TakeLab

Laboratorij za analizu teksta i inženjerstvo znanja
Text Analysis and Knowledge Engineering Lab
Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva
Unska 3, 10000 Zagreb, Hrvatska

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 5325

Predviđanje konteksta pitanja za interaktivno jezično sučelje bazi podataka

Juraj Malenica

Zagreb, lipanj 2017.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 3. ožujka 2017.

ZAVRŠNI ZADATAK br. 5325

Pristupnik: **Juraj Malenica (0036484713)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Predviđanje konteksta pitanja za interaktivno jezično sučelje bazi podataka**

Opis zadatka:

Jezično sučelje bazama podataka omogućava postavljanje upita nad bazom u kontroliranom prirodnom jeziku. Dodatnu fleksibilnost pruža mogućnost interaktivnog postavljanja upita, kod kojega se pojedini upiti nadovezuju na prethodne upite, nasljeđujući ili mijenjajući kontekst uspostavljen prethodnim upitima. Ključni korak kod interaktivnog odgovaranja na pitanja u prirodnome jeziku jest utvrditi otvara li upit novi kontekst ili se nadovezuje na postojeći. Ako se pitanje nadovezuje, potrebno je nadopuniti ili ažurirati postojeći kontekst.
Tema završnoga rada jest predikcija konteksta pitanja za interaktivno odgovaranje na pitanja nad bazom podataka o poznatim osobama na engleskome jeziku. Osmisliti postupak koji će za pitanja korisnika odrediti nadovezuje li se pitanje na prethodni kontekst na temelju analize pitanja i informacije o trenutačnom kontekstu te u slučaju višeznačnosti postaviti dodatno pitanje korisniku. Pretpostaviti da su pitanja semantički obrađena, tj. da je poznata vrsta pitanja i da su iz pitanja ekstrahirane relevantne informacije potrebne za generiranje upita nad bazom podataka. Izgraditi prikladnu ispitnu bazu semantički obrađenih pitanja u prirodnome jeziku grupiranih u interaktivne sjednice. Provesti vrednovanje modela, usporedbe s referentnim modelom, statističku obradu rezultata te analizu pogrešaka. Radu priložiti izvorni i izvršni kod razvijenog sustava, označene skupove podataka i potrebnu dokumentaciju te citirati korištenu literaturu.

Zadatak uručen pristupniku: 10. ožujka 2017.
Rok za predaju rada: 9. lipnja 2017.

Mentor:

Izv. prof. dr. sc. Jan Šnajder

Djelovođa:

Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za završni rad modula:

Prof. dr. sc. Siniša Srbljić

Zahvaljujem kolegama iz Vingd d.o.o. na savjetima, Franu i Ivanu na odličnoj suradnji te obitelji i djevojci na potpori.

Sadržaj

Popis slika

Popis tablica

1. Uvod

Kontinuiranim napretkom tehnologije dolazi do ubrzane proizvodnje digitalnih podataka. Od 2012. godine, dnevno se proizvede 2.5 eksabajta podataka (10^{18} bajta) i ta se brojka se udvostručuje otprilike svakih 40 mjeseci (McAfee et al., 2012). Kako bi svi mogli koristiti podatke iz javno dostupnih baza podataka, potrebno je razviti jednostavnije sučelje koje neće zahtijevati detaljno znanje strukture baze podataka kao ni pisanja upita na bazu. Interakcija s tim sučeljem trebala bi podsjećati na stvarni razgovor, čiji veliki dio je razumijevanje konteksta. Također, današnja sučelja su ili prekompleksna ili nedovoljno učinkovita. Potencijalno rješenje tog problema je u korištenju konteksta za modularizaciju upita.

Cilj ovog završnog rada – zajedno sa završnim radovima kolega Frana Arbanasa i Ivana Mršića – jest napraviti interaktivno jezično sučelje bazi podataka. Kolege su metodama obrade prirodnog jezika (engl. *Natural Language Processing*) dali strukturu upitima korisnika. Ovaj rad se bavi metodom izgradnje baze znanja željenog ponašanja pri korištenju modela kao i pronalaskom ispravnog tumačenja upita korisnika za dani kontekst. Rad će biti konkretno primijenjen na bazu filmova i glumaca koja će biti dostupna kroz sustav *Neo4j*.[1]

Rad je podijeljen u četiri logičke cjeline: (1) *Pregled područja*, (2) *Model za određivanje konteksta pitanja*, (3) *Implementacija* i (4) *Primjeri uporabe*. *Pregled područja* daje osvrt na postojeće radove u području određivanja konteksta i pronalaska ispravnog tumačenja upita. *Model za određivanje konteksta pitanja* ulazi u detalje primijenjenog pristupa pri rješavanju problema na apstraktnoj razini. U poglavlju *Implementacija* detaljno se obrađuje programski pristup rješavanju problema kroz opise pojedinih klasa i njihovog uklapanja u cjelinu. Konačno, poglavlje *Primjeri uporabe* navodi stvarne primjere uporabe modela i demonstrira njegove najvažnije funkcionalnosti.

[1] `https://neo4j.com/`

2. Pregled područja

Kontekst je vrlo važan jer daje informacije o trenutnom stanju ljudi, mjesta, stvari i uređaja u okolini. Kontekst je bilo kakva informacija kojom se može okarakterizirati situacija o entitetu. Entitet je osoba, mjesto ili objekt koji se smatra relevantnim za interakciju između korisnika i aplikacije, uključujući lokaciju, vrijeme, aktivnosti i preference svakog entiteta (Dey, 2001). Osviještenost o kontekstu (engl. *context-awareness*) znači moći koristiti informacije iz konteksta. Sustav je osviješten o kontekstu ako može izvući, protumačiti i iskoristiti informaciju iz konteksta te prilagoditi njenu funkcionalnost trenutnom korištenom kontekstu (Byun i Cheverst, 2004).

S obzirom na podatke iz 237 radova o sustavima osviještenima o kontekstu, određena je opća struktura arhitekture jednog takvog sustava koja se sastoji od četiri sloja: (1) mrežni sloj (engl. *network layer*) sačinjava mrežu koja podržava sustav osivješten o kontekstu i senzore za skupljanje kontekstnih informacija, (2) *middleware* sprema kontekstne informacije te upravlja procesima, (3) aplikacijski sloj korisniku daje odgovaraju-u aplikaciju i (4) korisnička infrastruktura (Hong et al., 2009).

Prema (Brézillon, 2011), postoji pet pristupa za rješavanje problema donošenja odluka koji su se kroz povijest pojavili u području umjetne inteligencije. Svih pet je detaljnije opisano u nastavku.

2.1. Ekspertni sustavi

Ekspertni sustavi nastali su prebacivanjem fokusa s algoritamskih rješenja na rješenja temeljena na činjenicama i pravilima. Sustav se temelji na principu *ako–onda*, gdje svi uvjeti pravila moraju biti ispunjena kako bi se hipoteza smatrala ispravnom. Prema (Brézillon, 2011), problem nastaje s pravilima koja nisu eksplicitno zadana.

Primjerice, ako postoji pravilo “Da bi glumac bio dobar, mora imati barem 10

godina iskustva", znači li to da glumci koji imaju 9 godina i 11 mjeseci iskustva nisu dobri glumci?

2.2. Zajednički kognitivni sustav

Zajednički kognitivni sustav (engl. *joint cognitive system*) nastao je kako bi se naglasila činjenica da niti jedan sustav i niti jedan korisnik samostalno ne mogu riješiti problem (Woods, 1985). Implicitno, najbolje odluke će biti donesene ako se sustav i korisnik razumiju i zajednički donose odluke. Kako bi se stvorila uspješna komunikacija, naglasak pri izgradnji sustava je na: dijeljenju kognitivnih reprezentacija, mogućnosti pratiti zaključivanje jedan drugoga, mogućnosti izmjenjivanja objašnjenja, mogućnosti učenja od drugoga i dijeljenja kontrole interakcije. Brézillon (2011) navodi kako je interes u zajedničkim kognitivnim sustavima na određivanju trenutnog stanja i konteksta za akciju – koje je sljedeće stanje.

2.3. Inteligentni sustavi

Teza inteligentnih sustava navodi da se potrebe korisnika predviđaju poznavanjem konteksta (poput trenutne lokacije, vremena i slično) i statističkom analizom. Međutim, prema (Brézillon, 2011), to nije dobar pristup pri razlučivanju jer sustav i korisnik pri razlučivanju nisu istovjetni. Promjene potrebne za inteligente sustave da bi bili korisni korisniku su takve da se oni više ne bi smatrali inteligentnim sustavima nego inteligentnim asistentima. Primjeri nekih promjena su: (1) davanje aproksimacije trendova i događaja u okolini, (2) isticanje korisnih informacija u velikim količinama podataka i (3) privlačenje korisnikove pažnje na postojeće i nadolazeće strateške probleme.

2.4. Sustavi inteligentnih asistenata

Sustav inteligentnog asistenta (IAS) ima dvije strane: stranu procesa i ljudsku stranu. IAS mora razumjeti: (1) stvarne procese, (2) zadatak koji treba riješiti i (3) ponašanje korisnika. IAS može naučiti promatrajući ponašanje korisnika i procesa. To je inkrementalni pristup učenju gdje sustav uči kada je to potrebno i iznad svega u kontekstu u kojem se koristi.

Ovisno o korisnikovom iskustvu, prema (Brézillon, 2011) interakcija se može provesti na dva načina: (1) tihi način, ako ne postoji odstupanje između ponašanja korisnika i ponašanja sustava te (2) kroz suradnju, ako postoji odstupanje u ponašanju korisnika ili budućim posljedicama.

Generalno, postoje različiti pristupi za izvođenje zadataka i izbor metode ponajviše se oslanja na poznavanje konteksta i ekspertizu korisnika. Očito IAS mora biti formaliziran u reprezentaciji znanja, zaključivanja i konteksta. Također, formalizacija mora dopustiti mogućnost inkrementalnog učenja i učenja vježbom (Brézillon, 2011).

2.5. Sustavi inteligentnih asistenata temeljenih na kontekstu

Relacija $ist(c, p)$ (propozicija p je istinita u kontekstu c) prvi put je uvedena u području umjetne inteligencije s McCarthy (1993) kao njenim glavnim zagovornikom. Kao posljedicu, McCarthy je pokazao da:

- kontekst je uvijek relativan prema drugom kontekstu,
- kontekst ima beskonačno dimenzija,
- kontekst ne može biti u potpunosti opisan i
- kada dođe do konflikta više konteksta, postoji jedan zajednički kontekst iznad svih njih.

Danas postoji konceptualni okvir za modeliranje konteksta i njegovo upravljanje te je njegova implementacija izvedena kroz softver zvan „Kontekstualni grafovi" (engl. *Contextual graphs*). Radna definicija konteksta je „Kontekst ograničava fokus bez eksplicitnog uplitanja". Kao posljedica, (1) kontekst je relativan fokusu, (2) kako fokus evolvira, tako evolvira i kontekst, (3) kontekst je izrazito ovisan o domeni. Povezivanjem konteksta na fokus moguće je zaključiti da fokus dijeli kontekst u eksterno znanje i kontekstualno znanje.

Kontekstualni se grafovi koriste u nekoliko stvarnih aplikacija u različitim domenama. Kontekstualni grafovi su osnova u izgradnji inteligentnih asistenata temeljenima na iskustvu (Brézillon, 2011).

3. Skup podataka

Kako bi se model mogao izgraditi i testirati, trebalo je navesti različite moguće scenarije uporabe modela. Skup podataka dijeli se na originalne upite koji se šalju cijelom modelu te standardizirane upite koji proizlaze iz originalnih, ali su standardizirani za potrebe ovog završnog rada. Upiti su pisani na temelju podatka koji se nalaze u bazi*Neo4j* i sadrže informacije o glumcima te filmovima. *Neo4j* je baza grafova i ima veliku primjenu u spremanju podataka bez da se gubi na semantičkom značenju (Miller, 2013). Baza se sastoji od preko 50000 glumaca i 12000 filmova. Skup podataka se temelji na bazi *Neo4j* glumaca i filmova jer sadrži dovoljno podataka i jer je tematika privlačna mnogim ljudima. Pisanje različitih scenarija napravljeno je ručno u suradnji s kolegama Franom Arbanasom i Ivanom Mršićem te je napisano preko 350 scenarija. Scenariji su podjeljeni u grupe upita s prosječnom duljinom od pet upita. Također, trebalo je paziti da prvi upit u sljedećoj grupi upita bude logički nespojiv s prethodnom grupom – time se stvara potreba za određivanjem novog scenarija. Tako primjerice možemo započeti scenarij s upitom *Give me all actors from the US* i nastaviti s upitima poput *Add those from Germany* i *Leave only those older than* 40. Tada bi prvi upit u sljedećoj grupi trebao biti poput upita *Give me actors from movie Inception.* Očigledno zadnji upit nije dio postojećeg scenarija.

4. Model za određivanje konteksta pitanja

Model za određivanje konteksta pitanja se temelji na izgradnji stabla poznatih scenarija. U fazi pripreme modela za korištenje, ekspert za područje u kojem će se model koristiti treba proći kroz vjerojatne scenarije upotrebe. To znači da ekspert simulira korištenje modela kako bi ga koristili drugi korisnici i uči model kako se dani upit korisnika uklapa u cjelinu. Tako ekspert određuje kontekst te situacije. Konteksti se nadovezuju jedan na drugi i stvaraju scenarije korištenja koji se onda spremaju u stablo. U fazi korištenja modela ideja je da se korisnik susretne sa što manje nepoznatih scenarija kako bi nesmetano mogao raditi upite. U slučaju da korisnik prouzroči nepoznati scenarij upitima koje ekspert nije prošao, model će i tada nastaviti učiti kroz postavljanje pitanja korisniku.

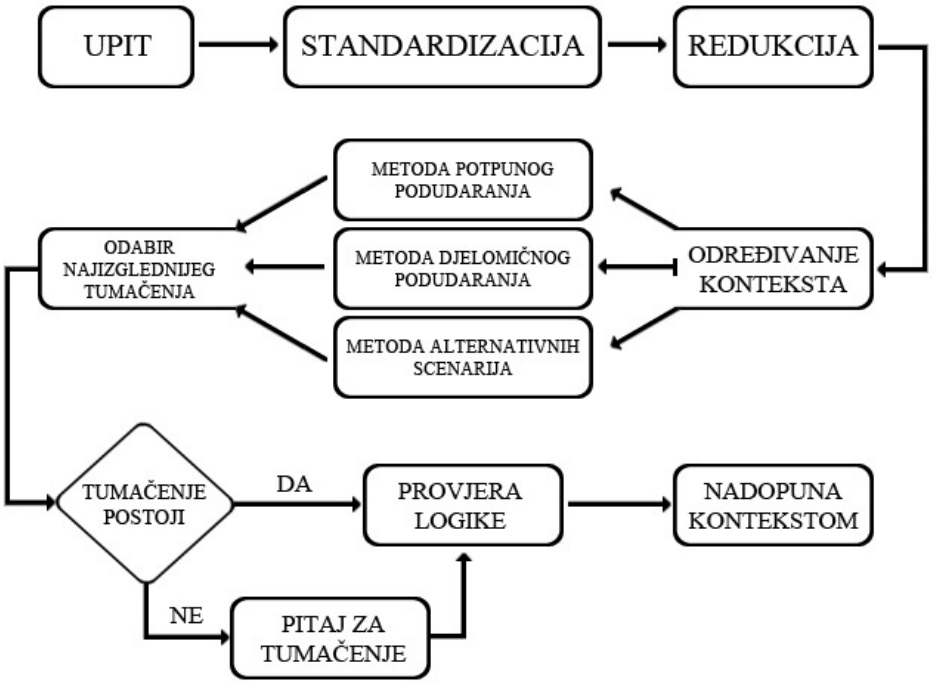
Na slici 4.1 je prikazan tok cijelog modela na apstraktnoj razini.

4.1. Standardizirani upit

Osnovna jedinica modela za određivanje konteksta je standardizirani upit (engl. *query*).[1] Korisnik šalje upit kroz interaktivno jezično sučelje u čijoj pozadini se određuju tipovi i vrijednosti namjere (engl. *intent*) i entiteta (engl. *entity*) tog upita. Tipovi i vrijednosti namjere i entiteta zajedno čine standardizirani upit. Moguće vrijednosti namjere i entiteta su predodređene mogućim vrijednostima u bazi podataka. Tipovi namjere određeni su mogućnostima baze podataka i navedeni su u tablici 4.1. Tipovi entiteta ograničeni su vrijednostima iz baze podataka i prikazani su u tablici 4.2.

Primjerice, za upit *Give me all actors from Croatia*, namjera je tipa *WHERE* s vrijednosti *POSITIVE*, entitet za pretraživanje je tipa *COUNTRY* s vrijednosti

[1] Implementaciju standardizacije upita rade kolege Arbanas i Mršić



Slika 4.1: Apstraktni prikaz ciljanog modela.

Croatia, traženi entitet je *PERSON* i poveznica (engl. *relationship*) je *LIVES_IN*. Te vrijednosti zajedno predstavljaju standardizirani upit.

Pojednostavljeni tip standardiziranog upita je reducirani upit. Reducirani upit predstavlja standardizirani upit gdje su poznati tipovi i vrijednosti namjera te tip entiteta, ali je vrijednost entiteta obrisana. U prethodnom primjeru bi vrijednost *Croatia* bila obrisana dok bi ostale vrijednosti bile spremljene.

Reducirani oblik upita je koristan pri uspoređivanju upita jer se uspoređuju samo bitne vrijednosti. Tako bi primjerice upit *Give me all actors from Croatia* i upit *Give me all actors from US* prilikom usporedbe bili jednaki jer su njihovi reducirani oblici jednaki.

4.2. Kontekst

Kontekst (engl. *context*) je logička jedinica koja se temelji na standardiziranom upitu. Korisnik slanjem upita navodi svoju namjeru. Ovisno o tome koliko je bio detaljan u svome upitu, prilikom standardizacije se definira jedno ili više moguće tumačenje entiteta i namjera. Zbog te nesigurnosti, prilikom standardizacije se također određuju i vjerojatnosti ispravnog tumačenja za svaku mogućnost. Model

Tablica 4.1: Struktura namjere standardiziranog upita

Tip	Vrijednost
WHERE	POSITIVE, NEGATIVE, ∅
OPERATOR	UNION, REDUCE, ∅
AGGREGATION	COUNT, MAX, MIN, SUM, AVG, ∅
DISTINCT	*True*, *False*, ∅
RELATIONSHIP	STARRED_IN, LIVES_IN, WRITTEN_BY, ∅
TOP	*True*, *False*, ∅
ORDER_BY	ASCENDING, DESCENDING, ∅

na temelju već poznatih vrijednosti koje je skupio kroz interakciju s korisnikom ili na temelju ekspertnog znanja pokušava nadopuniti tumačenja upita i jednoznačno odrediti ciljano tumačenje. Ako ne uspije, model pita korisnika da izabere između najvjerojatnijih opcija tumačenja upita ili da sam upiše alternativno.

Primjerice, nakon upita *Give me all actors from Croatia* u kontekstu je zapisano da je korisnik dodao glumce prema parametru *country*. Zato će znati sljedeći nadopuniti upit *And from Germany* koji bi bez poznavanja konteksta bio nedovoljno određen. Ne bi bilo jasno želi li korisnik dodati glumce iz Njemačke ili ih maknuti, ali poznavanjem konteksta se može zaključiti da ih želi dodati.

4.3. Scenarij

Scenarij (engl. *scenario*) se sastoji od niza konteksta poredanih kronološki prema redoslijedu slanja upita. Na početku interakcije korisnika i modela scenarij predstavlja niz koji se sastoji od praznog konteksta. Svakim upitom se scenarij proširuje kontekstom koji ispravno predstavlja korisnikovu namjeru. Korisnik može odrediti da kontekst poslanog upita nije dio trenutnog scenarija te tada postojeći scenarij završava i započinje novi koji sadrži taj kontekst.

Trenutni kontekst može pristupiti prethodnim kontekstima unutar svog scenarija kako bi nadopunio tumačenja upita i točnije odredio ciljano tumačenje. Ako je kontekst u trenutnom scenariju poznat od prije – ranije ga je definirao ekspert ili neki drugi korisnik – kontekst može na temelju budućih konteksta odrediti najizglednije tumačenje upita i izbjeći potrebu za potvrdom tumačenja od strane korisnika.

Primjerice, ako je korisnik kroz interakciju s modelom u scenariju koji je mo-

Tablica 4.2: Struktura entiteta standardiziranog upita

Tip	Vrijednost
age	$[0 - 100]$
gender	male, female
movie	Taken, Batman begins, ...
rating	$[0 - 10]$
role	lead, supporting
nominations	$[0 - \infty]$
name	Tom Cruise, Ben Afflek
country	Croatia, US, ...
oscars	$[0 - \infty]$
maritial status	married, single
revenue	$[0 - \infty]$
continent	Europe, Asia, ...

delu poznat od prije i ako pošalje upit *Only over* 60, model može na temelju budućih (od prije definiranih) konteksta u trenutnom scenariju odrediti misli li korisnik na glumce starije od 60 godina ili na glumce sa zaradom većom od 60 milijuna dolara.

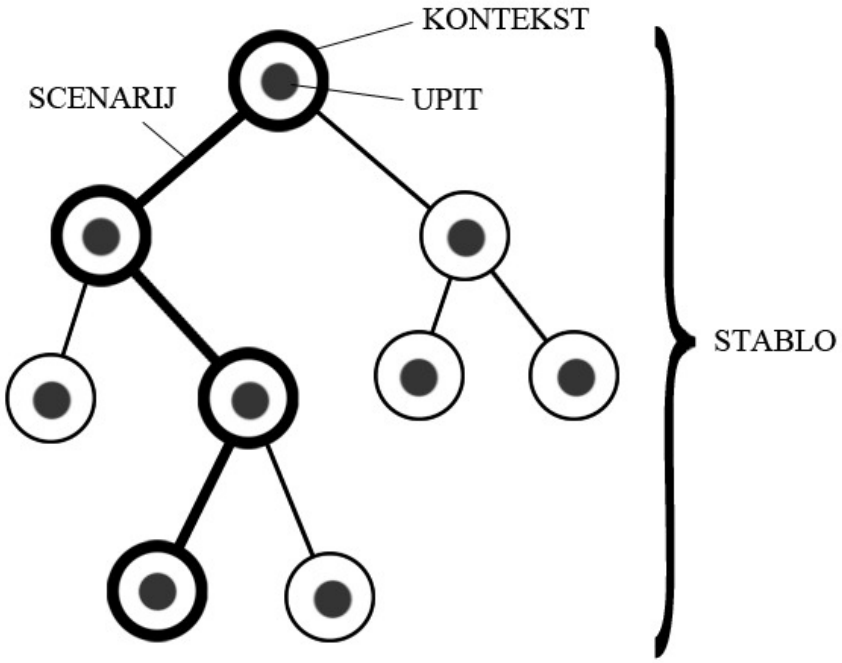
4.4. Stablo

Stablo (engl. *tree*) je struktura podataka često korištena u bazama podataka za optimiziranije operacije nad podacima. Implementacija stabla znači stvoriti odnos roditelj–dijete između dva čvora, te da svaki čvor ima samo jednu ulaznu poveznicu (jednog roditelja) (Graefe et al., 2017).

U ovom slučaju stabla svaki čvor predstavlja jedan kontekst dok niz čvorova od korijena do lista predstavlja jedan scenarij. Prijelaz između dva čvora je određen reduciranim upitom korisnika koji povezuje trenutni kontekst s novonastalim kontekstom. Svaki čvor je ujedno i korijen svog podstabla. Struktura stabla grafički je prikazana na slici 4.2.

Neka dva scenarija u stablu smatraju se jednakima ako se njihovi reducirani upiti međusobno slažu i ako se slažu po redoslijedu konteksta u scenariju.

Kako bi model bio prikladan u danom području za sve korisnike, potrebno je proširiti stablo. Interakcijom s ekspertom za dano područje model proširuje sta-



Slika 4.2: Logička struktura stabla

blo scenarija vjerodostojnim informacijama. Te informacije se odnose na odnose između upita koji slijede jedan iza drugoga.

Primjerice, ako u istom scenariju imamo upit *Older than 20* i poslije njega upit *Younger than 40*, moguće je naći više interpretacija drugog upita. On može poništiti prvi (engl. *replace*), upiti mogu biti neovisni jedan o drugom (engl. *or*) i mogu se nadopunjavati (engl. *and*). Također, prvi upit može poništiti drugi ako se korisnik predomislio. Uloga eksperta je odrediti potrebe korisnika u toj i sličnim situacijama.

4.5. Pronalazak vjerojatnog tumačenja upita

Korisnik šalje upit koji se standardizira i reducira. Prilikom standardizacije može nastati jedno ili više mogućih tumačenja upita. Pomoću trenutnog konteksta, trenutnog scenarija i cijelog stabla moguće je pronaći vjerojatno tumačenje upita iz ponuđenih opcija. Vjerojatno tumačenje se može pronaći na tri načina: (1) metoda potpunog podudaranja, (2) metoda djelomičnog podudaranja i (3) metoda alternativnih scenarija. Svaki način za neko tumačenje daje i koeficijent snage koji predstavlja kvalitetu tumačenja – koliko je vjerojatno to tumačenje. U

konačnici se kao najbolje tumačenje se uzima ono s najvećim koeficijentom.

4.5.1. Metoda potpunog podudaranja

Metoda potpunog podudaranja temelji se na prijelazima iz trenutnog konteksta u scenariju u buduće kontekste. Ti prijelazi moraju biti poznati od prije što znači da trenutni scenarij mora postojati u stablu od prije. Ovom metodom se uspoređuju trenutni reducirani upit i reducirani upiti u prijelazima do sljedećeg konteksta.

Koeficijent snage u metodi potpunog podudaranja određen je vrijednošću 1. Pretpostavka je da će metoda potpunog podudaranja vratiti najvjerojatnije tumačenje.

4.5.2. Metoda djelomičnog podudaranja

Metoda djelomično podudaranja također se temelji na prijelazima iz trenutnog konteksta u scenariju u buduće kontekste. Razlika je u tome što se ne moraju potpuno poklapati oba reducirana upita, jedan može biti podskup drugoga. Koeficijent snage se računa kao broj podudaranja s obzirom na maksimalni mogući broj podudaranja tipova u reduciranim upitima. Koeficijent se množi s konstantom 0.8 kako bi tumačenja dobivena metodom potpunog podudaranja imala prednost nad tumačenjima metode djelomičnog podudaranja. Cijeli izraz prikazan je u formuli (4.1).

$$koeficijentsnage = 0.8 \cdot \frac{brojpodudaranja}{maksimalnimogućibrojpodudaranja} \tag{4.1}$$

Primjerice, ako za dva reducirana izraza prvi izraz bude podskup drugoga i dijele tri od četiri vrijednosti, koeficijent snage će prema formuli 4.2 biti jednak $koeficijentsnage = 0.8 \cdot \frac{3}{4} = 0.6$.

4.5.3. Metoda alternativnog scenarija

Metoda alternativnog scenarija uzima sve reducirane upite od početka trenutnog scenarija do zadnjeg danog upita i pokušava naći druge scenarije takve da se pojavljuju svi isti reducirani upiti kao i u trenutnom, ali u drugom redoslijedu. To znači da u stablu mora postojati scenarij s istim skupom pitanja, u drugom

redoslijedu. Koeficijent snage u metodi alternativnih scenarija ovisi o broju čvorova koji su na istome mjestu u oba scenarija. Koeficijent – udio čvorova na istom mjestu s obzirom na sve čvorove – se množi konstantom 0.5 kako bi smanjili važnost tumačenja dobivenih metodom alternativnog scenarija s obzirom na druge dvije metode. Cijeli izraz zapisan je formulom (4.1).

$$koeficijentsnage = 0.5 \cdot \frac{brojčvorovanaistommjestuuobascenarija}{ukupanbrojčvorovauscenariju} \tag{4.2}$$

Najjednostavniji primjer primjene ove metode bi bio uzeti dva scenarija:

1. *Give me all actors from US*, *Show only older than 20*, *Remove those with 30 mil. or less in revenue.*,
2. *Give me all actors from US*, *Remove those with 30 mil. or less in revenue.*, *Show only older than 20.*

Scenarij 2 će u svom trećem upitu metodom alternativnih scenarija pronaći scenarij jedan s koeficijentom snage $koeficijentsnage = 0.5 \cdot \frac{1}{3} = 0.1667$.

4.6. Provjera logike tumačenja

Nakon odabira vjerojatnog tumačenja upita potrebno je provjeriti kako se logika iza tog tumačenja uklapa u postojeći kontekst. Postoje dva slučaja kada je potrebno verificirati logiku.

Kada je namjera korisnika unija (engl. *union*) – dodavanje novog skupa podataka – potrebno je provjeriti želi li korisnik postojeće redukcije primijeniti i na te podatke. Ako korisnik postavi upite poput: *Give me actors from Croatia* i *Show me only older than* 30, te nakon toga *Add those from Germany*, postavlja se pitanje treba li model vratiti sve glumce iz Njemačke ili treba vratiti sve glumce iz Njemačke starije od 30 godina. Ovisno o tome postoji li taj novi kontekst već u stablu, model će sam zaključiti što treba napraviti ili će pitati korisnika želi li primijeniti prijašnje redukcije na uniju podataka.

Kada je namjera korisnika redukcija (engl. *reduction*) s numeričkim vrijednostima, potrebno je provjeriti postoje li već redukcije istog tipa u trenutnom scenariju i u kakvom su one odnosu. Redukcije mogu biti neovisne jedna o drugoj (engl. *or*), mogu se nadopunjavati (engl. *and*), te novija može poništiti stariju (engl. *replace*). Primjerice, ako u scenariju imamo upite *Show actors with more*

than 3 *oscar nominations* i *Leave actors with less than* 2 *oscar nominations*, moguće je prikazati glumce s manje od dvije nominacije za Oskara, glumce koji imaju više od tri i manje od dvije nominacije (što je nemoguće), te glumce koji imaju više od tri ili manje od dvije nominacije. Model za ispravnu traži postojeći kontekst u stablu te ako ne nađe onda za opciju pita korisnika.

4.7. Određivanje entiteta

Ako je korisnik poslao upit kojem pri standardizaciji nije bilo moguće odrediti tip entiteta, zadatak je modela pokušati predvidjeti o čemu se radi. Prolaskom kroz cijelo stablo model bilježi sve tipove entiteta čiji su reducirani upiti jednaki redukciji poslanog upita te koliko često se ti entiteti pojavljuju. Prema formuli 4.3 moguće je odrediti najvjerojatnije tipove entiteta za dani upit te pitati korisnika koji od njih je ispravan. Također, korisnik može odbiti ponuđene entitete i definirati svoj.

$$vjerojatnosttipa = \frac{brojpojavljivanjatipaustablu}{ukupanbrojčvorovaustablu} \tag{4.3}$$

4.8. Ograničenja modela i prijedlozi za unaprijeđenja

Ovaj model ima određena ograničenja te mogućnosti unaprjeđenja. Osnovni model u ovom radu ima svrhu dokazivanja koncepta i nije išao u toliko detaljnu specifikaciju.

4.8.1. Pokrivenost scenacija

Ispitivanjem eksperta će se pokriti samo scenariji koje on smatra dovoljno važnima. Ako model nema dovoljnu pokrivenost mogućih scenarija, često će doći do potrebe za ispitivanjem korisnika o upitu s obzirom na postojeći scenarij. Moguće rješenje je sustavno generirati scenarije za koje će ekspert odrediti kako se uklapaju u cjelinu.

4.8.2. Točnost interpretacije

Također, problem nastaje ako ekspert ili korisnik odredi ponašanje modela za određeni scenarij, a to ponašanje nije očekivano. Primjerice, ekspert može odrediti da se upiti *Only older than 30* i *Only younger than 25* nadopunjuju. U takvim situacijama model nema način ispraviti se. Moguće rješenje bi bilo tražiti potvrdu točnosti interpretacije scenarija više puta kako bi osigurali ispravnost odabira. Korisnik bi također trebao moći za svaku interpretaciju naznačiti ako nije dobra, iako model s velikom sigurnošću odredi da je ispravna.

4.8.3. Postavljanje filtera u uniji

Kada korisnik dodaje novi skup podataka (radi uniju), model određuje sve filtre koji bi na njega mogli utjecati. Korisnik ima mogućnost prihvatiti sve filtre ili ih sve odbiti. Nedostatak tog pristupa je ograničenje slobode izbora. Ako želi samo pojedine filtre od svih navedenih, korisnik mora odbiti sve i ponovno upisati filtre koje želi.

4.8.4. Opcije pri ručnom upisu tipa entiteta

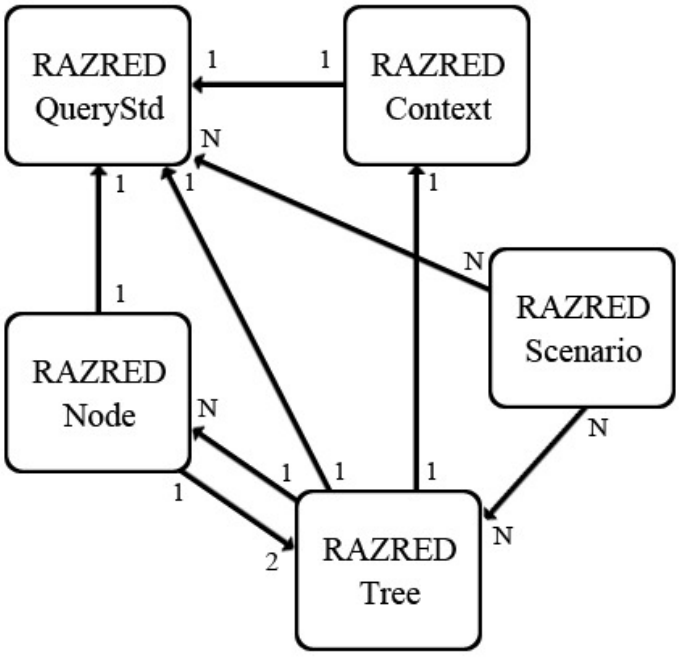
Ako korisnik napiše nejasan upit iz kojeg nije moguće odrediti tip entiteta, te ako model ne ponudi ispravni entitet kao opciju korisniku, on može ručno upisati entitet. U tom slučaju nema provjere postoji li taj entitet u bazi. Korisnik bi trebao vidjeti sve opcije za tip entiteta te bi trebao izabrati jednu od ponuđenih.

5. Implementacija

Osim teorijskog razmatranja modela za određivanje konteksta pitanja, rezultat ovog završnog rada je implementacija u tehnologiji Python. Python je programski jezik namijenjen za brzo prototipiranje hipoteza s odličnom podrškom za strojno učenje te odličnom dokumentacijom (Pedregosa et al., 2011). Vizualizacija stabla je izvedena pomoću python alata *ETE*.[1]

Model se sastoji od (1) dijela za izgradnju stabla i njegovo spremanje, (2) dijela za određivanje konteksta, (3) dijela za nadopunjavanje konteksta, (4) dijela za testiranje ispravnosti rada te (5) dijela za korištenje s preskočenim korakom standardizacije i s uključenim korakom standardizacije.

Na slici 5.1 prikazane su relacije između razreda koji čine cijeli model.



Slika 5.1: Prikaz isprogramirane strukture ciljanog modela.

[1] Alat *ETE* dostupan je na `http://etetoolkit.org`

5.1. Razred *QueryStd*

U razred *QueryStd* sprema se originalni upit korisnika kao i njegov standardizirani ili reducirani oblik. Sve operacije na objektima *QueryStd* rade se kroz metode klase. Slijedi opis osnovnih metoda:

def *___init___(self, source, **kwargs)* – konstruktor koji obavlja sljedeće zadatke:

- učitavanje originalnog upita (engl. *source*) i
- popunjavanje entiteta i namjera ako su proslijeđene, inače postavlja na *None*.

def *update(self , **kwargs)* – metoda za osvježavanje sadržaja objekta koja također provjerava ispravnost podataka s kojima se objekt želi osvježiti.

def *dumps(self)* – metoda za formatiranje cijelog objekta u niz znakova pogodan za spremanje.

def *eq(self , obj, exclude)* – metoda za usporedbu dva objekta *QueryStd*. Pri tome sve parametre poput originalnog upita proslijeđene u listi *exclude* zanemaruje pri usporedbi.

def *part_eq(self , obj, exclude)* – slično metodi *eq*, *part_eq* uspoređuje dva objekta *QueryStd* tako da *obj* mora biti jednak objektu *self* ili mora biti njegov podskup. Također, *part_eq* vraća i koeficijent snage prema formuli (4.1).

def *tree_display(self)* – vraća znakovnu reprezentaciju upita namijenjenu za prikaz u stablu.

def *loads(data)* – učitava objekt *QueryStd* iz znakovne reprezentacije dobivene *dumps* naredbom.

def *clone(self)* – sve vrijednosti u objektu *QueryStd* kopira u novi objekt *QueryStd*.

def *normalize(self)* – vraća reducirani klon objekta *QueryStd*.

5.2. Razred *Context*

Razred *Context* je centralni dio logike modela. U njemu se pronalazi najvjerojatnija interpretacija upita i odvija se sva komunikacija s korisnikom. Također, tu se validira logika i osvježava kontekst. Najbitnije metode ovog razreda su:

def *process(self , possibilities , scenario)* – glavna funkcija razreda zadužena za prolazak kroz cijeli proces obrade svih primljenih mogućnosti standardiziranog upita. Procesi koji se tu izvršavaju su:

– dohvaćanje mogućih tumačenja upita i prosljeđivanje mogućnosti za određivanje najvjerojatnijeg tumačenja,

– postavljanje zastavica koje će kasnije koristiti razred *Scenario* kako bi ispravno spremio kontekst u stablo,

– komunikacija s korisnikom kako bi se odredilo jeli najvjerojatnije tumačenje konteksta ispravno. Ako je trenutni scenarij od prije poznat, ovaj korak se preskače,

– slanje konačnog tumačenja upita metodi *update_context* kako bi se kontekst osvježio.

def *find_possible(self , possibilities , scenario)* – vraća listu svih mogućih tumačenja upita te koeficijent snage za svako tumačenje. Tumačenja se dohvaćaju metodama potpunog podudaranja, djelomičnog podudaranja i alternativnog scenarija. Također, na izlaz svake metode se postavlja odgovarajuća zastavica koja naznačuje metodu.

def *verify_logic(self , option, tree , queries , clear)* – određuje kako se model treba ponašati ako je interpretacija tipa namjere jednaka uniji tako što uzima pristup iz budućeg scenarija ako postoji ili ispitivanjem korisnika što treba napraviti. Također određuje kako se model treba ponašati u slučaju da se radi o redukciji čiji entitet ima numeričke vrijednosti na isti način.

def *get_probable(self , possible)* – vraća najvjerojatnije tumačenje upita uzimajući tumačenje s najvećim koeficijentom snage.

def *update_context(self, option, tree , clear)* – ako model nije uspio odrediti o kojem tipu entiteta se radi, daje prijedloge korisniku – koje dobije pozivajući metodu *predict_context_value* na *Tree* objektu – i s obzirom na korisnikov odgovor osvježava tip entiteta u upitu.

5.3. Razred *Scenario*

Object razreda *Scenario* predstavlja interakciju korisnika s modelom. Kako bi se koristio model za određivanje konteksta pitanja kreira se objekt razreda *Scenario* i pozove metoda *process* detaljnije opisana u nastavku:

def *process(self , possibilities)* – za primljene moguće interpretacije entiteta i namjera stvara novi kontekst te pomoću metode *process* u razredu *Context* proširuje stablo i sprema ga. Način na koji će proširiti stablo te hoće li napraviti novi scenarij ovisi o zastavicama koje postavi *process* metoda u razredu *Context*.

5.4. Razred *Tree*

Razred *Tree* ima ulogu strukturiranja podataka i prolazak kroz njih. Svaki objekt tog razreda ima definiran svoj kontekst, originalni objekt *QueryStd* i listu djece (engl. *children*). Djeca su instance razreda *Node*. Najvažnije metode su:

def *add(self , query, new_context)* – dodaje objekt razreda *Node* korijenu trenutnog stabla kao dijete.

def *shallow_filter (self , query, partial)* – za dani objekt klase *QueryStd* pronalazi sve odgovarajuće prijelaze – s jednakim objektom *QueryStd* – i vraća listu objekata *Tree* te njihove koeficijente snage koji mogu biti jednaki 1 ili izračunati prema formuli (4.1).

def *deep_filter (self , queries)* – rekurzivnim prolaskom kroz stablo nalazi alternativne scenarije s jednakim kontekstima ali drugim poretkom.

def *construct(self)* – pomoću alata *ETE* prikazuje trenutno stablo tako što na svakom čvoru (kontekstu) ispisuje tip entiteta spremljen u objektu *QueryStd* tog čvora. Ako je taj čvor list koji je nastao kao rezultat prekida konteksta, ispisat će se vrijednost *NONE*.

5.4.1. Razred *Node*

Node objekt je poveznica između dva objekta *Tree* za zadani objekt *QueryStd*. To znači da kako bi iz čvora roditelja prešao u čvor djeteta, model mora dobiti upit korisnika čiji reducirani oblik odgovara reduciranom obliku objekta *QueryStd* spremljenog u objektu *Node*.

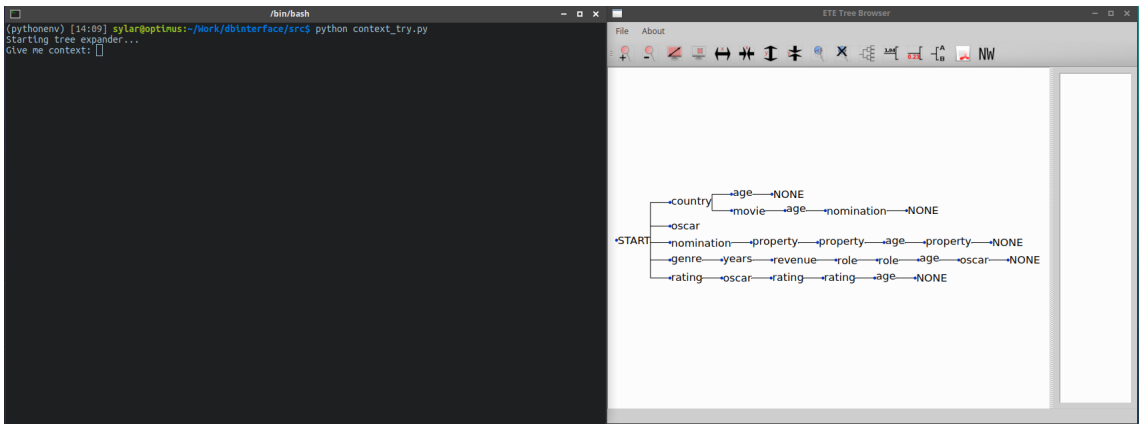
5.4.2. Spremanje stabla

Kako bi model mogao sačuvati naučeno stablo, koristi se *pickle*, *Pythonova* biblioteka za serijalizaciju objekata. *Pickle* pretvara *Pythonov* objekt u niz bajtova (engl. *pickling*) na način da ga se može pohraniti i poslije ponovno učitati kao isti *Pythonov* objekt (engl. *unpickling*).[2]

[2] `https://docs.python.org/3/library/pickle.html`

6. Primjeri uporabe

Funkcionalnosti modela za određivanje konteksta pitanja mogu se podijeliti u četiri kategorije. To su: (1) izgradnja stabla, (2) prepoznavanje postojećeg scenarija, (3) određivanje tipa namjere i (4) testiranje kompletnog modela. Prve tri kategorije funkcionalnosti testiraju se s direktno zadanim standardiziranim upitima kako bi se uklonila odstupanja od ispravnog rješenja prouzročena netočnom i/ili nepotpunom standardizacijom upita. U tim primjerima ulazni podaci će biti u obliku rječnika koji reprezentira objekt *QueryStd* u znakovnoj reprezentaciji. Zadnja kategorija funkcionalnosti demonstrira kako se model uklapa u cjelokupni model. Model se pokreće u programu *Terminal* u *Linux* okruženju prikazano na slici 6.1.



Slika 6.1: Okruženje za pokretanje modela.

6.1. Izgradnja stabla

Na slici 6.2 je prikazano prazno stablo. Trenutno nema niti jedan spremljeni scenarij.

Prvi upit će dohvatiti sve glumce iz Italije. S obzirom na to da je trenutni scenarij prazan, model ne pita korisnika nastavlja li se njegov upit na trenutni

Slika 6.2: Prazno stablo

scenarij. Umjesto toga automatski doda novi čvor u stablo s oznakom *country* jer je to tip entiteta koji se traži. Novo stanje stabla prikazano je na slici 6.3.

```
Upit:
{
  'src': 'Give me all italian actors',
  'MATCH': ('country', {'value': 'Italy'}),
  'WHERE': 'POSITIVE',
  'RELATIONSHIP': ':LIVES_IN',
  'OPERATOR': 'UNION',
}
```

Slika 6.3: Stablo s jednim čvorom.

Trenutni scenarij u sebi ima jedan kontekst. Zbog toga će model za sljedeći upit pitati jeli on dio konteksta. Potvrdnim odgovorom novi se kontekst dodaje u scenarij i proširuje se stablo (slika 6.4a), te se traženi podaci odnose na talijanske glumce stare 60 godina ili manje. Na slici 6.4b prikazan je format u kojem model pita jeli upit dio konteksta.

```
Upit:
{
  'src': 'Filter out those older than 60',
  'MATCH': ('age', {'value': 60, 'numeric': True, 'prep': 'over'}),
  'WHERE': 'NEGATIVE',
  'OPERATOR': 'REDUCE'
}
```

(a) Prikaz stabla nakon proširenja scenarija.

U sljedećem primjeru kada model pita je li upit dio scenarija korisnik odabire opciju 0 i stvara novi scenarij, kao što je prikazano na slici 6.5.

```
Is any of these part of context?
[1] {   'AGGREGATION': None,
    'DISTINCT': None,
    'LIMIT': None,
    'MATCH': ('age', {'numeric': True, 'prep': 'over', 'value': 60}),
    'MATCHED': None,
    'MULTIPLE_FILTERS': [],
    'OPERATOR': 'REDUCE',
    'ORDER_BY': None,
    'REDUCE_FILTER': None,
    'RELATIONSHIP': None,
    'TOP': None,
    'WHERE': 'NEGATIVE',
    'src': 'Filter out those older than 60'}
Press number for option or 0: 1
```

(b) Prikaz upita modela za proširenje scenarija.

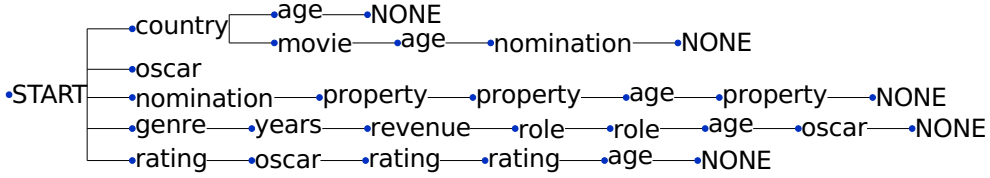
Slika 6.4: Proširivanje postojećeg scenarija.

```
Upit:
{
  'src': 'All movies that won Oscar',
  'MATCH': ('oscar', {'value': 0, 'numeric': True, 'prep': 'over'}),
  'WHERE': 'POSITIVE',
  'OPERATION': 'UNION'
}
```

Slika 6.5: Korisnik stvara novi scenarij.

6.2. Prepoznavanje postojećeg scenarija

Stablo je prošireno kako bi sljedeće primjere bilo lakše demonstrirati. Novo stanje stabla je prikazano na slici 6.6. Postoje tri metode prepoznavanja: (1) metoda potpunog podudaranja, (2) metoda djelomičnog podudaranja i (3) metoda alternativnog scenarija. Slijede primjeri za sve tri metode.



Slika 6.6: Stablo korišteno za prepoznavanje postojećeg scenarija.

6.2.1. Metoda potpunog podudaranja

Započet je novi scenarij. Prvi upit dohvaća sve glumce iz Španjolske. Kako u stablu već postoji scenarij čiji prvi upit ima tip entiteta *country*, model ne pravi novu granu već odlazi u postojeću na čvor *country*.

```
Upit:
{
  'src': 'Show me all actors from Spain',
  'MATCH': ('country', {'value': 'Spain'}),
  'WHERE': 'POSITIVE',
  'RELATIONSHIP': ':LIVES_IN'
}
```

6.2.2. Metoda djelomičnog podudaranja

Sljedeći upit je unutar istog konteksta. Korisnik želi dohvatiti sve glumce starije od 30 godina, ali zbog nedovoljno definiranog upita pri standardizaciji nisu uspješno određeni svi parametri. Model prima standardizirani upit bez parametara *OPERATOR* i *WHERE*, ali svejedno zaključuje da je to dio istog konteksta jer trenutni – već poznati scenarij – ima čvor dijete s istim tipom entiteta.

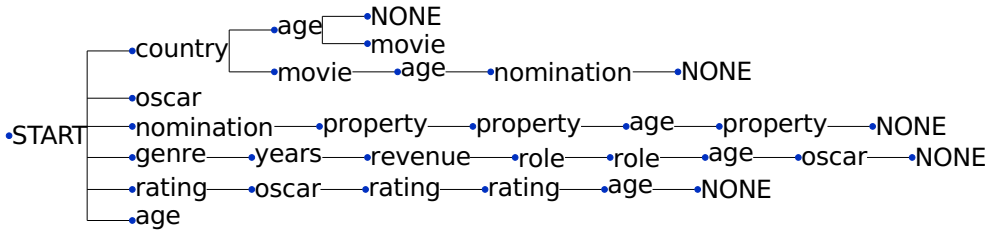
```
Upit:
{
  'src': 'Filter out those older than 60',
  'MATCH': ('age', {'value': 60, 'numeric': True, 'prep': 'over'}),
}
```

6.2.3. Metoda alternativnog scenarija

Zadnji upit za testiranje prepoznavanja postojećeg konteksta ostavlja glumce koji su glumili u filmu "Kum". U trenutnom scenariju nije moguće pronaći prijelaz s tipom entiteta *movie*, ali postoji alternativni scenariji čiji upiti imaju tipove entiteta redom *country*, *movie* i *age*, što odgovara trenutnom scenariju, samo u drugom redoslijedu. Trenutni scenarij ima upite s tipom entiteta redom: *country*, *age* i *movie*. Zbog toga model neće pitati jeli upit dio scenarija, ali će proširiti stablo kao na slici 6.7.

```
Upit:
```

```
{
  'src': 'Leave actors from movie The Godfather',
  'MATCH': ('movie', {'value': 'The Godfather'}),
  'RELATIONSHIP': ':STARRED_IN',
  'WHERE': 'POSITIVE',
  'OPERATOR': 'REDUCTION'
}
```



Slika 6.7: Prikaz stabla nakon pronalaska alternativnog scenarija.

6.3. Određivanje tipa entiteta

Određivanje tipa entiteta koristi se kada korisnik ne definira dovoljno tip entiteta pa kod standardizacije ta vrijednost izostane. Model tada prolaskom kroz cijelo stablo pokuša zaključiti o čemu se radi i na kraju daje listu prijedloga korisniku da izabere ispravni.

U sljedećem primjeru korisnik je htio filtrirati po tipu *age*, ali nije dovoljno jasno formulirao upit. Model je pronašao sve opcije i ispisao ih korisniku. Primjer komunikacije je vidljiv na slici 6.8.

```
Upit:
{
  'src': 'Filter out those over 70',
  'WHERE': 'NEGATIVE',
  'OPERATOR': 'REDUCE'
}
```

6.4. Testiranje kompletnog modela

Konačan rezultat cijelog projekta – u suradnji s kolegama – je interaktivno jezično sučelje koje prima upit, standardizira ga i predviđa kontekst pitanja. U ovom

```
Give me context: {'src': 'Filter out those over 70', 'WHERE': 'NEGATIVE', 'OPERATOR': 'REDUCE'}
Is any of these the entity you are searching for?
[1] age
Press number for option or 0: 1
```

Slika 6.8: Predviđanje mogućeg tipa entiteta.

primjeru korisnik traži glumce po zemlji stanovanja i po spolu, kao što se može vidjeti na slikama 6.9a i 6.8a. Upit je naveden nakon teksta *Input your query* nakon čega se učitaju potrebni podaci kako bi se upit mogao rastaviti na dijelove. Drugi upit (slika 6.9a) demonstrira odgovor modela na nepoznati scenarij gdje s vjerojatnošću od 100% (vidljivo iz varijable *probability*) prepoznaje da se radi o spolu. Također, model je prepoznao da se radi o osobama, da je u pitanju redukcija trenutnog seta glumaca i da filtrirani glumci ostaju – ostali glumci se odbacuju. Model nije dobro prepoznao vrijednost po kojoj se filtrira (*Show women*) te je postavio krivu vezu (*:LIVES_IN*).

```
Input your query: Actors from Croatia
2017-06-30 00:03:12,318 : INFO : loading projection weights from GoogleNews-vectors-negative300.bin
2017-06-30 00:03:12,361 : INFO : loaded (5000, 300) matrix from GoogleNews-vectors-negative300.bin
Predicted context:
{   'AGGREGATION': None,
    'DISTINCT': True,
    'LIMIT': None,
    'MATCH': (   'nominations',
                 {   'numeric': False,
                     'operator': None,
                     'probability': 1.0,
                     'units': 'nominations ',
                     'value': 'Croatia'}),
    'MATCHED': 'PERSON',
    'MULTIPLE_FILTERS': [],
    'OPERATOR': None,
    'ORDER_BY': None,
    'REDUCE_FILTER': None,
    'RELATIONSHIP': ':LIVES_IN',
    'TOP': None,
    'WHERE': None,
    'src': 'Actors from Croatia'}
```

(a) Pretraživanje po zemlji stanovanja.

Slika 6.8: Demonstracija rada modela.

```
Input your query: Show only women
2017-06-30 00:09:17,421 : INFO : loading projection weights from GoogleNews-vectors-negative300.bin
2017-06-30 00:09:17,465 : INFO : loaded (5000, 300) matrix from GoogleNews-vectors-negative300.bin
Predicted context:
Is any of these part of context?
[1] {   'AGGREGATION': None,
    'DISTINCT': None,
    'LIMIT': None,
    'MATCH': (   'gender',
                 {   'numeric': False,
                     'operator': None,
                     'probability': 1.0,
                     'units': 'gender ',
                     'value': 'Show women'}),
    'MATCHED': 'PERSON',
    'MULTIPLE_FILTERS': [],
    'OPERATOR': 'REDUCE',
    'ORDER_BY': None,
    'REDUCE_FILTER': None,
    'RELATIONSHIP': ':LIVES_IN',
    'TOP': None,
    'WHERE': 'POSITIVE',
    'src': 'Show only women'}
Press number for option or 0:
```

(a) Pretraživanje po spolu.

7. Zaključak

S pristupom velikim količinama podataka, sve je veća potreba za jednostavnim sučeljima bazama podataka koja neće zahtijevati znanja u području baza podataka i o strukturama pojedine baze. Postojeća sučelja su prekompleksna ili nedovoljno učinkovita. Virtualni asistenti poput *Siri* i *Google Assistanta* pokušavaju pokriti sva područja interesa čovjeka i samim time nisu dovoljno dobri za ozbiljniju uporabu. Također, komunikacija s virtualnim asistentima i drugim sličnim modelima ne izgleda poput pravog razgovora jer oni ne prate kontekst razgovora.

Cilj ovog rada bio je napraviti model koji će pratiti kontekst razgovora korisnika sa sučeljem i koji će moći nadopunjavati upite korisnika kako bi se dobio format pogodan za pretraživanje baze. Napravljen je model koji uz početno učenje od eksperta može prepoznati postojeće scenarije i stvoriti nove kao i nadopuniti nejasan upit te odabrati njegovo najbolje tumačenje. Također, model prepoznaje situacije u kojima treba tražiti pojašnjenje od strane korisnika.

Izrađeni model bi se mogao poboljšati kroz ljepše sučelje za interakciju, stvaranje bolje pokrivenosti scenarija, unaprjeđenje modela za interpretaciju, više opcija pri postavljanju filtra prilikom unije podataka i boljom interakcijom pri ručnom upisu tipa entiteta.

Literatura

Patrick Brézillon. From expert systems to context-based intelligent assistant systems: a testimony. *The Knowledge Engineering Review*, 26(01):19–24, 2011.

Hee Byun i Keith Cheverst. Utilizing context history to provide dynamic adaptations. *Applied Artificial Intelligence*, 18(6):533–548, 2004.

Anind K Dey. Understanding and using context. *Personal and ubiquitous computing*, 5(1):4–7, 2001.

Goetz Graefe, Harumi Kuno, i Hideaki Kimura. Tree data structure, Travanj 18 2017. US Patent 9,626,398.

Jong-yi Hong, Eui-ho Suh, i Sung-Jin Kim. Context-aware systems: A literature review and classification. *Expert Systems with Applications*, 36(4):8509–8522, 2009.

Andrew McAfee, Erik Brynjolfsson, et al. Big data: the management revolution. *Harvard business review*, 90(10):60–68, 2012.

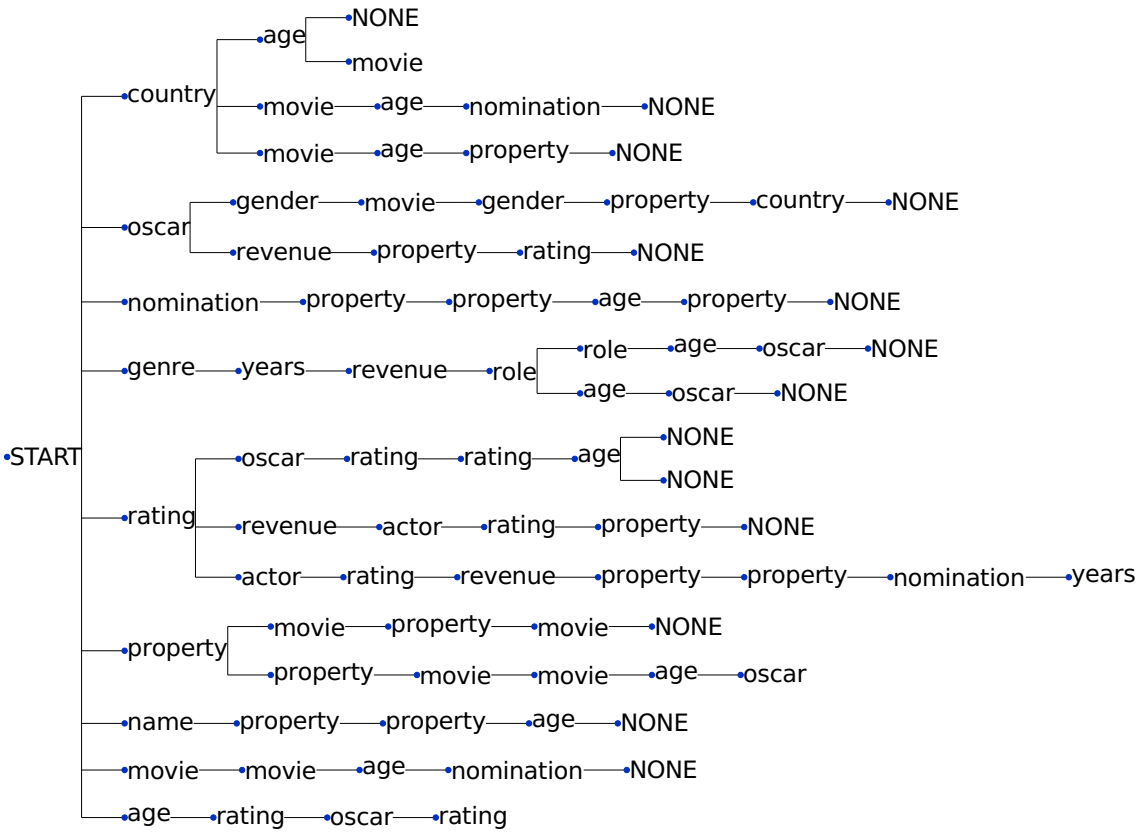
John McCarthy. Notes on formalizing context. 1993.

Justin J Miller. Graph database applications and concepts with neo4j. U *Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA*, svezak 2324, stranica 36, 2013.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

David D Woods. Cognitive technologies: The design of joint human-machine cognitive systems. *AI magazine*, 6(4):86, 1985.

Dodatak A
Stablo za veći broj scenarija



Slika A.1: Prikaz stabla za veliki broj scenarija.

Predviđanje konteksta pitanja za interaktivno jezično sučelje bazi podataka

Sažetak

Stvara se sve veća potreba za interaktivnim tekstnim sučeljima bazama podataka s praćenjem konteksta. Postojeća sučelja su prekompleksna i/ili nedovoljno učinkovita. Praćenje konteksta pomaže u rješavanju tog problema. Cilj ovog rada bio je osmisliti model za određivanje konteksta koji bi nadopunjavao nepotpune upite kao i razrješavao moguće nejasnoće pri stvaranju upita na bazu. Implementirana je struktura stabla za pohranu konteksta kao i model za pronalazak najvjerojatnijeg tumačenja danog upita za postojeći kontekst. Jezik implementacije je *Python*.

Ključne riječi: predviđanje konteksta, interaktivno jezično sučelje, baza podataka, prepoznavanje scenarija.

Question Context Prediction for an Interactive Natural Language Database Interface

Abstract

There is a growing need for an interactive textual language interface to databases while keeping track of context. Existing interfaces are either too complex or inefficient. Keeping track of context helps in solving this problem. The goal of this thesis was to construct a model for determining context which would complement incomplete queries and resolve possible obscurities whilst constructing database queries. A tree data structure was implemented for context storage as well as a system for finding the most probable query interpretation for a given context. The system was implemented in Python.

Keywords: context prediction, interactive language interface, database, scenario recognition.