Postupci mjerenja semantičke sličnosti tekstova

Kisiček, Ivan

SVEUČILIŠTE U RIJECI
ODJEL ZA INFORMATIKU

Studijski program Informacijski i komunikacijski sustavi

Ivan Kisiček

POSTUPCI MJERENJA SEMANTIČKE SLIČNOSTI TEKSTOVA

Diplomski rad

Mentorica : izv. prof. dr. sc. Ana Meštrović

Rijeka, 2018.

SADRŽAJ

SAŽETAK

U ovom diplomskom radu bit će opisani postupci za mjerenje semantičke sličnosti tekstova povodom čega je napravljena i detaljna analiza postupaka za otkrivanje semantičkih sličnosti. Predstaviti ću i opisati metode i algoritme koji se koriste u svrhu određivanja sličnosti tekstova. Opisati ću i programski sustav koji će određivati sličnosti izvornih tekstova. Dva su osnovna pristupa u mjerenju semnatičke sličnosti tekstova; prvi se zasniva na korpusima (engl. corpus-based semantic similarity measures), dok se drugi pristup zasniva na znanju (engl. knowledge-based semantic similarity measures). Mjere semantičke sličnosti koje se zasnivaju na korpusima određuju stupanj sličnosti riječi/tekstova na temelju njihove povezanosti u velikim korpusima koje metoda koristi. To su mjere koje koriste ideju "slične riječi pojavljuju se u sličnom kontekstu". Mjere semantičke sličnosti koje se temelje na znanju koriste neke vanjske izvore kao što su, npr. WordNet ili ontologije kao znanje o leksičkim relacijama između riječi u tekstu. Iz tih izvora mogu se automatski dobiti skupovi sinonima koji se onda koriste za mjerenje sličnosti. U radu će biti opisana oba pristupa. Također, napraviti ću i usporedbu između odabranih pristupa (FuzzyWuzzy i WordNet) na skupu tekstualnih dokumenata kako bi se vidjeli njihovi rezultati kod određivanja sličnosti tekstova.

1. UVOD

Proučavanje semantičke sličnosti među riječima već dugo vremena ima svoju primjenu u obradi prirodnog jezika kao i u srodnim područjima. Određivanje semantičke sličnosti tekstova bazira se na dodjeljivanju određene metrike, odnosno matematičke funkcije za računanje sličnosti, paru tekstova na osnovu stupnja povezanosti njihovih značenja [1]. Sustavi za određivanje semantičke sličnosti kao izlaz daju rezultate u rasponu od 0 do 1, gdje 0 predstavlja potpuno različite tekstove, dok 1 predstavlja da se tekstovi potpuno semantički preklapaju.

Jedan od prvih modela razvijen za mjerenje sličnosti tekstova je vektorski model koji je razvijen za traženje informacije (eng. Information retrieval) [2]. U vektorskom modelu dokument koji je najrelevantniji za ulazni upit se određuje rangiranjem dokumenta u zbirci, u obrnutom redoslijedu njihove sličnosti sa danim upitom. Sličnost tekstova također ima primjene kod klasifikacije tekstova, procjene koherencije (povezanosti) teksta, te kod metoda strojnog prevođenja.

Najčešći pristup koji se koristi u pronalaženju sličnosti između dva teksta je korištenje jednostavne metode leksičke podudarnosti [3] i stvaranje rezultata sličnosti na temelju broja leksičkih jedinica koje se događaju u oba teksta. Iako su do određene mjere uspješne, navedene metode leksičke sličnosti ne mogu uvijek identificirati semantičku sličnosti tekstova. Za primjer, možemo uzeti dvije jednostavne rečenice «Ja posjedujem psa. » i «Ja imam životinju. » među kojima postoji očita sličnost. No, većina trenutačnih metoda za mjerenje sličnosti tekstova neće uspjeti identificirati bilo kakvu vezu između tih tekstova.

Postoji veliki broj problema koji se pojavljuju u postupcima određivanja sličnosti. Na primjer, tokom vremena, značenja pojedinih riječi i asocijacije na njih su se dosta promijenile. Pravi primjer za to je i pojam «apple» koji je u rječnicima naveden u kontekstu voća, dok će korisnici koji preko Interneta pretražuju taj pojam uglavnom biti zainteresirani za «apple» kao tvrtku, a ne kao voće. Nove se riječi konstantno stvaraju, te se tako i postojećim riječima dodaju nova značenja.

Postoji veliki broj mjera za određivanje semantičke sličnosti između riječi koje se baziraju na znanju i na korpusima[1] [4]. Takve se mjere uspješno primjenjuju na zadatke jezične obrade kao što su malopripizam koji predstavlja miješanje riječi uslijed nepoznavanja pravilnog oblika riječi, rješavanje značenja riječi i identificiranje sinonima. Metoda semantičke analize jedna je od najčešćih metoda za određivanje semantičke sličnosti tekstova, a radi na principu da mjeri sličnost tekstova iskorištavanjem redoslijeda riječi automatski dobivenih iz velikih zbirka tekstova.

[1] Korpus – zbirka tekstova prirodnog jezika

U ovom će radu biti predstavljene dvije metode: metoda za mjerenje semantičke sličnosti tekstova koja se temelji na Wordnetu i metoda računanja pomoću biblioteke FuzzyWuzzy, te će se izvršiti usporedba rezultata navedenih metoda, kao i sam njihov opis. Nadalje, u radu će biti opisane dvije mjere (Uzajamna zajednička informacija I Latentna semantička analiza) koje se temelje na korpusima i šest mjera semantičke sličnosti (Leacock & Chodorow, Lesk, Wu & Palmer, Resnik, Lin, Jiang & Conrath) temeljenih na znanju, te će se prikazati kako se one mogu koristiti za izradu mjera sličnosti tekstova. U prvom djelu rada biti će opisana semantika, semantička sličnost te metode i način određivanja semantičke sličnosti među riječima i rečenicama. Nakon toga, slijedi poglavlje u kojem se opisuju pristupi za računanje semantičke sličnosti. Tu su navedeni algoritmi za određivanje sličnosti te su opisane mjere bazirane na znanju i mjere bazirane na korpusima. U ovom djelu rada opisane su i tehnike za definiranje mjere sličnosti. U završnom djelu rada na praktičnom je primjeru pomoću programskog jezika Python prikazan način na koji se izračunava semantička sličnost riječi i tekstova, te usporedba različitih pristupa za njezino izvršavanje.

2. SEMANTIČKA SLIČNOST

2.1. Pojam semantičke sličnosti

Semantika je grana lingvistike koja se bavi proučavanjem značenja jezičnih znakova [5]. Kao disciplina, semantika obuhvaća različite razine i aspekte značenja : leksičke, formalne, funkcionalne i strukturne [6].

Semantička sličnost je koncept povezivanja izraza zasnovanih na sličnostima njihovih značenja [7]. Koncept semantičke sličnosti ima vrlo važnu ulogu kod razumijevanja prirodnih jezika zato što omogućuje izvođenje usporedbi i zaključivanja.

Određivanje semantičke sličnosti veliku primjenu ima u kategorizaciji tekstova, strojnom prevođenju, traženju informacija i mnogim granama umjetne inteligencije. Također, semantička je sličnost kratkih tekstova danas posebno važna jer su takvi tipovi teksta u širokoj upotrebi u obliku upita i rezultata pretraživača, naslova vijesti, komentara na društvenim mrežama i slično. Sama semantička sličnost ima bitnu ulogu u postupcima automatske obrade tekstova te se najčešće koristi kako bi se izračunala sličnost između pojedinih tekstova ili rečenica.

2.2. Određivanje semantičke sličnosti

Semantička sličnost predstavlja koncept dodjeljivanja mjere skupovima riječi koje se zasnivaju na sličnosti njihovog značenja.[2] Iz tog razloga semantička sličnost ima bitnu ulogu kod kategorizacije i sumarizacije tekstova, kao i kod pronalaženja informacija. Također, semantičko uspoređivanje ima vrlo veliki značaj i kod kratkih tekstova, jer se oni redovito koriste kao upiti kod web tražilica.

Jedan od pristupa koji se koristi za određivanje semantičke sličnosti tekstova temelji se na korištenju korpusa tekstova. Taj se pristup zasniva na hipotezi da se riječi sa sličnim značenjem pojavljuju u sličnim kontekstima [9]. Prema tome može se kreirati prostor u kojemu je za svaku riječ koja se nalazi u korpusu označeno koliko se puta pojavila. Odnosno, za postupak određivanja semantičke sličnosti koriste se vektorski prostori da bi se izrazila korelacija riječi dobivenih iz određenog korpusa. U takvom je prostoru semantička sličnost između riječi izražena kao relacija

[2] (Vuk Batanović, 2018)

između tih vektora. Također, semantička se sličnost dokumenata određuje na sličan način s time da su dokumenti predstavljeni vektorima.

Za određivanje semantičke sličnosti razvijeni su algoritmi koji se mogu podijeliti u dvije velike skupine, algoritmi za leksičku sličnost i algoritmi za semantičku sličnost [8]. Kako bi se odredila leksička sličnosti između riječi moguće je koristiti algoritam LCS i njegove modificirane verzije, a to su MCLSC1 koja radi na principu da se traži najduža sekvenca poklapanja dužeg i kraćeg niz počevši od prvog znaka i MCLSCN koji radi na istom principu, ali je početak mogući sa bilo koje pozicije.

Kad govorimo o semantičkoj sličnosti, za određivanje se koristi metoda pomoću baze sinonima-rječnika koja mjeru sličnosti računa pomoću operacije prolaska kroz graf povezanosti. Osim ove metode, tu je i metoda koja se bazira na algoritmima koji mjeru semantičke sličnosti uče iz velikih zbirka tekstova, odnosno korpusa.

Algoritmi koji se baziraju na učenju iz korpusa od pročišćenog korpusa pokušavaju napraviti semantički prostor na osnovu distribucije riječi u korpusu. U navedenom prostoru svaka od riječi ima svoj vektor, a relacija između tih vektora predstavlja semantičku sličnost između riječi.

2.3. Korpusi

Postoji mnogo definicija pomoću kojih je moguće opisati korpuse, no najjednostavnije rečeno to je zbirka tekstova prirodnog jezika koja je sastavljena po određenom kriteriju [10].

Korpusi se koriste kod izrada rječnika, lingvističkih i govornih istraživanja, prevođenja, te učenja stranih jezika. Po svojoj definiciji, svaki korpus morao bi biti autentičan, relativno velik, kvalitetan i jednostavan, te dokumentiran. Postoji više različitih vrsta korpusa kao što su:

- Tekstualni i govorni
- Opći i specijalizirani
- Cjeloviti i djelomični
- Statički i promjenjivi
- Označeni i neoznačeni
- Jednojezični i višejezični

Računalni korpusi su kodirani i standardizirani, te optimizirani za pretragu i analizu, a nalaze se pohranjeni u računalnim bazama. Najčešće se sastoje od izrazito velikog broja riječi iz različitih jezičnih i društvenih izvora. Korpusi su izrazito važni kad je u pitanju semantika jer pružaju mnogo podataka o značenjima riječi.

3. PRISTUPI ZA RAČUNANJE SEMANTIČKE SLIČNOSTI

3.1. Algoritmi za određivanje semantičke sličnosti

Postoji mnogo pristupa za određivanje semantičke sličnosti tekstova, no neki od najčešće korištenih temelje se na sljedećim mjerama:

- Uzajamna zajednička informacija (PMI, eng. Pointwise Mutual Information)
- Latentna semantička analiza (LSA, eng. Latent Semantic Analysis)
- Leacock & Chodorow
- Lesk
- Wu & Palmer
- Resnik
- Lin
- Jiang & Conrath

PMI mjera kao korpus koristi sadržaj iz indeksa pretraživača AltaVista, dok mjera LSA za svoj korpus koristi Britanski nacionalni korpus. Ove mjere koriste pristup koji uspoređuje tekstove i sprječava uparivanje riječi koje ne pripadaju istim vrstama. Osim ove dvije navedene, postoje i druge mjere kao što je SOC-PMI koja se također zasniva na Britanskom korpusu, ali koristi leksičku sličnost riječi. Tu je još i model DM (distribuirana memorija) koji se zasniva na tenzorskom računu (generalizacija skalara i vektora) i kao korpus koristi Wikipediju, te Britanski nacionalni korpus. Također imamo i dosta jednostavniji pristup nazvan SDS koji se zasniva na primjeni distribucijske semantike, te također koristi Britanski nacionalni korpus.

Sljedeća tablica predstavlja navedene mjere, te njihovu točno i preciznost kod određivanja semantičke sličnost.

MJERA	TOČNOST	PRECIZNOST
PMI	69,90%	70,20%
LSA	68,40%	69,70%
SOC-PMI	72,64%	74,70%
DM	73,51%	/
SDS	73,04%	/

Tablica 1: Algoritmi za određivanje semantičke sličnosti [12]

3.2. Mjere bazirane na korpusima

3.2.1. Uzajamna zajednička informacija (Pointwise mutual information)

Pointwise mutual information je mjera koju predlaže Turney 2001. godine [17]. Koristi podatke prikupljene povratnim informacijama (PMI-IR) kao mjeru za procjenu semantičke sličnosti riječi. Temelji se na zajedničkom pojavljivanju riječi korištenjem brojanja pomoću velikih korpusa.
Ako su dane dvije riječi $w1$ i $w2$ njihov se PMI-IR mjeri kao:

$$PMI - IR(w1, w2) = log2\, p(w1\&w2)\, p(w1) * p(w2) \quad (2)$$

što pokazuje da se stupanj statičke ovisnosti između $w1$ i $w2$ može koristiti i kao mjera semantičke sličnosti između $w1$ i $w2$. Turney je također predložio i 4 različite vrste upita od kojih je nabolje prihvaćen NEAR upit (zajedničko pojavljivanje unutar prozora od deset riječi) koji predstavlja ravnotežu između učinkovitosti i točnosti.
Sljedeći upit predstavlja korištenje NEAR-a na primjeru prikupljanja računa iz AltaVista tražilice:

$$pNEAR(w1\&w2) \,/\, hits(w1\ NEAR\ w2)$$

Kada se $p\ (wi)$ aproksimira sa $hits(w1)/WebSize$ dobije se sljedeća PMI-IR mjera:

$$log2\ hits(w1\ AND\ w2) * WebSize\ hits(w1) * hits(w2)$$

U nizu eksperimenta koji se temelje na TOEFL testovima sinonima, mjera PMI-IR je pomoću operatora NEAR precizno identificirala točan odgovor u 72,5% slučajeva.

3.2.2. Latentna semantička analiza (Latent semantic analysis)

Još jedna mjera semantičke sličnosti bazirana na korpusu je latentna semantička analiza (LSA) koju je 1998. godine predložio Landauer [18]. U LSA je pojam zajedničkog pojavljivanja u korpusu zatvoren na principu smanjenja dimenzionalnosti kojom upravlja jednostruka dekompozicija vrijednosti (SVD) na matrici T koja predstavlja korpus. SVD je operacija u

linearnoj algebri koja se može primijeniti na bilo koju pravokutnu matricu kako bi se pronašle korelacije među redovima i stupcima.

LSA se može promatrati i kao način za savladavanje nedostatka modela vektorskog prostora. Zapravo, LSA sličnost se računa u donjem dimenzionalnom prostoru u kojem se iskorištavaju odnosi drugog reda između pojmova i tekstova. Tada se sličnost dobivena u vektorskom prostoru mjeri sa standardnom sličnosti kosinusa. Isto tako, LSA daje model vektorskog prostora koji omogućuje homogenu reprezentaciju i usporedbu riječi, skupova riječi i tekstova. U praksi je svaki tekst zastupljen u prostoru LSA sumiranjem normaliziranih LSA vektora od svih konstruktivnih riječi, koristeći i tf-idf težine.

3.3. Mjere bazirane na znanju

Postoji niz mjera koje su razvijene kako bi se odredio stupanj do kojeg su dvije riječi semantički povezane korištenjem informacija izvučenih iz semantičkih mreža [13]. U radu su predstavljene i opisane mjere za koje je utvrđeno da dobro funkcioniraju na hijerarhiji WordNeta. Sve te mjere kao ulaz pretpostavljaju par pojmova, te vraćaju vrijednost koja ukazuje na njihovu semantičku povezanost. Sljedećih šest mjera baziranih na znanju odabrano je iz razloga što imaju zapažen učinak u aplikacijama za obradu jezika, te zbog relativno visoke računalne učinkovitosti. U ovom je djelu rada opisano sljedećih šest mjera baziranih na znanju:

1. Leacock & Chodorow
2. Lesk
3. Wu & Palmer
4. Resnik
5. Lin
6. Jiang & Conrath

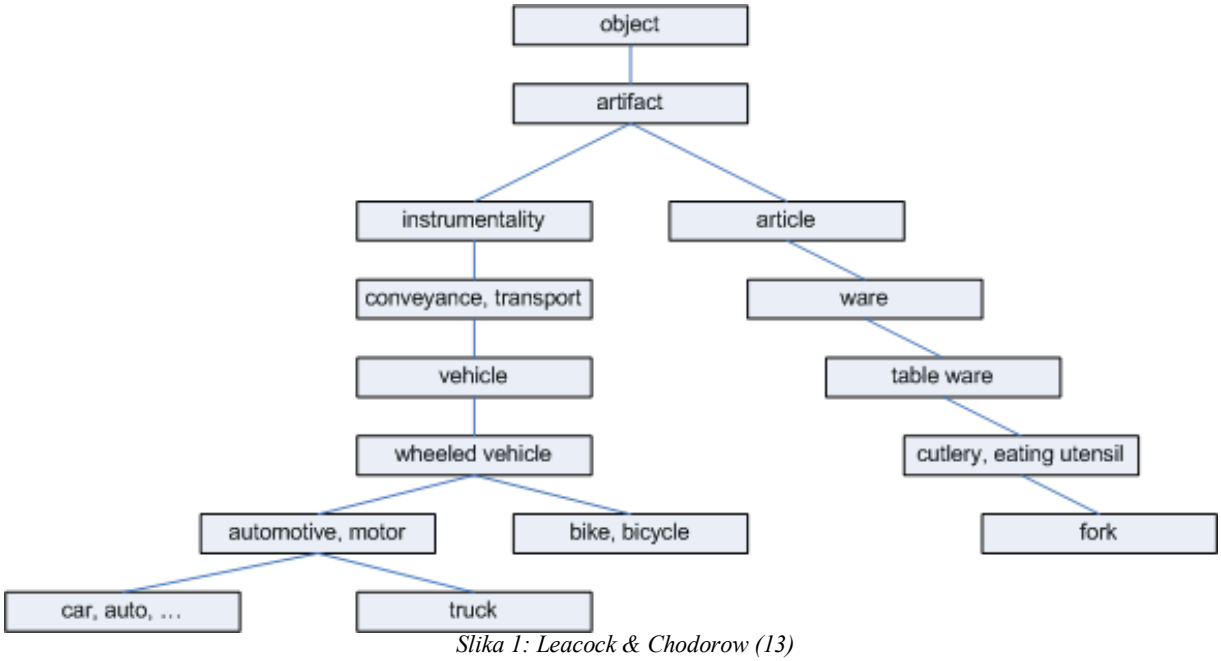
Sve navedene mjere definirane su između pojmova. U nastavku je dan kratak opis za svaku od navedenih šest mjera.

3.3.1. Leacock & Chodorow

Leacock & Chodorow metoda sličnost definirana je kao:

$$Sim = -log\ length\ 2 * D$$

gdje length predstavlja duljinu najkraćeg puta između dva koncepta koristeći brojanje čvorova, a D je maksimalna dubina taksonomije.



Slika 1: Leacock & Chodorow (13)

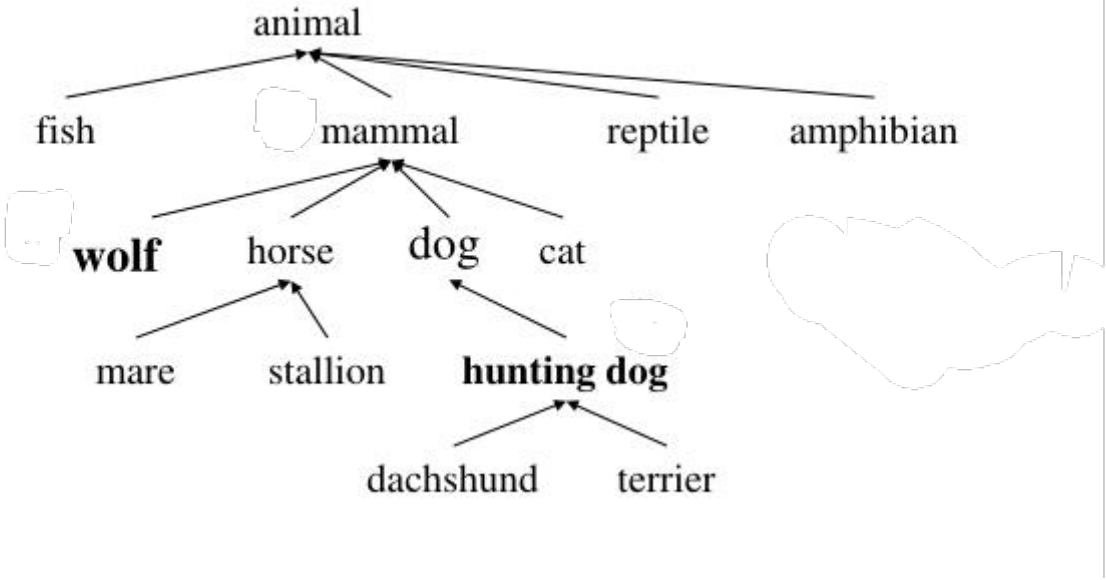
3.3.2. Lesk

Lesk metoda sličnosti između dva pojma se definira kao funkcija preklapanja između odgovarajućih definicija, kao što je navedeno u rječniku. Temelji se na algoritmu kojeg je 1986.g. predložio Lesk. Primjena Lesk mjere sličnosti nije ograničena na semantičke mreže, te se može koristiti zajedno sa bilo kojim rječnikom koji daje definicije riječi.

3.3.3. Wu i Palmer

Metoda sličnosti Wu i Palmer mjeri dubinu dva određena pojma u WordNet taksonomiji i dubinu najnižeg zajedničkog pretka (LCS), te kombinira navedene vrijednosti u sličnosti:

$$Sim = 2 * depth(LCS) depth(concept1) + depth(concept2)$$



Slika 2: Wu & Palmer [14]

3.3.4. Resnik

Mjera koju je 1995.god. uveo Resnik vraća informacijski sadržaj (IC) LCS od dva pojma:

$$Sim = IC(LCS)$$

gdje je IC definiran kao:

$$IC(c) = -log\ P(c)$$

i P (c) je vjerojatnost da se susreće sa primjerom pojma c u velikom korpusu.

Ova mjera koristi sadržaj informacija zajedničkih roditelja. Načelo mjere Resnik je sljedeće: dva pojma su sličnija ako predstavljaju više zajedničkih informacija, a informacije koje dijele dva koncepta C1 i C2 označene su informacijskim sadržajem pojmova.

Resnik mjera formalno je definirana kao:

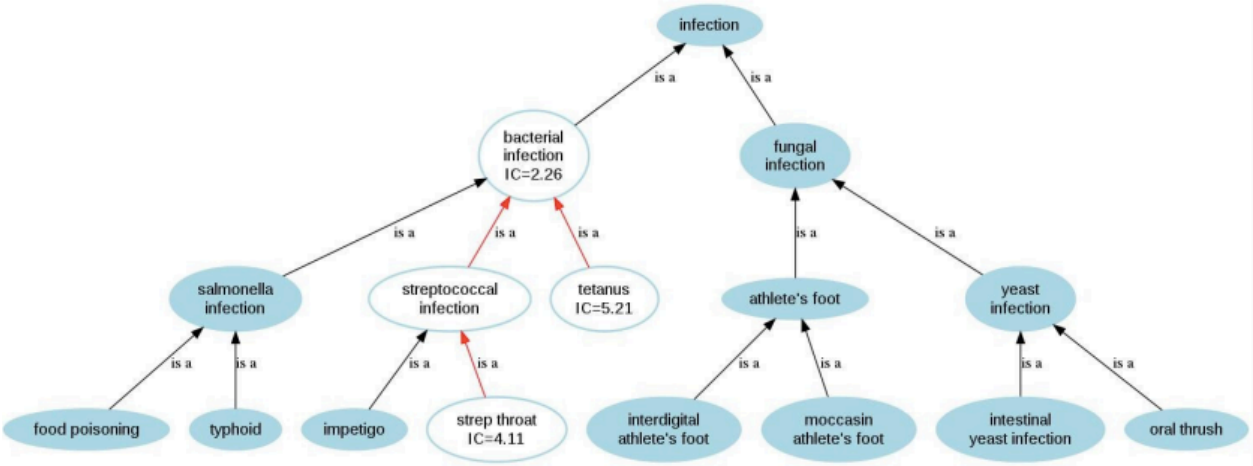
$$sim\ (C1, C2) = -ln(p_{mis}(C1, C2))$$

Ova mjera pruža nam podatke poput veličine korpusa, no smatra se pomalo grubom jer mnogi različiti parovi koncepata mogu dijeliti istog najmanjeg zajedničkog pretka (LCS).

3.3.5. Lin

Sljedeća mjera koja ću opisati je mjera koju je 1998.god. uveo Lin, koja se temelji na Resnikovoj mjeri sličnosti i dodaje faktor normalizacije koji se sastoji od informacijskog sadržaja dvaju ulaznih pojmova.

$$Sim = 2 * IC(LCS)\, IC(concept1) + IC(concept2)$$



Slika 3: Lin [15]

3.3.6. Jiang & Conrath

Konačno, posljednja mjera je razvijena 1997.god. i naziva se Jiang&Conrath :

$$Sim = 1\, IC(concept1) + IC(concept2) - 2 * IC(LCS)$$

Svi navedene mjere sličnosti su normalizirane tako da njihovi rezultati padaju u raspon od 0 do 1. Normalizacija se izvodi dijeljenjem rezultata sličnosti sa maksimalnim mogućim rezultatom za tu mjeru.

3.4. Metoda bazirana na grafu

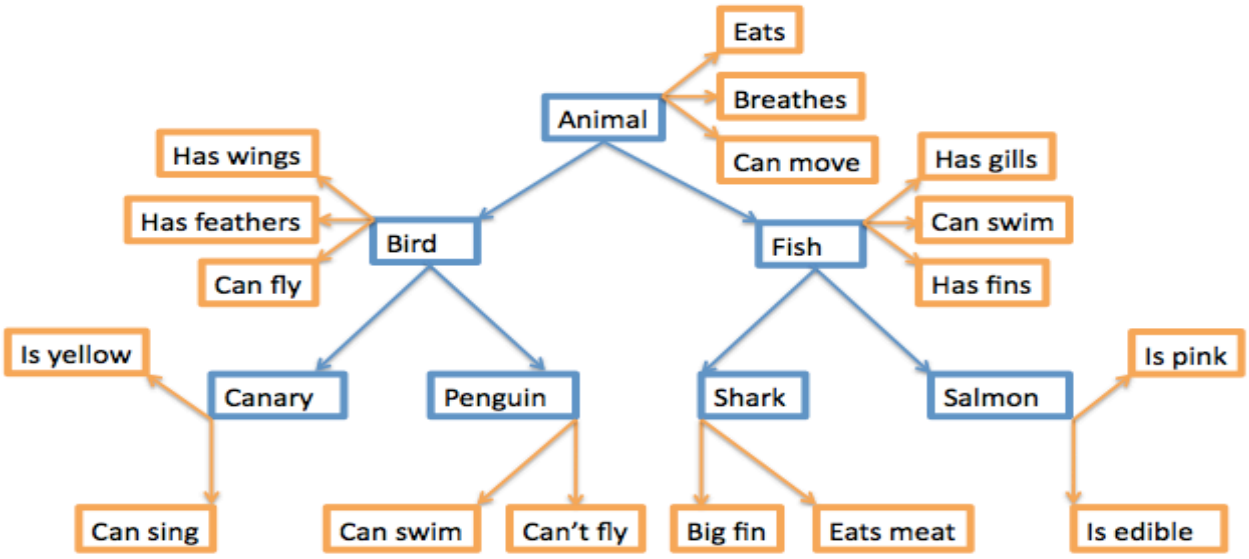
U ovom djelu rada opisati ću postupak po kojem će se za dvije dane riječi $w1$ i $w2$ odrediti njihova semantička sličnost $s(w1, w2)$ temeljena na grafu.

To možemo napraviti analizom leksičke baze znanja (u našem konkretnom slučaju to je WordNet) zato što su u njemu riječi organizirane u skupove sinonima, te pokazuju odnose prema drugim riječima. Metoda koja se koristi za izračun sličnosti bazira se na pronalasku minimalne dužine puta koji povezuje dvije riječi. No, sličnost između riječi ne određuje se samo dužinom puta, već i dubinom.

Prema navedenom, sličnost $s(w1, w2)$ između riječi $w1$ i $w2$ je funkcija duljine i dubine puta koja glasi:

$$s(w1, s2) = f(l, h)$$

gdje je l duljina najkraćeg puta između $w1$ i $w2$, a h dubina od prethodnika u hijerarhijskim semantičkim mrežama.



Slika 4: Primjer semantičke hijerarhijske mreže [11]

Za semantičku hijerarhijsku mrežu duljina puta između dvije riječi $w1$ i $w2$ može se odrediti prema sljedećima slučajevima:

1. $w1$ i $w2$ su u istom skupu sličnih riječi
2. $w1$ i $w2$ nisu u istom skupu sličnih riječi, ali njihovi skupovi sličnih riječi sadrže jednu ili više zajedničkih riječi
3. $w1$ i w2 nisu u istom skupu riječi, te njihovi skupovi ne sadrže zajedničke riječi

Prvi slučaj pokazuje da riječi $w1$ i $w2$ imaju isto značenje, te u tom slučaju semantička duljina puta iznosi 0. U drugom slučaju riječi $w1$ i $w2$ pokazuju djelomično iste značajke i semantička duljina puta između navedenih riječi iznosi 1. Za treći slučaj potrebno je računati stvarnu dužinu puta između $w1$ i $w2$, čija je vrijednost prema ograničenjima u rasponu od 0 do 1.

3.5. Levenshtein algoritam za određivanje sličnosti

Levenshtein algoritam radi na principu da računa najmanji broj promjena koje je potrebno napraviti kako bi se jedan niz znakova pretvorio u drugi. Računanje Levenshteinovog algoritma najčešće se radi korištenjem dinamičkog programiranja koje složenijim problemima pristupa na način da ih podjeli na manje.

Za računanje udaljenosti među riječima koristi matricu $[m, n]$, u kojoj m predstavlja broj slova prve riječi, a n broj slova druge riječi.

		m	e	i	l	e	n	s	t	e	i	n
	0	1	2	3	4	5	6	7	8	9	10	11
l	1	1	2	3	3	4	5	6	7	8	9	10
e	2	2	1	2	3	3	4	5	6	7	8	9
v	3	3	2	2	3	4	4	5	6	7	8	9
e	4	4	3	3	3	3	4	5	6	6	7	8
n	5	5	4	4	4	4	3	4	5	6	7	7
s	6	6	5	5	5	5	4	3	4	5	6	7
h	7	7	6	6	6	6	5	4	4	5	6	7
t	8	8	7	7	7	7	6	5	4	5	6	7
e	9	9	8	8	8	7	7	6	5	4	5	6
i	10	10	9	8	9	8	8	7	6	5	4	5
n	11	11	10	9	9	9	8	8	7	6	5	4

Slika 5: Primjer Levenshtein algoritma (15)

Primjer rada Levensthein algoritma prikazan je pomoću riječi «lijek» i «riječ» gdje je udaljenost između dvije navede riječi jednaka tri, odnosno potrebne su tri promjene da bi se riječ «lijek» pretvorila u riječ «riječi» [16].

1. Promjena:
   lijek → rijek

U prvoj promjeni obavljena je zamjena početnog slova «l» sa slovom «r».

2. Promjena
   rijek → riječ

Drugi korak promjene je zamjena posljednjeg slova «k» sa slovom «č».

3. Promjena:
   riječ → riječi

U posljednjem koraku dodano je slovo «i» na kraj, te je time transformacija u potpunosti obavljena.

3.6. Ostale mjere sličnosti

U ovom poglavlju razmotrit ćemo još neke mjere sličnosti. Neke od najčešće korištenih mjera za određivanje sličnosti su :

1. Kosinus mjera sličnosti
2. Jacard mjera sličnosti – Jaccardov koeficijent
3. Euklidova udaljenost

3.6.1. Kosinus mjera sličnosti

Kada su dokumenti predstavljeni kao vektori pojmova, tada sličnost između dva dokumenta odgovara korelaciji između dva vektora. To se izmjeri kao kosinus kuta između vektora, te se zato naziva kosinusna sličnost. Kosinus mjera sličnosti jedna je od najpopularnijih koja se primjenjuje na tekstualne dokumente.

Ako su dana dva dokumenta $\vec{a}$ i $\vec{b}$ tada je kosinus mjera sličnosti jednaka:

$$SIM\ (\vec{a}\ ,\vec{b}\ )\ = \vec{a} * \vec{b}\ /\ (\ |\ \vec{a}\ |\ |\ \vec{b}\ |\ )$$

Kosinusova sličnost između dva vektora (ili dva dokumenta u vektorskom prostoru) je mjera koja izračunava kosinus kuta između njih, te se često koristi kod uspoređivanja dokumenata u pretraživanju i obradi teksta. Ona se prikazuje pomoću skalarnog produkta dvaju vektora $\vec{a}$ i $\vec{b}$.

Postupak se odvija tako što se skalarni produkt vektora $(a * b)$ dijeli sa Euklidovom udaljenosti između tih dvaju vektora.
Kao rezultat izračunavanja kosinusne sličnosti dobiva se pozitivna vrijednost u rasponu od 0 do 1.

Važna karakteristika kosinusne sličnosti je neovisnost o dužini dokumenta. Na primjer, ako kombiniramo dvije identične kopije dokumenta kako bi se dobio novi dokument D, tada je kosinusna sličnost između bilo navedenih dokumenta i dokumenta D jednaka 1, odnosno oni se smatraju identičnima.

3.6.2. Jacard mjera sličnosti

Jaccardov koeficijent, poznat i kao Tanimoto mjera, dokumente gleda kao skupove. Radi na principu da dijeli veličinu skupa presjeka sa veličinom skupa unije dvaju skupova. Domena ove mjere je [0,1], što znači da koeficijent vraća vrijednost 0 ako dva skupa nemaju presjek, odnosno 1 ako su skupovi identični.

Jaccardov koeficijent se kod računanja sličnosti tekstualnih dokumenta definira kao rezultat djeljenja broja zajednički elemenata sa brojem svih elemenata.Jednadžba za izračunavanje glasi:

$$SIM(A, B) = |A \cap B| / |A \cup B|$$

3.6.3. Euklidova udaljenost

Euklidova udaljenost predstavlja standardnu metriku za geometrijske probleme, odnosno definira se kao najkraći razmak između dvije točke u jednom prostoru. Mjerenje udaljenosti između tekstualnih dokumenta pomoću Euklidove udaljenost računa se na način da su dva tekstualna dokumenta d_a i d_b predstavljena pomoću vektorskih udaljenosti , te se sličnost računa pomoću jednadžbe :

$$D_E\left(\overrightarrow{t_a}, \overrightarrow{t_b}\right) = \left(\sum_{t=1}^{m} \left|w_{t,a} - w_{t,b}\right|^2\right)^{1/2}$$

Gdje su $\overrightarrow{t_a}$ i $\overrightarrow{t_b}$ vektori dokumenta d_a i d_b, te se koristi tf-idf mjera:

$$w_{t,a} = tfidf(d_a, t)$$

u kojoj t predstavlja vektor dokumenta.

4. IMPLEMENTACIJA MODULA ZA RAČUNANJE SEMANTIČKE SLIČNOSTI U PYTHONU

4.1. Izračun sličnosti između tekstualnih dokumenta

Kako bi odredili sličnost između dva ili više tekstualnih dokumenta u Pythonu koristiti ćemo NLTK platformu. NLTK je implementiran kako bi se pojednostavila obrada prirodnog jezika u Pythonu. Nudi jednostavno sučelje za korištenje, te korpuse i leksičke izvore kao što je WordNet. Uz to, nudi i paket biblioteka za obradu teksta koji omogućuje klasifikaciju, označavanje, tokenizaciju i rasčlanjivanje. Organiziran je u vidu modula od kojih su najvažniji nltk.corpus, nltk.tokenize, nltk.tag, nltk.classify, te mnogi drugi.

Dokument se gleda kao vektor u kojemu vrijednost svake dimenzije odgovara broju pojavljivanja tog pojma u dokumentu. U programu je također korištena kosinusova mjera sličnosti kao tehnika pomoću koje će se dobiti rezultat sličnosti dokumenta.

Prvi korak u izradi programa je uvoz svih relevantnih paketa. Otvaramo datoteku, pročitamo sve retke i riječi, te ih pozovemo. Također, obavlja se i konvertiranje riječi u mala slova.

Nakon toga koristi se Porter Stemmer da bi izvukli osnove riječi i uklonili njihove nastavke. Stemming je proces koji se koristi u morfologiji i pronalaženju informacija na način da se različiti oblici riječi ili izvedenice riječi pretvaraju u njihov osnovni ili korijenski oblik. Na primjer, riječ «dogs» se pretvara u korijen te riječi koji glasi «dog».

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import numpy as np
import nltk

def funkcija(file):
    raw = open(file).read()
    tokeni = word_tokenize(raw)
    words = [w.lower() for w in tokeni]

    porter = nltk.PorterStemmer()
    istaknuti_tokeni = [porter.stem(t) for t in words]

    stop_rijeci = set(stopwords.words('english'))
    filtrirani_tokeni = [w for w in istaknuti_tokeni if not w in stop_rijeci]

    count = nltk.defaultdict(int)
    for word in filtrirani_tokeni:
        count[word] += 1
    return count;
```

Proces u kojem se podaci pretvaraju u nešto što računalo može razumjeti zove se priprema podataka, a jedan od takvih primjera je i filtriranje beskorisnih podataka.

Sljedeći korak je uklanjanje takozvanih «stop riječi» odnosno riječi koje nemaju svrhu u obradi prirodnog jezika. To su riječi koje ne nose nikakvo značenje za sadržaj teksta, kao što su «the», «a», «an», «in» i slične.

Nakon toga računamo pojavu svake riječi u dokumentu.

Kada su obavljeni dosad navedeni koraci, slijedi postupak izračuna sličnosti dokumenta pomoću kosinusove sličnosti (Cosine Similarity) koja je opisana u poglavlju 4.6.1. Kosinusova sličnost između dva vektora(dokumenta) predstavlja mjeru koja izračunava kosinus kuta između njih, te se koristi kod uspoređivanja dokumenata i obradi teksta. Računa se na način da se skalarni produkt vektora dijeli sa Euklidovom udaljenosti između tih dvaju vektora.

```
def kosinus_slicnost(a,b):
    skalarni_produkt = np.dot(a, b)
    norm_a = np.linalg.norm(a)
    norm_b = np.linalg.norm(b)
    return skalarni_produkt / (norm_a * norm_b)
```

Sljedeći korak i posljednja funkcija u programu je funkcija za izračun sličnosti između dva tekstualna dokumenta. Implementacija navedene funkcije u Python programu prikazana je u sljedećem isječku koda.

```
def izracunslicnosti(tekst1, tekst2):
    all_words_list= []
    for key in tekst1:
        all_words_list.append(key)
    for key in tekst2:
        all_words_list.append(key)
    all_words_list_size = len(all_words_list)

    v1 = np.zeros(all_words_list_size, dtype=np.int)
    v2 = np.zeros(all_words_list_size, dtype=np.int)
    i = 0
    for (key) in all_words_list:
        v1[i] = tekst1.get(key, 0)
        v2[i] = tekst2.get(key, 0)
        i = i + 1
    return kosinus_slicnost(v1, v2);
```

Za kraj programa potrebno je ispisati rezultate navedenih sličnosti. Navedeni dio koda prikazuje kraj programa i način na koji je izveden ispis rezultata između tri tekstualna dokumenta : bug1.txt, bug2.txt i bug3.txt.

```
if __name__ == "__main__":
    tekst1 = funkcija ('/home/ivan/Desktop/bug1.txt')
    tekst2 = funkcija ('/home/ivan/Desktop/bug2.txt' )
    tekst3 = funkcija ('/home/ivan/Desktop/bug3.txt' )
    print ("Slicnost tekst 1 i tekst 2:  ", izracunslicnosti(tekst1,tekst2))
```

```
    print ("Slicnost tekst 1 i tekst 3:  ", izracunslicnosti(tekst1,tekst3))
    print ("Slicnost tekst 2 i tekst 3 ", izracunslicnosti(tekst2,tekst3)
```

U nastavku je prikazan kompletni kod za izračun sličnosti između tekstualnih dokumenta.

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import numpy as np
import nltk

def funkcija(file):
    raw = open(file).read()
    tokeni = word_tokenize(raw)
    words = [w.lower() for w in tokeni]

    porter = nltk.PorterStemmer()
    istaknuti_tokeni = [porter.stem(t) for t in words]

    stop_rijeci = set(stopwords.words('english'))
    filtrirani_tokeni = [w for w in istaknuti_tokeni if not w in stop_rijeci]

    count = nltk.defaultdict(int)
    for word in filtrirani_tokeni:
        count[word] += 1
    return count;

def kosinus_slicnost(a,b):
    skalarni_produkt = np.dot(a, b)
    norm_a = np.linalg.norm(a)
    norm_b = np.linalg.norm(b)
    return skalarni_produkt / (norm_a * norm_b)

def izracunslicnosti(tekst1, tekst2):
    all_words_list= []
    for key in tekst1:
        all_words_list.append(key)
    for key in tekst2:
        all_words_list.append(key)
    all_words_list_size = len(all_words_list)

    v1 = np.zeros(all_words_list_size, dtype=np.int)
    v2 = np.zeros(all_words_list_size, dtype=np.int)
    i = 0
    for (key) in all_words_list:
        v1[i] = tekst1.get(key, 0)
        v2[i] = tekst2.get(key, 0)
        i = i + 1
    return kosinus_slicnost(v1, v2);

if __name__ == "__main__":
    tekst1 = funkcija ('/home/ivan/Desktop/bug1.txt')
    tekst2 = funkcija ('/home/ivan/Desktop/bug2.txt' )
    tekst3 = funkcija ('/home/ivan/Desktop/bug3.txt' )
    print ("Slicnost tekst 1 i tekst 2:  ", izracunslicnosti(tekst1,tekst2))
    print ("Slicnost tekst 1 i tekst 3:  ", izracunslicnosti(tekst1,tekst3))
    print ("Slicnost tekst 2 i tekst 3 ", izracunslicnosti(tekst2,tekst3))
```

Za primjer kod pokretanja koda odabrana su tri tekstualna dokumenta, dva gotovo potpuno jednaka (bug2.txt i bug3.txt), te jedan veoma različit od navedenih. Rezultati sličnosti tih dokumenta u skladu su sa stvarnom sličnosti, prema čemu se može zaključiti da je program ispravan, te obavlja svoju funkciju.

Rezultat pokretanja programa daje rezultate sličnost navedenih dokumenta u rasponu od 0 do 1.

```
ivan@LIFEBOOK-AH532:~$ python semslicnost.py
('Slicnost tekst 1 i tekst 2:  ', 0.18628620339432228)
('Slicnost tekst 1 i tekst 3:  ', 0.1988893210439325)
('Slicnost tekst 2 i tekst 3 ', 0.9667616297706062)
```

Slika 6: sličnost dokumenta - rezultati

4.2. Računanje semantičke sličnosti korištenjem Wordneta

Uobičajeno je da se kod obrade prirodnih jezika računa semantička sličnost rečenica, a za takve se stvari najčešće koristi WordNet. On predstavlja alat koji je od velike pomoći kada se radi sa tekstom, kao i kod rješavanja problema semantičke sličnosti.

WordNet je hijerarhijski organizirana leksička baza podataka za engleski jezik. Radi na principu da grupira engleske riječi na temelju sinonima koji se nazivaju synsets, te sadrži kratko pojašnjenje riječi i primjer njezina korištenja.

Synsets je skup sinonima, odnosno skup riječi koje su međusobno zamjenjive u nekom kontekstu bez promjene njegovog značenja.

Pri određivanju mjere semantičke sličnosti potrebno je da se zadovolje navedena tri svojstva:

1. $Sličnost\ (S1, S2) = Sličnost\ (S2, S1)$ – sličnost između prvog i drugog teksta mora biti jednaka sličnosti između drugog i prvog teksta.

2. $Sličnost\ (S, S) = 1$ – sličnost između dvije identične rečenice je maksimalna, odnosno ona iznosi 1.

3. Kada imamo 3 identične rečenice koje se razlikuju u jednoj riječi, tada se dvije rečenice kod kojih je ta riječ slična smatraju najsličnijima

Za posljednju stavku, kako bi odredili koje su dvije riječi sličnije koristi ćemo WordNet, te pomoću synsetsa odrediti najsličnije riječi.

Također, treba napomenuti da u Pythonu kod upita za ispis sličnih riječi one su poredane po tome koliko se često pojedini pojam pojavljuje u korpusu.

4.2.1. Implementacija funkcije za računanje sličnosti riječi primjenom Wordneta

Sljedeći isječak koda programskog jezika Python predstavlja kod koji će ispisivati najčešće sinonime pojedinih riječi.

```
from nltk.corpus import wordnet as wn

print = wn.synsets('cat', 'n')
print = wn.synsets('dog', 'n')
print = wn.synsets('feline', 'n')
print = wn.synsets('cat', 'n')
print = wn.synsets('animal', 'n')
print = wn.synsets('man', 'n')
```

Kao rezultat pokretanja navedenog programa dobije se ispis sličnih riječi za sve navedene.

```
ivan@LIFEBOOK-AH532:~$ python dipp.py
[Synset('cat.n.01'), Synset('guy.n.01'), Synset('cat.n.03'), Synset('kat.n.01'), Syns
et('cat-o'-nine-tails.n.01'), Synset('caterpillar.n.02'), Synset('big_cat.n.01'), Syn
set('computerized_tomography.n.01')]
[Synset('dog.n.01'), Synset('frump.n.01'), Synset('dog.n.03'), Synset('cad.n.01'), Sy
nset('frank.n.02'), Synset('pawl.n.01'), Synset('andiron.n.01')]
[Synset('feline.n.01')]
[Synset('cat.n.01'), Synset('guy.n.01'), Synset('cat.n.03'), Synset('kat.n.01'), Syns
et('cat-o'-nine-tails.n.01'), Synset('caterpillar.n.02'), Synset('big_cat.n.01'), Syn
set('computerized_tomography.n.01')]
[Synset('animal.n.01')]
[Synset('man.n.01'), Synset('serviceman.n.01'), Synset('man.n.03'), Synset('homo.n.02
'), Synset('man.n.05'), Synset('man.n.06'), Synset('valet.n.01'), Synset('man.n.08'),
 Synset('man.n.09'), Synset('man.n.10'), Synset('world.n.08')]
ivan@LIFEBOOK-AH532:~$
```

Slika 7: Rezultati ispisa sinonima

Nakon što smo u prvom programu prikazali ispis sličnih riječi za svaku pojedinu riječ, sljedeći kod prikazuje kako se određuje sličnost između navedenih riječi. Za primjer je odabrana riječ «cat» koju ćemo uspoređivati sa svakom od prije navedenih riječi.

```
from nltk.corpus import wordnet as wn
print = wn.synsets('cat', 'n')
print = wn.synsets('dog', 'n')
print = wn.synsets('feline', 'n')
print = wn.synsets('cat', 'n')
print = wn.synsets('animal', 'n')
print = wn.synsets('man', 'n')

cat = wn.synsets ('cat', 'n') [0]
print cat.lemmas()[0].count()


dog = wn.synsets ('dog', 'n') [0]
feline = wn.synsets ('feline', 'n') [0]
```

```
mammal = wn.synsets ('mammal', 'n') [0]
cat = wn.synsets ('cat', 'n') [0]
anima = wn.synsets ('animal', 'n') [0]
man = wn.synsets ('man', 'n') [0]

for synset in [dog, feline, mammal, cat, animal, man]:
        print     "Slicnost(%s,     %s)     =     %s"     %     (cat,     synset,
cat.wup_similarity(synset))
```

U prvom dijelu koda prikazuje se koliko je česta lema, koja predstavlja osnovnu riječ u rječniku ili korpusu.

Nakon toga određuju se slične riječi za svaku od navedenih riječi, te se na kraju uspoređuje sličnost svake od tih riječi sa onom koju smo odabrali i ispisuju se rezultati tih sličnosti.

Rezultat pokretanja programa ispisuje kolika je sličnost između riječi «cat» i svake od navedenih.

```
ivan@LIFEBOOK-AH532:~$ python dip.py
18
Slicnost(Synset('cat.n.01'), Synset('dog.n.01')) = 0.857142857143
Slicnost(Synset('cat.n.01'), Synset('feline.n.01')) = 0.962962962963
Slicnost(Synset('cat.n.01'), Synset('mammal.n.01')) = 0.833333333333
Slicnost(Synset('cat.n.01'), Synset('cat.n.01')) = 1.0
Slicnost(Synset('cat.n.01'), Synset('animal.n.01')) = 0.666666666667
Slicnost(Synset('cat.n.01'), Synset('man.n.01')) = 0.521739130435
ivan@LIFEBOOK-AH532:~$
```

Slika 8: Rezultati sličnosti među riječima

U nastavku je prikazan cijeli programski kod u Pythonu za izračun sličnosti među riječima:

```
from nltk.corpus import wordnet as wn

cat = wn.synsets('cat', 'n')[0]
print cat.lemmas()[0].count()

dog = wn.synsets('dog', 'n')[0]
feline = wn.synsets('feline', 'n')[0]
mammal = wn.synsets('mammal', 'n')[0]
cat = wn.synsets('cat', 'n')[0]
animal = wn.synsets('animal', 'n')[0]
man = wn.synsets('man', 'n')[0]

for synset in [dog, feline, mammal, cat, animal, man]:
        print     "Slicnost(%s,     %s)     =     %s"     %     (cat,     synset,
cat.wup_similarity(synset))
```

4.2.2. Implementacija funkcije za računanje sličnosti tekstova primjenom Wordneta

Analiziranje kratkih tekstova u današnje je vrijeme tema velike većine istraživanja i projekata. Nekada se ovakav tip problema rješavao pristupom računanja leksičke sličnosti među

riječima, dok se u današnje doba sve više algoritma bazira na upotrebi semantičke sličnosti. Cilj kod semantičke sličnosti tekstova je, s obzirom na unos dva teksta, generirati rezultat koji pokazuje njihovu sličnost na semantičkoj razini.

Nakon što smo izradili program koji računa sličnost riječi, prikazati ću i kako se pomoću programskoj jezika Python računa sličnost između rečenica, odnosno dijelova teksta.

Na samom početku programa potrebno je, osim samog korpusa, uključiti tokenizaciju riječi i označavanje vrsta riječi (POS_tag). Tokenizacija predstavlja proces u kojemu se korpus dovodi u stanje u kojem su sve riječi identificirane i obilježene. Označavanje vrste riječi, odnosno Part-of-speech (POS) označavanje je pridruživanje gramatičkih kategorija svakoj riječi u tekstu.

U prvom dijelu programa izvršava se pretvorba POS oznaka u pojednostavljene WordNet oznake. POS oznake sadrže sljedeće vrste riječi: imenice (N), glagoli (V), pridjevi (J) i prilozi (R).

```
from nltk import word_tokenize, pos_tag
from nltk.corpus import wordnet as wn

def pretvorba_oznaka(tag):
    if tag.startswith('N'):
        return 'n'

    if tag.startswith('V'):
        return 'v'

    if tag.startswith('J'):
        return 'a'

    if tag.startswith('R'):
        return 'r'

    return None
```

Nakon što je prvi korak napravljen, korištenjem Wordneta se kreće u definiranje funkcije za sličnost tekstova:

```
def slicnost_tekstova(text1, text2):
    text1 = pos_tag(word_tokenize(text1))
    text2 = pos_tag(word_tokenize(text2))

    synset1 = [synset_oznake(*tagged_word) for tagged_word in text1]
    synset2 = [synset_oznake(*tagged_word) for tagged_word in text2]

    synset1 = [ss for ss in synset1 if ss]
    synset2 = [ss for ss in synset2 if ss]

    rezultat, brojac = 0.0, 0

    for synset in synset1:
        # izracunaj slicnost sa najslicnijom rijeci u drugoj recenici
        best_score = max([synset.path_similarity(ss) for ss in synset2])
```

```
        if best_score is not None:
            rezultat += best_score
            brojac += 1

    rezultat /= brojac
    return rezultat
```

Na samom kraju programa, upisujemo tekstove koje ćemo uspoređivati sa glavnim tekstom. Za primjer glavnog teksta uzeta je rečenica «Cats are cute» koju uspoređujemo sa navedenim dijelovima teksta:

- r1 = "Man is ugly, but strong"
- r2 = "The world is wonderful."
- r3 = "This house is empty."
- r4 = "Dogs are cute"
- r5 = "Cats are cute, but they are very bad."

```
sentences = [
    "Man is ugly, but  strong",
    "The world is wonderful.",
    "This house is empty.",
    "Dogs are cute",
    "Cats are cute, but they are very bad.",
]

glavna = "Cats are cute."
```

Završni korak je ispis rezultata sličnosti za navedene usporedbe.

```
for sentence in sentences:
    print   "Slicnost(\"%s\",   \"%s\")   =   %s"   %   (glavna,   sentence,
slicnost_tekstova(glavna, sentence))
    print   "Slicnost(\"%s\",   \"%s\")   =   %s"   %   (sentence,   glavna,
slicnost_tekstova(sentence, glavna))
    print
```

Rezultat pokretanja programa za izračun semantičke sličnosti između tekstova daje sljedeće rezultate:

```
ivan@LIFEBOOK-AH532:~$ python dip4.py
Slicnost("Cats are cute.", "Man is ugly, but  strong") = 0.541666666667
Slicnost("Man is ugly, but  strong", "Cats are cute.") = 0.541666666667

Slicnost("Cats are cute.", "The world is wonderful.") = 0.538461538462
Slicnost("The world is wonderful.", "Cats are cute.") = 0.538461538462

Slicnost("Cats are cute.", "This house is empty.") = 0.533333333333
Slicnost("This house is empty.", "Cats are cute.") = 0.533333333333

Slicnost("Cats are cute.", "Dogs are cute") = 0.733333333333
Slicnost("Dogs are cute", "Cats are cute.") = 0.733333333333

Slicnost("Cats are cute.", "Cats are cute, but they are very bad.") = 1.0
Slicnost("Cats are cute, but they are very bad.", "Cats are cute.") = 1.0

ivan@LIFEBOOK-AH532:~$
```

Slika 9: Rezultati sličnosti rečenica

4.3. Računanje semantičke sličnosti primjenom biblioteke FuzzyWuzzy

FuzzyWuzzy je Python biblioteka za uspoređivanje niza znakova a temelji se na Levenshteinovom algoritmu koji radi na principu da računa najmanji broj promjena koje je potrebno napraviti kako bi se jedan niz znakova pretvorio u drugi. Fuzzy uparivanje samo po sebi predstavlja traženje znakova koji su gotovo jednaki.

Da bi se izvršila usporedba tekstova pomoću biblioteke FuzzyWuzzy potrebno je koristit funkciju za uspoređivanje znakova fuzz.ratio:

$$>> \ fuzz.ratio\ (text1, text2)$$

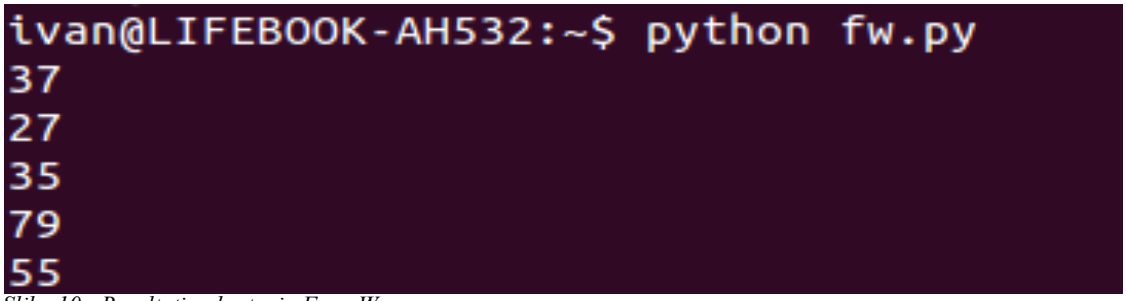
Za primjer rada programa uzeta je rečenica “Cats are cute” koju uspoređujemo sa sljedećim rečenicama:

- r1 = "Man is ugly, but strong"
- r2 = "The world is wonderful."
- r3 = "This house is empty."
- r4 = "Dogs are cute"
- r5 = "Cats are cute, but they are very bad."

U nastavku je prikazan kod koji izvršava usporedbu navedenih tekstova.

```
from fuzzywuzzy import fuzz
from fuzzywuzzy import process
print fuzz.ratio ("Cats are cute.", "Man is ugly, but strong.")
print fuzz.ratio ("Cats are cute.", "The world is wonderful.")
print fuzz.ratio ("Cats are cute.", "This house is empty.")
print fuzz.ratio ("Cats are cute.", "Dogs are cure.")
print fuzz.ratio ("Cats are cute.", "Cats are cute, but they are very bad.")
```

Na sljedećoj slici prikazani su rezultati sličnosti između rečenica koji su dobiveni pokretanjem prethodno navedenog koda. Sličnost tekstova izražena je u rasponu od 0 do 100, gdje 0 predstavlja potpunu suprotnost, a 100 identične tekstove.



Slika 10: Rezultati pokretanja FuzzyWuzzy

U sljedećoj tablici prikazana je usporedba rezultata sličnosti tekstova dobivena korištenjem Wordneta i Fuzzywuzzyja :

	WordNet (od 0 do 1)	FuzzyWuzzy (od 0 do 100)
Dio teksta	Glavna rečenica	
r1	0,54	37
r2	0,53	27
r2	0,53	35
r4	0,73	79
r5	1	55

Tablica 2: Usporedba rezultata

Rezultati dobiveni korištenjem oba pristupa u velikoj su mjeri veoma slični, dok se velika razlika primjećuje jedino kod rezultata r5. Tu su se uspoređivale rečenice “Cats are cute.” i "Cats are cute, but they are very bad.". Kod navedenih rečenica možemo primjetiti da druga rečenica u potpunosti sadrži cijeli sadržaj prve rečenice, te zbog toga WordNet to interpretira kao maksimalnu sličnost. S druge strane, korištenjem FuzzyWuzzya rezultat sličnosti je podosta manji iz razloga što navedeni pristup daje maksimalan rezultat jedino tekstovima koji su u potpunosti jednaki, što u ovom slučaju nisu. Taj pristup nastavak druge rečenice koji slijedi nakon zareza ("but they are very bad.") ne prepoznaje u sadržaju prve rečenice i zbog toga radi razliku među njima.

Moje je mišljenje ipak da su rezultati dobiveni korištenjem FuzzyWuzzy točniji, odnosno realniji, iako je to dosta subjektivno mišljenje i teško je dati ispravan odgovor što je točnije u ovom slučaju.

4.4. Usporedba pristupa za računanje sličnosti

4.4.1. Primjeri tekstova za usporedbu

U završnom dijelu rada na praktičnom primjenu prikazana je primjena programa za računanje semantičke sličnosti između tekstova. Za primjer je uzeto 7 tekstova dostupnih na Internetu, od kojih su neki vrlo slični te govore o istim stvarima, do neki koji su potpuno različiti i rezultat semantičke sličnosti između takvih tekstova ne bi trebao biti veliki.
Sljedeća tablica prikazuje i opisuje tekstove koji su uzeti za primjer istraživanja rezultata semantičke sličnosti, dok se u poglavlju «Dodaci» nalaze u svom cijelom sadržaju:

NAZIV DATOTEKE	SADRŽAJ	LINK
it1.txt	Informacijska tehnologija	https://searchdatacenter.techtarget.com/definition/IT
it3.txt	Informacijska tehnologija	https://en.wikipedia.org/wiki/Information_technology
it5.txt	Informacijska tehnologija	https://searchdatacenter.techtarget.com/definition/IT
croatia.txt	O Hrvatskoj	https://wikitravel.org/en/Croatia
croatia2.txt	O hrvatskoj	https://wikitravel.org/en/Croatia
dog.txt	O psima	https://simple.wikipedia.org/wiki/Dog
apple.txt	O korporaciji «Apple»	https://en.wikipedia.org/wiki/Apple_Inc
Lond1.txt	Grad London	https://www.jumeirah.com/en/destinations/london/about-london/
Lond2.txt	Grad London	https://www.jumeirah.com/en/destinations/london/about-london/
Lond3.txt	Grad London	https://simple.wikipedia.org/wiki/London

Tablica 3: Popis tekstualnih dokumenta

4.4.2. Rezultati semantičke sličnosti tekstova dobiveni primjenom Wordneta

Kako bi izvršili usporedbu navedenih tekstova i dobili vrijednosti koje opisuju rezultate njihove semantičke sličnosti potrebno je u ranije opisanom programu uvrstiti dokumente koje želimo uspoređivati, te pojedinačno ispisati rezultate sličnosti za svaki par dokumenta.

Za dobivanje rezultata semantičke sličnosti u kodu se koristi kosinusova mjera sličnosti.

Pokretanjem na ranije opisani način modificiranog koda u Terminalu nam se ispisuju navedeni rezultati u rasponu od 0 do 1 koji pokazuju semantičke sličnosti između danih tekstova:

```
('Slicnost IT1 i IT3:   ', 0.7599722072389079)
('Slicnost IT1 i IT5:   ', 0.96277470564597079)
('Slicnost IT1 i Croatia:    ', 0.68402046438943964)
('Slicnost IT1 i Croatia2:    ', 0.68402046438943964)
('Slicnost IT3 i IT5:   ', 0.84038512006270072)
('Slicnost IT3 i Croatia:   ', 0.64173083043186741)
('Slicnost IT3 i Croatia2:   ', 0.64173083043186741)
('Slicnost IT5 i Croatia:   ', 0.68419553681596035)
('Slicnost IT5 i Croatia2:   ', 0.68419553681596035)
('Slicnost Croatia i Croatia2:   ', 0.99999999999999978)
('Slicnost Apple i IT1:   ', 0.82974221636301171)
('Slicnost Apple i IT3:   ', 0.69939966514069496)
('Slicnost Apple i IT5:   ', 0.80014398349844196)
('Slicnost Apple i Croatia:   ', 0.70374232109852874)
('Slicnost Apple i Croatia2:   ', 0.70374232109852874)
('Slicnost Dog i IT1:   ', 0.13018891098082389)
('Slicnost Dog i IT3:   ', 0.28005601680560194)
('Slicnost Dog i IT5:   ', 0.22792115291927589)
('Slicnost Dog i Croatia:   ', 0.3511234415883917)
('Slicnost Dog i Croatia2:   ', 0.3511234415883917)
('Slicnost Dog i Apple:   ', 0.13639886789409469)
('Slicnost London1 i IT1:   ', 0.556221314792725 63)
('Slicnost London1 i IT3:   ', 0.5094818021126 6769)
('Slicnost London1 i IT5:   ', 0.5492308098543 5135)
('Slicnost London1 i Croatia:   ', 0.5476415514324 7517)
('Slicnost London1 i Croatia2:   ', 0.5476415514324 7517)
('Slicnost London1 i Apple:   ', 0.5722584330369 5079)
('Slicnost London1 i Dog:   ', 0.2294157338705 6174)
('Slicnost London1 i London2:   ', 0.9113223768657 6711)
('Slicnost London1 i London3:   ', 0.4675719001195 1805)
('Slicnost London2 i IT1:   ', 0.454219979166 1351)
('Slicnost London2 i IT3:   ', 0.463600445571 75339)
('Slicnost London2 i IT5:   ', 0.475971293846 98361)
('Slicnost London2 i Croatia:   ', 0.5111012519999 5193)
('Slicnost London2 i Croatia2:   ', 0.5111012519999 5193)
('Slicnost London2 i Apple:   ', 0.467316168256 78751)
('Slicnost London2 i Dog:   ', 0.3333333333333331)
('Slicnost London2 i London3:   ', 0.5468165968051 5482)
('Slicnost London3 i IT1:   ', 0.258598269552 36936)
('Slicnost London3 i IT3:   ', 0.342142879617 80465)
('Slicnost London3 i IT5:   ', 0.316145447190 50064)
('Slicnost London3 i Croatia:   ', 0.4181210050035 4538)
('Slicnost London3 i Croatia2:   ', 0.4181210050035 4538)
('Slicnost London3 i Apple:   ', 0.277735039447 22594)
('Slicnost London3 i Dog:   ', 0.424264068711 92851)
```

Slika 11: rezultati sličnosti – WordNet

Rezultati sličnosti između pojedinih dokumenta prikazani su u sljedećoj tablici :

	IT1	IT3	IT5	Cro	Cro2	Apple	Dog	Lon1	Lon2	Lon3
IT1	-	0,76	0,96	0,68	0,68	0,82	0,13	0,56	0,45	0,26
IT3	0,76	-	0,84	0,64	0,64	0,699	0,28	0,51	0,46	0,34
IT5	0,96	0,84	-	0,68	0,68	0,8	0,23	0,55	0,48	0,32
Cro	0,68	0,64	0,68	-	1	0,7	0,35	0,54	0,51	0,42
Cro2	0,68	0,64	0,68	1	-	0,7	0,35	0,54	0,51	0,42
Apple	0,82	0,699	0,8	0,7	0,7	-	0,13	0,57	0,47	0,28
Dog	0,13	0,28	0,23	0,35	0,35	0,13	-	0,23	0,33	0,42
Lon1	0,56	0,51	0,55	0,54	0,54	0,57	0,23	-	0,92	0,47
Lon2	0,45	0,46	0,48	0,51	0,51	0,47	0,33	0,92	-	0,55
Lon3	0,26	0,34	0,32	0,42	0,42	0,28	0,42	0,47	0,55	-

Tablica 4: Rezultati sličnosti - WordNet

Rezultati pokretanja koda daju relativno visoke rezultate sličnosti između tekstova it1.txt, it3.txt i it5.txt, što je i razumljivo jer navedeni tekstovi sadrže definicije informacijske tehnologije, te kako govore o istim stvarima automatski je i njihova sličnosti vrlo velika. To se posebno odnosi na it1.txt i it5.txt koji su preuzeti sa istog linka, te im je velik dio teksta identičan.

Rezultati usporedbe sa tekstom apple.txt također daju relativno visoke rezultate, čemu je razlog vjerojatno taj što se također radi o informatičkim stvarima, te se većina pojmova podudara.

Također, rezultat sličnosti između tekstova croatia.txt i croatia2.txt je 0.99999, što predstavlja rezultat maksimalne sličnosti kakav bi i trebao biti s obzirom da se radi o jednakim tekstovima.

Tekstovi čiji sadržaj govori o gradu Londonu ne pokazuju veliku sličnost sa ostalim tekstovima, ali zato u međusobnom uspoređivanju daju vrlo visoke rezultate, što se pogotovo odnosi na tekstove london1.txt i london2.txt čija je izmjerena sličnost jednaka 0,92.

I za kraj, usporedbe teksta o psima sa ostalim tekstovima daju vrlo niske rezultate sličnosti što je i razumljivo jer govorimo o tekstovima potpuno različite tematike i sadržaja.

4.4.3. Rezultati sličnosti tekstova dobiveni primjenom biblioteka FuzzyWuzzy i difflib

Usporedbu sličnosti svih ranije navedenih tekstova pomoću biblioteke FuzzyWuzzy obavljamo tako što u prethodno opisani programski kod ubacujemo tekstove koje želimo usporediti. U kodu sam također unio biblioteku difflib koja sadrži alate za rad sa tekstom, te je iz biblioteke difflib korištena klasa SequenceMatcher koja se koristi kod uspoređivanja parova tekstova.

Semantička sličnost između tekstova izražena je u rasponu od 0 do 1, gdje naravno 0 pokazuje da tekstovi nemaju nikakvu sličnosti, dok je se rezultat 1 odnosi na identične tekstove.

Pokretanjem koda dobiju se navedeni rezultati koji prikazuju koliko je tekst it1.txt sličan sa ostalim tekstovima.

```
ivan@LIFEBOOK-AH532:~$ emacs -nw fw.py
ivan@LIFEBOOK-AH532:~$ python fw.py
Slicnost it1 i it3 :
0.172949002217
Slicnost it1 i it5 :
0.852390852391
Slicnost it1 i croatia :
0.029106029106
Slicnost it1 i croatia2 :
0.029106029106
Slicnost it1 i apple :
0.0627943485086
Slicnost it1 i dog :
0.117647058824
Slicnost it1 i london1 :
0.350877192982
Slicnost it1 i london2 :
0.0332541567696
Slicnost it1 i london3 :
0.029484029484
```

Slika 12: Rezulatati sličnosti među dokumentima - FuzzyWuzzy

Rezultati sličnosti između pojedinih dokumenta dani su u sljedećoj tablici:

	IT1	IT3	IT5	Cro	Cro2	Apple	Dog	Lon1	Lon2	Lon3
IT1	-	0,17	0,85	0,029	0,029	0,062	0,12	0,35	0,03	0,03
IT3	0,17	-	0,21	0,27	0,27	0,056	0,099	0,23	0,026	0,022
IT5	0,85	0,21	-	0,025	0,025	0,057	0,089	0,32	0,028	0,025
Cro	0,029	0,27	0,025	-	1	0,011	0,096	0,24	0,04	0,05
Cro2	0,029	0,27	0,025	1	-	0,011	0,096	0,24	0,04	0,05
Apple	0,062	0,056	0,057	0,01	0,01	-	0,049	0,176	0,03	0,019
Dog	0,12	0,099	0,089	0,096	0,096	0,049	-	0,14	0,017	0,018
Lon1	0,35	0,23	0,32	0,24	0,24	0,176	0,14	-	0,776	0,118
Lon2	0,03	0,026	0,028	0,04	0,04	0,03	0,017	0,776	-	0,096
Lon3	0,03	0,022	0,025	0,05	0,05	0,019	0,018	0,118	0,096	-

Tablica 5: Rezultati sličnosti - FuzzyWuzzy

Sam pogled na rezultate sličnosti tekstova dobivene na ovaj način pokazuje da su rezultati poprilično manji od rezultata dobivenih korištenjem Wordneta, odnosno da su sličnosti tekstova manje.

No, neke stvari se pokazuju vrlo slične u oba načina, a to je prije svega visoka sličnost tekstova it1.txt i it5.txt (0,85) koji u sebi sadrže veći dio teksta koji je identičan, te maksimalna sličnost koju pokazuju tekstovi croatia.txt i croatia2.txt što je i opravdano jer se radi o apsolutno jednakim tekstovima.

Rezultati sličnosti za datoteke apple.txt i dog.txt pokazuju da navedeni tekstovi imaju vrlo malo sličnosti sa ostalima, što je i opravdano jer se sadržaj tih tekstova poprilično razlikuje od ostalih.

Također, tekstovi koji govore o Londonu daju zanemarivo malu sličnost sa ostalim tekstovima, osim kada se uspoređuju međusobno gdje je sličnost tekstova london1.txt i london2.txt vrlo visoka i iznosi 0,776.

Iz navedenih se rezultata može zaključiti da se pomoću biblioteke FuzzyWuzzy i difflib mogu dobiti vrlo realni rezultati sličnosti između tekstova.

5. ZAKLJUČAK

U ovom diplomskom radu opisana je semantička sličnost tekstova, postupci za njezino mjerenje, te je na konkretnom primjeru tekstova i programa za računanje semantičke sličnosti prikazano kako se dobivaju rezultati sličnosti tekstova.

Cilj je bio prikazati i opisati postupke za mjerenje semantičke sličnosti tekstova, kao i predstaviti algoritme koji se koriste u tu svrhu.

U radu je dan i pregled mjera semantičke sličnosti koje se koriste za procjenu sličnosti između pojmova i tekstova, te je prikazana i usporedba navedenih mjera, kao i praktična primjena.

Također, u radu su predstavljeni i pristupi za mjerenje semantičke sličnosti koji se dijele na dvije velike cjeline:

- Pristup baziran na korpusima
- Pristup baziran na znanju

Određivanje semantičke sličnosti ima bitnu ulogu kod kategorizacije i sumiranja tekstova, te kod pronalaženja informacija. Također, ima važnu primjenu kod uspoređivanja sličnosti tekstova, odnosno pronalaženja plagijata.

Sama primjena semantičke sličnosti na rečenicama i tekstovima prikazana je pomoću programskog jezika Python u kojem sam implementirao tehnike za računanje sličnosti. Rezultati dobiveni primjenom FuzzyWuzzy i pomoću Wordneta se međusobno razlikuju, što nam daje na znanje da različite metode dobivanja rezultata semantičke sličnosti ne daju iste rezultate kao što bi se očekivalo.

Prikazani pristup sustava za određivanje semantičke sličnosti koji se temelji na korpusima moguće je poboljšati na način da se koristi veći korpus, jer je samim time i veći broj riječi.

LITERATURA

[1] Corley, C., Mihalcea, R. (2006.)
Corpus-based and Knowledge-based Measures of Text Semantic Similarity; University of North Texas, Department of Computer Science

[2] Corley, C., Mihalcea, R. (2005.)
Measuring the Semantic Similarity of Texts; University of North Texas, Department of Computer Science

[3] Pawar, A., Mago, V. (2018.)
Calculating the similarity between words and sentences using a lexical database and corpus statistics, *IEEE transactions on knowledge and data engineering*

[4] Huang, L. (2017.)
Measuring Similarity Between Texts in Python

[5] Batanović, V., Furlan. B., Nikolić, B. (2014.)
Softverski sistem za određivanje semantičke sličnosti kratkih tekstova na srpskom jeziku

[6] Pal, S. (2014.)
Computing Semantic Similarity for Short Sentences

[7] Althobaiti, A. F. S. (2017.)
Comparison of Ontology-Based Semantic Smiliarity Measures in the Biomedical Text. Department of Computer and Information Sciences, Al-Imam Muhammad Ibn Saud Islamic University, Riyadh, Saudi Arabia

[8] Measures of Semantic Similarity
https://www.3pillarglobal.com/insights/measures-of-semantic-similarity

[9] Bekavac, B. (2001).
Primjena računalnojezikoslovnih alata na hrvatske korpuse. Sveučilište u Zagrebu.

[10] Li, Y., McLean, D., Bandar, Z. A., O'shea, J. D., & Crockett, K. (2006). Sentence similarity based on semantic nets and corpus statistics. IEEE transactions on knowledge and data engineering, 18(8), 1138-1150.

IZVORI

http://www.mensa.hr/glavna/misli-21-stoljeca/652-semanticki-web. (n.d.).

Matešić, M. (3. 5 2018). *SEMANTIČKI WEB*. Dohvaćeno iz Hrvatska Mensa: http://www.mensa.hr/glavna/misli-21-stoljeca/652-semanticki-web

Pejić, T. (1. 9 2015). *Računalni učenički korpusi i učenički korpus hrvatskog jezika.* Dohvaćeno iz Računalni učenički korpusi i učenički korpus hrvatskog jezika: http://darhiv.ffzg.unizg.hr/id/eprint/6096/1/Pejic_Tea_Ra%C4%8Dunalni%20u%C4%8Deni%C4%8Dki%20korpusi%20i%20u%C4%8Deni%C4%8Dki%20korpus%20hrvatskog%20jezika.pdf

Vuk Batanović, B. F. (10. 5 2018). *Softverski sistem za određivanje semantičke sličnosti kratkih tekstova na srpskom jeziku.* Dohvaćeno iz Softverski sistem za određivanje semantičke sličnosti kratkih tekstova na srpskom jeziku: http://www.academia.edu/8520980/Softverski_sistem_za_odre%C4%91ivanje_semanti%C4%8Dke_sli%C4%8Dnosti_kratkih_tekstova_na_srpskom_jeziku_A_software_system_for_determining_the_semantic_similarity_of_short_texts_in_Serbian

2. (n.d.). Dohvaćeno iz https://bib.irb.hr/datoteka/231724.AutomatskoIndeksiranjeDokumenataUModeluVektorskogProstora.pdf.

3. (n.d.). Dohvaćeno iz http://sigurnost.zemris.fer.hr/ostalo/detekcija_plagijata/2011_baotic/ab39079.pdf

4. (n.d.). Dohvaćeno iz https://pdfs.semanticscholar.org/1374/617e135eaa772e52c9a2e8253f49483676d6.pdf

5. (n.d.). Dohvaćeno iz http://www.mensa.hr/glavna/misli-21-stoljeca/652-semanticki-web

6. (n.d.). Dohvaćeno iz http://www.mensa.hr/glavna/misli-21-stoljeca/652-semanticki-web

7. (n.d.). Dohvaćeno iz http://ms1psz.etf.rs/projekat/Projektni%20zadatak%202014-2015.pdf

8. (n.d.). Dohvaćeno iz http://home.etf.rs/~bfurlan/publications/TELFOR2011.pdf

9. (n.d.). Dohvaćeno iz https://pdfs.semanticscholar.org/1374/617e135eaa772e52c9a2e8253f49483676d6.pdf

10. (n.d.). Dohvaćeno iz http://darhiv.ffzg.unizg.hr/id/eprint/6096/1/Pejic_Tea_Ra%C4%8Dunalni%20u%C4%8Deni%C4%8Dki%20korpusi%20i%20u%C4%8Deni%C4%8Dki%20korpus%20hrvatskog%20jezika.pdf

11. (n.d.). Dohvaćeno iz https://www.slideshare.net/Cogpsychteacher/lb-wk-12

12. (n.d.). Dohvaćeno iz https://www.researchgate.net/profile/Vuk_Batanovic/publication/261120444_Evaluacija_i_klasifikacija_koriscenja_sintaksnih_informacija_u_odredivanju_semanticke_slicnosti_kratkih_tekstova_Evaluation_and_Classification_of_Syntax_Information_Usage_in_Determining_Short-Text_Sem/links/54289a4c0cf2e4ce940c4f5d/Evaluacija-i-klasifikacija-koriscenja-sintaksnih-informacija-u-odredivanju-semanticke-slicnosti-kratkih-tekstova-Evaluation-and-Classification-of-Syntax-Information-Usage-in-Determining-Short-Text-S.pdf
13. (n.d.). Dohvaćeno iz https://pdfs.semanticscholar.org/1374/617e135eaa772e52c9a2e8253f49483676d6.pdf
13. (n.d.). Dohvaćeno iz https://www.slideshare.net/sharaf/text-similarity
14. (n.d.). Dohvaćeno iz https://www.slideshare.net/sharaf/text-similarity
15. (n.d.). Dohvaćeno iz http://www.d.umn.edu/~tpederse/Tutorials/IHI2012-semantic-similarity-tutorial.pdf
15. (n.d.). Dohvaćeno iz http://www.d.umn.edu/~tpederse/Tutorials/IHI2012-semantic-similarity-tutorial.pdf
15. (n.d.). Dohvaćeno iz https://bib.irb.hr/datoteka/888255.Raunalna_detekcija_plagijata-pregled_metoda_i_algoritama.pdf
16. (n.d.). Dohvaćeno iz https://bib.irb.hr/datoteka/888255.Raunalna_detekcija_plagijata-pregled_metoda_i_algoritama.pdf
1. (n.d.). Dohvaćeno iz http://home.etf.rs/~bfurlan/publications/TELFOR2011.pdf
1. (n.d.). Dohvaćeno iz https://pdfs.semanticscholar.org/1374/617e135eaa772e52c9a2e8253f49483676d6.pdf
17. (n.d.). Dohvaćeno iz https://pdfs.semanticscholar.org/1374/617e135eaa772e52c9a2e8253f49483676d6.pdf
18. (n.d.). Dohvaćeno iz https://pdfs.semanticscholar.org/1374/617e135eaa772e52c9a2e8253f49483676d6.pdf

DODACI

DODATAK A

Dokumenti korišteni kod izračuna semantičke sličnosti tekstualnih dokumenta:

bug1.txt :

«This dog sleeps on the bed every night.»

bug2.txt :

«Information technology is the use of computers to store, retrieve, transmit and manipulate data, or information, often in the context of a business or other enterprise. It is considered to be a subset of information and communications technology. Humans have been storing, retrieving, manipulating, and communicating information since the Sumerians in Mesopotamia developed writing in about 3000 BC, but the term information technology in its modern sense first appeared in a 1958 article published in the Harvard Business Review, authors Harold J. Leavitt and Thomas L. Whisler commented that "the new technology does not yet have a single established name. We shall call it information technology." Their definition consists of three categories: techniques for processing, the application of statistical and mathematical methods to decision-making, and the simulation of higher-order thinking through computer programs. The term is commonly used as a synonym for computers and computer networks, but it also encompasses other information distribution technologies such as television and telephones. Several products or services within an economy are associated with information technology, including computer hardware, software, electronics, semiconductors, internet, telecom equipment, and e-commerce.»

bug3.txt :

«Information technology is the use of computers to store, retrieve, transmit and manipulate data, or information, often in the context of a business or other enterprise. It is considered to be a subset of information and communications technology. Humans have been storing, retrieving, manipulating, and communicating information since the Sumerians in Mesopotamia developed writing in about 3000 BC, but the term information technology in its modern sense first appeared in a 1958 article published in the Harvard Business Review, authors Harold J. Leavitt and Thomas L. Whisler commented that "the new technology does not yet have a single established name. We shall call it information technology." Their definition consists of three categories: techniques for processing, the application of statistical and mathematical methods to decision-making, and the simulation of higher-order thinking through computer programs.»

DODATAK B

U nastavku je prikazan cijeli programski kod u Pythonu za izračun sličnosti između tekstova korištenjem Wordneta:

```
from nltk import word_tokenize, pos_tag
from nltk.corpus import WordNet as wn

def pretvorba_oznaka(tag):
    if tag.startswith('N'):
        return 'n'

    if tag.startswith('V'):
        return 'v'

    if tag.startswith('J'):
        return 'a'

    if tag.startswith('R'):
        return 'r'

    return None

def synset_oznake(word, tag):
    wn_tag = pretvorba_oznaka(tag)
    if wn_tag is None:
        return None

    try:
        return wn.synsets(word, wn_tag)[0]
    except:
        return None

def slicnost_tekstova(text1, text2):
    text1 = pos_tag(word_tokenize(text1))
    text2 = pos_tag(word_tokenize(text2))

    synset1 = [synset_oznake(*tagged_word) for tagged_word in text1]
    synset2 = [synset_oznake(*tagged_word) for tagged_word in text2]

    synset1 = [ss for ss in synset1 if ss]
    synset2 = [ss for ss in synset2 if ss]

    rezultat, brojac = 0.0, 0

    for synset in synset1:
        # izracunaj slicnost sa najslicnijom rijeci u drugoj recenici
        best_score = max([synset.path_similarity(ss) for ss in synset2])

        if best_score is not None:
            rezultat += best_score
            brojac += 1

    rezultat /= brojac
    return rezultat

sentences = [
    "Man is ugly, but  strong",
    "The world is wonderful.",
    "This house is empty.",
    "Dogs are cute",
    "Cats are cute, but they are very bad.",
]

glavna = "Cats are cute."

for sentence in sentences:
```

```
    print "Slicnost(\"%s\", \"%s\") = %s" % (glavna, sentence, slicnost_tekstova(glavna,
sentence))
    print    "Slicnost(\"%s\",    \"%s\")    =    %s"    %    (sentence,    glavna,
slicnost_tekstova(sentence, glavna))
    print
```

DODATAK C

U nastavku je prikazan cijeli programski kod u Pythonu za izračun sličnosti između tekstova korištenjem biblioteke FuzzyWuzzy i difflib:

```
from fuzzywuzzy import fuzz
from fuzzywuzzy import process
from difflib  import SequenceMatcher as SM

it1 = 'Information technology is the use of any computers, storage, networking and other
physical devices, infrastructure and processes to create, process, store, secure and
exchange all forms of electronic data.'
it3 = 'Information technology is the use of computers to store, retrieve, transmit and
manipulate data, or information, often in the context of a business or other enterprise.
It is considered to be a subset of information and communications technology.'
print ("Slicnost it1 i it3 : ")
print SM(None, it1, it3).ratio()

it1 = 'Information technology is the use of any computers, storage, networking and other
physical devices, infrastructure and processes to create, process, store, secure and
exchange all forms of electronic data.'
it5 = 'Information technology is the use of any computers, storage, networking and other
physical devices, infrastructure and processes to create, process, store, secure and
exchange all forms of electronic data. Information Technology is a business sector that
deals with computing.'
print ("Slicnost it1 i it5 : ")
print SM(None, it1, it5).ratio()

it1 = 'Information technology is the use of any computers, storage, networking a\
nd other physical devices, infrastructure and processes to create, process, sto\
re, secure and exchange all forms of electronic data.'
cro = 'Croatia is a country situated in Central Europe. It is to the east side of the
Adriatic Sea, to the east of Italy. It is also bordered by Slovenia to the northwest,
Hungary to the north, Bosnia and Herzegovina to the southeast, Serbia in the east, and
Montenegro to the south.'
print ("Slicnost it1 i croatia : ")
print SM(None, it1, cro).ratio()

it1 = 'Information technology is the use of any computers, storage, networking a\
nd other physical devices, infrastructure and processes to create, process, sto\
re, secure and exchange all forms of electronic data.'
cro2 = 'Croatia is a country situated in Central Europe. It is to the east side of the
Adriatic Sea, to the east of Italy. It is also bordered by Slovenia to the northwest,
Hungary to the north, Bosnia and Herzegovina to the southeast, Serbia in the east, and
Montenegro to the south.'
print ("Slicnost it1 i croatia2 : ")
print SM(None, it1, cro2).ratio()

it1 = 'Information technology is the use of any computers, storage, networking a\
nd other physical devices, infrastructure and processes to create, process, sto\
re, secure and exchange all forms of electronic data.'
apple = 'Apple is an American multinational technology company headquartered in
Cupertino, California, that designs, develops, and sells consumer electronics, computer
software, and online services. The company is hardware products include the iPhone
smartphone, the iPad tablet computer, the Mac personal computer, the iPod portable media
```

```
player, the Apple Watch smartwatch, the Apple TV digital media player, and the HomePod
smart speaker.'
print SM(None, it1, apple).ratio()

it1 = 'Information technology is the use of any computers, storage, networking a\
nd other physical devices, infrastructure and processes to create, process, sto\
re, secure and exchange all forms of electronic data.'
dog = 'Cats are animal.'
print ("Slicnost it1 i dog : ")
print SM(None, it1, dog).ratio()

it1 = 'Information technology is the use of any computers, storage, networking a\
nd other physical devices, infrastructure and processes to create, process, sto\
re, secure and exchange all forms of electronic data.'
london1 = 'London is the political, economic and cultural capital of Britain, and its
world class tourist attractions are renowned across the globe.'
print ("Slicnost it1 i london1 : ")
print SM(None, it1, london1).ratio()

it1 = 'Information technology is the use of any computers, storage, networking a\
nd other physical devices, infrastructure and processes to create, process, sto\
re, secure and exchange all forms of electronic data.'
london2= 'London is the political, economic and cultural capital of Britain, and its
world class tourist attractions are renowned across the globe. The Greater London area
is bursting with attractions for visitors of all ages.'
print ("Slicnost it1 i london2 : ")
print SM(None, it1, london2).ratio()

it1 = 'Information technology is the use of any computers, storage, networking a\
nd other physical devices, infrastructure and processes to create, process, sto\
re, secure and exchange all forms of electronic data.'
london3 = 'London is the capital and largest city of England and the United Kingdom.
The River Thames travels through the city. London is the biggest city in western Europe,
and the world largest financial centre.'
print ("Slicnost it1 i london3 : ")
print SM(None, it1, london3).ratio()
```

DODATAK D

U nastavku su dani tekstovi koji se koriste za usporedbu u kodu za izračunavanje semantičke sličnosti između tekstualni dokumenata, te linkovi sa kojih su navedeni tekstovi preuzeti:

it1.txt:

Information technology (IT) is the use of any computers, storage, networking and other physical devices, infrastructure and processes to create, process, store, secure and exchange all forms of electronic data.Typically, IT is used in the context of enterprise operations as opposed to personal or entertainment technologies. The commercial use of IT encompasses both computer technology and telephony. The term information technology was coined by the Harvard Business Review, in order to make a distinction between purpose-built machines designed to perform a limited scope of functions and general-purpose computing machines that could be programmed for various tasks. As the IT industry evolved from the mid-20th century, it encompassed transistors and integrated circuits -- computing capability advanced while device cost and energy consumption fell lower, a cycle that continues today when new technologies emerge.
IT includes several layers of physical equipment (hardware), virtualization and management or automation tools, operating systems and applications (software) used to perform essential

functions. User devices, peripherals and software, such as laptops, smartphones or even recording equipment, can be included in the IT domain. IT can also refer to the architectures, methodologies and regulations governing the use and storage of data.

https://searchdatacenter.techtarget.com/definition/IT

it3.txt:

Information technology is the use of computers to store, retrieve, transmit and manipulate data, or information, often in the context of a business or other enterprise. It is considered to be a subset of information and communications technology. Humans have been storing, retrieving, manipulating, and communicating information since the Sumerians in Mesopotamia developed writing in about 3000 BC, but the term information technology in its modern sense first appeared in a 1958 article published in the Harvard Business Review, authors Harold J. Leavitt and Thomas L. Whisler commented that "the new technology does not yet have a single established name. We shall call it information technology." Their definition consists of three categories: techniques for processing, the application of statistical and mathematical methods to decision-making, and the simulation of higher-order thinking through computer programs.

https://en.wikipedia.org/wiki/Information_technology

it5.txt:

Typically, IT is used in the context of enterprise operations as opposed to personal or entertainment technologies. The commercial use of IT encompasses both computer technology and telephony.
The term information technology was coined by the Harvard Business Review, in order to make a distinction between purpose-built machines designed to perform a limited scope of functions and general-purpose computing machines that could be programmed for various tasks. As the IT industry evolved from the mid-20th century, it encompassed transistors and integrated circuits - computing capability advanced while device cost and energy consumption fell lower, a cycle that continues today when new technologies emerge.
IT includes several layers of physical equipment (hardware), virtualization and management or automation tools, operating systems and applications (software) used to perform essential functions. User devices, peripherals and software, such as laptops, smartphones or even recording equipment, can be included in the IT domain. IT can also refer to the architectures, methodologies and regulations governing the use and storage of data.

https://searchdatacenter.techtarget.com/definition/IT

croatia.txt:

Croatia is a country situated in Central Europe. It is to the east side of the Adriatic Sea, to the east of Italy. It is also bordered by Slovenia to the northwest, Hungary to the north, Bosnia and Herzegovina to the southeast, Serbia in the east, and Montenegro to the south.

https://wikitravel.org/en/Croatia

croatia2.txt:

Croatia is a country situated in Central Europe. It is to the east side of the Adriatic Sea, to the east of Italy. It is also bordered by Slovenia to the northwest, Hungary to the north, Bosnia and Herzegovina to the southeast, Serbia in the east, and Montenegro to the south.

https://wikitravel.org/en/Croatia

dog.txt:

Dogs are domesticated mammals, not natural wild animals. They were originally bred from wolves. They have been bred by humans for a long time, and were the first animals ever to be domesticated.
Today, some dogs are used as pets, others are used to help humans do their work. They are a popular pet because they are usually playful, friendly, loyal and listen to humans. Thirty million dogs in the United States are registered as pets. Dogs eat both meat and vegetables, often mixed together and sold in stores as dog food. Dogs often have jobs, including as police dogs, army dogs, assistance dogs, fire dogs, messenger dogs, hunting dogs, herding dogs, or rescue dogs.
They are sometimes called "canines" from the Latin word for dog - canis. Sometimes people also use "dog" to describe other canids, such as wolves. A baby dog is called a pup or puppy. A dog is called a puppy until it is about one year old.
Dogs are sometimes referred to as "man's best friend" because they are kept as domestic pets and are usually loyal and like being around humans.

https://simple.wikipedia.org/wiki/Dog

apple.txt:

Apple Inc. is an American multinational technology company headquartered in Cupertino, California, that designs, develops, and sells consumer electronics, computer software, and online services. The company's hardware products include the iPhone smartphone, the iPad tablet computer, the Mac personal computer, the iPod portable media player, the Apple Watch smartwatch, the Apple TV digital media player, and the HomePod smart speaker. Apple's software includes the macOS and iOS operating systems, the iTunes media player, the Safari web browser, and the iLife and iWork creativity and productivity suites, as well as professional applications like Final Cut Pro, Logic Pro, and Xcode. Its online services include the iTunes Store, the iOS App Store and Mac App Store, Apple Music, and iCloud.

https://en.wikipedia.org/wiki/Apple_Inc.

lond1.txt :

London is the political, economic and cultural capital of Britain, and its world class tourist attractions are renowned across the globe.

https://www.jumeirah.com/en/destinations/london/about-london/

lond2.txt :

London is the political, economic and cultural capital of Britain, and its world class tourist attractions are renowned across the globe. The Greater London area is bursting with attractions for visitors of all ages.
https://www.jumeirah.com/en/destinations/london/about-london/

lond3.txt :

London is the capital and largest city of England and the United Kingdom. The River Thames travels through the city.
London is the biggest city in western Europe, and the world's largest financial centre.

https://simple.wikipedia.org/wiki/London