FILOZOFSKI FAKULTET SVEUČILIŠTA U ZAGREBU
ODSJEK ZA INFORMACIJSKE ZNANOSTI
2010/2011

Dario Novosel – Taradi

Stvaranje korpusa i njegova obrada pomoću NLTK-a

Završni rad

dr. sc. Nikola Ljubešić

Zagreb, 2011

Sadržaj

1. Uvod

Korpus je poveći i organizirani skup tekstova prikupljen prema određenim kriterijima. Koriste se za provedbu statističke analize te provjere raznih lingvističkih hipoteza, kao i za proučavanje jezika na općenitijoj razini. Dio ovog rada bit će posvećen stvaranju korpusa na primjeru foruma Filozofskog Fakulteta te primjeru njegove obrade – stvaranju naivnog Bayesovog klasifikatora koji postove klasificira prema odsjeku. U tu svrhu koristit će se programski jezik Python te Natural Language Toolkit – skup modula za obradu prirodnog jezika.

2. Korpusna lingvistika

Korpusna lingvistika je grana lingvistike koja jezik proučava kroz korpuse – skupove tekstova spremljenih na računalu. Principi korpusne lingvistike postojali su i prije razvoja računalne tehnologije. Leksikografi su za potrebe pisanja rječnika prikupljali tekstove da bi što točnije mogli opisati pojedine riječi. No moderna korpusna lingvistika počinje 1961, stvaranjem prvog modernog – računalnog – korpusa. Bio je to Brownov korpus, i sastojao se od oko milijuna riječi. (McEnery, 2003)

Razvojem korpusne lingvistike rasla je i veličina korpusa; druga generacija, primjerice COBUILD korpus, sastojao se od oko deset milijuna riječi. Današnji korpusi – na primjer BNC (British National Corpus) – sastoje se od oko stotinu milijuna riječi.

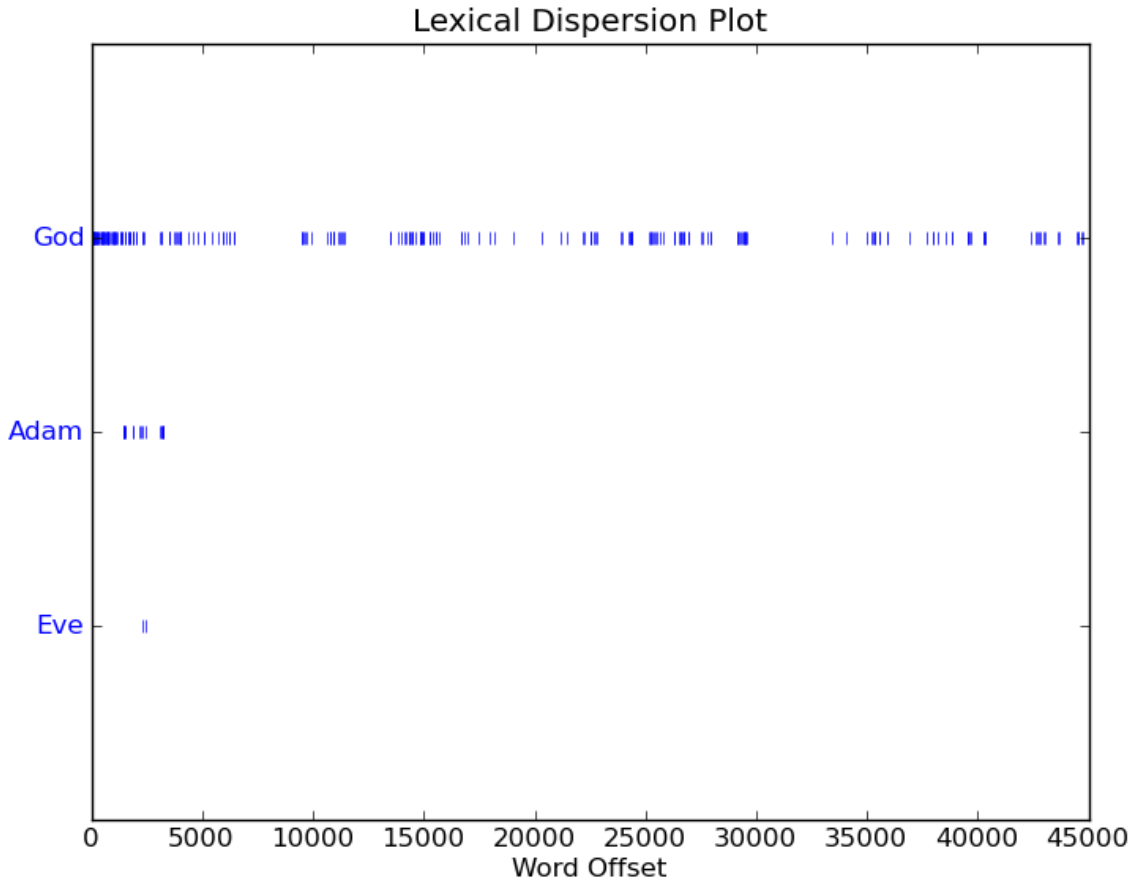
Postoji nekoliko podjela korpusa – mogu biti opće-jezični ili specijalizirani te jednojezični ili višejezični. Korpusi se mogu sastojati od tekstova izgovorene ili pisane riječi. Postovi na forumima, kao i direktna komunikacija preko online chatova spada negdje između kategorija izgovorene i pisane riječi.

Jedan od najčešćih načina pretraživanja podataka korpusa je konkordancija; popis pojava određene riječi, dijela riječi ili kombinacije riječi u kontekstu, izvučene iz korpusa. Riječ koja se traži ponekad se naziva ključna riječ.

Povijest konkordancije duža je od povijesti same korpusne lingvistike te datira još iz srednjeg vijeka. Originalna svrha konkordancije, koju su provodili redovnici u srednjem

vijeku, bila je stvaranje popisa najčešće analiziranih tekstova (npr. Biblije) koji su sadržavali indeksirane popise svih nefunkcijskih riječi izvornog djela, zajedno sa kontekstima u kojima se te riječi pojavljuju.

Ostale metode korištenja korpusa koriste statističke podatke bez analiziranja pojedinačnih dijelova teksta. Iako se takvim metodama gubi mnogo lingvističkih podataka, statistički-orijentirane metode su jedini razumno primjenjiv način analiziranja velikih korpusa. Štoviše, statističke metode analiziranja korpusa omogućavaju proučavanje jezičnih obrazaca i jezičnih pojava dajući uvid u jezik na razini iznad pojedinačnog iskustva te dajući objektivne dokaze učestalosti riječi ili jezičnih obrazaca. (McEnery, 2003)



Slika 1 – primjer grafičkog prikaza raspodjele riječi

Horizontalna os ovog prikaza predstavlja lokacije nađenih pojava riječi navedenih na vertikalnoj osi – u ovom primjeru, Adam i Eva se ne spominju nakon pet tisuća riječi, dok se "Bog" spominje kroz cijeli tekst.

3. NLTK

NLTK – Natural Language ToolKit – je skup Python modula koji pružaju mnoge tipove podataka za potrebe obrade prirodnog jezika, primjere i čitače korpusa i razne ugrađene algoritme. Primjeri i čitači korpusa uključuju Brownov korpus, CoNLL-2000 korpus, CMU rječnik izgovora, PP korpus afiksa, WordNet.

NLTK je prilagođen studentima koji počinju učiti obradu prirodnog jezika ili provode istraživanja u području obrade prirodnog jezika i povezanim područjima. NLTK je već uspješno korišten alat za poučavanje, samostalno učenje te kao osnova za stvaranje sustava za istraživanja. Na Filozofskom Fakultetu, NLTK se koristi za poučavanje, zadavanje zadataka te kao alat kod završnih projekata za kolegij Pretraživanje Obavijesti i Obrada Prirodnog Jezika.

Programski jezik NLTK-a je Python, koji je izabran zbog lakoće korištenja i učenja, intuitivne sintakse te dobre obrade tekst-varijabli. Nadalje, Python dopušta istraživanje NLTK-a kroz interpreter bez potrebe za kompajliranjem ili pisanjem cijelih programa prije testiranja. Budući da je objektno-orijentirani jezik, Python dopušta ponovno korištenje koda u daljnjim programima, a dolazi sa modulima za grafičko programiranje kao i numeričko procesiranje. (Bird, 2009)

Od alata korisnih za rad sa korpusima, NLTK sadrži funkciju za prikaz već spomenutih konkordancija – concordance():

```
Displaying 10 of 231 matches:
                                God created the heaven and the earth . An
 face of the deep . And the Spirit of God moved upon the face of the waters . A
ved upon the face of the waters . And God said , Let there be light : and there
 be light : and there was light . And God saw the light , that it was good : an
aw the light , that it was good : and God divided the light from the darkness .
ded the light from the darkness . And God called the light Day , and the darkne
 the morning were the first day . And God said , Let there be a firmament in th
vide the waters from the waters . And God made the firmament , and divided the
above the firmame and it was so . And God called the firmament Heaven . And the
the morning were the second day . And God said , Let the waters under the heave
```

Slika 2 – konkordancija riječi "God"

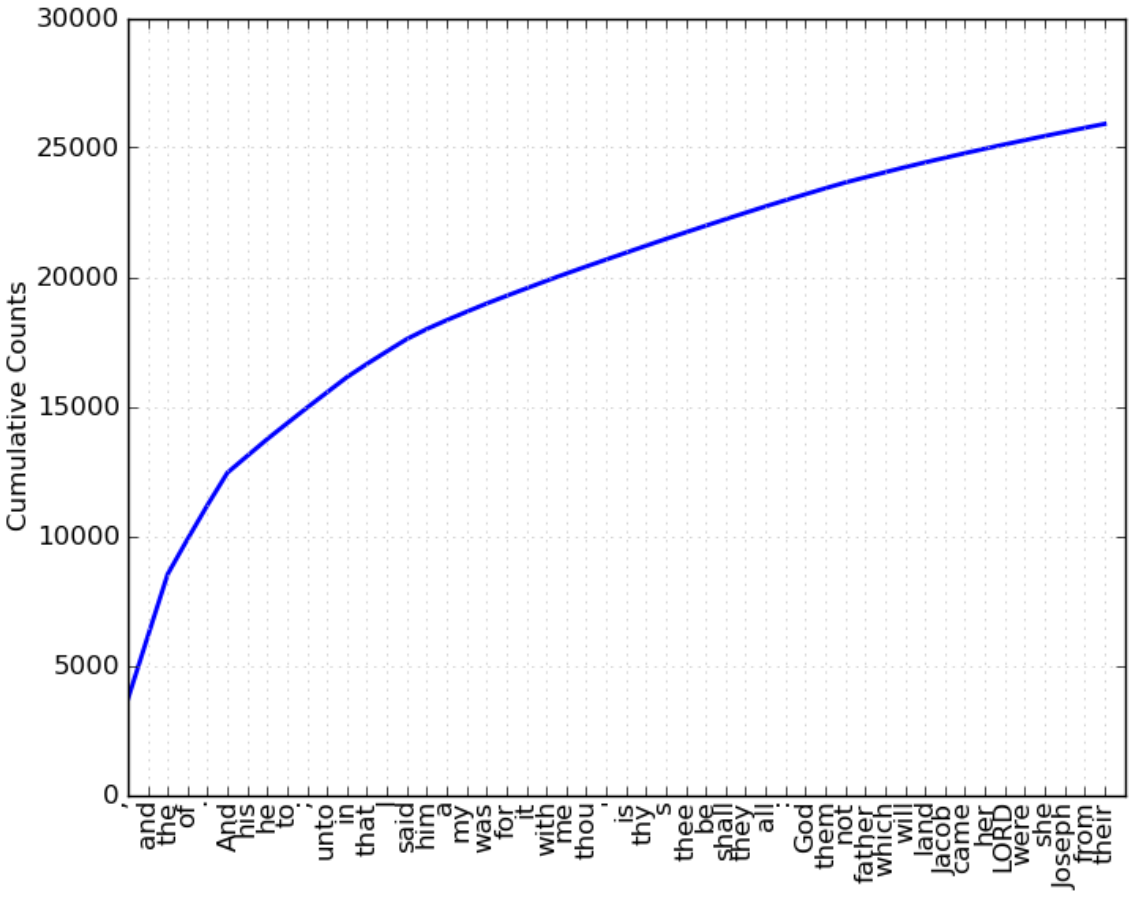
Ovaj primjer prikazuje prvih deset redaka konkordancije riječi "God" iz Knjige Postanka. Kolokacije su kombinacije riječi koje se često pojavljuju jedna uz drugu, specifične su za pojedini jezik te ne postoje opća pravila za njihovo stvaranje. Otkrivanje tih kombinacija je

također moguće pomoću NLTK funkcije collocations():

```
Building collocations list
said unto; pray thee; thou shalt; thou hast; thy seed; years old;
spake unto; thou art; LORD God; every living; God hath; begat sons;
seven years; shalt thou; little ones; living creature; creeping thing;
savoury meat; thirty years; every beast
```

Slika 3 – kolokacije Knjige Postanka

Pozivom funkcije collocations() nad tekstom – u ovom slučaju Knjizi Postanka – NLTK dijeli tekst na bigrame, uređene parove riječi te ispisuje bigrame riječi koji se često pojavljuju. Na manjem tekstu često će se preskočiti neke kolokacije te prikazati kombinacije riječi koje autor često koristi jedne uz drugu bez da kolociraju na jezičnoj razini, no takvih je pojava sve manje što je veći korpus.



Slika 4 – prikaz kumulativne čestotne razdiobe najčešćih riječi Knjige Postanka

Jedna od važnijih funkcija za analizu tekstova (i time korpusa) je mogućnost stvaranja čestotnih razdioba, lako postignuto korištenjem funkcije FreqDist(), a NLTK može čestotne razdiobe prikazati i grafički.

4. Prikupljanje podataka

Početak stvaranja korpusa je odabir tekstova koje će korpus sadržavati. Ovisno o tipu korpusa, podaci se mogu prikupljati "ručno", dodavajući tekstove u korpus jedan po jedan, ili automatizirano – koristeći program za prikupljanje podataka.

Jedan od načina automatskog prikupljanja podataka je korištenje web crawlera – programa koji preuzima podatke sa određenih web mjesta prema kodiranim uvjetima, bilo to sve .hr stranice ili tekstovi pojedinih foruma ili blogova.

Web crawling je veoma prisutan na internetu – budući da se stranice mijenjaju često, bez nekog vremenskog obrasca promjene, a isto tako i nastaju i nestaju nova web mjesta i stranice, tražilice nisu u mogućnosti držati statičan popis stranica na bilo koje poduže vrijeme; potrebno je često i temeljito provjeravati stranice za promjene kao i za pojavu novih stranica ili uklanjanje postojećih. Uključujući, među najpoznatijima, Google, Yahoo, i slične tražilice, stotine web crawlera konstantno preuzimaju podatke sa weba. (Pant, 2003)

Da bi se ograničio pristup web crawlerima, moguće je na web mjesto staviti datoteku robots.txt te u njoj definirati kojim crawlerima je pristup dozvoljen. Prije pregledavanja stranice – "crawlanja" – crawler traži datoteku robots.txt i u skladu sa tamo naznačenim uputstvima nastavlja rad ili prelazi na sljedeće web mjesto. Robots.txt je de facto standard, iako ne pripada ni jednoj organizaciji za postavljanje standarda. Jednostavnim unosom

```
User-agent: *
Disallow: /
```

zabranjuje se pristup svim crawlerima. Moguće je i dopustiti pristup određenim crawlerima, a ne ostalima, primjerice:

```
User-agent: Google
Disallow:
User-agent: *
Disallow: /
```

Takav robots.txt dopustit će pristup crawleru Google-a no nijednom drugom. Važno je napomenuti da robots.txt nije efektivna sigurnosna mjera; crawleri Googlea i slični poštivat će uputstva nađena u toj datoteci, no crawleri napisani bez pažnje na robots.txt, primjerice maliciozni softver, ignorirat će takva uputsva.

U pisanju crawlera, također je važno obratiti pozornost na frekvenciju pristupa poslužitelju; programu nije problem stvoriti popriličan broj veza u sekundi, no pojava takvog broja povezivanja najvjerojatnije će biti primjećen kao pokušaj napada te može rezultirati zabranom pristupa poslužitelju.

5. Stvaranje korpusa

Podaci prikupljeni web crawlerom – često direktno preuzete HTML datoteke – nisu prikladni za direktno uvrštavanje u korpus. Za pisanje web crawlera, NLTK nije potreban, no za uređivanje preuzetih podataka u oblik koristan za korpus, NLTK sadrži neke korisne alate. Funkcija clean_html() miče sve HTML oznake iz danog teksta – tekst u ovom slučaju je sadržaj HTML datoteke – da bi se dobio sadržaj stranice bez html koda.

Nakon čišćenja teksta na taj način, primjenjuje se funkcija word_tokenize(), koja dani tekst dijeli u popis riječi. U takvom obliku, tekst se može primijeniti za potrebe stvaranja korpusa.

Iako se korpus može sastaviti od skupova tekstova kao jednostavnih skupova riječi, poželjno je dodati podatke tim riječima. NLTK sadrži i "tagger" – "označivač" – algoritam koji riječima pridružuje njihovu vrstu riječi.

```
>>> import nltk
>>> sentence='All mimsy were the borogoves, And the mome raths outgrabe.'
>>> tokens=nltk.word_tokenize(sentence)
>>> tokens
['All', 'mimsy', 'were', 'the', 'borogoves', ',', 'And', 'the', 'mome', '
raths', 'outgrabe', '.']
>>> nltk.pos_tag(tokens)
[('All', 'DT'), ('mimsy', 'NN'), ('were', 'VBD'), ('the', 'DT'), ('borogo
ves', 'NNS'), (',', ','), ('And', 'CC'), ('the', 'DT'), ('mome', 'NN'), (
'raths', 'NNS'), ('outgrabe', 'VBP'), ('.', '.')]
```

Slika 5 – označivanje vrsta riječi

Funkcionalnost označivača vrsta riječi vidi se jasno na poznatoj rečenici iz *Alise u zemlji čudesa*. NLTK-om rečenicu najprije razdijelimo na riječi – primijetimo da su interpunkcijski

znakovi označeni kao riječi, no to je manje važno. Nakon toga NLTK stvara listu uređenih parova riječi i pripadajuće vrste riječi. Potrebu za daljnjim radom na korpusima i označivačima vidimo u pogrešno označenoj riječi "mimsy" – NN predstavlja opću imenicu, dok je riječ 'mimsy', iako nepostojeća, iz konteksta primjetno pridjev.

6. Klasifikacija teksta

Klasifikacija teksta je podjela teksta u određene kategorije prema zadanim uvjetima. U najjednostavnijim načinima klasifikacije, svaki tekst koji se razmatra gleda se kao neovisna jedinica, prema predodređenim parametrima. Primjeri klasifikacije su određivanje da li je mail spam, da li je tekst novinski članak te da li je novinski članak pisan o sportu, tehnologiji ili politici.

Klasifikacija se dijeli na nadziranu i nenadziranu. Klasifikacija je nadzirana ako je osnova klasifikacije primjerak za treniranje koji već imamo točno klasificiran. Nenadzirana klasifikacija nema pripremljeni uzorak točno klasificiranih tekstova te algoritam mora primijetiti značajke tekstova koje definiraju kategorije.

7. Korpus postova foruma Filozofskog Fakulteta

7.1 Prikupljanje podataka

Tekstovi pojedinih postova prikupljaju se pomoću programa za skidanje web stranica – već spomenutih web crawlera – koji skida cijele stranice te se naknadno iz njih izvlači tekst te njegovi meta-podaci. Za algoritam crawlera korisna je struktura foruma – phpBB. Budući da je zbog takve strukture moguće pristupiti postovima u nekoliko koraka, algoritam slijedi postupke koje bi slijedio korisnik za pristupanje pojedinim postovima. Budući da je forum podijeljen u pod-forume koji odgovaraju odsjecima Filozofskog Fakulteta, crawler počinje otvarajući te stranice – adrese kojih se razlikuju samo u jednoj brojci, koje se mogu lako očitati na početnoj stranici foruma. Ovaj dio se mora obaviti ručno – pogledati te brojke i zapisati ih za program, jer forum sadrži i pod-forume nevezane uz odsjeke, poput pod-foruma za brucoše ili KSFF.

Slika 6 – poveznice na stranice popisa tema foruma

Nakon otvaranja stranice nekog od pod-foruma, crawler stvara listu adresa svih stranica tog pod-foruma – svaka sadrži do pedeset tema. Kao i adrese samih pod-foruma, razlika u adresama je u jednoj brojci – na kojoj temi po redu počinje popis tema te stranice.

Potrebno je saznati najvišu brojku povezanu uz broj tema, jer postoji samo šest poveznica na stranice sa starijim temama – prve tri stranice i posljednje tri stranice. Dohvativši posljednju, adrese svih ostalih stranica se mogu lako generirati dodajući na osnovni URL brojeve od nule do posljednjeg broja, u koracima od pedeset.

U primjeru pod-foruma informacijskih znanosti, vidimo da postoji 29 stranica te će kao najveći broj u odgovarajućim URLovima crawler pronaći 1400. Otkrivši 1400 kao posljednju stranicu, crawler generira URLove svih postojećih istovrsnih stranica, dodavši svaki broj od 0 do 29, pomnožen sa 50, na osnovni URL stranice pod-foruma

Teme su slično podijeljene; na prvoj stranici dane teme nalaze se poveznice na prve tri stranice i posljednje tri stranice postova, no za razliku od popisa tema, postova je petnaest na svakoj stranici. Crawler, sad sa svim potrebnim podacima, skida stranicu po stranicu postova. Kod pisanja crawlera, važno je i paziti na ograničenje od jednog otvaranja stranice u sekundi, da bi se izbjegla moguća zabrana pristupa forumu. Preporučljivo je ostaviti html u originalnom obliku, da uvijek ostane mogućnost provjere točnosti algoritama ili vađenja dodatnih meta-podataka.

Kod preuzimanja postova, korisno je provjeriti da li je tema već preuzeta – takozvane "sticky" teme pojavit će se na svakoj stranici te će bez provjere biti preuzete za svaku stranicu popisa tema.

7.2 Stvaranje korpusa

Nakon što je cijeli forum preuzet, potrebno je podatke urediti na način prikladniji za korištenje; XML pruža lako razumljiv format, a prednost mu je da je obični tekst te ga lako čita bilo koji program. Radi olakšavanja pristupa korpusu i njegovog prijenosa, svi se postovi zapisuju u jednu datoteku. Program napisan za tu svrhu prolazi kroz sve html datoteke koje je spremio crawler te iz njih vadi postove. Dodatno formatiranje je potrebno jer su postovi zapisani u html-u. Potrebno je ukloniti html oznake, no i sačuvati smiley-e jer su i oni potencijalni subjekt analize korpusa. Svaki post je jedan XML zapis, jedan redak u

datoteci. Za svaki post dostupno je ime autora, URL posta, redni broj posta u temi, pod-forum kojem post pripada, datum i vrijeme stvaranja posta te naslov teme kojoj post pripada.

```
'<post url="http://forum.ffzg.hr/viewtopic.php?t=10045&start=0" postnumber="1" subforum="Inf" au
thor="oi!" title="ra&amp;#269;unaraska lingvistika" timestamp="02 ruj 2004 23:46">\n<p>dobis nek
o pitanje tipa sta je granaljka. i onda pises sastavak.</p><p>a onda te pit kao usmeno neko pita
nje slicno tome.</p><p>smjesno je jako</p>\n</post>'
```

Slika 7 – redak XML datoteke

U takvom formatu, poprilično je lako dohvatiti postove i njihove meta-podatke da bi se kreirao korpus. No da bi bili koristan resurs, potrebno je meta-podatke spremiti u novu strukturu podataka.

7.3 Post klasa

```
class Post(str):
    "Post class. Attributes: author, number, subforum, title, text."
    def __init__(self, sPostTag=None):
        if (sPostTag!=None):
            try:
                self.id=re.findall(r'\?t=(.+?)&', sPostTag, re.UNICODE)[0]
                self.author=re.findall(r'author="(.+?)"',sPostTag, re.UNICODE)[0
                self.number=re.findall(r'postnumber="(.+?)"',sPostTag, re.UNICOD
                self.subforum=re.findall(r'subforum="(.+?)"',sPostTag, re.UNICOD
                self.title=re.findall(r'title="(.+?)"',sPostTag, re.UNICODE)[0]
                dt=re.findall(r'timestamp="(.+?)"',sPostTag, re.UNICODE)[0]
                dt=dt.split(' ')
                dt[1]=dMonths.get(dt[1],'3')
                dt.append(dt[-1].split(':')[1])
                dt[-2]=dt[-2].split(':')[0]
                self.timestamp=datetime.datetime(int(dt[2]),int(dt[1]),int(dt[0]
                lLines=re.findall(r'<p>(.+?)</p>',sPostTag, re.UNICODE)
                sLines=''
                for i in range(len(lLines)):
                    if i is 0:
                        sLines=sLines+lLines[i]
                    else:
                        sLines=sLines+'\n'+lLines[i]
                lSmiles=re.findall(r'<smiley type=".+?.gif"/ >',sLines)
                for i in lSmiles:
                    sLines=sLines.replace(i,i[14:-8])
                self.text=sLines+' '
            except:
                print 'Invalid post format.'
    def sents(self):
        return nltk.sent_tokenize(self.text)
    def words(self):
        wordsies=[]
        sents=nltk.sent_tokenize(self.text)
        for i in sents:
            for j in nltk.word_tokenize(i):
                wordsies.append(j)
        return wordsies
```

Slika 8 – definicija klase post

Postavivši klasu na ovaj način, svaki post iz pripremljene XML datoteke – svaki redak - prosljeđuje se toj klasi, što omogućava lak pristup meta-podacima posta. Na primjeru posta iz pod-foruma Informacijskih znanosti:

```
>>> print ('Autor:\t'+pInf.author+'\nNaslov teme:\t'+pInf.title+'\nVrijeme posta:\t'+str(pInf.timestam
p)+'\nTekst:\t'+pInf.text)
Autor:  oi!
Naslov teme:    ra&amp;#269;unaraska lingvistika
Vrijeme posta:  2004-09-02 23:46:00
Tekst:  dobis neko pitanje tipa sta je granaljka. i onda pises sastavak.
a onda te pit kao usmeno neko pitanje slicno tome.
```

Slika 9 – primjena klase post

7.4 Klasifikacija postova

Naivni Bayesov klasifikator radi na principu Bayesovog teorema i nezavisnosti svojstava – pretpostavlja da za dana dva svojstva, prisutnost ili odsutnost prvog nije povezana sa prisutnosti ili odsutnosti drugog. Jednostavna implementacija naivnog Bayesovog klasifikatora uključena je u NLTK te se sa nešto pripreme može direktno iskoristiti u Python programu. Klasifikator mora najprije "naučiti svojstva" na određenom skupu podataka – ovaj tip strojnog učenja naziva se nadzirano učenje, jer je za algoritam potrebno pripremiti podatke kojima je klasifikacija već poznata. To nije problem u ovom slučaju jer svaki post već ima poznatu pripadnost pod-forumu Da bi postupak bio što jasniji, biti će prikazan i dio izvornog koda konfiguracije klasifikatora. (Bird, 2009)

```
def randomSplit(lListToSplit, iSplitPercentage=90):
    split=int(len(lListToSplit)*iSplitPercentage/100.0)
    random.shuffle(lListToSplit)
    lTrain=lListToSplit[:split]
    lTest=lListToSplit[split:]
    random.shuffle(lTrain)
    random.shuffle(lTest)
    return lTrain, lTest
```

Slika 10 – funkcija za podjelu postova

Jedan od ključnih dijelova pripreme za stvaranje i testiranje klasifikatora je podjela podataka na dio za učenje i dio za testiranje. Kao što se može vidjeti u argumentima funkcije, podjela je 90-10 u korist podataka za učenje. Zbog algoritma stvaranja datoteke korpusa, postovi su poredani prema odsjecima te bi podjelom 90-10 svi postovi većine odsjeka završili u dijelu za trening - ne bi bili testirani uopće a klasifikator bi bio puno

manje učinkovit. Ova funkcija to rješava primjenjujući metodu shuffle – stvarajući nasumični poredak postova.

Sljedeći korak je definicija svojstava po kojima će klasifikator pokušati odrediti pripadnost danog posta odsjeku.

```
def features(post):
    features={}
    try:
        asdw=post.words()
        asds=post.sents()
        asdt=post.text
        features['First Word']=asdw[0]
        features['Number Of words']=len(asdw)
        features['Number Of Sentences']=len(asds)
        features['English Letter Count']=asdt.count('x')+asdt.count('y')+asdt.count('w')+asdt.count('q')

        featurewords=pickle.load(open('features.dct','r'))
        for i in featurewords.keys():
            features['Count of '+i]=asdt.count(featurewords[i])


    except:
        pass
    return features
```

Slika 11 – funkcija za pronalaženje karakterističnih svojstava postova

Prikazana funkcija definira ta svojstva. Ta su svojstva broj riječi u postu, prva riječ posta, broj rečenica u postu i broj slova svojstvenih engleskoj abecedi za razliku od hrvatske.
Na slici 12 vidi se rezultat funkcije features primijenjene na nasumično odabranom postu.

```
>>> print features(pInf)
{'First Word': 'dobis', 'English Letter Count': 0, 'Number Of words': 27, '
Number Of Sentences': 4}
```

Slika 12 – rezultat funkcije features

Nadalje, kao dodatnih stotinu karakteristika uzima se broj ključnih riječi u postu – riječi dobivenih stvaranjem čestotne razdiobe riječi na cjelokupnom forumu te uzimanjem trećih stotinu najčešćih da bi se izbjegle gramatičke riječi.

```
fTest=([(post, post.subforum) for post in lTest])
fTrain=([(post, post.subforum) for post in lTrain])

fitjures=[(features(i),j) for (i,j) in fTrain]
```

Slika 13 - stvaranje skupova uređenih parova za klasifikator

Za treniranje i testiranje klasifikatora, stvaraju se uređeni parovi – skup postova i vrijednost pripadajućeg svojstva. Za skup postova za trening stvara se i dodatna varijabla koja sadržava uređene parove. Prvi element tih parova je popis svojstava posta, a drugi vrijednost traženog svojstva posta.

```
classifier=nltk.NaiveBayesClassifier.train(fitjures)
```

Slika 14 – inicijalizacija klasifikatora

Djelićem koda sa slike 14 stvara se klasifikator i pokreće njegovo učenje.

8. Zaključak

Korpusi se već poduže vrijeme koriste kao jezični resursi, no dostupnost korpusa na hrvatskom jeziku je jedva postojeća. Radom na stvaranju i označavanju korpusa znatno bi se moglo olakšati istraživački rad u područjima leksikografije, semantike te sociolingvistike na hrvatskim govornim područjima.

Što se tiče klasifikatora postova, rezultati nisu impresivni – točnost klasifikatora, dobivena provjerom nad preostalih 10% postova, je 18.28%, a najinformativnija karakteristika posta je početna riječ. Iz toga se može zaključiti da je postove teško diferencirati - to jest, postovi pojedinih pod-foruma ne razlikuju se previše po korištenom vokabularu – cijeli forum se može gledati kao jedinstvena kategorija. Iz toga se može izvući zaključak da tekstovi studenata Filozofskog Fakulteta u Zagrebu imaju više zajedničkih točaka nego onih koje bi ih diferencirale ovisno o odsjeku.

Bibliografija

Bird S., Klein E., Loper E., Natural Language Processing with Python, Prvo izdanje. Sebastopol: O'Reilly Media, Inc., 2009

Gautam Pant, Padmini Srinivasan, Filippo Menczer, Crawling the Web, 11. 06. 2003, http://dollar.biz.uiowa.edu/~pant/Papers/crawling.pdf, 13. 08. 2011.

McEnery T., Wilson A., Corpus linguistics: an introduction. Drugo izdanje. Edinburgh: Edinburgh University Press, 2003.