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

KOMUNIKAČNÍ AGENT PRO INFORMACE O BRNĚ

BRNO COMMUNICATION AGENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **JURAJ JURKOVIČ**
AUTHOR

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

BRNO 2018

Abstrakt

Cieľom tejto práce je preskúmať a následne aplikovať techniky a technické riešenia pri vývoji informačných agentov. Práca sa zameriava na riešenia jednotlivých podproblémov pomocou existujúcich systémov, prepojenie týchto systémov, jejich prispôsobenie pre danú doménu a implementáciu jednotlivých modulov. Uživatelské rozhranie je postavené na multiplatformnej chatovacej aplikácii Telegram. Extrakciu informáciu zo vstupu uživatela vykonáva služba Dialogflow. Pre uspokojenie požiadavky uživatela je použitých niekoľko externých služieb. Pre vyhladavanie v strukturovaných dátach je použitá technológia Elasticsearch. Pre extrakciu odpovedí z volného textu je použitý system R-net. Vysledkom je systém ktorého znalostnú bázu, ako aj množinu dotazov ktore je schopný uspokojiť, možno jednoducho rozšíriť a ktorý môže byť nasadený na ľubovolnú chatovaciu platformu.

Abstract

The goal of this thesis is explore and subsequently apply techniques and technical solutions in development of information agents. Thesis primarily focuses on solving individual sub tasks using state of the art systems, interconnecting these systems, their adoption for specific domain and implementation of individual modules of communication agent system. User interface is based on multi-platform chat application Telegram. Information extraction from user input is executed by Dialogflow. Several external services are used for user request fulfillment. Elasticsearch is used for searching structured data. For answering open domain questions from free text we use R-net implementation. The resulting can have both ,its knowledge base and range of requests it can fulfill, easily extended and can be deployed to chat platform of choice.

Kľúčové slová

rozpoznávanie pomenovaných entít,rozpoznávanie entít, odpovedanie na otázky, extrakcia informácií, umelá inteligencia, strojové učenie, četboti, zistovanie zámeru, dialogové systémy, informační agenti

Keywords

named entity recognition, entity recognition, question answering, information extraction, artificial intelligence, machine learning, chatbots, intent classification, dialogue systems, information agents

Citácia

JURKOVIČ, Juraj. *Komunikační agent pro informace o Brně*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. RNDr. Pavel Smrž, Ph.D.

Komunikační agent pro informace o Brně

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána doc. Smrža. Uviedol som všetky literárne zdroje a publikácie, z ktorých som čerpal.

.....
Juraj Jurkovič
18. mája 2018

Poďakovanie

Ďakujem vedúcemu práce doc. Smržovi za jeho trpezlivosť a ochotu a svojej sestre Michaele za to že ma do poslednej chvíle podporovala.

Obsah

Kapitola 1

Úvod

Cieľom tejto práce je preskúmať a následne aplikovať techniky a technické riešenia pri vývoji informačných agentov. Práca sa zameriava na riešenia jednotlivých podproblémov pomocou existujúcich systémov, prepojenie týchto systémov, ich prispôsobenie pre danú doménu a implementáciu jednotlivých komponent. Uživateľské rozhranie je postavené na multiplatformnej chatovacej aplikácii Telegram. Extrakciu informácii zo vstupu užívateľa vykonáva služba Dialogflow. Pre uspokojenie požiadavky užívateľa je použitých niekoľko externých služieb. Pre vyhľadávanie v štruktúrovaných dátach je použitá technológia Elasticsearch. Pre extrakciu odpovedí z voľného textu je použitý systém R-net. Práca demonštruje prepojenie jednotlivyých systémov do funkčného celku, vytvorenie doménovo špecifických znalostných bázi pre jednotlive podsystémy. Následne tieto riešenia a systémy aplikuje na konkrétnu doménu ktorou je v tomto prípade poskytovanie turistických informácii o Brne.

Práca skúma oblasti extrakcie informácii z textu v prirodzenom neštruktúrovanom jazyku, hľadanie odpovede na otázku vo voľnom texte a extrakcie relevantných dokumentov z korpusu. Výsledkom je systém ktorého znalostnú bázu, ako aj množinu dotazov ktoré je schopný uspokojiť, možno jednoducho rozšíriť a ktorý môže byť nasadený na ľubovolnú chatovaciu platformu.

Kapitola 2

Úvod do problematiky

Hlavné výzvy s ktorými sa komunikačný agent musí vysporiadať je porozumenie dotazu a vhodná reakcia na tento dotaz.

2.1 Porozumenie prirodzenému jazyku

2.1.1 Rozpoznávanie entít

Pomenovaná entita je postupnosť slov, ktoré označujú určitý objekt reálneho sveta napríklad „Slovensko“, „Steve Jobs“, „Microsof“ a podobne. Úlohou rozpoznania pomenovaných entít (ang. named entity recognition) je identifikácia pomenovaných entít vo voľnom texte a ich zaradenie do množiny preddefinovaných typov, ako sú osoba, organizácia alebo lokácia. Často krát táto úloha nemôže byť jednoducho splnená porovnaním reťazcov vopred zostavených slovníkov, pretože pomenované entity daného typu zvyčajne netvoria uzavretý súbor, a preto by bol akýkoľvek slovník neúplný. Ďalším dôvodom je, že typ pomenovanej entity môže byť závislý od kontextu. Napríklad, „JFK“ môže odkazovať na osobu „John F. Kennedy“, lokáciu „Medzinárodné Letisko JFK“, alebo iný subjekt zdieľajúci rovnakú skratku. Pre určenie typu entity výrazu „JFK“, ktorý sa vyskytuje v konkrétnom dokumente, musí byť braný v úvahu jeho kontext.

Rozpoznávanie pomenovaných entít je pravdepodobne najdôležitejšou úlohou v oblasti získavania informácií. Extrakcia zložitejších štruktúr ako napr relácie a udalosti závisia od presného rozpoznania entít. Rozpoznávanie pomenovaných entít , okrem toho, že je základným kameňom pri extrakcii informácií, má mnoho ďalších využití. Napríklad v odpovedaní na otázky sú kandidátne odpovede často pomenovanými entitami, ktoré je potrebné najprv získať a klasifikovať [14]. V entitovo orientovanom vyhľadávaní je identifikácia pomenovaných entít v dokumentoch ako aj v dotazoch je prvý krok smerom k vysokej relevancii výsledkov vyhľadávania [13], [8].

Hoci štúdie o rozpoznaní pomenovaných entít sa datujú od začiatku 90. rokov [22], úloha bola formálne predstavená v roku 1995 na konferencii Message Understanding Conference (MUC-6) ako podúloha získavania informácií [10]. Od tej doby rozpoznaniu pomenovaných entít venuje výskumná komunita veľkú pozornosť. Na túto úlohu bolo vytvorených niekoľko vyhodnocovacích programov.

Najčastejšie študované typy pomenovaných entít sú osoby, organizácie a lokácie, ktoré boli prvýkrát definované MUC-6. Tieto typy sú dostatočne všeobecné na to, aby boli užitočné pre viacero aplikačných domén. Extrakcia vyjadrenia dátumov, časov, peňažných hodnôt a percentuálnych hodnôt, ktoré boli tiež uvedené na MUC-6, je tiež často študovaná

v rámci rozpoznávania pomenovaných entít, aj keď striktne vzaté, sa nejedná o pomenované entity. Okrem týchto všeobecných typov entít, sú pre konkrétne domény a aplikácie zvyčajne definované ďalšie typy. Napríklad korpus GENIA používa a jemnozrnnú ontológiu na klasifikáciu biologických entít [19]. V on-line vyhľadávaní a reklame sa zasa často stretávame s ťažením názvov produktov.

Prvotné riešenia pre rozpoznávanie pomenovaných entít sa spoliehali na ručne vytvorené vzory. Nakoľko sa jedná o časovo náročnú činnosť ktorá vyžaduje odbornú spôsobilosť, novšie systémy sa pokúšajú o naučenie týchto vzorov z anotovaných dát. Novšie systémy pre rozpoznanie pomenovaných entít používajú štatistické metódy strojového učenia. Tieto systémy využívajú modely ako sú skryté Markovove modely, modely maximálnej entropie, Markovove modely s maximálnou entropiou a podmienené náhodne polia (ang. conditional random fields, CRF)[15] [9] [4] [12]

2.2 Odpovedanie na otázky

Odpovedanie na otázky (ang. question answering, dalej QA) je oblasťou umelej inteligencie, ktorá sa zaoberá vývojom systémov a technik, ktoré odpovedajú na otázky zadané v prirodzenom jazyku.

2.2.1 Otázka

Jedna z definícií slova otázka by mohlo byť 'požiadavka informácie'. Ako však takú požiadavku rozpoznáme? V písomnej forme jazyka sa často spoliehame na otázniky, ktoré indikujú to, že veta predstavuje otázku. Táto indikácia však môže byť zavádzajúca. Rečnícke otázky sú často ukončené otáznikom aj napriek tomu, že nevyžadujú odpoveď, zatiaľ čo výrazy ktoré požadujú nejakú informáciu nemusia byť formulované ako otázky.

Napríklad otázka „Ktorí politici kandidujú na prezidenta?“ by mohla byť napísaná aj ako výraz „Vymenuj prezidentských kandidátov“.

Obe žiadajú rovnaké informácie, ale jedna je formulovaná ako otázka a jedna ako príkaz. Ľudia dokážu ľahko vyhodnotiť tieto rôzne výrazy, pretože máme tendenciu sústrediť sa na význam (sémantiku) výrazu a nie na presnú formuláciu (syntax). Môžeme teda využiť plnú vyjadrovaciu schopnosť jazyka pretože vieme, že človek ktorého sa otázku pýtame, otázke porozumie a bude na ňu schopný zareagovať.

Hoci existuje niekoľko rôznych foriem otázok, táto práca je primárne zameraná na faktické otázky a definičné otázky. Faktické otázky sú tie, pre ktoré je odpoveďou jediný fakt. Napríklad „Kedy sa narodil Miloš Zeman“,„Kedy bola postavená Eiffelova veža“ a „Kde sídli spoločnosť Microsoft?“ sú všetko príklady faktických otázok.

Existuje veľa otázok, ktoré sa nepovažujú za faktografické otázky a na ktoré sa táto práca nevzťahuje. Patria sem otázky, ktoré môžu mať odpovede áno / nie, ako napríklad „Žije na Slovensku viac ľudí ako v Českej Republike?“,

ako aj otázky založené na postupnosti inštrukcií (napr. „Ako sadiť jahody?“) Existuje aj ďalší typ otázok, ktoré budeme nazývať zoznamové otázky (ang. list questons), ktorý úzko súvisí s faktickými otázkami. Zoznamové otázky sú faktické otázky, ktoré vyžadujú viac ako jednu odpoveď.

Napríklad „Aké hrozno sa používa pri výrobe vína?“ Je zoznamová otázka.

Zatiaľ čo zoznamové otázok nebudú obsiahnuté v tejto práci, väčšina prístupov k odpovedi na faktickú otázku môže byť použitá aj na odpoveď na zoznamovú otázku.

Definičné otázky, na rozdiel od faktických otázok, vyžadujú komplexnejšiu odpoveď, zvyčajne vytvorenú z viacerých zdrojových dokumentov. Odpoveďou by mal byť krátky odsek ktorý stručne definuje definendum (vec - či už ide o osobu, organizáciu, objekt alebo udalosť, často označovaná ako cieľ), o ktorej si užívateľ želá vedieť viac. Dobrá odpoveď na definičnú otázku by mala mať podobnú formu ako paragraf z encyklopédie. Napríklad, ak sa otázka pýta na osobu, potom používateľa budú pravdepodobne zaujímať dôležité dátumy v ich živote (narodenie, manželstvo a smrť), ich hlavné úspechy a ďalšie zaujímavosti. Pre otázky o organizáciách by mala odpoveď obsahovať informácie o jej náplni, kedy a kým bola založená počet zamestnancov a ďalšie zaujímavé fakty podľa povahy organizácie.

2.2.2 Odpoveď

Ak je otázka žiadosťou o informácie a odpovede sú poskytnuté v reakcii na otázky, potom odpovede musia byť reakciami na žiadosti o informácie. Ale čo robí odpoveď odpoveďou? Takmer každý výrok môže byť odpoveďou na na nejakú otázku a rovnakým spôsobom, ako môže existovať mnoho rôznych spôsobov, ako vyjadriť tú istú otázku, existuje mnoho spôsobov, ako opísať tú istú odpoveď.

Napríklad každá otázka, ktorej odpoveď je číselná, môže mať odpoveď vyjadrenú nekonečným počtom spôsobov. Zatiaľ čo môže existovať niekoľko spôsobov, ako vyjadriť správnu odpoveď na otázku, nie všetky splnia potreby osoby ktorá sa pýta. Napríklad, ako by na otázku „Kde sa nachádza múzeum Ľudovíta Štúra?“ bola odpoveď „V Modre“, aj napriek tomu že to je správna odpoveď, bude užitočná len pre niekoho, kto vie, kde je Modra - na západe Slovenska.

Na účely tejto práce budeme predpokladať, že užívateľ je priemerne inteligentný a ovláda anglický jazyk na komunikačnej úrovni.

2.2.3 Proces odpovedania na otázky

Ak uvažujeme o odpovedi na otázky o ľudskej činnosť, čo očakávame keď osoba ktorú žiadame o odpoveď na otázku nevie odpoveď? V takomto prípade je možné, že osoba, ktorej otázka bola položená, by sa pokúsila informáciu dohľadať v nejakej znalostnej báze (kniha, knižnica, internet ...) s cieľom nájsť nejaký text, ktorý by si mohla prečítať a pochopiť, čo by im umožnilo určiť odpoveď na otázku. Oni by potom tomu, kto pôvodne položil otázku, túto odpoveď mohli predať. Mohli by tiež uviesť, kde našli odpoveď, ktorá by v očiach osoby, ktorá kládla otázku, zvýšila dôveryhodnosť odpovede.

To, čo by osoba nerobila v prípade že by odpoveď nepoznala, je že by jednoducho odovzdala knihu alebo niekoľko dokumentov, o ktorých si mysli, že by mohli obsahovať odpoveď na otázku, ktorú osoba položila. To je to, s čím sa väčšina používateľov Internetu v tejto chvíli musí uspokojiť. Mnohí ľudia by chceli používať Internet ako zdroj vedomostí, v ktorom by mohli nájsť odpovede na ich otázky. Aj keď mnoho vyhľadávačov naznačuje, že sa môžete opýtať otázky v prirodzenom jazyku, výsledky, ktoré vrátia, sú zvyčajne časti dokumentov ktoré môžu a nemusia obsahovať odpoveď, ale ktoré majú veľa spoločných slov s otázkou. Je to preto, lebo odpovedanie na otázky vyžaduje oveľa hlbšie pochopenie a spracovanie textu, než ktoré v súčasnosti väčšina webových vyhľadávacích nástrojov vykonáva.

Výzvy v QA

Čo teda robí odpovedanie na otázky náročným? Takže. Čo robia ľudia pri čítaní textu, aby našli odpoveď na otázku? Ako musia byt počítače naprogramované aby dokázali fungovať podobným spôsobom?

Je často veľa spôsobov, ako požiadať o rovnaké informácie. Niektoré variácie sú jednoduché preformovanie otázky a inokedy sú variácie závislé od kontextu, v ktorom sa otázka kladie, alebo znalosti osoby ktorej sa pýtame.

Podobne existuje mnoho spôsobov, ako opísať rovnakú odpoveď, a tak kontext a znalosti používateľa môžu zohrávať úlohu v spôsobe definovania odpovedí. Napríklad odpoveď „minulý utorok“ neznamená nič bez vedomia aktuálneho dátumu alebo dátumu, kedy bola odpoveď napísaná. Nielenže to sťažuje systému, zisťovanie či sú dve odpovede rovnocenné (na účely zjednocovanie a hodnotenia odpovedí), ale aj vyhodnocovanie (ang. evaluation) výstupu systémov QA je problematické.

Skutočnosť, že aj otázky aj odpovede môžu byť napísané mnohými rôznymi spôsobmi, stažuje získanie dokumentov, ktoré obsahujú informácie o téme danej otázky. Napríklad odpoveď na otázku „Kto je prezident na Slovensku“ môže byť veta ako „Andrej Kiska je slovenský prezident“ v rámci dokumentu. Otázka a odpoveď by však mohli byt napísané aj ako „Kto je slovenský prezident“ a „Na Slovensku je prezident Andrej Kiska“. Tento príklad nielenže ukazuje, že otázky a odpovede môžu byť zapísané viacerými spôsobmi, ale aj to, že pokiaľ otázka a odpoveď nie sú vyjadrené podobným spôsobom, medzi otázkou a odpoveďou môže byť veľmi málo spoločného (v tomto prípade len slovo „prezident“ je spoločné pre rôzne formy), čo veľmi sťažuje získavanie dokumentov obsahujúcich odpoveď.

Aj keď otázky ako „Čo je to aspirín?“ (zvyčajne označované ako definičné otázky) vykazujú malú variáciu, ich odpovede sú zvyčajne rozmanitejšie ako pri otázkach, ktoré zaujíma jedna skutočnosť, napríklad meno osoby alebo dátum narodenie. Získanie relevantných dokumentov pre takéto otázky je mimoriadne náročné. Jediné slová ktoré by mohli obmedziť vyhľadávanie na vhodné dokumenty sú zvyčajne práve slová z výrazu ktorého definíciu hľadáme a tieto slová sa často používajú v dokumente bez toho, aby boli predtým v dokumente definované. To znamená, že veľa dokumentov, v ktorých sa hľadaný výraz nachádza, nebude pre systém, ktorý sa pokúša vytvoriť definíciu, nijako užitočný.

Ako odpovedať na otázky

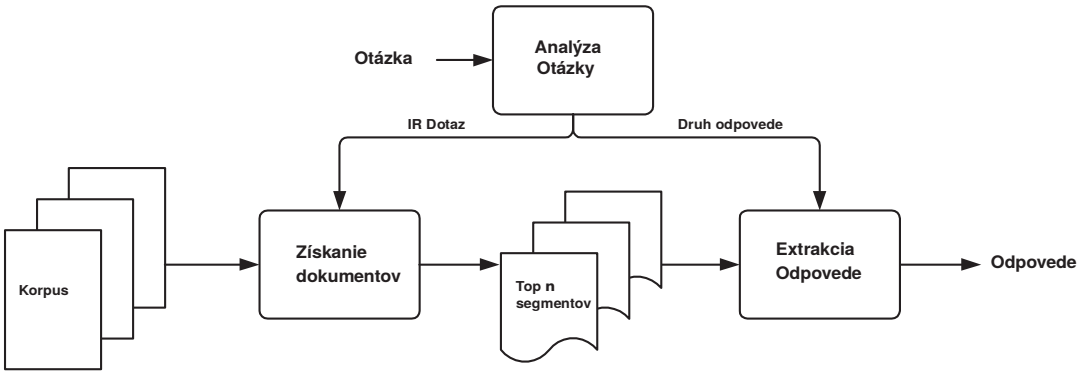
Väčšina súčasných QA systémov, ktoré sú navrhnuté tak, aby odpovedali na faktické otázky alebo zoznamové otázky (a do určitej miery aj definície), pozostávajú z troch komponentov: analýza otázky, vyhľadávanie dokumentov alebo pasáží a extrakcia odpovedí. Zatiaľ čo tieto tri zložky môžu byť a často sú rozdelené na operácie nižšej úrovne, ako je predbežné spracovanie dokumentov a analýza kandidátskych dokumentov. Trojkomponentová architektúra opisuje prístup k budovaniu systémov QA v tejto práci a v širšej literatúre.

Architektúra všeobecného troj zložkového systému QA je vidieť na obrázku 2.1.

Treba poznamenať, že zatiaľ čo tri zložky riešia úplne oddelené aspekty odpovedania na otázky, často je ťažké vedieť, kde umiestniť hranice každej zložky.

Napríklad zložka analýzy otázok je zvyčajne zodpovedná za generovanie IR dotazu z otázky v prirodzenom jazyku, ktorú potom môže komponenta na vyhľadávanie dokumentov použiť na výber podskupiny dostupných dokumentov.

Ak však prístup k vyhľadávaniu dokumentov vyžaduje určitú formu iteračného procesu na výber kvalitných dokumentov, ktoré zahŕňajú úpravu IR dotazu, potom je ťažké rozhodnúť, či by sa táto zmena mala klasifikovať ako súčasť analýzy otázok alebo procesu



Obr. 2.1: Schéma troj zložkového QA systému

vyhľadávania dokumentov . Hoci to nie je problém pri vývoji systému QA, môže to predstavovať výzvu pri pokuse o zdokumentovanie rôznych prístupov k zabezpečeniu kvality.

V tejto práci sa opisujú prístupy, ktoré sa zdajú prekračovať hranice komponentov, kde je to najvhodnejšie.

Táto kapitola stručne opisuje úlohy, ktoré tri komponenty zohrávajú pri odpovedaní na otázky týkajúce sa faktov. Každá komponenta je opísaná pomocou príkladov rôznych prístupov, ktoré sa uvádzajú v literatúre, ktoré s ňou súvisia.

Nasledujúce tri kapitoly uvádzajú priblíženie konkrétnych prístupov.

2.2.4 Analýza otázky

Prvou fázou spracovania v akomkoľvek QA systéme je analýza otázky. Vo všeobecnosti je vstupom pre túto fázu spracovania otázka v prirodzenom jazyku. Je však nutné podotknúť, že niektoré systémy zjednodušujú komponentu analýzy tým, že používajú iba podmnožinu prirodzeného jazyka s ohľadom na syntax a slovnú zásobu alebo používajú inak obmedzenú formu vstupu. Mnohé rozhrania prirodzeného jazyka do databáz majú nejakým spôsobom omedzený jazyk, ktorý môže byť použitý a niektoré QA systémy, ako S HAPAQA [6], vyžadujú aby informácie boli špecifikované explicitne pomocou rozhrania s formulárovým štýlom.

Ďalším problémom s ktorým sa komponenty analýzy otázky musia vysporiadať je kontext. Ak by bola každá otázka úplne uzavretá a nezávislá od ostatných otázok mohol by ju systém spracovať tak ako je. Ak je však otázka súčasťou dlhšej sekvencie otázok (často

nazývaná aj scenár alebo kontextové otázky) potom vedomosti ako sú odpovede alebo rozsah predchádzajúcich otázok musia byt vzané do úvahy. Zdá sa že týmto smerom sa uberá výskum v oblasti QA. Otázky v TREC 2004 QA vyhodnocovaní sú rozdelené do sád otázok o danej cieľovej entite [7]. Úlohou komponenty pre analýzu otázky je potom vyprodukovať výstup, ktorý reprezentuje otázku tak, aby mohol byť použitý ostatnými systémami pre zistenie odpoveď na danú otázku. Toto často zahrnuje vytvorenie niekoľkých reprezentácií výstupov pre rozdielne časti systému. Na príklad, komponenty pre analýzu otázky môžu dodať komponente pre získanie dokumentov správne utvorený dotaz pre získanie informácie (ang. Information Retrieval query; dalej IR dotaz) vygenerovaný z otázky, zatiaľ čo komponente pre extrakciu odpovede dodajú nejakú formu sémantickej reprezentácie očakávanej odpovede. Bolo vydaných niekoľko publikácii, ktoré sa zaoberajú oboma aspektami analýzy otázky a v nasledujúcich sekciách budu znalosti z dostupnej literatúry zhrnuté.

Vytváranie IR dotazu

Väčšina (ak nie všetky) komponenty na analýzu otázky vytvárajú IR dotaz tak, že začnú vytvorením bag-of-words modelu [24] - teda pre vytvorenie dotazu sa použijú všetky slová z otázky, pričom sa ignoruje veľkosť písmen a poradie slov. Rozdiel medzi systémami spočíva v tom ako sa k jednotlivým slovám pristupuje, aby sa vytvoril finálny IR dotaz.

Jeden bežný krok spracovania je upraviť všetky slová na slovotvorný základ, aby sa odstránili morfologické variácie.

Toto môže byt súčasťou komponenty na spracovanie IR dotazu alebo vykonané priamo v komponente pre analýzu otázky. Pre príklad, Porterov stemmer upraví slová „walk“, „walks“, „walked“ a „walking“ na slovo „walk“, čo znamená, že vyhľadaním ktoréhokoľvek z týchto slov spôsobí zhodu s dokumentom, ktorý obsahuje ktorékoľvek z týchto slov. Toto sa môže zdať byť ideálne, nakoľko dokumenty, ktoré obsahujú ktorékoľvek zo slov sú pravdepodobne relevantné vzhľadom k otázke, ktorá obsahuje slovo „walking“, avšak tento prístup má isté úskalia. Napríklad slovo „heroine“ (ang. hrdinka) bude upravené na slovo „heroin“ (heroín) , ktoré samotné ostane bez zmeny. Toto vedie k nežiadúcej situácii kde vyhľadávania dokumentov o hrdinkách vráti taktiež dokumenty o heroíne, ktoré z veľkou pravdepodobnosťou neobsahujú správnu odpoveď. Bilotti et al.[5] idú ešte ďalej a ukazujú, že v niektorých prípadoch vykonávanie úpravy na slovotvorný základ znižuje celkovú výkonnosť v porovnaní s jednoduchým prístupom bag-of-words. Ukazujú, že citlivosť aj hodnotenie relevantných dokumentov je negatívne ovplyvnené takouto úpravou. Ako alternatívu navrhujú aby systémy pristúpili k problému morfologickej variácie vytvorením dotazu, ktorý bude obsahovať rozdielne morfologické variácie jednotlivých slov. Napríklad veta „What lays blue eggs“ by bola pre tri rôzne prístupy prevedená na:

- Bag-of-Words: blue $\wedge$ eggs $\wedge$ lays
- Slovotvorný základ: blue $\wedge$ egg $\wedge$ lai
- Morfologická Expanzia: blue $\wedge$ (eggs $\vee$ egg) $\wedge$ (lays $\vee$ laying $\vee$ lay $\vee$ laid)

Zatiaľ čo toto odstraňuje problém kde sa významovo rôzne slová upravia na rovnaký základ, je zjavné, že generovanie dotazov pomocou morfologickej expanzie zväčší veľkosť indexu do korpusu (pretože pre každú morfologickú variantu bude potrebný nový záznam do indexu) a bude vyžadovať viac práce pre generovanie zoznamu alternatív. Ďalším problémom s morfologickou expanziou je, že sa vráti viac relevantných dokumentov no býva to za cenu toho že relevantné dokumenty sú horšie ohodnotené. Inými slovami tento prístup

má väčší dopad na nerelevantne dokumenty ako na relevantné. Toto môže mať negatívny dopad na výkon niektorých komponent pre extrakciu odpovede. Ktorúkoľvek metódu pre riešenie problému s morfologickými variáciami zvolíme, vždy budú existovať ďalšie problémy s formuláciou dotazu. Štandardné IR dotazy su zvyčajne ciele o ktorých by mal systém lokalizovať relevantné dokumenty, čo znamená že väčšina potrebných informácií je v dotaze samotnom. V QA je malý rozdiel v tom, že sa nesnažíme nájsť dokumenty o konkrétnom cieli, ale skôr o nejakom jeho atribúte, ktorý môže ale nemusí byť hlavným zameraním dokumentov, ktoré obsahujú odpoveď na otázku. Môže tiež nastať prípad kde zameranie vyhľadávania nie je explicitne spomenuté v otázke a otázka a dokumenty ktoré obsahujú odpoveď majú len veľmi málo spoločného. Napríklad otázka „Kde sa narodil Hans Christian Andersen?“ nijakým spôsobom nespomína odpoveď a zároveň neodkazuje na nič čo by mohlo byť hlavným zameraním relevantných dokumentov. Je napríklad celkom možné že článok o Dánsku by mohol obsahovať text „... rodisko Hansa Christiana Andersena“, ktorý aj keď odpovedá na otázku, má s ňou veľmi málo spoločného. Vo žiadne slovo okrem mena nie je prítomné v oboch.

Na druhej strane dokument o dielach Hansa Christiana Andersena by mohol spomínať jeho meno pomerne často a tak by bol vysoko relevantný, nakoľko by obsahoval mnoho termov z dotazu bez toho, aby niekedy zmienil kde sa narodil.

Bolo odskúšaných niekoľko prístupov pre riešenie tohoto problému vrátane expanzie synoným [11] a expanzia očakávaného typu odpovede [16] Aj keď oba prístupy sľubujú riešenie problému, často vedú k príliš komplikovaným dotazom, ktoré musia byť iteratívne zmiernené pred tým dojdú zmysluplné množstvo dokumentov. Navyše tiež prinášajú nové problémy. Napríklad expanzia synoným vyžaduje presné rozpoznanie významu slov v danom kontexte pre otázku predtým než môže expanzia nastať.

Odvodzovanie typu očakávanej odpovede a obmedzenia

Ako už bolo spomenuté, komponenta analýzy otázky, okrem generovania dotazu pre získanie dokumentu, taktiež odvodzuje možné obmedzenia pre očakávanú odpoveď, čo môže byť použité komponentou na extrakciu odpovede pre obmedzenie vyhľadávacieho priestoru. Tieto obmedzenia majú väčšinou sémanticku povahu. Napríklad správna odpoveď na otázku „Kde sa narodil Hans Christian Andersen?“ bude mať podobu nejakej lokácie.

Naivný prístup pre odvodenie typu očakávanej odpovede by bolo priradenie typu na základe hlavného opytovacieho zámena. „Kde“ by značilo lokáciu, „kedy“ čas alebo dátum a „kto“ zvyčajne osobu. Bohužiaľ, aj keď tento prístup funguje pre jednoduché príklady, otázky ako „Ktorú vysokú školu vyštudoval Miloš Zeman?“ vyžadujú detailnejšie spracovanie pre odvodenie že typ odpovede je univerzita (podtyp organizácie). Jeden prístup je vytvorenie hierarchie typov odpovedí a konštrukcia klasifikátora, ktorý pre vopred neznámu otázku priradí typ z hierarchie. Napríklad Li and Roth (2002) definujú dvojvrstvú hierarchiu pozostávajúcu zo šiestich hrubých tried a päťdesiatich jemných tried, pričom každá hrubá trieda obsahuje neprekrývajúcu sa množinu jemných tried. Hovy et al. (2000) vyvinuli komplexnejšiu hierarchiu pozostávajúcu z 94 uzlov, ručnou analýzou 17,384 otázok. Ak máme k dispozícii hierarchiu, systémy vyžadujú metódu priradenia špecifickej triedy k otázke. Množstvo prístupov založených na strojovom učení vrátane naivného Bayseovho klasifikátoru, rozhodovacích stromov a support vector machines boli použité s výsledkami nad jemnými triedami v rozsahu od 68% a 85%

Ďalší prístup pre odvodenie typu odpovede je modifikácia sémantickeho analyzátora aby produkoval výstup ktorý obsahuje klasifikáciu otázky. Prístup ktorý zvolili okrem iných aj systémy Webclopedia (Hovy et al., 2000) and QA-LaSIE (Greenwood et al., 2002).

Akonáhle bol určený typ očakávanej odpovede, zvyšnou povinnosťou komponenty pre analýzu otázky je odvodiť akékoľvek ostávajúce obmedzenia odpovede, ktoré by mohli komponente na extrakciu odpovede pomôcť zúžiť vyhľadávací priestor. Opäť sa vyskytuje niekoľko prístupov. Od jednoduchej extrakcie kľúčových slov pre okienkové metódy pre plnú analýzu a hľadanie vzťahov ako podmet-prísudok-predmet.

Napríklad QA-LaSIE (Greenwood et al., 2002) pri spracovaní otázky „Kto napísal Hamleta“ vyprodukuje nasledujúcu reprezentáciu v quasi-logickej forme (QLF) (Gaizauskas et al., 2005b; Gaizauskas et al., 2005c)

```
qvar(e1), qatr(e1,name), person(e1), lsubj(e2,e1),
write(e2), time(e2,past), aspect(e2,simple),
voice(e2,active), lobj(e2,e3), name(e3,'Hamlet')
```

Z tejto QLF reprezentácie otázky môže systém usúdiť nie len to že očakávaná odpoveď bude typu osoba (qvar reprezentuje vyhladávanú entitu, v tomto prípade e1 čo je podla QLF typ osoba) ale aj to, že táto osoba sa môže vo vete vyskytovať na mieste podmetu k prísudku „písať“ a predmetom viazaným k tomuto prísudku je Hamlet. Samozrejme existujú aj alternatívne metódy zisťovania obmedzení odpovede. Napríklad môžeme očakávať, že odpoveď na otázku „S kým je ženatý Jaromír Jágr“ bude typu osoba. Ďalej môžme usúdiť, že odpoveď bude ženské meno nakoľko vieme že manželstvá väčšinou spájajú muža a ženu a keďže otázka obsahuje mužské meno môžeme predpokladať že odpoveď bude ženské meno a naopak.

2.2.5 Získanie dokumentu

Aj keď už bolo spomenuté že komponenta analýzy otázky je zvyčajne zodpovedná za konštrukciu vhodného IR dotazu, neznamená to, že komponenta na získanie dokumentu používa na získanie relevantného dokumentu iba tento dotaz.

V skutočnosti môže byť komponenta na získanie dokumentu celkom komplexná a môže zahŕňať stratégie pre získanie dokumentov ako aj k ním náležiace dáta a tiež pre odhadnutie množstva a štruktúry texty ktorý bude predaný ďalšej komponente v systéme. Úplná syntaktická a sémantická analýza môže byť časovo náročný proces, ale mnoho QA systémov sa o ňu opiera pre extrakciu odpovede. Čas potrebný na vytvorenie úplnej syntaktickej a sémantickej reprezentácie často limituje jej využitie v odpovedaní na otázky v reálnom čase. Jedným z riešení, aspoň pre QA nad uzavretými kolekciami, by bolo pred spracovať celú kolekciu a uložiť výsledné syntaktické stromy a sémantickú reprezentáciu. ExtrAns (Mollá Aliod et al., 1998) je systém, ktorý robí práve toto. Odvodzuje a ukladá logické reprezentácie všetkých dokumentov do kolekcie pred akýmkoľvek odpovedaním na otázky.

Samozrejme môžu byť vopred vykonané aj iné, menej časovo náročné operácie. Vlastne neexistuje dôvod prečo by všetky operácie nezávislé na otázke(tokenizácia, NER, POS značkovanie) nemohli byť vykonané vopred, pokiaľ čas potrebný na získanie dát nie je dlhši ako ten potrebný na ich vygenerovanie, nakoľko to by pokorilo hlavný zmysel predspracovania – rýchlejšie odpovedanie na otázky.

Hlavnou úlohou fáze získavania dokumentu je však získať podmnožinu celej kolekcie ktorá bude detailne spracovaná komponentou na extrakciu odpovede. Hlavný problém v tomto bode je výber správneho IR paradigmatu a objem a struktúra textu, ktorý extrahu-

jeme. Mnoho QA systémov používa radiacie IR systémy ako Okapi (Robertson and Walker, 1999) napriek tomu že mnoho výskumov navrhuje že boolovské získavanie dokumentov je lepšie uspôsobené problémom odpovedania na otázky (Moldovan et al., 1999; Saggion et al., 2004) Formulácia dotazu pre boolovský IR prístup je oveľa komplexnejší, nakoľko je nutné zvážiť ako zoradiť a obmedziť neusporiadaný zoznam relevantných dokumentov, niečo čo sa deje automaticky s radiacimi IR systémami. Hlavnou výhodou boolovských systémov je že ich správanie je ľahšie pozorovateľné, keďže proces vyhľadávania je transportný s takmer žiadnou interakciou medzi jednotlivými vyhľadávanými termami.

Logicky by znelo, že komponenta pre získavanie dokumentov by mala vrátiť veľké množstvo dokumentov pre každú otázku aby sa zaistilo, že aspoň jeden dokument bude relevantný obzvlášť, keďže výsledky (Hovy et al., 2000) ukazujú že aj pri vrátení 1000 segmentov textu pre 8% otázok niesu žiadne dokumenty relevantné. Naneštastie skúsenosti a zverejnené výsledky ukazujú, že zatiaľ čo zvyšovanie množstva textu zvyšuje pokrytie odpovede, môže to tiež znížiť presnosť komponenty pre extrakciu odpovede. Komponenta pre získanie dokumentu je zodpovedná za udržiavanie rovnováhy medzi pokrytím a presnosťou extrahovanej odpovede. Samozrejme celkové množstvo textu predané komponenete na extrakciu odpovede môže byť štruktúrované niekoľkými spôsobmi.

Ak komponenta pre extrakciu odpovede pracuje lepšie s malým množstvom textu potom je pravdepodobné že ak jej predáme celý dokument bude fungovať horšie ako keď jej predáme väčšie množstvo krátkych relevantných pasáží, ktoré dokopy tvoria rovnaký objem textu.

Toto je preto že druhý prístup z vysokou pravdepodobnosťou obsahuje viac inštancií odpovede ako prvý (za predpokladu že odpoveď sa v jednom dokumente priemerne vyskytne nanajvýš raz – rozumný ale nepodložený predpoklad).

Množstvo publikácií udáva niekoľko spôsobov ako segmentovať celé dokumenty pre poskytnutie výberu pasáží v pokuse o zvýšenie pokrytia a zároveň udržaní celkového objemu textu na minime. Roberts a Gaizauskas (2004) a Tellex et al. (2003) preberajú niekoľko prístupov k výberu pasáže a zoradeniu zatiaľ čo Monz (2004) navrhuje prístup k výberu pasáže založený nie na fixnej veľkosti pasáží ale skôr na najmenších logických textových jednotkách ktoré obsahujú termy otázky.

2.2.6 Extrakcia odpovede

Zatiaľ čo predchádzajúce sekcie mali ozrejmiť že iba spracovanie otázky za účelom nájdenia relevantných dokumentov alebo pasáží je samo o sebe náročná úloha, je treba pamätať na to že väčšina práce potrebnej k tomu aby sme odpovedali na otázku ktorú sme predtým nevideli sa odohráva v komponente na extrakciu odpovede.

Podľa zvoleného prístupu k extrakcii odpovede bude táto komponenta musieť vykonať radu krokov spracovania aby transformovala dokumenty, ktoré sú jej predané z procesu získavania dokumentov na reprezentáciu z ktorej budú môcť byť odpovede lokalizované a extrahované. Je treba poznamenať že aj keď mohlo byť spracovanie vykonané v predstihu a údaje zaznamenané spolu s textom dokumentu, v tejto sekcii budu preberané odlišné prístupy.

Väčšina komponent na extrakciu odpovede sa spolieha na to že relevantné dokumenty boli subjektom niekoľkých bežných procedúr spracovania textu aby poskytli bohatšiu reprezentáciu než len slová tak, ako sa vyskytujú v dokumente. Toto zvyčajne zahŕňa tokenizáciu, delenie vety, POS (part of sentence) označovanie a rozpoznávanie pomenovaných entít.

V závislosti na tom aký presne postup bol zvolený pre extrakciu odpovede. Ostatné techniky budú aplikované za použitie výstupu týchto jednoduchých techník.

Napríklad povrchové vyhľadávanie textových vzorov (ang. surface matching text patterns) (Soubbotin and Soubbotin, 2001; Ravichandran and Hovy, 2002; Greenwood and Saggion, 2004) zvyčajne nevyžadujú žiadne ďalšie spracovanie dokumentov.Tieto prístupy jednoducho extrahujú odpovede z povrchovej štruktúry získaných dokumentov tým že sa spoliehajú na pomerne rozsiahly zoznam povrchových vzorov. Pre príklad, otázky pre ktoré je odpoveďou dátum narodenia môžu byť zodpovedané extrakciou odpovede použitím vzorov ako sú tieto (Ravichandran and Hovy, 2002)

```
<NAME> ( <ANSWER> - )
<NAME> was born on <ANSWER>,
<NAME> was born <ANSWER>
<NAME> ( <ANSWER>)
```

Zatiaľ čo zostavovanie rozsiahlych zoznamov takýchto vzorov môže byť časovo náročné a obtiažne, môžu byť potom výnimočne presné pre tak jednoduchý prístup.

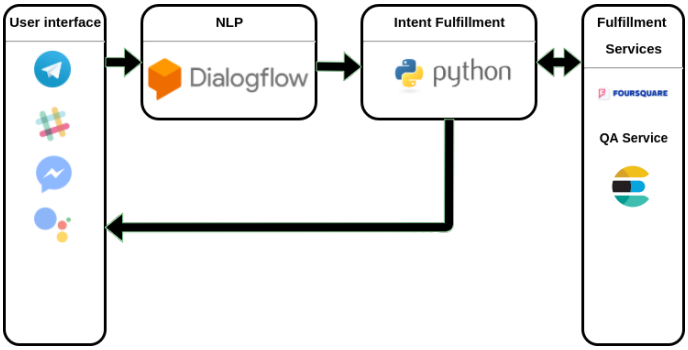
jeden systém (Soubbotin and Soubbotin, 2001) ktorý používal tento prístup ako hlavný spôsob odpovedania na otázky bol vo skutočnosti najlepšie fungujúcim systémom na TREC 2001 QA vyhodnotení, a správne odpovedal na 69% otázok (oficialnou metrikou bolo MRR (mean reciprocal rank) pre ktorú tento systém dosiahol striktné skóre 0.676). Povrchové vyhľadávanie textových vzorov funguje prekvapivo dobre s použitím maleo množstva NLP (natural language processing) techník iných než základné rozpoznávanie pomenovaných entít. Extrakčné systémy sémantického typy však vyžadujú detailnejšie spracovanie zvažovaných dokumentov. Komponenty pre extrakciu odpovede, ako tie ktoré sú opísané v Greenwood (2004a) pracujú tak, že jednoducho extrahujú najčastejšie sa vyskytujúcu entitu očakávaného typu. Toto vyžaduje schopnosť rozoznať každú entitu očakávaného typu vo voľnom texte, čo má za následok komplexnejšie rozpoznávanie entít ako to vyžadované povrchovým vyhľadávaním textových vzorov. Zatiaľ čo takéto systémy vyžadujú detailnejšie spracovanie ako povrchové vyhľadávanie, nevyžadujú hlboké lingvistické spracovanie ako úplná syntaktická alebo sémantická analýza relevantných dokumentov.

Čo je dôležité si uvedomiť je, že zatiaľ čo komponenty analýzy a získavania dokumentov je vo väčšine QA systémov relatívne podobná, komponenty extrakcie odpovede sa veľmi líšia a práve na tieto sa väčšina výskumu v tejto oblasti sústredí.

Kapitola 3

Návrh systému a voľba jednotlivých prvkov

Cieľom je zvoliť a navrhnúť vhodné riešenia tak, aby boli jednotlivé moduly na sebe do istej miery nezávisle a aby bolo možné ich funkcionalitu jednoducho rozširovať. Tok programu a prepojenie modulov znázorňuje 3.1.



Obr. 3.1: Tok programu

3.1 Užívateľské rozhranie

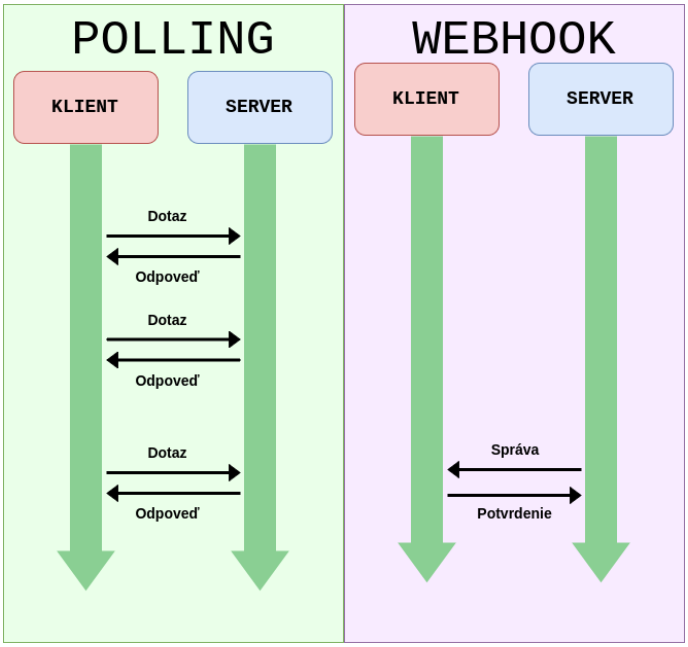
Voľba užívateľského rozhrania nebola v tomto prípade kritická. Na výber mame k dispozícii hneď niekoľko chatovacích platforiem. V tejto implementácii bude použitá platforma Telegram, pretože ponúka možnosť voľby, akým spôsobom budeme prijímať správy z platformy. Na výber mame režim dopytovania (ang. polling) alebo tzv. webhook [29] [26]. Toto nám dáva väčšiu flexibilitu pri návrhu celkového riešenia.

3.1.1 Polling a Webhook

Pri metóde sa všetky správy, ktoré prichádzajú z platformy telegram ukladajú do internej fronty na strane servera. Náš systém sa v tomto prípade musí periodicky dotazovať na obsah tejto fronty a v prípade, že sa v nej nachádzajú neobslúžené požiadavky, obslúžiť ich. Táto metóda síce viac zaťažuje sieť, ale na rozdiel od metódy webhook umožňuje, aby náš systém bežal kdekoľvek v sieti internet.

Pri metóde webhook náš systém poskytuje HTTP rozhranie na ktorom čaká na prichádzajúce požiadavky. V prípade novej správy potom služba Telegram túto správu pošle na toto HTTP rozhranie. Výhodou je že náš systém nezaťažuje sieť a odozva medzi dobou kedy užívateľ správu odošle a kedy táto správa príde do nášho systému je nižšia pretože správa je obdržaná ihneď, nie až keď sa dopytujeme. Nevýhodou je, že systém musí vytvoriť rozhranie, ktoré je prístupné z externej siete.

Toto porovnanie je znázornene na obrázku 3.2



Obr. 3.2: Porovnanie metód polling a webhook

3.2 Dialógový systém/spracovanie požiadavky užívateľa

Pre spracovanie a extrahovanie informácií z požiadavku užívateľa bol zvolený systém Dialogflow. V porovnaní s ostatnými systémami, ktoré boli pre účely tejto implementácie preskúmané, ponúka najväčšie množstvo vstavaných typov entít, najviac hodnôt entít a podporu pre udržiavanie kontextu. Systém je teda stavový a dokáže tak udržiavať konverzácie s niekoľkými užívateľmi naraz.

Táto služba podporuje priamu integráciu pre väčšinu platforiem a následne predanie spracovaných informácii pomocou volania webhook. My však budeme používať službu pomocou volania gRPC[25] API. Toto nám umožní väčšiu flexibilitu, nakoľko pri volaní cez webhook služba očakáva odpoveď s určitou maximálnou dobou odozvy. Keďže operácie ako odpovedanie na otázky z voľného textu sú časovo náročné, je volanie gRPC vhodnejším riešením.

3.2.1 Detekcia zámeru

Pomocou definovania zámerov (ang. intent) definujeme správanie systému a to akým druhom požiadaviek od užívateľa bude rozumieť. V zámeroch definujeme tzv. sloty. Tieto sloty definujú, aké hodnoty akého typu očakávame že sa môžu vyskytnúť vo vstupe od užíva-

teľa. Každému slotu teda prináleží typ entity. Niektoré sloty máme možnosť definovať ako povinné. V prípade, že užívateľský vstup sa detekuje ako konkrétny zámer ale nevyskytne sa v ňom entita, ktorá by mohla byť pre tento slot použitá, systém vygeneruje doplňujúcu otázku. Sloty reprezentujú konkrétny význam danej entity v konkrétnej vete. Napríklad entita typu geolokácia môže byť použitá pre sloty miesto odletu a miesto príletu v prípade, že by sme definovali zámer, kde sa užívateľ snaží vyhľadať lety.

Pri detekcii konkrétneho zámeru máme možnosť nastaviť kontexty pre aktuálnu reláciu/užívateľa. Tieto kontexty udržiavajú stav konkrétnej konverzácie. Pri definícií zámeru máme možnosť definovať vstupné kontexty, ktoré musia byt aktívne, aby mohol byť vstup vyhodnotený ako daný zámer. Toto dovoľuje vetvenie dialógu a udržiavanie konverzácie.

3.2.2 Entity

Systém definuje množstvo vstavaných entít no zároveň dovoľuje vytvárať vlastné. Pre nami vytvorené entity je možné definovať konkrétne hodnoty ktoré môže entita nadobúdať ale zároveň umožňuje automatické odvodzovanie hodnôt. Pri automatickom odvodzovaní môžu byt v zámeroch detekované aj také hodnoty, ktoré nie sú explicitne pre daný typ entity definované. Pre každú hodnotu ďalej dovoľuje definovať synonymá, ktoré budu namapované na konkrétnu hodnotu.

3.3 Uspokojenie požiadaviek

Naša služba pre uspokojenie požiadaviek prijíma spracovaný užívateľský vstup z dialógoveho systému. Podla detekovaného zámeru potom zvolí vhodnú službu a predá jej parametre/sloty, ktoré boli dialógovým systémom rozpoznané. Následne výsledky zo zvolenej služby spracuje, sformátuje a pomocou služby, ktorá zabezpečuje komunikáciu s užívateľským rozhraním odošle užívateľovi.

3.3.1 Udalosti

V prípade, že sa užívateľ zaujíma o nejaký druh udalosti v meste, použijeme službu na prehľadávanie udalostí pre uspokojenie tohoto dotazu. Táto služba pracuje s databázou udalosti, ku ktorej pristupujeme pomocou systému Elasticsearch. Výhodou tohoto prístupu je, že systém elastic search má podporu pre full-textove dotazy takže požidavaku od užívateľa nemusíme nijakým spôsobom konvertovať ale v prípade, že je tento zámer detekovaný, môžeme elastic searchu predať pôvodnú správu od užívateľa. Výhodou je, že Elastisearch ponuka veľké množstvo typov dotazov a úpravou dotazov vieme vyladiť správanie systému.

3.3.2 Podniky

V prípade, že sa jedná o zámer, keď sa uživateľ snaží vyhladať špecifické podniky v meste, použijeme externú službu Foursquare [2]. Prístup k tejto službe je zabezpečený pomocou REST rozhrania. Parametre z dialógového systému konvertujeme na formát, ktorý tato služba očakáva. Odpoveď od služby následne opät prevedieme na vhodný formát a výsledky vrátime užívateľovi. Toto je príklad použitia externej služby pre uspokojenie požiadavky užívateľa.

3.3.3 QA

Pre všeobecné doménovo nešpecifické otázky/dotazy použijeme našu QA službu. Táto služba používa Elastisearch ako komponentu pre získanie dokuemntov a systém R-net [17] a jeho neoficiálnu implementáciu[30] implementovanú vo frameworku DeepPavlov[18]ako komponentu pre extrakciu odpovede.

Elasticsearch

Pre uloženie našej znalostnej báze a extrakciu relevantných dokumentov použijeme systém Elasticsearch.

Výhodou je že Elasticsearch priamo podporuje full-textové dotazy, takže na našej strane nie je potrebne nijaké predspracovanie užívateľovho dotazu. Pomocou dotazu pre Elasticsearch vyberieme z korpusu N najrelevantnejších dokumentov. Tieto dokumenty následne použijeme ako kontext v ktorom z ktorého sa R-net bude pokúšať extrahovať odpoveď. Hodnotenie relevancie v Elasticsearch funguje nasledovne.

Elasticsearch používa Boolovský model[27] pre vyhľadanie odpovedajúcich dokumentov, a formulu practical scoring function [1]pre výpočet relevantnosti. Táto formula využíva koncepty z modelu frekvencia termu/inverzná frekvencia dokumentu (ang.term frequency/inverse document frequency, alebo TF/IDF)[28] ale pridáva moderné prvky ako koordinačný faktor, normalizácia dĺžky pola a posilovanie termu alebo dotazu.

3.3.4 Boolovský model

Boolovský modej jednoducho aplikuje podmienky logického súčinu, súčtu a negácie vyjadrené v dotaze aby našiel dokumenty ktoré vyhovujú. Dotaz $full \wedge text \wedge search \wedge (elasticsearch \vee lucene)$ bude zahrnovať iba dokumenty ktoré obsahujú všetky z termov „(“full), „text“ a „search“ a „elasticsearch“ alebo „lucene“.

Tento proces je rýchli a jednoduchý. Je použitý na vylúčenie dokumentov, u ktorých je vylúčené, že by mohli vyhovovať dotazu.

3.3.5 TF/IDF

Po tom čo získame vyhovujúce dokumenty je nutné ich ohodnotiť podla relevantnosti. Nie všetky dokumenty budú obsahovať všetky termy a niektoré termy sú dôležitejšie ako iné. Skóre relevantnosti celého dokumentu závisí (z časti) na váhe jednotlivých termov dotazu ktoré sa v dokumente vyskytujú. Váha termu je ovplyvnená troma faktormi, ktoré sme v úvode predstavili.

3.3.6 Frekvencia termu

Ako často sa term vyskytuje v dokumente? Čím častejšie tým väčšia váha. Pole ktoré obsahuje päť výskytov rovnakého termu je relevantné s vyššou pravdepodobnosťou ako pole ktore obsahuje iba jeden výskyt. Frekvencia termu sa počíta ako:

$$tf(tind) = \sqrt{frequency}$$

3.3.7 Inverzná frekvencia dokumentu

Ako často sa term vyskytuje vo všetkých dokumentoch v kolekcii? Čím častejšie tým nižšia váha. Bežne vyskytujúce sa termy ako neurčité členy v angličtine neprispievajú rele-

vantnosti, nakoľko sa vyskytujú vo väčšine dokumentov, zatiaľčo menej bežné termy ako „elastic“ nám pomáhajú zamerať sa na tie najzaujímavejšie dokumenty. Inverzná frekvencia dokumentu sa počíta ako:

$$idf(t) = 1 + log(numDocs/(docFreq + 1))$$

3.3.8 Norma dĺžky poľa

Aké dlhé je pole? Čím kratšie je pole, tým vyššia váha. Ak sa term vyskyuje v krátkom poli, ako je napríklad pole ktoré označuje sumár článku, je pravdepodobnejšie že obsah tohoto poľa je o danom terme ako keď sa ten istý term vyskytne v oveľa väčšom poli. Normu dĺžky poľa spočítame ako:

$$norm(d) = 1/\sqrt{numTerms}$$

3.3.9 Dávame to dokopy

Tieto tri faktory — frekvencia termu, inverzná frekvencia dokumentu a norma dĺžky poľa sú spočítané a uložené v čase indexovania. Spolu sú použité pre výpočet váhy jedného termu v konkrétnom dokumente.

R-NET - komponenta pre extrakciu odpovede

Pre extrakciu odpovede zo získaného kontextu použijeme systém R-NET[17] a jeho neoficiálnnu implementáciu[30] vo frameworku DeepPavlov[18].

R-net sa primárne zameriava na dátovú sadu SQuAD pre trénovanie a vyhodnotenie tohoto modelu. SQuAD sa skladá z vyše 100,000 otázok vypracovaných dobrovoľníkmi nad 536 článkami wikipédie. Dátová sada je náhodne rozdelená na trénovaciu sadu (80%), vývojovú sadu (10%) a vyhodnocovaciu sadu (10%). Odpoveďou na každú otázku je úsek korešpondujúcej pasáže. Systém používa tokenizér, ktorý je súčasťou Stanford CoreNLP pre predspracovanie každej pasáže a otázky. Jednotka Gated Recurrent Unit (dalej GRU) [3] ktorá je variantov LSTM (Long Short-Term Memory) je použitá naprieč modelom. Pre vnorenia slov sú použité predtrénované GloVe vnorenia [20] pre otázky aj pasáž a počas trénovania sú fixované. Pre slová mimo slovníka sú použité nulové vektory. Pre výpočet vnorení na úrovni znakov je použitá jedna vrstva obojsmernej GRU a pre kódovanie otázok a pasáží 3 vrstvy obojsmernej GRU. Dĺžka skrytého vektora je nastavená na 75 pre všetky vrstvy. Skrytá veľkosť použitá pre výpočet skóre pozornosti je tiež 75. Aplikujeme výpadok (ang. dropout) [23] medzi vrstvami s rýchlosťou výpadku 0.2. Model je optimalizovaný s AdaDelta (Zeiler, 2012) s počiatočnou rýchlosťou učenia 1. Hodnoty ρ a ϵ použité v AdaDelta sú 0.95 a $1e^{-6}$

Sú využívané dve metriky pre vyhodnotenie výkonnosti modelu pre SQuAD: Presná zhoda (ang. Exact match; ďalej EM) a F1 skóre. EM udáva aké percento predikcií sa presne zhoduje s pevnými faktom (ang. ground truth). F1 udáva prekrytie medzi predikciou a pevnými faktami a udáva maximálne F1 spomedzi všetkých pevných faktov.

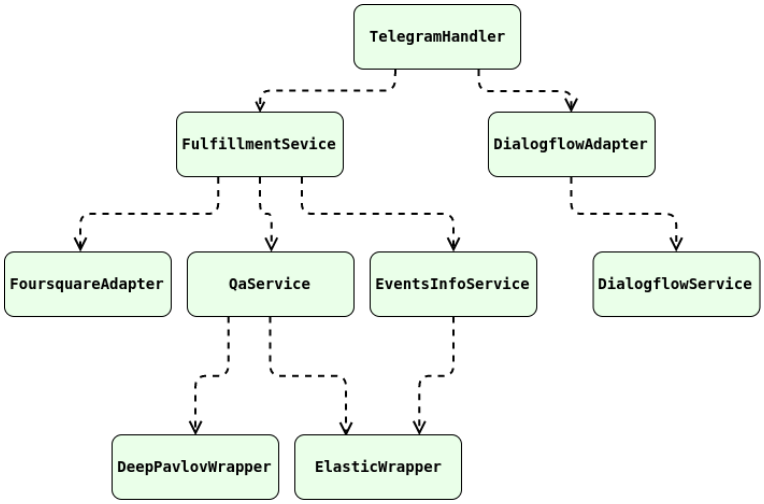
Autor neoficiálnej implementácie ktorú používa náš systém udáva hodnoty: EM=71.07, F1=79.51 vyhodnotené na sade SQUAD

Model ktorý použijeme v našom riešení bude taktiež predtrénovaný na SQUAD sade.[21]

Kapitola 4

Implementácia a prepojenie jednotlivých modulov

Nasledujuca kapitola popisuje prepojenie a implementaciu jednotlivých časí agenta. Diagram 4.1 popisuje zavislosti medzi jednotlivými triedami.



Obr. 4.1: Diagram závislostí

4.1 Užívateľské rozhranie

Pre spracovanie správ zo služby Telegram je implementovaná tieda `TelgramMessageHandler` . V tejto triede definujeme metódy pre obsluhu jednotlivých druhov správ z platformy (textové, hlasové, lokácia...). Objekt tejto triedy po inicializácii následne pravidelne dopytuje správovy server a zisťuje, či sa vo fronte nachádzajú neobslúžené správy. Neobslúžené správy v poradí vyťahuje z fronty a obsluhuje ich zaregistrovaná obslužná metóda. Pre obsluhu každej správy vznika nove vlákno a tak je možne spracovávať novo prichádzajúce správy aj počas obsluhy inej správy.

4.2 Spracovanie prirodzeného jazyka a extrakcia informácii

Spracovanie prirodzeného jazyka, detekciu a rozpoznanie entit v požiadavku a detekciu zámeru zaisťuje služba Dialogflow. Pre tuto službu je vytvorená trieda `DialogflowAdapter`, ktorá obaľuje vzdialené volanie tejto služby pomocou protokolu gRPC[25]. Hlavnou metódou tejto triedy je `DialogflowAdapter.detect_intet`, ktorá vracia objekt obsahujúci informácie extrahované zo správy užívateľa. Objekt, ktorý táto metóda vracia má formát A.11. Do volania tejto metódy predávame neupravený textový vstup od užívateľa. Prípadne zvukovú stopu v prirodzenom jazyku.

4.2.1 Definícia správania dialógového systému

Správanie NLP systému a definícia zámerov a entít, ktoré dokáže v súčastnosti detektovať je definovaná v súboroch `data\intents` a `data\entities`. Správanie tohoto systému je možne meniť pomocou príkazov uvedených v 5.2.1 Správanie systému je stavové a konverzácie sú izolované pre jednotlivých užívateľov. Momentálne sú v agentovi definované 4 zámery

- factual_question- pri detekcii tohoto zámeru nedetekujeme žiadne entity. Celý vstup od užívateľa predávame QA modulu.
- venues.eating_out.search a venues.nightlife.search tieto zámery uspokojuje služba Foursquare. Zo zámeru extrahujeme parametre ktoré adaptujeme na parametre tejto API
- events.search tento zámer uspokojujeme pomocou prehľadávania indexu Elasticsearch. Z detekovaného zámeru extrahujeme parametre ktoré následne použijeme ako termy pre vyhľadávanie v indexe

4.3 Služby pre uspokojenie požiadavku

Služba pre uspokojenie požiadavku ako vstup prijíma objekt, ktorý je výstupom NLP služby. Na základe detekovaného zámeru potom rozhodne, ktorej podslužbe tento objekt predá. Dana podslužba z objektu extrahuje informácie, ktoré potrebuje - väčšinou detekované enity a ich hodnoty. Pomocou týchto potom vykoná aktivity, na ktoré je služba určená a výsledok vráti hlavnej službe. Tá výsledok sformátuje pre cieľovú platformu a vráti ako spravu pripravenú na odoslanie užívateľovi. Jednotlive podslužby sú popísane v nasledujúcich kapitolách. V prípade že NLP služba nedetekuje žiaden zámer alebo je zámer neúplný - teda chýbajú hodnoty povinných parametrov prepošleme užívateľovi správu vygenerovanú NLP systémom ktorá bud užívateľa dotáže na hodnoty povinných parametrov, alebo ho informuje že daná požiadavka nebola rozpoznaná

4.3.1 Odpovedanie na otázky

Získanie dokumentov

Pre získanie dokumentov pre komponentu pre extrakciu odpovede je použitý Elasticsearch. Pristúp do databáze zaisťuje trieda `ElasticWrapper`. Definicia dotazu pre získavanie relevantných zo dokumentov je definovaná v metóde `get_relevant_docs` pomocou dotazu 4.1 Správanie tohoto modulu je možne upraviť zmenou tohoto dotazu.

```
"query": {{
   "multi_match" :{{
       "query" :"{question}",
       "fields": ["text"]
  }}
}}
```

Dotaz 4.1: Získanie dokumentov z korpusu

Extrakcia odpovede

Pre extrakciu odpovede zo získaného kontextu je použitý QA modul frameworku DeepPavlov. Rozhranie nad týmto systémom zabezpečuje trieda `DeepPavlovWrapper` Tento framework využíva pre extrakciu odpovedi neoficiálnu implementáciu systému R-net navrhnutý firmou Microsoft. Framework DeepPavlov ponuka prístup k tejto funkcionalite aj pomocou HTTP rozhrania a REST API ktoru budeme využívať my. V triede `DeepPavlovWrapper` je definovaná adresa, na ktorej tento server beží a kam budeme posielať požiadavky. Nakoľko pôvodná implementácia HTTP servera vo frameworku DP bola nefunkčná bolo nutne ju mierne upraviť. HTTP server poskytuje jediný endpoint a podporuje jednu metódu POST. Tato metóda očakáva parametre context_raw a question_raw, kde context raw je text v ktorom sa bude systém pokúšať nájsť odpoveď na otázku question_raw

4.3.2 Vyhľadávanie udalosti

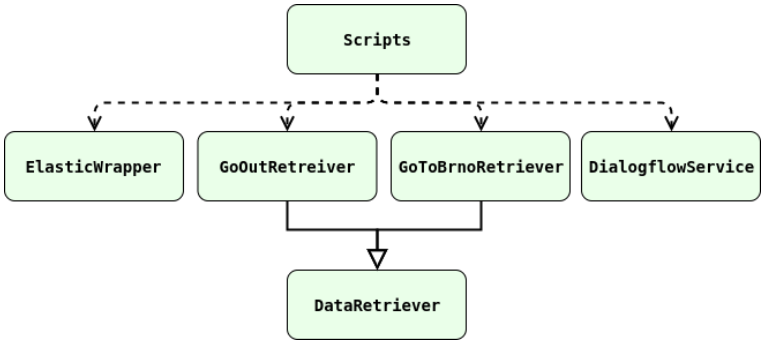
Vyhľadávanie a uloženie báze udalostí je riešené pomocou triedy `EventsInfoService`. Samotné uloženie udalostí a ich vyhľadávanie je z tejto triedy realizované pomocou volania služby pre prístup do systému Elasticsearch. Z užívateľského zámeru sa extrahujú parametre a tieto sa použijú pre krížové vyhľadávanie v báze udalostí. Výber udalosti je obmedzený na konkrétny dátum alebo rozsah dátumov zadaný užívateľom. Ostatné kľúčové slova ako druh udalosti, názov skupiny alebo názov udalosti sú krížovo porovnané s hodnotami atribútov jednotlivých záznamov.

4.3.3 Vyhľadávanie podnikov

Vyhľadávanie podnikov je realizované pomocou externej služby Foursquare[2] a triedy `FoursquareAdapter` ktorá obaľuje toto volanie. Parametre sú extrahované z uživatelského zámeru a z nich je vytvorený dotaz pre túto službu.

4.4 Skripty

Pre prácu so znalostnými databázami a pre úpravu správania dialógového systému bola implementovanie trieda, ktorá ponuka príkazy pre tuto manipuláciu. Trieda implementuje jednoduché rozhranie príkazového riadka. Použitie týchto príkazov je popísané v 5.2.1. Diagram závislostí tejto triedy je znázornený na diagrame 4.2.



Obr. 4.2: Diagram závislostí

Kapitola 5

Použitie

5.1 Služby

Pre použitie a správne fungovanie agenta je nutné zabezpečiť chod niekoľkých služieb.

5.1.1 Telgram GUI

Užívateľské rozhranie k nášmu agentovi je prístupne cez miltiplatformnú četovaciu aplikáciu Telegram. Adresa nášho agenta v rámci tejto platformy je t.me/IBT_BrnoBot. Pre použitie agenta je potrebné navigovať na túto adresu pomocou aplikácie telegram na ktorejkoľvek podporovanej platforme. Verzia webového rozhrania je dostupná na adrese https://web.telegram.org/#/im?p=@IBT_BrnoBot

5.1.2 Elasticsearch server

Náš agent využíva systém Elasticsearch pre ukladanie a vyhľadávanie udalostí a korpusu dokumentov pre QA systém. Pre účely testovania je možné použiť server, ktorý beží na adrese 85.216.203.159:9200. Na tomto serveri už sú vytvorené indexy a sú naplnené vzorovými údajmi. Pre manipuláciu s týmito databázami je možné použiť príkazy 5.2.2

5.1.3 QA server

Je potrebné zabezpečiť beh servera, ktorý bude slúžiť ako komponenta pre extrakciu odpovede nášho QA riešenia. Tu využijeme framework DeepPavlov[18], ktorý túto komponentu poskytuje a zároveň ponúka beh tejto služby ako REST API server. Server môžeme pustiť lokálne pomocou príkazu 5.1 alebo použiť testovací server, ktorý beží na adrese 85.216.203.159:5200. Testovací server používa konfiguráciu, ktorá je trénovaná na sade SQuAD.

```
DeepPavlov$ python3.6 -u -m deeppavlov.deep riseapi <config-file.json>
```

Príkaz 5.1: Spustenie QA servera

5.2 Spustenie a konfigurácia

Po zabezpečení behu všetkých potrebných služieb je potrebné agenta nakonfigurovať.

Adresu spusteného QA servera je potrebné nastaviť v triede `DeepPavlovWrapper`5.2. Adresa servera musí byť prístupná zo siete v ktorej bude bežať náš agent.

```
class DeepPavlovWrapper:
  SERVER =<adresa_qa_servera>
  ...
```

Výpis 5.2: Nastavnie adresy QA servera

Adresu Elasticsearch servera je potrebné nastaviť v triede `ElasticWrapper` 5.3. Adresa servera musí byť prístupná zo siete v ktorej bude bežať náš agent

```
class ElasticWrapper(Elasticsearch):
  SERVER =<adresa_elastic_servera>
  ...
```

Výpis 5.3: Nastavnie adresy Elasticsearch servera

5.2.1 Konfigurácia dialógového systému

Pre konfiguráciu dialógového systému bola implementovaná trieda `Scripts` ktorá ponúka jednoduché rozhranie príkazového riadka. K dispozícii sú nasledujúce príkazy.

`$python scripts.py create-intent <intent-file>`

Príkaz vytvorí v dialógovom systéme nový užívateľský zámer. `<intent-file>` je súbor s definíciou zámeru vo formáte A.1.

`$python scripts.py create-entity-type <entitytype-file>`

Príkaz vytvorí v dialógovóm systéme nový typ entity. `<entitytype-file>` je súbor s definíciou typu entity vo formáte A.9.

`$python scripts.py create-entity-values <entity-name> <entities-file>`

Príkaz pridá typu entity `<entity-name>` hodnoty ktoré môže nadobúdať. `<entities-file>` je súbor s hodnotami vo formáte 5.4

```
{
  "entities": [
    object(Entity)
  ],
  "languageCode": string
}
```

Štruktúra 5.4: Formát súboru `<entities-file>`

`$python scripts.py delete-all-entities-values <entity-name>`

Príkaz odstráni všetky hodnoty ktoré môže typ entity `<entity-name>` nadobúdať.

`$python scripts.py delete-all-intents`

Príkaz odstráni všetky užívateľské zámery, ktoré sú v agentovi definované.

`$python scripts.py delete-entity-type <entity-name>`

Príkaz odstráni typ entity `<entity-name>` z agenta. Aby bolo odstránenie možné nesmie byt tento typ entity použitý v žiadnom užívateľskom zámere.

`$python scripts.py delete-intent <intent-name>`

Príkaz odstráni užívateľský zámer `intent-name` z definície agenta.

`$python scripts.py get-intent <intent-name>`

Príkaz zobrazí definíciu užívateľského zámeru `<intent-name>`. Formát výstupu je A.1

`$python scripts.py show-all-entity-types`

Príkaz zobrazí definíciu všetkých typov entít, ktoré sú v agentovi definované. Výstup má formát 5.5

```
{
  "entityTypes": [
    object(EntityType)
  ]
}
```

Štruktúra 5.5: Formát výstupu metódy `show-all-entity-types`

`$python scripts.py show-all-intents`

Príkaz zobrazí všetky uživateľské zámery definované v agentovi. Výstup ma formát 5.6

```
{
  "entityTypes": [
    object(Intent)
  ]
}
```

Štruktúra 5.6: Formát výstupu metódy `show-all-intents`

`$python scripts.py train-agent`

Príkaz spustí tréning agenta s aktuálne definovanými zámermi a trénovacími vetami

5.2.2 Manipulácia s databázami

Vytvorenie indexu

V prípade že nepoužijeme predvolený Elasticsearch server kde už su indexy vytvorené je nutné ich vytvoriť pomcou príkazu 5.2.2

`$curl -X PUT "<adresa_servera>/<nazov_indexu>"`

Názvy indexov je potom nutné nastaviť na príslušných miestach. Názov indexu pre uloženie udalostí je nutné nastaviť v triede `ElasticWrapper` 5.7

```
class ElasticWrapper(Elasticsearch):

  def __init__(self, **kwargs):
    ...
    self.events_index =<nazov_indexu>
    ...
```

Výpis 5.7: Nastavnie indexu udalostí

Názov indexu pre uloženie korpusu pre QA je nutné nastaviť v triede `ElasticWrapper` 5.8

```
class ElasticWrapper(Elasticsearch):

  def __init__(self, **kwargs):
    ...
    self.qa_index =<nazov_indexu>
    ...
```

Výpis 5.8: Nastavnie indexu QA korpusu

Práca s dátami

`$curl -X POST "<server_address>/<qa_index_name>/_doc/" -H 'Content-Type: application/json' -d @input_file.json`

Príkaz vloží záznam do QA indexu. Formát súboru `input_file.json` je 5.9

```
{
  "id": "nazov dokumentu",
  "text": "obsah dokumentu"
}
```

Štruktúra 5.9: Formát súboru `input_file.json`

`$curl -X POST "<server_address>/<index_name>/_doc/<event_id>" -H 'Content-Type: application/json' -d @input_file.json`

Príkaz vloží záznam do indexu udalostí. Formát súboru `input_file.json` je 5.10

```
{

  "event_id": <event_id>, #identifikator udalosti vacsinou prevzaty z originalneho zdroja
  "event_title": "Nazov udalosti",
  "event_description": "Opis udalosti",
  "date_from": "Datum zaciatku", #format YYYY-MM-DD
  "date_to": "Datum Konca", #format YYYY-MM-DD
  "price_from": 0,
  "price_to": 0,
  "location": "Polarka Theatre, Tuckova 34, Brno - stred",
  "categories": ["zoznam", "kategorii"],
  "url": "odkaz na povodny zdroj"
}
```

Štruktúra 5.10: Formát súboru `input_file.json`

`$python scripts.py populate-events-index`

Príkaz vloží do indexu aktuálne udalosti z webov gotobrno.cz a goout.net

`$curl -X DELETE "<server_address>/<nazov_indexu>/_doc/<_id>"`

Príkaz odstránni dokument z indexu

5.2.3 Spustenie

Po spustení všetkých potrebných služieb, naplnení indexov a konfigurácii jednotlivých modulov je možné agenta spustiť pomocou príkazu 5.11. Spustením agenta sa vyprázdni fronta neobslúžených správ a agent začne obsluhovať nové správy. Agent je prístupný cez rozhranie aplikácie Telegram.

```
$ python3.6 bot.py start_bot
```

Príkaz 5.11: Spustenie bota

5.3 Testovanie na užívateľoch

Výsledný systém bol testovaný na malej vzorke anglicky hovoriacich užívateľov. Bolo im vysvetlené, aké typy požiadaviek dokáže agent splniť a akým spôsobom s ním pracovať. Následne bola formou dialógu zozbieraná spätná väzba. Z odpovedí respondentov vyplynulo, že za najpresnejšiu považujú službu na vyhľadanie udalosti. Pri službe na vyhľadanie prevádzok niektorým vadilo, že služba neberie v potaz aktuálnu polohu užívateľa (nefunguje tým pádom zoradenie podľa vzdialenosti). So službou na odpovedanie na otázky boli vo väčšine prípadov nespokojní. Užívateľská odozva bola zohľadnená v 6.1.

Kapitola 6

Záver

Práca ukazuje, ako je možné použiť existujúce riešenia v oblasti spracovania prirodzeného jazyka a komunikačných agentov a spojiť tieto riešenie do funkčného celku. Za hlavný prínos práce považujem to, že počas vývoja bolo pre každú časť celkového riešenia preskúmaných niekoľko systémov a niekoľko spôsobov riešení a následne boli zvolené tie, ktoré som považoval za najvhodnejšie či už hľadiska použiteľnosti alebo rozšíriteľnosti. Vo výslednej architektúre je možné jednotlivé komponenty nezávisle meniť, pričom sú tieto komponenty jednoducho upraviteľné. Je teda možné správanie systému prispôsobiť rôznym prípadom použitia a cieľovým skupinám úpravami alebo doplnením jednotlivých modulov. Práca demonštruje 3 druhy naplnenia užívateľských požiadaviek. Vyhľadávanie odpovedi v neštruktorovanom texte, vyhľadávanie odpovedi v štrukturovaných dátach a uspokojenie pomocou volania externej služby. Množinu dotazov, ktoré dokáže systém uspokojiť je teda možné zvýšiť rozšírením znalostných báz alebo napojením ďalších externých služieb (napr. hromadná doprava, počasie). Práca splna zadanie, pričom necháva veľký priestor na množstvo úprav a vylepšení.

6.1 Možné optimalizácie a vylepšenia

- Odpovedanie na otázky pomocou QA service je jednou z časovo najnáročnejších operácii v celom systéme. Nakoľko sa táto operácia začne vykonávať až po tom, čo sa vyhodnotí užívateľský zámer, doba odozvy je relatívne vysoká oproti iným operáciám. Tomuto by sa dalo predísť tak, že by sa ihneď po príchode správy od užívateľa táto sprava predala QA modulu, ktorý by počas blokujúceho volania dialogflow service zatiaľ vyhodnotil vstup od užívateľa ako otázku. V prípade, že dialogflow vyhodnotí užívateľský zámer ako dotaz pre QA service, vráti sa už predpočítaná hodnota. Týmto by sa mohla výrazne urýchliť doba odozvy na tieto typy užívateľských dotazov.

- QA systém ako celok sa skladá z dvoch častí. R-net sa pokúša nájsť najvhodnejšiu odpoveď na otázku v dodanom kontexte. Tento kontext je výstupom document retrieveru. Úspešnosť celého Qa modulu je teda z veľkej časti závislá na tom, či doc retriever poskytne Rnetu vhodný kontext. Tu vidím veľký priestor na vylepšenie pomocou supervised data alebo spätnou väzbou od užívateľov.

- Nakoľko veľké množstvo súčasných chatovacích platforiem ponúka možnosť posielať hlasové správy, bolo by dobré poskytnúť pre tento typ správ podporu. Od API verzie 2 ma dialogflow priamu podporu pre spracovanie zvukového vstupu, takže nebude nutné

riešiť prevod reči na text iným spôsobom. Jediné, čo bude nutné zabezpečiť je kompatibilita kódovania zvukových sprav. Prichádzajúce spravy z jednotlivých platforiem bude nutné prekódovať na audio formát podporovaný touto službou.

- Pri plnení databázy udalostí v meste sa získavajú pomocou HTTP protokolu informácie zo zoznamu zdrojov. HTTP dotazy sú v tomto prípade blokujúce a pri sériovom spracovaní jednotlivých URL a väčšom množstve URL sa čas potrebný na spracovanie predlžuje. Tento proces by bolo vhodne upraviť na asynchrónne volania a paralelizovať. Nakoľko ale túto dávku nie je nutné volat viac ako 1 krát denne, nie je to zatiaľ nutné.

- Architektúra programu by sa mala upraviť tak, aby lepšie podporovala nasadenie na iné chat platformy. Konkrétne by bolo vhodné upraviť fulfillment service tak, aby vracala výsledky jednotlivých služieb vo všeobecnom formáte, napríklad v slovníku. Pre každú novú platformu by bolo potom potrebné implementovať konvertor, ktorý by dáta z fulfillment service konvertoval na formát správy špecificky pre konkrétnu platformu. Podobnú úpravu by bolo nutne spraviť pre volanie dialogflow service, kde by sa prichádzajúce spravy z jednotlivých platforiem museli konvertovať na jednotný formát ako vstup pre dialogflow service.

- Rozšírenie o možnosť uspokojiť ďalšie typy požiadaviek od užívateľa. Toto by vyžadovalo rozšíriť fulfillment service o funkcionalitu, ktorá by tieto nove typy požiadaviek uspokojovala. Ďalej by bolo nutne rozšíriť konfiguráciu dialogflow o definície nových entít, akcii, slotov, zámerov a anotovaných príkladov.

- Po každom splnenom dotaze vypýtať od užívateľa spätnú väzbu a tieto údaje využiť pre posilňované učenie.

- Často sa stane, že sa užívateľ pri zadávaní požiadavky „preklepne“. Bolo by celkom užitočné, ak by systém dokázal automaticky detekovať a opravovať chyby. Toto by sa mohlo diať bez vedomia užívateľa alebo by mohol byť užívateľ oboznámený s tým, že výsledný dotaz ktorý systém vyhodnocuje, sa líši od toho, čo zadal on sám.

Literatúra

[1] Elasticsearch. https://www.elastic.co/guide/en/elasticsearch/guide/current/practical-scoring-function.html, 2018, [Online; accessed 15-May-2018].

[2] Foursquare. https://developer.foursquare.com/docs, 2018, [Online; accessed 15-May-2018].

[3] Bahdanau, D.; Cho, K.; Bengio, Y.: Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] Bikel, D. M.; Miller, S.; Schwartz, R.; aj.: Nymble: a high-performance learning name-finder. In *Proceedings of the fifth conference on Applied natural language processing*, Association for Computational Linguistics, 1997, s. 194–201.

[5] Bilotti, M. W.; Katz, B.; Lin, J.; aj.: What works better for question answering: Stemming or morphological query expansion. In *Proceedings of the Information Retrieval for Question Answering (IR4QA) Workshop at SIGIR*, ročník 2004, 2004, s. 1–3.

[6] Buchholz, S.; Daelemans, W.: Complex answers: a case study using a www question answering system. *Natural language engineering*, ročník 7, č. 4, 2001: s. 301–323.

[7] Buckley, C.; Voorhees, E. M.: Retrieval evaluation with incomplete information. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, 2004, s. 25–32.

[8] Cheng, T.; Yan, X.; Chang, K. C.-C.: Supporting entity search: a large-scale prototype search engine. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, ACM, 2007, s. 1144–1146.

[9] Fader, A.; Soderland, S.; Etzioni, O.: Identifying relations for open information extraction. In *Proceedings of the conference on empirical methods in natural language processing*, Association for Computational Linguistics, 2011, s. 1535–1545.

[10] Grishman, R.; Sundheim, B.: Message understanding conference-6: A brief history. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*, ročník 1, 1996.

[11] Hovy, E. H.; Gerber, L.; Hermjakob, U.; aj.: Question Answering in Webclopedia. In *TREC*, ročník 52, 2000, s. 53–56.

[12] Jiang, J.: Information extraction from text. In *Mining text data*, Springer, 2012, s. 11–41.

[13] Jiang, J.; Zhai, C.: Exploiting domain structure for named entity recognition. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, Association for Computational Linguistics, 2006, s. 74–81.

[14] Li, X.; Roth, D.: Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, Association for Computational Linguistics, 2002, s. 1–7.

[15] Mintz, M.; Bills, S.; Snow, R.; aj.: Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, Association for Computational Linguistics, 2009, s. 1003–1011.

[16] Monz, C.: From document retrieval to question answering. 2003.

[17] Natural Language Computing Group, M. R. A.: *R-NET: MACHINE READING COMPREHENSION WITH SELF-MATCHING NETWORKS*. [Online; navštíveno 01.09.2017].
URL https://www.microsoft.com/en-us/research/wp-content/uploads/2017/05/r-net.pdf

[18] Networks, N.; MIPT, D. L. L. .: *DeepPavlov*. [Online; navštíveno 01.09.2017].
URL https://github.com/deepmipt/DeepPavlov

[19] Ohta, T.; Tateisi, Y.; Kim, J.-D.: The GENIA corpus: An annotated research abstract corpus in molecular biology domain. In *Proceedings of the second international conference on Human Language Technology Research*, Morgan Kaufmann Publishers Inc., 2002, s. 82–86.

[20] Pennington, J.; Socher, R.; Manning, C.: Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, s. 1532–1543.

[21] Rajpurkar, P.; Zhang, J.; Lopyrev, K.; aj.: SQuAD: 100, 000+ Questions for Machine Comprehension of Text. *CoRR*, ročník abs/1606.05250, 2016, 1606.05250.
URL http://arxiv.org/abs/1606.05250

[22] Rau, L. F.: Extracting company names from text. In *Artificial Intelligence Applications, 1991. Proceedings., Seventh IEEE Conference on*, ročník 1, IEEE, 1991, s. 29–32.

[23] Srivastava, N.; Hinton, G.; Krizhevsky, A.; aj.: Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, ročník 15, č. 1, 2014: s. 1929–1958.

[24] Wikipedia contributors: Bag-of-words model — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Bag-of-words_model&oldid=832576977, 2018, [Online; accessed 15-May-2018].

[25] Wikipedia contributors: GRPC — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=GRPC&oldid=839436625, 2018, [Online; accessed 15-May-2018].

[26] Wikipedia contributors: Polling (computer science) — Wikipedia, The Free Encyclopedia. 2018, [Online; accessed 17-May-2018].
URL https://en.wikipedia.org/w/index.php?title=Polling_(computer_science)&oldid=840161305

[27] Wikipedia contributors: Standard Boolean model — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Standard_Boolean_model&oldid=824761769, 2018, [Online; accessed 14-May-2018].

[28] Wikipedia contributors: Tf–idf — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Tf%E2%80%93idf&oldid=841608474, 2018, [Online; accessed 13-May-2018].

[29] Wikipedia contributors: Webhook — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Webhook&oldid=833510949, 2018, [Online; accessed 15-May-2018].

[30] wzhouad: *Tensorflow Implementation of R-Net*. [Online; navštíveno 01.09.2017].
URL https://github.com/HKUST-KnowComp/R-Net

Príloha A

Dialogflow datové typy

```
{
  "name": string,
  "displayName": string,
  "webhookState": enum(WebhookState),
  "priority": number,
  "isFallback": boolean,
  "mlDisabled": boolean,
  "inputContextNames": [
    string
  ],
  "events": [
    string
  ],
  "trainingPhrases": [
    object(TrainingPhrase)
  ],
  "action": string,
  "outputContexts": [
    object(Context)
  ],
  "resetContexts": boolean,
  "parameters": [
    object(Parameter)
  ],
  "messages": [
    object(Message)
  ],
  "defaultResponsePlatforms": [
    enum(Platform)
  ],
  "rootFollowupIntentName": string,
  "parentFollowupIntentName": string,
  "followupIntentInfo": [
    object(FollowupIntentInfo)
  ]
}
```

Štruktúra A.1: Objekt `Intent`; Definícia polí: A.1

```
{
  "name": string,
  "type": enum(Type),
  "parts": [
    object(Part)
  ],
  "timesAddedCount": number
}
```

Štruktúra A.2: Objekt `TrainingPhrase` Definícia polí A.2

```
{
  "text": string,
  "entityType": string,
  "alias": string,
  "userDefined": boolean
}
```

Štruktúra A.3: Objekt `Part` Definícia polí A.3

```
{
  "name": string,
  "lifespanCount": number,
  "parameters": {
    object
  }
}
```

Štruktúra A.4: Objekt `Context` Definícia polí A.5

```
  {
    "name": string,
    "displayName": string,
    "value": string,
    "defaultValue": string,
    "entityTypeDisplayName": string,
    "mandatory": boolean,
    "prompts": [
      string
    ],
    "isList": boolean
  }
```

Štruktúra A.5: Objekt `Parameter` Definícia polí A.6

```
{
  "platform": enum(Platform),
  text": {
    object(Text)
  }
}
```

Štruktúra A.6: Objekt `Message` Definícia polí A.7

```
{
  "followupIntentName": string,
  "parentFollowupIntentName": string
}
```

Štruktúra A.7: Objekt `FollowupIntentInfo` Definícia polí A.9

```
{
  "text": [
    string
  ]
}
```

Štruktúra A.8: Objekt `Text` Definícia polí A.12

```
{
  "name": string,
  "displayName": string,
  "kind": enum(Kind),
  "autoExpansionMode": enum(AutoExpansionMode),
  "entities": [
    object(Entity)
  ]
}
```

Štruktúra A.9: Objekt `EntityType` Definícia polí A.13

```
  {
    "value": string,
    "synonyms": [
      string
    ]
  }
```

Štruktúra A.10: Objekt `Entity` Definícia polí A.16

```
{
"queryText": string,
"languageCode": string,
"speechRecognitionConfidence": number,
"action": string,
"parameters": {
object
},
"allRequiredParamsPresent": boolean,
"fulfillmentText": string,
"fulfillmentMessages": [
{
object(Message)
}
],
"webhookSource": string,
"webhookPayload": {
object
},
"outputContexts": [
{
object(Context)
}
],
"intent": {
object(Intent)
},
"intentDetectionConfidence": number,
"diagnosticInfo": {
object
}
}
```

Štruktúra A.11: Objekt `QueryResult` Definícia polí A.17

name	**string** Povinné pre všetky metódy okrem create (pri vytváraní sa meno vytvori automaticky). Unikátny identifikátor. Formát: projects/<Project ID>/agent/intents/<Intent ID>.
displayName	**string** Povinné. Meno zámeru.
webhookState	**enum (WebhookState)** Povinné. Indikuje či má byť daný zámer uspokojený pomocou webhooku.
priority	**number** Voliteľné. Priorita zámeru. Vyššie čislo indikuje vyššiu prioritu . Záporne čislo alebo nula značí že zámer je neaktívny.
isFallback	**boolean** Voliteľné. Indikuje či je tento zámer záložný.
mlDisabled	**boolean** Voliteľné. Indikuje, či je pre tento zámer povolené strojové učenie.
inputContextNames[]	**string** Voliteľné. Zoznam kontextov ktoré musia byť aktívne aby mohol byť tento zámer detekovaný. Formát: projects/<Project ID>/agent/sessions/-/contexts/<Context ID>.
events[]	**string** Voliteľné. Kolekcia mien udalostí ktoré spustia tento zámer.
trainingPhrases[]	**object(TrainingPhrase)** Voliteľné. Kolekcia názornych viet a vzorov na ktorej sa agent bude trénovať.
action	**string** Voliteľné. Meno akcie spojené so zámerom.
outputContexts[]	**object(Context)** Voliteľné. Kolekcia kontextov ktoré sa nastavia ak bude tento zámer detekovaný. Formát: projects/<Project ID>/agent/sessions/-/contexts/<Context ID>.
resetContexts	**boolean** Voliteľné. Resetuje všetky kontexty ak je detekovaný tento zámer.
parameters[]	**object(Parameter)** Voliteľné. Kolekcia parametrov/slotov tohoto zámeru.
messages[]	**object(Message)** Voliteľné. Kolekcia formátovaných správ.
defaultResponsePlatforms[]	**enum (Platform)**Voliteľné. Zoznam platforiem pre ktoré bude odpoved vybraná z odpovedí priradených DEFAULT_PLATFORM.
rootFollowupIntentName	**string** Jedinečný identifikátor koreňového zámeru zo série navazujúcich zámerov. Identifikuje správnu postupnosť navazujúcich zámerov pre tento zámer. Formát: projects/<Project ID>/agent/intents/<Intent ID>.
parentFollowupIntentName	**string** Unikátny identifikátor rodičovského zámeru v postupnosti naväzujúcich zámerov. Identifikuje rodičovský naväzujúci zámer. Formát: projects/<Project ID>/agent/intents/<Intent ID>.
followupIntentInfo[]	**object(FollowupIntentInfo)** Voliteľné. Kolekcia informácii o naväzujúcich zámeroch, ktoré majú tento zámer ako svoj root_name.

Tabuľka A.1: Objekt `Intent`

name	**string** Povinné. Unikátny identifikátor tejto trénovacej vety.
type	**enum (Type)**Povinné. Typ vety.
parts[]	**object(Part))**Povinné. Kolekcia častí trénovacej vety (môžu byť anotované). Polia: entityType, alias a userDefined by mali byť vyplnené iba pre anotované časti vety.
timesAddedCount	**number** Voliteľné. Indikuje koľko krát bol tento príklad pridaný do zámeru.

Tabuľka A.2: Objekt `TrainingPhrase`

text	**string** Povinné. Text korešpondujuci príkladu v prípade, že sa v ňom nenachádzajú anotácie. Pre anotované príklady je to text jednej časti vety.
entityType	**string** Voliteľné. Meno typu entity prefixované znakom @. Toto pole je povinné pre anotované časti a vzťahuje sa iba na príklady.
alias	**string** Voliteľné. Meno parametru pre hodnotu extrahovanú z anotovanej časti príkladu.
userDefined	**boolean** Voliteľné. Indikuje či bol text mauálne anotovaný vývojárom.

Tabuľka A.3: Objekt `Part`

WEBHOOK_STATE_UNSPECIFIED	Webhook je neaktívny pre daného agenta a zámer.
WEBHOOK_STATE_ENABLED	Webhook je aktívny pre daného agenta a zámer.
WEBHOOK_STATE_ENABLED_FOR_SLOT_FILLING	Webhook je aktívny pre daného agenta a zámer. Zároveň je každá výzva na vyplnenie povinného parametra presmerovaná.

Tabuľka A.4: Výčtový typ WebhookState

name	**string** Povinné. Unikátny identifikátor kontextu. Formát: projects/<Project ID>/agent/sessions/<Session ID>/contexts/<Context ID>.
lifespanCount	**number** Volitelné. Počet konverzačných dotazov po ktorých kontext expiruje. Ak je nastavený na 0 (predvolené), kontext expiruje okamžite. Kontexty expirujú automaticky po desiatich minútach bez toho, aby prišiel dotaz.
parameters	**object (Struct)** Volitelné. Kolekcia parametrov spojená s týmto kontextom.

Tabuľka A.5: Objekt `Context`

name	**string** Unikátny identifikátor tohoto parametra.
displayName	**string** Povinné. Meno parametra.
value	**string** Volitelné. Definícia hodnoty parametra. Môže to byť: - konštantný reťazec, - hodnota parametra definovaná ako $parameter_name, - pôvodná hodnota parametra definovaná ako $parameter_name.original, - hodnota parametra z kontextu definovaná ako #context_name.parameter_name.
defaultValue	**string** Volitelné. Predvolená hodnota ktorá buďe použitá v prípade že by hodnota bola prázdna. Predvolené hodnoty môžu byt extrahované z kontextu s použitím syntaxe: #context_name.parameter_name.
entityTypeDisplayName	**string** Voliteľné. Meno typuentity prefixované znakom @, ktoré opisuje hodnoty parametra. Musí byť poskytnuté, ak je parameter povinný.
mandatory	**boolean** Volitelné. Indikuje či je parameter povinný. Teda či može byť zámer kompletný bez získania hodnoty tohoto parametra.
prompts[]	**string** Volitelné. Kolekcia výziev ktoré môže agent predložiť uživateľovi aby získal hodnotu parametra.
isList	**boolean** Volitelné. Indikuje ,či parameter predstavuje zoznam hodnôt.

Tabuľka A.6: Objekt `Parameter`

platform	**enum(Platform)** Volitelné. Platforma pre ktorú je správa určená.
text	**object(Text)**Textová odpoveď.

Tabuľka A.7: Objekt `Message`

PLATFORM_UNSPECIFIED	Nešpecifikované.
FACEBOOK	Facebook.
SLACK	Slack.
TELEGRAM	Telegram.
KIK	Kik.
SKYPE	Skype.
LINE	Line.
VIBER	Viber.

Tabuľka A.8: Výčtový typ `Platform`

followupIntentName	**string** Unikátny identifikátor naväzujúceho zámeru. Formát: projects/<Project ID>/agent/intents/<Intent ID>.
parentFollowupIntentName	**string** Unikátny identifikátor rodiča naväzujúceho zámeru. Formát: projects/<Project ID>/agent/intents/<Intent ID>.

Tabuľka A.9: Objekt `FollowupIntentInfo`

TYPE_UNSPECIFIED	Nešpecifikované. Táto hodnota by sa nikdy nemala použiť.
EXAMPLE	Príklady neobsahujú mená typov entít prefoxované znakom @, ale časti príkladu môžu byť anotované typmi entít.
TEMPLATE	Šablóny nie su anotované typmi entít, ale obsahujú mená typov entít prefixované znakom @ ako podreťazce.

Tabuľka A.10: Výčtový typ `Type`

fields	**map<string,value>** Asociatívne pole dynamicky typovaných hodnôt. .

Tabuľka A.11: objekt `Struct`

text[]	**string** Voliteľné. Kolekcia odpovedí agenta.

Tabuľka A.12: Objekt `Text`

name	**string** Povinné pre všetky metódy okrem create (create vytvorí identifikátor automaticky). Unikátny identifikátor typu entity. Formát: projects/<Project ID>/agent/entityTypes/<Entity Type ID>.
displayName	**string** Povinné. Meno entity.
kind	**enum(Kind)** Povinné. Indikuje druh typu entity.
autoExpansionMode	**enum(AutoExpansionMode)** Volitelné. Indikuje či môže byť entita automaticky expandovaná.
entities[]	**object(Entity)** Volitelné. Kolekcia hodnôt entít spojená s daným typom entity.

Tabuľka A.13: Objekt `EntityType`

KIND_UNSPECIFIED	Nešpecifikované. Táto hodnota by nemala byť nikdy použitá.
KIND_MAP	Umožňuje mapovanie skupiny synoným na kanonickú hodnotu.
KIND_LIST	Zoznamové typy entít obsahujú množinu záznamov ktoré nie su mapované na kanonickú hodnotu. Môžu však obsahovať odkazy na iné typy entít.

Tabuľka A.14: Výčtový typ `Kind`

AUTO_EXPANSION_MODE_UNSPECIFIED	Automatická expanzia vypnutá.
AUTO_EXPANSION_MODE_DEFAULT	Dovoľuje agentovi rozpoznať hodnoty ktoré neboli pre danú entitu explicitne uvedené.

Tabuľka A.15: Výčtový typ `AutoExpansionMode`

value	**string** Povinné. Pre KIND_MAP typy entít: Kanonické meno ktoré bude použité namiesto synoným. Pre entity typu KIND_LIST: Reťazec ktorý môže osahovať odkazy na iné typy entít.
synonyms[]	**string** Povinné. Kolekcia synoným. Pre entity typu KIND_LIST musí obsahovať práve jedno synymum rovné poľu value.

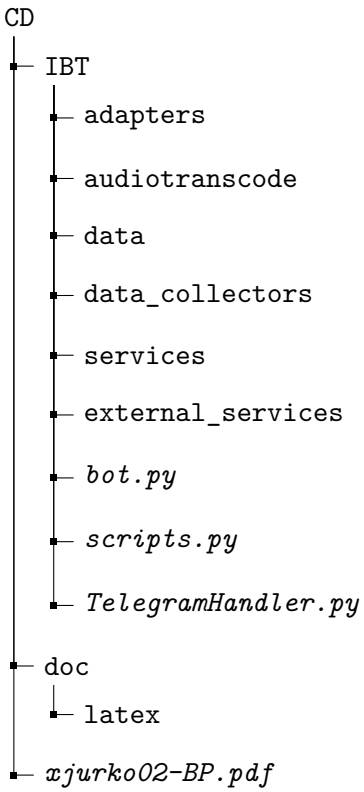
Tabuľka A.16: Objekt `Entity`

queryText	**string** Pôvodný text konverzačného dotazu: - Ak bol vstupom text v prirodzenom jazyku , queryText obsahuje kópiu vstupu. - Ak bola vstupom zvuková stopa v prirodzenom jazyku, queryText obsahuje prepis reči. Ak rozpoznávač reči vygeneroval viac výstupov je zvolený jeden konkrétny.
languageCode	**string** Detekovaný jazyk vstupu.
speechRecognitionConfidence	**number**Istota rozpoznávača reči medzi 0.0 a 1.0. Vyššie číslo znamená odhadovanú vyššiu pravdepodobnosť, že sú slová rozpoznané správne. Predvolená hodnota 0.0 slúži ako zarážka a indikuje že pole nebolo nastavené.
action	**string** Akcia spojená s daným zámerom v prípade zhody.
parameters	**object** (**Struct**)Kolekcia extrahovaných parametrov.
allRequiredParamsPresent	**boolean**Toto pole je nastavené na: - false ak detekovaný zámer má povinné parametre ale nie všetky majú nastavenú hodnotu. - true Ak všetky povinné parametre majú nastavenú hodnotu alebo ak zámer nemá žiadne povinné parametre.
fulfillmentText	**string** Text ktorý má byt uživateľovi prezentovaný.
fulfillmentMessages[]	**object**(**Message**) Kolekcia správ ktoré môžu byť predané uživateľovi.
webhookSource	**string** Ak bol dotaz uspokojený pomocou webhooku, toto pole obsahuje hodnotu ktorú vrátil webhook.
webhookPayload	**object** (**Struct**) Ak bol dotaz uspokojený pomocou webhooku, toto pole obsahuje hodnotu ktorú vrátil webhook v tele odpovede.
outputContexts[]	**object**(**Context**) Kolekcia výstupných kontextov. Ak je to možné outputContexts.parameters obsahuje záznamy s menom <parameter name>.original ktoré obsahujú pôvodné hodnoty pred spracovaním dotazu.
intent	**object**(**Intent**) Zámer ktorý bol detekovaný v konverzačnom dotaze. Nie všetky hodnoty sú naplnené. Medzi hodnoty ktoré budu neprázdne patria name, displayName and webhookState.
intentDetectionConfidence	**number** Istota detekcie zámeru. Hodnoty sa pohybujú v rozsahu 0.0 (úplna neistota) do 1.0 (úplna istota).
diagnosticInfo	**object** (**Struct**) Diagnostické informácie vo voľnej forme.

Tabuľka A.17: Objekt `QueryResult`

Príloha B

Obsah priloženého média



Adresár IBT obsahuje zdrojové súbory agenta.