VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

STROJOVÉ UČENÍ PRO ODPOVÍDÁNÍ NA OTÁZKY V ČEŠTINĚ

MACHINE LEARNING FOR QUESTION ANSWERING IN CZECH

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. PETER PASTOREK**
AUTHOR

VEDOUCÍ PRÁCE — **prof. RNDr. PAVEL SMRŽ, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání diplomové práce

Student: **Pastorek Peter, Bc.**
Program: Informační technologie Obor: Inteligentní systémy
Název: **Strojové učení pro odpovídání na otázky v češtině**
Machine Learning for Question Answering in Czech
Kategorie: Databáze

Zadání:

1. Seznamte se s metodami odpovídání na otázky v přirozeném jazyce se zaměřením na techniky strojového učení pro extrakci odpovědí, případně přenos výsledků napříč jazyky
2. Shromážděte data pro testování průběžného vývoje i pro závěrečné vyhodnocení úspěšnosti zvolených metod v českém jazyce.
3. Navrhněte a realizujte systém, který s využitím existujících implementací dokáže odpovídat na otázky nad českou wikipedií, případně nad doplňkovými zdroji informací.
4. Vyhodnoťte výsledky systému na shromážděných datech a diskutujte možná zlepšení.
5. Vytvořte stručný plakát prezentující práci, její cíle a výsledky.

Literatura:

- dle domluvy s vedoucím

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Smrž Pavel, doc. RNDr., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 20. května 2020
Datum schválení: 1. listopadu 2019

Abstrakt

Táto diplomová práca sa zaoberá učením neurónových sietí na odpovedanie otázok v češtine. Neurónové siete sú vytvorené v jazyku Python použitím knižnice PyTorch. Vytvorené sú na základe štruktúry LSTM. Učené sú na českej dátovej sade SQAD. Pretože dátová sada je menšia ako anglické dátové sady, rozširujem neurónové siete o algoritmické postupy. Pre jednoduchšiu aplikáciu algoritmov a lepšiu presnosť rozdeľujem odpovedanie na otázku do menších častí.

Abstract

This Master's thesis deals with teaching neural network question answering in Czech. Neural networks are created in Python programming language using the PyTorch library. They are created based on the LSTM structure. They are trained on the Czech SQAD dataset. Because Czech data set is smaller than the English data sets, I opted to extend neural networks with algorithmic procedures. For easier application of algorithmic procedures and better accuracy, I divide question answering into smaller parts.

Klíčová slova

strojové učenie, neurónové siete, spracovanie prirodzeného jazyka, odpovedanie na otázku

Keywords

machine learning, neural network, natural language processing, question answering

Citace

PASTOREK, Peter. *Strojové učení pro odpovídání na otázky v češtině.* Brno, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. RNDr. Pavel Smrž, Ph.D.

Strojové učení pro odpovídání na otázky v češtině

Prohlášení

Prehlasujem, že som túto semestrálnu prácu vypracoval samostatne pod vedením pána doc. RNDr. Pavla Smrže, Ph.D. Ďalšie informácie mi poskytol pán Ing. Martin Fajčík. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Peter Pastorek
3. června 2020

Poděkování

Chcel by som sa poďakovať vedúcemu práce pánovi doc. RNDr. Pavlu Smržovi, Ph.D. a Ing. Martinovi Fajčíkovi za smerovanie pri práci a odbornú podporu.

Obsah

Kapitola 1

Úvod

Najčastejšie sa vyvíja odpovedanie na otázky ako spôsob získania dát z textu, ale odpovedanie na otázky môže byť považované za najvšeobecnejší problém spracovania prirodzeného jazyka, pretože ďalšie problémy spracovania prirodzeného jazyka sa je možné vyjadriť ako otázky (Aký je preklad textu do angličtiny? Aký má text sentiment? Ktoré pomenované entity sa nachádzajú v texte? . . .) [8, 13].

Moja diplomová práca sa zameriava na adaptovanie výskumu ohľadom odpovedania na otázky do češtiny. Spracovanie češtiny strojovým učením bude mať dva problémy, ktoré pri angličtine nie sú až také závažné. Prvým problémom je menšie množstvo dát pre strojové učenie. Pri vytváraní a rozširovaní dátových sád pre spracovanie prirodzeného jazyka je potreba ľudská asistencia. Ľudí, ktorý vyvíjajú české dátové sady, je oveľa menej ako ľudí, ktorý vyvíjajú anglické dátové sady. Čeština je jazyk s bohatou morfológiou a pomerne voľným slovosledom, a to môže spôsobovať problémy pri strojovom spracovaní češtiny. Slová v češtine majú rôzne tvary v rôznych gramatických kategóriách. Jednoduché strojové spracovanie môže považovať dve slová opisujúce ten istý predmet ako dva predmety, pretože slová majú rozdielny tvar. V angličtine ohýbanie slov nie je až taký veľký problém ako v češtine.

Zbytok tejto technické správy se skladá z následujúcich častí: Nasledujúca kapitola sa zoberá výskumom odpovedania na otázky v angličtine a aplikáciou daného výskumu na češtinu. Dátová sada na učenie neurónových sietí je popísaná v kapitole 3. Návrh mnou vytvoreného systému je v kapitole 4. Výsledky systému sú v kapitole 5.

Systém na odpovedanie otázok som sa riešil riešiť v programovacom jazyku Python s použitím knižnice PyTorch na základe kódu vytvoreného Ing. Martinom Fajčíkom a poskytnutého študentom Vysoké učení technické v Brně Fakulty informačních technologií. Spracovanie prirodzeného textu som riešil primárne neurónovými sieťami na základe štruktúry LSTM naučenými na českej dátovej sade SQAD_v3. Pretože neurónové siete majú nevýhody na malej dátovej sade rozhodol som sa ich rozšíriť s algoritmickými riešeniami. Najlepší algoritmus, ktorý som testoval, bol algoritmus tf.idf. Algoritmus tf.idf v jeho základnej forme sa používa na vypočítanie relevancie vety alebo textu k súboru dokumentov. Pretože algoritmus tf.idf nie je použiteľný na celý problém odpovedania na otázku, rozdelil som ho na menšie časti. Základné časti sú získanie článku, redukcia kontextu pre odpoveď a extrakcia odpovede na otázku. Kvôli rôznym druhom odpovedí, extrakcia odpovede bude musieť byť rozdelená na ďalšie menšie časti.

Cieľom odpovedania na otázku po krokoch je aj zlepšenie presnosti a zmenšenie výpočtovej náročnosti. Na začiatku má veľké množstvo dát, ktoré nie je možné podrobne spracovať.

Postupne sa zmenší počet dát a mala by sa zlepšiť podrobnosť spracovania. Systém by sa mal postupne dopracovať bližšie k odpovedi.

Kapitola 2

Odpovedanie otázok nad textom

Na odpovedanie otázok a spracovanie prirodzeného jazyka sa najčastejšie prevedú slová na vektory a potom sa na získané vektory aplikuje reťaz vektorových operácií. Najobľúbenejší základ pre systémy na odpovedanie otázok sú rekurentné neurónové siete rozšírené o pamäť a to najme Long short-term memory (LSTM). LSTM je vhodný aj pre češtinu, pretože nepotrebuje pre fungovanie externé zdroje, ktoré neexistujú alebo nie sú kvalitné v češtine. [5, 8, 30, 24, 1, 12, 26]

Systémy na odpovedanie otázok sa bežne rozdeľujú na systémy s uzavretou doménou a systémy s otvorenou doménou. Systém s otvorenou doménou dostane ako vstup iba otázku. Systém s uzavretou doménou potrebuje ako vstup otázku a aj text, v ktorom má hľadať odpoveď. Systémy s otvorenou doménou riešia veľké množstvo dát na spracovanie postupným izolovaním odpovede. Systém s uzavretou doménou často extrahuje odpoveď z textu v jednom kroku. Systém s otvorenou doménou nemusí vedieť počet krokov, ktoré použije pri získaní odpovede, pred pokusom o získanie odpovede. [12, 31, 22, 2, 23]

2.1 Prevedenie textu na vektory

Na to aby sa nad textom mohli robiť rôzne operácie, je potrebné text transformovať do vhodného, matematike priateľskejšieho tvaru. Ako štandard sa v poli spracovania prirodzeného jazyka usadil pojem "embedding". Ide o prevedenie textu na reťazec vektorov. Najčastejšie sa na vektor transformujú slová. Transformácia väčších celkou, ako napríklad viet, na vektor je možná, ale ťažko sa štandardizuje a pri znížení rozlíšenia sa potenciálne zníži aj presnosť. Transformovať sa dajú aj jednotlivé písmená. Transformácia jednotlivých písmen sa v niektorých prípadoch robí a aj dokáže zlepšiť prenos učeného systému, ale zároveň drasticky zväčší veľkosť spracovávaných dát. Transformácia jednotlivých písmen sa používa iba keď sa spracováva malé množstvo textu inak časová a pamäťová zložitosť presiahnu hranicu strojov, ku ktorým má bežní smrteľník prístup. [12, 9, 10]

Vektor patriaci k slovu závisí na sémantickom význame slova. Synonymá a významovo podobné slová majú podobné vektory. Zároveň boli experimentálne získané ďalšie vlastnosti vektorov. Pri súčte vektorov vznikne vektor podobný spojeniu sémantického významu slov. Napríklad súčet vektorov pre človek a lyže vznikne vektor podobný slovu lyžiar. Podobne to funguje aj s odčítaním vektorov. Ak sa od vektoru patriacemu slovu kráľ odčíta vektor slova muž a pričíta vektor slova žena, tak by mal vzniknúť vektor podobný vektoru slova kráľovná. Rovnaké slová v rôznom gramatickom tvare sa pri prevedená považujú za rozdielne slová. Z pohľadu mapovania textu na vektory sa môžu považovať za synonymá, a preto dobré

mapovanie by im malo dať podobné vektory. Pretože mapovanie textu na vektory je riešené pomocou algoritmu bez zásahu človeka, aj menej populárne jazyky vy mali mať kvalitný systém na mapovanie textu na vektory. Potrebné je iba veľké množstvo textu v danom jazyku. Pretože je text konvertovaný na vektor, problém rôznych tvarou českých slov sa dá považovať za čiastočne vyriešený. [12, 9, 6, 10]

Veľkosť vektora sa pohybuje medzi 100 až 300 dimenziami. 100 dimenzií je štandardná veľkosť vektora. Pri 100 dimenzionálnych vektoroch je presnosť dostatočne dobrá a zároveň výpočty s nimi nie sú veľmi časovo a pamäťovo náročné. Pri 300 dimenzionálnych vektoroch je presnosť lepšia ako pri 100 dimenzionálnych vektoroch, ale sú časovo a pamäťovo zložité. Pod 100 dimenziách je presnosť nedostatočná a nad 300 dimenzií sa presnosť takmer vôbec nezlepšuje. [12]

Transformácia slova na vektor je momentálne jediná časť spracovania prirodzeného jazyka, ktorá sa dá zdielať medzi rôznymi problémami spracovania prirodzeného jazyka. Považuje sa za normálne nevytvárať vlastné mapovanie, ale použiť už naučený systém. [12, 10]

2.2 Long short-term memory

Long short-term memory sa dá preložiť do slovenčiny ako dlhá krátkodobá pamäť. Namiesto celého názvu Long short-term memory sa používa skratka LSTM. LSTM je špecializovaná verzia rekurentnej neurónovej siete. Rekurentné neurónové siete sú výnimočné pretože neberú iba jeden vektor, ale spracovávajú reťazec vektorov. Vektory v reťazci musia mať rovnakú a predom určenú veľkosť, ale dĺžka reťazca vektorov sa môže meniť. LSTM je rekurentná sieť špecializovaná na šírku kontextu. Pri rekurentných neurónových sietiach je problém miznúceho gradientu, a preto výsledný vektor slova záleží iba od malého okolia slova. Obyčajná rekurentná neurónová sieť má iba krátkodobú pamäť. LSTM je špeciálne upravená rekurentná neurónová sieť zameraná na odstránenie problému miznúceho gradientu, a tým rozšíri dosah pamäte. LSTM zabúda pomalšie ako obyčajná rekurentná neurónová sieť, a preto je dlhá. [4, 12, 19, 3, 25]

Výstup LSTM je reťazec vektorov rovnakej dĺžky ako vstupný reťazec, ale veľkosť a hodnota vektorov je nová. Keď sa zoberie text a každé slovo sa prevedie na vektor rovnakej dĺžky, tak takto vytvorený reťazec spĺňa všetky podmienky pre použitie LSTM. V takom prípade bude výstupom LSTM reťazec vektorov dlhý ako počet slov v texte, ale namiesto vektoru reprezentujúceho slovo bude vo výstupe vektor reprezentujúci vlastnosť, ktorá bola naučená LSTM. Neznáma dĺžka vety a tým aj neznáma dĺžka reťazca vektorov je dôvod, ktorá robí z LSTM a podobných modelov najpopulárnejšie konštrukcie neurónovej siete používané na spracovanie prirodzeného jazyka. [4, 12, 19, 3]

Pri použití LSTM na odpovedanie otázky nastanú dva problematické body. Ten menší z nich je redukcia textu na vektor. Použitie posledného stavu LSTM je jedna z validných možností redukcie textu na vektor. Ďalšia možnosť je spracovanie výstupného reťazca vektor LSTM. Je potrebné zmenšiť reťazec vektorov neznámej dĺžky bez použitia ďalšej rekurentnej neurónovej siete, pretože LSTM už je rekurentná neurónová sieť. Prvok ako lineárny filter sa nedá použiť, pretože potrebuje predom známu veľkosť vektora. Lineárny filter by sa dal použiť keď sa mu vstup oreže ale vyplní na presnú dĺžku. Algoritmické riešenia ako získanie maxima cez vektory alebo získanie priemeru cez vektory sú možné. Komplikovanejší problém pri odpovedaní na otázky je kombinácia otázky a textu. Kombinovať otázku a text sa dá násobením matice textu a matice otázky alebo konkatenáciou. Pri násobenie matíc sa môže zároveň použiť pre redukciu dĺžky vektora, správnym upravením dimenzie matíc. Pri

konkatenácií sa vektory a matice vždy zväčšujú, a preto konkatenácia musí byť nasledovaná komplexnejším spracovaním ako produkt matíc. [4, 12]

Pri spracovaní textu sa LSTM väčšinou vykonáva z oboch strán. Pri obojstrannom vykonávaní hodnota jedného výstupného vektora záleží od vstupného vektora a vektorov okolo pozície vstupného vektora. Takže vektor vypočítaný pre slovo záleží na kontexte okolo slova. [4, 12, 25]

Najjednoduchšie použitie LSTM

LSTM sa dá veľmi jednoducho použiť na klasifikáciu prirodzeného textu. Najskôr sa text prevedie na vektory. Reťazec vektorov sa pošle cez obojstranné LSTM, ale získa sa z neho posledný vnútorný stav LSTM. Ak je potrebné text klasifikovať do piatich kategórií, tak sa výstupný vektor LSTM zmenší na päť dimenzií použitím lineárneho filtra. Lineárny filter obsahuje vnútornú pamäť, ktorá sa učí. Vypočítané skóre pri dimenzii určuje príslušnosť textu danej kategórií. [4, 12]

Pri vyberaní odpovede z textu sa najčastejšie predpovedá začiatok a koniec odpovede. Najskôr sa otázka a text prevedú na vektory. Otázka sa konkatenuje pred text. Reťazec vektorov sa pošle cez obojstranné LSTM a získa sa reťazec vektorov vlastností dlhý ako súčet dĺžky otázky a textu. Z vektoru vlastností sa zahodí časť pre otázku, ktorá slúžila iba na ovplyvnenie výpočtu vlastností pre text. Každý vektor pre vlastnosť sa zmenší na jedno číslo napríklad lineárnym filtrom. Ak budú použité dva lineárne filtre, tak sa naraz získa predpoveď začiatku aj konca odpovede. Teoreticky by to malo fungovať, ale efektivita bude klesať s dĺžkou textu. [4, 12]

2.3 Dynamic memory network

Dynamic memory network bol vytvorený za účelom použitia jednej architektúry na rôzne problémy spracovania prirodzeného jazyka. Architektúra dynamic memory network sa dá použiť na odpovedanie otázok, analyzovanie sentimentu alebo klasifikáciu pomenovaných entít. Pri probléme ako analyzovanie sentimentu je otázka vždy rovnaká: „Aký má text sentiment?“ Medzi rôznymi problémami spracovania prirodzeného jazyka sa zdieľa iba architektúra. Jeden systém nemôže byť naučený na dva rôzne problémy spracovania prirodzeného jazyka. [12, 8, 30]

Dynamic memory network sa skladá zo štyroch častí: [12, 8, 30]

- Vstupný modul spracováva text článku z Wikipédie. Text zakóduje do vektorovej reprezentácie. Vektorová reprezentácia musí popisovať vlastnosti textu článku. Modul používa LSTM alebo inú podobnú sieť ako napríklad Gated recurrent network (GRU). Vstupný text sa rozdelí na vety a za každou vetou sa vloží špeciálny symbol, ktorý sa v texte nenachádza a určí konce viet. Každá veta sa považuje za jeden fakt. LSTM spracuje reťazec vektorov, ktoré reprezentujú text článku, na nový reťazec vektorov, ktoré reprezentujú vlastnosti článku. Pri spracovaní textu článku pomocou LSTM sa text článku nerozdeľuje, ale spracováva sa celý naraz. Výstupom vstupného modulu je reťazec vektorov, ale má dĺžku počtu faktov a nie dĺžku článku. Výstupný vektor sa poskladá z výstupu LSTM na miestach špeciálnych symbolov, ktoré určujú koniec vety a tým aj fakt. [12, 8, 30]

- Modul otázky zakóduje otázku do vektorovej reprezentácie. Vektorová reprezentácia musí popisovať vlastnosti otázky. Vektor otázky sa použije na inicializáciu epi-

zodickej pamäte. Modul otázky používa LSTM alebo inú podobnú sieť ako napríklad Gated recurrent network (GRU). LSTM spracuje reťazec vektorov, ktoré reprezentujú otázku, na nový reťazec vektorov, ktoré reprezentujú vlastnosti otázky. Výstup modulu otázky musí byť jeden vektor, a preto sa použije posledný vektor reťazca. [12, 8, 30]

- Epizodická pamäť má na vstupe reťazec vektorov reprezentujúci fakty a vektor reprezentujúci otázku. V epizodickej pamäti je vnútorná pamäť, mechanizmus pozornosti a rekurentná neurónová sieť ako LSTM alebo GRU. Mechanizmus pozornosti zoberie vektor faktu, vektor vnútornej pamäte a vektor otázky. Z mechanizmu pozornosti sa získa dôležitosť faktu v kontextu otázky a dosial získaných znalostí, ktoré sú uložené vo vnútornej pamäti. Pozdvihnutím alebo utlmením jednotlivých faktov aplikovaním mechanizmu pozornosti sa získa epizóda. Nový stav vnútornej pamäte sa získa aplikovaním rekurentnej neurónovej siete (LSTM) na epizódu, kde počiatočný stav LSTM je predchádzajúci stav vnútornej pamäte. Úplne prvý stav vnútornej pamäte sa inicializuje na vektor otázky. Modul sa nazýva epizodická pamäť pretože má vnútornú pamäť a proces výpočtu epizódy sa počíta niekoľko krát za iných podmienok. Podmienka, ktorá sa mení, je stav vnútornej pamäte. Každý problém spracovania prirodzeného jazyka má iný optimálny počet cyklov epizodickej pamäte. Najlepší počet cyklov pre problém spracovania prirodzeného jazyka bol získaný experimentálne. Odpoveď na otázky sa často nenachádza v jednej vete (jednom fakte), ale je potrebné nájsť odpoveď spojením faktov z niekoľko viet. [12, 8, 30]

- Výstupný modul vytvorí odpoveď z finálneho vektora epizódy epizodickej pamäte, finálneho stavu vnútornej pamäte epizodickej pamäte, výstupu modulu otázky a vstupu. Pomocou finálneho vektora epizodickej pamäte sa izoluje veta (fakt), ktorá obsahuje odpoveď. Z izolovanej vety (faktu) a vektora odpovede sa vyberie slovo alebo slovné spojenie, ktoré budú tvoriť odpoveď. Výstupný model na získanie odpovede použije rekurentnú neurónovú sieť ako LSTM alebo GRU. LSTM je inicializovaná na finálny stav vnútornej pamäte epizodickej pamäte. LSTM bude postupne dostávať na vstup vstupné vektory s prilepeným vektorom otázky. Na vybranie odpovede sa najčastejšie používa predpovedanie začiatku a konca odpovede, ale rôzne problémy spracovania prirodzeného jazyka musia mať špeciálne upravené výstupné moduly. [12, 8, 30]

2.4 Kompozitná neurónová sieť

Odpovedanie na otázku je špeciálne v tom že má dva vstupy otázku a text, ktorý má obsahovať odpoveď. Otázka a text majú rovnakú formu prirodzeného jazyka, ale nemajú rovnakú dôležitosť a význam. Väčšinou sa otázka a text spája vytvorením produktu vektorov alebo konkatenáciou vektorov [12]. V kompozitnej neurónovej sieti sa vytvára neurónová sieť, ktorá bude spracovávať text článku, na základe konštrukcie otázky. Neurónová sieť na spracovanie článku sa skladá z modulov. Každý z modulov je špecializovaná neurónová sieť. Zdrojový článok identifikuje nasledovné moduly: [1]

- Modul Lookup vyhľadá slová v texte a zdôrazni ich. Môže sa použiť na vyhľadanie identifikačných slov, ktoré sú prítomné v otázke, v texte. [1]

- Modul Find spracováva získané údaje a pokúsi sa nájsť predmet modulu zadaný. Môže sa použiť na nájdenie predmetu otázky. [1]

- Modul Relate zdôraznení slovo v texte na základe iného zdôraznenia. Môže sa použiť keď je dôležité slovo iba v prítomnosti špecifického kontextu. [1]
- Modul And rozvetvuje neurónovú sieť podľa spojky a (and). Môže sa použiť keď je potrebné splniť viacero podmienok zároveň. [1]
- Modul Describe produkuje odpoveď na základe vypočítaného zdôraznenia. Používa sa keď odpoveď je niekoľko slov. [1]
- Modul Exists produkuje odpoveď na základe vypočítaného zdôraznenia, ale odpoveď je iba áno alebo nie. [1]

Moduly sa zostavujú podľa sémantickej analýzy otázky. Napríklad otázka: „Aké mestá sú na Morave?“ sa prevedie na Describe(And(Find(mestá), Relate(na, Lookup(Morave)))). Sieť vyhľadá všetky mestá a všetky entity na Morave. Potom vráti mená entít, ktoré splňujú obe podmienky. [1]

Na trénovanie systému by mali stačiť trojice: otázky, textu a odpovede. Model sa naraz učí zostavovať moduly a aj váhy v jednotlivých moduloch. Rozdelením naučených častí do modulov sa zabezpečí veľká hĺbka neurónovej siete, ale narozdiel od konvenčných hlbokých neurónových sietí je kompozitná neurónová sieť rýchlejšia a aj efektívnejšia na menšej dátovej sade. Kompozitná neurónová sieť sa špecializuje na otázky ohľadom objektov na obrázku a geografické otázky. [1]

2.5 Zlepšenie predpovede použitím kategórie odpovede

Odpovede sa dajú rozdeliť do kategórií. Ak má neurónová sieť explicitný prístup ku kategórií odpovede, tak by sa presnosť predpovedania odpovede mala zlepšiť. Predpovedanie kategórie odpovede je oveľa spoľahlivejšie ako predpovedanie odpovede. Kategórie sú napríklad: osoba, miesto, organizácia, dátum alebo číslo. Keď sa použije predikcia kategórie ako vstup pre predpoveď odpovede, tak predpovedaná odpoveď by mala byť bližšie častiam textu, ktoré patria danej kategórií. Aj keď odpoveď bude predpovedaná nesprávne, tak stále by mala byť vybraná odpoveď danej kategórie. Al sa očakáva dátum ako odpoveď, tak explicitné predpovedanie kategórie odpovede zväčší pravdepodobnosť dátumu ako odpoveď. Stroj učený na predikciu odpovede by mal byť schopný rozpoznať kategóriu odpovede, ale explicitné izolovanie kategórie odpovede by malo byť spoľahlivejšie a menej náchylnejšie na náhodnosť pri učení. [3]

Ak rozdielne kategórie potrebujú rozdielny formát výstupu, tak sa predikcia kategórie môže použiť na vybranie špeciálneho výstupného modulu. Každý výstupný modul je špeciálne naučený dať určitý výstup. Napríklad jeden výstupný modul môže byť naučený na extrakciu odpovede z textu a druhý výstupný modul môže byť naučený na kategorizovanie kombinácie otázky a textu. Keď sa očakáva odpoveď na otázku "áno"alebo "nie", tak je formát odpovede drasticky rozdielny než časť z textu. Pri rozdelených výstupných moduloch sa nezdieľa pochopenie vstupného textu medzi modulmi a dátová sada je rozdelená medzi výstupné moduly. Jeden výstupný modul, ktorý pokýva všetky formáty výstupov, by mal byť teoreticky presnejší ako viacero výstupných modulov, ale použitie viacero modulov môže byť jednoduchšie rozšíritelne. [19]

2.6 Riešenie odpovedania na otázky bez strojového učenia

Odpovedania na otázku sa ťažko robí bez použitia strojového učenia, ale existujú gramatiky, ktoré dokážu odpovedať na otázku aj bez strojového učenia. Keď sa rozdelí problém odpovedania na otázku do rôznych častí, tak niektoré časti sa dajú riešiť algoritmom oveľa jednoduchšie. Pri menšej dátovej sade ako sú české dátové sady je zaujímavé rozšírenie a porovnanie algoritmického riešenia s naučenou neurónovou sieťou. [27, 12, 21]

Priradenie vety, ktorá obsahuje odpoveď, k otázke sa dá riešiť použitím algoritmu. Každá veta v texte sa môže považovať za samostatnú jednotku. Veta sa môže vybrať skúmaním podobnosti vety a otázky. Ak veta obsahuje rovnaké kľúčové slová ako otázka, tak môže obsahovať odpoveď. Editačná vzdialenosť vety a otázky môže byť tiež indikátor existencie odpovede vo vete. Problém je v tom že vety nie sú od seba nezávislé. Skóre priradené jednej vete môže byť ovplyvnené zo skóre vedľajších viet. Je to podobné prvku LSTM pri učení neurónových sietí, ale narozdiel od strojového učenia je potrebné veľkosť ovplyvnenie vedľajších viet určiť manuálne [12]. Priradenie článku z Wikipédie otázke sa dá riešiť podobne ako priradenie vety otázke, ale články z Wikipédie sú od seba naozaj nezávislé. Hľadanie podobnosti medzi otázkou a prvých pár viet článku, ktoré obsahujú najdôležitejšie a najobecnejšie informácie o článku, by malo mať dobrú úspešnosť.

Pridanie článku otázke sa dá urobiť aj pomocou bežného internetového vyhľadávania. Odpovedanie na otázku je špecifický a rozšírený problém internetového vyhľadávania. Internetové vyhľadávanie používa algoritmické riešenia a aj strojové učenie. Internetové vyhľadávače sa bežne používajú na získanie dokumentu alebo článku na ďalšie spracovanie [19].

Algoritmus tf.idf

Algoritmus tf.idf je rozšírená verzia algoritmu Bag of Words. Bežne sa používa na priradenie vety dokumentu na základe podobnosti slov, ale môže sa použiť aj na klasifikáciu dokumentov. [17, 18, 29, 31]

Pri algoritme tf.idf skratka tf znamená Term Frequency alebo po slovensky frekvencia termínu a skratka idf znamená Inverse Document Frequency alebo po slovensky inverzná frekvencia dokumentu. Algoritmus tf.idf pracuje na základe relevancie slova alebo vety k dokumentom. Term Frequency počíta ako silno je spojené slovo s dokumentom. Čím je Term Frequency väčšia, tým je slovo viac spomenuté v dokumente. Inverse Document Frequency označuje ako rozhodujúce je dané slovo. Čím je Inverse Document Frequency väčšia, tým je slovo spomenuté v menej dokumentoch. Skóre pre slovo v tf.idf sa vypočíta ako vynásobenie Term Frequency a Inverse Document Frequency. Inverse Document Frequency nemá význam keď sa jedná iba o jedno slovo, pretože závisí iba od slova a nezávisí od dokumentu. Pri vete sa skóre vypočíta ako suma skóre pre jednotlivé slová a vtedy Inverse Document Frequency určuje dôvernosť skóre pre špecifické slovo. Slová, ktoré sa nachádzajú v málo dokumentoch, majú kvôli Inverse Document Frequency väčší vplyv na výslednú odpoveď. Term Frequency a Inverse Document Frequency nie sú pevne dané a môžu sa meniť. [17, 18, 29]

Algoritmus tf.idf nemusí pracovať iba po slovách (unigrams), ale môže pracovať s dvojicami slov (bigrams). Zhodné dvojice slov budú v dokumentoch vzácnejšie, a preto budú mať menšiu Term Frequency a väčšiu Inverse Document Frequency. Viacej ako dvojice alebo trojice slov sa málokedy používajú, pretože sú príliš špecifické. Väčšinou keď sa používajú dvojice slov, tak sa používajú aj jednotlivé slová. [18, 31]

Kapitola 3

Dátová sada

Pre strojové učenie je prvá a jedna z najdôležitejších častí získanie dát pre učenie neurónových sietí. Získané dáta pre učenie budú ovplyvňovať každú ďalšiu časť strojového učenia. [12]

Ako základ pre moju diplomovú prácu používam českú dátovú sadu SQAD_v3 (2019-10-14) vytvorenú Marekom Medveďom a Alešom Horákom z Masarykovej univerzity v Brne. SQAD je skratka pre Simple Question Answering Database čo sa dá preložiť ako dátová sada pre jednoduché problémy odpovedania na otázky. Dátová sada je vytvorená z českej Wikipédie, a preto je vhodná na použitie pri mojej diplomovej práci. [14]

3.1 Analýza českej dátovej sady SQAD

Česká dátová sada SQAD_v3 má páry otázok a odpovedí indexované od 000001 do 013476, ale nie všetky sú použiteľné. Skutočne som ich schopný použiť 13457. Nepoužiteľných je ich 19, pretože nemajú správne priradený článok z Wikipédie. Anglická dátová sada Stanford Question Answering Dataset má viac ako 100000 otázok s odpoveďami. [14, 16]

Pri jednoduchých otázkach, odpoveď vždy priamo vyplýva z textu a kontext potrebný na odpovedanie je jedna až tri vety. Pri analyzovaní 50 párov otázky a odpovede: 40 odpovedí sa dalo získať iba z jednej vety, na 5 odpovedí bol potrebný kontext z dvoch viet a na ďalších 5 odpovedí boli potrebné 3 vety na zostavenie kontextu odpovede.

Každý pár otázky a odpovede sa skladá z viacerých súborov:

1. Otázka na ktorú sa pýta. Otázka nie je v prirodzenej forme, ale je morfologicky anotovaná.

2. Odpoveď, ktorá je v najjednoduchšom možnom tvare. Nejedná sa o vetu, ale skladá sa iba z niekoľko relevantných slov. Odpoveď môže byť aj "áno"alebo "nie". Slová sú v gramatickom tvare, ktorý závisí na otázke a nemusia sa zhodovať tvarovo s textom. Odpoveď nie je v prirodzenej forme, ale je morfologicky anotovaná.

3. Text článku z Wikipédie, ktorý obsahuje odpoveď. Text nie je v prirodzenej forme, ale je morfologicky anotovaný. Text článku má značky, ktoré ho rozdelia na vety. Niektoré texty majú značky, ktoré ho rozdelia na paragrafy, ale tieto nie sú pri každom texte.

4. URL odkazujúci na článok z Wikipédie.

5. Metadáta, ktoré obsahujú dve značky: typ otázky a typ odpovede.

- Typy otázky. V zátvorke je počet otázok danej kategórie v dátovej sade.
  - ENTITY (2472): „Jak se jmenuje planeta, která je nejblíže ke Slunci?“
  - VERB_PHRASE (2263): „Je Bryan Adams fotograf?“
  - DATETIME (1979): „V jakém roce přistáli první lidé na Měsíci?“
  - PERSON (1764): „Byl Johannes Gensfleisch vynálezce technologie mechanického knihtisku?“
  - LOCATION (1665): „Kdo je sousedem Portugalska?“
  - ADJ_PHRASE (1508): „Jakou barvu má chryzopras?“
  - NUMERIC (981): „Kolik dní má červenec?“
  - CLAUSE (477): „Co je hlavním úkolem plodnice u hub?“
  - ABBREVIATION (333): „Jaký je chemický vzorec pro sůl?“
  - OTHER (15): „Co je těžší, zemské jádro nebo zemský plášť?“
- Typy odpovede. V zátvorke je počet odpovedí danej kategórie v dátovej sade.
  - YES_NO (2257): „ano“, „ne“
  - OTHER (2246): „Křižovatka Západu“, „pro jednoznačnou identifikaci knižních vydání“
  - DATETIME (1967): „16. dubna“, „13. března 1781“
  - PERSON (1777): „Williamem Lassellem“, „Dany Zátopkové“
  - ENTITY (1770): „Aluminium“, „Pedagogickou fakultu“
  - LOCATION (1655): „Syrůvka“, „Brno-venkov“
  - NUMERIC (998): „tři“, „32 bitů“
  - ABBREVIATION (321): „SPZ“, „ThDr.“
  - ORGANIZATION (283): „Australské státy a teritoria“, „Airbus“
  - DENOTATION (183): „asyriologie“, „turistika“

6. Veta obsahujúca odpoveď. Nejde o odpoveď v tvare vety, ale o práve jednu vetu z článku z Wikipédie, ktorá obsahuje odpoveď. Aj pri odpovediach "áno"a "nie", ktoré neobsahujú slová vybrané z článku z Wikipédie, veta je vždy text z článku. Vďaka tomu faktu sa dá overiť že systém naozaj vie o relevantných faktoch a odpoveď "áno"alebo "nie"nebola vybraná náhodne. Odpoveď nie je v prirodzenej forme, ale je morfologicky anotovaná.

7. Predpoveď odpovede. Je pri niektorých pároch otázky a odpovede rovnaká ako veta z článku obsahujúca odpoveď. Pri iných pároch otázky a odpovede nie je vôbec prítomná. Dátová sada nebola vyrábaná celá naraz, ale bola vyrábaná postupne. Štýl tvorby sa v čase menil a predpokladám že túto časť prestali považovať za potrebnú. Presnejšie predpoveď odpovede je pri takmer všetkých pároch otázky a odpovede od indexu 1 do indexu 8155 a v ďalších otázkach sa predpoveď odpovede nevyskytuje. Jedna z predchádzajúcich verzií českej dátovej sady SQAD_v2.0 z roku 2017 obsahovala páry otázok a odpovedí iba po index 8765. Predpoveď odpovede nie je v prirodzenej forme, ale je morfologicky anotovaná.

8. Kontext odpovede obsahuje niekoľko viet aj s vetou, ktorá obsahuje odpoveď. Kontext odpovede nie je pri každom páre otázky a odpovede. Presnejšie kontext odpovede je iba pri 378 pároch otázky a odpovede. Pretože kontext odpovede je pri príliš málo pároch otázky a odpovede, nie je použiteľný pri strojovom učení. Kontext nie je v prirodzenej forme, ale je morfologicky anotovaný.

9. Extrakcia odpovede obsahuje krátku verziu odpovede, ale na rozdiel od krátkej odpovede 2 je väčšinou v gramatickom tvare v akom sa vyskytuje v texte. Odpoveď môže byť aj "áno"alebo "nie". Pretože je odpoveď v rovnakom tvare ako aj v texte, tak sa dá oproti odpovedi 2 jednoduchšie použiť pri extrakcií odpovede z textu. Odpoveď nie je v prirodzenej forme, ale je morfologicky anotovaná.

Niektoré súbory páru otázky a odpovede sú v špeciálnom vertikálnom formáte (`.vert`). V použitom špeciálnom formáte nie je text v prirodzenej forme, ale je rozdelený na jednotlivé prvky v morfologickej anotácií. Každý prvok anotovaného textu je na samostatnom riadku. Prvok môže byť atomická jednotka vzatá z textu a rozšírená o ďalšie doplňujúce informácie alebo značka podobná značkám html a xml pridaná k rozdeleniu textu. Doplňujúca značka je vždy osamotená. Značky `<s>` a `</s>` vymedzujú jednu vetu. Ich použitie je v celku dobré. Niektoré vety sú rozdelené na skratke ukončenej bodkou, ako napríklad titul osoby, aj napriek tomu že daná bodka vetu neukončuje. Značky `<p>` a `</p>` rozdeľujú text na paragrafy, ale nie sú prítomné pri okolo 65% článkov. 63% článkov nie je rozdelených na paragrafy a zároveň má viacej ako 100 slov, a preto rozdelenie článku na menšie tématické celky použitím značiek pre paragrafy nie je možné. Značka `#` určuje pri odpovedi alternatívne odpovede. Alternatívna odpoveď môže byť vypísanie číslovky textom alebo číslom. Alternatívna odpoveď sa vyskytne aj keď v jednej vete dve slová označujú tú istú entitu a odpoveď môže byť ktorékoľvek z použitých slov. Napríklad vo vete: „Paříž (francouzsky Paris)...[1]“ slovo „Paříž“ a aj slovo „Paris“ označujú mesto Paríž a obi dva výrazy môžu byť použité ako odpoveď.

V dátovej sade sú dva typy krátkej odpovede. Odpoveď v súbore 02 je vo formáte otázky, a preto sa dá nájsť ako kus textu iba pri 8398 otázkach čo je 74.98%. Odpoveď v súbore 09 je bližšia formátu textu, napriek tomu sa dá nájsť ako kus textu iba pri 10965 otázkach čo je 97.9%. Ako krátku odpoveď budem ďalej považovať odpoveď zo súboru 09.

Atomická jednotka vzatá z textu môže vyť slovo alebo znak. Pri slove alebo znaku z textu sú vždy štyri položky alebo stĺpce oddelené tabulátorom:

- Prvá položka je slovo, číslo alebo znak presne v tvare v akom sa vyskytuje v texte.
- Druhá položka závisí od typu prvku, o ktorý sa jedná. Ak je prvok slovo z textu, tak druhý prvok je jeho základný tvar. Napríklad ak je prvok a prvá položka slovo "je", tak druhá položke je základ slova "je"slovo "být". Ak je prvok textu číslo, tak je v druhej položke kľúčové slovo "[number]". Ak je prvok znak z textu (napríklad bodka alebo zátvorka), tak druhá položka je ten istý znak.
- Tretia položka je POS značka. POS je skratka pre Part of speech. POS značka vyjadruje typ prvku. Bežné POS značky sú podstatné meno, sloveso, prídavné meno, prísudok, číslovka, ... POS značka má zakódované aj ďalšie informácie o solve ako jeho gramatické kategórie.
- Štvrtá položka je zopakovanie prvej alebo druhej položky. Podľa môjho pozorovania sa väčšinou jedná o druhú položku. Výnimky sú čísla, kde sa použije číslo čo je prvá položka. Ďalšou výnimkou sú mená, kde sa použije prvá položka a tým sa zachová meno človeka. Štvrtá položka vyzerá ako vylepšenie druhej položky.

[1] https://cs.wikipedia.org/wiki/Pa%C5%99%C3%AD%C5%BE

Články v dátovej sade

13457 párov otázok a odpovedí dátovej sady SQAD_v3 sa týka 7193 článkov z Wikipédie. V priemere je približne 1.87 otázok na jeden článok, ale v skutočnosti 4941 článkov (68.7%) má iba jednu otázku a 1022 článkov (14.21%) má iba dve otázky. Viac ako päť otázok má 456 článkov (6.34%). Najpopulárnejší článok z Wikipédie v dátovej sade je článok "Česko", ktorý má priradených 55 otázok. Počet otázok na článok je dôležitý a môže ovplyvniť výsledný systém. Ak je na článok iba jedna otázka, tak odpoveď bude pravdepodobne v prvom paragrafe, ktorý obsahuje všeobecné informácie o entite opisovanej v článku. Z toho vyplýva že nebude veľmi problematické nájsť odpoveď v článku, ale hlavnou úlohou bude vybrať článok z Wikipédie, ktorý obsahuje odpoveď. Pre otázky na všeobecnejšie články nebude problém nájsť ktorý článok obsahuje odpoveď, ale problematickejšie bude nájsť odpoveď v článku. Ide o dva rôzne problémy, ktoré kompetentný systém na zodpovedanie otázok nad Wikipédiou bude musieť vyriešiť. Väčší počet rôznych článkov zároveň spôsobí že systém učený na dátovej sade sa stretne s rôznym textom, a tým by sa presnosť výsledkov mala zväčšiť. [12]

Učenie stroja na vyberanie článku bude problematické s českou dátovou sadou SQAD_v3. Pri učení musí byť dátová sada rozdelená na dve časti. Prvá časť bude slúžiť na trénovanie stroja a druhá časť bude slúžiť na validáciu naučeného stroja. Validácia naučeného stroja slúži na overenie všeobecnosti naučeného stroja a zamedzuje pretrénovaniu na trénovacie dáta. Bez validácie by sa stroj mohol naučiť príliš konkrétne na dátovú sadu, a preto nebude použiteľný na otázky mimo dátovej sady. 68.7% článkov má iba jednu otázku, ktorá bude v jednej z častí a to nie je vhodné pre strojové učenie. Keď sa bude učiť stroj na hľadanie článku, tak bude potrebné zložitejšie učenie. [12]

Zložitosť dátovej sady a bias strojového učenia

Zložitosť dátovej sady je odolnosť dátovej sady oproti triviálnym riešeniam. Cieľom dátovej sady je naučiť systém rozpoznávať vlastnosti medzi otázkou a textom, ktoré môžu byť použité na extrahovanie odpovede. Vlastnosti dátovej sady podporujúce bias strojového učenia zvýšia pravdepodobnosť rozpoznania odpovede dátovej sady, ale nebudú fungovať pri použití naučeného systému v praxi. Napríklad jeden bias strojového učenia je štatistický bias. Najjednoduchší štatistický bias môže byť použitie štatisticky najpravdepodobnejšej odpovede pre každú otázku. Napríklad ak sa očakáva odpoveď "áno"alebo "nie"a systém sa naučí na každú otázku odpovedať "áno", tak môže dosiahnuť úspech až 60+%. Nie každá dátová sada je k takýmto prípadom vyvážená. Úspešnosť, ktorá sa dá dosiahnuť inteligentným odhadovaním, môžeme nazvať minimálnou hranicou. Čím je menšia minimálna hranica, tým sa dá dátová sada považovať za zložitejšiu. Aj jednoduchá dátová sada sa dá použiť na trénovanie systémov, ale je potrebné upraviť postup učenia na odstránenie biasu pre strojové učenie. Zložitejšie dátové sady majú výhodu jednoduchšieho učenia. [12, 7]

Pri testovaní sa dá bias postrážiť jednoducho. Každý naučený systém s presnosťou pod minimálnou hranicou alebo na minimálnej hranice sa dá považovať za neadekvátny. Na hodnotenie naučených systémov sa bežne používa viacero metrík. Návrh a použiteľnosť špecifickej metriky závisí od výstupného formátu naučeného systému a zložitosti dátovej sady. Najjednoduchšia metrika je presnosť. Presnosť sa dá použiť pri každom formáte výstupu, ale zložitosť dátovej sady dokáže presnosť nepríjemne ovplyvniť. Metrika $f1$ je zložitejšia a trošku odolnejšia ako presnosť, ale dá sa použiť iba pri binárnej klasifikácií. [12]

Pri odpovedaní na otázku je potrebné v niektorých prípadoch vybrať kus textu. Pri takomto prípade sa dá použiť metrika $f1$ na porovnanie správnej a predikovanej odpovede.

Ako p sa zoberie prekrytie odpovedí podelené dĺžkou predikovanej odpovede. Ako r sa zoberie prekrytie odpovedí podelené dĺžkou správnej odpovede. Potom sa $f1$ vypočíta ako $f1 = 2\frac{pr}{p+r}$. Metrika $f1$ je nežnejšia na drobné chyby pri vybraní kusu textu než presnosť. [4]

Pri učení sa bias strojového učenia stráži zložitejšie. Prístupy na zmiernenie biasu pri učení sa dajú rozdeliť do dvoch kategórií:

- Vyváženie dátovej sady zmenšením počtu prvkov dátovej sady. Ak dátová sada nie je štatisticky vyvážená, tak sa dá vyvážiť vypustením niektorých párov otázky a odpovede. Napríklad ak chceme vyvážiť dátovú sadu oproti odpovediam "áno"alebo "nie", tak sa vypustia najčastejšie odpovede dokým nebude pomer odpovedí vyvážený. Zmenšenie dátovej sady zmenší počet situácií, s ktorými sa učený systém stretne, a tým sa môže zmenšiť potenciálny výkon učeného systému. Problém bude vybrať otázky a odpovede, ktoré budu vypustené z učenia systému. Cieľom vypustenia je zlepšenie presnosti učeného systému. Pretože je systém učený vo viacerých iteráciách, vypustené prvky sa môžu pri každej iterácií meniť. Teoreticky sa učený systém stretne s každou otázkou v dátovej sade, ale v každom cykle trénovania bude pomer odpovedí vyvážený. Tento prístup vytvorí šum, ale učenie niektorých konštrukcií neurónovej siete, ako napríklad LSTM, má šum rado. [12]

- Diskriminácia medzi prvkami dátovej sady. Ak nie je vhodné pri učení zmenšovať dátovú sadu a zároveň je požadované vyvážené učenie, tak sa dá diskriminovať medzi jednotlivými prvkami dátovej sady a nebrať ich všetky s rovnakou váhou. V článku o zmierňovaní biasu [11] je predložený systém, ktorý by mal zmierniť bias strojového učenia pridaním druhého systému pri učení. Na učenie sa používa spätná propagácia a učené sú neurónové siete. Prvá neurónová sieť je normálna sieť, ktorú je potrebné niečo naučiť. Druhá neurónová sieť sa učí iba bias strojového učenia. Druhá neurónová sieť má prázdny vstup alebo má iba čiastočný vstup. Napríklad pri učení odpovedania na otázku dostane iba text bez otázky. Prvá aj druhá neurónová sieť sa učia zároveň. Môžu zdieľať vektorovú reprezentáciu vstupu, ale druhá neurónová sieť nemôže pri spätnej propagácií zdielaný prvok meniť. Pred spätnou propagáciou sa od výstupu prvej neurónovej siete odčíta výstup druhej neurónovej siete. Ak aj druhá neurónová sieť dokáže správne predpokladať odpoveď, tak bude odčítaná väčšia hodnota a spätnej propagácií sa vloží menšia hodnota. Keď sa spätnej propagácií vloží menšia stratová hodnota pre určitý prvok dátovej sady, tak ten prvok dátovej sady bude mať menší vplyv na učenie prvej neurónovej siete. Problém použitia dvoch neurónových sietí je časová a pamäťová náročnosť. Pri učení neurónovej siete na grafickej karte je pamäťová náročnosť dôležitá. Potenciálne lepšie by bolo ohodnotiť prvky dátovej sady neurónovou sieťou na detekciu biasu strojového učenia pred učením normálnej neurónovej siete a to hodnotenie by menilo vplyv prvku na učenie. [11]

Zložitosť českej dátovej sady SQAD

V dátovej sade sa dá riešiť viacero pod problémov, ktoré budú prispievať odpovedaniu na otázku. Pre pod problémy budem skúmať zložitosť pod problému a potenciálny bias strojového učenia.

- Vyhľadávanie článku s odpoveďou. 4941 článkov (68.7%) má iba jednu otázku a 1022 článkov (14.21%) má iba dve otázky. Najčastejší článok má iba 55 otázok asociovaných s ním. Vyhľadávanie článku s odpoveďou je odolné voči biasu strojového učenia a je zložité pre strojové učenia.

- Vyhľadávač (retriever) vyberie vetu z textu, ktorá by mala obsahovať odpoveď. Text rozdeľujem na vety podľa značenia textu už prítomného v dátovej sade. Pozícia odpovede nebola nájdená pre 32 párov otázky a odpovede čo je 0.24%. Odpoveď je prvá veta pre 6411 otázok čo je 47.64%. Odpoveď je druhá veta pre 1212 otázok čo je 9.01%. Odpoveď je tretia veta pre 757 otázok čo je 5.63%. Väčšina odpovedí ja na začiatku Wikipédia článku. Prvý paragraf článku z Wikipédie obsahuje základné údaje o entite opísanej v článku, a preto sa v prvom paragrafe dá nájsť veľa odpovedí. Kvôli zložitosti dátovej sady, správne naučený vyhľadávač by mal mať presnosť väčšiu ako 50%. Bias nie je dosť veľký na znemožnenie správneho učenia, ale implementácia postupov na limitovanie biasu pri učení by mohla zlepšiť presnosť vyhľadávača.

- Odpovedanie na otázku "áno"alebo "nie". Odpovedí v kategórií "YES_NO"je 2257. Odpovedí, ktoré sa dajú považovať za "áno", sa mi podarilo nájsť 1510 čo je 66.9%. Odpovedí, ktoré sa dajú považovať za "nie", sa mi podarilo nájsť 738 čo je 32.7%. Nepodarilo sa mi zaradiť 7 odpovedí, ktoré majú typ "YES_NO", do kategórií "áno"alebo "nie". Medzi 66.9% a 32.7% je dosť veľkí rozdiel na to aby bol potrebný systém na limitovanie biasu strojového učenia. Správne naučený systém na odpovedanie by mal mať presnosť aspoň 68%.

- Na odpovedanie nektorých otázok je potrebné prezentovať dáta z článku. V takom prípade je potrebné extrahovanie krátkej odpovede z vybranej vety. Takýchto príkladov je 11200. Môže to byť číslo, dátum, entita alebo aj viacero rôznych prvkov z textu. Za krátku odpoveď sa berie spojitý kus textu. Problém je nájsť odpoveď vo vete a získať potrebné identifikátory odpovede. Jedna z najpopulárnejších a zároveň aj dobre testovaná metóda je predpokladanie začiatku a konca odpovede [12]. Najčastejší začiatok a koniec odpovede je prvé slovo. Prvé slovo je začiatok pre 2259 odpovedí čo je 20.6%. Prvé slovo je koniec odpovede pre 1240 odpovedí čo je 11.31%. Predikcia začiatka a konca odpovede je dostatočne odolná voči biasu strojového učenia, a preto učený systém sa naozaj musí niečo poriadne naučiť. Predpokladať sa dá aj začiatok odpovede a dĺžka odpovede alebo koniec odpovede a dĺžka odpovede, ale predpokladám že dĺžka odpovede je náchylnejšia na bias strojového učenia. Najčastejšia dĺžka odpovede je jedno slovo. Jedno slovných odpovedí je 5046 čo je 46.02%. Dĺžka odpovede je oveľa menej odolnejšia voči biasu strojového učenia než začiatok a koniec odpovede, a preto predpokladám že sa až tak veľmi nepoužíva. Predpokladať sa dá aj každé jedno slovo odpovede, ale takýto postup nemá lepšiu presnosť ako predpokladanie začiatku a konca odpovede [12].

3.2 Doplnenie dátovej sady

Systémy na odpovedanie otázok sa zvyčajne testuje na kvízoch a ďalších vedomostných súťažiach pre ľudí [13]. Dátová sada sa dá doplniť o otázky a odpovede z rôznych kvízoch a súťaží [5]. Pri dátach zo súťaží väčšinou nie je prítomný text, v ktorom je odpoveď. Chýbanie explicitného textu je problém pri trénovaní. Otázky a odpovede získané z kvízov sa budú pravdepodobne používané iba pri testovaní systémov na odpovedanie otázok. Otázky a odpovede sa dajú extrahovať, ale na to musí kvíz odpovede poskytovať. Testovať sa dá aj na kvízoch, ktoré odpovede neposkytujú, ale testovanie musí byť online. Ak ide o kvíz na internete, tak na jeho prejdenie je vhodná knižnica `selenium` pre jazyk Python.

Ďalší používaný spôsob na vytvorenie veľkej dátovej sady je použitie strojového učenia na už izolované dáta. Postupne sa vytvárajú vedomostné databáze, ktoré sú optimalizo-

vané na čítanie strojom. Narozdiel od vedomostných databáz pre ľudí, ktoré sú napísané v prirodzenom jazyku, databáze pre stroje sú štruktúrované a normalizované. Príkladom vedomostnej databáze pre ľudí je Wikipédia a príkladom vedomostnej databáze pre stroje je Wikidata. Na vytvorenie dátovej sady musí učený systém vytvárať prirodzený text z dát. Učený systém dostane znalosť z vedomostnej databáze pre stroje a vyprodukuje otázku a odpoveď, ktoré sú v prirodzenom jazyku. Potrebné bude aj získať odkaz do vedomostnej databáze pre ľudí, v ktorej sa dá odpoveď nájsť. Na vytvorenie otázky a odpovede sú učené systémy inšpirované štatistickým strojovým prekladom. Kvalita otázok a odpovedí vytvorených stojom musí vyť overená človekom. [20]

Kapitola 4

Návrh systému

Systém na odpovedanie otázok som sa rozhodol riešiť v programovacom jazyku Python s použitím knižnice PyTorch na základe kódu vytvoreného Ing. Martinom Fajčíkom a poskytnutého študentom Vysoké učení technické v Brně Fakulty informačních technologií. Základový kód popisuje jednoduchú neurónovú sieť na odpovedanie otázok používajúcu LSTM, konvertovanie slov alebo znakov na vektory, učenie neurónovej siete, validácia a metriky popisujúce výsledky neurónovej siete a aj testovanie a analýzu neurónovej siete. Ako dátová sada sa používa anglická dátová sada SQAD. Základová neurónová sieť má ako vstup otázku a paragraf článku, ktorý obsahuje odpoveď. Ako výstup vypočíta začiatok a koniec odpovede. [4]

Z formátu odpovedí v dátovej sade rozdeľujem odpovede na odpovede "áno/nie"a odpovede, ktoré používajú slová z textu. Jeden z problémov základového systému je neschopnosť spracovať odpovede "áno/nie". Rozlišovanie medzi typmi odpovede a samotné odpovedanie "áno/nie"by sa malo dať skombinovať do jednej siete a jedného učenia, ale považujem za jednoduchšie a praktickejšie rozdeliť jednotlivé pod problémy do rozdielnych neurónových sietí a učiť ich separovane. Separované učenie umožní sústrediť učenie časti na potrebnú časť a zároveň zmenší pamäťovú náročnosť. Pri odpovedaní na otázku z použitím Wikipédie je potrebné použiť celý článok a to je pamäťovo náročné. Odpovedanie na otázku rozdelím na nasledovné pod-problémy. Rozdelenie odpovedí na "áno/nie"alebo vybranie odpovede z textu je špeciálny problém, na ktorý je vhodné mať zvlášť modul. Ďalším problémom základového systému je veľmi malá presnosť pri vyberaní odpovede z textu. Predpovedanie začiatku a konca odpovede je problematické pri dlhom texte, a preto je vhodné zmenšiť kontext odpovede na to špecializovaným modulom. Vybranie kontextu iba jedným parametrom sa dá docieliť vybraním jednej vety z článku. Aj keď sa vyberie iba jedna veta, kontext môže obsahovať aj niekoľko viet okolo vybranej vety. Samotné odpovedanie "áno/nie"alebo časť textu sa dá riešiť zvlášť, ale spojené riešenie môže potenciálne zlepšiť presnosť zdieľaním častí na pochopenie textu. Pridanie rozlišovania odpovede do modulu pre získanie odpovede nemusí byť najlepšie, pretože sa jedná o dva zásadne odlišné problémy. [4]

Pri kombinácií všetkých systém sa najskôr na základe otázky vyberie článok, v ktorom sa predpokladá existencia odpovede. Z vybraného článku sa na základe otázky izoluje kontext, ktorý by mal mať všetky potrebné informácie na odvodenie odpovede. Z kontextu a otázky sa extrahuje konečná odpoveď. Formát konečnej odpovede a modul, ktorý ju má na starosti získať, je vybraný na základe predikcie typu odpovede. Na predikciu typu odpovede sa používa iba otázka.

Pretože prevedenie textu na vektory je jedna z častí, ktorá sa dá zdielať medzi problémami spracovania prirodzeného jazyka, je bežné použiť už natrénovaný model namiesto

trénovania vlastného modelu [12]. Používal som dva české natrénované modely. Najskôr som používal model 37[1], ktorý bol vygenerovaný algoritmom Word2Vec Continuous Skipgram z korpusu Czech CoNLL17 corpus [10]. Jeho veľkosť výsledných vektorov je 100 dimenzií. Prešiel som na model FastText[2], ktorý bol vytvorený Facebook's AI Research lab. Jeho veľkosť výsledných vektorov je 300 dimenzií a má lepšie alebo rovnaké výsledky ako model 37 s horšou pamäťovou náročnosťou. Pri učení sa konvertovanie slov na vektory upravuje. Získaný model sa používa iba ako začiatok. Pri každom učení vznikne unikátny model konvertovania slov na vektory. Pretože učím modely zvlášť, vznikne pre každý model unikátne mapovanie slov na vektory.

Skratky pri maticiach

- b – veľkosť dávky (batch)
- q – dĺžka otázky
- d – počet viet v dokumente
- s – dĺžka vety
- k – dĺžka kontextu
- e – počet dimenzií výstupu konverzie slova na vektor (embedding)
- $lstm$ – počet dimenzií vnútorného stavu LSTM
- $2lstm$ – počet dimenzií pri obojstrannom LSTM
- $iter$ – pri iteratívnych modeloch určuje počet iterácií.
- a – počet spracovávaných článkov

4.1 Rozdelenie otázok podľa typu odpovede

V dátovej sade sú použité dva základné typy odpovedí: odpovede "áno/nie"a odpoveď ako kus textu. Od rozdelenia otázok bude závisieť modul, od ktorého sa bude požadovať zistenie odpovede. Odpoveď ako kus textu sa dá ešte ďalej deliť, ale formát odpovede je vždy stavaný zo slov textu a nie je potrebné ďalej deliť modul na extrahovanie odpovede z textu. Rozdelenie by malo závisieť iba na otázke a text by na typ odpovede nemal mať vplyv.

Neurónová sieť sa učí po dávkach. Neurónová sieť berie ako vstup iba otázku rozdelenú na slová. Výstup neurónovej siete je zaradenie otázky do jednej z dvoch kategórií. Vzniknú dve čísla na otázku. Ak je prvé číslo väčšie, tak má otázka odpoveď kus z textu. Ak je druhé číslo väčšie, tak má otázka odpoveď "áno/nie".

Model v1

1 Otázka sa najskôr skonvertuje na vektory.

- Veľkosť matice otázky: $[b, q, e]$

[1] http://vectors.nlpl.eu/repository/
[2] https://fasttext.cc/

2 Na maticu sa použije LSTM. LSTM pôjde obojstranne. LSTM nájde príznaky v otázke a vytvorí novú maticu.

- Veľkosť matice otázky: $[b, q, 2lstm]$

2 Dimenzia dĺžky otázky sa odstráni použitím získania maxima cez dĺžku otázky. Maximum cez dĺžku otázky spojí najextrémnejšie vlastnosti všetkých vektorov.

- Veľkosť matice: $[b, 2lstm]$

3 Pretože sa otázka kategorizuje do dvoch kategórií, je potrebné zmenšiť výstupný vektor na dve dimenzie. Vektor sa zmenší použitím lineárneho filtra. Lineárny filter sa učí.

- Veľkosť výstupnej matice: $[b, 2]$

Model v2

1 Otázka sa najskôr skonvertuje na vektory.

- Veľkosť matice otázky: $[b, q, e]$

2 Na maticu sa použije LSTM. LSTM pôjde obojstranne. Získa sa posledný stav LSTM. LSTM nájde príznaky v otázke a vytvorí novú maticu.

- Veľkosť matice otázky: $[2, b, lstm]$

2 Dva vnútorné stavy LSTM sa konkatenujú. Jeden stav je pre prechod spredu a druhý stav je pre prechod zozadu.

- Veľkosť matice: $[b, 2lstm]$

3 Pretože sa otázka kategorizuje do dvoch kategórií, je potrebné zmenšiť výstupný vektor na dve dimenzie. Vektor sa zmenší použitím lineárneho filtra. Lineárny filter sa učí.

- Veľkosť výstupnej matice: $[b, 2]$

4.2 Vyhľadanie vety s odpoveďou

Pri extrahovaní odpovede je potrebné predikovať dve čísla: začiatok a koniec odpovede. Predikcia oboch čísel vyplýva z rovnakého základu, ale výsledky sú od seba nezávislé. Predikovanie jedného čísla môže zlepšiť presnosť systému, a preto sa najskôr vyberie veta z textu a potom vo vybranej vete alebo rozšírenom kontexte sa bude hľadať odpoveď. Vety sú izolované a obmedzené časti textu, a preto stačí jedno číslo na vybranie vety. Kontext pre odpoveď môže presahovať vetu, ale text odpovede vetu nepresahuje. Vyberať by sa dali aj paragrafy, ale v použitej dátovej sade rozdelenie na paragrafy nie je dostatočne kvalitné. Dátová sada má rozdelenie na vety dostatočne presné na použitie. Najväčší rozdiel medzi vyhľadaním vety a bežným spracovaním prirodzeného textu je pridanie dimenzie na vstupe. [28]

Neurónová sieť sa učí po dávkach. Na vstupe je otázka rozdelená na slová a text rozdelený na vety a slová. Výstup neurónovej siete je predikcia vety, v ktorej je odpoveď. Výstup má veľkosť počtu viet. Index s najväčšou hodnotou označuje index vety odpovede.

Model r1

1 Otázka a text sa konvertujú na vektory.

- Veľkosť matice otázky: $[b, q, e]$
- Veľkosť matice textu: $[b, d, s, e]$

2 Každá veta sa zhrnie na vektor. LSTM berie iba troj dimenzionálne matice, a preto je potrebné maticu rozdeliť. Matica sa rozdelí podľa dávok.

- Veľkosť čiastkovej matice: $[d, s, e]$

2 Na čiastkovú maticu sa aplikuje LSTM. Pretože na pozícií dimenzie dávok bude počet viet, jednotlivé vety sa nebudú ovplyvňovať.

- Veľkosť čiastkovej matice: $[d, s, 2lstm]$

2 Dimenzia dĺžky vety sa odstráni použitím získania maxima cez dĺžku vety. Maximum cez dĺžku vety spojí najextrémnejšie vlastnosti všetkých vektorov.

- Veľkosť čiastkovej matice: $[d, 2lstm]$
- Veľkosť spojenej matice textu: $[b, d, 2lstm]$

3 Matica otázky sa zmenší na vektor. Najskôr na maticu otázky sa aplikuje LSTM.

- Veľkosť matice otázky: $[b, q, 2lstm]$

3 Dimenzia dĺžky otázky sa odstráni použitím získania maxima cez dĺžku otázky. Maximum cez dĺžku otázky spojí najextrémnejšie vlastnosti všetkých vektorov.

- Veľkosť matice otázky: $[b, 2lstm]$

4 Matica textu sa prevedie cez lineárny filter bez zmeny veľkosti. Lineárny filter sa učí. K matici otázky sa pridá prázdna dimenzia.

- Veľkosť matice textu: $[b, d, 2lstm]$
- Veľkosť matice otázky: $[b, 2lstm, 1]$

5 Matica textu a matica otázky sa vynásobia po dávkach.

- Veľkosť produktu: $[b, d, 1]$

5 Po odstránení prázdnej dimenzie vznikne výstupná matica.

- Veľkosť výstupnej matice: $[b, d]$

Model r2

1 Otázka a text sa konvertujú na vektory.

- Veľkosť matice otázky: $[b, q, e]$
- Veľkosť matice textu: $[b, d, s, e]$

2 LSTM berie iba troj dimenzionálne matice, a preto je potrebné maticu rozdeliť. Matica sa rozdelí podľa dávok.

- Veľkosť čiastkovej matice: $[d, s, e]$

2 Pred každú vetu sa konkatenuje otázka.

- Veľkosť čiastkovej matice: $[d, q + s, e]$

2 Na čiastkovú maticu sa aplikuje LSTM. Pretože na pozícií dimenzie dávok bude počet viet, jednotlivé vety sa nebudú ovplyvňovať.

- Veľkosť čiastkovej matice: $[d, q + s, 2lstm]$

2 Dimenzia dĺžky otázky a vety sa odstráni použitím získania maxima cez dĺžku otázky a vety. Maximum cez otázky a vety spojí najextrémnejšie vlastnosti všetkých vektorov. Týmto by pre každú vetu mal vzniknúť vektor, ktorý má vlastnosti o vete a jej relevancie k otázke.

- Veľkosť čiastkovej matice: $[d, 2lstm]$
- Veľkosť spojenej matice: $[b, d, 2lstm]$

3 Na spojenú maticu sa aplikuje ďalšie LSTM, ktoré má inú dĺžku vstupu od predchádzajúce LSTM a aj je zvlášť učené. Aplikovaním LSTM sa budú zdielať informácie medzi jednotlivými vektormi patriacimi vetám, a tým aj zdielať informácie medzi vetami.

- Veľkosť spojenej matice: $[b, d, 2lstm]$

4 Dimenzia výstupu LSTM sa zruší použitím lineárneho filtra. Lineárny filter sa naučí vypočítať z vektora jedno číslo.

- Veľkosť spojenej matice: $[b, d, 1]$

5 Po odstránení prázdnej dimenzie vynikne výstupná matica.

- Veľkosť výstupnej matice: $[b, d]$

Model r3

1 Otázka a text sa konvertujú na vektory.

- Veľkosť matice otázky: $[b, q, e]$
- Veľkosť matice textu: $[b, d, s, e]$

2 Otázka sa zmenší na vektor použitím prvého LSTM a získania maxima cez dĺžku otázky.

- Veľkosť matice otázky: $[b, 2lstm]$

3 LSTM berie iba troj dimenzionálne matice, a preto je potrebné maticu rozdeliť. Matica sa rozdelí podľa dávok.

- Veľkosť čiastkovej matice: $[d, s, e]$

3 Pred každý vektor slova sa konkatenuje vektor otázky. Konkatenovaním vektora pred každý vektor slova sa snažím imitovať spojenie otázky a textu použitého v Dynamic memory network2.3.

- Veľkosť čiastkovej matice: $[d, s, e + 2lstm]$

3 Na čiastkovú maticu sa aplikuje druhé LSTM, ktoré má inú dĺžku vstupu od predchádzajúceho LSTM a aj je zvlášť učené. Pretože na pozícií dimenzie dávok bude počet viet, jednotlivé vety sa nebudú ovplyvňovať.

- Veľkosť čiastkovej matice: $[d, s, 2lstm]$

3 Dimenzia dĺžky vety sa odstráni použitím získania maxima cez dĺžku otázky a vety. Maximum cez otázky a vety spojí najextrémnejšie vlastnosti všetkých vektorov. Týmto by pre každú vetu mal vzniknúť vektor, ktorý má vlastnosti o vete a jej relevancie k otázke.

- Veľkosť čiastkovej matice: $[d, 2lstm]$

- Veľkosť spojenej matice: $[b, d, 2lstm]$

4 Na spojenú maticu sa aplikuje tretie LSTM, ktoré má inú dĺžku vstupu od predchádzajúcich LSTM a aj je zvlášť učené. Aplikovaním LSTM sa budú zdielať informácie medzi jednotlivými vektormi patriacimi vetám, a tým aj zdielať informácie medzi vetami.

- Veľkosť spojenej matice: $[b, d, 2lstm]$

5 Dimenzia výstupu LSTM sa zruší použitím lineárneho filtra. Lineárny filter sa naučí vypočítať z vektora jedno číslo.

- Veľkosť spojenej matice: $[b, d, 1]$

6 Po odstránení prázdnej dimenzie vynikne výstupná matica.

- Veľkosť výstupnej matice: $[b, d]$

Model r5

1 Otázka a text sa konvertujú na vektory.

- Veľkosť matice otázky: $[b, q, e]$

- Veľkosť matice textu: $[b, d, s, e]$

2 LSTM berie iba troj dimenzionálne matice, a preto je potrebné maticu rozdeliť. Matica sa rozdelí podľa dávok.

- Veľkosť čiastkovej matice: $[d, s, e]$

2 Na čiastkovú maticu sa aplikuje LSTM. Pretože na pozícií dimenzie dávok bude počet viet, jednotlivé vety sa nebudú ovplyvňovať.

- Veľkosť čiastkovej matice: $[d, s, 2lstm]$

2 Dimenzia dĺžky vety sa odstráni použitím získania maxima cez dĺžku otázky a vety. Maximum cez otázky a vety spojí najextrémnejšie vlastnosti všetkých vektorov.

- Veľkosť čiastkovej matice: $[d, 2lstm]$
- Veľkosť spojenej matice: $[b, d, 2lstm]$

3 Otázka sa zmenší na vektor použitím LSTM a získania maxima cez dĺžku otázky.

- Veľkosť matice otázky: $[b, 2lstm]$

3 Prvý prvok pamäte sa inicializuje na reprezentáciu otázky.

- Veľkosť vnútornej pamäte: $[b, 1, 2lstm]$

4 Vnútorná pamäť sa konkatenuje pred redukovaný vektor dokumentu.

- Veľkosť spojenej matice: $[b, 1 + d, 2lstm]$

4 Na spojenú maticu sa aplikuje ďalšie LSTM, ktoré má inú dĺžku vstupu od predchádzajúce LSTM a aj je zvlášť učené. Aplikovaním LSTM sa získa závislosť medzi vnútornou pamäťou a dokumentom. Z posledného stavu LSTM sa získa vektor reprezentujúci získané vlastnosti.

- Veľkosť spojenej matice: $[b, 1 + d, 2lstm]$

4 Vektor reprezentujúci získané vlastnosti sa konkatenuje za vnútornú pamäť.

- Veľkosť vnútornej pamäte: $[b, 2, 2lstm]$

4 Krok 4 sa opakuje predom stanovený počet iterácií. V každej iterácií sa pamäť rozšíri o nový vektor reprezentujúci získané vlastnosti.

- Veľkosť poslednej spojenej matice: $[b, iter + d, 2lstm]$

5 Časť spojenej matice pre pamäť sa odstráni.

- Veľkosť matice: $[b, d, 2lstm]$

6 Dimenzia výstupu LSTM sa zruší použitím lineárneho filtra. Lineárny filter sa naučí vypočítať z vektora jedno číslo.

- Veľkosť spojenej matice: $[b, d, 1]$

6 Po odstránení prázdnej dimenzie vynikne výstupná matica.

- Veľkosť výstupnej matice: $[b, d]$

Model r6

Model r6 má rovnakú štruktúru ako model r5 až na jeden rozdiel. Pri modely r6 sa pamäť pri iteráciách nezväčšuje. Pri modely r5 sa pri každej iterácií pridal nový prvok do pamäte. Pri modely r6 je pamäť nahradená novým prvkom.

Pri jednej iterácií sú modely r5 a r6 rovnaké.

Model r7

1 Otázka a text sa konvertujú na vektory.

- Veľkosť matice otázky: $[b, q, e]$
- Veľkosť matice textu: $[b, d, s, e]$

2 LSTM berie iba troj dimenzionálne matice, a preto je potrebné maticu rozdeliť. Matica sa rozdelí podľa dávok.

- Veľkosť čiastkovej matice: $[d, s, e]$

2 Na čiastkovú maticu sa aplikuje LSTM. Pretože na pozícií dimenzie dávok bude počet viet, jednotlivé vety sa nebudú ovplyvňovať.

- Veľkosť čiastkovej matice: $[d, s, 2lstm]$

2 Dimenzia dĺžky vety sa odstráni použitím získania maxima cez dĺžku otázky a vety. Maximum cez otázky a vety spojí najextrémnejšie vlastnosti všetkých vektorov.

- Veľkosť čiastkovej matice: $[d, 2lstm]$
- Veľkosť spojenej matice: $[b, d, 2lstm]$

3 Otázka sa zmenší na vektor použitím LSTM a získania maxima cez dĺžku otázky.

- Veľkosť matice otázky: $[b, 2lstm]$

3 Prvý prvok pamäte sa inicializuje na maticu otázky.

- Veľkosť vnútornej pamäte: $[b, 2lstm]$

4 Vnútorná pamäť sa konkatenuje za každý vektor vety.

- Veľkosť spojenej matice: $[b, d, 4lstm]$

4 Na spojenú maticu sa aplikuje ďalšie LSTM, ktoré má inú dĺžku vstupu od predchádzajúceho LSTM a aj je zvlášť učené. Aplikovaním LSTM sa získa závislosť medzi vnútornou pamäťou a dokumentom. Z posledného stavu LSTM sa získa vektor reprezentujúci získané vlastnosti.

- Veľkosť spojenej matice: $[b, d, 2lstm]$

4 Vnútorná pamäť sa nahradí posledným stavom LSTM.

- Veľkosť vnútornej pamäte: $[b, 2lstm]$

4 Krok 4 sa opakuje predom stanovený počet iterácií. V každej iterácií sa pamäť rozšíri o nový vektor reprezentujúci získané vlastnosti.

- Veľkosť poslednej spojenej matice: $[b, d, 2lstm]$

5 Dimenzia výstupu LSTM sa zruší použitím lineárneho filtra. Lineárny filter sa naučí vypočítať z vektora jedno číslo.

- Veľkosť spojenej matice: $[b, d, 1]$

5 Po odstránení prázdnej dimenzie vynikne výstupná matica.

- Veľkosť výstupnej matice: $[b, d]$

Model r7b

Model r7b má rovnakú štruktúru ako model r7 až na jeden rozdiel. Na rozdiel od modelu r7 má každá iterácia samostatné LSTM. Každá iterácia by sa mohla špecializovať na jednu vlastnosť. Nevýhodou je náchylnosť na syndróm miznúceho gradientu. Celá váha predpovede sa posunie na poslednú iteráciu a skoršie iterácie nebudú učené.

Pri jednej iterácií sú modely r7 a r7b rovnaké.

4.3 Odpovedanie áno alebo nie

Odpovedanie na otázku "áno"alebo "nie"sa dá zjednodušiť na binárnu klasifikáciu na základe otázky a textu. Odpovedanie "áno/nie"je riešené na zmenšenom kontexte a nie na celom článku.

Neurónová sieť sa učí po dávkach. Ako vstup zoberie otázku rozdelenú na slová a kontext rozdelený na slová. Výstup neurónovej siete je zaradenie otázky do jednej z dvoch kategórií. Vzniknú dve čísla na otázku a vetu. Ak je prvé číslo väčšie, tak je odpoveď na otázku nie. Ak je druhé číslo väčšie, tak je odpoveď na otázku áno.

Model yn1

1 Otázka a text sa konvertujú na vektory.

- Veľkosť matice otázky: $[b, q, e]$
- Veľkosť matice textu: $[b, k, e]$

2 Otázka sa konkatenuje pred kontext.

- Veľkosť matice: $[b, q + k, e]$

3 Na maticu sa použije LSTM. LSTM nájde príznaky v kombinácií otázky a textu.

- Veľkosť matice: $[b, q + k, 2lstm]$

4 Dimenzia dĺžky otázky a vety sa odstráni použitím získania maxima cez dĺžku otázky a vety. Maximum cez vektory otázky a vety spojí najextrémnejšie vlastnosti všetkých vektorov.

- Veľkosť matice: $[b, 2lstm]$

5 Vektor sa zmenší na potrebnú veľkosť použitím lineárneho filtra. Lineárny filter sa učí.

- Výstupná matica: $[b, 2]$

Model yn2

1 Otázka a text sa konvertujú na vektory.

- Veľkosť matice otázky: $[b, q, e]$
- Veľkosť matice textu: $[b, k, e]$

2 Na otázku a vetu sa aplikuje LSTM zvlášť.

- Veľkosť matice otázky: $[b, q, 2lstm]$
- Veľkosť matice textu: $[b, k, 2lstm]$

3 Dimenzia dĺžky otázky a vety sa odstráni použitím získania maxima cez dĺžku otázky a vety. Maximum cez otázky spojí najextrémnejšie vlastnosti vektorov otázky. Maximum cez vety spojí najextrémnejšie vlastnosti vektorov vety.

- Veľkosť matice otázky: $[b, 2lstm]$
- Veľkosť matice textu: $[b, 2lstm]$

4 Matice sa konkatenujú.

- Veľkosť matice: $[b, 4lstm]$

5 Vektor sa zmenší na potrebný veľkosť použitím lineárneho filtra. Lineárny filter sa učí.

- Výstupná matica: $[b, 2]$

4.4 Vybranie krátkej odpovede z vety

Pri vybraní krátkej odpovede sa predpovedá začiatok a koniec odpovede. Koniec odpovede musí byť vždy po začiatku odpovede, a preto vybranie najväčších pravdepodobností nezávisle nie je možné. Predikcie budú normalizované funkciou softmax aby začiatok a koniec mali rovnaký podiel na predikcií. Najlepšia odpoveď bude mať najväčšiu sumu predikcií, ale predikcia konca musí nasledovať predikcií začiatku. V normálnych prípadoch sa preferujú kratšie odpovede, ale vyberá sa iba z krátkeho kontextu, a preto všetky potenciálne odpovede sú dostatočne krátke.

Neurónová sieť sa učí po dávkach. Ako vstup berie otázku rozdelenú na slová a vetu rozdelenú na slová. Neurónová sieť má dva výstupy. Výstupy majú veľkosť počtu slov. Jeden z výstupov je predikcia začiatka odpovede a druhý výstup je predikcia konca odpovede.

Model e1

Model e1 je kópiou základového modelu [4].

1 Otázka a text sa konvertujú na vektory.

- Veľkosť matice otázky: $[b, q, e]$
- Veľkosť matice textu: $[b, k, e]$

2 Matica otázky a matica vety sa prevedú cez LSTM. Používa sa rovnaké LSTM z pohľadu učenia, ale prechádzajú ním zvlášť. LSTM by sa mal naučiť identifikovať sémantický význam vety a otázky z pohľadu odpovedania na otázku.

- Veľkosť matice otázky: $[b, q, 2lstm]$
- Veľkosť matice textu: $[b, k, 2lstm]$

3 Matica otázky sa zmenší na vektor. Dimenzia dĺžky otázky sa odstráni použitím získania maxima cez dĺžku otázky. Maximum cez dĺžku otázky spojí najextrémnejšie vlastnosti všetkých vektorov.

- Veľkosť matice otázky: $[b, 2lstm]$

4 Na maticu otázky sa aplikujú dva lineárne filtre bez zmeny veľkosti. Jeden sa učí predikovať začiatok odpovede a druhý sa učí predikovať koniec odpovede. Vzniknú dve matice otázky. K maticiam otázky sa pridá prázdna dimenzia.

- Veľkosť matice textu: $[b, k, 2lstm]$
- Veľkosť dvoch matíc otázky: $[b, 2lstm, 1]$

5 Matica textu sa po dávkach vynásobí s každou maticou otázky. Vzniknú tak dve výstupné matice.

- Veľkosť matíc: $[b, k, 1]$

5 Po odstránení prázdnej dimenzie dostaneme výstupné vektory.

- Veľkosť matíc: $[b, k]$

Model e2

Model e2 má rovnakú štruktúru ako model e1 až na jeden rozdiel. Model e2 má iný spôsob redukcie otázky na vektor. Model e1 zoberie výstup z LSTM a zmenší ho na jeden vektor extrahovaním maxima cez vektory. Model e1 redukuje otázku ne vektor konkatenáciou vnútorných stavov LSTM. LSTM je použitý obojstranne, a preto má dva vnútorné stavy.

Model e3a

1 Otázka a text sa konvertujú na vektory.

- Veľkosť matice otázky: $[b, q, e]$
- Veľkosť matice textu: $[b, k, e]$

2 Matica otázky sa prevedie cez LSTM. Zoberie sa posledný stav LSTM.

- Veľkosť matice otázky: $[2, b, lstm]$

2 Dva vnútorné stavy LSTM sa konkatenujú. Jeden stav je pre prechod spredu a druhý stav je pre prechod zozadu.

- Veľkosť matice otázky: $[b, 2lstm]$

3 Na maticu otázky sa aplikujú dva lineárne filtre, ktoré zmenia veľkosť vektora na e. Jeden sa učí predikovať začiatok odpovede a druhý sa učí predikovať koniec odpovede. Vzniknú dve matice otázky. K maticiam otázky sa pridá prázdna dimenzia.

- Veľkosť dvoch matíc otázky: $[b, 1, e]$

4 Za oboje matice otázky sa konkatenuje text.

- Veľkosť matíc: $[b, 1 + k, e]$

5 Matice prejdu cez rovnaké LSTM aké je použité v bode 2. Pretože je na začiatku vektor reprezentujúci otázku, jednotlivé hodnoty pre slová textu by mali byť závislé od otázky a aj okolného kontextu.

- Veľkosť matíc: $[b, 1 + k, 2lstm]$

5 Z matíc sa odstráni prvý prvok.

- Veľkosť matíc: $[b, k, 2lstm]$

6 Dimenzia výstupu LSTM sa zruší použitím lineárneho filtra. Lineárny filter sa naučí vypočítať z vektora jedno číslo.

- Veľkosť matíc: $[b, k, 1]$

6 Po odstránení prázdnej dimenzie vynikne výstupná matica.

- Veľkosť výstupných matíc: $[b, k]$

Model e3b

1 Otázka a text sa konvertujú na vektory.

- Veľkosť matice otázky: $[b, q, e]$
- Veľkosť matice textu: $[b, k, e]$

2 Otázka sa konkatenuje pred text. Ide o najjednoduchšiu kombináciu otázky a textu.

- Veľkosť matice: $[b, q + k, e]$

3 Kombinovaná matica prejde cez LSTM.

- Veľkosť matice: $[b, q + k, 2lstm]$

4 Časť pre otázku bude odstránená z matice.

- Veľkosť matice: $[b, k, 2lstm]$

5 Dimenzia výstupu LSTM sa zruší použitím lineárneho filtra. Použijú sa dva lineárne filtre. Jeden pre predikciu začiatku odpovede a druhý pre predikciu konca odpovede. Lineárny filter sa naučí vypočítať z vektora jedno číslo.

- Veľkosť matíc: $[b, k, 1]$

5 Po odstránení prázdnej dimenzie vynikne výstupná matica.

- Veľkosť výstupných matíc: $[b, k]$

Model e6a

1 Otázka a text sa konvertujú na vektory.

- Veľkosť matice otázky: $[b, q, e]$
- Veľkosť matice textu: $[b, k, e]$

2 Matica otázky prejde cez LSTM. Z LSTM sa získa posledný vnútorný stav.

- Veľkosť matice otázky: $[2, b, lstm]$

2 Dva vnútorné stavy LSTM sa konkatenujú. Jeden stav je pre prechod spredu a druhý stav je pre prechod zozadu.

- Veľkosť matice otázky: $[b, 2lstm]$

3 Matica otázky sa prevedie cez dva lineárne filtre, ktoré nemenia veľkosť vektora. Jeden sa bude učiť predpovedať začiatok odpovede a druhý sa bude učiť predpovedať koniec odpovede.

- Veľkosť matíc otázky: $[b, 2lstm]$

4 Matica otázky sa konkatenuje za každý vektor textu.

- Veľkosť matíc textu: $[b, k, e + 2lstm]$

5 Matice textu sa prevedú cez LSTM. LSTM má inú veľkosť vstupu ako LSTM na konverziu otázky, a preto učené nezávisle od prvého LSTM.

- Veľkosť matíc textu: $[b, k, 2lstm]$

6 Dimenzia výstupu LSTM sa zruší použitím lineárneho filtra. Lineárny filter sa naučí vypočítať z vektora jedno číslo.

- Veľkosť matíc textu: $[b, k, 1]$

6 Po odstránení prázdnej dimenzie vynikne výstupná matica.

- Veľkosť výstupných matíc: $[b, k]$

Model e6b

Model e6b má rovnakú štruktúru ako model e6a až na jeden rozdiel. V modele e6b sú v kroku 6 použité dva rôzne lineárne filtre. Jeden lineárny filter sa učí predpovedať začiatok odpovede a druhý lineárny filter sa učí predpovedať koniec odpovede. Model e6b rozlišuje medzi predpoveďou začiatku a konca odpovede v krokoch 3 a 6.

Model e6c

Model e6c má rovnakú štruktúru ako model e6a až na dva rozdieli. V modele e6c sú v kroku 6 použité dva rôzne lineárne filtre rovnako ako v modele e6b. V modele e6c je na rozdiel od modelu e6a a e6b vynechaný krok 3. Model e6c rozlišuje medzi predpoveďou začiatku a konca odpovede iba v kroku 6.

4.5 Vyhľadanie článku s odpoveďou

Vyhľadanie článku s odpoveďou má unikátne problémy, ktoré nie sú prítomné pri ostatných moduloch. Dátovú sadu je potrebné rozdeliť na trénovaciu a validačnú časť. Dátová sada obsahuje veľa (68.7%) článkov, ktoré majú iba jednu otázku. Keď sa bude neurónová sieť učiť iba na otázkach, tak učenie nebude efektívne, kvôli nedostatočne veľkej dátovej sade. Jeden zo vstupov neurónovej siete by mala byť aj Wikipédia. Wikipédia ako vstup sa pri jednotlivých otázkach nemení, a preto nemusí byť priamym vstupom. Wikipédia môže byť vstupom aj pri trénovaní neurónovej siete. Ak bude Wikipédia vstup iba pri trénovaní, tak sa jedná o problém podobný klasifikácií textu, kde je počet tried veľký ako počet článkov a každá trieda patrí jednému článku. Ak je Wikipédia priamym vstupom neurónovej siete, tak sa jedná o problém podobný porovnaniu textov. Otázku je potrebné porovnať s každým článkom a vybrať článok s najväčšou pravdepodobnosťou odpovede. Zakaždým spracovávať články je časovo náročné a spracovávať všetky články naraz je pamäťovo náročné.

Na učenie vyhľadania článku nemusí stačiť iba normálne učenie, a preto pre dosiahnutie dobrých výsledkov je potrebné vytvoriť viacero spôsobov učenia.

- Kategorizácia iba na základe otázok je štýl modelu kde sa Wikipédia poskytne neurónovej sieti iba pri učení. Otázky sú rozdelené medzi trénovacou a validačnou časťou dátovej sady. Je to najjednoduchšie rozdelenie, ale zároveň nebude veľmi efektívne. Bude sa dať použiť maximálne pri porovnaní výsledkov.

- Kategorizácia na základe článkov je štýl modelu kde sa Wikipédia poskytne neurónovej sieti iba pri učení. Do trénovacej časti sa dajú vety jednotlivých článkov. Vo validačnej časti budú otázky z dátovej sady. Modul bude trénovaný na priradenie vety z článku danému článku, ale bude validovaný na priradenie článku otázke. Zlepší to problém nedostatočne veľkej dátovej sady drastickým zväčšením počtu trénovacích dát. Trénovacích dát môže byť príliš veľa a dĺžky článkov sú drasticky rozdielne. Článok, ktorý má 5 viet, bude oveľa menej reprezentovaný ako článok, ktorý má 800 viet. Obmedzenie počtu viet na článok dokáže vyvážiť trénovacie dáta a potenciálne zlepší presnosť. Aj keď sa bude modul trénovať na priradenie vety článku, bude sa validovať na priradenie článku otázke.

- Kategorizácia na základe článkov a otázok je štýl modelu kde sa Wikipédia poskytne neurónovej sieti iba pri učení. Otázky sú rozdelené medzi trénovaciou a validačnou časťou dátovej sady. Do trénovacej časti sa pridajú vety článku. Modul sa bude naraz učiť priraďovať vety článkom a priraďovať články otázkam. Modul by mal mať dostatočný počet dát k článkom a aj formát otázky, ale nie je separácia medzi vetou článku a otázkou.

- Kategorizácia na základe článkov a otázok s rozdeleným učením je štýl modelu kde sa Wikipédia poskytne neurónovej sieti iba pri učení. Modul je učený vo dvoch fázach. V oboch fázach bude validačná časť skladaná z otázok dátovej sady. V prvej fáze bude

trénovacia časť skladaná z viet článkov a modul sa bude učiť priradiť vetu článku. Druhá trénovacia časť bude skladaná z otázok dátovej sady. V druhej fáze sa zoberie modul naučený na vetách článkov a doučí sa na otázkach. V prvej fáze by sa modul mal naučiť vlastnosti jednotlivých článkov a v druhej fáze by sa mal modul naučiť naviazať vlastnosti článkov na formát otázky.

- Priradenie článku porovnaním článku a otázky je štýl modelu kde sa Wikipédia poskytne neurónovej sieti pri učení a aj pri každom výpočte. Modul bude mať na vstupe otázky a články. Na výstupe bude skóre pre každú kombináciu otázky a článku. Na vstup modulu nemôžu byť dané všetky otázky naraz, kvôli pamäťovej zložitosti. Keď skóre priradené článku nie je ovplyvnené ostatnými článkami, tak spracovanie článkov po dávkach by nemalo zmeniť výsledky.

Model ak

1 Otázka alebo veta sa najskôr skonvertuje na vektory.

- Veľkosť matice: $[b, q, e]$

2 Na maticu sa použije LSTM. LSTM pôjde z oboch strán. LSTM nájde príznaky v otázke a vytvorí novú maticu.

- Veľkosť matice: $[b, q, 2lstm]$

2 Dimenzia dĺžky otázky alebo vety sa odstráni použitím získania maxima cez dĺžku otázky. Maximum cez dĺžku otázky spojí najextrémnejšie vlastnosti všetkých vektorov.

- Veľkosť matice: $[b, 2lstm]$

3 Použitím lineárneho filtra sa získané vektory prevedú na veľkosť podľa počtu rozpoznávaných článkov. Lineárny filter funguje ako učiaca sa pamäť.

- Veľkosť výstupnej matice: $[b, a]$

Model ac

1 Otázka a články sa najskôr konvertujú na vektory.

- Veľkosť matice otázky: $[b, q, e]$
- Veľkosť matice článkov: $[a, k, e]$

2 Na maticu otázky a článkov sa použije LSTM. LSTM pôjde z oboch strán. LSTM nájde príznaky v otázke a vytvorí novú maticu.

- Veľkosť matice otázky: $[b, q, 2lstm]$
- Veľkosť matice článkov: $[a, k, 2lstm]$

2 Dimenzia dĺžky otázky a článkov sa odstráni použitím získania maxima cez dĺžku otázky. Maximum cez dĺžku otázky alebo článkov spojí najextrémnejšie vlastnosti všetkých vektorov.

- Veľkosť matice otázky: $[b, 2lstm]$
- Veľkosť matice článkov: $[a, 2lstm]$

3 Matice článkov sa zopakuje pre každú otázku dávky. K matice otázky sa pridá prázdna dimenzia.

- Veľkosť matice článkov: $[b, a, 2lstm]$
- Veľkosť matice otázky: $[b, 2lstm, 1]$

4 Matica článkov sa po dávkach vynásobí s maticou otázky.

- Veľkosť matíc: $[b, a, 1]$

5 Po odstránení prázdnej dimenzie dostaneme výstupnú maticu.

- Veľkosť matíc: $[b, a]$

4.6 Použitie tf.idf pri získaní odpovede

Algoritmus tf.idf je založený na inom koncepte ako učenie neurónovej siete, a preto majú rozdielne výhody a problémy. Často sa používa pri vyhľadávaní na internete. Pretože algoritmus tf.idf sa používa na priradenie textu do množiny textov, dá sa použiť pri vyhľadávaní článku s odpoveďou alebo vyhľadávaní vety s odpoveďou. Neurónové siete môžu byť úplne nahradené algoritmom tf.idf alebo skóre z oboch systémov môže byť kombinované.

Term Frequency a Inverse Document Frequency nie sú pevne dané a môžu sa meniť. Najbežnejšia funkcia pre Term Frequency je normalizovaný počet $tf(d, w) = \frac{w_{count}}{d_{len}}$, kde w je slovo, d je dokument, w_{count} je počet výskytu slova v dokumente a d_{len} je dĺžka dokumentu. Ak by sa počet výskytov nedelil dĺžkou dokumentu, tak by boli preferované dlhé dokumenty. Ďalšou častou funkciou pre Term Frequency je existencia slova v dokumente $tf(d, w) = \begin{cases} 1 & w \text{ in } d \\ 0 & w \text{ not in } d \end{cases}$. Najbežnejšia funkcia pre Inverse Document Frequency je $idf(w) = \log(\frac{d_{total}}{d_w + 1})$, kde w je slovo, d_{total} je počet všetkých dokumentov a d_w je počet dokumentov, ktoré obsahujú slovo w. K d_w sa pripočíta 1 aby nevzniklo delenie nulou. [18]

Príklad výpočtu tf.idf

Ak sa hľadá otázka: „Čím je pokrytý fotografický film?“

Na výber sú vety:

1. „Fotografický film je plastový pás z polyesteru, nitrocelulózy nebo acetátu celulosy, pokrytý tenkou vrstvou emulze obsahující světlocitlivé halogenidy stříbra vázané v želatině, s rozdílnou velikostí krystalů, určující citlivost a zrnitost (rozlišení) filmu.“
2. „Když je emulze vystavena působení dostatečného množství světla, nebo jiného elektromagnetického záření jako např. rentgen, vytvoří se latentní (neviditelný) obraz.“
3. „Chemickými procesy se poté na filmu může vytvořit obraz viditelný.“

Ak text nie je normalizovaný, Term Frequency je normalizovaný počet a berú sa do úbahy iba jednotlivé slová. tak sa skóre pre jednotlivé vypočíta ako:

$$tf.idf(q, d_1) = tf(\text{Čím}, d_1)idf(\text{Čím}) + tf(\text{je}, d_1)idf(\text{je}) + tf(\text{pokrytý}, d_1)idf(\text{pokrytý}) + tf(\text{fotografický}, d_1)idf(\text{fotografický}) + tf(\text{film}, d_1)idf(\text{film})$$

$$tf.idf(q, d_1) = \frac{0}{32}\log(\frac{3}{1}) + \frac{1}{32}\log(\frac{3}{3}) + \frac{1}{32}\log(\frac{3}{2}) + \frac{1}{32}\log(\frac{3}{2}) + \frac{1}{32}\log(\frac{3}{2})$$

$$tf.idf(q, d_1) = 0 + 0.03125\ 0 + 0.03125\ 0.176 + 0.03125\ 0.176 + 0.03125\ 0.176$$

$$tf.idf(q, d_1) = 0.0055 + 0.0055 + 0.0055$$

$$tf.idf(q, d_1) = 0.0165$$

$$tf.idf(q, d_2) = tf(\text{Čím}, d_2)idf(\text{Čím}) + tf(\text{je}, d_2)idf(\text{je}) + tf(\text{pokrytý}, d_2)idf(\text{pokrytý}) + tf(\text{fotografický}, d_2)idf(\text{fotografický}) + tf(\text{film}, d_2)idf(\text{film})$$

$$tf.idf(q, d_2) = \frac{0}{20}\log(\frac{3}{1}) + \frac{1}{20}\log(\frac{3}{3}) + \frac{0}{20}\log(\frac{3}{2}) + \frac{0}{20}\log(\frac{3}{2}) + \frac{0}{20}\log(\frac{3}{2})$$

$$tf.idf(q, d_2) = 0$$

$$tf.idf(q, d_3) = tf(\text{Čím}, d_3)idf(\text{Čím}) + tf(\text{je}, d_3)idf(\text{je}) + tf(\text{pokrytý}, d_3)idf(\text{pokrytý}) + tf(\text{fotografický}, d_3)idf(\text{fotografický}) + tf(\text{film}, d_3)idf(\text{film})$$

$$tf.idf(q, d_3) = \frac{0}{10}\log(\frac{3}{1}) + \frac{0}{10}\log(\frac{3}{3}) + \frac{0}{10}\log(\frac{3}{2}) + \frac{0}{10}\log(\frac{3}{2}) + \frac{0}{10}\log(\frac{3}{2})$$

$$tf.idf(q, d_3) = 0$$

Pri takomto nastavení by z viet bola vybraná prvá veta, pretože sú v nej slová "pokrytý", "fotografický"a "film". Slovo "je"nemá vplyv na vybranie vety, pretože je príliš všeobecné, a to je rozdiel medzi tf.idf a Bag of Words. Správna odpoveď je: „tenkou vrstvou emulze“, a preto prvá veta je vybraná správne.

Pri praktickom príklade vidno potrebu normalizácie textu. Slová "film"a "filmu"sú brané ako dve rozdielne slová. V angličtine by to bolo jedno slovo "film".

Ak by boli brané do úvahy aj dvojice slov (bigramy), tak by bolo v prevej vete nájdené slovné spojenie "fotografický film", a to by zlepšilo šancu vybrania prvej vety.

4.7 Kombinácia modelov

Jednotlivé moduly dokážu spracovať iba časť problému, a preto na zodpovedanie otázky musí byť použitá kombinácia modelov. Pri niektorých pod problémoch sa dá použiť aj algoritmus tf.idf.

Odpovedanie áno alebo nie

Odpovedanie "áno/nie"sa dá špecializovaným modulom, ale kombinovanie odpovedania "áno/-nie"a extrahovanie časti textu do jednej neurónovej siete môže potenciálne zlepšiť presnosť zdieľaním časti neurónovej siete na pochopenie textu. Kombinácia "áno/nie/časť textu"sa dá riešiť rovnakým modelom ako extrahovanie časti textu, ale výstup musí byť rozšírený. Rozšírenie výstupu sa dá najlepšie urobiť rozšírením vstupu o špeciálne symboly. Rozšírenie vstupu spôsobí že všetky časti neurónovej siete sa budú podieľať na získaní odpovede, a preto sa použijú všetky prvky neurónovej siete. Ak by bolo rozšírenie výstupu vykonané uprostred neurónovej siete, tak by sa časť ako konvertovanie textu na vektor nemohla použiť na odpovedanie "áno/nie".

Na vstup sa dá priradiť jeden symbol. Pri symbole skóre začiatku odpovede by mohlo označovať "áno"a skóre konca odpovede by mohlo označovať "nie".

Ak by boli na vstup priradené dva symboly, tak jeden symbol bude označovať odpoveď "nie"a druhý symbol bude označovať odpoveď "áno". Skóre pre odpoveď "áno/nie"bude súčet skóre pre začiatok a koniec odpovede na špeciálnom symbole.

Pri odpovedaní "áno/nie/časť textu"musí byť stále použitý modul na rozlišovanie medzi odpoveďami "áno/nie"alebo "časť textu". Rozlišovanie medzi typmi odpovede a samotným odpovedaním je príliš zložité pre jeden modul. Pri rozdelenom odpovedaní "áno/nie"alebo "časť textu"sa vyberie modul na spracovanie. Pri odpovedaní "áno/nie/časť textu"sa vyberie časť odpovede, ktorá je relevantná pre typ odpovede.

Výber z viacero odpovedí

Predpovede neurónových sietí nemusia byť veľmi presné. Keď sa vyberie niekoľko najlepších predpovedí, tak pravdepodobnosť že sa odpoveď nachádza v nich sa zlepšuje, ale vznikne nový problém vybrania medzi kandidátmi. Vybrať sa dá niekoľko článkov a z každého článku sa dá vybrať niekoľko viet. Vybraním odpovede kombináciu skóre z každej časti by sa mala teoreticky zlepšiť presnosť vybrania odpovede.

1. Na základe otázky sa vyberie a článkov.
2. Z každého článku sa vyberie s viet. Vznikne $a * s$ kontextov.
3. Z každého kontextu sa izoluje odpoveď. Vznikne $a * s$ kandidátov na odpoveď.
4. Z kandidátov sa vyberie odpoveď.

Časová náročnosť takéhoto systému môže byť ťažko zvládnuteľná, a preto môže byť vhodné redukovať počet kandidátov uprostred spracovania. Redukovanie možností po kroku 1 by bolo rovnaké ako redukovanie a. Krok 3 nemení počet kandidátov. Krok 4 je redukovanie kandidátov na jeden výsledok po kroku 3. Jediné miesto vhodné na redukovanie možností je medzi krokmi 2 a 3. Na redukovanie možností sa dá použitím skóre z predchádzajúcich dvoch modulov alebo sa dá použiť aj tf.idf na kandidátne kontexty a otázku. Kontexty budú považované za dokumenty a počíta sa skóre otázky ku kontextom. Ak sa v kroku 2 použije neurónová sieť, tak použitím tf.idf medzi krokmi 2 a 3 bude získaná výhoda použitia viacerých konceptov.

Kapitola 5

Analýza výsledkov

Dátovú sadu je potrebné rozdeliť na trénovaciu a validačnú časť. Do validačnej časti pôjde každý tretí prvok a do trénovacej časti pôjdu ostatné. Týmto rozdelením budú v ténovacej časti dve tretiny dátovej sady a vo validačnej časti bude jedna tretina dátovej sady. Zároveň sa týmto spôsobom zlepší pestrosť jednotlivých modelov, zlepšením pestrosti trénovacej a validačnej časti dátovej sady. Po sebe idúce otázky sa môžu týkať rovnakého článku a mohli byť vyrábané v rovnakom čase tým istým človekom. Rozprestrením častí cez celú dátovú sadu by sa mala zabezpečiť najlepšia presnosť. [12]

Skratky

Skratka "Base"označuje inteligentné tipovanie. Ak neurónová sieť získa presnosť nad inteligentné typovanie, tak by to malo znamenať že sa logické spojenia naozaj naučila. Inteligentné typovanie bolo dosiahnuté učením neurónovej siete, ktorá nebrala do úvahy bežný vstup. Napríklad predpovedanie bez otázky alebo bez textu.

Neurónové siete sú označované skratkou modelu neurónovej siete. Ak skratka nie je rozšírená o ďalšie údaje, tak je učená a testovaná na normálom rozdelení a model nemá ďalšie nastavenia.

- Ak je model iteračný, tak je počet iterácií označovaný ako "x it.". Ak je model najskôr trénovaný na jednom nastavení iterácií, a potom dotrénovaný na druhom nastavení iterácií, tak to označujem "$x \rightarrow y$ it.".
- Ak vstup je získaný kontext, tak šírka kontextu alebo polomer kontextu môže byť nastaviteľný. Ak sa modelu pošle iba veta s odpoveďou alebo polomer kontextu je 0, tak to značím "k: 1". Ak je polomer kontextu 1, tak sa modelu pošle veta s odpoveďou a jedna veta pred a po nej, a tým vzniknú tri vety "k: 3". Značkou "k: 5"sa značí kontext s polomerom 2, kde sa modelu pošle veta s odpoveďou a dve vety pred a po nej, a tým vnikne päť viet. Ak sa model trénuje a testuje na rozdielnych kontextoch, tak prvé číslo značí kontext pri trénovaná a druhé číslo značí kontext pri testovaní. Napríklad značka "k: 3, 1"znamená že model bol trénovaný na kontexte o troch vetách a testovaný bol na kontexte o jednej vete.
- Modely na vybranie článku môžu byť trénované na viacerých vstupoch. Ak je model trénovaný iba na otázkach, tak to značím "t: o". Ak je model trénovaný na vetách článku z Wikipédie, tak to značím "t: v". Ak je model trénovaný na kombinácií otázok a vetách z článku Wikipédie, tak to značím "t: o+v". Ak je model trénovaný najskôr

na vetách z článku Wikipédie, a potom dotrénovaný na otázkach, tak to značím "t: v → o".

Algoritmus tf.idf má rozličné nastavenia.

- Ak je text normalizovaný na základný tvar slova pre algoritmus tf.idf, tak to značím skratkou "n". Ak text nie je normalizovaný, tak algoritmus nemá značku pre typ textu.
- V projekte používam dve rôzne funkcie tf. Funkcia tf reprezentujúca existenciu slova v článku, bude mať priradený značku "e". Funkcia tf reprezentujúca počet výskytov slova v článku normalizovaná dĺžkou článku, bude mať značku "np".
- Algoritmus tf.idf počíta jednotlivé slová alebo unigramy, a to budem značiť skratkou "u". Ak bude algoritmus tf.idf brať do úvahy aj dvojice slov alebo bigramy, tak to budem značiť skratkou "u+b".

Rozdelenie otázok podľa typu odpovede

Úspešnosť rozdeľovania otázok podľa typu odpovede je v tabuľke 5.1. Najlepší klasifikátor je v1 s presnosťou 98.60%. Po naučení neurónová sieť v1 kategorizovala správne 8 otázok, ktoré neurónová sieť v2 nevedela kategorizovať. Neurónová sieť v2 kategorizovala správne 5 otázok, ktoré neurónová sieť v1 kategorizovala nesprávne.

Najčastejší problém je s otázkami typu "OTHER": „Je entita typ jedna alebo typ dva?“ Napríklad: „Je dubové dřevo tvrdé, nebo měkké?“ Sieť sa pozrie na časť „Je entita typ jedna. . . “ a povie že odpoveď je "áno/nie", ale odpoveď musí byť "typ jedna"alebo "typ dva"čo je kus z textu.

Vyhľadanie vety s odpoveďou

Úspešnosť vybrania vety z článku je v tabuľke 5.2 obsahuje všetky systémy na získanie vety s odpoveďou. Buď sa jedná o neurónovú sieť alebo algoritmus tf.idf s rôznymi nastaveniami. Pri niektorých neurónových sietiach je rôzne nastavenie počtu iterácií. Pre systémy sa sleduje presnosť pri vybraní jednej vety a presnosť nájdenia vety pri vybraní piatich najlepšie ohodnotených viet. Najlepší systém je r2 s presnosťou 69.26% pri výbere jednej vety a 86.41% pri výbere z piatich najlepšie ohodnotených viet. V experimentálnych výsledkoch sa dá pozorovať negatívna závislosť medzi úspešnosťou a zložitosťou neurónovej siete. Zo zväčšením počtu iterácií sa presnosť zhoršuje, napriek tomu že má neurónová sieť viacej príležitostí pracovať nad dátami. Zníženie presnosti pri komplikovaných sietiach môže mať na príčine fenomén miznúceho koeficientu, a preto som pridal prípad kde sa učí neurónová sieť najskôr na jednej iterácií a potom až na dvoch iteráciách. Učenie najskôr na jednej iterácií a potom prejdenie na dve iterácie je lepšie ako učenie pri dvoch iteráciách, ale horšie ako neurónové siete učené pri jednej iterácií.

V tabuľke 5.3 je porovnanie medzi niektorými systémami na vybranie vety z článku. V políčku mriežky je počet otázok, ktoré systém v riadku predikuje správne, ale systém v stĺpci nie. Porovnaním s inteligentným odhadom sa dá vyčítať že neurónové siete viacej spoliehajú na inteligentný odhad ako algoritmus tf.idf. Rozdieli medzi neurónovými sieťami sú väčšie ako rozdieli medzi nastaveniami algoritmu tf.idf, a to môže byť spôsobené náhodnosťou učenia neurónových sietí. Pri algoritme tf.idf náhodnosť nie je. Rozdieli medzi neurónovými sieťami a nastaveniami algoritmu tf.idf sú väčšie ako rozdieli vo vlastných kategóriách, a to môže byť dôkazom použitia rozdielnych spôsobom pre vyhodnocovanie.

Neurónová sieť a algoritmus tf.idf majú oveľa viac rozdielnych správnych odpovedí, než by ich rozdiel v presnosti naznačoval.

Úspešnosti niektorých systémov sú rozdelené podľa typu otázky a odpovede v tabuľke 5.4. Neurónové siete majú problémy s otázkami typu "CLAUSE"a "OTHER". Otázok typu "OTHER"je málo, a preto sa neurónové siete na ne ťažko učia. Otázky typu "CLAUSE"sú zložité pravdepodobne pretože majú hlavnú časť otázky doloženú o ďalšiu informáciu a hľadať viacero informácií zároveň je zložité pre neurónovú sieť. Algoritmus tf.idf nemá typy otázky a odpovede, ktoré sú pre neho veľmi nezvládnuteľné. Pri algoritme tf.idf nezáleží na logike alebo exotickosti otázky, ale ide iba o podobnosť otázky a odpovede.

Odpovedanie áno alebo nie

Úspešnosť odpovedania "áno/nie"je v tabuľke 5.5. Najúspešnejší vyhľadávač je model yn1 s presnosťou 72.21%, ak je model učený a validovaný iba na správnej vete s odpoveďou. Základ pre odpovedanie "áno/nie"je 67.94%, keď sa na každú otázku odpovie "áno". Ak sa modul učí a validuje na rozšírenom kontexte aj s vetami okolo vety s odpoveďou, tak sa presnosť modulu zhoršuje. Keď sa zoberie 5 viet, tak výsledky naučeného modelu sú rovnaké ako inteligentné typovanie. Kombinovanie odpovedania "áno/nie"a vybrania vety časti textu nezlepšuje presnosť odpovedania "áno/nie". Špecializovaný systém je úspešnejší než kombinovaný.

V tabuľke 5.6 je porovnanie medzi niektorými systémami na odpovedanie "áno/nie". V políčku mriežky je počet otázok, ktoré systém v riadku predikuje správne, ale systém v stĺpci nie. Medzi systémami nie sú dostatočne veľké rozdieli na to aby sa nedali vysvetliť náhodnosťou pri učení. Systém, ktorý vyčnieva najviac je, "e3b + yn, k: 1"odpovedá najviac správnych odpovedí "nie". Aj napriek tomu že nemá zlú presnosť, v porovnaní s ostatnými systémami neodpovedá správne na významnejšie množstvo otázok.

Úspešnosti niektorých systémov sú rozdelené podľa typu otázky a odpovede v tabuľke 5.7. Typ odpovede je iba jeden a aj typov otázky je málo. Žiadny typ otázky nie je ťažko zvládnuteľný.

Vybranie krátkej odpovede z vety

Úspešnosť extrahovania odpovede vybraním časti textu je v tabuľke 5.8. Najlepší systém má presnosť 59.13%. Je založený na neurónovej sieti e1. Neurónová sieť je učená na kombináciu predpovedania "áno/nie"a získanie časti textu. Najlepší výsledok sa docieli pri vyberaní iba z jednej vety. Mierka "em"určuje počet presných odpovedí a mierka "f1"určuje priemernú relevanciu odpovede. Mierka "f1"je oveľa nežnejšia, keď sa neurónová sieť zmýli iba o jedno slovo.

V tabuľke 5.9 je porovnanie medzi niektorými systémami na extrakciu odpovede z textu. V políčku mriežky je počet otázok, ktoré systém v riadku predikuje správne, ale systém v stĺpci nie. Aj napriek tomu že majú systémy podobné presnosti, ich správne odpovede sú dosť odlišné. Aj napriek tomu že všetky systémy pracujú na základe LSTM, zložitosť extrakcie odpovede z textu a náhodnosť pri učení spôsobia rôzne odpovede.

Úspešnosti niektorých systémov sú rozdelené podľa typu otázky a odpovede v tabuľke 5.10. Otázky typu "CLAUSE", "VERB_PHRASE"a "OTHER"majú veľmi nízku presnosť. Extrakcia odpovede je zložitý problém, a preto je potrebné veľké množstvo otázok a odpovedí daného typu na správne naučenie. Pri odpovediach majú horšiu presnosť odpovede, ktoré nemajú presný tvar ako "OTHER"a "ORGANIZATION". Typ "OTHER"znamená že odpoveď nemá kategóriu a tým ani presný tvar. Názvy organizácií sa medzi sebou líšia

oveľa viacej ako mená ľudí. Dátum má ešte presnejší tvar. Ak systém vie že hľadá dátum a dostane správnu vetu, tak z veľkou pravdepodobnosťou vyberie správne.

Vyhľadanie článku s odpoveďou

Pri vyberaní článku beriem do úvahy iba 7193 článkov prítomných v dátovej sade. Presnosti naučených modelov sú v tabuľke 5.11. Najlepšia naučená neurónová sieť je ak 53.1%, ktorá bola učená naraz otázkach a prvých 50 vetách z článkov. Najlepšie nastavenie algoritmu tf.idf 50.08% obsahovalo normalizovanie textu a otázky na základný tvar slov, funkcia tf rozlišovala iba existenciu slova v článku nedbajúc na počet výskytov a hľadali sa jednotlivé slová (unigramy) aj dvojice slov (bigramy). Najlepšie výsledky mala kombinácia najlepšie naučenej neurónovej siete a algoritmu tf.idf 76.86%. Skóre oboch boli jednoducho sčítané bez akéhokoľvek upravovania alebo normalizácie.

V tabuľke 5.12 je porovnanie medzi niektorými systémami na vyhľadanie článku s odpoveďou. V políčku mriežky je počet otázok, ktoré systém v riadku predikuje správne, ale systém v stĺpci nie. Systémy "ak, t: o"a "ak, t: v"majú nízku presnosť a je to pri porovnaní vidno. Výsledky "ak, t: o"sa viacej podobajú neurónovým sietiam a výsledky "ak, t: v"sa viacej podobajú algoritmu tf.idf. Pri pridaní algoritmu tf.idf k neurónovej sieti, je vidno stratenie niektorých správnych odpovedí neurónovej siete a algoritmu tf.idf, ale kombinovaní systém získa veľké množstvo iných správnych odpovedí. Pretože systém ac porovnáva text s otázkou, predpokladal som že jeho výsledky budú podobné algoritmu tf.idf, ale správna priradenie systému ac je podobnejšie ostatným neurónovým sietiam.

Úspešnosti niektorých systémov sú rozdelené podľa typu otázky a odpovede v tabuľke 5.13. Otázky typu "ENTITY"a "PERSON"nie sú najlepšie pre neurónové siete, pretože sa často pýtajú práve na meno článku a nájsť meno článku znamená nájsť odpoveď. Pre algoritmus tf.idf sú extrémne zlé otázky aj odpovede kategórie ABBREVIATION. Otázky na skratky sú dosť normalizované. Potrebný článok sa pravdepodobne bude odvíjať od jedného slova v otázke. Algoritmus tf.idf presmeruje systém na článok o skratkách namiesto článku o entite, ktorej sa skratka týka. Teoreticky vy sa mohla skratka nájsť aj vo všeobecnom článku o skratkách, ale hodnotenie iba získania článku to bude považovať za nesprávny predpoklad.

Kombinácia modelov

Tabuľka úspešnosti odpovedania na otázku pri kombinácií všetkých modelov je 5.14. Najlepšia presnosť je 46.67%. Dosiahnutá je pri systéme v1 na predpoveď typu odpovede. Článok sa predpovedá kombináciou neurónovej siete a algoritmu tf.idf. Neurónová sieť je naraz učená na otázkach a vetách článkov. Algoritmus tf.idf používa normalizovaný text pre články a otázku. Ako funkciu tf používa existenciu slova vo vete. Pracuje z jednotlivými slovami a aj z dvojicami slov. Na získanie vety s odpoveďou sa používa algoritmus tf.idf. Ako funkciu tf používa existenciu slova vo vete. Pracuje iba so samostatnými slovami. Na normalizácií textu nezáleží. Model na odpovedanie "áno/nie"je yn1 učený na jednej vete bez rozšíreného kontextu. Model na extrahovanie odpovede z textu je e1. Naučený bol na kombinácií extrahovania odpovede textu zároveň s odpovedaním "áno"alebo "nie". Učený bol iba na jednej vete bez rozšíreného kontextu. Pri kombinácií modelov nebol použitý rozšírený kontext, ale odpoveď bola hľadaná iba v jednej vete. Pri každom pod probléme bola vybraná bola iba najlepšia možnosť a viacero kandidátnych odpovedí generovaných nebolo.

Pre získanie typu odpovede sa vždy používa systém "v1". Medzi systémami "v1"a "v2"nie sú takmer žiadne rozdieli vo výsledkoch, a preto ich nie je potrebné meniť.

Pre získanie konečnej odpovede som testoval dva systémy pre odpovedanie "áno/nie"a dva systémy pre extrakciu odpovede z textu. Medzi výsledkami nebol veľký rozdiel a lepšie systémy samostatne mali lepšie výsledky pri kombinácií.

Systém na získanie článku má oveľa väčší vplyv na výslednú odpoveď. Kombinácia neurónovej siete a algoritmu tf.idf v porovnaní so samostatnou neurónovou sieťou alebo so samostatným algoritmom tf.idf spôsobí zásadné zlepšenie presnosti získania odpovede.

Pri výbere kontextu pre odpoveď je algoritmus tf.idf lepší ako neurónová sieť. Pravdepodobne poskytuje systému na získanie konečnej odpovede pre neho jednoduchší kontext. Pri rozhodovaním medzi viacero odpoveďami sa rozdiel medzi neurónovou sieťou a algoritmom tf.idf zväčší.

Aj napriek tomu že správna odpoveď je medzi kandidátmi skóre systému na extrakciu odpovede nie je stavané na vybranie odpovede z kandidátov. Redukcia kandidátov algoritmom tf.idf zlepšuje presnosť získania odpovede, keď sa používa neurónová sieť pre výber vety. Redukcia kandidátov algoritmom tf.idf nemá až taký veľký vplyv na presnosť získania odpovede, keď sa veta vyberá algoritmom tf.idf.

Vybranie širšieho kontextu pre odpoveď zhorší presnosť získania odpovede. Pre systém na extrakciu odpovede je širší kontext rušivý šum namiesto namiesto dát pre lepšiu predpoveď odpovede.

Ak sa vybranie odpovede z kandidátov nenechá iba na systém pre extrakciu odpovede ale používa sa kombinácia skóre všetkých častí, tak sa presnosť zlepší. Aj napriek tomu že medzi kandidátmi je viac správnych odpovedí, presnosť stále nie je lepšia ako pri vybraním najlepšej predikcie pri každej časti a ignorovaní kandidátov.

	Počet prvkov	v1[%]	v2[%]
Všetky	4486	99.60	99.53
Odpoveď			
Časť z textu	3723	99.65	99.57
Áno/nie	763	99.34	99.34
Kategória otázky			
ENTITY	849	100.00	99.88
VERB_PHRASE	765	99.35	99.35
DATETIME	618	100.00	100.00
PERSON	595	99.83	99.83
LOCATION	566	99.82	99.82
ADJ_PHRASE	497	99.40	99.40
NUMERIC	330	99.09	98.48
CLAUSE	158	100.00	100.00
ABBREVIATION	102	100.00	100.00
OTHER	6	16.67	16.67
Kategória odpovede			
YES_NO	763	99.34	99.34
OTHER	758	99.08	99.08
DATETIME	615	100.00	100.00
ENTITY	610	100.00	99.84
PERSON	602	99.67	99.67
LOCATION	561	99.82	99.82
NUMERIC	333	99.10	98.50
ORGANIZATION	99	100.00	100.00
ABBREVIATION	96	100.00	100.00
DENOTATION	49	100.00	100.00

Tabulka 5.1: Tabuľka úspešnosti predikcie typu odpovede.

Systém	em1[%]	em5[%]
Base	45.96	71.13
r1	66.15	85.37
r2	69.66	89.05
r3	61.44	84.16
r5, 1 it.	54.62	77.12
r6, 1 it.	55.63	77.86
r7, 1 it.	57.71	80.18
r7b, 1 it.	59.50	81.81
r5, 2 it.	52.73	76.18
r6, 2 it.	46.43	71.89
r7, 2 it.	50.36	74.51
r7b, 2 it.	45.02	68.59
r5, 3 it.	51.09	72.52
r6, 3 it.	46.69	71.25
r7, 3 it.	46.65	69.06
r7b, 3 it.	44.77	69.93
r5, 1 $\rightarrow$ 2 it.	57.28	79.51
r6, 1 $\rightarrow$ 2 it.	50.47	72.21
r7, 1 $\rightarrow$ 2 it.	55.16	76.72
r7b, 1 $\rightarrow$ 2 it.	56.17	79.09
tf.idf, e, u	62.72	81.13
tf.idf, e, u+b	61.89	80.95
tf.idf, n, e, u	66.05	85.93
tf.idf, n, e, u+b	65.91	85.28
tf.idf, np, u	52.22	79.03
tf.idf, np, u+b	54.01	79.90
tf.idf, n, np, u	54.41	82.00
tf.idf, n, np, u+b	56.62	82.69

Tabulka 5.2: Tabuľka úspešnosti vyhľadávania vety v článku.

	Base	r2	r7b	r5, 2 it.	r5, 1 → 2 it.	tf.idf, e, u	tf.idf, n, e, u	tf.idf, n, e, u+b	tf.idf, n, np, u+b
Base	0	189	259	248	178	799	786	760	1017
r2	1250	0	757	1008	834	949	889	872	1158
r7b, 1 it.	865	302	0	584	433	914	885	874	1151
r5, 2 it.	551	250	281	0	233	882	863	840	1086
r5, 1 → 2 it.	685	280	334	437	0	946	914	906	1168
tf.idf, e, u	1550	639	1059	1330	1190	0	278	266	669
tf.idf, n, e, u	1686	728	1179	1460	1307	427	0	137	662
tf.idf, n, e, u+b	1654	705	1162	1431	1293	409	131	0	632
tf.idf, n, np, u+b	1495	575	1023	1261	1139	396	240	216	0

Tabulka 5.3: Porovnanie výsledkov medzi jednotlivými systémami pre vyhľadávania vety v článku. V políčku mriežky je počet otázok, ktoré systém v riadku predikuje správne, ale systém v stĺpci nie.

	Prvkov	Base	r2[%]	r7, 3 it.[%]	tf.idf, e, u[%]	tf.idf, n, e, u+b[%]
Všetky	4476	45.96	69.66	46.65	62.72	65.91
Kategória otázky						
ENTITY	849	44.52	69.26	49.12	67.65	71.29
VERB_PHRASE	760	36.45	65.66	38.82	69.21	73.29
DATETIME	615	52.52	74.96	47.97	57.56	56.91
PERSON	594	57.58	71.89	57.58	63.13	65.66
LOCATION	566	61.31	77.92	60.60	57.07	58.48
ADJ_PHRASE	496	45.16	68.35	47.98	61.49	67.34
NUMERIC	330	16.36	63.33	13.94	64.85	66.06
CLAUSE	158	21.52	41.77	22.15	55.70	59.49
ABBREVIATION	102	74.51	85.29	74.51	44.12	66.67
OTHER	6	33.33	16.67	16.67	50.00	50.00
Kategória odpovede						
YES_NO	758	37.34	66.49	39.71	69.00	73.22
OTHER	757	36.86	61.03	39.10	61.82	66.31
DATETIME	612	52.61	75.00	48.04	57.52	56.70
ENTITY	610	48.03	69.51	52.13	69.07	72.50
PERSON	601	57.24	71.71	57.24	63.06	65.39
LOCATION	561	61.85	78.43	61.14	56.86	58.82
NUMERIC	333	16.52	63.36	14.41	65.47	66.67
ORGANIZATION	99	41.41	65.66	44.44	56.57	61.62
ABBREVIATION	96	76.04	85.42	76.04	42.71	66.67
DENOTATION	49	40.82	81.63	55.10	61.22	69.39

Tabulka 5.4: Tabuľka úspešnosti vyhľadávania vety v článku.

Systém	em[%]	nie[%]	áno[%]
Všetky prvky	760	245	515
Base	67.94	0	100.00
yn1, k: 1	72.03	25.51	93.98
yn2, k: 1	71.77	14.81	98.64
yn1, k: 3	69.39	20.58	92.43
yn2, k: 3	68.60	29.63	86.99
yn1, k: 5	67.94	0	100.00
yn2, k: 5	67.94	0	100.00
yn1, k: 3, 1	69.39	20.99	92.23
yn2, k: 3, 1	69.66	31.28	87.77
yn1, k: 5, 1	67.94	0	100.00
yn2, k: 5, 1	67.94	0	100.00
e1 + yn, k: 1	66.97	8.98	94.56
e2 + yn, k: 1	68.03	6.94	97.09
e3a + yn, k: 1	67.24	1.22	98.64
e3b + yn, k: 1	65.00	31.43	80.97
e6a + yn, k: 1	65.26	7.35	92.82
e6b + yn, k: 1	66.45	6.12	95.15
e6c + yn, k: 1	65.00	7.76	92.23
e1 + yn, k: 3	66.45	7.35	94.56
e2 + yn, k: 3	65.39	12.24	90.68
e3a + yn, k: 3	64.08	26.12	82.14
e3b + yn, k: 3	68.16	14.29	93.79
e6a + yn, k: 3	64.47	11.84	89.51
e6b + yn, k: 3	67.37	0	99.42
e6c + yn, k: 3	65.66	22.04	86.41
e1 + yn, k: 3, 1	66.58	11.84	92.62
e2 + yn, k: 3, 1	63.95	19.18	85.24
e3a + yn, k: 3, 1	60.13	34.69	72.23
e3b + yn, k: 3, 1	67.63	24.49	88.16
e6a + yn, k: 3, 1	61.18	25.71	78.06
e6b + yn, k: 3, 1	67.24	0	99.22
e6c + yn, k: 3, 1	64.08	28.57	80.97

Tabulka 5.5: Tabuľka úspešnosti odpovedania áno alebo nie.

	Base	yn1, k: 1	yn2, k: 1	yn1, k: 3	yn2, k: 3, 1	e2 + yn, k: 1	e3b + yn, k: 1	e6c + yn, k: 3	e1 + yn, k: 3, 1
Base	0	31	7	39	63	15	98	70	38
yn1, k: 1	62	0	27	50	70	73	118	104	87
yn2, k: 1	36	25	0	38	62	49	112	93	66
yn1, k: 3	50	30	20	0	67	61	121	100	80
yn2, k: 3, 1	76	52	46	69	0	83	119	102	87
e2 + yn, k: 1	17	44	22	52	72	0	100	76	44
e3b + yn, k: 1	77	66	62	89	85	77	0	92	74
e6c + yn, k: 3	54	57	48	73	73	58	97	0	68
e1 + yn, k: 3, 1	29	47	28	60	65	33	86	75	0

Tabulka 5.6: Porovnanie výsledkov medzi jednotlivými systémami pre odpovedanie "áno/-nie". V políčku mriežky je počet otázok, ktoré systém v riadku predikuje správne, ale systém v stĺpci nie.

	Prvkov	Base	yn1, k: 1[%]	yn1, k: 3[%]	e3b + yn, k: 1[%]	e3b + yn, k: 3[%]
Všetky	758	67.94	72.03	69.39	65.00	68.16
Kategória otázky						
VERB_PHRASE	748	67.65	71.79	69.12	65.24	68.05
ADJ_PHRASE	9	88.89	88.89	88.89	44.44	88.89
PERSON	1	100.00	100.00	100.00	100.00	100.00
Kategória odpovede						
YES_NO	758	67.94	72.03	69.39	65.04	68.34

Tabulka 5.7: Tabuľka úspešnosti odpovedania áno alebo nie.

Systém	em[%]	f1[%]
e1, k: 1	56.88	70.23
e2, k: 1	56.47	70.53
e3a, k: 1	46.86	60.80
e3b, k: 1	52.13	65.95
e6a, k: 1	49.16	62.82
e6b, k: 1	47.79	63.36
e6c, k: 1	57.48	69.91
e1, k: 3	53.51	65.76
e2, k: 3	49.70	60.70
e3a, k: 3	44.60	58.08
e3b, k: 3	49.37	62.28
e6a, k: 3	51.29	63.07
e6b, k: 3	51.43	62.85
e6c, k: 3	53.43	63.77
e1, k: 3, 1	56.19	69.17
e2, k: 3, 1	58.14	70.93
e3a, k: 3, 1	40.79	55.63
e3b, k: 3, 1	45.48	61.04
e6a, k: 3, 1	53.36	66.81
e6b, k: 3, 1	56.93	69.54
e6c, k: 3, 1	52.79	65.56
e1 + yn, k: 1	59.13	72.41
e2 + yn, k: 1	58.05	71.66
e3a + yn, k: 1	43.13	57.82
e3b + yn, k: 1	54.10	67.66
e6a + yn, k: 1	55.44	69.41
e6b + yn, k: 1	56.21	69.61
e6c + yn, k: 1	58.05	71.56
e1 + yn, k: 3	53.80	65.76
e2 + yn, k: 3	53.64	65.42
e3a + yn, k: 3	40.38	53.70
e3b + yn, k: 3	50.67	62.86
e6a + yn, k: 3	54.87	67.28
e6b + yn, k: 3	51.19	64.63
e6c + yn, k: 3	57.40	68.97
e1 + yn, k: 3, 1	56.54	69.32
e2 + yn, k: 3, 1	56.93	69.48
e3a + yn, k: 3, 1	31.91	48.43
e3b + yn, k: 3, 1	45.99	59.84
e6a + yn, k: 3, 1	54.70	67.26
e6b + yn, k: 3, 1	52.58	65.55
e6c + yn, k: 3, 1	50.60	64.45

Tabulka 5.8: Tabuľka úspešnosti extrakcie odpovede z textu.

	e1, k: 1	e6c, k: 1	e1, k: 3	e6b, k: 3, 1	e1 + yn, k: 1	e6c + yn, k: 1	e1 + yn, k: 3	e6c + yn, k: 3, 1
e1, k: 1	0	323	428	363	285	314	427	498
e6c, k: 1	345	0	457	352	274	267	447	485
e1, k: 3	308	315	0	335	289	324	322	442
e6b, k: 3, 1	365	332	457	0	321	322	428	432
e1 + yn, k: 1	365	332	489	399	0	305	463	478
e6c + yn, k: 1	355	286	485	361	266	0	451	474
e1 + yn, k: 3	316	314	331	315	272	299	0	417
e6c + yn, k: 3, 1	340	305	404	272	240	275	370	0

Tabulka 5.9: Porovnanie výsledkov medzi jednotlivými systémami pre extrakciu odpovede z textu. V políčku mriežky je počet otázok, ktoré systém v riadku predikuje správne, ale systém v stĺpci nie.

	Prvkov	e1, k: 1[%]	e1, k: 3[%]	e6c + yn, k: 1[%]	e6c + yn, k: 3[%]
Všetky	3641	56.88	53.51	58.05	57.40
Kategória otázky					
ENTITY	835	50.06	44.44	53.41	49.46
DATETIME	614	79.97	78.66	80.94	82.41
PERSON	577	66.38	57.69	69.32	66.67
LOCATION	549	61.20	56.65	58.47	58.47
ADJ_PHRASE	476	42.23	39.83	41.39	42.14
NUMERIC	316	46.52	53.16	46.84	51.27
CLAUSE	153	15.03	12.42	18.95	16.99
ABBREVIATION	102	70.59	70.59	74.00	74.00
VERB_PHRASE	13	0	0	0	0
OTHER	6	0	33.33	0	16.67
Kategória odpovede					
OTHER	738	35.91	33.29	37.26	35.86
DATETIME	611	80.20	79.05	81.34	82.82
ENTITY	602	52.49	46.52	54.15	51.82
PERSON	584	65.58	57.34	68.49	66.38
LOCATION	544	61.03	56.43	58.27	58.27
NUMERIC	319	46.71	53.29	46.39	51.10
ORGANIZATION	98	37.76	30.61	43.88	36.73
ABBREVIATION	96	73.96	75.00	78.72	75.53
DENOTATION	48	58.33	54.17	66.67	64.58

Tabulka 5.10: Tabuľka úspešnosti extrakcie odpovede z textu.

Systém	em1[%]	em5[%]
ak, t: o	30.99	39.43
ak, t: v	29.31	53.97
ak, t: o+v	53.10	76.15
ak, t: v → o	51.00	71.40
ac	48.13	67.92
tf.idf, e, u	40.18	69.45
tf.idf, e, u+b	48.65	74.95
tf.idf, n, e, u	32.11	62.58
tf.idf, n, e, u+b	50.08	77.78
tf.idf, np, u	31.89	59.19
tf.idf, np, u+b	35.32	62.69
tf.idf, n, np, u	35.17	65.14
tf.idf, n, np, u+b	39.54	69.60
ak, t: o+v, tf.idf, e, u	72.49	90.30
ak, t: o+v, tf.idf, e, u+b	76.71	93.07
ak, t: o+v, tf.idf, n, e, u	69.93	89.17
ak, t: o+v, tf.idf, n, e, u+b	76.86	94.05
ak, t: o+v, tf.idf, np, u	53.43	76.46
ak, t: o+v, tf.idf, np, u+b	53.66	76.59
ak, t: o+v, tf.idf, n, np, u	53.52	76.42
ak, t: o+v, tf.idf, n, np, u+b	53.79	76.59
ak, t: v → o, tf.idf, e, u	70.66	89.81
ak, t: v → o, tf.idf, e, u+b	75.68	92.80
ak, t: v → o, tf.idf, n, e, u	68.12	88.54
ak, t: v → o, tf.idf, n, e, u+b	76.55	93.80
ak, t: v → o, tf.idf, np, u	51.43	71.62
ak, t: v → o, tf.idf, np, u+b	51.54	71.87
ak, t: v → o, tf.idf, n, np, u	51.36	71.71
ak, t: v → o, tf.idf, n, np, u+b	51.60	71.96
ac, tf.idf, e, u	71.07	88.70
ac, tf.idf, e, u+b	74.70	91.48
ac, tf.idf, n, e, u	68.79	87.85
ac, tf.idf, n, e, u+b	75.86	93.45
ac, tf.idf, np, u	48.55	68.23
ac, tf.idf, np, u+b	48.71	68.37
ac, tf.idf, n, np, u	48.57	68.35
ac, tf.idf, n, np, u+b	48.80	68.59

Tabulka 5.11: Tabuľka úspešnosti výberu článku.

	ak, t: o	ak, t: v	ak, t: o+v	ak, t: o+v, tf.idf, n, e, u+b	ak, t: v → o	ak, t: v → o, tf.idf, n, e, u+b	ac	ac, tf.idf, n, e, u+b	tf.idf, e, u	tf.idf, n, e, u+b
ak, t: o	0	962	274	149	242	138	367	201	923	780
ak, t: v	887	0	308	131	353	144	515	182	675	541
ak, t: o+v	1266	1375	0	93	596	229	880	363	1364	1134
ak, t: o+v, tf.idf, n, e, u+b	2207	2264	1159	0	1370	305	1582	442	1807	1342
ak, t: v → o	1140	1326	502	210	0	108	823	350	1355	1118
ak, t: v → o, tf.idf, n, e, u+b	2182	2263	1281	291	1254	0	1577	413	1780	1300
ac	1136	1359	657	293	694	302	0	99	1285	1057
ac, tf.idf, n, e, u+b	2214	2270	1384	397	1465	382	1343	0	1763	1280
tf.idf, e, u	1336	1163	785	162	870	149	929	163	0	297
tf.idf, n, e, u+b	1637	1473	999	141	1077	113	1145	124	741	0

Tabulka 5.12: Porovnanie výsledkov medzi jednotlivými systémami pre vybranie článku s odpoveďou. V políčku mriežky je počet otázok, ktoré systém v riadku predikuje správne, ale systém v stĺpci nie.

	Prvkov	ak, t: o+v[%]	ac[%]	tf.idf, e, u[%]	ac, tf.idf, n, e, u+b[%]	ak, t: o+v, tf.idf, n, e, u+b[%]
Všetky	4486	53.10	48.13	40.18	50.08	76.86
Kategória otázky						
ENTITY	849	40.87	40.75	43.18	54.94	73.85
VERB_PHRASE	765	57.39	52.16	49.41	62.88	82.75
DATETIME	618	66.83	67.48	35.44	42.39	79.29
PERSON	595	40.50	37.31	38.15	51.09	75.63
LOCATION	566	63.43	55.65	31.63	39.40	78.27
ADJ_PHRASE	497	50.91	39.84	46.88	52.11	73.84
NUMERIC	330	57.88	44.24	34.85	42.42	76.06
CLAUSE	158	51.27	34.18	39.87	48.10	73.42
ABBREVIATION	102	52.94	58.82	18.63	31.37	65.69
OTHER	6	66.67	33.33	50.00	50.00	66.67
Kategória odpovede						
YES_NO	763	57.40	53.08	49.80	63.04	82.96
OTHER	758	49.34	38.39	43.27	50.79	73.88
DATETIME	615	66.83	67.64	35.28	42.28	79.35
ENTITY	610	41.31	39.18	44.68	55.16	73.77
PERSON	602	40.86	37.38	37.71	50.33	75.25
LOCATION	561	62.92	55.44	31.55	39.04	77.72
NUMERIC	333	57.96	44.44	35.44	42.94	76.28
ORGANIZATION	99	44.44	44.44	40.40	62.63	75.76
ABBREVIATION	96	52.08	61.46	17.71	30.21	64.58
DENOTATION	49	42.86	42.86	53.06	57.14	75.51

Tabulka 5.13: Tabuľka úspešnosti vybranie článku s odpoveďou.

					k	a	s	rs	Skóre	em[%]
v1	ak, t: o+v	r2	yn1, k: 1	e1 + yn, k: 1	1	1	1	1	0:0:1	33.99
v1	ak, t: o+v	r2	yn1, k: 1	e2, k: 3	1	1	1	1	0:0:1	33.43
v1	ak, t: o+v	r2	e3b + yn, k: 3	e1 + yn, k: 1	1	1	1	1	0:0:1	33.76
v1	tf.idf, n, e, u+b	r2	yn1, k: 1	e1 + yn, k: 1	0	1	1	1	0:0:1	33.70
v1	ak, t: o+v, tf.idf, n, e, u+b	r2	yn1, k: 1	e1 + yn, k: 1	0	1	1	1	0:0:1	44.22
v1	ak, t: o+v, tf.idf, n, e, u+b	r1	yn1, k: 1	e1 + yn, k: 1	0	1	1	1	0:0:1	42.88
v1	ak, t: o+v, tf.idf, n, e, u+b	tf.idf, e, u	yn1, k: 1	e1 + yn, k: 1	0	1	1	1	0:0:1	46.67
v1	ak, t: o+v, tf.idf, n, e, u+b	tf.idf, n, e, u	yn1, k: 1	e1 + yn, k: 1	0	1	1	1	0:0:1	46.67
v1	ak, t: o+v, tf.idf, n, e, u+b	tf.idf, n, e, u+b	yn1, k: 1	e1 + yn, k: 1	0	1	1	1	0:0:1	46.36
v1	ak, t: o+v, tf.idf, n, e, u+b	r2	yn1, k: 1	e1 + yn, k: 1	0	5	5	25	0:0:1	19.92
v1	ak, t: o+v, tf.idf, n, e, u+b	tf.idf, e, u	yn1, k: 1	e1 + yn, k: 1	0	5	5	25	0:0:1	43.93
v1	ak, t: o+v, tf.idf, n, e, u+b	r2	yn1, k: 1	e1 + yn, k: 1	0	3	3	9	0:0:1	27.57
v1	ak, t: o+v, tf.idf, n, e, u+b	tf.idf, e, u	yn1, k: 1	e1 + yn, k: 1	0	3	3	9	0:0:1	45.31
v1	ak, t: o+v, tf.idf, n, e, u+b	r2	yn1, k: 1	e1 + yn, k: 1	0	5	5	5	0:0:1	33.10
v1	ak, t: o+v, tf.idf, n, e, u+b	tf.idf, e, u	yn1, k: 1	e1 + yn, k: 1	0	5	5	5	0:0:1	45.46
v1	ak, t: o+v, tf.idf, n, e, u+b	r2	yn1, k: 1	e1 + yn, k: 1	0	3	3	5	0:0:1	34.05
v1	ak, t: o+v, tf.idf, n, e, u+b	tf.idf, e, u	yn1, k: 1	e1 + yn, k: 1	0	3	3	5	0:0:1	45.62
v1	ak, t: o+v, tf.idf, n, e, u+b	r2	yn1, k: 1	e1 + yn, k: 1	3	1	1	1	0:0:1	39.63
v1	ak, t: o+v, tf.idf, n, e, u+b	tf.idf, e, u	yn1, k: 1	e1 + yn, k: 1	3	1	1	1	0:0:1	41.21
v1	ak, t: o+v, tf.idf, n, e, u+b	r2	yn1, k: 1	e1 + yn, k: 1	3	5	5	5	0:0:1	31.62
v1	ak, t: o+v, tf.idf, n, e, u+b	tf.idf, e, u	yn1, k: 1	e1 + yn, k: 1	3	5	5	5	0:0:1	40.07
v1	ak, t: o+v, tf.idf, n, e, u+b	r2	yn1, k: 1	e1 + yn, k: 1	3	3	3	5	0:0:1	30.91
v1	ak, t: o+v, tf.idf, n, e, u+b	tf.idf, e, u	yn1, k: 1	e1 + yn, k: 1	3	3	3	5	0:0:1	40.47
v1	ak, t: o+v, tf.idf, n, e, u+b	r2	yn1, k: 1	e1 + yn, k: 1	1	3	3	9	0:1:1	27.12
v1	ak, t: o+v, tf.idf, n, e, u+b	tf.idf, e, u	yn1, k: 1	e1 + yn, k: 1	1	3	3	9	0:1:1	37.51
v1	ak, t: o+v, tf.idf, n, e, u+b	r2	yn1, k: 1	e1 + yn, k: 1	3	3	3	9	0:1:1	26.61
v1	ak, t: o+v, tf.idf, n, e, u+b	tf.idf, e, u	yn1, k: 1	e1 + yn, k: 1	3	3	3	9	0:1:1	34.41
v1	ak, t: o+v, tf.idf, n, e, u+b	r2	yn1, k: 1	e1 + yn, k: 1	1	3	3	9	1:1:1	32.52
v1	ak, t: o+v, tf.idf, n, e, u+b	tf.idf, e, u	yn1, k: 1	e1 + yn, k: 1	1	3	3	9	1:1:1	37.51
v1	ak, t: o+v, tf.idf, n, e, u+b	r2	yn1, k: 1	e1 + yn, k: 1	3	3	3	9	1:1:1	31.38
v1	ak, t: o+v, tf.idf, n, e, u+b	tf.idf, e, u	yn1, k: 1	e1 + yn, k: 1	3	3	3	9	1:1:1	34.41

Tabulka 5.14: Tabuľka úspešnosti kombinácie sietí. Prvé políčka tabuľky určujú zostavenie systému na predikciu. k – šírka kontextu pre získanie odpovede. a – počet vybraných článkov. s – počet viet na článok. rs – počet článkov po redukcií. Skóre – váhy pre jednotlivé skóre (článok:veta:odpoveď).

Kapitola 6

Záver

Systém bol rozdelený na časti:

1. V dátovej sade sú dva typy odpovedí, ktoré sa správajú dostatočne rozdielne na to aby bolo správanie upravené podľa typu odpovede. Odpoveď môže byť "áno/nie"alebo časť vybraná z textu. Rozdeľujem iba tieto dva typy odpovede, pretože sa správajú rozlične. Rozdeľovať či je odpoveď dátum alebo meno osoby nie je potrebné, pretože je to stále časť textu.

2. Najskôr je potrebné k otázke priradiť článok z Wikipédie, v ktorom sa predpokladá existenciu odpovede. V práci som zmenšil Wikipédiu na 7193 článkov, ktoré sa v dátovej sade používajú. Prejsť celú Wikipédiu vlastnou neurónovou sieťou alebo mnou špecifikovaným algoritmom je nereálne. Zároveň Wikipédia sa mení. Viem že niektoré odpovede už v najnovšej verzií Wikipédie niesu, aj napriek tomu že tam v minulosti boli.

3. Pri predikovaní odpovede najskôr zmenšujem kontext predikcie. Hľadať odpoveď v celom článku na jeden krok je náročný problém, a preto je ho dobré rozdeliť na časti. V článku sa najskôr predikuje veta, v ktorej sa očakáva odpoveď.

4. Pri odpovedaní "áno"alebo "nie"sa kombinácia otázky a predom získaného kontextu rozdelí do dvoch kategórií. Jedna bude určovať odpoveď "áno"a druhá bude určovať odpoveď "nie".

5. Pri extrakcii krátkej odpovede z textu sa vyberá celistvý kus textu výberom začiatku a konca odpovede. Je možné predpovedať odpoveď aj iným spôsobom, ale predpovedanie začiatku a konca odpovede je najpopulárnejšie [4, 12].

Rozdelenie problému na menšie časti rozdelí počet vlastností, ktoré sa má neurónová sieť naučiť, a tým zlepší presnosť. Keď som učil jeden systém riešiť bod 1 a 4, tak dosahoval horšiu presnosť ako pri rozdelení problému, pretože sa jedná o zásadne rozličné vlastnosti. Bod 3 je dôležité oddeliť od samotného získanie odpovede. Experimentálne som zistil že zmenšenie kontextu pred hľadaním odpovede zlepší presnosť. Aj keď sa odpoveď môže stratiť pri izolovaní kontextu, tak je pravdepodobne zložitá a stratila bi sa aj pri predikovaní odpovede.

Aj keď by bolo problematické riešiť celý problém algoritmom, niektoré časti sa dajú vylepšiť použitím algoritmu tf.idf. Časti 2 a 3 sa dajú riešiť algoritmom tf.idf, pretože ide

o priradenie otázky súboru dokumentov. Pri priraďovaní článku otázke, súbor dokumentov sú texty článkov. Pri vyberaní vety z článku, súbor dokumentov sú jednotlivé vety článku.

Rozdelením problému na časti a kombinovaním algoritmov a neurónových sietí som dosiahol lepšiu presnosť než sa dá dosiahnuť samotnou neurónovou sieťou, ale stále by malo byť možné získať lepšie výsledky. Odpovedanie na otázky je stále skúmaná oblasť, ktorá nemá optimálne riešenie. Pri testovaní neurónových sietí majú komplikované neurónové siete horšie výsledky ako jednoduchšie neurónové siete. Môže to byť spôsobené malou dátovou sadou, pretože v literatúre majú jednoduché aj komplexné systémy dobré výsledky. Kombinácia neurónovej siete a algoritmu je aspekt, ktorý som skúmal, a zlepšuje presnosť výsledného systému. Normalizáciu textu som robil iba pri algoritme tf.idf a nerobil som ju pri neurónových sietiach. Normalizácia textu môže zmeniť význam textu. Narozdiel od algoritmu tf.idf, neurónové siete sa snažia text pochopiť. Normalizácia podstatného mena by mala mať menší vplyv na význam vety ako normalizácia slovesa. Zrušením času slovies sa môže stratiť časť faktu vo vete. Tvar ukazovacieho zámena určuje identitu predmetu z predchádzajúceho kontextu, a preto normalizovaním zámena sa môže stratiť kontext vety. Keď sa začne normalizovať text pre neurónové siete, vznikne veľké množstvo skúmateľných teórií, ktoré môžu zmeniť výsledky predikcie. Jeden z aspektov, ktorý som neriešil a mohol by zlepšiť presnosť, je manipulácia a optimalizácia algoritmu samotného učenia.

Literatura

[1] Andreas, J., Rohrbach, M., Darrell, T. a Klein, D. Learning to Compose Neural Networks for Question Answering. *CoRR*. 2016, abs/1601.01705. Dostupné z: http://arxiv.org/abs/1601.01705.

[2] Das, R., Dhuliawala, S., Zaheer, M. a McCallum, A. Multi-step Retriever-Reader Interaction for Scalable Open-domain Question Answering. *CoRR*. 2019, abs/1905.05733. Dostupné z: http://arxiv.org/abs/1905.05733.

[3] Di Gennaro, G., Buonanno, A., Di Girolamo, A. a Palmieri, F. A. Intent Classification in Question-Answering Using LSTM Architectures. *ArXiv preprint arXiv:2001.09330*. 2020.

[4] Fajčík, M. *BISSIT 19 NLP excercise QA student*. Cvičenie dostupné pre študentov VUT FIT.

[5] Iyyer, M., Boyd Graber, J., Claudino, L., Socher, R. a Daumé III, H. A Neural Network for Factoid Question Answering over Paragraphs. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, říjen 2014, s. 633–644. DOI: 10.3115/v1/D14-1070. Dostupné z: https://www.aclweb.org/anthology/D14-1070.

[6] Joulin, A., Grave, E., Bojanowski, P. a Mikolov, T. Bag of tricks for efficient text classification. *ArXiv preprint arXiv:1607.01759*. 2016.

[7] Kaushik, D. a Lipton, Z. C. How much reading does reading comprehension require? a critical investigation of popular benchmarks. *ArXiv preprint arXiv:1808.04926*. 2018.

[8] Kumar, A., Irsoy, O., Ondruska, P., Iyyer, M., Bradbury, J. et al. Ask me anything: Dynamic memory networks for natural language processing. In: *International conference on machine learning*. 2016, s. 1378–1387.

[9] Levy, O. a Goldberg, Y. Dependency-based word embeddings. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2014, s. 302–308.

[10] Lilleberg, J., Zhu, Y. a Zhang, Y. Support vector machines and word2vec for text classification with semantic features. In: IEEE. *2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*. 2015, s. 136–140.

[11] Mahabadi, R. K. a Henderson, J. Simple but effective techniques to reduce biases. *ArXiv preprint arXiv:1909.06321*. 2019.

[12] Manning, C. a Lamm, M. *CS224n Natural Language Processing*. Prednášky Stanford University School of Engineering. Dostupné z: http://web.stanford.edu/class/cs224n/.

[13] Markoff, J. Computer Wins on 'Jeopardy!': Trivial, It's Not. *The New York Times*. 2011. Dostupné z: https://www.nytimes.com/2011/02/17/science/17jeopardy-watson.html.

[14] Medveď, M. a Horák, A. *Sqad 3.0*. 2019. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University. Dostupné z: http://hdl.handle.net/11234/1-3069.

[15] Pastorek, P. *Strojové učení pro odpovídání na otázky v češtině*. Brno, 2020. Semestrální projekt. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. RNDr. Pavel Smrž, Ph.D.

[16] Rajpurkar, P., Zhang, J., Lopyrev, K. a Liang, P. Squad: 100,000+ questions for machine comprehension of text. *ArXiv preprint arXiv:1606.05250*. 2016.

[17] Ramos, J. et al. Using tf-idf to determine word relevance in document queries. In: Piscataway, NJ. *Proceedings of the first instructional conference on machine learning*. 2003, sv. 242, s. 133–142.

[18] Salton, G. a Buckley, C. *Term weighting approaches in automatic text retrieval*. Cornell University, 1987.

[19] Sarrouti, M. a El Alaoui, S. O. SemBioNLQA: A semantic biomedical question answering system for retrieving exact and ideal answers to natural language questions. *Artificial Intelligence in Medicine*. Elsevier. 2020, sv. 102, s. 101767.

[20] Serban, I. V., García Durán, A., Gulcehre, C., Ahn, S., Chandar, S. et al. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. *ArXiv preprint arXiv:1603.06807*. 2016.

[21] Severyn, A. a Moschitti, A. Automatic feature engineering for answer selection and extraction. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 2013, s. 458–467.

[22] Strzalkowski, T. a Harabagiu, S. *Advances in open domain question answering*. Springer Science & Business Media, 2006.

[23] Sun, H., Bedrax-Weiss, T. a Cohen, W. W. PullNet: Open Domain Question Answering with Iterative Retrieval on Knowledge Bases and Text. *CoRR*. 2019, abs/1904.09537. Dostupné z: http://arxiv.org/abs/1904.09537.

[24] Tan, M., Xiang, B. a Zhou, B. LSTM-based Deep Learning Models for non-factoid answer selection. *CoRR*. 2015, abs/1511.04108. Dostupné z: http://arxiv.org/abs/1511.04108.

[25] Ture, F. a Jojic, O. No Need to Pay Attention: Simple Recurrent Neural Networks Work!(for Answering"Simple"Questions). *ArXiv preprint arXiv:1606.05029*. 2016.

[26] Wang, D. a Nyberg, E. A long short-term memory model for answer sentence selection in question answering. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. 2015, s. 707–712.

[27] Wang, M., Smith, N. A. a Mitamura, T. What is the Jeopardy model? A quasi-synchronous grammar for QA. In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. 2007, s. 22–32.

[28] Wang, S., Yu, M., Jiang, J., Zhang, W., Guo, X. et al. Evidence aggregation for answer re-ranking in open-domain question answering. *ArXiv preprint arXiv:1711.05116*. 2017.

[29] Wu, H. C., Luk, R. W. P., Wong, K. F. a Kwok, K. L. Interpreting tf-idf term weights as making relevance decisions. *ACM Transactions on Information Systems (TOIS)*. ACM New York, NY, USA. 2008, sv. 26, č. 3, s. 1–37.

[30] Xiong, C., Merity, S. a Socher, R. Dynamic memory networks for visual and textual question answering. In: *International conference on machine learning*. 2016, s. 2397–2406.

[31] Yu, L., Hermann, K. M., Blunsom, P. a Pulman, S. Deep learning for answer sentence selection. *ArXiv preprint arXiv:1412.1632*. 2014.