SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Filip Novački

TextBlob

PROJEKT

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Filip Novački

Matični broj: 44531/15-R

Studij: Informatika u obrazovanju

TextBlob

Projekt

Mentor:

Prof. dr. sc. Miroslav Bača

Varaždin, lipanj 2020.

Filip Novački

Izjava o izvornosti

Izjavljujem da je moj projekt izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

`Textblob` je biblioteka koja služi za rad u području neurolingvističkog programiranja na visokoj razini. Biblioteka je napravljena nad bibliotekom `NLTK` i `pattern` na način da je jednostavno početi s osnovnim primjerima i s konceptima NLP-a, a opet da se korisnik osigura od početnih pogrešaka koje se mogu dogoditi kad se radi u nekoj složenijoj biblioteci kao što su `NLTK`, `Stanford NLP` i slični. Rad opisuje osnovne koncepte NLP-a kako bi se mogao razumjeti `TextBlob` te su ti koncepti demonstrirani na primjeru u `TextBlobu`. Osim NLP-a i `TextBloba` u radu je prikazan i okvirni izgled jednog NLP projekta kako bi se primjena te biblioteke mogla staviti u kontekst *stvarnog* svijeta. Na kraju su predstavljeni i zaključci koji se mogu zaključiti s temeljima u rezultatima NLP-a.

Ključne riječi: neurolingvističko programiranje, NLP, TextBlob, NLTK, jezik, lingvistika, python, govor, riječi

Sadržaj

1. Uvod

Cilj ovog projekta je upoznati se s alatom `TextBlob` kojeg razvija Steven Loria. `TextBlob` je Python biblioteka koji služi za neurolingvističko programiranje. Napisana je za Python 2 verzije $>= 2.7$ te za Python 3 verzije $>= 3.5$. Cilj `TextBloba` pružiti je jednostavan pristup neurolingivstičkom programiranju tako da se zaobiđu detalji koji postoje u biblioteci `NLTK`.

Neurolingvističko programiranje (kratko NLP) interdisciplinarno je područje koje stoji između računalnih znanosti, lingvistike i umjetne inteligencije. Cilj NLP-a je približiti jezik računalu te mu omogućiti razumijevanje i analizu jezika. Tako NLP danas ima veliku ulogu u mnogim područjima - npr. u marketingu se prate objave, komentari, poruke itd. kako bi se vidjelo koliko su kupci zadovoljni određenim proizvodima i zašto jesu ili zašto nisu. NLP također služi i za *chat botove* gdje oni imaju prvi kontakt s čovjekom te poruke, ovisno o njihovom sadržaju, preusmjeri osobama koji su zaduženi za područje koje je korisnik upitao.

Jedan od najopsežnijih alata za NLP je `NLTK` (***n**euro**l**inguistic **t**ool**k**it*). On pruža analizu jezika na niskoj razini. To znači da korištenjem `NLTK-a` korisnik ima potpunu kontrolu nad svojim radom, no istovremeno je toliko ogromno da je potrebno izrazito mnogo znanja za njegovo korištenje, a polustručno korištenje može dovesti do mnogo grešaka u analizi. Stoga `TextBlob` pruža dovoljnu razinu apstrakcije da se korisnik može bezbolno uvesti u svijet NLP-a.

S obzirom na to da su `TextBlob` i `NLTK` biblioteke pisane u Pythonu, u ovom će radu biti poseban naglasak stavljen na biblioteke pisane za Python.

U radu se neće opisivati instalacija biblioteke i potrebnih uvjeta jer je proces izrazito jednostavan, a u detalje je opisan na `GitHub` stranicama kao i na stranicama dokumentacije, a svi su oni popisani na zadnjoj stranici u literaturi.

2. Izgled NLP projekata

Projekti u području neurolingivstike uglavnom sadrže nekoliko koraka:

- odabir teme projekta
- sakupljanje podataka
- obrada (prilagodba) podataka
- analiza podataka
- zaključci

Ovaj projekt neće u potpunosti obraditi jedan NLP projekt jer je to puno šira tema od primjene `TextBloba`. Međutim, za razumijevanje biblioteke potreban je i kontekst u kojem se koristi `TextBlob`.

2.1. Odabir teme projekta

Kod odabira teme projekta moramo prvo spoznati koji je cilj onoga što mi želimo. Dakle, mi možemo biti dio marketinga nekog poduzeća i želimo saznati kako ljudi reagiraju na novonastale promjene. Skupljat ćemo podatke koji imaju neko zajedničko obilježje, npr. `#coronaBeer` ako želimo promatrati kako ljudi u ovoj godini reagiraju na pivo. Tako možemo analizirati *tweetove*, komentare po raznim platformama itd.

Tema može biti i analiza odluka koje se donose u politici i stavovi naroda. Tako bismo mogli analizirati kakvi su sentimenti u *tweetovima* koji su odgovor na Trumpove *tweetove* itd.

Ograničenja što se domene tiče gotovo pa i nema, samo je važno da postoji dovoljna količina podataka nad kojima se može vršiti analiza. Važno je i uzeti u obzir koje sve jezike podržavaju biblioteke, odnosno postoje li neke baze s jezicima koje se mogu uvesti u biblioteku za NLP. Engleski je poprilično zastupljen jezik, no hrvatski baš i nije. Stoga želimo li napraviti projekt vezan za hrvatsko govorno područje, trebali bismo napraviti i neke predradnje kako bi se napravila baza za hrvatski.

2.2. Sakupljanje podataka

Podatci se mogu prikupiti na razne načine. Ukoliko želimo saznati kako ljudi reagiraju na proizvode, vjerojatno će najprikladnije biti napraviti *web scraping*, odnosno izvući podatke iz HTML dokumenata te ih potom obraditi. Jedna od opcija je i sakupljati *feedback* korisnika usluga. A ono što koriste *chat botovi* je analiza poruka koje se šalju botu.

Za sakupljanje vrijedi isto što i za odabir teme - granice realno ne postoje, na nama je da budemo dovoljno snalažljivi i pronađemo podatke koji nam trebaju.

Za *web scraping* postoje biblioteke u raznim programske jezike. U Pythonu je aktualan `BeautifulSoup` u kombinaciji s `Requests` koji služe preuzimanje stranica te za njihovo parsiranje.

U svrhu ovog projekta primjeri će se pokazati na podatcima koji su već prije sakupljeni, a radi se o objavama Donalda Trumpa na *twitteru*.

Za organizaciju podataka je dobro koristiti neku biblioteku za agregaciju podataka. Ovdje će se koristiti `pandas`.

[1]:
```
import pandas as pd
import re
```

[2]:
```
df = pd.read_csv('../data/realdonaldtrump.csv', index_col='date')
```

[3]:
```
data = pd.DataFrame(df.content)
```

[4]:
```
data
```

[4]:
```
2009-05-04 Be sure to tune in and watch Donald Trump on L...
2009-05-04 Donald Trump will be appearing on The View tom...
2009-05-08 Donald Trump reads Top Ten Financial Tips on L...
2009-05-08 New Blog Post: Celebrity Apprentice Finale and...
2009-05-12 "My persona will never be that of a wallflower...
...                                                      ...
2020-04-15 Our GREAT Senator from South Carolina, @ Senat...
2020-04-15 # ThanksForDelivering @ UPS!https://twitter.co...
2020-04-15 My condolences to the Steinbrenner family, and...
2020-04-15 We are having very productive calls with the l...
2020-04-15 White House news conference today at 5:30 P.M...
```

2.3. Odrada (prilagodba) podataka

Obrada odnosno prilagodba podataka je proces u kojem se podatci prilagođavaju tako da budu čitljivi algoritmima. Kroz niz koraka na nama je da izbacimo sve ono što algoritam ne može protumačiti kako bismo dobili što točnije rezultate. Taj se proces uglavnom prilagođava izvoru podataka, a alati su uglavnom opće namjene, kao što su regularni izrazi, *find and replace* alati, alati za manipulaciju stringovima itd.

Glavni cilj je da podatci za NLP budu isključivo stringovi malim slovima, bez brojeva, poveznica itd. Također se trebaju izbaciti kratke riječi koje ne nose značenje, tzv. *stop words* kao što su *any*, *with*, *and*, *or* itd. Stoga se to čišćenje podataka radi po potrebi s obzirom na izvor kakav imamo.

[59]:
```
def cleaning(x):
    # mala slova:
```

```
    x = x.lower()

    # URL-ovi
    x = re.sub(r'\w+:\/{2}[\d\w-]+(\.[\d\w-]+)*(?:(?:\/[^\s/
↪]*))*', '', str(x))

    # specijalni znakovi
    x = re.sub(r'[\W]', ' ', str(x))

    # brojevi
    x = re.sub(r'\w*\d\w*', '', str(x))

    # razmaci
    x = re.sub(r'\s+', ' ', str(x))
    return x
```

[60]:
```
data = data.content.apply(cleaning)
```

[63]:
```
data = pd.DataFrame(data)
data
```

Nakon čišćenja izgleda ovako:

[63]:
```
date
2009-05-04  be sure to tune in and watch donald trump on l...
2009-05-04  donald trump will be appearing on the view tom...
2009-05-08  donald trump reads top ten financial tips on l...
2009-05-08  new blog post celebrity apprentice finale and ...
2009-05-12   my persona will never be that of a wallflower...
...                                                       ...
2020-04-15  our great senator from south carolina senatort...
2020-04-15                          thanksfordelivering ups
2020-04-15  my condolences to the steinbrenner family and ...
2020-04-15  we are having very productive calls with the l...
2020-04-15  white house news conference today at p m easte...
```

U ovom primjeru je potrebno još i izbaciti kratke riječi, jednoslovne riječi (ostatak od izraza *PM* koji se vidi u zadnjem *tweetu*), no za pokaz primjera je sasvim dovoljno. Čišćenje se može i jako detaljno napraviti, no to često nije potrebno jer `TextBlob` i drugi NLP alati neće javljati greške kad neće znati nešto analizirati, nego će samo preskočiti te riječi. Naravno, uz to postoji rizik od krivog rezultata, no ako su podatci dovoljno dobro pročišćeni, i rezultati će biti dovoljno dobri.

2.4. Analiza podataka

Analiza podataka je proces koji se zapravo najviše tiče samog NLP-a. Ovdje se istražuju sve specifičnosti jezika, rečenica, izražavanja itd. koje mogu biti od značaja za projekt. Neke od tehnika koje se primjenjuju su:

- *stemming* - uzimanje korijena riječi kako bi se neutralizirali različiti oblici govora (rezultat nema semantičku vrijednost)
- *lemmatization* - slično kao i *stemming*, samo što rezultat ima semantičku vrijednost i osnovni je oblik neke riječi
- *POS (Point of Speech)* - pridruživanje vrsta riječi i sličnih oznaka riječima u rečenici (imaju i šire oznake kako bi oznake bile primjenjivije računalu)
- *named entity recognition* - klasificiranje riječi prema njenoj semantici u nekoj rečenici, npr. prepoznavanje koji dio teksta je ime organizacije, uloga u organizaciji, ime osobe itd.
- TF-IDF (*term frequency multiplied by ducument frequency*) - računanje učestalosti pojavljivanja neke riječi u tekstu, često korisno za analizu dokumenata i većih tekstova te označavanja njihovih tema
- *N-Grams* - grupiranje riječi u skupine veličine N (*2-grams*, *3-grams*...), izrazito korisno za automatsko nadopunjavanje teksta u često korištenim frazama kao što su *thank you*, *I am* itd.

Ove teme će se pokazati detaljnije na primjeru u `TextBlobu` tako da se ovdje neće detaljnije razrađivati.

3. Odabrana poglavlja NLP-a i primjeri u `TextBlobu`

Sad kad smo odgonetnuli otprilike kontekst cijele priče u kojoj se koristi `TextBlob` vrijeme je da se pokaže kako to izgleda u samoj biblioteci.

Upute za instalaciju se ovdje neće obrađivati, one su obrađene u dokumentaciji i u repozitoriju s bibliotekom i nema nečega što je važno spomenuti.

Za početak potrebno je uvesti biblioteku. Najviše ćemo koristiti klasu `TextBlob`. Uz to pripremamo i string koji ćemo koristiti za isprobavanje funkcionalnosti.

[3]:
```
from textblob import TextBlob
```

[2]:
```
string = "Negation combines with modifiers in an interesting way:␣
↪in addition to multiplying by -0.5 for the polarity, the inverse␣
↪intensity of the modifier enters for both polarity and␣
↪subjectivity."
```

[4]:
```
tb = TextBlob(string)
```

3.1. Osnovni dijelovi rečenice

Prije početka rada u području NLP-a moramo prvo odrediti dijelove rečenice. Naravno, to su riječi i rečenice. To su vrlo jednostavne operacije jer su riječi zapravo odvojene razmakom, a rečenice točkom. Kod odvajanja riječi nestaju specijalni znakovi, što uključuje točke i zareze.

[9]:
```
tb.words
```

[9]:
```
WordList(['Negation', 'combines', 'with', 'modifiers', 'in', 'an',␣
↪'interesting', 'way', 'in', 'addition', 'to', 'multiplying',␣
↪'by', '0.5', 'for', 'the', 'polarity', 'the', 'inverse',␣
↪'intensity', 'of', 'the', 'modifier', 'enters', 'for', 'both',␣
↪'polarity', 'and', 'subjectivity'])
```

Popis riječi koristan je jer je svaka riječ tipa `Word`, što je jako korisno u kasnijim fazama analize zbog operacije nad pojedinačnim riječima, kao što su transformacije iz jednine u množinu i obrnuto, traženje korijena riječi itd.

Također, riječi u `TextBlobu` povezane su i s rječnicima pa je moguće svojstvom `definition` dobiti sve definicije odabrane riječi.

[10]:
```
tb.sentences
```

[10]:
```
[Sentence("Negation combines with modifiers in an interesting way:␣
↪in addition to multiplying by -0.5 for the polarity, the inverse␣
↪intensity of the modifier enters for both polarity and␣
↪subjectivity.")]
```

Rečenice u `TextBlobu` su tipa `Sentence`, a korisne su jer svaka od njih se može odvojeno analizirati. Rečenice mogu imati svoje sentimente, može se procijeniti jesu li rečenice pozitivne ili negativne temeljeno naučenim itd.

3.2. Oznake (*point of speech, tags*)

Sad kad smo pripremili string u obliku klase `TextBlob`, spremni smo za daljnju analizu. Za početka možemo krenuti s oznakama (*tags*). Oznake služe za određivanje službe riječi u rečenici (en. *POS*). korištenjem svojstva `tags` pristupamo popisu oznaka riječi.

[6]:
```
tb.tags
```

[6]:
```
[('Negation', 'NNP'),
 ('combines', 'NNS'),
 ('with', 'IN'),
 ('modifiers', 'NNS'),
 ('in', 'IN'),
 ('an', 'DT'),
 ...
 ('enters', 'NNS'),
 ('for', 'IN'),
 ('both', 'DT'),
 ('polarity', 'NN'),
 ('and', 'CC'),
 ('subjectivity', 'NN')]
```

Popis oznaka je relativno velik i nema smisla stavljati ga u ovaj rad. Međutim, za razumijevanje bit će objašnjene oznake koje se javljaju u ovom primjeru:

- NN - imenica (NNS je imenica u množini, NNP je vlastita imenica, a NNPS vlastita imenica u množini)
- DT - član (*a*, *an*...)
- TO - riječ *to* u funkciju mijenjanja padeža iduće riječi (na hrvatskom kao riječ *prema*, *k*...)
- CC - veznik
- IN - prijedlog
- JJ - pridjev (JJS je pridjev u superlativu, JJR je pridjev u komparativu)

Temeljem oznaka moguće je stvoriti jezično stablo neke rečenice te se pomoću tog stabla jasno može vidjeti koje se riječi i dijelovi rečenica međusobno vežu temeljeno na znanju oznaka - redoslijed riječi u rečenici uvijek je uvjetovan upravo tim oznakama.

3.3. Skupine riječi

`TextBlob` koristan je jer zna dobro grupirati riječi koje zajedno čine neki smisao. U NLP-u se razlikuje više načina grupiranja riječi.

3.3.1. `n-grams`

Tehnika `n-gram` je grupiranje riječi koje stoje jedne pored drugih. Naziv implicira varijabilnost slova n – ta varijabla može biti bilo koji broj. `n-gramove` dobivamo korištenjem metode `ngrams(n)`, gdje je n broj riječi koje želimo spojiti. U ovom primjeru skraćen je izlaz na pet `n-grama` zbog duljine popisa.

```
[46]: tb.ngrams(3)[:5]
```

```
[46]: [WordList(['Negation', 'combines', 'with']),
       WordList(['combines', 'with', 'modifiers']),
       WordList(['with', 'modifiers', 'in']),
       WordList(['modifiers', 'in', 'an']),
       WordList(['in', 'an', 'interesting'])]
```

Strojnim učenjem ovih popisa računalo može zaključiti koje se riječi često pojavljuju zajedno i koja riječ često slijedi nakon određene riječi. To značajno olakšava pisanje na uređajima koji nemaju fizičke tipkovnice, npr. mobilni uređaji. Stoga nakon pisanja riječi *thank* logično je predložiti nastavak *you*.

3.3.2. Imenične skupine (*noun phrases*)

Imenične skupine su skupine riječi koje se vežu za imenice. One su određene isključivo POS oznakama prema kojima `TextBlob` zna koje grupacije čine subjektnu skupine. U hrvatskom jeziku je to predstavljeno kao imenica (subjekt) uz koju se nalaze objekti i/ili atributi. Objekti i atributi pobliže označavaju imenicu.

```
[7]: tb.noun_phrases
```

```
[7]: WordList([
         'negation',
         'interesting way',
         'inverse intensity',
         'modifier enters'
         ])
```

3.4. Obrada riječi

Kao što je već spomenuto kod riječi, klasa `Word` služi za obradu pojedinačnih riječi. Ona nasljeđuje klasu `BaseBlob` tako da se mnogo metoda klase `TextBlob` može naći i u klasi `Word`.

3.4.1. Općenite značajke

Klasa `Word` zna mnoge funkcionalnosti koje imaju rječnici. Tako možemo pronaći definicije, korijene riječi itd.

3.4.1.1. Korijen riječi

Za početak možemo pogledati osnovne riječi koje dobijemo iz izvedenica. U ovom slučaju izvedenica je *playing* i cilj je pronaći osnovnu riječ od te riječi za slučaj da je ona:

- imenica
- glagol

Vrstu riječi prosljeđujemo metodi `lemmatize()` tako da je početno slovo vrste riječi argument funkcije. Tako za riječ *playing* postoje dva slučaja:

[82]:
```
w = Word('playing')
print(w.lemmatize("n")) #noun
print(w.lemmatize("v")) #verb
```

```
playing
play
```

3.4.1.2. Provjera pisanja (*spellcheck*)

Ponekad se dogodi da u brzini pisanja se slovo propusti. Za provjeru može se iskoristiti nekoliko metoda klase `Word`. Metoda `spellcheck()` nudi riječi koje najbolje opisuju pogrešno napisanu riječ. Algoritam koji stoji iza ovoga gleda koje se sve riječi mogu ispraviti u dva poteza (dodati slovo, obrisati slovo, zamijeniti slova se broje kao jedan potez) te daje vjerojatnosti za svaku od mogućih opcija. Opcije se rangiraju prema tome koje su riječi rangirane kao češće korištene.

[99]:
```
Word('playying').spellcheck()
```

[99]:
```
[('playing', 1.0)]
```

[101]:
```
Word('haee').spellcheck()
```

```
[101]: [('have', 0.9828362408553742),
        ('hare', 0.010129431626336522),
        ('hate', 0.005627462014631401),
        ('haze', 0.0011254924029262803),
        ('hale', 0.0002813731007315701)]
```

Ukoliko imamo dovoljno povjerenja u biblioteku, možemo automatski prihvatiti najvjerojatniju opciju za ispravljanje korištenjem metode `correct()`.

```
[87]: TextBlob('playying').correct()
```

```
[87]: TextBlob("playing")
```

3.4.1.3. Rječnički alati

Nakon što su sve riječi ispravljene, one se mogu i potražiti u rječnicima. Njihove se definicije mogu pronaći metodom `define()` te svojstvom `definitions`. Metoda `define()` omogućava prosljeđivanje `POS` oznake kako bi se prema njoj definirala riječ. I `define()` i `definitions` vraćaju listu stringova s definicijama.

```
[89]: tb_w.define('n')
```

```
[89]: ['the act of playing a musical instrument',
       'the action of taking part in a game or sport or other␣
       ↪recreation',
       'the performance of a part or role in a drama']
```

`TextBlob` zna i prevoditi riječi na druge jezike koristeći *Google Translate* API-je.

```
[91]: TextBlob('Cjelovito').detect_language()
```

```
[91]: 'hr'
```

Ovo nije posve pouzdano jer API ne vraća sve moguće rezultate jezika tako da neke riječi koje su univerzalne u balkanskim područjima obično vraćaju bosanski ili srpski jezik, a zapravo nema naznaka da je to isključivo taj jezik.

Kako se ne bi dogodila takva greška, moguće je detektirati više riječi ili rečenice istovremeno, a moguće je odmah i prevoditi. Metoda `translate()` prevodi s detektiranog jezika na engleski.

```
[105]: TextBlob('Ovo vjerojatno nitko ne razumije').translate()
```

```
[105]: TextBlob("Probably no one understands this")
```

Ukoliko imamo nekih posebnih želja za jezikom, možemo defintirati argumente `from_lang` i `to`. Kad se malo poigramo opisanim funkcionalnostima možemo napraviti i naš mali prevoditelj koji umjesto sinonima opisuje te riječi na jeziku kojem želimo.

[113]:
```
for definition in tb_w.define('n'):
    print(TextBlob(definition).translate(from_lang='en', to='hr'))
```

```
čin sviranja glazbenog instrumenta
radnja sudjelovanja u igri ili sportu ili drugoj rekreaciji
izvođenje dijela ili uloge u drami
```

3.4.2. Imenice

Ovdje će biti pokazano kako `TextBlob` radi s imenicama. Uzmimo riječ *playing* za primjer (koja, ovisno o kontekstu, može biti imenica i glagol).

[83]:
```
tb_w = Word('playing')
```

Prvo ju pretvorimo u jedninu i množinu. Ovo je zapravo jako jednostavna operacija s obzirom na to da se radi o dodavanju i oduzimanju slova `s` na kraju riječi.

[95]:
```
tb_w.pluralize()
```

[95]: `'playings'`

[96]:
```
tb_w.pluralize().singularize()
```

[96]: `'playing'`

Kako bismo provjerili pouzdanost pretvaranja jednina i množina pokušajmo uzeti neku težu riječ kao što je *octopus*. U engleskom je generalno pravilo dodati *-s* na kraj riječi kako bi se dobila množina. Iz staroengleskog vuče se još nekoliko starih množina s dodavanjem sufiksnog digrafa *-en* (*child – children*). Ponešto riječi koje dolaze iz latinskog (osobito one koje završavaju na *-us*) se deklniraju i mijenjaju oblike po pravilima latinskog (npr. *focus – foci*, a pridjev *focal* te *virus – viri*, a pridjev *viral* itd.)

Međutim malotko zna množinu riječi *octopus*. Na nju se ne dodaje nastavak *-s* pa da dobijemo *octopuses*, niti se nastavak *-us* mijenja u *-i* kao u latinskom. Zapravo je to grčka riječ te se deklinira kao *octopus-octopodes*. Ovo je pravi test za provjeru znanja riječi.

[14]:
```
Word('octopus').pluralize()
```

[14]: `'octopodes'`

3.5. Sentimenti

Analiza sentimenata je analiziranje mišljenja i osjećaja koje je netko izrazio u nekom tekstu. Analiza sentimenata obično ima dvije dimenzije:

- *polarity* - pozitivna ili negativna izjava, a rezultati su u opsegu $[-1:1]$, gdje je -1 krajnje negativna, a 1 krajnje pozitivna izjava

- *subjectivity* - koliko je izjava subjektivna, rezultati su u opsegu $[0 : 1]$, gdje je 1 krajnje subjektivna, a 0 krajnje objekivna izjava

Kod analize sentimenata moramo uzeti u obzir da sentimenti nemaju veze s provjerom iznesenih rečenica. One, dakle, mogu biti krajnje istinite ili krajnje lažne, ali ako su formirane kao činjenice, imat će nizak rezultat u svojstvu *subjectivity*.

U prvom primjeru prikazan je primjer rečenice koja je činjenične naravi unatoč tome što je autor uvjeren da je Antananarivo glavni grad Hrvatske, a ne Madagaskara. Sentimentima pristupamo pomoću svojstva `sentiment`, a pojedinačnim sentimentima možemo pristupiti pomoću svojstava `polarity` i `subjectivity`.

[24]:
```
a = TextBlob("Antananarivo is a capital of Croatia")
```

[25]:
```
a.sentiment
```

[25]:
```
Sentiment(polarity=0.0, subjectivity=0.0)
```

Svojstvo *polarity* iznosi 0, što znači da je neutralna izjava u dihotomiji pozitivno-negativno, a *subjectivity*, koji također iznosi 0, znači da je izjava objektivne naravi.

Idući primjeri pokazuju izjave koje su više naravi mišljenja, a ne činjenice.

[24]:
```
b = TextBlob("In my humble opinion we should buy ice cream")
c = TextBlob("I think we can complete this task")
```

[26]:
```
b.sentiment
```

[26]:
```
Sentiment(polarity=-0.2, subjectivity=0.4)
```

[27]:
```
c.sentiment
```

[27]:
```
Sentiment(polarity=0.1, subjectivity=0.4)
```

4. Primjer primjene `TextBloba`

Kao grandiozni završetak rada bit će predstavljen primjer stvarnog rada pomoću `TextBloba` uz zajedničke snage s bibliotekama `pandas` i `matplotlib`.

U ranijem dijelu rada pokazano je kako izgleda čišćenje podataka na primjeru *tweetova* Donalda Trumpa. Na tom primjeru možemo demonstrirati trendove u sentimentima nad sadržajem *tweetova*.

Za početak analize sentimenata potrebno je napraviti *lambda* izraze pomoću kojih će se napraviti izračuni.

[9]:
```
polarity = lambda x: TextBlob(x).sentiment.polarity
subjectivity = lambda x: TextBlob(x).sentiment.subjectivity
```

U tim podatcima idući je korak dodati stupce u kojima će se nalaziti izračuni sentimenata. Ti će se stupci zvati po imenu sentimenta koji se analiziraju.

[10]:
```
data['polarity'] = data.content.apply(polarity)
data['subjectivity'] = data.content.apply(subjectivity)
```

Sad kad je *dataframe* popunjen svim podatcima koji su nam potrebni, moramo ih jedino još nacrtati na graf. S obzirom na to da je u podatcima prisutno preko 40000 *tweetova*, potrebno je učiniti nešto za ublažavanje strmih uspona i padova. Za to se koristi *rolling average* koja uzima prosjek vrijednosti u nekom okviru (u primjeru okvir je velik 1000 vrijednosti).

[79]:
```
rolling = data.[polarity|subjectivity].rolling(window=1000).mean()
rolling.index = pd.to_datetime(df.index)
fig = rolling.plot(
        grid=True,
        style='g-',
        title="Polaritet/Subjektivnost Trumpovih tweetova kroz␣
 ↪vrijeme"
)
fig.set_xlabel("Godina")
fig.set_ylabel("polaritet/subjektivnost")
```

Na kraju izvršavanja ovih naredbi s modifikacijom riječi `polarity` i `subjectivity` dobivamo grafove:

Slika 1: Prikaz subjektivnosti Trumpovih *tweetova* kroz zadnjih osam godina

Slika 2: Prikaz polariteta Trumpovih *tweetova* kroz zadnjih osam godina

Iz danih dijagrama lako možemo povući zaključke – unatoč tome što autor nije upućen u društveno-politička zbivanja u SAD-u, lako je bilo pokazati da je za vrijeme predsjedničke kampanje subjektivnost i polaritet *tweetova* drastično pao. Također se može zaključiti da polaritet nije pao u negativno, već su *tweetovi* postali *manje pozivitivni*.

5. Zaključak

Ovaj projekt omogućio je svakom čitatelju upoznavanje s `TextBlobom` te uvod u neurolingvističko programiranje. `TextBlob` je poprilično jednostavan alat i stoga omogućuje početnicima u NLP-u strmiju krivulju učenja koja je poprilično polegnuta kod korištenja biblioteke `NLTK`.

Takav pojednostavljeni pristup NLP-u je dobar za početak, no dosta se rano mogu doživjeti ograničenja koja se mogu nadići tek malo snažnijim bibliotekama kao što su `NLTK`, `Stanford CoreNLP`, `spaCy` i slični.

Bibliografija

[1] Shivam Bansal. Ultimate guide to understand and implement natural language processing (with codes in python). https://www.analyticsvidhya.com/blog/2017/01/ultimate-guide-to-understand-implement-natural-language-processing-codes-in-python/. Zadnje dostupno: 02.06.2020.

[2] Steven Loria. sloria/textblob. https://github.com/sloria/TextBlob. Zadnje dostupno: 02.06.2020.

[3] Steven Loria. Textblob: Simplified text processing. https://textblob.readthedocs.io/. Zadnje dostupno: 02.06.2020.

[4] Usman Malik. Python for nlt. Serija članaka, https://stackabuse.com/tag/nlp/. Zadnje dostupno: 02.06.2020.

[5] Allison Parrish. Textblob sentiment: Calculating polarity and subjectivity. http://rwet.decontextualize.com/book/textblob/. Zadnje dostupno: 02.06.2020.

[6] Dipanjan Sarkar. Emotion and sentiment analysis: A practitioner's guide to nlp. https://www.kdnuggets.com/2018/08/emotion-sentiment-analysis-practitioners-guide-nlp-5.html. Zadnje dostupno: 02.06.2020.

[7] Aaron Schumacher. Textblob sentiment: Calculating polarity and subjectivity. https://planspace.org/20150607-textblob_sentiment/. Zadnje dostupno: 02.06.2020.

Zbog prirode projekta citiralo se nije slijedno i prema dijelu u kojem je nešto pročitano, već su ovi izvori poslužili kao izvori znanja. Stoga je autor pisao svoje primjere temeljeno na ovim izvorima, a ništa se nije doslovno preuzimalo.