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. 5321

Predviđanje vrste pitanja za jezično sučelje bazi podataka

Fran Andrija Arbanas

Zagreb, srpanj 2017.

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

Zagreb, 3. ožujka 2017.

ZAVRŠNI ZADATAK br. 5321

Pristupnik: **Fran Andrija Arbanas (0036487458)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Predviđanje vrste pitanja za jezično sučelje bazi podataka**

Opis zadatka:

Jezično sučelje bazama podataka omogućava postavljanje upita nad bazom u kontroliranom prirodnom jeziku. Istraživanja u ovom području povezana su s istraživanjima u području automatskog odgovaranja na pitanja. Ključni korak kod automatskog odgovaranja na pitanja jest utvrđivanje vrste pitanja odnosno očekivanog odgovora, o kojemu ovisi daljnji tijek obrade upita.
Tema završnoga rada jest automatska klasifikacija korisničkog pitanja primjenom strojnog učenja u kontekstu sustava za pristup bazi podataka o poznatim osobama. Potrebno je proučiti postupke za klasifikaciju pitanja u okviru literature koja se bavi sustavima odgovaranja na pitanjima, s naglaskom na pristupe temeljene na strojnom učenju, te modele nadziranoga strojnog učenja za jednoklasnu i višeklasnu klasifikaciju. Razviti model za klasifikaciju pitanja na engleskome jeziku koja uključuje odluku o tome je li pitanje odgovorivo te, ako jest, koja je očekivana vrsta odgovora. Izgraditi prikladnu zbirku za učenje i ispitivanje modela koja će sadržavati primjere pitanja s ispravnim oznakama. Provesti vrednovanje modela, usporedbu 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 obitelji na konstantnoj podršci tijekom studiranja.
Zahvaljujem mentoru na pomoći i na savjetima.
Zahvaljujem kolegama Ivanu i Juraju na odličnoj suradnji pri izradi ovog rada.
Zahvaljujem kolegama iz Vingd d.o.o na pomoći pri odabira teme.

Sadržaj

1. Uvod

Svakoga dana u svijetu se stvaraju ogromne količine podataka. Prema podacima iz 2015. godine dnevno se stvara čak 2.5 kvintilijuna bajtova.[1] Dio tih podataka se sprema u baze podataka radi strukturiranosti i lakšeg pretraživanja. Kako bi se omogućilo pretraživanje baza podataka korisnicima bez znanja o istima, rodila se ideja o jezičnom sučelju prema bazama podataka. Ideja je da bilo tko može u prirodnom jeziku pisati upite te dobiti željene rezultate, što ima više potencijalnih prednosti:

1. Korisnici ne moraju učiti specifični jezik da bi mogli napisati upit,
2. Količina upita koje mogu napisati je slična količini koju mogu napisati u formalnom jeziku za upite,
3. Moguć je dialog i popravljanje upita.

Jezično sučelje prema bazi podataka već je nastalo 60-ih godina prošlog stoljeća [9]. Znanstvenici su smatrali da je to dovoljno ograničen projekt da bi ga se moglo provesti u djelo te da će, ako uspiju, jezično sučelje bazi podataka biti iznimno korisno. Iako je tijekom proteklih desetljeća nastala nekolicina implementacija jezičnih sučelja, ona su imala previše ograničenja na pisanje upita u prirodnom jeziku da bi to korisnici mogli koristiti.

Problemi s kojima se susrećemo odmah na početku su prepoznavanje neispravnih upita i predviđanje vrste pitanja. U ovom radu obradit će se upravo ti problemi.

U stručnoj literaturi nalazi se mnogo radova koji rješavaju problem prepoznavanja neispravnih upita. Dva najzastupljenija pristupa rješavanju tog problema su detekcija netipičnih vrijednosti (engl. *outlier detection*) i otkrivanje novih vrijednosti (engl. *novelty detection*). Za potrebe ovog rada odabran je pristup otkrivanja novih vrijednosti. Ti pristupi su detaljnije opisani u poglavlju 4.

[1] https://www-01.ibm.com/software/data/bigdata/what-is-big-data.html

Problem predviđanja vrste pitanja svodi se na višeklasni klasifikacijski problem strojnog učenja (engl. *machine learning*). U sklopu ovog rada predviđanje vrste pitanja će se odnositi na klasifikaciju upita u jednu od 14 klasa (npr., *where*, *group by*, *order by*, ...). Detaljnije je opisano u poglavlju 5.

Sustav za koji je ovaj rad namijenjen je jezično sučelje graf bazi neo4j,[2] iako ga je moguće uz manje preinake primijeniti i na druge sustave. Domena na koju je ograničeno programsko rješenje je baza filmova i glumaca. Konkretno, primjer uporabe su upiti na engleskom jeziku nad bazom filmova i glumaca. Za rješavanje problema otkrivanja novih vrijednosti te predviđanja vrste pitanja korištene su tehnike obrade prirodnog jezika (engl. *natural language processing*) te strojnog učenja, a izvedba će se svoditi na izgradnju dva modela strojnog učenja.

Ovaj rad dio je razvoja složenijeg sustava koji se sastoji od tri komponente, gdje je prva komponenta tema ovog rada. Druga je prepoznavanje i klasifikacija entiteta unutar upita, a treća je prepoznavanje konteksta upita, odnosno nadovezuje li se prethodni upit na onaj trenutni.

Programsko rješenje napisano je u programskom jeziku Python, verziji 3.5.2, dok su se kao pomoćne knjižnice koristile `sklearn`,[3] `nltk`[4] i `gensim`.[5]

Za otkrivanje novih vrijednosti korišten je model nenadziranog učenja (engl. *unsupervised learning*) zvan jednoklasni stroj potpornih vektora (engl. *one-class SVM*), dok je za predviđanje vrste pitanja korišteno nekoliko modela iz grupe modela nadziranog učenja (engl. *supervised learning*).

U poglavlju 2 daje se pregled srodnih radova. Poglavlje 3 sadrži opis skupa podataka koji je korišten te način na koji je taj skup podataka pripremljen za rješavanje problema u ovom radu. U poglavlju 4 opisan je problem otkrivanja novih vrijednosti, teorija iza algoritma jednoklasnog stroja potpornih vektora koji će biti korišten u radu te implementacija i evaluacija rješenja. Poglavlje 5 dat će opis problema predviđanja vrste pitanja, pregled, teorijsku podlogu algoritama koji su korišteni, implementaciju rješenja te na kraju evaluaciju. Rad završava zaključkom u poglavlju 6.

[2] https://neo4j.com/
[3] http://scikit-learn.org/stable/
[4] http://www.nltk.org/
[5] https://radimrehurek.com/gensim/

2. Srodni radovi

U ovom poglavlju dan je pregled srodnih radova. Područja radova koja su relevantna za ovaj rad su: detekcija netipičnih vrijednosti i otkrivanje novih vrijednosti za problem prepoznavanja neispravnih upita, te radovi iz područja sustava na odgovaranje pitanja za predviđanje vrste pitanja.

U radu [14] dan je odličan pregled otkrivanja novih vrijednosti i metoda kojima se taj problem može riješiti. Predstavljene su probabilističke metode, metode po udaljenosti, metode rekonstrukcije, domene i informacijske teorije. Od tih metoda, za ovaj rad odabrana je metoda domene koja se svodi na definiranje granice oko tipičnih vrijednosti.

Rad [12] uspoređuje različite implementacije jednoklasnog stroja potpornih vektora međusobno i sa metodom neuronskih mreža. Od tih implementacija najbolje rezultate je dala implementacija koja će biti korištena u ovom radu, ona Bernharda Schölkopfa [16]. U tom radu korištene su značajke vreće riječi (engl. *bag of words*), ali je ustanovljeno da su bolji rezultati bili kod binarne reprezentacije, odnosno kada se označava samo postoji li riječ u upitu, a ne i koliko puta se pojavila. U ovome radu dodana je i značajka vektorske reprezentacije riječi (engl. *word embeddings*) koja u tom radu nije bila. Prema tablicama rezultata u radu vidi se da je jednoklasni stroj potpornih vektora prema Schölkopfu jako osjetljiv na izbor jezgrine funkcije (engl. *kernel*), značajki i duljine značajki.

Problem predviđanja vrste pitanja sličan je klasifikaciji vrste pitanja u sustavima odgovaranja na pitanja (engl. *question answering system*), stoga će se pregled srodnih radova u nastavku usredotočiti na radove iz tih područja.

U radu [18] dana je usporedba nekoliko algoritama koji su isprobani i u ovom radu: naivni Bayes, k najbližih susjeda, stablo odluke te stroj potpornih vektora. Najveća razlika njihovog i ovog rada je u broju klasa: u radu [18] klasificira se u 6 grubih te 50 finih klasa dok se u ovom radu klasificira u 14 klasa. Rezultati koje su dobili ukazuju na nadmoć stroja potpornih vektora, što u ovom radu nije bio slučaj.

3. Skup podataka

Kao što je spomenuto u uvodu, domena na koju je ovaj rad ograničen je baza filmova i glumaca, te su upiti na engleskom jeziku. Kako bi mogli naučiti modele da prepoznaju određene tipove upita, nužno je izraditi upite na kojima će modeli učiti.

Za potrebe ovog rada skupovi podataka ručno su izrađeni. Radi se o dva skupa podataka, gdje jedan skup podataka sadrži 294 tipična upita dok drugi skup sadrži 46 netipična upita. Skup netipičnih upita bitan je radi provjere točnosti modela za otkrivanje novih vrijednosti. Primjeri tipičnih upita:

```
Give me all actors from the USA.
Add actors from Germany.
Actors who won Oscar.
Order by name.
```

Primjeri netipičnih upita:

```
That is a painting created by Van Gogh.
A user can upload two files containing sequences from two libraries.
```

Skup s tipičnim upitima dodatno je označen kako bi se mogao koristiti prilikom treniranja modela za predviđanje tipa pitanja. Oznake koje su korištene dobivene su iz jezika Cypher, koji se u neo4j-u koristi za kreiranje upita. Također, odabrane su samo oznake koje su jednostavne za pretvorbu u SQL upit kako bi se sustav mogao jednostavno testirati. Ukupan broj korištenih oznaka je 14. Oznake koje su korištene prilikom označavanja prikazane su u tablici 3.1.

Napisan je još jedan dio skupa podataka koji sadrži više primjera za klase s manje ponavljanja, ali taj dio skupa podataka nije odgovarao smjernicama za pisanje upita, pa nije korišten u konačnim eksperimentima.

Prilikom učitavanja skupa podataka u program svaki se upit rastavlja na listu riječi (engl. *tokenize*) pomoću funkcije `word_tokenize` iz knjižnice `nltk` te

Tablica 3.1: Oznake za predviđanje tipa pitanja

Oznaka	Broj ponavljanja
WHERE | POSITIVE	119
WHERE | NEGATIVE	50
ORDER_BY | ASC	17
ORDER_BY | DESC	20
UNION	35
GROUP_BY	25
TOP	4
AGGREGATION | MIN	2
AGGREGATION | MAX	2
AGGREGATION | SUM	2
AGGREGATION | AVG	4
AGGERGATION | COUNT	4
LIMIT	7
DISTINCT	3

se potom upitu pridjeljuje njegova oznaka za predviđanje tipa pitanja. Kako bismo dobili bolje predviđanje, za neke upite su napravljene parafraze koje su spremljene posebno jer se prilikom kreacije značajki evaluiraju isto kao i upit na koji su parafraze.

Za potrebe otkrivanja novih vrijednosti prilikom evaluacije modela uzimaju se dva testna skupa podataka, kao što je prikazano u tablici 3.2. Preostali dio tipičnih primjera uzima se za treniranje modela. Netipični primjeri nužni su kako bismo znali kolika je točnost prepoznavanja modela za otkrivanje novih vrijednosti.

Tablica 3.2: Skupovi podataka za testiranje otkrivanja novih vrijednosti

Tip podataka	Broj primjera
Tipični	59
Netipični	46

Prilikom izrade skupa podataka sudjelovalo je troje ljudi, te je to trajalo tjedan dana. Skup podataka napisan je na engleskom jeziku.

4. Otkrivanje novih vrijednosti

4.1. Opis problema

Kod problema klasifikacije podataka iznimno je teško pokriti sve slučajeve koji se mogu dogoditi. Stoga se prije klasifikacije provjerava odgovaraju li podaci našoj domeni. U konkretnom opsegu ovog rada cilj je ustanoviti radi li se o upitu koji ne možemo obraditi. U stručnoj literaturi postoje nesuglasice oko toga što se smatra detekcijom netipičnih vrijednosti (engl. *Outlier detection*), a što otkrivanjem novih vrijednosti (engl. *Novelty detection*) te postoji li uopće razlika između tog dvoje. U ovom radu ćemo se povesti definicijom iz knjige [17], koja se također koristi kod definicije problema na stranici `sklearn`.[1] Ta dva tipa ne razlikuju se mnogo, kod otkrivanja novih vrijednosti u skupu podataka za treniranje nalaze se samo tipične (engl. *inlier*) vrijednosti dok se kod detekcije netipičnih vrijednosti također nalaze i netipične (engl. *outlier*). Unatoč tome, cilj oba tipa svodi se na isto – novi upit treba označiti kao tipičan ili netipičan. U ovome radu bit će korišteno otkrivanje novih vrijednosti.

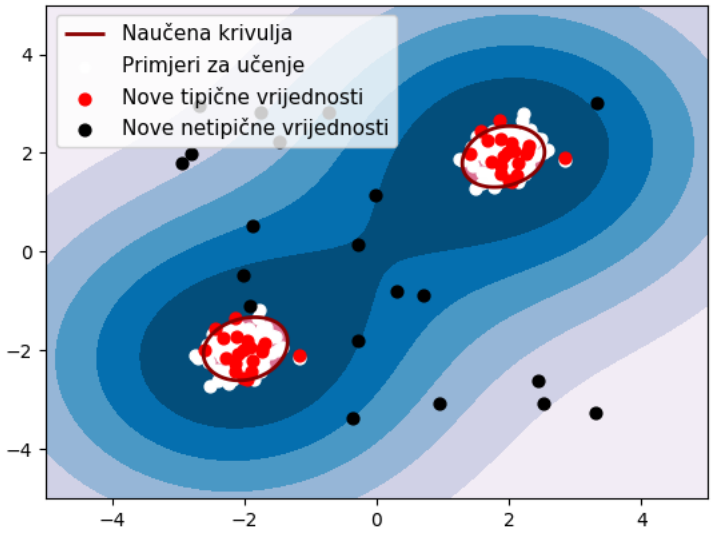
Kao što je već spomenuto, u skupu podataka za treniranje modela otkrivanja novih vrijednosti ne nalaze se netipične vrijednosti. Model za otkrivanje novih vrijednosti stoga treba naučiti krivulju koja obavija tipične vrijednosti te se klasifikacija novih vrijednosti svodi na provjeru nalazi li se ta vrijednost unutar granica krivulje. Najpoznatiji model za otkrivanje novih vrijednosti je jednoklasni model potpornih vektora (engl. *One-class SVM*) koji je predstavio Bernhard Schölkopf [16].

U narednim odjeljcima bit će dana detaljnija razrada rješenja koje je implementirano u ovome radu. U odjeljku 4.2 bit će obrađen model jednoklasnog stroja potpornih vektora te njegova teorijska podloga. U odjeljku 4.3 bit će obrađena implementacija modela u ovom radu, dok će u odjeljku 4.4 biti objašnjena

[1] http://scikit-learn.org/stable/modules/outlier_detection.html

odabrana metoda evaluacije te će biti prezentirani rezultati.

4.2. Jednoklasni stroj potpornih vektora



Slika 4.1: Primjer jednoklasnog stroja potpornih vektora, slika je preuzeta sa: http://scikit-learn.org/stable/auto_examples/svm/plot_oneclass.html

Pregled algoritma jednoklasnog stroja potpornih vektora učinjen je prema radu [16] uz pomoć internetske stranice [3].

Za rješavanje problema otkrivanja novih vrijednosti u ovome je radu odabran jednoklasni stroj potpornih vektora (engl. *One-class SVM*) koji je nenadzirani (engl. *unsupervised*) model strojnog učenja (engl. *machine learning*). Taj model predstavio je Bernhard Schölkopf [16]. U nastavku odjeljka biti će opisano koja je ideja iza tog modela.

Kao što ime modela i kaže, temelji se na stroju potpornih vektora koji je detaljnije opisan u pododjeljku 5.2.1. Cilj jednoklasnog stroja potpornih vektora nije da razdvoji dvije ili više klasa primjera, već se svodi na učenje krivulje koja dovoljno dobro grupira pozitivne odnosno tipične primjere. Za primjere koji se nalaze unutar krivulje model vraća da su tipični (1), dok za primjere izvan krivulje vraća da su netipični (-1). Jednoklasni stroj potpornih vektora kao i stroj potpornih vektora može se poslužiti jezgrenim trikom kako bi preslikao primjere u višu dimenziju. Više o jezgrenom triku u 5.2.1.

Za razliku od stroja potpornih vektora, jednoklasni stroj potpornih vektora nema hiperparametar C, već ima parametar ν koji označava koliko primjera iz skupa podataka se može smatrati netipičnim vrijednostima.

Funkcija odluke (4.2) dobiva se rješavanjem kvadratnog programa (engl. *quadratic program*) (4.1).

$$\min \ \frac{1}{2}||w||^2 + \frac{1}{\nu \cdot l}\sum_i \xi_i - \rho \tag{4.1}$$

$$f(\boldsymbol{x}) = sgn(\sum_i \alpha_i k(\boldsymbol{x_i}, \boldsymbol{x}) - \rho) \tag{4.2}$$

U okviru ovog rada, bit će korištena implementacija jednoklasnog stroja potpornih vektora iz knjižnice `sklearn`.[2] Primjer rada tog modela iz knjižnice `sklearn` nalazi se na slici 4.1

4.3. Implementacija modela

Za model je korištena knjižnica `sklearn`. Kako bi se model mogao trenirati, prvo je potrebno od upita izgraditi značajke na kojima će model učiti. Potom treba obaviti podešavanje hiperparametara pomoću pretraživanja rešetke (engl. *Grid search*) i unakrsne provjere (engl. *Cross validation*) i na kraju treba model evaluirati. U sljedećim pododjeljcima bit će detaljno opisane značajke (4.3.1), podešavanje hiperparametara (4.3.2) te evaluacija modela.

4.3.1. Značajke

Značajke s kojima je model isproban su vreća riječi (engl. *bag of words*) i vektorske reprezentacije riječi (engl. *word embeddings*).

1. **Vreća riječi**

   Radi se o reprezentaciji teksta (odnosno upita u ovom slučaju) multiskupom (engl. *multiset*) njegovih riječi.[3] Uzimamo skup jedinstvenih riječi u upitima koje promatramo te za svaki pojedini upit pobrojimo koje riječi se pojavljuju a za riječi koje se ne pojavljuju stavljamo 0. Primjerice, ako bismo imali sljedeće upite

[2] http://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html
[3] https://en.wikipedia.org/wiki/Bag-of-words_model

```
Give me all actors from the USA.
Add actors from Germany.
```

Tada bi skup riječi bio

```
{"Give", "me", "all", "actors", "from",
 "the", "USA", "Add", "Germany"}
```

Dok bi vektori značajki za te rečenice bili

```
[1, 1, 1, 1, 1, 1, 1, 0, 0]
[0, 0, 0, 1, 1, 0, 0, 1, 1]
```

U ovom radu su iz skupa riječi izbačene zaustavne riječi pomoću modula `stopwords` iz knjižice `nltk` te su sve riječi zapisane isključivo malim slovima kako ne bi dolazilo do loših rezultata radi krivo napisanih velikih slova.

2. **Vektorska reprezentacija riječi**

   Vektorska reprezentacija riječi je grupni naziv za skup modela pretvorbe riječi u vektor značajki visoke dimenzionalnosti (engl. *word embeddings*). Radi se o modelima koji pomoću neuronskih mreža riječi opisuju vektorom proizvoljne duljine. Modele je predstavila grupa znanstvenika iz Googlea, na čelu sa znanstvenikom Tomasom Mikolovom [13]. Postoje dva modela koja se koristite prilikom kreiranja vektora, a to su neprekidna vreća riječi (engl. *Continuous bag of words, CBOW*) i *skip-gram*, koji su opisani na web-stranici [6].

   U ovom radu korištena je već istrenirana zbirka riječi Google News, koja sadrži 3 milijuna riječi s vektorima od 300 značajki.[4]

4.3.2. Podešavanje hiperparametara

Kako bi model jednoklasnog stroja potpornih vektora radio što bolje, treba obaviti podešavanje hiperparametara, obično kroz metode pretraživanja rešetke i unakrsne provjere.

[4] https://code.google.com/archive/p/word2vec/

Unakrsna provjera je način treniranja modela gdje se skup podataka za treniranje podjeli na nekoliko cjelina (preklopa), jedna se uzme za testiranje, a ostale za treniranje. Korišteno je 5 preklopa. Model se potom evaluira, te se procedura ponovi tako da se druga cjelina uzme za testiranje i ostale za treniranje. Postupak se ponavlja dok model nije istreniran na cijelom skupu podataka.

Metoda pretraživanja rešetke svodi se na traženje optimalne kombinacije hiperparametara kroz isprobavanje svake kombinacije te potom evaluiranje modela.

Za vrijednosti funkcije jezgre isprobane su vrijednosti linearne (engl. *linear*) te radijalne bazne funkcije (engl. *radial basis function*). Hiperparametar γ označava koliko određeni primjer utječe na druge primjere, gdje niske vrijednosti γ označavaju da slabo utječe, a visoke vrijednosti da jako utječe. Parametar ν određuje koliki postotak primjera smije biti krivo klasificiran. U tablici 4.1 nalaze se vrijednosti hiperparametara korištene tijekom treniranja modela.

Tablica 4.1: Isprobane vrijednosti hiperparametara

Hiperparametar	Raspon vrijednosti
Jezgra	linear/RBF
γ	$2^{-15} \ldots 2^{10}$
ν	$0.01, 0.02, \ldots 0.1$

U ovom radu unakrsna validacija i podešavanje hiperparametara za jednoklasni stroj potpornih vektora napravljena je ručno te implementirana u funkciji `one_class_hiperparameter_optimization`. Kako bismo dobili bolji uvid u odabir hiperparametara, izgrađen je još jedan skup podataka u kojem se nalaze samo netipične vrijednosti na kojima će se model evaluirati. U funkciji `one_class_hiperparameter_optimization` se nakon svakog odabira hiperparametara model evaluira pozivom funkcije `one_class_score` koja testira točnost modela na skupu podataka za testiranje te na skupu podataka s netipičnim vrijednostima.

4.4. Evaluacija i rezultati

Nakon treniranja optimalnog modela, spremamo model na disk pomoću `joblib.dump` funkcije koja se nalazi u `sklearn.externals` modulu.[5] Spremanje modela na disk

[5] http://scikit-learn.org/stable/modules/model_persistence.html

se izvršava kako ne bismo morali svaki puta trenirati model (treniranje modela zauzima veliku količinu računalnih resursa).

Evaluacija modela izvršava se pozivom `spam_test` funkcije koja se nalazi u datoteci `pipeline.py`. Ta funkcija inicijalizira vektor značajki za testiranje za tipične i netipične vrijednosti, te vektor za treniranje modela. Prilikom evaluacije koristi se ugniježđena unakrsna provjera, sa 10 preklopa u vanjskoj petlji te 5 preklopa u unutarnjoj. Podsjećamo da se skup podataka sa netipičnim primjerima koristi isključivo za evaluaciju, te se na njemu ne trenira model. U vanjskoj petlji skup podataka dijeli se na skup podataka za treniranje koji sadrži 265 primjera i skup podataka za testiranje koji sadrži 29 primjera. Potom se u unutarnjoj petlji za optimizaciju hiperparametara koristi 265 primjera, koji se dijele na 212 primjera za treniranje i 53 primjera za testiranje. Nakon optimizacije hiperparametara model se ocjenjuje na 29 tipičnih i 46 netipičnih primjera. Točnost se računa omjerom ispravno klasificiranih primjera i ukupnim brojem primjera (29 za tipične, odnosno 46 za netipične vrijednosti). Konačna točnost koja se nalazi u tablici 4.3 je prosjek 10 dobivenih točnosti s različitim kombinacijama značajki. Kao osnovni model korišten je model koji sve upite smatra ispravnim.

Tablica 4.2: Rezultati evaluacije

Značajke	Tipične vrijednosti	Netipične vrijednosti	Oboje
Osnovni model	100%	0%	86.5%
Vreća riječi	58.1%	68.0%	64.1%
Vektorska reprezentacija	88.4%	84.3%	85.9%
Oboje	89.4%	79.6%	83.4%

Prema tablici se vidi da se najbolji rezultati dobiju kada se vektor značajki kreira samo s metodom vektorske reprezentacije riječi. U teoriji vektorska reprezentacija riječi zamjenjuje vreću riječi jer daju isti tip značajki.

Za obavljanje provjere daje li odabrani model bolje rezultate od osnovnog modela napravili smo permutacijski test sa hipotezom H_0 da je osnovni model jednak odabranom modelu. Odabrani model je u ovom slučaju jednoklasni stroj potpornih vektora sa značajkom vektorske reprezentacije riječi, za koji su dobiveni najbolji rezultati.

Permutacijski test se radi zato što ne možemo pretpostaviti koja je distribucija vrijednosti podataka, a želimo provjeriti jesu li dva modela jednako dobra. Također permutacijski testovi koriste se kada imamo mali broj podataka na kojima

želimo testirati neku hipotezu, jer oni spadaju u metode ponovnog uzorkovanja koje omogućuju generalizaciju na malom broju podataka.

Metoda permutacijskog testa svodi se na računanje predviđanja oba modela te se potom odaberu slučajna mjesta u tim vektorima predviđanja na kojima će se zamijeniti predviđanje jednog i drugog modela. Zatim se izračuna točnost tih izmijenjenih vektora. Izračuna se apsolutna vrijednost razlike te dvije dobivene točnosti koja se usporedi sa apsolutnom vrijednosti razlike točnosti koje se nalaze u tablici 4.3 (za osnovni model 0.865 te za odabrani 0.859, te je stoga razlika 0.006). Postupak se ponovi velik broj puta, u ovom radu tisuću puta. Zanima nas koliko puta je originalna razlika (dobivena iz tablice) po apsolutnoj vrijednosti veća od razlike dobivene permutacijom. Taj dobiveni broj potom podijelimo s brojem ponavljanja postupka permutacije kako bismo dobili p-vrijednost. Provođenjem tog postupka dobili smo p-vrijednost od 0.781 koja nam govori da ne možemo odbaciti hipotezu H_0 na razini značajnosti od 5%. Zaključujemo da su osnovni model i odabrani model jednako dobri.

Ovakav rezultat testa dobiven je radi nedostatka negativnih primjera, odnosno, zato što je negativnih primjera prilikom računanja testa bilo samo 46, a pozitivnih 265. Radi toga je osnovni model davao bolje rezultate iako je očito da u stvarnosti bi bolje rezultate davao jednoklasni stroj potpornih vektora sa značajkom vektorske reprezentacije riječi.
Primjeri klasificiranih upita:

Tablica 4.3: Primjeri klasifikacije

Upit	Klasifikacija	Ispravna klasifikacija
Give me all actors from europe	1	1
Order them by age	1	1
F-score is ill-defined	-1	-1
Group actors by revenue	1	1
Where is the nearest market	1	-1
Can a bird fly	-1	1
Group countries by language spoken	1	-1

5. Predviđanje vrste pitanja

5.1. Opis problema

Nakon prepoznavanja radi li se o ispravnom upitu, prilikom prevođenja prirodnog jezika u jezik za upite bitna je detekcija vrste pitanja. U slučaju strukturiranog jezika upita (engl. *structured query language, SQL*) vrste pitanja bi bile klase `WHERE`, `ORDER BY`, `GROUP BY` i druge. Svaka od tih klasa određivala bi neki tip pitanja, pa se stoga može zaključiti da se radi o višeklasnom klasifikacijskom problemu.

Klasifikacija je jedan od osnovnih problema strojnog učenja za koji su u literaturi predloženi različiti pristupi. Razvijeni su mnogi algoritmi koji se bave problemom klasifikacije, ali svaki od njih je bolji u određenom specifičnom slučaju. Kako klase u našim podacima nisu jednoliko zastupljene, očekuje se da će naučeni model davati bolje rezultate za zastupljenije klase nego za one koje nisu toliko zastupljene.

Također, dolazimo do pitanja je li problem što se radi o višeklasnoj klasifikaciji? Većina algoritama napisana je tako da se može koristiti u višeklasnim klasifikacijskim problemima, dok oni koji nisu se mogu svesti na višeklasne pomoću jedne od strategija. Strategije koje svode binarne klasifikacijske probleme na višeklasne zovu se jedan protiv svih (engl. *one-vs.-all*) te jedan protiv jednog (engl. *one-vs.-one*). Ukratko će biti objašnjena strategija jedan protiv jednog iz razloga što se koristi u stroju potpornih vektora koji je najzastupljeniji algoritam za rješavanje klasifikacijskih problema.

Strategija jedan protiv jednog zasniva se na ideji da se uzimaju parovi klasa koje treba naučiti te se za svaki par izgradi i nauči binarni klasifikator temeljen na određenom algoritmu. Prilikom predviđanja klase za podatak, svi izgrađeni klasifikatori daju svoje predviđanje klase nad tim podatkom te se kao predviđena klasa uzme ona koja je dobila najviše "glasova".

U ovom radu umjesto strukturiranog jezika upita odlučili smo se za graf bazu

neo4j koja koristi jezik Cypher za kreiranje upita.[1] Klase koje taj jezik podržava i koje smo odlučili klasificirati prikazane su u tablici 3.1.

U odjeljku 5.2 bit će detaljno opisani algoritmi koji su korišteni. U odjeljku 5.3 bit će opisana implementacija rješenja u opsegu ovog rada te će u odjeljku 5.4 biti prikazani rezultati.

5.2. Algoritmi

U radovima koji se bave klasifikacijom pitanja u sustavima s odgovaranjem na pitanja najčešće se kao najbolji algoritam ističe stroj potpornih vektora. U radu su korišteni sljedeći algoritmi za predviđanje vrste pitanja:

1. Stroj potpornih vektora (engl. *Support vector machine, SVM*)
2. Klasifikator pojačanja gradijenta (engl. *Gradient boosting classifier*)
3. Klasifikator slučajne šume (engl. *Random forest classifier*)
4. Stablo odluke (engl. *Decision tree classifier*)
5. Naivni Bayesov klasifikator (engl. *Naive Bayes classifier*)
6. Klasifikator k najbližih susjeda (engl. *k nearest neighbors classifier*)

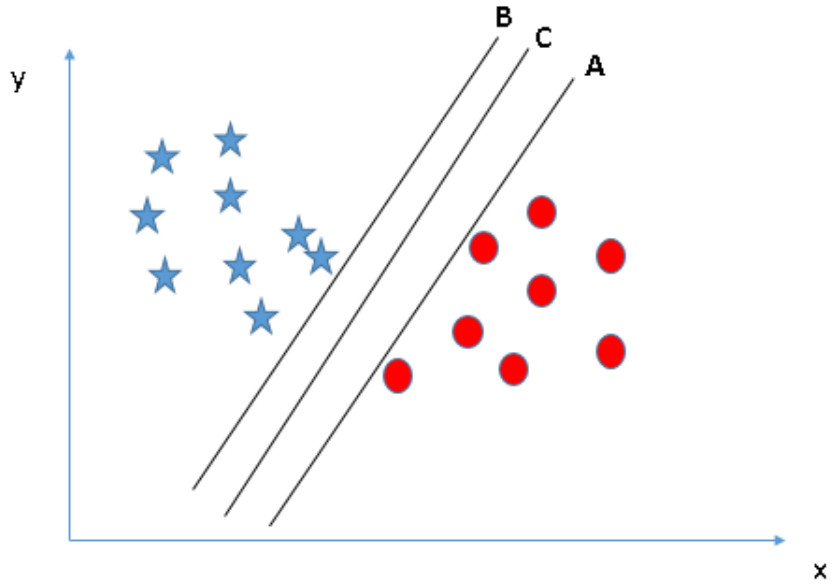
Od tih algoritama detaljnije će biti opisani stroj potpornih vektora, stablo odluke, naivni Bayesov klasifikator te klasifikator slučajne šume koji su u većini slučajeva davali najbolje rezultate.

5.2.1. Stroj potpornih vektora

Stroj potpornih vektora jedan je od najpopularnijih algoritama koji se koriste u klasifikacijskim problemima. Algoritam su 1963. godine osmislili Vladimir Vapnik i Alexey Chervonenkis. Godine 1992. tom algoritmu nadodan je jezgreni trik (engl. *kernel trick*) koji je omogućio da stroj potpornih vektora može rješavati probleme nelinearne klasifikacije [7]. Posljednji dodatak algoritmu bio je 1995. kada je objavljena metoda meke margine (engl. *soft margin*) [10]. U ovom pododjeljku razrada stroja potpornih vektora bit će objašnjena prema [5].

[1] https://neo4j.com/developer/cypher-query-language/

Osnovna ideja iza algoritma stroja potpornih vektora je da algoritam razdvoji klase tako da je između njih što veći prostor. Ta strategija razdvajanja naziva se maksimalna margina. Takvo razdvajanje klasa dat će manju šansu krive klasifikacije (slika 5.1).



Slika 5.1: Prikaz margina različite udaljenosti od primjera

Prvo ćemo obraditi linearan model i linearno odvojive probleme što se naziva metoda tvrde margine (engl. *hard margin*). Potom ćemo objasniti linearan model koji može rješavati nelinearno odvojive probleme (meka margina), a na kraju ćemo objasniti nelinearan model koji se temelji na jezgrenom triku.

Formula modela stroja potpornih vektora, odnosno formula hiperravnine, je:

$$h(\boldsymbol{x}) = \boldsymbol{w}^T x + w_0. \tag{5.1}$$

Problem maksimalne margine svodi se na traženje najveće udaljenosti hiperravnine od najbližeg primjera, što se matematički zapisuje:

$$\text{argmax}_{\boldsymbol{w},w_0}\{\frac{1}{||\boldsymbol{w}||}\min_i\{y^{(i)}(\boldsymbol{w}^T\boldsymbol{x} + w_0)\}\}. \tag{5.2}$$

Problem se pojednostavljuje uzimanjem da je $\min_i\{y^{(i)}(\boldsymbol{w}^T\boldsymbol{x} + w_0)\} = 1$, te se formula malo preuredi kako bi se na kraju dobilo:

$$\text{argmin}_{\boldsymbol{w},w_0}\frac{1}{2}||\boldsymbol{w}||^2, \tag{5.3}$$

uz ograničenje da je $y^{(i)}(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + w_0) \geq 1$. Rješavanje ovog optimizacijskog problema svodi se na rješavanje kvadratnog programa metodom Lagrangeovih multiplikatora, što neće biti opisano u opsegu ovog rada.

U slučaju problema koji nije linearno odvojiv, margini se dopušta da pogriješi na nekoliko primjera, ali se za svaki krivo klasificirani primjer plaća kazna ovisna o udaljenosti tog primjera od margine. Uvodi se varijabla labavosti (engl. *slack variable*) ξ_i koja predstavlja pogrešku klasificiranja primjera te parametar C koji kontrolira koliko se naplaćuje pogreška, odnosno, što je veći C, to će algoritam više izbjegavati pogreške. Dodavanjem varijable labavosti i parametra C stroj potpornih vektora prelazi u klasifikaciju meke margine. Formula koja u tom slučaju definira maksimalnu marginu:

$$\text{argmin}_{\boldsymbol{w},w_0} \frac{1}{2}||\boldsymbol{w}||^2 + C\sum \xi_i, \tag{5.4}$$

uz ograničenja $y^{(i)}(\boldsymbol{w}^T x^{(i)} + w_0) \geq 1 - \xi_i$ i $\xi_i \geq 0$.

Jezgrenim trikom idemo još korak dalje. Radi se o funkciji jezgre oblika

$$K(\boldsymbol{x}, \boldsymbol{y}) = (\phi(\boldsymbol{x}), \phi(\boldsymbol{y})) \tag{5.5}$$

koja preslikava vektore $\boldsymbol{x}$ i $\boldsymbol{y}$ u višu dimenziju kako bi se u toj višoj dimenziji klase mogle razdvojiti jednostavnim hiperravninama. U knjižnici `sklearn` nalazi se nekoliko jezgrenih funkcija, a to su:

Tablica 5.1: Jezgrine funkcije

Funkcija	$K(\boldsymbol{x}, \boldsymbol{y})$
Linearna	$\boldsymbol{x}^T\boldsymbol{y}$
Polinomna	$(\gamma * \boldsymbol{x}^T\boldsymbol{y} + r)^d$
RBF	$exp(-\gamma * ||\boldsymbol{x} - \boldsymbol{y}||^2)$
Sigmoidna	$tanh(\gamma * \boldsymbol{x}^T\boldsymbol{y} + r)$

Prilikom optimizacije hiperparametara u ovom radu korištene su linearna i RBF jezgrina funkcija.

5.2.2. Stablo odluke

Algoritam učenja stabla odluke osmislio je Ross Quinlan u svome radu *Induction of Decision Trees* 1986. godine [15]. Radi se o algoritmu koji od skupa podataka kreira stablo odluke. Prednosti stabla odluke su:

1. Bira najbolje atribute po kojima razdijeliti skup podataka, što smanjuje šansu da se koriste atributi s malim utjecajem na odluku,;

2. tolerantno na nedostajuće vrijednosti;

3. jednostavno za interpretaciju.

Loše strane stabla su da može jednostavno doći do prenaučenosti ili prevelike kompleksnosti stabla ako je skup podataka za učenje premalen ili ako podaci imaju šum. Takve situacije se obično rješavaju podrezivanjem stabla na određenoj dubini.

U ovom odjeljku bit će objašnjen način na koji taj algoritam radi, prema radu [15] i stranici [1].

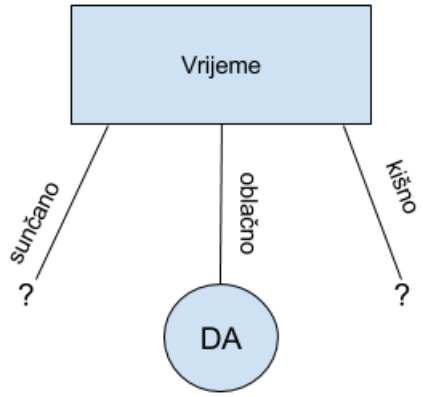
Algoritam radi tako da za svaki stupanj stabla odabire atribut koji najbolje klasificira primjere. Taj atribut postaje čvor, a njegove vrijednosti postaju grane. Potom se skup podataka razdijeljuje prema vrijednostima atributa te se cijeli postupak ponavlja za sljedeći stupanj stabla. Primjerice, imamo skup podataka kao što je prikazano u tablici 5.2.

Tablica 5.2: Skup podataka

Vrijeme	Temperatura	Vlaga	Vjetar	Igrati nogomet
Sunčano	Vruće	Visoka	Slab	Ne
Sunčano	Vruće	Visoka	Jak	Ne
Oblačno	Vruće	Visoka	Slab	Da
Kišno	Umjereno	Visoka	Slab	Da
Kišno	Hladno	Normalna	Slab	Da
Kišno	Hladno	Normalna	Jak	Ne
Oblačno	Hladno	Normalna	Jak	Da
Sunčano	Umjereno	Visoka	Slab	Ne
Sunčano	Hladno	Normalna	Slab	Da
Kišno	Umjereno	Normalna	Slab	Da
Sunčano	Umjereno	Normalna	Jak	Da
Oblačno	Umjereno	Visoka	Jak	Da
Oblačno	Vruće	Normalna	Slab	Da
Kišno	Umjereno	Visoka	Jak	Ne

Algoritam će pretražiti po kojem atributu je najbolje učiniti razdvajanje. U ovom slučaje taj atribut je vrijeme. Kada je vrijeme oblačno, u skupu poda-

taka piše da je uvijek dobro igrati nogomet. Nakon razdvajanja podataka prema atributu vrijeme stablo izgleda kao na slici 5.2.



Slika 5.2: Stablo odluke nakon prvog grananja

Pretraživanje najboljeg atributa radi se računanjem informacijske dobiti – vrijednosti koja govori koliko dobro pojedini atribut odjeljuje primjere za učenje u skladu s ciljnom klasifikacijom. Kako bismo mogli izračunati informacijsku dobit, potrebno nam je da izračunamo entropiju. Entropija se računa prema formuli:

$$E(S) = \sum_{i=1}^{c} -p_i \log_2 p_i, \tag{5.6}$$

gdje je S skup podataka nekog atributa, a c je broj različitih vrijednosti tog atributa. Informacijska dobit računa se prema:

$$I(S, A) = E(S) - \sum_{j=1}^{c} \frac{|S_j|}{|S|} E(S_j). \tag{5.7}$$

Varijabla A predstavlja atribut za koji se računa informacijska dobit, c je broj različitih vrijednosti tog atributa, a S_j je skup podataka S uz vrijednost atributa j.

5.2.3. Klasifikator slučajne šume

Klasifikator slučajne šume spada u algoritme ansambla (engl. *ensemble*), odnosno u algoritme koji kombiniraju više predviđanja različitih algoritama kako bi dobili bolje predviđanje. Prvi algoritam slučajnih šuma osmislio je Tin Kam Ho

[11], dok je moderni algoritam osmišljen od Lea Breimana i Adele Cutler [8]. Objašnjenje algoritma napisano je prema [4].

Algoritam se temelji na stablima odluke i agregaciji *bootstrapa* (engl. *bootstrap aggregation*).

Bootstrap je metoda uzimanja uzoraka gdje se iz skupa podataka uzima mnogo podskupova uz dopušteno ponavljanje istih. Na svakom od tih podskupova se izračuna vrijednost te se za krajnji rezultat uzima prosjek tih vrijednosti.

Primjena *bootstrap* metode na stabla odluke izgleda ovako:

1. Uzmi mnogo podskupova skupa podataka,
2. Nauči stablo odluke na svakom od tih podskupova podataka,
3. Izračunaj predviđanje svakog modela te uzmi predviđanje koje se najviše puta pojavilo.

U ovom slučaju stabla koja se nauče će biti u visokoj korelaciji, odnosno imat će sličnu strukturu što nam onda zapravo ne predstavlja veliku razliku od korištenja samo jednog stabla. Algoritam slučajne šume popravlja taj problem visoke korelacije tako da svako stablo dobije samo slučajan podskup značajki prema kojima može napraviti strukturu stabla.

Prednost slučajne šume nasuprot stabla odluke je u tome što se ne moramo zamarati prenaučenošću stabala, te se stoga stabla pušta da imaju mnogo razina i ne podrezuje ih se.

5.2.4. Naivan Bayesov klasifikator

Naivan Bayesov klasifikator je algoritam osmišljen na temelju Bayesovog teorema (formula 5.8), ali s pretpostavkom da su atributi unutra razreda nezavisni (formula 5.9). Razrada naivnog Bayesovog algoritma napravljena je prema [2].

$$P(h|x) = \frac{P(x|h)P(h)}{P(x)}. \tag{5.8}$$

$$P(\mathbf{x}|h) = P(x_1, \ldots, x_T|h) = \Pi_t P(x_t|h). \tag{5.9}$$

Gornje formule zahtijevaju malo objašnjenja. $P(h|x)$ je vjerojatnost pojave hipoteze(klase) h ako je zadan vektor značajki x. $P(x|h)$ je očigledno vjerojatnost pojave vektora značajki x ako je dana hipoteza h. $P(h)$ je vjerojatnost pojave hipoteze h dok je $P(x)$ vjerojatnost pojave vektora značajki x.

Kako bi se dobilo rješenje klasifikacije, naivni Bayesov klasifikator računa maksimalnu aposteriornu hipotezu ($P(h|x)$) (engl. *maximum A Posteriori, MAP*). MAP hipoteza slijedi iz formule 5.8.

$$h_{map} = argmax_{h_i \epsilon H} P(x|h_i)P(h_i). \tag{5.10}$$

$P(x)$ se izostavlja jer je konstanta te stoga ne utječe na izračun maksimalne aposteriorne vrijednosti.

U ovome radu korištena je implementacija naivnog Bayesovog klasifikatora iz knjižnice `sklearn`. Isprobani su modeli naivnog Bayesovog klasifikatora sa Gaussovom i Bernoullijevom razdiobom, pri čemu je Bernoullijeva razdioba dala mnogo bolje rezultate.

5.3. Implementacija modela

Model je implementiran pomoću knjižnice `sklearn` u kojoj se nalaze svi gore navedeni algoritmi. Implementacija se svela na tri dijela: na testiranje algoritama i značajki, odabir najboljeg algoritma te na treniranje i spremanje krajnjeg modela.

Testiranje algoritama sastojalo se od kreiranja matrice značajki od skupa podataka za učenje (pododjeljak 5.3.1), podešavanja hiperparametara (više u pododjeljku 5.3.2) te izračunavanja točnosti naučenog modela (pododjeljak 5.3.3). Prema uobičajenoj praksi testiranja algoritama, skup podataka za učenje smo razdvojili na dva dijela, jedan za učenje (80%) te drugi za testiranje (20%). To razdvajanje smo napravili funkcijom `train_test_split` koja se nalazi u modulu `sklearn.model_selection`.

Poslije odabira najtočnijeg algoritma, model je naučen na cijelom skupu podataka te je pohranjen na disk pomoću `joblib.dump` funkcije koja se nalazi u modulu `sklearn.externals`.

5.3.1. Značajke

Značajke koje su korištene prilikom predviđanja vrste pitanja su sljedeće:

1. Vreća riječi
2. Vektorska reprezentacija riječi

3. Bigrami, odnosno, vreća bigrama
4. Broj slova u upitu
5. Broj različitih slova u upitu
6. Broj riječi u upitu
7. Nalazi li se riječ *by* u upitu
8. Redni broj upita unutar razgovora
9. Nalazi li se broj u upitu

Povodili smo se pristupom "kuhinjskog sudopera" (engl. *kitchen sink*), odnosno ubacivanja svih značajki koje se mogu jednostavno izračunati.

Vreća riječi i vektorska reprezentacija riječi detaljnije su opisani u pododjeljku 4.3.1. Za neke od preostalih značajki bi bilo dobro istaknuti motivaciju. Za sedmu značajku je motivacija bila da ako se u upitu nalazi engleska riječ *by*, to povećava šansu da se radi o ORDER_BY ili GROUP_BY, te u rijetkim slučajevima (npr., *Show me all movies directed by Stanley Kubrick*) o klasi WHERE. Slična je motivacija i za devetu značajku, naime ako se nalazi broj unutar upita, vjerojatnije je da se radi o klasama WHERE ili LIMIT. Redni broj upita unutar razgovora može također pomoći klasifikaciji, najčešće se na početku nalaze upiti koji spadaju u klasu WHERE dok se pred kraj nalaze ORDER_BY, GROUP_BY te upiti agregacije.

5.3.2. Podešavanje hiperparametara

Problem podešavanja hiperparametara detaljno je opisan u pododjeljku 4.3.2 te ovdje neće biti osvrta na teoriju koja stoji iza toga. No, prilikom podešavanja hiperparametara u problemu predviđanja vrste pitanja koristili smo gotovu funkciju za pretragu rešetke te za unakrsnu validaciju. Ta funkcija nalazi se u modulu `sklearn.model_selection` te se zove `GridSearchCV`. Ta funkcija je svela implementaciju podešavanja hiperparametara samo na odabir vrijednosti koje će se isprobati za hiperparametre.

5.3.3. Bodovanje

Bodovanje rezultata nije univerzalno, odnosno postoji više različitih metoda bodovanja koje daju različite rezultate, stoga treba uvijek odabrati metodu bodovanja koja najviše odgovara trenutnom problemu. Najčešće se govori o mjerama preciznosti (engl. *precision*) i odziva (engl. *recall*) za klasifikacijske probleme.

Uvedimo pojmove ispravnih pozitivnih rezultata (TP), ispravnih negativnih rezultata (TN), lažnih pozitivnih rezultata (FP) te lažnih negativnih rezultata (FN).

Preciznost je definirana kao omjer broja ispravnih pozitivnih rezultata i broja vraćenih pozitivnih rezultata. Npr., recimo da je model vratio 12 pozitivnih rezultata, od kojih su zapravo 3 negativna (lažna pozitivna). To znači da je preciznost u tom slučaju $\frac{9}{12}$. Formula izračuna preciznosti:

$$P = \frac{TP}{TP + FP}. \tag{5.11}$$

Odziv je pak definiran kao omjer broja ispravnih pozitivnih rezultata i ukupnog broja pozitivnih rezultata. U primjeru od maloprije, ako kažemo da je ukupno bilo 15 pozitivnih rezultata, od kojih nam je model vratio 12, tada je odziv $\frac{9}{15}$.

$$R = \frac{TP}{TP + FN}. \tag{5.12}$$

Također bitno je spomenuti pojmove mikro-prosječne (engl. *micro-averaged*) i makro-prosječne (engl. *macro-averaged*) evaluacijske mjere. Obje su nam iznimno korisne u klasifikacijskim sustavima. Ako imamo neravnomjerno zastupljene klase u skupu podataka, mikro-prosječna mjera će biti nagnutija klasama koje su više zastupljene dok će makro-prosječna mjera biti nagnutija klasama koje su manje zastupljene. Ako imamo višu mikro-prosječnu vrijednost to znači da nam sustav bolje klasificira klase koje su više zastupljene.

Mikro-prosječna preciznost dana je formulom 5.13 dok je mikro-prosječan odziv dan formulom 5.14. Makro-prosječna preciznost dana je formulom 5.15, a makro-prosječan odziv dan formulom 5.16.

$$P = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + FP_i}, \tag{5.13}$$

$$R = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + FN_i}. \tag{5.14}$$

$$P = \frac{\sum_{i=1}^{n} P_i}{n}, \tag{5.15}$$

$$R = \frac{\sum_{i=1}^{n} R_i}{n}. \tag{5.16}$$

U ovom radu korištene su mjere točnosti (engl. *accuracy*), F1-mikro (engl. *F1-micro*) te F1-makro (engl. *F1-macro*) mjera. U nastavku je dan pregled izračuna svake od tih mjera.

Točnost se računa kao omjer ispravno klasificiranih primjera i ukupnog broja primjera.

$$ACC = \frac{\sum_{i=1}^{n} T_i}{\sum_{i=1}^{n} T_i + N_i}, \tag{5.17}$$

gdje je n broj klasa, T_i broj točno klasificiranih primjera za tu klasu i N_i broj krivo klasificiranih primjera za tu klasu.

Mjera F1 se računa kao harmonijska sredina preciznosti i odziva. Ako se radi o mikro-prosječnoj mjeri F1 uzimaju se formule za preciznost i odziv 5.13 i 5.14, a u slučaju makro prosječne mjere F1 uzimaju se formule 5.15 i 5.16.

$$F1 = 2 \cdot \frac{P \cdot R}{P + R}. \tag{5.18}$$

S obzirom na to da klase u skupu podataka koji imamo nisu ujednačene (vidi tablicu 3.1), F1 mikro će nam davati više rezultate, jer će naš model bolje pogađati više zastupljene klase nego one za koje imamo samo nekoliko primjera. Takvi rezultati vidjet će se u sljedećem odjeljku.

5.4. Rezultati

U nastavku će biti dani rezultati treniranja svakog od isprobanih algoritama u dva slučaja, kada je uključena vreća riječi kao značajka (5.3) i kada vreća riječi nije uključena kao značajka (tablica 5.4). Osim vreće bigrama, sve druge značajke su uključene u vektor. Za optimizaciju hiperparametara razdvojen je skup podataka na skup podataka za treniranje koji sadrži 235 primjera i skup podataka za testiranje koji sadrži 59 primjera. Potom je korištena unakrsna provjera sa 5 preklopa nad skupom podataka za treniranje, gdje od ukupno 235 primjera svaki preklop sadrži 47 primjera. Rezultati prikazani u tablicama dobiveni su nad skupom za testiranje pomoću funkcije `cross_val_score` uz broj preklopa postavljen na pet. Funkcija će izračunati bodovanje na pet različitih razdioba skupa

podataka te će vratiti rezultate za svaki. Rezultati koji su prikazani u tablici su srednje vrijednosti tih 5 različitih bodovanja. Uz to bi trebalo napomenuti da su standardne devijacije kod nekih algoritama išle i do 20% (pojačanje gradijenta, k najbližih susjeda i naivni Bayes sa Gaussovom razdiobom).

Tablica 5.3: Bodovanje algoritama sa vrećom riječi

Algoritam	Točnost	F1-mikro	F1-makro
Naivni Bayes s Bernulijevom razdiobom	**78,2** %	**78,2** %	**65,2** %
Slučajna šuma	71,3%	71,3%	54,2%
SVM	66,5%	66,5%	54,3%
Naivni Bayes s Gausovom razdiobom	64,7%	64,7%	58,1%
Stablo odluke	60,6 %	60,6 %	51,9 %
K najbližih susjeda	65,5%	65,5%	49,9%
Pojačanje gradijenta	57,7 %	57,7 %	46,9 %

Tablica 5.4: Bodovanje algoritama bez vreće riječi

Algoritam	Točnost	F1-mikro	F1-makro
Naivni Bayes s Bernoullijevom razdiobom	**74,8** %	**74,8** %	**60,6** %
Slučajna šuma	72,2%	72,2%	59,8%
SVM	66,8%	66,8%	51,3%
Naivni Bayes s Gaussovom razdiobom	65%	65%	45,6%
Stablo odluke	62,6 %	62,6 %	50 %
K najbližih susjeda	61,4%	61,4%	46,1%
Pojačanje gradijenta	60,4 %	60,4 %	51,8 %

Može se primijetiti da se rezultati razlikuju, te da su bolji kada se uključi vreća riječi kod većine algoritama, osim kod algoritama baziranih na stablima odluke (slučajna šuma i pojačanje gradijenta). Ipak, naivan Bayes sa Bernoullijevom razdiobom se pokazao kao najtočniji te je stoga on odabran kao model za predviđanje vrste pitanja.

Kao osnovni model sa kojim ćemo uspoređivati rezultate odabran je model koji sve upite klasificira u klasu WHERE. Rezultati koje taj model daje prikazani su u tablici 5.5.

Prema rezultatima se vidi da je osnovni model lošiji od onog dobivenog algoritmom naivnog Bayesa sa Bernoullijevom razdiobom. Kako bismo ustanovili je

Tablica 5.5: Osnovni model

Algoritam	Točnost	F1-mikro	F1-makro
Osnovni model	40,4 %	40,4 %	4,1 %

li to slučaj, proveden je permutacijski test kao što je objašnjeno u odjeljku 4.4. Rezultati koji su dobiveni tim testom prikazani su u tablici 5.6.

Tablica 5.6: Rezultati permutacijskog testa

Bodovanje	p-vrijednost
Točnost	0.01
F1-mikro	0.01
F1-makro	0.003

Ovi rezultati testa pokazuju da je uistinu odabrani algoritam bolji od osnovnog modela, na razini značajnosti od 5%. U nastavku je prikazano nekoliko primjera ispravne i neispravne klasifikacije (tablica 5.7).

Tablica 5.7: Primjeri

Upit	Klasifikacija	Ispravna klasifikacija
Give me all actors from europe	WHERE	WHERE
Order them by age	ORDER_BY	ORDER_BY
Group actors by revenue	WHERE	GROUP_BY
List me all movies by Stanley Kubrick	WHERE	WHERE
Top movie by revenue	LIMIT	TOP
Max of their revenues	MAX	MAX
Sum me their revenues	MAX	SUM
Only distinct actors	DISTINCT	DISTINCT

6. Zaključak

Kako bi se smanjila barijera potrebna za korištenje velikih količina podataka spremljenih u baze podataka, osmišljena su jezična sučelja bazama podataka. Cilj ovoga rada bio je izgraditi modele za otkrivanja novih vrijednosti te predviđanje vrste pitanja u jezičnom sučelju bazi podataka. Ti problemi obrađeni su korištenjem tehnika strojnog učenja (engl. *machine learning*) i obrade prirodnog jezika (engl. *natural language processing*).

Obrađeni su problemi otkrivanja novih vrijednosti te predviđanja vrste pitanja. Otkrivanje novih vrijednosti riješeno je učenjem jednoklasnog stroja potpornih vektora s vektorskom reprezentacijom riječi (engl. *word embeddings*). Na skupu podataka za testiranje koji sadrži pozitivne i negativne primjere (prema tablici 3.2) dobiveni su rezultati od 98% točnosti za tipične te 82% za netipične vrijednosti. Iako su to dobri rezultati, tijekom interaktivnog testiranja sustava s novim upitima ponekad se događalo da su odbačeni dobri upiti. Bolja preciznost bi se mogla postići treniranjem na većem skupu podataka i dodavanjem drugih značajki.

Za predviđanje vrste pitanja razvijen je model temeljen na algoritmu naivnog Bayesa s Bernoullijevom razdiobom. Značajke koje su korištene opisane su u potpoglavlju 5.3.1. Najbolji rezultati dobiveni su kada smo uključili sve značajke osim vreće bigrama te tada ti rezultati iznose 78% za točnost i F1-mikro bodovanje, a 65% za F1-makro bodovanje. Tijekom interaktivnog testiranja predviđanja vrste pitanja klase `WHERE` i `ORDER BY` su bile najviše puta pogođene dok su klase s niskom zastupljenošću u skupu podataka bile rijetko pogođene. Povećanjem skupa podataka za testiranje postigla bi se bolja preciznost za slabije zastupljene klase, ali bi i rezultati bodovanja bili konzistentniji, ne bi toliko ovisili o podskupu na kojem se boduje. Podešavanjem značajki te osmišljavanjem novih rezultati bi se još više mogli poboljšati.

Literatura

[1] Decision tree - classification. http://www.saedsayad.com/decision_tree.htm. Accessed: 2017-05-29.

[2] Naive-bayes classification algorithm. http://software.ucv.ro/~cmihaescu/ro/teaching/AIR/docs/Lab4-NaiveBayes.pdf. Accessed: 2017-05-30.

[3] Introduction to one-class support vector machines. http://rvlasveld.github.io/blog/2013/07/12/introduction-to-one-class-support-vector-machines/. Accessed: 2017-05-28.

[4] Bagging and random forest ensemble algorithms for machine learning. http://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/. Accessed: 2017-05-30.

[5] Stroj potpornih vektora (svm). http://nbviewer.jupyter.org/github/jsnajder/StrojnoUcenje/blob/master/notebooks/SU-2015-8-SVM.ipynb. Accessed: 2017-05-30.

[6] Vector representations of words. https://www.tensorflow.org/tutorials/word2vec. Accessed: 2017-05-29.

[7] Bernhard E. Boser, Isabelle M. Guyon, i Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. U *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, stranice 144–152, New York, NY, USA, 1992. ACM. ISBN 0-89791-497-X. doi: 10.1145/130385.130401. URL http://doi.acm.org/10.1145/130385.130401.

[8] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, Listopad 2001.

ISSN 0885-6125. doi: 10.1023/A:1010933404324. URL http://dx.doi.org/10.1023/A:1010933404324.

[9] Ann Copestake i Karen Sparck Jones. Natural language interfaces to databases. *The Knowledge Engineering Review*, 5(4):225–249, 1990. doi: 10.1017/S0269888900005476.

[10] Corinna Cortes i Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, Rujan 1995. ISSN 0885-6125. doi: 10.1023/A:1022627411411. URL http://dx.doi.org/10.1023/A:1022627411411.

[11] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(8):832–844, Kolovoz 1998. ISSN 0162-8828. doi: 10.1109/34.709601. URL http://dx.doi.org/10.1109/34.709601.

[12] Larry M. Manevitz i Malik Yousef. One-class svms for document classification. *J. Mach. Learn. Res.*, 2:139–154, Ožujak 2002. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944790.944808.

[13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, i Jeff Dean. Distributed representations of words and phrases and their compositionality. U *Advances in Neural Information Processing Systems 26*, stranice 3111–3119. Curran Associates, Inc., 2013. URL http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and\-their-compositionality.pdf.

[14] Marco A.F. Pimentel, David A. Clifton, Lei Clifton, i Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99:215 – 249, 2014. ISSN 0165-1684. doi: https://doi.org/10.1016/j.sigpro.2013.12.026. URL http://www.sciencedirect.com/science/article/pii/S016516841300515X.

[15] J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, Ožujak 1986. ISSN 0885-6125. doi: 10.1023/A:1022643204877. URL http://dx.doi.org/10.1023/A:1022643204877.

[16] Bernhard Schölkopf, Robert C Williamson, Alexander J Smola, John Shawe-Taylor, John C Platt, et al. Support vector method for novelty detection. U *NIPS*, svezak 12, stranice 582–588, 1999.

[17] Vic Barnett, Toby Lewis. *Outliers in Statistical Data*. 1994.

[18] Dell Zhang i Wee Sun Lee. Question classification using support vector machines. U *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, stranice 26–32, New York, NY, USA, 2003. ACM. ISBN 1-58113-646-3. doi: 10.1145/860435.860443. URL `http://doi.acm.org/10.1145/860435.860443`.

Predviđanje vrste pitanja za jezično sučelje bazi podataka

Sažetak

S ciljem približavanja baza podataka ljudima bez znanja potrebnih za upravljanje istima počela su se razvijati jezična sučelja bazama podataka. Kako bi se izgradilo stabilno jezično sučelje nužno je odbacivati loše upite. Taj problem riješen je u ovom radu uporabom metoda otkrivanja novih vrijednosti i strojnog učenja. Sljedeći korak pri izgradnji jezičnog sučelja je predviđanje vrste pitanja u upitu, kako bi se upit mogao ispravno prevesti u jezik upita. U opsegu ovog rada taj problem riješen je klasifikacijom u 14 klasa koristeći metode obrade prirodnog jezika i strojnog učenja. U radu su detaljno opisani algoritmi koji su korišteni te postupci kojima su se rezultati dobili. Skup podataka na kojima su modeli učeni ručno je izgrađen te označen.

Ključne riječi: Obrada prirodnog jezika, strojno učenje, jezično sučelja bazi podataka, stroj potpornih vektora, naivni Bayesov klasifikator

Predicting question type for natural language interface to database

Abstract

Natural language interfaces to databases began to develop with the goal to bring databases closer to the people with no knowledge about them. To make a natural language interface it is necessary to filter out spam queries. That problem is solved in this thesis by using methods of novelty detection and machine learning. The next step in creating a natural language interface is predicting the question type in query, in order for the query to be easily translated to query language. In the scope of this thesis that problem was solved by classification into 14 classes using methods of natural language processing and machine learning. Algorithms that were used in the thesis are also thoroughly explained as well as the methods by which we got results. Dataset on which the models were trained was compiled and labeled by hand.

Keywords: Natural language processing, machine learning, natural language interface, support vector machine, naive Bayes classifier