regăsire, unde răspunsul este stocat într-o sursă de date şi returnat pe baza unor metode de pattern matching. Cea de-a doua variantă o reprezintă modelele generative, care produc un răspuns dinamic folosind diverse metode din teoria probabilităţilor.

2.1 Modele de regăsire

Modelele de acest tip folosesc o sursă de date care conţine numeroase răspunsuri predefinite. Răspunsul este ales folosind o metodă euristică pentru o potrivire cât mai bună cu putinţă, luând în considerare intrarea şi contextul. Tipul de euristică folosit poate fi ceva simplu precum o expresie bazată pe o regulă de potrivire sau ceva mai complex cum ar fi un clasificator de Machine Learning. Se poate deduce foarte uşor că aceste sisteme nu generează text nou. O problemă uriaşă a acestor modele o reprezintă incapacitatea de a reacţiona la cazuri nemaiîntalnite pentru care nu există un răspuns potrivit. Acestea au totuşi avantajele lor. Datorită sursei de date cu răspunsuri create

de oameni, aceste metode nu produc erori gramaticale. În prezent, acestea reprezintă abordarea sigură pentru problemele în care răspunsul este unul sensibil, într-un domeniu precum cel medical, de exemplu.

2.1.1 Duolingo

Populara aplicaţie de învăţare a limbilor străine Duolingo (Figura 2.1) foloseşte o abordare interesantă cu privire la chatbots. Aceasta doreşte să îşi ajute utilizatorii să practice o nouă limbă prin conversaţii cu chatbots. Având în vedere că o conversaţie este considerată a fi printre cele mai bune moduri de a învăţa o limbă străină, utilizatorii Duolingo pot vorbi cu bot-ul oricât de mult îşi doresc, iar acesta îi va corecta şi le va propune răspunsuri potrivite. Mai mult de atât, poate estima progresul utilizatorului pentru a-şi creşte nivelul de dificultate, păstrând astfel constantă provocarea.

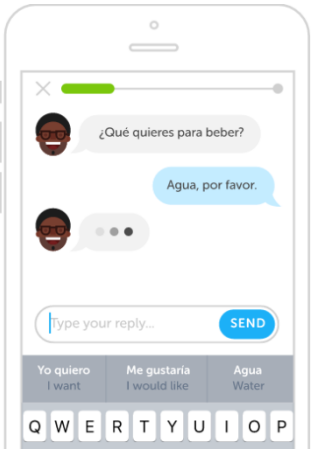

Figura 2.1: Conversaţie cu chatbot folosind Duolingo

2.1.2 Facebook Messenger

Facebook s-a alăturat întru totul afacerii conversaţionale, astfel încât şi-a transformat aplicaţia Messenger într-un business de mesagerie. Compania a integrat plaţile peer-to-peer în Messenger în anul 2015, apoi urmând să lanseze un API pentru chatbots, astfel încât business-urile să poată crea interacţiuni pentru clienţi. Poţi comanda flori, să navighezi printre ultimele trend-uri în materie de modă, să comanzi Uber, toate dinăuntrul chat-ului de Messenger (Figura 2.2).

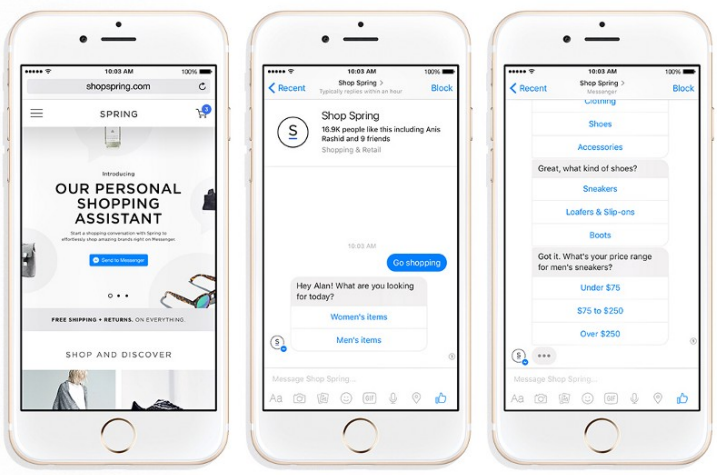

Figura 2.2: Cumpărături de haine folosind chatbot-ul companiei Spring pe Facebook Messenger

2.2 Modele generative

Spre deosebire de modelele anterioare, cele generative nu se bazează pe răspunsuri predefinite ci generează noi răspunsuri pornind de la zero. Modelele generative folosesc deobicei tehnici din Machine Translation, dar în loc de a traduce dintr-o limbă într-alta, vom "traduce" de la o intrare la o ieşire (răspuns). Acestea oferă o mai bună impresie de comunicare cu un om real. Totuşi, ele sunt extrem de greu de antrenat, sunt predispuse la erori gramaticale (în special unde lungimea propoziţiei este mai mare) şi necesită o cantitate mare de date de antrenare.

Prezentul este încă sub semnul întrebării pentru acest tip de model, însă în următorii ani, acestea vor căpăta tot mai multă atenţie şi popularitate devenind tot mai performante. Dacă un chatbot va doborî testul Turing, şansele ca acesta să fie generativ sunt destul de mari. Deoarece modelele generative reprezintă o arie de cercetare încă nefinisată, acestea nu sunt folosite momentan în producţie. În capitolul 3 vor fi prezentate diferite arhitecturi de reţele neurale, capabile să modeleze conversaţii.

Capitolul 3

Arhitectura

Modelarea unui limbaj reprezintă o sarcină extrem de dificilă pentru un calculator. Recente progrese în aria Deep Learning au făcut posibile diverse dezvoltări în această direcţie, însă lucrurile sunt departe de a fi rezolvate. Arhitectura folosită pentru construcţia chatbot-ului prezentat în această lucrare va fi expusă precum un bloc de construcţii, plecând de la noţiunile de bază, până la arhitectura finală.

3.1 Reţele neurale artificiale

O reţea neurală artificială (RNA) reprezintă o paradigmă bazată pe procesare de informaţii a cărei inspiraţii provine din sistemul nervos biologic. Precum creierul uman, o RNA este compusă dintr-un număr mare de elemente de procesare interconectate (neuroni) lucrând la unison pentru a rezolva diverse probleme. RNA, precum oamenii, învaţă din exemple. Învăţarea în sistemele biologice implică ajustarea conexiunilor sinaptice care există între neuroni. Acest principiu se aplică şi acestor reţele.

Ca şi modelare matematică propriu-zisă, o RNA poate fi observată în Figura 3.1. Avem o intrare reprezentată în figură de vectorul n-dimensional (x1, x2, x3, x4). Această intrare reprezintă caracteristicile unui eşantion care face parte dintr-un set de date. Straturile intermediare se numesc straturi ascunse, iar rolul lor este să producă o abstractizare cât mai complexă a datelor de intrare, folosind diverse funcţii cu activare neliniară. Ultimul strat se numeşte strat de ieşire, reprezentând eticheta (clasa) vectorului de intrare.

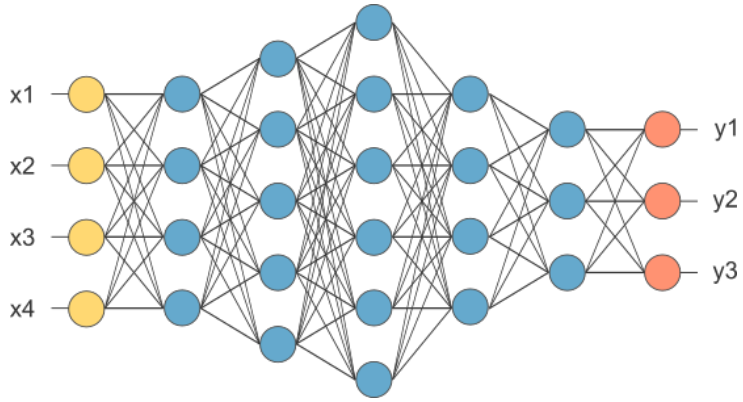

Figura 3.1: Exemplu de reţea neurală artificială

Ceea ce de fapt acest tip de reţele incearcă să inveţe sunt ponderile dintre straturile sale. Se pleacă de la un set de ponderi alese aleator [1] şi se ajustează folosind algoritmul de propagare înapoi a erorii până când reţeaua se stabilizează.

TO CONTINUE

3.2 Reţele neurale artificiale recurente

După cum se poate observa, o RNA este utilă atunci când intrarea este formată dintr-un singur element, de exemplu o imagine. Modelarea unui limbaj însă presupune ca intrările să fie fraze. O frază este formată din mai multe elemente (cuvinte), deci o RNA nu poate modela o astfel de intrare. Soluţia pentru această problemă este oferită de reţelele neurale artificiale recurente (RNAR). Acestea sunt folosite pentru a modela secvenţe unde există o dependenţă temporală (fraze, serii de timp). Fiecare intrare curentă din secvenţă este condiţionată de cele precedente. O astfel de reţea se poate observa în Figura 3.2. Straturile ascunse într-o RNAR au rolul de a păstra o captură a tot ceea ce s-a oferit ca intrare până în momentul curent. Valoarea fiecărui strat ascuns curent depinde astfel atât de intrarea curentă cât şi de ieşirea stratului ascuns anterior. Putem considera toată această modelare ca pe o probabilitate condiţionată $P(x_n|x_1, x_2, ..., x_{n-1})$, unde în cazul unei fraze $x_1, x_2, ..., x_n$ reprezintă cuvintele acesteia.

[1] Valorile ponderilor sunt deobicei subunitare iar iniţializarea este făcută urmând diverse principii matematice bine definite. Pentru cazuri simpliste, iniţializarea poate fi totuşi facută folosind o distribuţie uniformă.

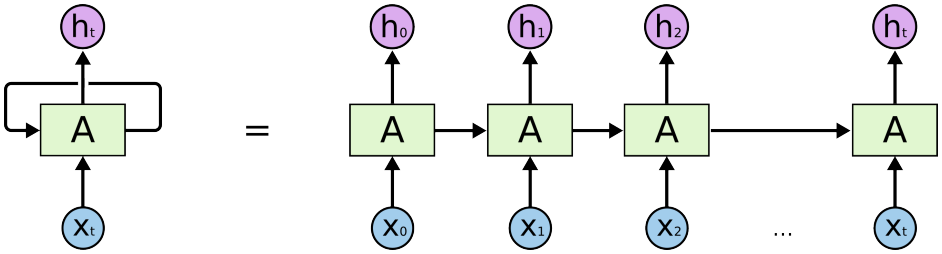

Figura 3.2: Exemplu de reţea neurală artificială recurentă

Algoritmul de reglare a ponderilor se numeşte propagarea înapoi în timp a erorii. O problemă serioasă care a apărut odată cu introducerea acestor reţele se numeşte risipirea gradienţilor. În cazul în care secvenţa de intrare are o lungime mare, informaţia nu reuşeşte să se propage în timp deoarece gradientul ponderilor se risipeşte, ponderile devenind 0. Mecanismul care înlătură această problemă se numeşte Long Short Term Memory (LSTM). Principiul este ca prin folosirea unor porţi de transmitere a informaţiei, gradientul să fie stabilizat. Acest mecanism a reprezentat un avans spectaculos pentru Deep Learning, permiţând modelarea secvenţială cu reţinere a informaţiei pe o perioadă îndelungată de timp. Ecuaţiile 3.1 descriu calculele pentru porţile LSTM.

$$\begin{aligned} f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\ i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\ o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\ c_t = f_t \circ c_{t-1} &+ i_t \circ \tanh(W_c x_t + U_c h_{t-1} + b_c) \\ h_t &= o_t \circ \tanh(c_t) \end{aligned} \tag{3.1}$$

Poarta f_t se numeşte poartă de uitare (forget gate). Rolul ei este să stabilească câtă informaţie se va uita din trecut. Poarta i_t este poarta de intrare (input gate). Aceasta delimitează care este cantitatea de informaţie care se păstrează din intrarea curentă. Poarta o_t este cea de ieşire (output gate). Ea controlează ce informaţie va fi transmisă către ieşire. c_t se numeşte starea internă a celulei LSTM. Aceasta este calculată ca o combinaţie între starea anterioară a celulei, poarta de intrare şi cea de ieşire. h_t reprezintă ieşirea curentă a reţelei şi depinde de poarta de ieşire şi starea celulei LSTM.

O RNAN aduce mai aproape ideea de modelare lingvistica, însă se poate observa că aceasta nu permite ca intrare decât o frază pe rând. Modelarea dorită este o pereche de forma întrebare-răspuns.

3.3 Modele Sequence-to-Sequence

Translaţia a reprezentat mereu un punct de interes în mediul procesării naturale de limbaj, constituind de altfel o provocare uriaşă de-a lungul ultimilor ani. Până în anul 2014, majoritatea modelelor de translaţie se bazau pe lanţuri Markov ascunse (Hidden Markov Models - HMM), totul urmând a se schimba odată cu introducerea unei noi arhitecturi în acel an de către Cho et Al. Noul tip de arhitectură se numea Sequence-to-Sequence (seq2seq) şi urma să aducă imbunătăţiri spectaculoase atât pe partea de translaţie cât şi pe partea conversaţională. Modelul este vizibil în Figura 3.3.

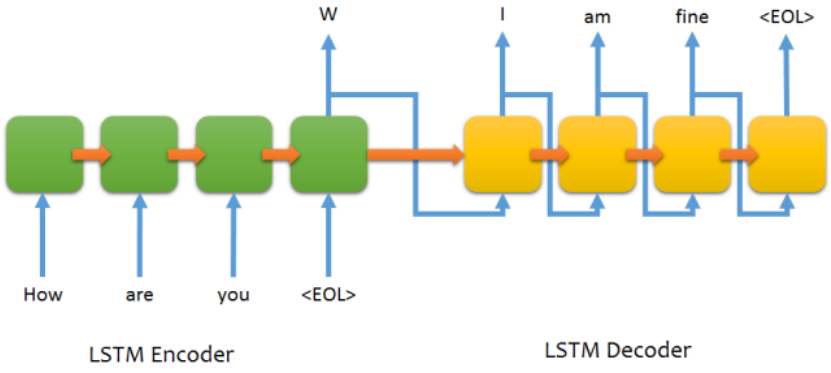

Figura 3.3: Modelul Sequence-to-Sequence

Aceast tip de arhitectură este împarţită în două jumătăţi: codor (encoder) şi decodor (decoder). Sarcina codorului este să primească o frază cu un număr variabil de cuvinte ca intrare iar unica sa ieşire [2] să fie o captură a întregii intrari (un vector de lungime fixă). Această captură poate fi privită ca o sumarizare a întregii fraze. Sarcina decodorului este de a învăţa fraze pornind de la ieşirea codorului, care va deveni prima intrare din decodor, şi restul intrarilor precedente. Intrarea curentă w_n în decodor este ieşirea de la timpul anterior, w_{n-1}. Modelarea se transformă astfel într-o probabilitate condiţionată: $P(w_n|w_1, w_2, ..., w_{n-1}, c)$, unde c reprezintă ieşirea codorului, deseori întâlnit sub numele de context în literatura de specialitate.

Intrările în acest model sunt de forma întrebare-răspuns. Original folosit ca model de translaţie, unde intrarea pentru codor era fraza într-o limbă iar intrarea în decodor era fraza tradusă în limba dorită, acest principiu se

[2] Ieşirea ultimului element din secvenţă

poate aplica la fel de uşor pentru a modela o conversaţie. Spre deosebire de traducere unde totul se întâmpla punctual iar răspunsul nu depinde decât de intrarea curentă, o conversaţie este foarte dependentă de un context. Fără acest context, partenerul angrenat în discuţie ar răspunde mereu luând în considerare doar ce i s-a spus în momentul de faţă, ignorând orice replică anterioară. Acesta nu este un comportament dorit în cadrul unei conversaţii şi de aceea modelul seq2seq nu este destul de puternic de sine stătător pentru a modela o discuţie.

3.4 Hierarchical Recurrent Encoder-Decoder

Modelarea contextului discuţiei reprezintă una dintre principalele nevoi în ceea ce priveşte o conversaţie care doreşte să pară cât mai reală. Până recent, cea mai apropiată arhitectură care realiza acest lucru era modelul seq2seq, însă contextul era reţinut doar la nivelul unei singure fraze. În anul 2016, Iulian Şerban et Al. a introdus reţeaua Hierarchical Recurrent Encoder-Decoder (HRED - Figura 3.4). Ea poate fi vazută ca o extensie peste seq2seq. Avantajul acesteia este că poate reţine contextul discuţiei.

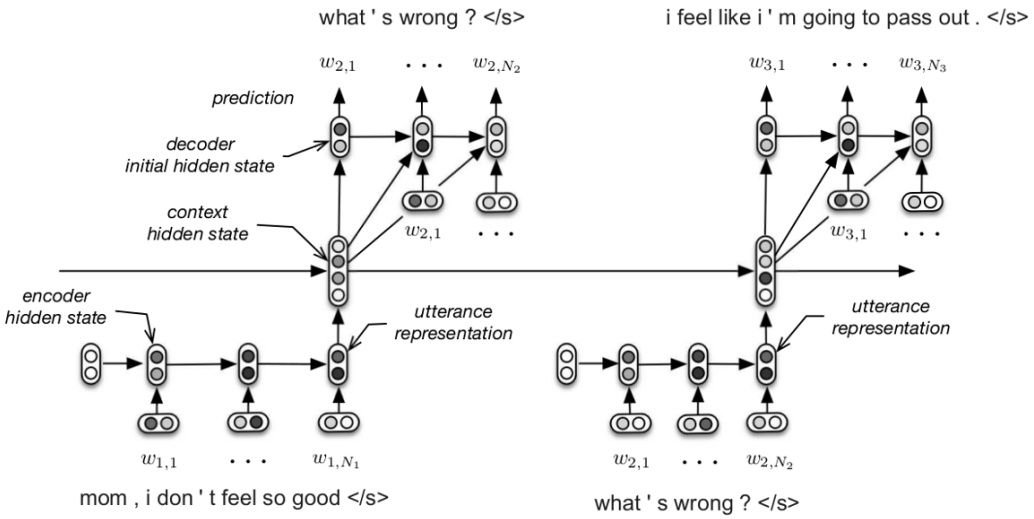

Figura 3.4: Hierarchical Recurrent Encoder-Decoder

După cum se poate observa, modelul este construit prin alăturarea mai multor reţele seq2seq legate între ele printr-o RNAR contextuală. După cum sugerează şi numele, rolul acestei RNAR este de a reţine contextul discuţiei de-a lungul mai multor fraze. Spre deosebire de seq2seq, ieşirea codorului nu

mai este oferită direct ca şi primă intrare pentru decodor, aceasta trecând mai întai prin RNAR contextuală. Pentru a fi mai uşor de înteles ce se întamplă, ne putem imagina toate cele n codoare şi decodoare ca fiind simple unităţi de intrare, respectiv ieşire într-o RNAR obişnuită. Precum sunt într-o RNAR ieşirile dependente atât de intrarea curentă, cât şi de cele precedente, prin analogie putem deduce că într-o reţea HRED, fiecare ieşire depinde de fraza curentă şi de contextul discuţiei (frazele precedente).

3.4.1 Celula GRU

Gated Recurrent Unit sau GRU este un tip de celulă pentru o RNAR, menită să o înlocuiască pe cea LSTM. Ecuaţiile din spatele celulei(3.2) sunt asemănătoare cu cele LSTM.

$$\begin{aligned} z &= \sigma(x_t U^z + s_{t-1} W^z) \\ r &= \sigma(x_t U^r + s_{t-1} W^r) \\ h &= \tanh(x_t U^h + (s_{t-1} \circ r) W^h) \\ s_t &= (1 - z) \circ h + z \circ s_{t-1} \end{aligned} \tag{3.2}$$

GRU are doar două porţi: poarta r de resetare (reset gate) şi poarta z de actualizare (update gate). Intuitiv, poarta de resetare determină cum se va combina noua intrare cu memoria precedentă iar cea de actualizare defineşte cât de multă memorie anterioară se păstrează. Principalele diferenţe între GRU şi LSTM sunt:

- GRU are doar două porţi, pe când LSTM trei.
- GRU nu posedă o memorie internă c_t care să fie diferită de starea ascunsă expusă. Nu conţine poarta de ieşire care este prezentă pentru LSTM.
- Poarta de intrare şi cea de ieşire sunt cuplate de poarta de actualizare, iar poarta de resetare este aplicată direct stării ascunse anterioare. Astfel, responsabilitatea porţii de resetare din LSTM este împărţită între cea de resetare şi cea de actualizare de la GRU.
- Nu se aplică o a doua neliaritate atunci când se calculează ieşirea.

Nu există o arhitectură câştigătoare clară între LSTM şi GRU. Având mai puţini parametri (U şi W), GRU se antrenează mai rapid şi are nevoie de mai puţine date pentru a generaliza. Pe de altă parte, o cantitate mare de date exprimă o putere mai mare de modelare din partea LSTM şi astfel

se pot obţine rezultate mai bune. Pentru realizarea acestei aplicaţii a fost aleasă o arhitectură de tip GRU.

3.4.2 HRED bidirecţional

Rolul codorului, precum a fost menţionat mai devreme, este să captureze informaţia unei fraze într-un vector de lungime fixă. În orice limbaj, sensul unui cuvânt nu este stabilit doar de cuvintele precedente ci şi de cele ce vor urma. Deoarece RNAR modelează cuvintele dintr-o frază pornind de la cuvântul w_1 la w_n, înţelesul din viitor al acestora este ignorat, iar captura frazei poate să nu fie îndeajuns de reprezentativă. Astfel, este propusă folosirea unei RNAR bidirecţionale (Figura 3.5) pentru codor. Acest tip de reţea foloseşte două parcurgeri ale secvenţei de intrare: cea înainte, care se desfăşoară în mod obişnuit şi cea inversă (de la w_n la w_1), pentru capturarea întelesului din viitor al cuvintelor. Ieşirea pentru pargurgerea înainte va fi la timpul n iar cea pentru parcurgerea inversă va fi la timpul iniţial. Vectorul de lungime fixă va fi format din concatenarea celor două ieşiri ale reţelei bidirecţionale.

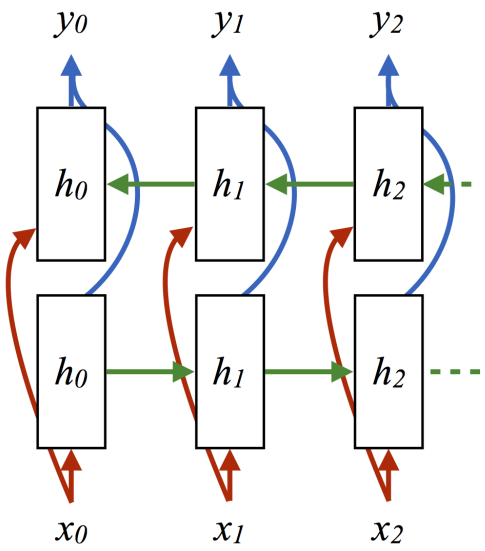

Figura 3.5: RNAR bidirecţional

3.5 Word Embeddings

TODO

3.6 Predicţie

Având modelul antrenat, acesta trebuie folosit pentru predicţie, adică generarea unor răspunsuri adecvate pentru contextul discuţiei. Generarea cuvintelor este condiţionată atât de context, cât şi de cuvintele anterior generate. Spre deosebire de antrenare, în momentul predicţiei, secveţa de ieşire este goală iniţial (un vector de zerouri). Cuvintele sunt generate secvenţial, unul câte unul şi adăugate pe poziţia t la care a ajuns secvenţa. Deoarece stratul softmax al secvenţei de ieşire returnează un vector de probabilităţi peste toate cuvintele din vocabular, pentru a returna cuvântul dorit se alege cel cu probabilitatea cea mai mare. Pornind de la această idee, există două variante de generare: metoda greedy şi beam search.

3.6.1 Greedy

Precum în teoria clasică a algoritmicii, scopul metodelor greedy este să selecteze întotdeauna optimul local. În cazul de faţă, optimul local reprezintă cuvântul cu probabilitatea de apariţie cea mai mare. Prin acestă metodă nu este garantat la final că fraza produsă este cea mai bună cu putinţă, deoarece o altă combinaţie de cuvinte poate produce oricând o frază cu un scor mai bun[3]. Astfel, singurul avantaj al acestei metode este viteza de predicţie rapidă. În practică, se preferă folosirea unor algoritmi mai complexi, capabili să aleagă fraza cea mai potrivită dintr-un set de fraze canditat.

3.6.2 Beam search

Algoritmul beam search foloseşte ideea de arbore de căutare pentru a genera fraze (Figura 3.6). În locul alegerii cuvântului cu probabilitatea cea mai mare la fiecare pas, se vor alege top k cuvinte cu cele mai mari probabilităţi. În literatură, k se mai numeşte şi beam size. Fiecare nod (cuvânt) ales va produce un număr n_c de copii. În felul acesta este garantată generarea mai multor fraze canditat din care se va alege. Fiecare frază are un scor $S(f)$, care este de forma ecuaţiei 3.3, unde $P(w_i)$ reprezintă probabilitatea cuvântului ales. O frază devine canditat atunci când cuvântul curent generat este simbolul de sfârşit de propoziţie. În momentul în care algoritmul a terminat de generat toate frazele canditat, cea mai bună este determinată ca fiind cea cu scorul $S(f)$ cel mai mare.

[3]Sumă din logaritmul probilităţilor frazei

$$S(f) = \frac{1}{n} \sum_{i=0}^{n} \log P(w_i) \tag{3.3}$$

Pentru ca acest arbore de căutare să nu capete o creşştere exponenţială, la fiecare pas se păstrează un numar egal cu beam size cele mai bune fraze iar restul sunt decartate. Acest algoritm nu asigură de departe optimul global, însă spaţiul căutării oferit de folosirea unui arbore este mult mai bine dezvoltat şi ales decât la metoda greedy. Cu cât beam size si numărul de copii generaţi sunt mai mari, cu atât algoritmul oferă soluţii cât mai apropiate de optimul global. De asemenea, odată cu creşterea acestor parametri, creşte şi timpul de căuture şi generare în arbore. De aceea, trebuie să existe un compromis între calitatea predicţiei şi viteza de execuţie a algoritmului.

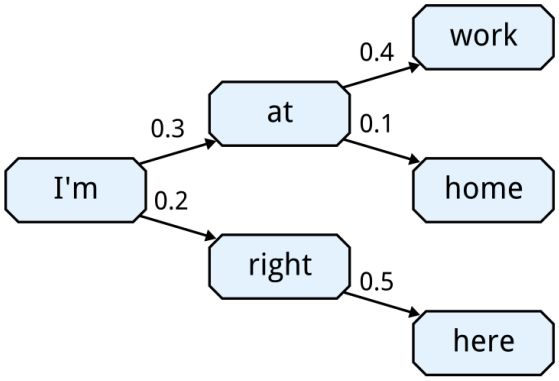

Figura 3.6: Arbore de căutare pentru algoritmul beam search