Problem binarne klasifikacije rečenica

Ovaj projekt nastao je u svrhu kolegija Strojno učenje na PMF - matematičkom odsjeku Sveučilišta u Zagrebu. Bavi se analizom prirodnog jezika, a problem koji rješava je klasifikacija dane rečenice na subjektivnu, odnosno objektivnu, koristeći dva klasifikatora, naive bayes i metodu potpornih vektora. Baza za učenje i testiranje sastoji se od 5000 subjektivnih (subj.txt) i 5000 objektivnih (obj.txt) rečenica.

Korišteni alati i instalacija

Program je pisan u programskom jeziku Python, na operativnom sustavu Windows.

NLTK - Natural language toolkit

Alat za obradu prirodnog jezika koji omogućava razne jednostavne, ali i mnoge složene zadatke u obradi teksta.
Budući da analiza subjektivnosti rečenica zahtjeva obradu teksta na različite načine, u ovom je projektu iscrpno korišten. Osim jednostavnijih zadataka poput pretvaranja rečenice u listu riječi, rješavao je i neke mnogo složenije poput izdvajanja bigrama, traženja sinonima riječi, izdvajanja nebitnih riječi (tzv. stopwords), pa čak i klasifikacije teksta pomoću Naive Bayes klasifikatora.
Upute za instalaciju i detaljnije informacije možete pronaći na http://nltk.org/

LIBSVM - A Library for Support Vector Machines

Za klasifikaciju rečenica metodom potpornih vektora korištena je ova biblioteka. U sklopu projekta implementirana je klasifikacija pomoću RBF jezgre i pomoću linearne jezgre te su uspoređeni rezultati. S obzirom na to da obje jezgre imaju varijabilne parametre (C i γ u RBF jezgri, odnosno C u linearnoj jezgri) koji reguliraju pristranost i varijancu, vrlo je važno dobro odrediti te parametre kako ne bi došlo do neželjenih efekata. Za određivanje parametara u RBF jezgri korištena je skripta grid.py koja je dio biblioteke. Skripta pomoću cross validacije određuje najbolje parametre iz zadanog skupa. Oslanja se i na gnuplot koji iscrtava graf po dobroti različitih parametara.
Upute za instalaciju i detaljnije informacije možete pronaći na
http://www.csie.ntu.edu.tw/~cjlin/libsvm/
U skopu projekta napravljena je i funkcija za određivanje parametara koja se može koristiti za obje jezgre. Korištena je za određivanje parametara u linearnoj jezgri te u nekim testiranjima i za određivanje parametara RBF jezgre.

SCIKIT - LEARN:

Budući da je unutar ovog projekta za odabir značajki korištena “bag of words” metoda uz mogućnost dodavanja bigrama, skup iz kojeg je potrebno izdvojiti potencijalno dobre značajke je jako velik. Ovaj alat nam omogućuje odabir podskupa značajki te redukciju dimenzionalnosti (eng. feature selection).
SCIKIT - LEARN oslanja se na numpy, scipy i matplotlib. Unutar ovog projekta korištena je metoda koja na temelju univarijantnih statističkih testova smanjuje dimenziju prostora značajki te odabire one najkorisnije. Precizno, metoda koja je korištena prima veliki skup potencijalnih značajki te na temelju chi2 testa odabire k najboljih, gdje je k fiksan parametar. Detalje o korištenoj funkciji (SelectKBest) i upute za instalaciju možete saznati na: http://scikit-learn.org/stable/modules/feature_selection.html

Metode odabira značajki

Budući da je značajke moguće odabrati na više načina, ukratko ćemo dati opis pojedinih funkcija za odabir te opisati njihove argumente i način pozivanja.

Funkcija **`najveca_frekvencija(obj,subj,dg1,gg1,dg2,gg2)`** vraća listu riječi koje se pojavljuju u dokumentu sortirane silazno po frekvenciji.
Argumenti:
`obj` i `subj` - definiraju dokument
`dg1,gg1,dg2,gg2` - definiraju intervale rečenica u dokumentu iz kojih punimo navedenu listu. Omogućuju čitanje riječi samo iz skupa za učenje da ne bi došli u doticaj s informacijama o riječima u skupu za testiranje.
Nakon što napunimo listu sa svim riječima koristimo gore opisani NLTK alat za sortiranje po frekvenciji te vraćanje liste bez ponavljanja `(nltk.FreqDist(all_words))`.

Funkcija **`omjer_pojavljivanja(documents,dg1,gg1,dg2,gg2)`** vraća riječi iz skupa za učenje sortirane po omjeru pojavljivanja u subjektivnim, odnosno objektivnim rečenicama. Pretpostavka je da će riječi koje se puno češće pojavljuju u jednoj od tih skupina biti vrlo korisne za klasifikaciju. Riječi koje se pojavljuju samo u jednoj skupini smatramo korisnima ako su se pojavile dovoljan broj puta
Argumenti:
`documents` - skup svih rečenica iz kojeg ćemo izdvojiti skup za treniranje
`dg1,gg1,dg2,gg2` - dva intervala oblika [dg1,gg1] i [dg2,gg2] koja omogućavaju izdvajanje skupa za treniranje od skupa svih dokumenata. Skup za testiranje čini ostatak dokumenata

Funkcija **`otkloni_nepozeljne(all_words,koliko)`** služi za otklanjanje nepoželjnih riječi iz danog skupa riječi.
Argumenti:
`all_words` - dani skup riječi koji filtriramo
`koliko` - koliko najfrekventnijih riječi iz filtriranog skupa želimo da funkcija vrati.
Riječ smatramo nepoželjnom ukoliko je interpunkcija, veznik, broj ili prijedlog. Za određivanje vrste riječi koristimo NLTK (`nltk.pos_tag(all_words)`).

Funkcija **`bigrami(documents,dg1,gg1,dg2,gg2)`** vraća bigrame u dokumentu.
Argumenti:
`documents` - skup svih rečenica
`dg1,gg1,dg2,gg2` - granice u dokumentu koje omogućuju uzimanje bigrama samo iz skupa za učenje.
Prilikom vraćanja bigrama, sortiramo ih po frekvenciji te izbacujemo one kod kojih je jedna od riječi stop_word ili interpunkcija. Navedene informacije o riječima dobivamo koristeći NLTK alat.

Funkcija **`izbaci_stop_words(all_words)`** koristi NLTK kako bi izbacila stop words iz danog skupa riječi.
Argumenti:
`all_words` - skup riječi koji želimo filtrirati

Funkcija **`stem(word)`** razdvaja riječ po apostrofu te vraća prvi dio riječi. Npr za word = “I’m” vraća I. Koristi NLTK.

Funkcija **`feature_selection(documents,koliko,dg1,gg1,dg2,gg2)`**
iz danog skupa featura vraća podskup koristeći SCIKIT - LEARN.
Argumenti:
`documents` - skup svih rečenica
`koliko` - koliko značajki želimo da funkcija vrati,ulazi kao argument u funkciju SelectKBest
`dg1,gg1,dg2,gg2` - granice u dokumentu
Funkcija SelectKBest odabire podskup značajki na temelju statističkog chi2 testa.

Funkcije za odabir parametara i testiranje

Funkcije **`words_for_feature_selection(document)`** i **`document_features(document)`** prolaze kroz dokument i značajke i zapisuju koje se značajke pojavljuju u dokumentu.
Argumenti:
`document` - rečenica
Služe kao priprema dokumenata za feature selection, odnosno za testiranje.

Funkcija **`sinonimi(documents,word_features,dg,gg)`** za riječi u skupu za testiranje koje nisu među značajkama traži pojavljuje li se njihov sinonim u značajkama i ako se nalazi, zamjenjuje riječ njenim sinonimom.
Argumenti:
`documents` - skup svih rečenica
`word_features` - značajke
`dg,gg` - granice intervala u kojem se nalazi skup za testiranje

Funkcije iz navedenog NLTK alata koje služe za treniranje i testiranje Naive Bayes modela:

```
nltk.NaiveBayesClassifier.train(train_set)
nltk.classify.accuracy(classifier, test_set)
```

Opis funkcija pogledati na http://nltk.org/.

Funkcije s primjerima poziva iz LIBSVM alata koje služe za treniranje,testiranje i odabir parametara SVM modela:

```
svm_train(px, param)
svm_predict(test_classes,test_set,m)
svm_problem(train_classes,train_set)
svm_parameter('-t 0')
find_parameters('D:\ADRESA_NA_KOJOJ_SE_NALAZE_PODACI', '-log2c
0,4,1 -log2g -10,-4,2')
```

Opis funkcija pogledati na http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

Funkcija **`nas_grid(tr_classes, tr_set, val_classes, val_set, cv, pocetnic, zavrsnic, korakc, pocetnig, zavrsnig, korakg, tip)`** traži najbolje parametre za rješavanje problema metodom potpornih vektora.

Argumenti:
`tr_classes, tr_set` - skup rečenica za treniranje odvojen na vektor klasa kojem rečenice pripadaju i vektor značajki svake rečenice
`val_classes, val_set` - skup rečenica za validaciju odvojen na analogan način. Koristi se samo ukoliko ne provodimo cross validaciju
`cv` - parametar za cross validaciju. Ako je ne želimo provesti, ovaj parametar mora biti 0. Inače se provodi cross validacija s `cv` podskupova
`pocetnic, zavrsnic, korakc` - granice koje određuju skup za testiranje parametra C. Testiranje počinje s C= $2^{pocetnic}$, prođe sve parametre oblika C= $2^{l*korakc}$ i završava s C= $2^{zavrsnic}$
`pocetnig, zavrsnig, korakg` - granice koje određuju skup za testiranje parametra γ, analogne su granicama za parametar C. Ukoliko ne želite testirati ovaj parametar, kao npr. u slučaju linearne jezgre, ulazni podaci trebaju biti oblika `pocetnig=zavrsnig` i `korakg>0`
`tip` - parametar koji označava tip jezgre i poprima jednake vrijednosti kao i u libsvm-u. 0 označava linearnu jezgru, 2 označava RBF jezgru

Funkcija `def ispuni_datoteku(ime_datoteke)` ispunjava datoteku podacima u formatu prilagođenom za funkciju `find_parameters` (grid.py)
Argumenti:
`ime_datoteke` - naziv datoteke koju želimo stvoriti i ispuniti podacima. Kasnije tu istu datoteku prosljeđujemo funkciji `find_parameters` (u kojoj je potrebno napisati cijeli put do datoteke)

K-struka cross validacija

Programi se pokreću korištenjem k-struke cross validacije, gdje je parametar k varijabilan i moguće ga je unijeti s komandne linije. Razlog zbog kojeg smo k - struku cross validaciju implementirale same je taj što značajke (eng. featuri) ovise o odabranom skupu za učenje te nam navedeni alati nisu omogućili taj način pozivanja funkcija za trening modela. Gotove funkcije koje koriste cross-validaciju bi primale fiksan skup featura te točnost koju bi dobile na taj način ne bi bila relevantna.
U svrhu dobivanja relevantne točnosti algoritama uzele smo usrednjenu vrijednost točnosti dobivenih cross validacijom.

Primjer pokretanja

naivebayes.py:

Program koristi Naive Bayes klasifikator iz NLTK alata.

Primjer pokretanja programa koristeći cmd/terminal:

Argumente komandne linije moguće je podesiti i u kodu (15-36 linija), taj dio koda izgleda ovako:

```
#-----PARAMETRI KOJE JE MOGUCE PODESITI-----#

#zelite li koristiti i bigrame
KORISTENJE_BIGRAMA = 0
#odabirom najveca frekvencija = 0, koristi se omjer_pojavljivanja
NAJVECA_FREKVENCIJA = 1
#koristenje otkloni nepozeljne
OTKLONI_NEPOZELJNE = 0
#koristenje stem-a
STEM = 0
#koristenje izbaci stop words
STOP = 0
#koristenje k-struke cross validacije
K = 4
#koristenje feature selectiona
FEATURE_SELECTION = 0
#koristenje sinonima kod pripreme test seta
SINONIMI = 0
#broj faetura
BROJ_FEATURA = 2000

#-----------------------------------------#
```

Argumenti se preko komandne linije unose gore navedenim predosljedom. Potrebno je unijeti svih 9 argumenata. Ukoliko jedan od argumenata nije unešen uzimaju se vrijednosti zapisane u kodu.

Primjer poziva iz cmd-a:
> python naivebayes.py 0 1 0 0 0 3 0 0 3000
Navedeni poziv za značajke koristi 3000 najfrekventnijih riječi i 3-struku cross validaciju.

svmsvm.py:

Program koristi svm.py iz biblioteke LIBSVM i grid.py iz iste biblioteke.
Pokretanje iz komandne linije analogno je pokretanju naivebayes.py.
Za korištenje skripte grid.py (odnosno funkcije find_parameters) potrebno je podesiti put do skripte svm-train.exe i do skripte pgnuplot.exe na početku skripte grid.py.
Isto tako, potrebno je u kodu (linije 345 i 349) podesiti ime datoteke u koju želite spremiti podatke, kao i put do nje.
Za mijenjanje skupa u kojem se traže parametri pomoću funkcije find_parameters potrebno je promijeniti podatke u drugom stringu, analogno kao i u gore opisanoj funkciji nas_grid (u sljedećem primjeru, pocetnic=0, zavrsnic=4, korakc=2):

```
find_parameters('PUT_DO_DIREKTORIJA\podaci', '-log2c 0,4,2
-log2g -8,-4,2')
```

Napomena: Grid.svm je iscrpna pretraga te zbog toga traje vrlo dugo. Ukoliko želite isključiti traženje najboljih parametara i uspisati svoje, potrebno je zakomentirati liniju koja poziva funkciju find_parameters i direktno upisati svoje parametre.
Analogno vrijedi za korištenje funkcije nas_grid.

Autori:

Hermina Petric Maretić i Anamarija Fofonjka