Tehničko veleučilište u Zagrebu

Završni rad 999

Računalna obrada prirodnih jezika

Fran Raknić

Zagreb, svibnja 2016.

Contents

[Uvod 4](#_Toc451901994)

[Prikupljanje podataka 5](#_Toc451901995)

[Natural language toolkit – Python biblioteka 6](#_Toc451901996)

[Opojavničenje / Tokenizing 6](#_Toc451901997)

[NLTK Tokenizer modul 6](#_Toc451901998)

[TweetTokenizer 7](#_Toc451901999)

[Lematizacija / Stemming 9](#_Toc451902000)

[Popis literature 10](#_Toc451902001)

# Uvod

# Prikupljanje podataka

# Natural language toolkit – Python biblioteka

Natural Language Toolkit skraćeno NLTK je platforma namjenjena obradi prirodnih jezika za programski jezik Python. Sadrži biblioteke za procesiranje teksta, kao što su: klasifikacija, tokenization, stemming, tagging, parsing itd. Također sadrži preko pedeset korpusa i trening modela za strojno učenje. Biblioteka je dostupna na Windows, Mac OS X te Linux/UNIX operacijskim sustavima te je licencirana pod Apache 2.0 licencom što znači da je kod otvoren te korištenje besplatno. Posljednja stabilna verzija je 3.2.1 distribuirana 9. travanja 2016. NLTK zahtjeva Python verzije 2.7 ili 3.2, neke funkcionalnosti za strojno učenje dodane u 3.0 verziji zahtjevaju Python 3.2 te neće raditi na 2.7 verziji.

## Opojavničenje / Tokenizing

Opojavničenje je proces određivanja lingvističkih jedinica (rečenice, riječi, znakovi...). Opojavničenje ili tokenizacija stavlja korpus u stanje u kojem su sve riječi-pojavnice identificirane i eksplicitno obilježene. Najjednostavniji i prilično učinkovit pristup je razdvajanje pojavnica znakom za obilježavanje razmaka tj. bijelinama. Automatizirati ovakav proces je prilično jednostavno, većina programskih jezika sadrži standardne biblioteke za string strukturu podataka koje sadrže funkcije za osnovnu obradu teksta (split, replace, upper, lower). Međutim, opojavničenje/tokenizacija u složenijim pristupima, primjerice pojavnice se mogu sastojati od dvije ili više riječi. Na primjer, datum *15. svibnja* ili *15. 5.* bi mogao biti obrađivan kao jedna pojavnica pa ranije spomenuti jednostavni pristup nebi mogao tokenizirati. Tekst koji se obrađuje može se opojavničiti više puta da bi se dobili željeni rezultati. Nakon dobivanja tokena koji su riječi ponekad je potrebno spojiti više riječi koje, gotovo uvijek, imaju isto značenje, primjerice „hard drive“. Nakon tokeniziranja manjih lingvističkih jedinica kao što su riječi ili datumi potrebno je označiti i rečenice. Postupci segmentacije na rečenice su složeni iz razloga što su oznake rečenične interpunkcije često višeznačne. Na primjer, točka može stajati uz redni broj, kraticu ili na kraju rečenice. Podaci također mogu dolaziti označeni nekim od jezika za označavanje (eng. *markup languages*) primjerice u HTML-u ili XML-u. Prije opojavničenja teksta prirodnog jezika potrebno je odvojiti jezik označavanja od teksta koji želimo obrađivati. Budući da su jezici označavanja (formalni, umjetni, kontrolirani?)uvelike standardizirani taj proces je olakšan radi mogućnosti korištenja gotovih riješenja koja uklanjaju elemente kao što su tagovi. U slučajevima kada su podaci označeni za aplikacije i procesore treće strane (eng. *third-party*) potrebno je proučiti logiku označavanja te izraditi vlastito riješenje koje uklanja ili odvaja tekst od jezika označavanja. Posebno teško je opojavničenje jezika u scriptio continua pismu gdje se riječi ne razdvajaju razmacima ili znakovima (Kineski, Thai, staro-Grčki).

Izazovi opojavničenja teksta komunikacije na internetu, posebice društvenim mrežama, uvelike ovise o ciljevima obrade i platformi na kojoj se komunikacija odvija. Svaka društvena mreža ili internetski servisi kao chat sobe imaju neke posebne znakove ili standarde. Primjerice „Tweeter“ koristi znak # (eng. *hashtag*) nakon kojeg, bez razmaka, slijedi riječ koja označava širu temu ili područje objave. Ovisno o ciljevima, aplikacija za obradu može tretirati takve tokene kao riječi, posebnu kategoriju ili odbacivati.

### NLTK Tokenizer modul

NLTK platforma sadrži Tokenizer paket. Bazna klasa svih tokenizera/opojavnjivača je „tokenize“, instanciranjem svojih objekata ili korištenjem postojećih modula sve klase proširuju „tokenize“. Također nasljeđuju tokenize metodu koja će biti opisana za svaki pod-modul posebno.

### TweetTokenizer

Prilagođen opojavničenju Tweet objava, fleksibilan i lako podesiv. Tuple regex\_strings definira listu regularnih izraza. Regularni izrazi su, po redu, kompajlirani u jedan regexp objekt naziva word\_re. Metoda tokenize prima dva dodatna argumenta, preserve\_case sa zadanom vrijednosti True, te reduce\_len sa zadanom vrijednosti False. U slučaju da korisnik prvi parametar postavi na False, metoda će sve tokene postaviti na mala slova dok će drugi parametar, ukoliko je True smanjiti sva uzastopna ponavljanja znaka na tri ponavljanja. Na primjer *jaaaaako* pretvara u *jaaako.*

EMOTICONS = r"""

(?:

[<>]?

[:;=8] # eyes

[\-o\\*\']? # optional nose

[\)\]\(\[dDpP/\:\}\{@\|\\] # mouth

|

[\)\]\(\[dDpP/\:\}\{@\|\\] # mouth

[\-o\\*\']? # optional nose

[:;=8] # eyes

[<>]?

|

<3 # heart

)"""

From nltk.tokenize import TweetTokenizer  
  
Tobjekt = TweetTokenizer()

niz = „Ovo je niz emotikona i :D :-P D <3“

Tobjekt.tokenize(niz)

Kod tipa teksta kao što su neformalne objave na društvenim mrežama, taj proces se komplicira. Primjerice neki korisnici koriste hrvatske dijakritike, neki ih pišu u obliku ch za ć ili dj za đ. Komunikacija preko društvenih mreža sadržava poveznice na druge stranice koje mogu dolaziti kroz API te kao niz znakova. Pokazati neke od implementacija (NLTK).

NLTK biblioteka nudi više modula za opojavničenje. U načelu svaki modul razdvaja niz znakova u određene lingvističke jedinice. Osnovni zadatak je razdvojiti tekst na riječi, tj. razdvojiti interpunkciju i posebne znakove od jedinica koje imaju značenje. Modul TweetTokenizer sadrži regularne izraze koji prepoznaju tipove kao što su URL, emotikoni te HTML.

## Lematizacija / Stemming

Lematizacija (stemming) je dugotrajnij i teži postupak od opojavničenja. Lematizacija je proces reduciranja flektivnih riječi da bi se dobio korijenski oblik, korijen ne mora nužno biti isti kao morfološki. Za dobivanje oblika riječi hrvatskog jezika moguće je koristiti Hrvatski Morfološki Leksikon[3], za precizniju lematizaciju podataka dobivenih s društvenih mreža ili bilo kojeg izvora neformalne komunikacije među korisnicima bilo bi potrebno napisati vlastita pravila ili izraditi leksikon za specifično područje. Primjeri implementacije...

## Popis literature