# Razvoj i ocjenjivanje alata za analizu sentimenta

**Herčko, Stjepan**




SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

# ZAVRŠNI RAD

**Stjepan Herčko**

Zagreb, 2024.

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

# ZAVRŠNI RAD

Mentori: Student:

Izv.prof.dr.sc. Tomislav Stipančić, dipl. ing. Stjepan Herčko

Zagreb, 2024.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se mentoru doc. dr. sc. Tomislavu Stipančiću na savjetima i pomoći u izradi ovog rada. Dodatnu zahvalu posvećujem cijeloj obitelji koja mi je pružala podršku i davala hrabrost kada mi je manjkala.

Stjepan Herčko

SVEUČILIŠTE U ZAGREBU
**FAKULTET STROJARSTVA I BRODOGRADNJE**
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne i diplomske ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

| Sveučilište u Zagrebu<br>Fakultet strojarstva i brodogradnje | |
|---|---|
| Datum | Prilog |
| Klasa: **602 – 04 / 24 – 06 / 1** | |
| Ur.broj: **15 – 24 –** | |

# ZAVRŠNI ZADATAK

Student: **Stjepan Herčko** JMBAG: **0035216486**

Naslov rada na hrvatskom jeziku: **Razvoj i ocjenjivanje alata za analizu sentimenata**

Naslov rada na engleskom jeziku: **Development and Assessment of a Sentiment Analysis Tool**

Opis zadatka:

Računalna analiza i prepoznavanje govora (engl. Natural Language Processing) je sposobnost računalnog agenta ili softvera da prepoznaje riječi i izraze u jeziku te ih analizira ili pretvara u ljudima čitljivi tekst. Programske aplikacije temeljene na NLP-u su brojne, te uključuju analizu osjećaja, izradu komunikacijskih agenata (engl. Chatbots), glasovno pretraživanje teksta, prepoznavanje govora, filtriranje elektroničke pošte, i dr.

Alati za analizu sentimenta se često temelji na vlastitom rječniku engleskog jezika kojeg je moguće po potrebi nadograđivati. Uključuju gramatička i sintaktička pravila te empirijski izvedene smjernice za utjecaj pomoću pravila na percipirani intenzitet osjećaja u tekstu na razini rečenice. Takvi alati čine jezgru brojnih aplikacija uključujući primjenu u sklopu socijalnih mreža ili informacijskih web portala.

U radu je potrebno:

- integrirati odgovarajući alat u sklopu programske aplikacije za analizu sentimenta u tekstu zajedno s pripadajućim grafičkim sučeljem
- analizirati uspješnost prepoznavanja sentimenta na različitom tekstu imajući u vidu više razina kompleksnosti teksta te različit kontekst
- evaluirati uspješnost alata te napraviti kritički osvrt.

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan: 24. 4. 2024.

Zadatak zadao: izv. prof. dr. sc. Tomislav Stipančić

Datum predaje rada:
**2. rok (izvanredni):** 11. 7. 2024.
**3. rok:** 19. i 20. 9. 2024.

Predviđeni datumi obrane:
**2. rok (izvanredni):** 15. 7. 2024.
**3. rok:** 23. 9. – 27. 9. 2024.

Predsjednik Povjerenstva:
prof. dr. sc. Damir Godec

# SADRŽAJ

# POPIS SLIKA

# POPIS TABLICA

# SAŽETAK

U ovome radu ćemo implementirati jedan od alata za analizu sentimenata unutar teksta i analizirati njegovu uspješnost na temelju unesenog teksta različite razine kompleksnosti kako bi odredili njegovu učinkovitost. VADER (*Valence Aware Dictionary and sEntiment Reasoner*) je alat za analizu sentimenta sa ugrađenim leksikonom engleskog jezika kojeg ćemo implementirati unutar programskog jezika *Pythona*. Također ćemo izraditi grafičko sučelje za navedeni program radi lakšeg unosa i pregleda podataka putem *NiceGUI* biblioteke za izradu sučelja.

Ključne riječi: analizu sentimenta, VADER, *Python*, računalna analiza, prepoznavanje govora

## SUMMARY

In this work we will be implementing a sentiment analysis tool and analysing its proficency based on various complexity text as an input. VADER (*Valence Aware Dictionary and sEntiment Reasoner*) is a sentiment analysis tool with an integrated lexicon of english language that we will implement inside *Python* programming language. Also we will create a graphics user interface for the program for easier input and overview od data with *NiceGUI* library for creating GUI.

Keywords: sentiment analysis , VADER, *Python*, natural language processing

# 1. UVOD

U današnje doba kad sve više čujemo pojam digitalizacija i umjetna inteligencija, čini nam se kao da su statistike jedini izvor informacija koje možemo primijeniti kako bi poboljšali našu uspješnost na tržištu i dobili povratne informacije vezane razvoj. Sa sve većim utjecajem društvenih mreža, foruma i internet portala nastaju ogromne količine podataka u obliku tekstualnih poruka koje predstavljaju izvrstan izvor informacija koje uz pravi način obrade i analize mogu dati vjerodostojne rezultate mišljenja populacije na veliki broj tema. Ovdje dolazimo do pojma analize sentimenta koji je razvijen da nam pomogne upravo u ovome zadatku. U ovome radu ćemo se osvrnuti na jedan takav alat (VADER) koji omogućuje da pomoću računalne analize i prepoznavanja govora (*natural language processing*) dobivamo povratne informacije o osjećajima, dojmovima i impresijama koje korisnik osjeća putem teksta unesenog u program. Također ćemo se osvrnuti na uspješnost i točnost ovog pristupa ispitujući alat na više razina kompleksnosti unesenog teksta kako bi bili sigurni u najbolje rezultate. Na samom kraju ćemo navesti sve moguće upotrebe ovoga alata i promjene koje donosi u današnjem razvoju digitalnog biznisa i marketinga.

# 2. OSNOVNI POJMOVI

U ovome odlomku proći ćemo neke od osnovnih pojmova koji su bitni za shvaćanje teme analize sentimenta te načina na koji se primjenjuje.

## 2.1. Strojno učenje

Definicija strojnog učenja je područje razvoja statističkih algoritama koji mogu učiti iz dobivenih podataka i izvršavati zadatke bez eksplicitnih instrukcija. Strojno učenje pronalazi primjenu u veoma širokom području od vizijskih sustava, prepoznavanja govora ili NLP-a. Pojam strojnog učenja je prvi put spomenut 1959. od strane zaposlenika IBM-a Arthura Samuela (Slika 1). Njegova prva namjena je bila u sklopu izračuna šanse za pobjedu u igri dame, no veća fascinacija je bila probati što vjernije prikazati način čovjekova razmišljanja i kognitivnog procesa. Strojno učenje pokušava stvoriti veze i uzorke između pojedinih primjera i opažanja. Mogućnost sustava koji u sebi imaju ugrađenu mogućnost strojnog učenja je višestruko vrjednija zbog mogućnosti rješavanja naprednih problema. Ono što javnost danas naziva umjetna inteligencija jest algoritam bazirano na analitičkim problemima koji generiraju pravila, predviđanja, odgovore i preporuke ovisno o našim potrebama. Strojno učenje je više puta pokazalo da za specifične zadatke u zatvorenom sustavu ima kognitivne sposobnosti koje mogu višestruko nadmašiti ljudske sposobnosti. Samim tim ovi sustavi su zadovoljili ljudsku potrebu da imamo samoupravljive sustave koji mogu donositi odluke bazirane na dostupnim podatcima, međutim radi se o veoma specijaliziranim slučajevima koji u velikoj većini ne doprinose daljnjem razvoju sustava u drugim granama nego je samo pokušaj dovesti umjetnu inteligenciju do teoretskog savršenstva na malo upotrebnom području.

Kada govorimo o strojnom učenju možemo ga podijeliti na 3 skupine: strojno učenje s nadgledanjem gdje se programu nude ulazni podatci te očekivani podatci na samom kraju programa tj. izlazu. Takav model se zatim dodatno kalibrira nakon svakog pokušaja kako bi se postigla što veća preciznost i točnost. Drugi tip se naziva strojno učenje bez nadgledanja gdje program treba sam prepoznati postojeće uzorke i kauzalne veze između ulaznih podataka te sam izvesti zaključak iz njih. Zadnji tip strojnog učenja nazivamo

pojačano učenje ili *reinforcement learning*. U ovome slučaju glavni cilj nam je umjesto ulaznih i izlaznih podataka sustava, zadati programu trenutno stanje, listu dopuštenih akcija, specificirati cilj i ograničenja te prepustiti sustav samome sebi. Na taj način će sustav na temelju pokušaja i pogreški zaključiti koji je njegov cilj te će biti optimalno nagrađen za taj uspjeh. Najveći uspjeh ovog tipa učenja je postignut unutar računalnih igara, ali također ima primjenu u stvarnome svijetu. Strojno učenje iako napredno nije bez svojih mana i kao takvo nailazi na poteškoće u korištenju na specifičnim zadatcima za koje nisu još dovoljno optimizirani ili prilagođeni. Korištenje strojnog učenja zahtjeva veliku količinu vremena i resursa ovisno o veličini zadatka te isto tako se razlikuje po primjenama. Mogućnost stvaranja veza unutar modela koje je nemoguće objasniti stvaraju nemogućnost logičkog zaključivanja kako će se model ponašati u specifičnom kontekstu zadatka. Kao alati za pomoć u razvoju poslovanja kako ga mi koristimo na primjeru analize sentimenata u ovome radu postaje dodatan alat na kojeg se možemo osloniti i koji nam može pomoći u donošenju odluka.

**Slika 1. Arthur Samuel**

## 2.2. Računalna analiza i prepoznavanje govora

Računalna analiza i prepoznavanje govora ili *natural language processing* (NLP) je tip strojnog učenja koji omogućuje strojevima da što bolje shvate komunikaciju putem ljudskog jezika. Primjena ovog tipa strojnog učenja ima široku upotrebu u današnjem društvu među kojima je najpoznatija za *chatbote* za glasovnim upravljanjem, *GPS* sustave s glasovnim navođenjem te digitalnim asistentima kao što su *Alexa* ili *Siri* na pametnim mobitelima.

Ovaj model nije savršen zbog raznih grešaka koje mogu nastati njegovim treniranjem i učenjem, krivom interpretacijom ili uvođenjem novih riječi u bazu podataka. NLP-ove dijelimo u 3 glavne skupine: Statistički NLP, *Deep learning* NLP te NLP na bazi pravila. NLP je usko vezan s analizom sentimenta jer pomaže u njegovom ostvarivanju. Prvo spominjanje pojma računalne analize i prepoznavanja govora je predložio 1940. godine otac računala Alan Turing. Spominjući u svom članku *„Computing Machinery and Intelligence"* danas poznati *Turingov* test inteligencije sustava spominje u sebi mogućnost ispitivanja prepoznavanja govora i njegove automatske generacije.

Simbolični NLP koji je razvijan između 1950. do 1990. najlakše je opisati putem eksperimenta koji je izveden naziva „Kineska soba". U ovome eksperimentu računalu je zadan strogi set pravila kojih se mora pridržavati te knjiga kineskih frazema sa pitanjima i odgovorima iz kojih je potrebno emulirati govor na način koji zadovoljava zadani set pravila. U današnjem vremenu je rijetkost vidjeti algoritme koji primjenjuju ovu metodu jer postoje naprednije metode za njegovo korištenje.

Statistički NLP čini iskorak u području generiranja govora jer je za vrijeme njegovog razvoja strojno učenje postalo revolucionarni novi pristup koji omogućuje umjesto ručnog pisanja pravila kao u slučaju statističkog NLP-a korištenje dodatnih algoritama za njegovo unaprjeđenje. Dodatno povećanje snage računalnog hardware-a omogućuje brže učenje i dodatan pomaku unaprijed unutar razvoja. Prve metode statističkog NLP-a su bile učene na više jezičnim zakonicima europskih zemalja čiji visoki stupanj tehničkog korištenja fraza i riječi čine dobar model za učenje jezika i njegovo usavršavanje. Daljnjim razvojem i dostupnosti interneta ovi modeli dobivaju dodatan izvor podataka za učenje i razvoj mogućnosti.

Zadnji tip NLP-a koji je i danas u razvoju jest Neuralni ili *deep learning* NLP kojeg je 2010. godine razvio Tomaš Mikolov. Ove metode su postale široko rasprostranjene te se u velikome broju koriste u medicni za analiziranje medicinskih dokumenata za poboljšanje razvoja zdravstvenog sustava i njegovog učinka. Uz analizu sentimenata koju smo već naveli kao jedan od mogućih upotreba NLP-a tu još postoje drugi zadatci u kojima nam on može pomoći. Neki od njih su prepoznavanje govora, segmentacija govora, optičko prepoznavanje simbola, pretvaranje teksta u govor, segmentacija teksta, gramatička indukcija te mnogi drugi. Diskusija o trendovima i mogućim budućim namjenama NLP-a vode na zaključak razvoja kognicije koju većina visoko razvijenih NLP aplikacija sadrži za inteligentno ponašanje i shvaćanje jezika i komunikacije.

## 2.3. Analiza sentimenta

Za analizu sentimenta možemo najjednostavnije reći da je proces koji potpomognut strojnim učenjem određuje za uneseni tekst kakvo mišljenje korisnik ima na određenu temu, objekt ili izjavu. Kada kažemo mišljenje mislimo na pozitivan, negativan ili neutralan osvrt korisnika. Kako bi to mogli postići potrebno je cijeli proces podijeliti na jednostavne cjeline kako bi mogli dobiti algoritam za rješavanje ovog problema. Postoje 3 različita algoritma koji se koriste za postizanje što bolji rezultata prilikom analize sentimenta putem strojnog učenja. Prvi od tih algoritama su algoritmi strojnog učenja u koje svrstavamo SVM (*Support Vector Machines*), *Naive Bayes* te *random forests* algoritme.

SVM algoritam je nadgledani model koji se koristi za analiziranje podataka i klasifikacija. Algoritam obrađuje veće količine podataka na način da pretvara podatke u koordinate unutar visoko dimenzijskog prostora i na taj način omogućuje linearnu klasifikaciju.

Sljedeći algoritam strojnog učenja je Naive Bayes koji je zapravo jednostavna tehnika za konstruiranje klasifikatora. Bolje objašnjenje njegovog rada u stvarnosti jest da algoritam svakom pojedinom podatku daje određenu vrijednost koja može biti nezavisna od ostalih podataka ili vezana putem vrijednosnih identifikatora. Ovaj pristup iako na prvu ruku

rudimentaran i jednostavan se pokazao kao prikladan algoritam za korištenje u kompleksnim situacijama u stvarnome svijetu pa tako i u analizi sentimenata.

Zadnji algoritam u sklopu strojnog učenja jest *random forests* ili *random decision forests*.. Sličan drugim algoritmima ove skupine, koristi se za metode klasifikacije i regresije. Ovaj algoritam je popularan izbor metode u korištenju za rješavanje problema strojnog učenja. Međutim ovaj algoritam ima velike nedostatke prilikom korištenja u analizi sentimenata zbog velikog broja grešaka koje se stvaraju prilikom učenja na većim modelima što stvara veće komplikacije u korištenju za slučaj alata za analizu sentimenata. Najjednostavniji prikaz bi bio:

1. Razdijeliti tekst u njegove gradivne dijelove/komponente (fraze, rečenice)
2. Identificirati svaku komponentu koja ima određeni sentiment
3. Dodijeliti ocjenu svakoj komponenti koja sadrži sentiment
4. Zbrojiti ocjene pojedinih komponenata kako bi dobili ukupni korisnikov dojam iznesen u tekstu

Najbolji primjer ovakvog pristupa možemo vidjeti na sljedećim rečenicama:

Kiša je jako padala.

Kiša je padala kao iz kabla.

Iako ove dvije rečenice prenose istu poruku jedna od njih je prenosi puno jačim intenzitetom. Ljudi poučeni iskustvom mogu interpretirati iz navedenih primjera da je druga rečenica puno negativnija zbog fraze koja je korištena na kraju rečenice. Analiza sentimenta radi na sličan način uspoređujući poznate pridjeve, fraze i druge oblike komunikacije (emotikoni) sa bazom podataka kako bi odredila njenu težinu/ocjenu i prikazala je krajnjem korisniku.

## 2.4. Baza podataka sentimenta / leksikon

Kako bi alat analize sentimenta izvršavao svoju funkciju, potrebna mu je baza podataka koja će služiti kao izvor informacija za ocjenjivanje i gradaciju izjava navedenih u tekstu. Na sličan način kako ljudski mozak kroz godine na temelju iskustva skuplja razne riječi i veže ih sa njihovim definicijama za što bolje korištenje i razumijevanje tako i baza podataka sentimenta sadrži velik broj riješi koje su ocjenjene od strane razvojnih programera prema svojim pozitivnim ili negativnim konotacijama te intenzitetu emocije koju prenosi.

Održavanje ovakvih baza podataka je veoma zahtjevno jer kao i živa bića jezici se kroz godine mijenjaju, adaptiraju pa neke riječi gube ili dobivaju drugačije težinske ocjene. Pristup analizi sentimenta zavisi od semantike teksta i kao takav je strogo vezan za polarnost riječi i fraza koje se u njemu pojavljuju. To se najviše odnosi na kontekstne riječi kao što su pridjevi, glagoli te fraze u kojima se koriste. Isto tako kod ručnog ocjenjivanja riječi je potrebno biti oprezan jer se radi o subjektivnom mišljenju osobe koja ocjenjuje pa može doći do velikih odstupanja ili varijacija među više osoba. Postoji mogućnost automatskog generiranja leksikona međutim takav pristup ostavlja mogućnost nastanka velikih grešaka i odstupanja a samim time je potrebno više vremena za njegovo treniranje i dovođenje na istu razinu kvalitete kao ručno ocjenjivan. Pošto se analiza sentimenta najčešće koristi na velikom broju korisnika još jedan problem koji nastaje jest izražavanje istog sentimenta u više jezika.

Rečenica iznesena na hrvatskom može imati puno drugačiju ocjenu od one iznesene na francuskom ili njemačkom jer se radi o sasvim drugačijim jezicima. Stoga je kvalitetna baza podataka sentimenta jednako bitna kao i alat koji služi za njegovu analizu.

Još jedna bitna stavka koja omogućuje da se baze podataka sentimenta razvijaju jest da je većina njih uzima engleski i kineski kao glavne jezike tj. „Više jezike“ dok svi drugi jezici imaju manju važnost. Takav pristup umanjuje raširenost korištenja metoda analize sentimenta zbog malog broja jezika koje podržava međutim korištenjem metoda i algoritama NLP-a moguće je pouzdano stvarati kvalitetne leksikone i na drugim jezicima koji otvaraju poglede na druga tržišta i za druge namjene osim engleskog i kineskog.

Iz primjera u stvarnome životu moguće je uvidjeti da korištenje leksikona za analizu sentimenata je poželjno kada se radi o malim količinama podatka jer omogućuju brže ljudsko ocjenjivanje te točnije i preciznije rezultate. Rezultati ispitivanja pokazuju da testiranja na alatima za analizu sentimenata koji su provedeni na klasifikaciji pozitivni/negativni sentiment ima točnost 95,5 – 99,8% dok kod klasifikacije pozitivni/negativni/neutralni točnost varira između 92 – 97,3%.

## 2.5. VADER

VADER ili *Valence Aware Dictionary and sEntiment Reasoner* je alat za analizu sentimenta specifično razvijen za analizu socijalnih poruka. Radi se o *open source* alatu koji je nadograđivan i usavršavan od njegove prvotne iteracije 2014. godine. VADER je moguće koristiti u velikom broj različitih programskih jezika (*Java, Javascript, C#, R*), ali je razvijen za *python* u kojem ćemo ga i koristiti. VADER također dolazi sa testiranom bazom podataka i leksikonom riječi koji omogućuju brz rad i točne povratne ocjene unesenih tekstualnih podataka. Pošto se radi o alatu za analizu na bazi leksikona, temeljen je na engleskome jeziku te nije multilingvalan. VADER svaku rečenicu unutar ulaznih podataka pretvara u vektor sa pozitivnim, negativnim ili neutralnim polaritetom. Pošto se radi o alatu koji se bazira na pravilima, puno je brži od algoritama baziranih na strojnome učenju jer ne zahtjeva period učenja. Uz VADER postoje i par drugih alata za analizu sentimenata kao što su Word2Vec koji dizajnira vektore iz pojedinih riječi te im pridodaje vrijednosti koje svojim spajanjem mogu dati vrijednosti slične riječima istog značenja i na taj način putem strojnog učenja procesuirati i naučiti njihovo značenje i sentiment.

TF-IDF je još jedan od pristupa analizi sentimenta a bazira se na pronalasku ključnih riječi unutar zadanog dokumenta te uspoređivanju sa zadanim setom dokumenata. Ovisno koliko često je navedena riječ spominjana joj se vrijednost povećava ili smanjuje. U ispitivanju provedenom 2019. na socijalnim mrežama za vrijeme COVID-19 pandemije je VADER korišten za analizu sentimenta vezano uz određene ključne riječi. U toj studiji je prikazana razlika koju čišćenje podataka prije unošenja u VADER čini. Razlika u bodovanju sentimenta je bila višestruko očita, a sami krajnji rezultati su odskakali od seta ulaznih podataka koji nisu bili pročišćeni. Priprema ulaznih podataka pročišćavanjem koje je

kasnije integrirano u sam program, skraćivanje poruka na one prihvatljive duljine radi što boljeg rezultata.

## 2.6. Druge metode analize sentimenta

Uz VADER koji je alat za analizu sentimenta koji se bazira na pristupu putem leksikona važno je spomenuti da postoje i drugi načini za analizu sentimenta. Pristup putem rječnika je drugi način koji je sličan pristupu putem leksikona u ideji no razlike su u samom algoritmu i načinu rješavanja problema. Alati sa pristupom putem rječnika koriste mali set poznatih riječi koje su ručno skupljanje te imaju poznatu polarnost sentimenta . Unesene riječi se potom korištenjem raznih internetskih rječnika i leksičkih baza podataka kao što je *Thesaurus* i *WordNet* pronalazi sinonime i antonime poznatih riječi.

Pronađene riječi su dodane u bazu podataka te im je zadana vrijednost za ocjenjivanje sentimenta. Nakon toga započinje nova iteracija programa dok više nema novih pronađenih riječi. Nakon što je proces završen rezultati se ručno pregledavaju za ispravak i uklanjanje grešaka. Nedostatak ovakvog pristupa je manjak konteksta za određene riječi te manjak kolokvijalnih izraza koje je moguće pronaći u svakodnevnom govoru. Statistički pristup je također jedan od pristupa alatima za analizu sentimenta. Pronalaženjem uzoraka riječi koje se često ponavljaju i njihovim uzorkovanjem možemo dobiti povratnu informaciju o polarnosti sentimenta prema tome koliko često je riječ korištena te unutar kojeg konteksta (pozitivan kontekst/negativan kontekst) u samome tekstu. Ovaj pristup iako dobar uvelike ovisi o veličini samog teksta na kojem se primjenjuje zbog samog principa pristupa. Što imamo veću količinu ulaznog teksta kojeg je potrebno raščlaniti to su veće šanse da će statistički pristup biti točniji. Nedostatak ovakvog pristupa je mogućnost zloupotrebe korištenjem lažnih recenzija ili postova na društvenim mrežama za mijenjanje javnog mišljenja. Zadnji bitan pristup koji važno spomenuti jest semantički pristup čiji rad se temelji na direktnom davanju ocjena i ovisi od drugih metoda računanja sličnosti između riječi. Najčešći pristup je korištenje spoja statističkog i semantičkog pristupa radi što boljeg postizanja rezultata. Dodatno postoji i mogućnost stvaranja modela leksikona za ovaj pristup no samim time se gubi mogućnost brze izrade programa zbog dodatne kompleksnosti provjere i ručnog ocjenjivanja pojedinih idioma ili sinonima.

# 3. PRINCIP RADA

U ovom poglavlju ćemo pojasniti princip rada VADER alata za analizu sentimenta prolazeći kroz program, evaluirajući njegove gradivne cjeline. Svaki pojedini dio koda je zadužen za svoju ulogu kao što je pročišćavanje teksta, provjera negacija ili normalizacija ocjena. Kroz svaki dio ćemo detaljno proći i objasniti čemu služi.

Prva stvar koju program izvodi jest da poziva leksikon riječi (Slika 2) u kojem se nalaze empirijske vrijednosti i ocjene riječi, akronima, slenga i emotikona koji se mogu pronaći u svakodnevnim porukama koje korisnici međusobno šalju ili objavljuju. VADER-ov leksikon sadrži preko 7000 riječi engleskog jezika koji su ručno ocjenjeni od strane objektivnih ispitivača koji osiguravaju točnost i nepristranost. Pošto se radi o programu koji je baziran na pravilima leksikona bez njegovog pozivanja i kasnije upotrebe, nije moguće pravilno ocijeniti daljnji proces ocjenjivanja sentimenta.

```
def make_lex_dict(f):
    return dict(map(lambda (w, m): (w, float(m)), [wmsr.strip().split('\t')[0:2] for wmsr in open(f) ]))

f = 'vader_sentiment_lexicon.txt'
try:
    word_valence_dict = make_lex_dict(f)
except:
    f = os.path.join(os.path.dirname(__file__),'vader_sentiment_lexicon.txt')
    word_valence_dict = make_lex_dict(f)
```

**Slika 2. Kod za pozivanje leksikona**

Naknadno sljedeći dio programa (Slika 3) pročišćava zadani ulazni tekst uklanjajući nepotrebne dijakritičke znakove i jednoslovne riječi engleskog jezika (I, a) te na taj način olakšava daljnju analizu i procesiranje unesenog teksta. Pročišćavanje teksta čini veliku razliku prilikom procesuiranja podataka jer VADER nije integriran za rad na dugim tekstovima pošto se radi najviše o alatu za analizu sentimenta na socijalnim mrežama. Ocijene sentimenata mogu višestruko odskakati od očekivanih parametara ako se tekst ne prilagodi VADER-u.

```
regex_remove_punctuation = re.compile('[%s]' % re.escape(string.punctuation))

def sentiment(text):
    """

    """
    wordsAndEmoticons = str(text).split()
    text_mod = regex_remove_punctuation.sub('', text)
    wordsOnly = str(text_mod).split()

    for word in wordsOnly:
        if len(word) <= 1:
            wordsOnly.remove(word)

    puncList = [".", "!", "?", ",", ";", ":", "-", "'", "\"",
                "!!", "!!!", "??", "???", "?!?", "!?!", "?!?!", "!?!?"]
    for word in wordsOnly:
        for p in puncList:
            pword = p + word
            x1 = wordsAndEmoticons.count(pword)
            while x1 > 0:
                i = wordsAndEmoticons.index(pword)
                wordsAndEmoticons.remove(pword)
                wordsAndEmoticons.insert(i, word)
                x1 = wordsAndEmoticons.count(pword)

            wordp = word + p
            x2 = wordsAndEmoticons.count(wordp)
            while x2 > 0:
                i = wordsAndEmoticons.index(wordp)
                wordsAndEmoticons.remove(wordp)
                wordsAndEmoticons.insert(i, word)
                x2 = wordsAndEmoticons.count(wordp)

    for word in wordsAndEmoticons:
        if len(word) <= 1:
            wordsAndEmoticons.remove(word)
```

**Slika 3. Kod za uklanjanje dijakritičkih znakova**

Nakon što je tekst pročišćen počinje provjera pojedinih riječi u tekstu počevši od negacija koje bi dale negativnu ocjenu/valenciju tekstu te smanjili njegovu ocjenu (Slika 4). Ako rečenica nema negacija preskače bodovanje i započinje sljedeći dio programa. Negacija u ovome slučaju kod ulaznih podataka stvara promjenu polariteta sentimenta kako bi vrijedilo kada čovjek sluša kroz kolokvijalni razgovor. Dvostruke negacije koje su rijetkost u engleskome jeziku se također provjeravaju radi dodatne primjene promjene polarnosti unutar konteksta unesene rečenice.

```
negate = ["aint", "arent", "cannot", "cant", "couldnt", "darent", "didnt", "doesnt",
          "ain't", "aren't", "can't", "couldn't", "daren't", "didn't", "doesn't",
          "dont", "hadnt", "hasnt", "havent", "isnt", "mightnt", "mustnt", "neither",
          "don't", "hadn't", "hasn't", "haven't", "isn't", "mightn't", "mustn't",
          "needn't", "needn't", "never", "none", "nope", "nor", "not", "nothing", "nowhere",
          "oughtnt", "shant", "shouldnt", "uhuh", "wasnt", "werent",
          "oughtn't", "shan't", "shouldn't", "uh-uh", "wasn't", "weren't",
          "without", "wont", "wouldnt", "won't", "wouldn't", "rarely", "seldom", "despite"]
def negated(list, nWords=[], includeNT=True):
    nWords.extend(negate)
    for word in nWords:
        if word in list:
            return True
    if includeNT:
        for word in list:
            if "n't" in word:
                return True
    if "least" in list:
        i = list.index("least")
        if i > 0 and list[i-1] != "at":
            return True
    return False
```

**Slika 4. Kod za provjeru negacija**

Sljedeći dio programa normalizira povratne rezultate iz zadanog teksta na vrijednost između 1 i -1 koristeći alfa vrijednost koja aproksimira maksimalnu očekivanu vrijednost (Slika 5). Iz ovog komada koda dobivamo povratnu informaciju o ukupnom rezultatu sentimenta rečenice te možemo zaključiti radi li se o pozitivnom ili negativnom sentimentu. Alfa vrijednost koja je navedena u kodu je empirijska vrijednost koju su programeri dobili prilikom testiranja koda.

```
def normalize(score, alpha=15):

    normScore = score/math.sqrt( ((score*score) + alpha) )
    return normScore

def wildCardMatch(patternWithWildcard, listOfStringsToMatchAgainst):
    listOfMatches = fnmatch.filter(listOfStringsToMatchAgainst, patternWithWildcard)
    return listOfMatches


def isALLCAP_differential(wordList):
    countALLCAPS= 0
    for w in wordList:
        if str(w).isupper():
            countALLCAPS += 1
    cap_differential = len(wordList) - countALLCAPS
    if cap_differential > 0 and cap_differential < len(wordList):
        isDiff = True
    else: isDiff = False
    return isDiff
isCap_diff = isALLCAP_differential(wordsAndEmoticons)
```

**Slika 5. Kod za normalizaciju rezultata**

Jednako bitno uz same riječi koje analiziramo za ocjenu su i sami pridjevi koje se nalaze ispred ili iza navedenih riječi. Njihova uloga može uvelike promijeniti kontekst rečenice i njezinu strukturu. Sljedeći dio programa (Slika 6) provjerava uneseni tekst za neke od navedenih pridjeva unutar engleskog jezika kao uvećanice ili umanjenice ocjene na način da ih množi sa standardiziranom vrijednošću varijable izračunate empirijski. Sama vrijednost ocjene sentimenta pridjeva uvelike utječe na kontekst cijele rečenice. Najviše zato jer se radi o kratkim rečenicama unesenim putem socijalnih mreža koje su namijenjene za kratke izjave i komunikacije. Prilikom testiranja programa o čemu je više navedeno u idućem poglavlju nijednom nije došlo do zabune samog alata u nekoj od kompliciranijih primjera teksta koji bi „prevarili“ sam program. Iako se radi o alatu koji se bazira na pravilima leksikona kod kojeg su greške moguće nijedna nije pronađena tokom testiranja.

```
b_incr = 0.293 #(empirically derived mean sentiment intensity rating increase for booster words)
b_decr = -0.293
# booster/dampener 'intensifiers' or 'degree adverbs'
booster_dict = {"absolutely": b_incr, "amazingly": b_incr, "awfully": b_incr, "completely": b_incr, "considerably": b_incr,
                "decidedly": b_incr, "deeply": b_incr, "effing": b_incr, "enormously": b_incr,
                "entirely": b_incr, "especially": b_incr, "exceptionally": b_incr, "extremely": b_incr,
                "fabulously": b_incr, "flipping": b_incr, "flippin": b_incr,
                "fricking": b_incr, "frickin": b_incr, "frigging": b_incr, "friggin": b_incr, "fully": b_incr, "fucking": b_incr,
                "greatly": b_incr, "hella": b_incr, "highly": b_incr, "hugely": b_incr, "incredibly": b_incr,
                "intensely": b_incr, "majorly": b_incr, "more": b_incr, "most": b_incr, "particularly": b_incr,
                "purely": b_incr, "quite": b_incr, "really": b_incr, "remarkably": b_incr,
                "so": b_incr,  "substantially": b_incr,
                "thoroughly": b_incr, "totally": b_incr, "tremendously": b_incr,
                "uber": b_incr, "unbelievably": b_incr, "unusually": b_incr, "utterly": b_incr,
                "very": b_incr,

                "almost": b_decr, "barely": b_decr, "hardly": b_decr, "just enough": b_decr,
                "kind of": b_decr, "kinda": b_decr, "kindof": b_decr, "kind-of": b_decr,
                "less": b_decr, "little": b_decr, "marginally": b_decr, "occasionally": b_decr, "partly": b_decr,
                "scarcely": b_decr, "slightly": b_decr, "somewhat": b_decr,
                "sort of": b_decr, "sorta": b_decr, "sortof": b_decr, "sort-of": b_decr}
sentiments = []
for item in wordsAndEmoticons:
    v = 0
    i = wordsAndEmoticons.index(item)
    if (i < len(wordsAndEmoticons)-1 and str(item).lower() == "kind" and \
       str(wordsAndEmoticons[i+1]).lower() == "of") or str(item).lower() in booster_dict:
        sentiments.append(v)
        continue
    item_lowercase = str(item).lower()
    if  item_lowercase in word_valence_dict:

        v = float(word_valence_dict[item_lowercase])
```

**Slika 6. Kod za provjeru pridjeva**

Za prenošenje emocija unutar teksta prilikom kolokvijalnog govora je također bitan i način pisanja rečenice, tj. riječi pisane velikim slovima daju dodatan naglasak na iznesenu riječ i njeno značenje. Tako sljedeći dio programa provjerava sve riječi unutar unesene rečenice za riječi napisane velikim slovima i radi li se o riječima koje podižu ocjenu rečenice ili je smanjuju. Za ovaj dio koda (Slika 7) možemo reći da se uvelike razlikuje od ostatka jer ne provjerava što je rečeno u rečenici i njezinim kontekstom nego načinom na koji je izrečen što se razlikuje od drugih načina provjere sentimenta kod drugih dostupnih alata.

```
c_incr = 0.733
if str(item).isupper() and isCap_diff:
    if v > 0: v += c_incr
    else: v -= c_incr


def scalar_inc_dec(word, valence):
    scalar = 0.0
    word_lower = str(word).lower()
    if word_lower in booster_dict:
        scalar = booster_dict[word_lower]
        if valence < 0: scalar *= -1

        if str(word).isupper() and isCap_diff:
            if valence > 0: scalar += c_incr
            else:  scalar -= c_incr
    return scalar
n_scalar = -0.74
if i > 0 and str(wordsAndEmoticons[i-1]).lower() not in word_valence_dict:
    s1 = scalar_inc_dec(wordsAndEmoticons[i-1], v)
    v = v+s1
    if negated([wordsAndEmoticons[i-1]]): v = v*n_scalar
if i > 1 and str(wordsAndEmoticons[i-2]).lower() not in word_valence_dict:
    s2 = scalar_inc_dec(wordsAndEmoticons[i-2], v)
    if s2 != 0: s2 = s2*0.95
    v = v+s2

    if wordsAndEmoticons[i-2] == "never" and (wordsAndEmoticons[i-1] == "so" or wordsAndEmoticons[i-1] == "this"):
        v = v*1.5

    elif negated([wordsAndEmoticons[i-2]]): v = v*n_scalar
if i > 2 and str(wordsAndEmoticons[i-3]).lower() not in word_valence_dict:
    s3 = scalar_inc_dec(wordsAndEmoticons[i-3], v)
    if s3 != 0: s3 = s3*0.9
    v = v+s3
```

**Slika 7. Kod za provjeru velikih slova**

Isti princip kao i u prošlom odlomku vrijedi i za korištenje dijakritičkih znakova na samim krajevima rečenice. Unošenje više uskličnika ili upitnika na krajevima rečenice može iznositi veliku promjenu raspoloženja ili čuđenje. Za potrebe ove primjene alata, potrebna je samo provjera standardnih dijakritičkih znakova (. , ! , ?) unutar komunikacijskog jezika. Provjeru navedenih okolnosti u unesenim rečenicama se provjerava sljedećim djelom programa (Slika 8).

```
ep_count = str(text).count("!")
if ep_count > 4: ep_count = 4
ep_amplifier = ep_count*0.292 #(empirically derived mean sentiment intensity rating increase for exclamation points)
if sum_s > 0:  sum_s += ep_amplifier
elif  sum_s < 0: sum_s -= ep_amplifier

qm_count = str(text).count("?")
qm_amplifier = 0
if qm_count > 1:
    if qm_count <= 3: qm_amplifier = qm_count*0.18
    else: qm_amplifier = 0.96
    if sum_s > 0:  sum_s += qm_amplifier
    elif  sum_s < 0: sum_s -= qm_amplifier

compound = normalize(sum_s)
```

**Slika 8. Kod za provjeru dijakritičkih znakova**

Zadnja provjera prije krajnjeg zbrajanja rezultata i ocjena jest provjera suvremenih idioma koje su česti u današnjoj komunikaciji i radi koji je potrebna dobra baza podataka pojedinih alata za analizu sentimenta zbog njihove same kompleksnosti i načina na koji mogu promijeniti značenje poruke u rečenici. Idiomi je opći neutralan naziv za svaki oblik jezika bez obzira na rang tj. radilo se o standardnom književnom jeziku, razgovornom jeziku ili dijalektu. Danas se nerijetko upotrebljava u značenju konglomerat sustava koji se koriste u današnjoj društveno povijesnoj zajednici. Pošto su idiomi specifični za pojedine jezike i regije, veoma ih je teško interpretirati putem strojnog učenja jer zahtjeva početni kontekst. U sljedećem dijelu koda prikazano je provjeravanje idioma (Slika 9).

```
special_case_idioms = {"the shit": 3, "the bomb": 3, "bad ass": 1.5, "yeah right": -2,
                       "cut the mustard": 2, "kiss of death": -1.5, "hand to mouth": -2}

#other_idioms = {"back handed": -2, "blow smoke": -2, "blowing smoke": -2, "upper hand": 1, "break a leg": 2,
#                "cooking with gas": 2, "in the black": 2, "in the red": -2, "on the ball": 2,"under the weather": -2}
onezero = "{} {}".format(str(wordsAndEmoticons[i-1]), str(wordsAndEmoticons[i]))
twoonezero = "{} {} {}".format(str(wordsAndEmoticons[i-2]), str(wordsAndEmoticons[i-1]), str(wordsAndEmoticons[i]))
twoone = "{} {}".format(str(wordsAndEmoticons[i-2]), str(wordsAndEmoticons[i-1]))
threetwoone = "{} {} {}".format(str(wordsAndEmoticons[i-3]), str(wordsAndEmoticons[i-2]), str(wordsAndEmoticons[i-1]))
threetwo = "{} {}".format(str(wordsAndEmoticons[i-3]), str(wordsAndEmoticons[i-2]))
if onezero in special_case_idioms: v = special_case_idioms[onezero]
elif twoonezero in special_case_idioms: v = special_case_idioms[twoonezero]
elif twoone in special_case_idioms: v = special_case_idioms[twoone]
elif threetwoone in special_case_idioms: v = special_case_idioms[threetwoone]
elif threetwo in special_case_idioms: v = special_case_idioms[threetwo]
if len(wordsAndEmoticons)-1 > i:
    zeroone = "{} {}".format(str(wordsAndEmoticons[i]), str(wordsAndEmoticons[i+1]))
    if zeroone in special_case_idioms: v = special_case_idioms[zeroone]
if len(wordsAndEmoticons)-1 > i+1:
    zeroonetwo = "{} {}".format(str(wordsAndEmoticons[i]), str(wordsAndEmoticons[i+1]), str(wordsAndEmoticons[i+2]))
    if zeroonetwo in special_case_idioms: v = special_case_idioms[zeroonetwo]
```

**Slika 9. Kod za provjeru idioma**

Na kraju program treba vratiti korisniku povratnu informaciju o iznesenim emocijama unutar zadanog teksta (Slika 10). Program raspodjeljuje izračunate ocjene pojedinih riječi u 4 posebne kategorije koje nam na kraju programa nudi kao izlaznu informaciju. Radi se o pozitivnim, negativnim i neutralnim emocijama te ukupnom rezultatu sentimenta unesene rečenice. Iz ovih podataka je moguće ocijeniti kakav je sentiment korisnik želio prenijeti i kada je primijenjen na veliki ulazni broj podataka moguće je dobiti širu i bolju sliku javnog mišljenja o određenim temama.

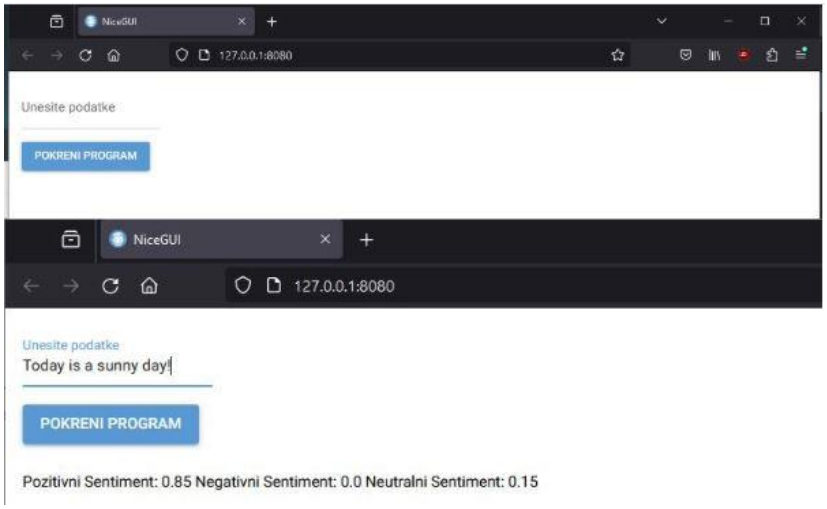
```
        pos_sum = 0.0
        neg_sum = 0.0
        neu_count = 0
        for sentiment_score in sentiments:
            if sentiment_score > 0:
                pos_sum += (float(sentiment_score) +1)
            if sentiment_score < 0:
                neg_sum += (float(sentiment_score) -1)
            if sentiment_score == 0:
                neu_count += 1

        if pos_sum > math.fabs(neg_sum): pos_sum += (ep_amplifier+qm_amplifier)
        elif pos_sum < math.fabs(neg_sum): neg_sum -= (ep_amplifier+qm_amplifier)

        total = pos_sum + math.fabs(neg_sum) + neu_count
        pos = math.fabs(pos_sum / total)
        neg = math.fabs(neg_sum / total)
        neu = math.fabs(neu_count / total)

    else:
        compound = 0.0; pos = 0.0; neg = 0.0; neu = 0.0

    s = {"neg" : round(neg, 3),
         "neu" : round(neu, 3),
         "pos" : round(pos, 3),
         "compound" : round(compound, 4)}
    return s
```

**Slika 10. Kod za ispis emocija**

# 4. IZRADA GRAFIČKOG SUČELJA

Program nije spreman za korištenje bez grafičkog sučelja koje omogućava korisniku laku upotrebu i pregled povratnih informacija. Pošto korisnik nema veliku količinu parametara koje mora definirati prilikom unošenja teksta sve što trebamo jest polje za unos podataka, gumb koji će korisnik stisnuti kada je završio s procesom unosa podataka i želi izlazni rezultat te polja za prikaz prethodno navedenih izlaznih rezultata. U ovome zadatku nam može pomoći jedna od mnogih *Python*-ovih biblioteka za izradu grafičkog sučelja. Za potrebe ovog rada ćemo koristiti *NiceGUI* biblioteku za stvaranje grafičkog sučelja kao *frontenda* našem kodu. Biblioteka omogućava korištenje bilo koje popularnije tražilice (Google Chrome, Mozilla Firefox i sl.) za jednostavno sučelje.

Za komercijalnu upotrebu ovakvih alata bi bile potrebne dodatne mogućnosti prikaza statistika kao što su grafički prikaz podataka koje je program vratio (grafovi, regionalizacija poruka, zadovoljstvo korisnika i sl.). Samim time bi količina informacija i kontekst poruka koje su korisnici ostavili bio puno veći te bi model alata (u ovom slučaju VADER) bilo moguće bolje prilagoditi za buduću upotrebu i razvoj za određene namjene. U tom slučaju bi se koristio i drugačiji pristup izradi grafičkog sučelja radi što kvalitetnijeg i profesionalnijeg izgleda. Daljnja mogućnost direktne komunikacije između stranaka bi dodatno poboljšala postojeći odnos i dala dodatne podatke na kojima alat za analizu sentimenta može učiti.




**Slika 11. Prikaz grafičkog sučelja**

# 5. TESTIRANJE PROGRAMA

Kako bi potvrdili ispravan rad alata za analizu sentimenta potrebno ga je testirati i proučiti rezultate kako bi se uvjerili u njihovu ispravnost. Pošto je VADER alat za analizu sentimenata koji može raditi na tekstovima različitih kompleksnosti, isprobat ćemo ga na više primjera kako bi potvrdili njegovu učinkovitost. Prvi set podataka koji ćemo unijeti u program će biti rečenice sa pozitivnim osjećajima/ valencijama. Koristit ćemo različiti broj pozitivnih pridjeva, lokacija i količine punktacija, idioma te emotikona kako bi usporedili ocjene navedenih. Uneseni primjeri su vidljivi na sljedećoj slici (Slika 11) skupa sa objašnjenima što se točno testira sa pojedinim unosom podataka. Sve unesene rečenice su na engleskome jeziku jer VADER nije multilingvalan alat i stoga ne prepoznaje unose na hrvatskome jeziku i nema izgrađen leksikon za njegovo ocjenjivanje.

```
"VADER is smart, handsome, and funny.",  # primjer pozitivne rečenice
"VADER is smart, handsome, and funny!",  # dobro isticanje interpunkcije (intenzitet sentimenta podešen)
"VADER is very smart, handsome, and funny.", # pomoćne riječi dobro prilagođene (intenzitet sentimenta podešen)
"VADER is VERY SMART, handsome, and FUNNY.",  # dobro isticanje riječi napisanih velikim slovima
"VADER is VERY SMART, handsome, and FUNNY!!!", # kombinacija signala - VADER prikladno prilagođava intenzitet
"VADER is VERY SMART, uber handsome, and FRIGGIN FUNNY!!!", #pomoćne riječi i interpunkcija čine ovo blizu gornje granice za rezultat
"The book was good.",  # pozitivna rečenica
"At least it isn't a horrible book.",  # negacija negativne rečenice s kontradikcijom
"The book was only kind of good.", # pozitivna rečenica je ispravno obrađena (intenzitet sentimenta podešen)
"Today only kinda sux! But I'll get by, lol", # primjer mješanog iznošenja sentimenta sa slengom i veznikom "ali"
"Make sure you :) or :D today!",  # korištenje emotikona
"Catch utf-8 emoji such as such as 💘 and 💋 and 😁",  # korištenje emoji-a
"Not bad at all"  # Kapitalizirana negacija
```

**Slika 12. Primjer pozitivnih sentimenta**

Nakon unošenja svih ovih primjera dobili smo povratne podatke koje je moguće vidjeti u tablici (Tablica 1) i koji daju bolju sliku rada programa. Vrijednosti sentimenta su iskazane na 3 decimale te zbroj pozitivnog, neutralnog i negativnog sentimenta treba uvijek dati broj 1. Isto tako možemo vidjeti u zadnjem stupcu sveukupni rezultat unesene rečenice koji se nalazi u rasponu od -1 do 1 te nam služi kao sveukupna ocjena iznesenih sentimenata u rečenici. U slučaju primjera koje smo unijeli u program sve vrijednosti bi trebale biti veće od nule jer prenose pozitivne osjećaje.

**Tablica 1. Ocjene prvog seta primjera**

| Rečenica | Pozitivni sentiment | Negativni sentiment | Neutralan sentiment | Ukupni rezultat |
|---|---|---|---|---|
| VADER is smart, handsome and funny. | 0.746 | 0 | 0.254 | 0.8316 |
| VADER is smart, handsome and funny! | 0.752 | 0 | 0.248 | 0.8439 |
| VADER is very smart,handsome and funny. | 0.701 | 0 | 0.299 | 0.8545 |
| VADER is VERY SMART, handsome and FUNNY. | 0.754 | 0 | 0.246 | 0.9227 |
| VADER is VERY SMART, handsome and FUNNY!!! | 0.767 | 0 | 0.233 | 0.9342 |
| VADER is VERY SMART, uber handsome and FRIGGIN FUNNY!!! | 0.706 | 0 | 0.294 | 0.9469 |
| The book was good. | 0.492 | 0 | 0.508 | 0.4404 |
| At least it isn't a horrible book. | 0.322 | 0 | 0.678 | 0.431 |
| The book was only kind of good. | 0.303 | 0 | 0.697 | 0.3832 |
| Today only kinda sux! But i'll get by, lol. | 0.317 | 0.127 | 0.556 | 0.5249 |
| Make sure you :) or :D today! | 0.706 | 0 | 0.294 | 0.8633 |
| Catch utf-8 emoji such as <3 and :D. | 0.385 | 0 | 0.615 | 0.875 |
| Not bad at all. | 0.487 | 0 | 0.513 | 0.431 |

Sljedeći set primjera (Slika 12) će biti korišteni za testiranje većinski negativnih sentimenata i kompleksnijih primjera rečenica. Kao i prilikom testiranja prvog seta koristit će se različiti slengovi i idiomi.

```
"Most automated sentiment analysis tools are shit.",
 "Sentiment analysis has never been good.",
 "I like to hate Michael Bay films, but I couldn't fault this one",
 "It's one thing to watch an Uwe Boll film, but another thing entirely to pay for it",
 "This movie was actually neither that funny, nor super witty.",
 "This movie doesn't care about cleverness, wit or any other kind of intelligent humor.",
 "Those who find ugly meanings in beautiful things are corrupt without being charming.",
 "the twin towers collapsed today",
 "However, Mr. Carter solemnly argues, his client carried out the kidnapping under orders and in the ''least offensive way possible.''"
```

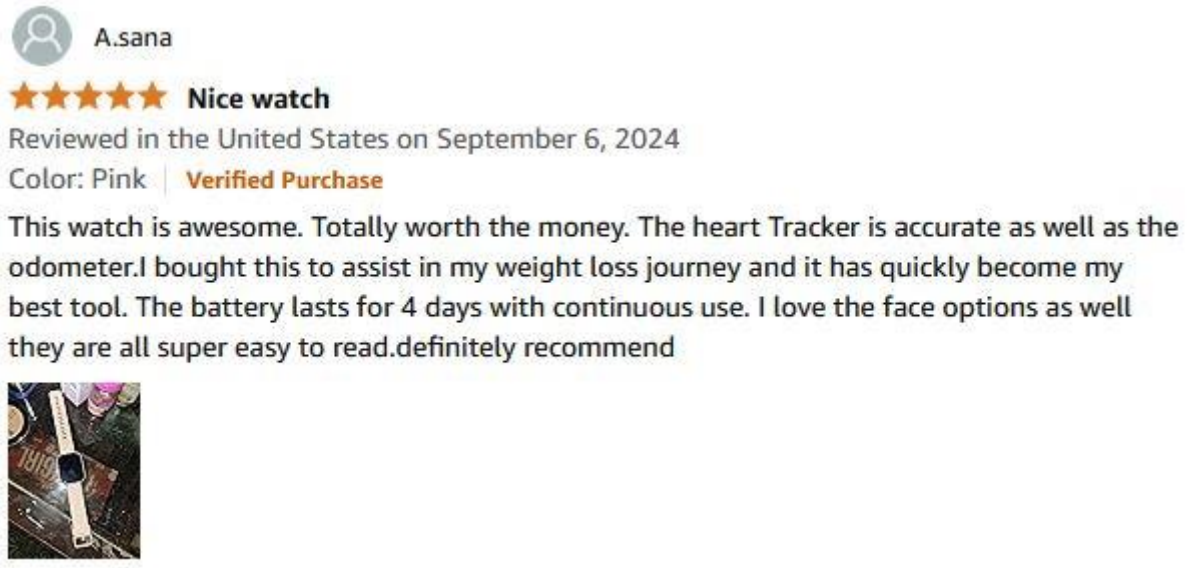
**Slika 13. Primjer negativnih sentimenta**

**Tablica 2. Ocjene drugog seta primjera**

| Rečenica | Pozitivni sentiment | Negativni sentiment | Neutralni sentiment | Ukupni rezultat |
|---|---|---|---|---|
| Most automated sentiment analysis tools are shit. | 0 | 0.375 | 0.625 | -0.5574 |
| Sentiment analysis has never been good. | 0 | 0.325 | 0.675 | -0.3412 |
| I like to hate Miachel Bay films, but i couldn't fault this one. | 0.273 | 0.138 | 0.589 | -0.3153 |
| It's one thing to to watch an Uwe Boll film, but another entirely to pay for it. | 0 | 0.112 | 0.888 | -0.2541 |
| This movie was actually neither that funny, nor super witty. | 0 | 0.41 | 0.59 | -0.6759 |
| This movie doesn't care about cleverness, wit or any other kind of intelligent humor. | 0.239 | 0.265 | 0.479 | -0.1338 |

| | | | | |
|---|---|---|---|---|
| Those who find ugly meanings in beautiful thing are corrupt without being charming. | 0.192 | 0.314 | 0.493 | -0.3553 |
| The twin tower collapsed today. | 0 | 0.344 | 0.656 | -0.2732 |
| However, Mr. Carter solemnly argues, his client carried out the kidnaping under orders and in the “least offensive way possible“. | 0.179 | 0.109 | 0.712 | 0.1729 |

Rezultati ovog seta primjera ponovo odgovaraju kontekstu zadanih rečenica i njihovim sentimentima. Samim tim možemo reći da je program točan u ocjenjivanju sentimenata, ali kao završni test ćemo uzeti jedan tekst iz stvarnog života kako bi u potpunosti sigurni (Slika 13). Tekst u pitanju je recenzija za pametni sat na web stranici Amazona. Alat za analizu sentimenata kao što je VADER je savršen za pregledavanje korisnikovog mišljenja o pojedinim značajkama proizvoda te mogućim nedostatcima.




**Slika 14. Primjer recenzije**

**Tablica 3. Ocjene primjera recenzije**

| Rečenica | Pozitivni sentiment | Negativni sentiment | Neutralni sentiment | Ukupni rezultat |
|---|---|---|---|---|
| This watch is awesome. | 0.577 | 0 | 0.423 | 0.6249 |
| Totally worth the money. | 0.422 | 0 | 0.578 | 0.2944 |
| The heart tracker is accurate as well as odometer. | 0.441 | 0 | 0.599 | 0.743 |
| I bought this to assist in my weight loss journey and it has quickly become my best tool. | 0.187 | 0.102 | 0.711 | 0.4404 |
| The battery lasts for 4 days with continous use. | 0 | 0 | 1 | 0 |
| I love face options as well they are all super easy to read. | 0.567 | 0 | 0.433 | 0.9201 |
| Definitely recommend. | 1 | 0 | 0 | 0.6369 |

Vidimo da su rezultati primjera iz stvarnog života dovoljno blizu onome što bi i prosječna osoba iznijela kao rezultat ako ju se pita o osjećajima koje može pronaći unutar teksta.

Pokušaji da se VADER zavara je teško ostvarivo jer se radi o alatu za analizu sentimenta na bazi primjene leksikona, jedan od načina koji bi javio grešku u slučaju pogrešne povratne informacije jest ako sama rečenica ima kontekstualne i gramatičke greške te ako koristi termine i idiome koji nisu uvršteni u programski leksikon. Koristeći više različitih alata za analizu sentimenta velika je vjerojatnost bi dobili rezultate koji su slični ovima koji su navedeni u tablicama, jedina razlika bi bila u intenzitetu polarnosti sentimenta.

Takvi rezultati su ovisni o više različitih čimbenika koji pošto se radi o alatima za analizu sentimenata na bazi strojnog učenja o vremenu utrošenom na fazu učenja te samoj temi za koju se alat koristi. Primjene za analizu poruka na socijalnim mrežama imaju drugačiji cilj i drugačiji tip podataka koji se pokušava proučiti. Dok u drugim slučajevima kao prilikom pregleda

medicinskih zapisa ili drugih visoko terminoloških spisa isti alat će imati probleme u prepoznavanju određenih sentimenata zbog različitog konteksta poruka.

Prema tome je teže i napraviti pravilnu strukturu za rad programa jer je jezik poprilično apstraktna ljudska građevina. Različiti oblici govora također imaju različite razine apstraktnosti te samim time i veću kompliciranost programa. Dodatna činjenica koju smo naveli u pisanju ovog rada jest razina teksta koju je potrebno raščlaniti. Analiza dokumenta od dvadesetak stranica punog raznolikih sentimenata i njegovo sumiranje u pozitivnu ili negativnu cjelinu nije jednostavan zadatak kao analiza rečenica ili postova na društvenim mrežama.

Mišljenja populacije se mogu veoma brzo mijenjati i sa dodatnim razvojem društvenih mreža potrebno je proširiti način gledanja analize sentimenta ne samo na tekst već i na druge tipove medije. Analiza sentimenta zvuka, slike ili videa je logičan sljedeći korak u razvoju alata kao što su VADER te bi mogli nadograditi postojeće sustave za što bolju točnost prepoznavanja sentimenta i njihovih polarnosti. Također prepoznavanje sentimenta u stvarnom vremenu je dodatan je zadatak koji je moguće ostvariti putem razvoja kao odgovor na moderne trendove.

Primjena alata kao što je VADER daje dovoljno kvalitetne rezultate kad se radi o malim količinama ulaznih podataka te kao takav može biti moćan alat u razvijanju poslovnih ideja te vaganja mišljenja korisnika dok kod primjena na većim sustavima je potrebno dodatno prilagođavanje.

# 6. KRITIČKO RAZMIŠLJANJE

Analiza sentimenata je poprilično kompleksan problem za riješiti, čak i na malim razinama. Ovakvi tipovi programa su uvijek pod utjecajem promjena i grešaka nastalih unutar ljudske komunikacije koje je nemoguće spriječiti. Sama količina podataka koju je moguće obraditi putem ovakvih aplikacija je zapanjujuća i kao takva stvara poseban izazov. Najveći problem ovakve analize podataka su kompleksnost ljudskih emocija koje nisu objektivne veličine i kao takve im je teško precizno odrediti kvalitativnu vrijednost.

Kada tome pridodamo jezične barijere, greške prilikom prevođenja tekstova, kulturološke interpretacije različitih izraza (znak palca gore u većini zapadnih zemalja prenosi pozitivni sentiment dok u nekim zemljama bliskog istoka se smatra uvredljivim) možemo dobiti različite odgovore za isti ulazni podatak. Dodatna otežavajuća okolnost jest i ograničenost same tehnologije. Analizirati par rečenica nije isti zadatak kao i analizirati cijeli tekst te zahtijeva drugačiji pristup problemu. Tu je također i pitanje prilagodbe unutar razgovora u stvarnom vremenu (kada govorimo o uporabi alata unutar *chatbot* aplikacija).

Svi ovi problemi ograničavaju korištenje na specijalne slučajeve no bitno je napomenuti da postoje načini da se greške nastale iz ovakvih ograničenja smanje. Korištenje podataka koji u sebi sadrže kontekstualnu informaciju, korištenje relevantnih izvora za treniranje modela te uvođenje procesa pročišćavanja i filtriranja ulaznih podataka za micanje nerelevantnih informacija će učiniti razliku u daljnjem provođenju istraživanja zadovoljstva korisnika.

Zadnja stavka na koju moramo računati prilikom korištenja VADER-a i sličnih alata jest da baza podataka koju koriste za ocjenjivanje sentimenata i broj riječi koje alat prepoznaje se stalno mijenja jer ovise o jeziku kojeg možemo smatrati živim bićem koje se neprestano mijenja. Idiomi koji se koriste su drugačiji sa svakom novom generacijom i stvaraju poteškoće u održavanju ovakvih alata za analizu ažuriranim te ovise o interpretaciji ljudi za ocjenjivanje sentimenata pojedinih izraza.

Daljnjim skupljanjem podataka i primjera iz stvarnog života moguće je učiniti da sustav napreduje no kako su osjećaji sposobnost živih bića moguće je da nikad neće biti savršen.

# 7. ZAKLJUČAK

Prilikom izrade ovog rada susreo sam se s velikim brojem novih pojmova i saznanja iz područja razvoja strojnog učenja. Napredak ove grane tehnologije je sve više vidljiv iz dana u dan sa povećanjem alata s područja umjetne inteligencije u analizi i generaciji slike, zvuka, teksta i drugih oblika multimedije. Također njihova međusobna integracija u pametne sustave kao autonomna vozila ili robotiku ih čini još impresivnijima.

Veoma zanimljiv misaoni eksperiment je probati zamisliti kakav će biti utjecaj umjetne inteligencije kroz idućih 10-20 godina. U današnje doba kada su informacije, njihovo posjedovanje i manipulacija postale vrijedni izvor profita moramo se zapitati o mogućnostima njihove zloupotrebe. Iznošenje mišljenja je osnovna ljudska sloboda i takva bi trebala ostati. Korištenje alata za analizu sentimenata bi mogla to promijeniti na način da cenzurira nepoželjne izjave i mišljenja pritiskom gumba. Ovo je ekstreman primjer zloupotrebe ovakvog alata no postoje i mogućnosti za njegovo korištenje u svrhu poboljšanja ljudskog života.

*ChatGPT* je glavni primjer u ovom slučaju. Pametan sustav koji odgovara na sva korisnikova pitanja potpomognut sustavom za osjećaje bi uvelike poboljšao interakciju i stvorio revoluciju u obrazovnom sektoru. Dodamo li mogućnost interakcije u virtualnoj stvarnosti smatram da bi učinili veliki skok unaprijed. Vrijeme će pokazati u kojem smjeru će razvoj umjetne inteligencije dalje napredovati no mogu reći da sa velikim uzbuđenjem iščekujem sljedeći korak i sve što nam on donosi.

# LITERATURA

[1] Lexalytics: Sentiment analysis explained

https://www.lexalytics.com/technology/sentiment-analysis/

(pristupljeno 1.7.2024)

[2] Medium: Top 5 open-source sentiment analysis projects in python every NLP engineer should know

https://mrinalwalia.medium.com/top-5-open-source-sentiment-analysis-projects-in-python-every-nlp-engineer-should-know-db12ca9c894b

(pristupljeno 1.7.2024)

[3] Wikipedia: Sentiment analysis
https://en.wikipedia.org/wiki/Sentiment_analysis#Methods_and_features

(pristupljeno 5.7.2024)

[4] IBM: What is NLP(Natural language processing)?

https://www.ibm.com/topics/natural-language-processing

(pristupljeno 6.7.2024)

[5] Medium: VADER: A comperhensive guide to sentiment analysis in python

https://medium.com/@rslavanyageetha/vader-a-comprehensive-guide-to-sentiment-analysis-in-python-c4f1868b0d2e

(pristupljeno 13.8.2024)

[6] Determ : Sentiment analysis Challenges: solutions and approaches

https://determ.com/blog/sentiment-analysis-challenges-solutions-and-approaches/
(pristupljeno 1.8.2024)

[7] Rosario Catelli: Lexicon-based vs. Bert based sentiment analysis:A comparative study in Italian

https://www.mdpi.com/2079-9292/11/3/374

(pristupljeno 9.9 2024)

[8] Toni Pano: A complete VADER based sentiment analysis of bitcoin (BTC) tweets during era of COVID 19

https://www.mdpi.com/2504-2289/4/4/33

(Pristupljeno 9.9.2024)

[9] Walaa Medhat: Sentiment analysis algorithms and applications: A survey

https://www.sciencedirect.com/science/article/pii/S2090447914000550#b0440

(pristupljeno 10.9.2024)

# PRILOZI

I. Link za pristup kodu te sam kod VADER alata za analizu sentimenta:

https://github.com/cjhutto/vaderSentiment

KOD:

```
#!/usr/bin/python
# coding: utf-8
'''
Created on July 04, 2013
@author: C.J. Hutto

Citation Information

If you use any of the VADER sentiment analysis tools
(VADER sentiment lexicon or Python code for rule-based sentiment
analysis engine) in your work or research, please cite the paper.
For example:

  Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for
  Sentiment Analysis of Social Media Text. Eighth International Conference on
  Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.
'''
import os, math, re, sys, fnmatch, string
reload(sys)
```

```
def make_lex_dict(f):
    return dict(map(lambda (w, m): (w, float(m)), [wmsr.strip().split('\t')[0:2] for wmsr in
open(f) ]))

f = 'vader_sentiment_lexicon.txt' # empirically derived valence ratings for words, emoticons,
slang, swear words, acronyms/initialisms

try:
    word_valence_dict = make_lex_dict(f)
except:
    f = os.path.join(os.path.dirname(__file__),'vader_sentiment_lexicon.txt')
    word_valence_dict = make_lex_dict(f)

# for removing punctuation
regex_remove_punctuation = re.compile('[%s]' % re.escape(string.punctuation))

def sentiment(text):
    """
    Returns a float for sentiment strength based on the input text.
    Positive values are positive valence, negative value are negative valence.
    """
    wordsAndEmoticons = str(text).split() #doesn't separate words from adjacent punctuation
(keeps emoticons & contractions)
    text_mod = regex_remove_punctuation.sub('', text) # removes punctuation (but loses
emoticons & contractions)
```

```
    wordsOnly = str(text_mod).split()
    # get rid of empty items or single letter "words" like 'a' and 'I' from wordsOnly
    for word in wordsOnly:
        if len(word) <= 1:
            wordsOnly.remove(word)
    # now remove adjacent & redundant punctuation from [wordsAndEmoticons] while
keeping emoticons and contractions
    puncList = [".", "!", "?", ",", ";", ":", "-", "'", "\"",
                "!!", "!!!", "??", "???", "?!?", "!?!", "?!?!", "!?!?"]
    for word in wordsOnly:
        for p in puncList:
            pword = p + word
            x1 = wordsAndEmoticons.count(pword)
            while x1 > 0:
                i = wordsAndEmoticons.index(pword)

                wordsAndEmoticons.remove(pword)
                wordsAndEmoticons.insert(i, word)
                x1 = wordsAndEmoticons.count(pword)

            wordp = word + p
            x2 = wordsAndEmoticons.count(wordp)

 while x2 > 0:
                i = wordsAndEmoticons.index(wordp)
```

```
            wordsAndEmoticons.remove(wordp)
            wordsAndEmoticons.insert(i, word)
            x2 = wordsAndEmoticons.count(wordp)
    # get rid of residual empty items or single letter "words" like 'a' and 'I' from
wordsAndEmoticons
    for word in wordsAndEmoticons:
        if len(word) <= 1:
            wordsAndEmoticons.remove(word)

    # remove stopwords from [wordsAndEmoticons]
    #stopwords = [str(word).strip() for word in open('stopwords.txt')]
    #for word in wordsAndEmoticons:
    #    if word in stopwords:
    #        wordsAndEmoticons.remove(word)

    # check for negation
    negate = ["aint", "arent", "cannot", "cant", "couldnt", "darent", "didnt", "doesnt",
              "ain't", "aren't", "can't", "couldn't", "daren't", "didn't", "doesn't",
              "dont", "hadnt", "hasnt", "havent", "isnt", "mightnt", "mustnt", "neither",
              "don't", "hadn't", "hasn't", "haven't", "isn't", "mightn't", "mustn't",
              "neednt", "needn't", "never", "none", "nope", "nor", "not", "nothing", "nowhere",
              "oughtnt", "shant", "shouldnt", "uhuh", "wasnt", "werent",
              "oughtn't", "shan't", "shouldn't", "uh-uh", "wasn't", "weren't",
              "without", "wont", "wouldnt", "won't", "wouldn't", "rarely", "seldom", "despite"]
```

```
  def negated(list, nWords=[], includeNT=True):
      nWords.extend(negate)
      for word in nWords:
         if word in list:
            return True
      if includeNT:
         for word in list:


 if "n't" in word:
                return True
      if "least" in list:
         i = list.index("least")
         if i > 0 and list[i-1] != "at":
            return True
      return False


   def normalize(score, alpha=15):
      # normalize the score to be between -1 and 1 using an alpha that approximates the max
expected value
      normScore = score/math.sqrt( ((score*score) + alpha) )
      return normScore


   def wildCardMatch(patternWithWildcard, listOfStringsToMatchAgainst):
      listOfMatches = fnmatch.filter(listOfStringsToMatchAgainst, patternWithWildcard)
      return listOfMatches
```

```
    def isALLCAP_differential(wordList):
        countALLCAPS= 0
        for w in wordList:
            if str(w).isupper():
                countALLCAPS += 1
        cap_differential = len(wordList) - countALLCAPS

    if cap_differential > 0 and cap_differential < len(wordList):
            isDiff = True
        else: isDiff = False
        return isDiff
    isCap_diff = isALLCAP_differential(wordsAndEmoticons)

    b_incr = 0.293 #(empirically derived mean sentiment intensity rating increase for booster
words)

 b_decr = -0.293
    # booster/dampener 'intensifiers' or 'degree adverbs'
http://en.wiktionary.org/wiki/Category:English_degree_adverbs
    booster_dict = {"absolutely": b_incr, "amazingly": b_incr, "awfully": b_incr, "completely":
b_incr, "considerably": b_incr,
                    "decidedly": b_incr, "deeply": b_incr, "effing": b_incr, "enormously": b_incr,
                    "entirely": b_incr, "especially": b_incr, "exceptionally": b_incr, "extremely":
b_incr,
```

```
                "fabulously": b_incr, "flipping": b_incr, "flippin": b_incr,
                "fricking": b_incr, "frickin": b_incr, "frigging": b_incr, "friggin": b_incr, "fully":
b_incr, "fucking": b_incr,
                "greatly": b_incr, "hella": b_incr, "highly": b_incr, "hugely": b_incr, "incredibly":
b_incr,
                "intensely": b_incr, "majorly": b_incr, "more": b_incr, "most": b_incr,
"particularly": b_incr,
                "purely": b_incr, "quite": b_incr, "really": b_incr, "remarkably": b_incr,
                "so": b_incr,  "substantially": b_incr,
                "thoroughly": b_incr, "totally": b_incr, "tremendously": b_incr,
                "uber": b_incr, "unbelievably": b_incr, "unusually": b_incr, "utterly": b_incr,
                "very": b_incr,

                "almost": b_decr, "barely": b_decr, "hardly": b_decr, "just enough": b_decr,
                "kind of": b_decr, "kinda": b_decr, "kindof": b_decr, "kind-of": b_decr,
                "less": b_decr, "little": b_decr, "marginally": b_decr, "occasionally": b_decr,
"partly": b_decr,
                "scarcely": b_decr, "slightly": b_decr, "somewhat": b_decr,
                "sort of": b_decr, "sorta": b_decr, "sortof": b_decr, "sort-of": b_decr}
    sentiments = []
    for item in wordsAndEmoticons:
        v = 0
        i = wordsAndEmoticons.index(item)
        if (i < len(wordsAndEmoticons)-1 and str(item).lower() == "kind" and \
            str(wordsAndEmoticons[i+1]).lower() == "of") or str(item).lower() in booster_dict:
```

```
 sentiments.append(v)
            continue
        item_lowercase = str(item).lower()
        if  item_lowercase in word_valence_dict:
            #get the sentiment valence
            v = float(word_valence_dict[item_lowercase])

            #check if sentiment laden word is in ALLCAPS (while others aren't)
            c_incr = 0.733 #(empirically derived mean sentiment intensity rating increase for using ALLCAPs to emphasize a word)
            if str(item).isupper() and isCap_diff:
                if v > 0: v += c_incr
                else: v -= c_incr

            #check if the preceding words increase, decrease, or negate/nullify the valence
            def scalar_inc_dec(word, valence):
                scalar = 0.0
                word_lower = str(word).lower()
                if word_lower in booster_dict:
                    scalar = booster_dict[word_lower]
                    if valence < 0: scalar *= -1
                    #check if booster/dampener word is in ALLCAPS (while others aren't)
                    if str(word).isupper() and isCap_diff:

              if valence > 0: scalar += c_incr
```

```
                else:  scalar -= c_incr
            return scalar
        n_scalar = -0.74
        if i > 0 and str(wordsAndEmoticons[i-1]).lower() not in word_valence_dict:
            s1 = scalar_inc_dec(wordsAndEmoticons[i-1], v)
            v = v+s1
            if negated([wordsAndEmoticons[i-1]]): v = v*n_scalar
        if i > 1 and str(wordsAndEmoticons[i-2]).lower() not in word_valence_dict:
            s2 = scalar_inc_dec(wordsAndEmoticons[i-2], v)

if s2 != 0: s2 = s2*0.95
            v = v+s2
            # check for special use of 'never' as valence modifier instead of negation
            if wordsAndEmoticons[i-2] == "never" and (wordsAndEmoticons[i-1] == "so" or
wordsAndEmoticons[i-1] == "this"):
                v = v*1.5
            # otherwise, check for negation/nullification
            elif negated([wordsAndEmoticons[i-2]]): v = v*n_scalar
        if i > 2 and str(wordsAndEmoticons[i-3]).lower() not in word_valence_dict:
            s3 = scalar_inc_dec(wordsAndEmoticons[i-3], v)
            if s3 != 0: s3 = s3*0.9
            v = v+s3
            # check for special use of 'never' as valence modifier instead of negation
            if wordsAndEmoticons[i-3] == "never" and \
              (wordsAndEmoticons[i-2] == "so" or wordsAndEmoticons[i-2] == "this") or \
```

```
            (wordsAndEmoticons[i-1] == "so" or wordsAndEmoticons[i-1] == "this"):
             v = v*1.25
        # otherwise, check for negation/nullification
        elif negated([wordsAndEmoticons[i-3]]): v = v*n_scalar

        # check for special case idioms using a sentiment-laden keyword known to SAGE
        special_case_idioms = {"the shit": 3, "the bomb": 3, "bad ass": 1.5, "yeah right": -2,

    "cut the mustard": 2, "kiss of death": -1.5, "hand to mouth": -2}
        # future work: consider other sentiment-laden idioms
        #other_idioms = {"back handed": -2, "blow smoke": -2, "blowing smoke": -2,
"upper hand": 1, "break a leg": 2,
        #              "cooking with gas": 2, "in the black": 2, "in the red": -2, "on the ball":
2,"under the weather": -2}
        onezero = "{} {}".format(str(wordsAndEmoticons[i-1]),
str(wordsAndEmoticons[i]))
        twoonezero = "{} {} {}".format(str(wordsAndEmoticons[i-2]),
str(wordsAndEmoticons[i-1]), str(wordsAndEmoticons[i]))

 twoone = "{} {}".format(str(wordsAndEmoticons[i-2]), str(wordsAndEmoticons[i-1]))
        threetwoone = "{} {} {}".format(str(wordsAndEmoticons[i-3]),
str(wordsAndEmoticons[i-2]), str(wordsAndEmoticons[i-1]))
        threetwo = "{} {}".format(str(wordsAndEmoticons[i-3]),
str(wordsAndEmoticons[i-2]))
        if onezero in special_case_idioms: v = special_case_idioms[onezero]
        elif twoonezero in special_case_idioms: v = special_case_idioms[twoonezero]
```

```
            elif twoone in special_case_idioms: v = special_case_idioms[twoone]
            elif threetwoone in special_case_idioms: v = special_case_idioms[threetwoone]
            elif threetwo in special_case_idioms: v = special_case_idioms[threetwo]
            if len(wordsAndEmoticons)-1 > i:
                zeroone = "{} {}".format(str(wordsAndEmoticons[i]),
str(wordsAndEmoticons[i+1]))
                if zeroone in special_case_idioms: v = special_case_idioms[zeroone]
            if len(wordsAndEmoticons)-1 > i+1:
                zeroonetwo = "{} {}".format(str(wordsAndEmoticons[i]),
str(wordsAndEmoticons[i+1]), str(wordsAndEmoticons[i+2]))
                if zeroonetwo in special_case_idioms: v = special_case_idioms[zeroonetwo]

            # check for booster/dampener bi-grams such as 'sort of' or 'kind of'
            if threetwo in booster_dict or twoone in booster_dict:

          v = v+b_decr

        # check for negation case using "least"
        if i > 1 and str(wordsAndEmoticons[i-1]).lower() not in word_valence_dict \
            and str(wordsAndEmoticons[i-1]).lower() == "least":
            if (str(wordsAndEmoticons[i-2]).lower() != "at" and str(wordsAndEmoticons[i-
2]).lower() != "very"):
                v = v*n_scalar
        elif i > 0 and str(wordsAndEmoticons[i-1]).lower() not in word_valence_dict \
            and str(wordsAndEmoticons[i-1]).lower() == "least":
            v = v*n_scalar
```

```
        sentiments.append(v)

    # check for modification in sentiment due to contrastive conjunction 'but'
    if 'but' in wordsAndEmoticons or 'BUT' in wordsAndEmoticons:
        try: bi = wordsAndEmoticons.index('but')
        except: bi = wordsAndEmoticons.index('BUT')
        for s in sentiments:
            si = sentiments.index(s)
            if si < bi:
                sentiments.pop(si)
                sentiments.insert(si, s*0.5)
            elif si > bi:
                sentiments.pop(si)
                sentiments.insert(si, s*1.5)

    if sentiments:
        sum_s = float(sum(sentiments))
        #print sentiments, sum_s

        # check for added emphasis resulting from exclamation points (up to 4 of them)
        ep_count = str(text).count("!")
        if ep_count > 4: ep_count = 4

    ep_amplifier = ep_count*0.292 #(empirically derived mean sentiment intensity rating
increase for exclamation points)
```

```
    if sum_s > 0:  sum_s += ep_amplifier
    elif  sum_s < 0: sum_s -= ep_amplifier

    # check for added emphasis resulting from question marks (2 or 3+)
    qm_count = str(text).count("?")
    qm_amplifier = 0
    if qm_count > 1:
        if qm_count <= 3: qm_amplifier = qm_count*0.18
        else: qm_amplifier = 0.96
        if sum_s > 0:  sum_s += qm_amplifier
        elif  sum_s < 0: sum_s -= qm_amplifier

    compound = normalize(sum_s)

    # want separate positive versus negative sentiment scores
    pos_sum = 0.0
    neg_sum = 0.0
    neu_count = 0
    for sentiment_score in sentiments:
        if sentiment_score > 0:
            pos_sum += (float(sentiment_score) +1) # compensates for neutral words that are
counted as 1
        if sentiment_score < 0:
            neg_sum += (float(sentiment_score) -1) # when used with math.fabs(), compensates
for neutrals
        if sentiment_score == 0:
```

```
            neu_count += 1


        if pos_sum > math.fabs(neg_sum): pos_sum += (ep_amplifier+qm_amplifier)
        elif pos_sum < math.fabs(neg_sum): neg_sum -= (ep_amplifier+qm_amplifier)



       total = pos_sum + math.fabs(neg_sum) + neu_count
        pos = math.fabs(pos_sum / total)
        neg = math.fabs(neg_sum / total)
        neu = math.fabs(neu_count / total)

    else:
        compound = 0.0; pos = 0.0; neg = 0.0; neu = 0.0

    s = {"neg" : round(neg, 3),
         "neu" : round(neu, 3),
         "pos" : round(pos, 3),
         "compound" : round(compound, 4)}
    return s


if __name__ == '__main__':
    # --- examples -------
    sentences = [
            "VADER is smart, handsome, and funny.",       # positive sentence example
```

"VADER is smart, handsome, and funny!", # punctuation emphasis handled correctly (sentiment intensity adjusted)

"VADER is very smart, handsome, and funny.", # booster words handled correctly (sentiment intensity adjusted)

"VADER is VERY SMART, handsome, and FUNNY.", # emphasis for ALLCAPS handled

"VADER is VERY SMART, handsome, and FUNNY!!!",# combination of signals - VADER appropriately adjusts intensity

"VADER is VERY SMART, really handsome, and INCREDIBLY FUNNY!!!",# booster words & punctuation make this close to ceiling for score

"The book was good.", # positive sentence

"The book was kind of good.", # qualified positive sentence is handled correctly (intensity adjusted)

"The plot was good, but the characters are uncompelling and the dialog is not great.", # mixed negation sentence

"A really bad, horrible book.", # negative sentence with booster words

"At least it isn't a horrible book.", # negated negative sentence with contraction

":) and :D", # emoticons handled

"", # an empty string is correctly handled

"Today sux", # negative slang handled

"Today sux!", # negative slang with punctuation emphasis handled

"Today SUX!", # negative slang with capitalization emphasis

"Today kinda sux! But I'll get by, lol" # mixed sentiment example with slang and constrastive conjunction "but"

]

```
paragraph = "It was one of the worst movies I've seen, despite good reviews. \
Unbelievably bad acting!! Poor direction. VERY poor production. \
The movie was bad. Very bad movie. VERY bad movie. VERY BAD movie. VERY BAD movie!"

from nltk import tokenize
lines_list = tokenize.sent_tokenize(paragraph)
sentences.extend(lines_list)

tricky_sentences = [
        "Most automated sentiment analysis tools are shit.",
        "VADER sentiment analysis is the shit.",
        "Sentiment analysis has never been good.",
        "Sentiment analysis with VADER has never been this good.",
        "Warren Beatty has never been so entertaining.",
        "I won't say that the movie is astounding and I wouldn't claim that the movie is too banal either.",
        "I like to hate Michael Bay films, but I couldn't fault this one",
        "It's one thing to watch an Uwe Boll film, but another thing entirely to pay for it",
        "The movie was too good",

      "This movie was actually neither that funny, nor super witty.",
        "This movie doesn't care about cleverness, wit or any other kind of intelligent humor.",
```

"Those who find ugly meanings in beautiful things are corrupt without being charming.",

"There are slow and repetitive parts, BUT it has just enough spice to keep it interesting.",

"The script is not fantastic, but the acting is decent and the cinematography is EXCELLENT!",

"Roger Dodger is one of the most compelling variations on this theme.",

"Roger Dodger is one of the least compelling variations on this theme.",

"Roger Dodger is at least compelling as a variation on the theme.",

"they fall in love with the product",

"but then it breaks",

"usually around the time the 90 day warranty expires",

"the twin towers collapsed today",

"However, Mr. Carter solemnly argues, his client carried out the kidnapping under orders and in the "least offensive way possible.""

I. ]

II. sentences.extend(tricky_sentences)

for sentence in sentences:

III. print sentence,

IV. ss = sentiment(sentence)

V. print "\t" + str(ss)

VI.

print "\n\n Done!"