Ekstraktivno sažimanje članaka Wikipedije na njemačkom jeziku

Beli, Dorian

Master's thesis / Diplomski rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Rijeka / Sveučilište u Rijeci**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:195:378309

Sveučilište u Rijeci – Odjel za informatiku

Informacijski komunikacijski sustavi

Dorian Beli

Ekstraktivno sažimanje članaka Wikipedije na njemačkom jeziku

Diplomski rad

Mentori: prof.dr.sc. Sanda Martinčić-Ipšić

dr.sc. Slobodan Beliga

Rijeka, rujan 2021.

Sadržaj

Predgovor

Prvenstveno bih se zahvalio svojim mentorima prof. dr. sc. Sandi Martinčić-Ipšić i dr. sc. Slobodanu Beligi na ukazanom mentorstvu i pruženoj prilici rada na ovoj dosta opsežnoj, ali interesantnoj temi. Također se zahvaljujem Odijelu za informatiku te cijelom stručnom kadru na pruženom znanju i iskustvu.

Veliko hvala svim prijateljima i poznanicima na podršci pruženoj tijekom svih godina, čija mi podrška mnogo značila.

Ipak najveće hvala mojim roditeljima na svemu do sada, pogotovo ocu koji je tijekom pisanja ovog rada imao uspješnu operaciju raka u ranom stadiju iza sebe. Hvala roditeljima, koji su unatoč ovako ozbiljnoj bolesti, ipak ostali snažni.

Veliko *HVALA* svima!

Sažetak

U svijetu ovisnom sve više o online i cloud tehnologijama gdje se u posljednje vrijeme odvija sve veća pohrana i obrada podataka, pojavljuje se potreba za algoritmima sažimanja i rangiranja tekstualnih datoteka, stranica i drugih izvora. Algoritmi ekstraktivnog i apstraktnog sažimanja tekstova nastoje na što brži, bolji i jednostavniji način obraditi i predstaviti velike količine tekstova u što kraćem vremenu sa što većom preciznošću. U ovom diplomskom radu riječ je upravo o nekoliko takvih algoritama te se prvenstveno usmjerava na algoritme ekstraktivne prirode. Kako bi se utvrdila važnost rečenice, njihova međusobna povezanost te koje rečenice imaju veće značenje razvijeni su različiti algoritmi sličnosti rečenica. Ovdje se primjenjuju algoritmi sličnosti kosinusa, Jaccardova sličnost, te Mihalcea sličnost (Mihalcea, Corley i Strapparava 2006) te algoritmi rangiranja i sažimanja SBKE (Beliga, Martinčić-Ipšić i Meštrović 2016) i LexRank. Na temelju formiranih grafova i primjene različitih mjera sličnosti rečenica pristupom stvaranja grafova, SBKE je ostvario najbolji rezultat u kombinaciji sa Jaccardovom mjerom sličnosti do 19,04% , LexRank u kombinaciji s Mihalcea algoritmom sličnosti rečenica do 16,26%, algoritam sažimanja na temelju Kullback_Leiblerove divergencije je ostvario 8,96% točnosti, a TextRank, koji koristi čisti PageRank algoritam bez mjera sličnosti, je ostvario rezultat od 17,04%. Također su primijenjeni i klasifikatori na temelju TF-IDF, word2vec i doc2vec vektora. Algoritam slučajnih šuma je ostvario najbolji rezultat od 76,51%, algoritam Naîvnog Bayesa je ostvario najbolji rezultat od 91,36%, dok je SVM klasifikator ostvario najbolji rezultat od 90,43%.

Ključne riječi, SBKE, LexRank, TextRank, ekstraktivne metode sažimanja, njemački jezik, sažimanje njemačkih tekstova, mjere sličnosti

Extraktive Zusammenfassung von deutsche Wikipedia

Zusammenfassung

In einer Welt von Online- und Cloud-Technologien zunehmenden Abhängigkeit werden heutzutage die Daten immer mehr gespeichert und verarbeitet und Algorithmen für die automatische Zusammenfassung und Ranking von Textdateien, Seiten und anderen Quellen werden benötigt. Die extraktiven und abstraktiven Algorithmen versuchen am schnellsten, mit größtmöglicher Präzision in kürzester Zeit die Textmengen zu verarbeiten und darzustellen. Diese Arbeit befasst sich mit mehreren solcher Algorithmen und konzentriert sich auf extraktive Algorithmen. Um die Wichtigkeit eines Satzes festzustellen, sein Zusammenhang und welche Sätze eine größere Wichtigkeit haben, wurden die verschiedene Satzähnlichkeitsalgorithmen entwickelt. In dieser Arbeit werden Kosinus-Ähnlichkeitsalgorithmen, Jaccard Ähnlichkeit und Mihalcea Ähnlichkeit sowie die Algorithmen für die automatische Zusammenfassung von Texten SBKE und LexRank verwendet. Auf der Grundlage der gebildeten Graphen und der Anwendung verschiedener Satzähnlichkeitsalgorithmen erzielte SBKE das beste Ergebnis in Kombination mit Jaccards Satzähnlichkeitsalgorithmus bis zu 19,04 Prozent, LexRank in Kombination mit Mihalcea Satzähnlichkeitsalgorithmus bis zu 16,26 Prozent, Kullback-Leibler Divergenz Algorithmus für die automatische Zusammenfassung erzielte das beste Ergebnis bis zu 8,96 Prozent und TextRank erreichte ein Ergebnis von 17,04 Prozent Die Klassifizierungsalgorithmen basierend auf TF-IDF, word2vec und doc2vec Vektoren wurden ebenfalls verwendet. Der Random Forest Algorithmus erzielte das beste Ergebnis bis zu 76,51 Prozent, der Naîve Bayes erzielte das beste Ergebnis bis zu 91,36 Prozent, während der SVM erzielte das beste Ergebnis bis zu 90,04 Prozent.

Schlüsselwörter: SBKE, LexRank, TextRank, extraktive automatische Zusammenfassung, deutsche Sprache, automatische Zusammenfassung von deutschsprachigen Texten, Ähnlichkeitsalgorithmen

Extractive Summarization of German Wikipedia

Abstract

In the world growingly more dependent on online and cloud technologies, where society stores and processes more and more data, the need for algorithms capable of summarizing and ranking texts, websites and other sources is constantly rising. Extractive and abstractive text summarization algorithms strive for faster, better and simpler solutions towards large quantities of text processing and representation in shortest time possible with the best precision. This paper talks about such algorithms with the focus on those with extractive nature. In order to decide on the importance of the sentence, various algorithms are developed, taking the interconnectedness of the sentences into account. Cosine similarity, Jaccard similarity and Mihalcea similarity (Mihalcea, Corley i Strapparava 2006), as well as LexRank and SBKE (Beliga, Martinčić-Ipšić i Meštrović 2016) summarizers, are being presented. Based on graphs using all the given sentence similarity measures, SBKE scored the best result of 19,04% using Jaccard similarity measure, LexRank resulted with 16,26% F1 score, the KL divergence summarizer resulted with 8,96% F1 score and TextRank, which uses the PageRank algorithm without any mentioned similarity measure, has an F1 score of 17,04%. For text classification, based on TF-IDF, word2vec and doc2vec algorithms, Random forest scored the best result of 76,51%, Naîve Bayes scored the best result of 91,36% and SVM scored the best result of 90,04%.

Keywords: SBKE, LexRank, TextRank, extractive summarization methods, German language, German language text summarization, similarity measures

1 Uvod

U današnjem svijetu gdje se većina podataka nalazi online, čime obuhvaćamo ogromnu količinu pisanih datoteka, videa i zapisa, koji u urbanom informatičkom svijetu donose iznimnu važnost. Kada govorimo o obradi tih podataka, njihova analiza u neformatiranom obliku zahtjevan je za procesiranje te je, kao takav, iznimno zasićen za današnje informatičko doba i njegove korisnike. Dakako govorimo li o ručnoj obradi, riječ o ogromnoj potrošnji vremena i resursa koji bi davali slabe ili čak loše rezultate. Kao odgovor na tu prepreku za željenom automatizacijom sažimanja navedenih datoteka koji će, ovisno o implementaciji, raditi sažimanje jednog dokumenta ili čak njih više ili ako govorimo o cijeloj niti e-pošte te će rezultirati smislenim i fluentnim sažetkom.

1.1 Metoda ekstrakcije

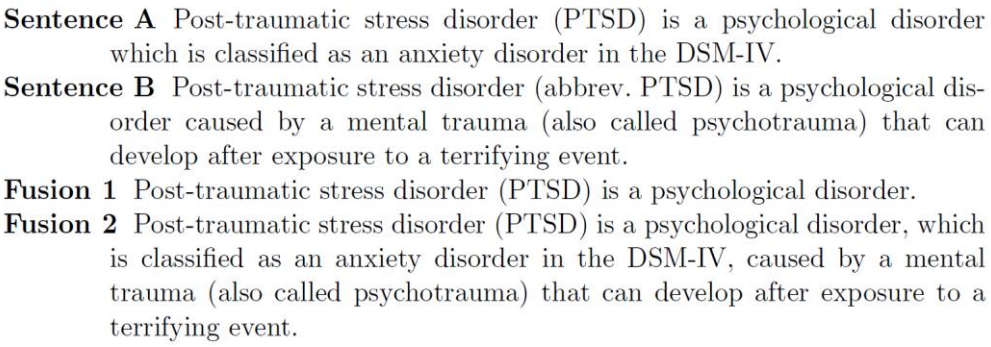
Metoda ekstrakcije koristi konkatenaciju više rečenica koje se originalno nalaze unutar teksta koji se skraćuje (Nenkova i McKeown, Automatic Summarization 2011). Jeda od poznatijih pristupa ovog načina predstavio je H.P. Luhn, čiji je rad baziran za sažimanje novinskih članaka i stručnih radova (Luhn 1958.). U svom radu Luhn tvrdi kako neke ključne riječi unutar dokumenta imaju deskriptivnu funkciju svog sadržaja te kako sve rečenice koje sadrže mnogo takvih riječi se većinom nalaze blizu jedna druge. Osim toga predlaže korištenje modela frekvencije pojavljivanja, kojim se utvrđuje koje riječi imaju opisno svojstvo teme dokumenta. U ovu vrstu riječi isključujemo interpunkciju, prepozicije i slične riječi (ovisno o jeziku koji skraćujemo) te ih se kao takve svrstava unutar posebne predefinirane liste. Osim navedenih također treba uzeti u obzir i riječi koje ne pripadaju prethodno definiranoj listi ali i dalje kao takve ne daju nikakve informacije ili ne indiciraju temu dokumenta, jer su česte za područje o kojem određeni tekst govori.

U kasnijim implementacijama također se koristi i TF-IDF kako bismo utvrdili koje riječi imaju željenu opisnu ulogu u danome tekstu. U kasnijim prijedlozima algoritma umjesto navedenoga, koristi se frekvencija pojavljivanja riječi kao bi se modelirale sličnosti između rečenica te kako bi se dobio uvid u značajnost tih sličnosti među rečenicama. Ovaj pristup dakako ima svoje nedostatke. Prvenstveno ovaj pristup ne uzima u obzir pojavljivanje riječi u drugim morfološkim oblicima, već će ih brojati kao novu riječ. Kako bi se riješio ovaj problem, primjenjuje se gruba aproksimacija prema morfološkoj analizi tako da se sve riječi koje su slične izuzev zadnjih šest slova (Nenkova i McKeown 2011).

1.2 Druge metode

Većina metoda primjenjuje već navedene pristupe ekstrakcije rečenica iz teksta, u postupku u potpunosti zaobilazeći probleme nazočne u kontekstu generiranja sažetaka. Kako bi se riješio navedeni problem, postoji nekoliko načina poboljšanja u tom području (Nenkova i McKeown 2011):

- **Redoslijed rečenica;** Ovdje koristimo metodu ekstrakcije kako bismo dobili najvažnije rečenice unutar zadanog teksta te ih potom organiziramo na najkoherentniji način u odnosu na semantičku smislenost generiranog sažetka.
- **Prerada rečenica;** Ovdje govorimo o prvom pokušaju rješavanja generacije teksta na određenom jeziku. Ovaj pristup uključuje ponovno korištenje teksta dobivenog kao rezultat sumarizacije kako bi se automatski modificirali dijelovi sažetka zamjenom određenih izraza drugim, semantički prikladnijim izrazima ovisno o kontekstu novog sažetka. Ranija rješenja su predlagala uklanjanje manje važnih dijelova rečenica, kombiniranjem informacije originalno prikazane u drugim rečenicama i supstitucijom zamjenice s nominalnom grupom koja pruža bolju opisnu ulogu ako kontekst sažetka zahtjeva takvu radnju. Ovakvo rješenje rezultiralo stvaranjem odvojenih grana istraživanja, pri čemu se svaka od grana koncentrira isključivo na jednu vrstu revizije (Nenkova 2006).
- **Stapanje rečenica;** Problemu pristupamo uzimajući dvije rečenice u kojima se informacija preklapa, ali kao takve sadržavaju dijelove koji su međusobno različiti. Krajnji cilj je stvoriti rečenicu koja sadrži uobičajenu informaciju u dvije rečenice, ili stvoriti rečenicu koja sadrži sve informacije u dvije rečenice bez redundancije (Nenkova 2006).
- **Kompresija rečenica;** Prilikom rješavanja problema ovim pristupom govorimo o stvaranju sažetka koji sadrži rečenice iz originalnog teksta, pri čemu su uklonjeni pojedini dijelovi rečenice što rezultira jezgrovitijim, razumljivijim sažetkom (Nenkova i McKeown 2011).



Slika 0-1: Primjer stapanja rečenica, pri čemu Fusion 1 predstavlja rečenicu koja sadrži informaciju sadržanu u obje rečenice A i B (njihov presjek) dok Fusion 2 predstavlja kombinaciju svih informacija dostupnih unutar rečenica A i B (unija rečenica A i B) (Nenkova, Summarization Evaluation for Text and Speech: Issues and Approaches 2006)

Ovaj diplomski rad se bavi ekstraktivnom metodom sažimanja rečenica na temelju izgradnje kompleksnih grafova između rečenica. Kako bi se odredila važnost i međusobna povezanost rečenica koriste se različite mjere sličnosti poput Kullback-Leiblerove divergencije, mjere sličnost definirane unutar (Mihalcea, Corley i Strapparava 2006), koju se, radi jednostavnosti referenciranja, još dodatno naziva i Mihalcea sličnost, mjere sličnosti kosinusa rečenica te Jaccardove mjere sličnosti rečenica. Principi rada navedenih algoritama moguće je pronaći pod naslovom 3 Algoritmi mjere sličnosti rečenica. Potom se na temelju LexRank (Mihalcea 2004) i SBKE (Beliga, Martinčić-Ipšić i Meštrović 2016) algoritama rangiraju rečenice u odnosu na primijenjeni prag. Opisi algoritama LexRank i SBKE se nalaze unutar cjeline 2.1 Metode nenadziranog učenja. Sama primjena algoritama mjera sličnosti rečenica i algoritama rangiranja rečenica su opisani unutar 5.5 Opis metoda ekstraktivnog sažimanja. Također se opisuje primjena algoritama klasifikacije na temelju TF-IDF, prosječnih word2vec i doc2vec vektora rečenica pod cjelinom 5.4 Klasifikacija teksta. Rezultate svih algoritama je moguće vidjeti u cjelini 6 Usporedba rezultata.

2 Utvrđivanje važnosti tijekom ekstrakcije rečenica

Postoji više načina kako bismo utvrdili važnost rečenice unutar danoga teksta. Unutar 1.1 Metoda ekstrakcije ukratko smo pojasnili jedan način dobivanja ključnih riječi i dijelova teksta, međutim to rješenje kao takvo nije robusno te je potrebno poduzeti još neke dodatne korake. Naš cilj je imati robustan algoritam koji je relativno ne zahtijevan za procesiranje značajki[1] za određivanje važnosti rečenica. Ako govorimo o takvim pristupima, tu ulazimo u područje nenadziranog, samostalnog učenja te takvi algoritmi ne zahtijevaju ljudski nadzor tijekom odlučivanja o važnosti rečenice unutar teksta, koja će se kao takva pojaviti unutar generiranog sažetka. Značajke koje su ključne za ovakve algoritme mogu varirati ovisno o sofisticiranosti primjene i kolika je njihova moć izražavanja. Također te iste značajke se pojavljuju u određenoj frekventnosti unutar danoga teksta, što pridonosi redundanciji. Frekvenciju je moguće dobit na više načina, pri čemu imamo prije spomenutu TF-IDF vjerojatnost pojavljivanja riječi i LLR (eng. *log-likehood ratio*) test koji pomaže odlučivanju znakovitosti riječi u odnosu na temu teksta (Nenkova i McKeown, Automatic Summarization 2011).

Ovdje se susrećemo s prvom preprekom tijekom računanja frekvencije, a to je pojavljivanje različitih leksičkih kategorija kao što su to sinonimi, zamjenice i slične pojavnice, koje se, bez prilagodbe algoritma, računaju kao nove riječi, umjesto da se njihovo pojavljivanje nadodaje na vrijednost postojeće riječi. Kako bi se riješila navedena prepreka, stvoreni su sustavi rješavanja koreferenci postojećih leksičkih resursa, koji su stvoreni ili automatski ili ručno te kao takvi sadrže informacije o srodnim riječima. Kada govorimo o ovakvim situacijama, tada je potrebno utvrditi važnost u odnosu na koncepte ili leksičku povezanost umjesto baziranja na temelju riječi. Kako bismo postigli navedeno potrebno je napraviti apstraktnu semantičku analizu teksta, kako bismo pokrili leksičku frekvenciju i leksičke odnose među riječima, promatrajući koje riječi se ponavljaju unutar velikog broja različitih tekstova određene tematike. Jedan od načina rješavanja problema jest korištenje različitih pristupa strojnog učenja.

2.1 Metode nenadziranog učenja

Postoji nekoliko metoda nenadziranog učenja kojima se želi riješiti problem automatskog generiranja sažetaka. Svi sljedeći navedeni algoritmi ne zahtijevaju ljudsku intervenciju kako bi učili statistički model važnosti te im nije potrebna ljudska odluka o tome koja rečenica je

[1] Navedene značajke o kojima govorimo isključivo se oslanjaju na informacije koje su ulaz u algoritme sažimanja.

pogodna za generirani sažetak. Važno je napomenuti kako takvi algoritmi nemaju nikakvo prethodno lingvističko znanje te ne procesiraju tekstove na takvoj razini.

2.1.1 Algoritmi bazirani na frekvenciji riječi

Kao što je navedeno unutar 1.1 Metoda ekstrakcije postoji više načina kako pristupiti ovome problemu. Najstariji način rješavanja ovog problema (sa svim svojim prije spomenutim negativnostima) jest računanje vjerojatnosti pojavljivanja riječi u određenom kontekstu, pri čemu za neke riječi koje budu imale dosta veliku vjerojatnost pojavljivanja postoji mogućnost ne indiciranja važne informacije teme određenog dokumenta ili skupine dokumenata s istom temom. Ovdje se računa TF-IDF težinsku vrijednost riječi, pri čemu tako rezultira točnijim vjerojatnostima riječi koje imaju bolju opisnu moć unutar teksta.

Računanje vjerojatnosti riječi jedan je od najjednostavnijih pristupa pri čemu se koristi učestalost pojavljivanja riječi kao indikatorom važnosti informacije[2]. Kako bismo izračunali vjerojatnost *p(w)* riječi *w* u obzir cijeli tekst te izračunati koliko često se pojavljuje riječ *w* u dokumentu ili nekom skupu dokumenata na određenu tematiku. Broj pojavljivanja označit ćemo s *c(w)* dok cjelokupni skup riječi unutar našeg ulaza predstavljamo s *N*, tada naša formula za vjerojatnost glasi:

$$p(w) = \frac{c(w)}{N} \qquad 1)$$

Kada gledamo vjerojatnost distribucije riječi, vjerojatnost sažetka je moguće izračunati na temelju multinomijalne distribucije[3]:

$$L[sum] = \frac{M!}{n_1!..n_r!} p(w_1)^{n_1} \dots \dots p(w_r)^{n_r} \qquad 2)$$

gdje s M predstavljamo ukupan broj riječi sažetka $n_1 + \dots . + n_r = M$, te svaki i n_i predstavlja broj pojavljivanja riječi w_i u sažetku te $p(w_i)$ vjerojatnost pojavljivanja w_i u sažetku procijenjenom na temelju ulaznih dokumenata (Nenkova i McKeown, Automatic Summarization 2011).

---

[2] Samo računanje frekvencije bi bilo jednostavnije, međutim korištenjem takvog pristupa dolazimo do problema ovisnosti o duljini dokumenta od kojeg generiramo sažetak, te stoga koristimo 1) i 2) kako bismo izračun prilagodili duljini dokumenta gdje određena frekventnost pojavljivanja riječi može rezultirati znatno lošije.

[3] Multinomijalna distribucija predstavlja generaliziranu binominalnu distribuciju, tj. njen nastavak kada govorimo o situacijama gdje svaki test ima $k \geq 2$ mogućih rezultata. Ako imamo n mogućih testova, u svakom od kojih postoji vjerojatnost međusobno isključivih događaja $E_1, \dots E_k$ te za svaki eksperiment E_j postoji vjerojatnost π_j pri čemu $\pi_1 + \pi_2 + \pi_k = 1$.

Kako bi se dobili još bolji rezultate, moguće je izračunati vjerojatnost pojavljivanja riječi kako bismo dobili sliku koje riječi su bitne, pa stoga za svaku rečenicu S_j unutar teksta pridodajemo težinsku vrijednost jednaku prosječnoj vjerojatnosti riječi u rečenici $p(w_i)$ što činimo na sljedeći način:

$$Weight(S_j) = \frac{\sum_{\omega_{i \in S_j}} p(w_i)}{|\{w_i | w_i \in S_j\}|} \quad 3) \ .$$

Potom se odabire ona rečenica koja ima najveći rezultat te sadrži riječ koja trenutno ima najveću vjerojatnost pod pretpostavkom kako je to riječ koja predstavlja najvažniju tematiku ulaznog dokumenta, kao željeni rezultat uzimajući rečenice koje najbolje pokrivaju tu riječ. Nakon što je odabrana najbolje rečenica, postavlja se manja vjerojatnost kako bi se vjerojatnost one riječi koja se pojavljuje više puta unutar teksta bila manja od riječi s jednom pojavnicom unutar teksta tijekom ponavljanja ovog postupka do željene dužine sažetka.

Alternativno, moguća je primjena TF-IDF težinskih vrijednosti. Problem takvog pristupa jest često pojavljivanje malog broja riječi te rijetko pojavljivanje većeg broja riječi, što najbolje opisujemo pomoću Zipfove distribucije[4]. Najčešće riječi koje ne doprinose važnosti teme sažetka primjerice zamjenice, veznici, u stranim jezicima poput njemačkog članovi imenica i sl. također zovemo i zaustavnim riječima (na eng. *stop words*) (Nenkova 2006). Potrebno je odrediti postotak tih riječi koji moramo ukloniti kako bismo dobili dobre rezultate u odnosu na određeni broj dokumenata i njihovu različitu dužinu. Kako bi se postiglo navedeno, definirane je lista najučestalijih stop-riječi u odnosu na jezik i domenu te riječi imaju iznimno malu ili nikakvu ulogu nositelja značajnosti rečenice u kojoj se nalaze. Ovdje u igru dolazi TF-IDF težinska vrijednost koja mjeri na cijeloj zbirci tekstova iz istog područja kao i dokument koji se skraćuje. Ovdje zbirka tekstova ima ulogu predstavljanja okvirne slike učestalosti pojavljivanja riječi unutar zadane teme te njihove okvirne uloge u opisnoj važnosti te tematike. Pri tome je potrebno odrediti frekventnost pojavljivanja termina $c(w)$, koja je potrebna za

[4]Zipfova distribucija sortira riječi prema silaznoj učestalosti pojavljivanja riječi, pri čemu bi riječ s najvećim brojem pojavljivanja dobila najviši rang. U statističkom pogledu, definiramo li Zipfov rang sa z, funkcija učestalosti pojavljivanja $f_z(z, N)$ mora zadovoljavati sljedeća dva uvjeta:

- z: Zipfov rang u listi riječi je poslože u odnosu na padajuću vrijednost
- $f_z(z, N)$: frekvencija je uzorak N tokena riječi sa Zipfovim rangom z.

Prema tome, Zipfova distribucija rang-frekvencije je inverzna empirijskom strukturalnom tipu distribucije: $g(m, N) = z \leftrightarrow f_z(z, N) = m$.
Generalno, riječ s frekvencijom m ima Zipfov rang z, tada poredak frekvencija Zipfovog ranga govori kako postoji z riječi s barem frekvencijom m, koja unazad ukazuje kako je $g(m, N) = z$ (Baayen 2001).

procesiranje težine riječi w unutar teksta koji je potrebno skratiti, je i broj dokumenata $d(w)$ u zbirci tekstova od D dokumenata koji sadrže zadanu riječ. Prema tome formula glasi:

$$TF^{*}IDF_{w} = c(w) \times \log\frac{D}{d(w)} \qquad 4)$$

Ovdje se primjećuje kako opisne riječi domene su one koje se često pojavljuju unutar ulaznog dokumenta, međutim kao takve se ne pojavljuju često unutar drugih dokumenata naše zbirke tekstova. Njihova IDF težinska vrijednost će biti veća za razliku od stop-riječi, čija IDF vrijednost će biti jako blizu nuli (Nenkova i McKeown, Automatic Summarization 2011).

Logaritamski omjer vjerojatnosti je još bolji pristup prepoznavanja riječi koje najbolje opisuju zadanu temu. Takve riječi najčešće zovemo i potpisnicima teme (na eng. *topic signatures*) (Nenkova i McKeown, Automatic Summarization 2011)te se kao takvi pojavljuju često unutar ulaznog teksta, međutim su rijetki unutar drugih članaka. Za razliku od TF-IDF-a, ovaj način omogućuje podjelu riječi teksta u opisne i one koje to nisu.

Informacija učestalosti pojavljivanja riječi u velikom pozadinskom skupu tekstova je potrebno procesirati statistiku na temelju koje se određuju riječi specifične za temu. Vjerojatnost ulaza *I* te pozadinske zbirke se računa na temelju dvije pretpostavke:

- (H1) vjerojatnost riječi unutar ulaznog teksta je jednaka vjerojatnosti pojavljivanja riječi unutar zbirke tekstova

H1: $P(w|I) = P(w|B) = p$ (*w* nije opisna riječ),

- (H2) riječ ima drukčiju, višu vjerojatnost pojavljivanja u ulaznom tekstu nego u zbirci tekstova

H2: $P(w|I) = p_I \; i \; P(w|B) = p_B \; i \; p_I > p_B$ (w je opisna riječ)

Vjerojatnost teksta, u odnosu na riječ *w*, procesira se na temelju binomialne distribucije. Ulazni tekst i pozadinski skup tekstova se tretira ka niz riječi $w_i: w_1 w_2 w_3 \ldots w_N$. Pojavljivanje svake riječi je zapravo Bernoullijeva jednadžba s vjerojatnosti uspjeha *p* koja se pojavljuje kada je $w_i = w$. Cjelokupna vjerojatnost u odnosu na riječ *w* koja se pojavljuje *k* puta u *N* pokušaja dobivaju binominalnu distribuciju

$$b(k, N, p) = \binom{N}{k} p^{k}(1-p)^{N-k} \qquad 4.5).$$

Vjerojatnost za H1 se procesira na temelju ulaznog teksta te skupa tekstova zajedno, dok, s druge strane, vjerojatnost za H2 se računa tako da se p_1 procesira na temelju ulaznih teksta, a

p_2 na temelju skupa tekstova. Tada je vjerojatnost cjelokupnog skupa podataka jednaka umnošku binominala ulaznog teksta i binominala skupa tekstova. Stoga, na temelju prethodnih jednadžbi, omjer vjerojatnosti je definiran:

$$\lambda = \frac{b(k, N, p)}{b(k_I, N_I, p_I) \cdot b(k_B, N_B, p_B)} \qquad 4.6).$$

pri čemu se iteracije indeksa *I* procesiraju jedino iz ulaznog teksta, a one s indeksom *B* procesirane iz skupa tekstova. Navedeno je također primjenjivo ne samo za binomialne distribucije, već i na multinomijalne distribucije te je u tim situacijama potrebno koristiti duple indekse i kratice:

$$P_i = p_{1i}, p_{2i}, \quad \ldots p_{ji}, \ldots,$$
$$K_i = k_{1i}, k_{2i}, \quad \ldots k_{ji}, \ldots,$$
$$Q = q_{1i}, q_{2i}, \quad \ldots q_{ji}, \ldots,$$

kako bi se mogli definirati produkti varijabli poput formule 4.6).

Iz formule 4.6), uzimajući logaritam omjera vjerojatnosti u obzir, dobivamo sljedeće rješenje:

$$-2\log\lambda = 2[\log b(k_1, N_1, p_1) + \log b(k_2, N_2, p_2) - \log b(k_1, N_1, p_1) - \log b(k_2, N_2, p_2)] \qquad 4.7).$$

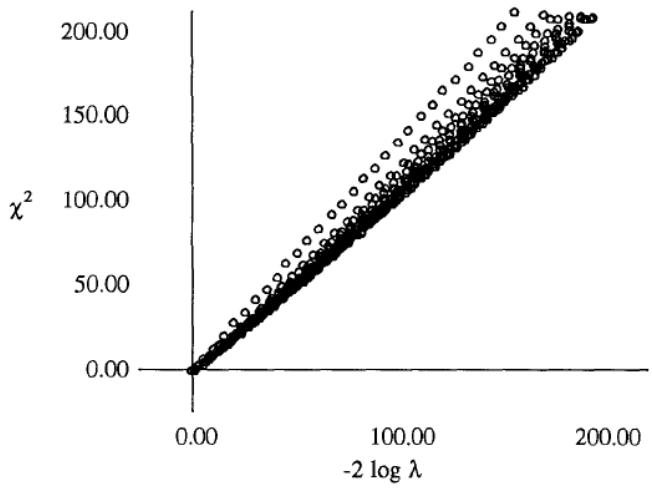
pri čemu vrijedi:

$$p_{ij} = \frac{k_{ij}}{\sum_i k_j} \qquad 4.8).$$

I

$$q_{ij} = \frac{\sum_i k_{ij}}{\sum_{ij} k_{ij}} \qquad 4.8).$$

Na temelju prethodnog, navodi se kako $-2\log\lambda$ tada ima poznatu vjerojatnost distribucije x^2 te ju je moguće iskoristiti pri odlučivanju koje riječi najbolje opisuju i predstavljaju zadanu temu (Dunning 1994).



Slika 0-1: Rezultati analize teksta od 30000 riječi metode binomialne logaritamske vjerojatnosti u usporedbi s Pearsonovim x^2 testom na temelju teksta o švicarskoj Union banci .Do divergencije između navedene dvije metode dolazi kada je k_1 veći od vrijednosti očekivane na temelju proučavane vrijednosti k_2 (Dunning 1994).

2.1.2 Metode rangiranja rečenica na bazi grafova

Ove metode najčešće koriste ponavljanje koje je vidljivo unutar ulaznog dokumenta, i to na razini riječi i rečenica. Kažemo da su dvije rečenice slične kada sadrže iste te njihovo često pojavljivanje dokazuje i povezuje međusobnu značajnost tijekom generiranja sažetka. Ovaj pristup na tako dobiva sve prednosti koje ima računanje frekventnosti riječi te prednosti metoda grupiranja rečenica, pri čemu se stvara model procesiranja važnosti rečenica.

U ovim pristupima (Dunning 1994) se ulazni tekst predstavlja kao visoko povezani graf. Rečenice su vrhovi takvih grafova, dok su bridovi koji povezuju rečenice pak dodijeljene težinske vrijednosti jednake sličnosti dviju povezanih rečenica. Kako bi se izračunale težinske vrijednosti sličnosti među rečenicama koriste se TF-IDF težinske vrijednosti riječi. Vrijednosti tih rubova će biti veće što je veći broj zajedničkih leksika. Osim navedenog načina, povezanost među rečenicama je također moguće utvrditi i binarnim načinom, a to je povezanost vrhova samo u slučaju kada je sličnost dvaju rečenica nadilazi unaprijed definirani prag. To čini cijelu strukturu grafa znatno prilagodljivijim u odnosu na metodu grupiranja rečenica. Ovdje su jasno vidljive usko povezane skupine unutar grafa, međutim, rečenice nisu pripadnošću ograničene na isključivo jednu skupinu (Nenkova, Summarization Evaluation for Text and Speech: Issues and Approaches 2006).

Značajnost rečenice, odnosno koliko je rečenica koja predstavlja vrh grafa centralni pojam teksta, moguće je primjerice izračunati pomoću PageRanka[5], pri čemu, nakon normalizacije težinskih vrijednosti bridova u svrhu postizanja distribucije vjerojatnosti čiji zbroj svih bridova iz jednog vrha je jednak jedan, graf postaje Markov lanac, a težinske vrijednosti bridova predstavljaju vjerojatnost prelaska između stanja. Vjerojatnost dolaska u stanje vrha grafa u određenom vremenu *t* te sa što većim brojem prijelaza, dolazi do izmjene vjerojatnosti svakog vrha. Takvu situaciju također nazivamo i stacionarnom distribucijom Markovog lanca[6] (na eng. *stationary distribution of the Markov chain)* te ju je moguće izračunati iterativno (Nenkova i McKeown, Automatic Summarization 2011).

Također jedna od metoda na temelju grafa je i *Selectivity Based Keyword Extraction* (Beliga, Martinčić-Ipšić i Meštrović 2016) koja pronalazi ključne riječi iz teksta reprezentiranog u obliku mreže. Vjerojatnost odabira čvora unutar mreže težinskih vrijednosti se računa kao prosječna vjerojatnost podijeljena na veze jednog čvora kako bi se utvrdilo koje riječi su najbitnije za izvlačenje.

Neka je *N* broj čvorova unutar mreže, a *M* broj veza, u mreži težinskih vrijednosti, svaka veza ima svoju vjerojatnost w_{ij}, pri čemu *i* i *j* predstavljaju povezane čvorove. Stupanj čvora k_i je određeni broj veza na čvoru pri čemu, unutar usmjerenih mreža, postoji stupanj ulaznih i izlaznih mreža $k_i^{in/out}$ (Beliga, Martinčić-Ipšić i Meštrović 2016) koji predstavljaju određeni broj susjednih čvorova na ulazi i izlazu u odnosu na čvor *i*. Kako bi se definirao stupanj važnosti, pri tome uzimajući u obzir ulazi i izlazni stupanj:

$$dc_i^{in/out} = \frac{k_i^{in/out}}{N-1} \qquad 5) \quad .$$

Samim time važno je također uzeti u obzir i koliko je termin približno značajan. Kako bismo odredili navedeno računamo centralnost čvora (na eng. *closeness centrality*) (Beliga, Martinčić-Ipšić i Meštrović 2016) kao sumu najkraćih sličnosti između čvorova. Ako najkraću sličnost puta između čvorova *i* i *j* predstavljamo s d_{ij}, tada definiramo formulu centralnosti čvora *i* kao:

[5] Algoritam prethodno osmišljen za rangiranje web stranica na Google pretraživaču, sada korišten za mnoge druge primjene. Važnost je tim veća, što je veći dodijeljeni postotak određenom vrhu grafa.

[6] Statistički model koji pobliže opisuje niz mogućih događaja u kojem vjerojatnost svakog događaja ovisi isključivo o stanju prethodnog događaja. Niz prebrojive beskonačnosti lanca u kojem se pomaci rade u vremenski diskretnim koracima daje Markov lanac diskretnog vremena (DTMC) dok proces kontinuiranog vremena daje Marko lanac kontinuiranog vremena (CTMC) (Gagniuc 2017).

$$cc_i = \frac{N-1}{\sum_{i \neq j} d_{ij}} \quad 6) \quad .$$

Na tom najkraćem putu susrećemo i druge čvorove koji imaju ulogu mosta preko kojeg se dolazi do drugog čvora. Ako je $\sigma_{jk}(i)$ broj puteva koji prolaze vrhom grafa *i*, tada broj takvih puteva bc_i računamo na sljedeći način:

$$bc_i = \frac{\sum_{i \neq j \neq k} \frac{\sigma_{jk}(i)}{\sigma_{jk}}}{(N-1)(N-2)} \quad 7)$$

Međutim SBKE (Beliga, Martinčić-Ipšić i Meštrović 2016) nije tipična metoda grafa, već se vrši odabir čvorova unutar mreže na temelju procijenjene težinske vrijednosti koju, unutar usmjerene mreže, računamo:

$$s_i^{in/out} = \sum_j w_{ji/ij} \quad 8)$$

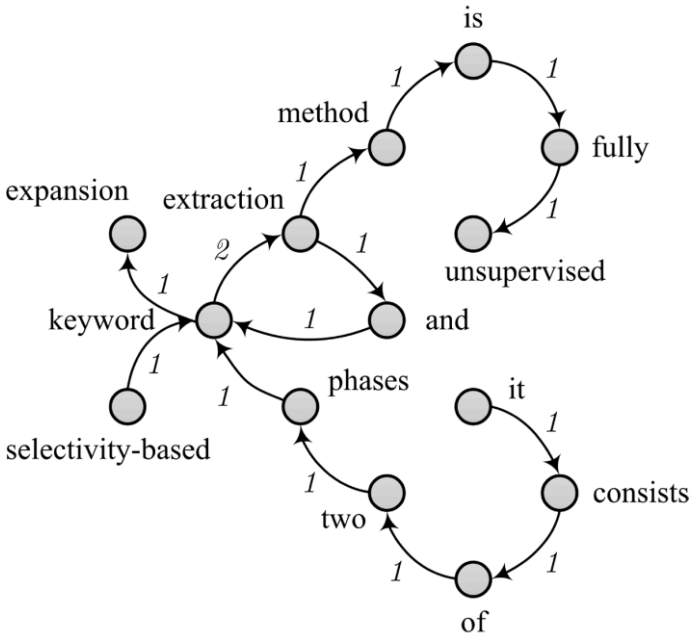
te, uvrštavanjem prethodnog rezultata, računamo vjerojatnost odabira unutar usmjerene mreže računamo na sljedeći način:

$$e_i^{in/out} = \frac{s_i^{in/out}}{k_i^{in/out}} \quad 9)$$

Kako bi se utvrdio stupanj odabira, prema (Opsahl, Agneessens i Skvoretz 2010), koristi se pozitivan parametar α koji je moguće izračunati na sljedeći način:

$$ge_i^{\alpha_{in/out}} = k_i^{in/out} \left(\frac{s_i^{in/out}}{k_i^{in/out}} \right)^{\alpha} \quad 10)$$

što omogućuje prilagodbu relativne važnosti u odnosu na broj veza i težinskih vrijednosti vezanih uz navedeni vrh grafa (Beliga, Martinčić-Ipšić i Meštrović 2016).



Slika 0-2: Primjer usmjerene mreže s vjerojatnostima pojavljivanja metode SBKE (Beliga, Martinčić-Ipšić i Meštrović 2016)

2.1.3 LexRank i uloga u sažimanju tekstova

Poput SBKE (Beliga, Martinčić-Ipšić i Meštrović 2016), LexRank je također metoda grafa za sažimanje tekstova. Poput prethodnih algoritama, LexRank (Erkan i Radev 2004.) se također oslanja na statističku pozadinu prepoznavanja važnosti riječi i rečenica u zadanoj tematici teksta. Ranija istraživanja su bazirana na osobinama rečenica poput pozicije unutar teksta, cjelokupne frekvencije riječi ili pak ključnih fraza koje ukazuju na važnost rečenice na temelju formule na stranici 16. Također je riječ o centralnosti rečenica kako bi se iz skupa mogle izlučiti one bitne za sažetak.

Pomoću definiranja važnosti rečenice nastoji se skupiti što više informacija vezano uz glavnu temu skupine tekstova kako bi se napravio dobar odabir podskupa rečenica iz originalnog teksta. Govoreći o centralnosti, najčešće se misli o samom značaju riječi sadržanih unutar rečenice te kao jedan od češćih pristupa problemu je definiranje samog centra u vektorskom prostoru skupine dokumenata. Takav centar je najčešće pseudo dokument sastavljen od riječi s TF-IDF vrijednostima iznad definiranog praga, pri čemu TF predstavlja učestalost pojavljivanja riječi unutar skupine, dok su IDF vrijednosti najčešće izračunate na puno većoj i sličnoj skupini podataka.

```
input  : An array S of n sentences, cosine threshold t
output: An array C of Centroid scores
Hash WordHash;
Array C;
/* compute tf×idf scores for each word */
for i ← 1 to n do
    foreach word w of S[i] do
        WordHash{w}{"tfidf"} = WordHash{w}{"tfidf"} + idf{w};
    end
end
/* construct the centroid of the cluster */
/* by taking the words that are above the threshold*/
foreach word w of WordHash do
    if WordHash{w}{"tfidf"} > t then
        WordHash{w}{"centroid"} = WordHash{w}{"tfidf"};
    end
    else
        WordHash{w}{"centroid"} = 0;
    end
end
/* compute the score for each sentence */
for i ← 1 to n do
    C[i] = 0;
    foreach word w of S[i] do
        C[i] = C[i] + WordHash{w}{"centroid"};
    end
end
return C;
```

Slika 0-3: Primjer algoritma koji računa vrijednosti centralnosti riječi (Erkan i Radev 2004.)

Kako bi se konkretno izračunala vjerojatnost važnosti rečenice, LexRank (Erkan i Radev 2004.) promatra dokumente kao mrežu međusobno povezanih rečenica. Pošto su neke rečenice sličnije jedne drugima, dok druge pak dijele samo poneke informacije s ostatkom, pretpostavlja se kako prva skupina ima veću ulogu pri opisivanju određene teme. U tom slučaju postoje dva vrlo bitna pitanja vezana uz definiranje važnosti rečenice, a to su samo definiranje sličnosti među rečenicama te način računanja centralnosti rečenice u odnosu na sličnost prema drugim rečenicama.

Kako bi se definirala sličnost između rečenica, koristi se princip vreće riječi (na eng. *bag-of-words*) (Erkan i Radev 2004.) čime je svaka rečenica predstavljena kao vektor s *N* dimenzija, pri čemu je *N* broj mogućih riječi. Za svaku riječ u rečenici vrijednost odgovarajuće dimenzije unutar vektora koji predstavlja rečenicu je umnožak broja pojavljivanja riječi u rečenici i IDF-a riječi te je sličnost rečenica definirana kosinusom dviju odgovarajućih vektora:

$$idf - modified - cos(x,y) = \frac{\sum_{w\in} tf_{w,x} tf_{w,y} (idf_w)^2}{\sqrt{\sum_{x_i \in x} (tf_{x_i,x} idf_{x_i})^2} \times \sqrt{\sum_{y_i \in y} (tf_{y_i,y} idf_{y_i})^2}} \qquad 11).$$

gdje $tf_{w,s}$ predstavlja koliko puta se riječ w pojavila u rečenici s. Skupinu dokumenata je moguće predstaviti matricom sličnosti kosinusa gdje svaki ulaz u matricu je sličnost između odgovarajućih parova rečenica.

Kako većina rečenica ima neku sličnost te je rijetkost da vrijednosti koje predstavljaju sličnost rečenica budu 0, postavlja se prag kako bi se odabrale one rečenice koje imaju veću vjerojatnost sličnosti. Tako se stvara graf u kojem svaka rečenica skupine je čvor, a sve rečenice sa značajnom sličnosti su međusobno povezane (Erkan i Radev 2004.).

Pri računanju centralnosti kod LexRank-a (Erkan i Radev 2004.) bridove se promatra kao načinom ravnopravnog glasanja kako bi se odlučila cjelokupna vrijednost važnosti svakog čvora. Međutim, postoji mogućnost u kojoj rečenice međusobno "glasaju" jedna za drugu te tako smanjuju kvalitetu sažetka radi nepotrebnog međusobnog povećavanja važnosti. Stupanj važnosti može u nekim situacijama negativno utjecati na kvalitetu sažetka te kako bi se spriječilo navedeno, u obzir se uzima od kuda je došao glas te koliko je rečenica koja "glasa" važna za temu, automatski dajući veće značenje glasanja onih rečenica koje imaju veću ulogu u tekstu. To znači da svaki čvor ima određenu vrijednost koja predstavlja kontekstualni značaj čvora koju reproducira na susjedne čvorove te se predstavlja kao:

$$p(u) = \sum_{v \in adj[u]} \frac{p(v)}{\deg(v)} \qquad 12).$$

pri čemu $p(u)$ predstavlja centralnost čvora u, $adj[u]$ je skup susjednih čvorova naprama u, a $\deg(v)$ predstavlja stupanj čvora v. Pošto je ovdje zapravo riječ o matricama sličnosti koje zapravo predstavljaju Markov lanac, potrebno ih je zadržati nesvodivima i planarnima. Kako bismo postigli navedeno, potrebno je odrediti malu vjerojatnost prijelaza na bilo koji čvor u grafu, efektivno definirajući PageRank (Erkan i Radev 2004.) na temelju jednadžbe 12):

$$p(u) = \frac{d}{N} + (1-d) \sum_{v \in adj[u]} \frac{p(v)}{\deg(v)} \qquad 13).$$

gdje N predstavlja ukupan broj čvorova u grafu, a d faktor prigušenja u intervalu $[0.1, 0.2]$. Navedeno je također moguće predstaviti u obliku matrice kao:

$$p = [dU + (1-d)B]^T p \qquad 14).$$

pri čemu U je matrica s elementima jednakima $1/N$. Prijelaz $[dU + (1-d)B]$ regulirajućeg Markovljevog lanca je kombinacija matrica U i B, gdje nasumičnim kretanjem kroz

Markovljev lanac se odabire najbliže stanje trenutnom stanju u vjerojatnosti $1-d$ ili se napravi prijelaz u bilo koje stanje u grafu, uključujući trenutno stanje s vjerojatnosti d.

Kako bi se postigli još bolji rezultati, moguće je iskoristiti činjenicu koliko jako su dvije rečenice međusobno povezane. Uključivanjem kosinusa u formuli 11) stvara se graf sličnosti koji je nešto veće gustoće, te je moguće normalizirati retke sume odgovarajućih prijelaza matrica, pri čemu dobivamo sljedeću jednadžbu:

$$p(u) = \frac{d}{N} + (1-d) \sum_{v \epsilon adj[u]} \frac{idf - modified - cos(u,v)}{\sum_{z \in adj[v]} idf - modified - cos(z,v)} p(v) \qquad 15).$$

Prednosti centralnosti riječi u metodama grafa su mnogobrojne:

- Uzima se u obzir informacija podpretpostavke između rečenica te, ako je informacija koju ta rečenice prenosi je u podređenom položaju u odnosu na drugu rečenicu unutar skupa, uključuje se ona koja ju nadjačava.
- Stupanj čvora unutar grafa kosinusne vrijednosti je pokazatelj koliko informacija dijeli rečenica s drugim rečenicama.
- Sprječava se pojavljivanje ne prihvatljivo visokih IDF vrijednosti kako bi se povisila važnost rečenice koja nije usko vezana uz temu sažetka (Erkan i Radev 2004.).

```
MInput An array S of n sentences, cosine threshold t  output: An array L of LexRanl
        scores
Array CosineMatrix[n][n];
Array Degree[n];
Array L[n];
for i ← 1 to n do
    for j ← 1 to n do
        CosineMatrix[i][j] = idf-modified-cosine(S[i],S[j]);
        if CosineMatrix[i][j] > t then
            CosineMatrix[i][j] = 1;
            Degree[i] + +;
        end
        else
            CosineMatrix[i][j] = 0;
        end
    end
end
for i ← 1 to n do
    for j ← 1 to n do
        CosineMatrix[i][j] = CosineMatrix[i][j]/Degree[i];
    end
end
L = PowerMethod(CosineMatrix,n,ϵ);
return L;
```

Slika 0-4:Algoritam izračuna LexRank rezultata (Erkan i Radev 2004.)

3 Algoritmi mjere sličnosti rečenica

Osim navedenih načina određivanja sličnosti rečenica u (Beliga, Martinčić-Ipšić i Meštrović 2016), (Erkan i Radev 2004.) i (Nenkova i McKeown, Automatic Summarization 2011) postoji još nekoliko algoritama mjerenja preklapanja rečenica u algoritmima koji generiraju sažetak na temelju odnosa rečenica unutar grafa. Tri su najpoznatije mjere u ovom slučaju:

- Jaccardova mjera sličnosti rečenica
- Mjera podudarnosti kosinusa
- algoritam predložen u (Mihalcea, Corley i Strapparava, Corpus-based and knowledge-based measures of text semantic similarity 2006) koji uzima u obzir semantičku podudarnost tekstova.

3.1 Jaccardova sličnost

Također poznat kao i Jacccardov indeks je algoritam koji mjeri podudarnost skupova podataka. Originalno algoritam je razvio Paul Jaccard te algoritam uzima u obzir omjer presjeka podijeljen unijom istih skupova podataka:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \qquad 16).$$

pri čemu vrijedi $0 < J(A,B) < 1$. Ako su A i B prazni skupovi, tada vrijedi $J(A,B) = 1$.

S druge strane Jaccardova sličnost mjeri koliko je odstupanje između dva skupa podataka te je dodatak Jaccardovom koeficijentu. Kako bismo izračunali Jaccardovu sličnost potrebno je oduzeti Jaccardov koeficijent od 1 ili pak podijeliti razliku veličina unije i presjeka dva skupa s veličinom unije skupova:

$$d_J(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|} \qquad 17).$$

Alternativno, Jaccardova sličnost se interpretira također kao i omjer veličine simetrične razlike :

$$A \Delta B = |A \cup B| - |A \cap B| \qquad 18).$$

Često se koristi kako bi se izračunala $N \times N$ matrica za grupiranje i multidimenzionalno skaliranje uzoraka skupova. Kako se ova mjera podudarnosti još poznata kao i Tanimotova sličnost i sličnost, prema (Moulton i Jiang 2018) ovdje se otvaraju dodatne mogućnosti definiranja sličnosti kada je Jaccardova podudarnost predstavljena kao bit vektor

$$f(A,B) = \frac{A \cdot B}{\|A\|^2 + \|B\|^2 - A \cdot B} \quad 19).$$

gdje je izračun predstavljen u obliku skalarnog umnoška vektora i magnitude. Pretpostavka se oslanja na činjenicu kako, gdje vrijednost svake dimenzije je ili 1 ili 0, vrijedi:

$$A \cdot B = \sum_i A_i B_i = \sum_i (A_i \wedge B_i) \quad .$$

i

$$\|A\|^2 = \sum_i {A_i}^2 = \sum_i A_i \quad .$$

što dodatno zbunjuje pojašnjenje definicije u ovom slučaju, pogotovo kada je funkcija definirana pomoću vektora više generalna, osim u slučaju kada je domena prethodno točno definirana.

3.2 Mjera sličnosti kosinusa

Jedan primjer kosinusne sličnosti je moguće vidjeti u 2.1.2 Metode rangiranja rečenica na bazi grafova, jednadžba 11). Ona služi kako bi se mjerila sličnost između dva ne-nulta vektora te se definira kosinusom kuta između navedenih vektora. Na temelju toga se odlučuje pokazuju li vektori relativno u istom smjeru.

Na ovaj način dokumenti mogu biti predstavljeni kroz mnogo atributa, gdje svaki pohranjuje frekvenciju određene riječi ili fraze unutar dokumenta. Zbog toga je svaki dokument predstavljen kao objektom poznatim još kao i vektorom frekvencije termina (na eng. *term-frequency vector*) (Han, Kamber i Pei 2011) koji sam po sebi zna biti dug te sadržavati mnogo vrijednosti 0.

Funkcija podudarnosti kosinusa služi za usporedbu više dokumenata ili pak rangiranje tih dokumenata u odnosu na zadani vektor upitnih riječi. Kako bi bilo razumljivije, (Han, Kamber i Pei 2011) definiraju x i y vektore za usporedbu te uz pomoć mjere sličnosti kosinusa, funkcija glasi:

$$sim(x,y) = \frac{x \cdot y}{\|x\|\|y\|} \quad 20).$$

pri čemu $\|x\|$ je euklidska norma vektora $x = (x_1, x_2, \ldots, x_p)$ definirana korijenom zbroja kvadrata $\sqrt{{x_1}^2 + {x_2}^2 + \cdots + {x_p}^2}$ te također isto i za vektor y. Ovime mjera računa kosinus kuta između navedenih vektora, što u slučaju kada je vrijednost 0 znači da su vektori na 90° jedan

od drugoga te nemaju nikakve zajedničke osobine. Suprotno, što je vrijednost bliža 1, kut je manji te je sama podudarnost vektora veća. Ova pravila vrijede za bilo koji broj dimenzija te je odlična mjera u ovom slučaju, jer je svaki termin dodijeljen drugoj dimenziji te je dokument okarakteriziran vektorima gdje vrijednost svake dimenzije odgovara broju pojavljivanja termina unutar teksta. Temeljem (Han, Kamber i Pei 2011), dolazimo do jednadžbe 20) koju je dalje moguće zapisati kao:

$$sim(x, y) = \frac{\sum_i x_i \cdot y_i}{\sqrt{\sum_i (x_i)^2} \cdot \sqrt{\sum_i (y_i)^2}} \quad 21).$$

iz čega je moguće vidjeti sličnost definicija iz jednadžbi 11).

Naknadno, važno je dotaknuti se i mjere sličnosti kosinusa koja se koristi kako bi se nadopunio pozitivni prostor:

$$D_C(A, B) = 1 - S_C(A, B) \quad 22).$$

gdje D_C predstavlja kosinus sličnosti, a S_C mjeru kosinusa podudarnosti.

Ova mjera ima i nekoliko dobih svojstava. Prvenstveno podudarnost kosinusa predstavlja relativnu usporedbu pojedinačnih vektorskih dimenzija, gdje za svaku konstantu α i vektor *V*, vektor i umnožak konstante i vektora posjeduju maksimalnu podudarnost.

3.3 Kullback-Leiblerova divergencija

U (Kullback i Leibler 1951) opisuje se mjera razlike između vjerojatnosti distribucija. Neke o primjena također uključuju Shannonovu entropiju u informacijskim sustavima te mnoge druge. Pretpostavimo da su *P* i *Q* vjerojatnosti distribucija, pri čemu *P* predstavlja podatke, promatranja ili pak vjerojatnost distribucije dok *Q* predstavlja teoriju, model opis i li aproksimaciju od *P*. Kullback-Leibler divergencija je u tom slučaju predstavlja prosječnu razliku broja bitova potrebnih za kodiranje uzorka od *P* koristeći optimizaciju koda za *Q* radije u odnosu na onog optimiziranog za *P*.

Prema (MacKay 2003) te distribucije *P* i *Q* se definiraju na istom vjerojatnosnom prostoru *X* kao:

$$D_{KL}(P \| \mathrm{Q}) = \sum_{x \in X} P(x) \log\left(\frac{P(x)}{Q(x)}\right) \quad 23).$$

što je ekvivalentno:

$$D_{KL}(P\|\mathrm{Q}) = -\sum_{x\in X} P(x)\log\left(\frac{Q(x)}{P(x)}\right) \qquad 24).$$

To znači kako se očekuje logaritamska razlika između P i Q pri čemu se koristi vjerojatnost P te ono što se tvrdi za P jest da je P apsolutno kontinuiran[7] u odnosu na Q Svaki puta kada je $P(x)$ jednak nuli, cjelokupni doprinos je jednak nuli, jer $\lim_{x\to 0^+} x\log x = 0$.

(Bishop 2021) naknadno opisuje kako za distribucije P i Q kontinuiranih slučajnih varijabli[8], njihova entropija se definira:

$$D_{KL}(P\|\mathrm{Q}) = \int_{-\infty}^{\infty} p(x)\log\left(\frac{p(x)}{q(x)}\right)dx \qquad 25).$$

pri čemu p i q predstavljaju gustoću vjerojatnosti skupova P i Q. (Kullback i Leibler, On Information and Sufficiency 1951) pojašnjavaju navedeno na primjeru u kojem su zadane u Tablica 1: Primjer distribucija zadan u distribucije za P i Q.

x	0	1	2
Distribucija $P(x)$	$9/25$	$12/25$	$4/25$
Distribucija $Q(x)$	$1/3$	$1/3$	$1/3$

Tablica 1: Primjer distribucija zadan u (Kullback 1959)

Slika 0-1: Prikaz rezultata zadanih tablicom prikazuje rezultate iz tablice. Prema primjeru P i Q su distribucije zadane tablicom. Na slici je vidljivo kako je P binomialna distribucija[9] dok je Q više uniformna distribucija. Primijer računanja i dobivanja rezultata na temelju podataka iz tablice izgleda ovako:

$$D_{KL}(P\|Q) = \sum_{x\in X} P(x)\ln\left(\frac{P(x)}{Q(x)}\right)$$

[7] Ovdje je riječ o svojstvu glatkoće formule koja je jača od same kontinuiranosti. Funkcija prestaje biti apsolutno kontinuirana kada nije zadovoljen uvjet uniformne kontinuiranosti, odnosno onog trena kada funkcija više ne garantira da će *f(x)* i *f(y)* biti toliko blizu jedna drugoj, pod uvjetom da su *x* i i *y* međusobno blizu jedno drugome. Neki od primjera kada funkcija prestaje biti apsolutno kontinuirana su $\tan(x)$ u intervalu $[0, {}^{\pi}/_{2})$, x^2 nad cijelim skupom realnih brojeva te $\sin({}^{1}/_{x})$ u intervalu $[0, 1)$.

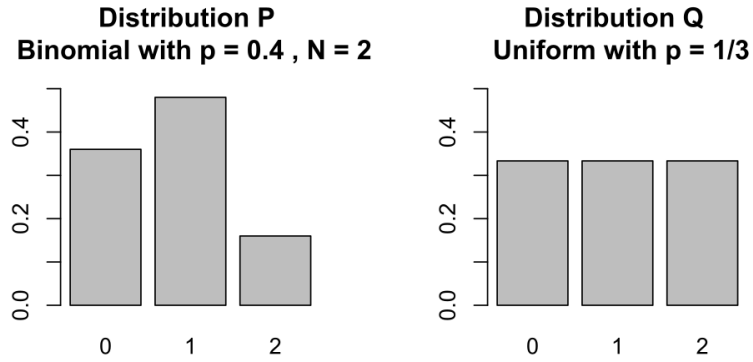
[8] Riječ je kontinuiranoj vjerojatnosti distribucije za vjerojatnosti distribucija kojima je baza neprebrojivi skup, unikatno karakteriziran funkcijom kumulativne distribucije koja računa vjerojatnost za svaki podskup baze.

[9] U statistici, binomialna distribucija predstavlja diskretnu distribuciju vjerojatnosti broja uspješnih nizova *n* nezavisnih eksperimenata u kojem rezultat može biti pozitivan te imati vjerojatnost *p* ili negativan sa vjerojatnosti $q = 1 - p$. Niz eksperimenata također zovemo i Bernoullijevim procesom.

$$= \frac{9}{25}\ln\left(\frac{9/25}{1/3}\right) + \frac{12}{25}\ln\left(\frac{12/25}{1/3}\right) + \frac{4}{25}\ln\left(\frac{4/25}{1/3}\right) \approx 0{,}0852996$$

$$D_{KL}(Q\|P) = \sum_{x\in X} Q(x)\ln\left(\frac{Q(x)}{P(x)}\right)$$

$$= \frac{1}{3}\ln\left(\frac{1/3}{9/25}\right) + \frac{1}{3}\ln\left(\frac{1/3}{12/25}\right) + \frac{1}{3}\ln\left(\frac{1/3}{4/25}\right) \approx 0{,}097455$$



Slika 0-1: Prikaz rezultata zadanih tablicom (Kullback 1959)

Kao i kod svakog algoritma postoji određena količina informacija koja se izgubi te je potrebno izmjeriti na određeni način. Poznavajući formulu 23) moguće je izračunati koliki je gubitak podataka, jer je riječ o logaritamskoj razlici vjerojatnosti podataka i originalne distribucije s distribucijom aproksimacije. Važno je napomenuti kako ovdje nije riječ o jedinici sličnosti između distribucija budući kako ova mjera nije simetrična (Kullback-Leibler Divergence Explained 2017.).

3.4 Drugi algoritmi na bazi grafova

U (Mihalcea, Corley i Strapparava 2006) i (Corley i Mihalcea 2005) predstavljaju se nove mjere koje kombiniraju mjerenje sličnosti između riječi u mjeru semantičke podudarnosti tekstova. Unatoč tome što je za pravu semantičku analizu tekstova i njihove sličnosti potrebno uzeti u obzir odnos između riječi, kao prvi korak gleda se semantička sličnost između komponenata riječi radi jednostavnosti. Kako bi se postigao optimalan rezultat koriste se (Lesk 1986) (Lealcock i Chodorow 1998) (Wu i Palmer 1994) (D. Lin 1998) i (Jiang i Conrath 1998) mjere naziranih na implementaciji WordNeta. Radi veće preciznosti algoritma te dodijele većih ili

manjih težinskih vrijednost riječima, u obzir se također uzima i specifičnost riječi u korpusu koja se koristi naknadno mjerenju semantičkih utjecaja i njihove hijerarhije.

Nakon postavljanja modela semantičke analize podudarnosti riječi te utjecaja i specifičnosti riječi, navedeni postupci se kombiniraju u mjeru semantičke sličnosti tekstova na temelju uparivanja najpodudaranijih riječi te određujući njihovu sličnost pomoću odgovarajućeg rezultata specifičnosti. Upotrebnom usmjerene mjere podudarnosti, u (Corley i Mihalcea 2005) se određuje semantička sličnost dijela teksta T_i u donosu na dio teksta T_j, pri tom kombinirajući sličnost među riječima s njihovim indikatorima specifičnosti:

$$sim(T_i, T_j) = \frac{\sum_{pos}(\sum_{w_k \in \{WS_{pos}\}}(maxSim(w_k) * idf_{w_k}))}{\sum_{w_k \in T_{t_{pos}}} idf_{w_k}} \quad 26).$$

pri čemu WS_{pos} predstavlja klasu riječi u kojoj sve podudarnosti imaju vrijednosti između 1 i 0 za svaku imenicu ili glagol, s najvećom semantičkom sličnosti ($maxSim$), u odgovarajućem skupu imenica ili glagola. Za druge vrste riječi provjeravaju se leksičke podudarnosti te su uključene u odgovarajući skup vrste riječi u slučaju pronađene podudarnosti.

Također u (Mihalcea 2004) te (Mihalcea i Tarau 2004) definiraju se naknadno i istražuju različiti modeli poput TextRanka te drugih algoritama za izvlačenje rečenica baziranih na modelu grafa. Ovdje se uspoređuju HITS (*Hyperlinked Induced Topic Search*), funkcija eksponenta pozicije te PageRank.

3.3.1 HITS algoritam

Ovo je iterativni algoritam dizajniran za rangiranje web stranica u odnosu na njihov stupanj dolaznih poveznica te to definirao kao “autoritet“. S druge strane stranice koje imaju veliki broj izlaznih poveznica su definirane kao središte.

Kako bi se definirale navedene dvije vrste poveznica, stvara se graf koji prepoznaje poveznicu te računa izlazni i ulazni stupanj za svaku od poveznica. Pri tome se nastoji orijentirati na bitne stranice kako se ne bi dodatno trošili resursi te se u obzir uzimaju oni skupovi web stranica koji su relativno mali, a da pri tome sadrže većinu ili puno autoritativnih stranica koje su ključne. Kako bi se postiglo navedeno, odabiru se stranice najvećeg ranga te se te stranice postavljaju kao korijen koji zadovoljava uvijete odabira relativno malog skupa koji je bogat bitnim stranicama.

Stvaranjem navedenih manjih grafova, u (Kleinberg 1999) promatra se struktura grafa te se u obzir uzimaju ulazni stupnjevi te moguća preklapanja u skupovima stranica, dok se za stranice s ulogom odredišta više promatra izlazni stupanj. Ono što je moguće primijetiti u određenim situacijama jest kako dobre odredišne stranice upućuju prema dobrim stranicama uloge autoriteta te je također moguće primijetiti kako dobre autoritativne stranice. Iterativnim putem se postavljaju zasebni utezi za autoritativne i odredišne stranice.

3.3.2 TextRank

TextRank je standardna metoda bazirana na principu grafa, gdje se značaj vrha grafa određuje na temelju rekurzivno prikupljenih informacija u grafu. (Mihalcea i Tarau 2004) koriste sličan princip glasanja kao i (Erkan i Radev 2004.) u svom radu, gdje vrhovi glasaju svojom vezom sa drugim vrhom. Što se više vrhova veže, tj. glasa za isti vrh, važnost tog vrha raste. Kako bi se odredila neophodnost generalnog glasanja, dodatno se u obzir uzima značaj glasanja vrha te se ta informacija također uključuje u rangiranje.

Definira se graf $G = (V, E)$ koji je usmjeren graf sa skupom vrhova V i bridova E, gdje je E podskup od $V \times V$. Za dani vrh V_i, neka $Ulaz(V_i)$ je skup svih vrhova koji pokazuju prema V_i, a skup $Izlaz(V_i)$ skup vrhova prema kojima V_i pokazuje. Formula tada glasi:

$$\mathrm{S}(V_i) = (1 - \mathrm{d}) + \mathrm{d} * \sum_{j \in Ulaz(V_i)} \frac{1}{|Izlaz(V_i)|} S(V_j) \qquad 27).$$

pri čemu d predstavlja faktor odbacivanja koji je moguće postaviti u rasponu od 0 do 1 te omogućuje skokove na nasumične vrhove grafa (Mihalcea i Tarau 2004).

Dodatno k tome se promatraju i pronalaze ključne riječi unutar tekstova na temelju odnosa podudaranja riječi, što znači da se dvije riječi povezane u slučaju podudarnosti njihovih leksičkih jedinica u rasponu od maksimalno *N* riječi, pri čemu je *N* moguće postaviti u rasponu od 2 do 10. Ovime se postiže dobra podloga prepoznavanja višeznačnosti te se stvaraju veze između sintaktičkih elemenata. Potom se pridodaje token svakoj riječi te postavlja PoS oznaka kako bi se mogla napraviti sintaktička filtracija. Izbjegava se korištenje ngmrama te se ključne riječi sadržane od više riječi rekonstruiraju. Sve leksičke jedinica koje zadovolje i prođu proces filtracije se dodaju u graf, a brid se dodaje između onih riječi koje zadovoljavaju podudarnost unutar zadanog broja riječi. Kada je stvoren graf i kada se napravilo više iteracija programa, vrhovi su sortirani u suprotnom redoslijedu na temelju ishoda te top *T* vrhova na ljestvici se zadržavaju za dodatnu obradu. Broj *T* se određuje na temelju veličine teksta, tj. trećinu od ukupnog broja vrhova u grafu.

4 Metode evaluacije klasifikacije

Postoje mnogi načini evaluacije, pri čemu se najčešće koriste pristupi ručne evaluacije skupine ljudi, što je podosta vremenski zahtjevno te potrebno podrobno planiranje. Kao odgovor na to pitanje, postoje mnogi računski pristupi i načini mjerenja. Kako bismo usporedili rezultate algoritama slučajne šume, naivnog Bayesa i SVM-a primjenjuje se nekoliko načina evaluacije. Iako neki od navedenih ne pružaju kompletan uvid u semantičku kvalitetu generiranih sažetaka, itekako nam omogućuju uvid u kvalitetu sažetaka.

4.1 Mjere odaziva i preciznosti

Kao što je vidljivo, većina algoritama ekstraktivnog sažimanja tekstova koriste rečenice sadržane unutar tekstova od kojih se stvaraju sažetci. Kada govorimo o ovakvim algoritmima, korištenje mjere odaziva i preciznosti su samo od nekih dostupnih načina evaluacije. Kako bismo dobili što bolji uvid u preciznost naših sažetaka, generirani sažetak S uspoređujemo sa sažetkom koji je sastavila osoba H te dijelimo s brojem ključnih riječi ili rečenica koje je prepoznao naš ekstraktivni algoritam:

$$P = \frac{|H \cap S|}{|S|} \qquad 28).$$

dok mjeru odaziva računamo tako da presjek skupova ključnih riječi ili rečenica koje je prepoznao čovjek H i naš algoritam sažimanja S dijelimo s brojem riječi ili rečenica koje je prepoznao čovjek:

$$P = \frac{|H \cap S|}{|H|} \qquad 29).$$

Ovakav način evaluacije nam ne govori o semantičkoj kvaliteti sažetka. Kao jedna od negativnih strana uzimanja rečenice za kao osnovne mjere preciznosti i mjere odaziva ne uzima u obzir mogućnost kako dvije kompletno različite rečenice mogu prenijeti istu informaciju. Čovjek bi obično odabrao samo onu rečenicu za koju osobno smatra da je najpovoljnija za sažetak.

Također, rečenice variraju u broju sadržanih riječi te po mogućnosti prenose različite količine informacija. Odabir dulje rečenice koja sadrži više informacije može biti poželjnije u odnosu na ostale. Kako bi se riješile navedene prepreke ovih načina evaluiranja klasifikacija, stvoreni su bolji

4.2 F1 mjera

Generalno, F mjere su statističke analize binarnih klasifikacija baziranih na temelju prethodno izračunatih mjera preciznosti i odaziva. F1 mjera je harmonična sredina mjera preciznosti i odaziva.

Generalni način računanja F_β koji koristi pozitivni stvarni faktor β , gdje je β odabrana tako da se mjera odaziva smatra β puta više značajnijom od mjere preciznosti:

$$F_\beta = (1 + \beta^{\,2}) \cdot \frac{preciznost \cdot odaziv}{(\beta^{\,2} \cdot preciznost) + odaziv} \qquad 30).$$

Time iz formule 30) i kada uzmemo u obzir da je F1 mjera harmonična sredina, tada naša jednadžba s F1 mjeru glasi:

$$F_1 = \frac{2}{preciznost^{-1} + odaziv^{-1}} = 2 \cdot \frac{preciznost \cdot odaziv}{preciznost + odaziv} \qquad 31).$$

4.3 ROUGE mjera

ROUGE ili Recall-Oriented Understudy for Gisting Evaluation uspoređuje automatski generirani sažetak s referenciranim sažetcima. Postoji više inačica ROUGE mjere:

- ROUGE-n prebrojava sve *n*-grame koje se preklapaju između automatski generiranog sažetka i onog sastavljenog od strane osobe. Prema (Steinberger i Jezek 2009.) kandidat ROUGE-n rezultata za sažetak se računa na slijedeći način:

$$ROUGE - n = \frac{\sum_{C \in RSS^{10}} \sum_{gram_n \in C} Broj_{podudaranja}(gram_n)}{\sum_{C \in RSS} \sum_{gram_n \in C} Broj(gram_n)} \qquad 32).$$

gdje $Broj_{podudaranja}(gram_n)$ predstavlja maksimalni broj *n*-grama koji se pojavljuju kao kandidat za sažetak te unutar referenciranog sažetka, a $Broj(gram_n)$ je broj *n*-grama (Beli 2018) unutar referenciranog sažetka.

- ROUGE-1 obuhvaća sva podudaranja riječi unutar referenciranog i generiranog sažetka
- ROUGE-L (na eng *Longest Common Subsequence (LCS)*) (Lin i Och 2004) koji uzima u obzir razinu sličnosti strukture rečenice te automatski identificira najduži niz podudarnih *n*-grama. Formula bi u tom slučaju glasila:

[10] RSS predstavlja skup referenciranih sažetaka (na eng. *reference summary skup*) (Steinberger i Jezek 2009.)

$$LCS(X,Y) = \frac{duljina(X) + duljina(Y) - edit_{di}(X,Y)}{2} \quad .$$

pri čemu su X i Y nizovi riječi ili lema, $LCS(X,Y)$ predstavlja najduži zajednički niz između X i Y, $duljina(X)$ je duljina stringa X, a $edit_{di}(X,Y)$ je uređena sličnosti između X i Y (Steinberger i Jezek 2009.).

Osim navedenih, postoje još i druge inačice navedene ROUGE mjere koje se ne obrađuju unutar ovog diplomskog rada.

5 Skup podataka, opis pristupa i statistički opis korpusa

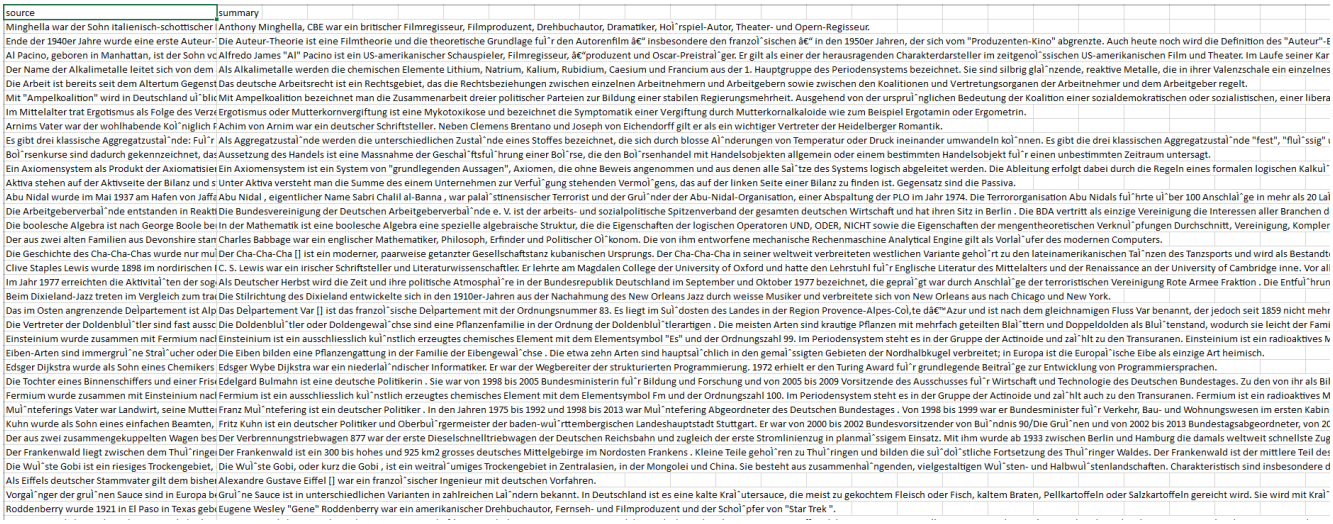
U ovom dijelu se opisuje praktični dio rada. Ukratko se opisuje sam skup podataka, od čega je sadržan te opis načina ekstrakcije rečenica koje formiraju. Koristi se nekoliko načina i algoritama kako bi se formirali grafovi rečenica i dobili rezultati te usporedili. Ti algoritmi su:

1. Klasifikacijski algoritmi:
    - Naivni Bayes,
    - SVM,
    - Klasifikator slučajne šume,
2. Algoritmi izračuna težine rečenica iz grafova:
    - LexRank,
    - SBKE,
    - TextRank,
3. Algoritmi mjera sličnosti:
    - Jaccardova sličnost,
    - Kosinusna sličnost,
    - Mihalceina sličnost,
    - Kullback-Leiblerova divergencija,
4. Algoritmi obrade i pronalaženje informacija:
    - TF-IDF vektor,
    - Prosječni word2vec vektor rečenica,
    - doc2vec vektor,
5. Evaluacija klasifikacije:
    - mjere preciznosti i odaziva,
    - F1 mjera.
6. Evaluacija sažimanja:
    - ROUGE mjera.

5.1 SwissText skup podataka

Ovo je skup podataka koji u sebi sadrži 100 000 tekstova napisanih na njemačkom jeziku te također referencirane sažetke navedenih tekstova. Ovaj skup podataka je sastavljen je na članaka njemačke Wikipedije te će sažetci koristit kasnije u postupku evaluacije i koliko je generirani sažetak točan i razumljiv. Nadalje kako bismo dobili podatke koji će nam koristiti pri dobivanju potrebno je izbaciti stop-riječi kako bi se smanjila dimenzionalnost problema.

Sam skup podataka je podijeljen u dva stupca: jedan sadrži članak, dok drugi sadrži referencirani sažetak koji će se koristiti za usporedbu i procjenu kvalitete generiranih sažetaka. Ovaj korpus je sastavljen u svrhu natjecanja u sažimanju tekstova 2019. godine kada su objavljeni mnogi radovi koji se koriste navedeni korpus (SwissText 2019).



source	summary
Minghella war der Sohn italienisch-schottischer I	Anthony Minghella, CBE war ein britischer Filmregisseur, Filmproduzent, Drehbuchautor, Dramatiker, Hol˜rspiel-Autor, Theater- und Opern-Regisseur.
Ende der 1940er Jahre wurde eine erste Auteur-˜	Die Auteur-Theorie ist eine Filmtheorie und die theoretische Grundlage ful˜r den Autorenfilm â€“ insbesondere den franzol˜sischen â€“ in den 1950er Jahren, der sich vom "Produzenten-Kino" abgrenzte. Auch heute noch wird die Definition des "Auteur"-E
Al Pacino, geboren in Manhattan, ist der Sohn vc	Alfredo James "Al" Pacino ist ein US-amerikanischer Schauspieler, Filmregisseur, â€"produzent und Oscar-Preistral˜ger. Er gilt als einer der herausragenden Charakterdarsteller im zeitgenol˜ssischen US-amerikanischen Film und Theater. Im Laufe seiner Kar
Der Name der Alkalimetalle leitet sich von dem	Als Alkalimetalle werden die chemischen Elemente Lithium, Natrium, Kalium, Rubidium, Caesium und Francium aus der 1. Hauptgruppe des Periodensystems bezeichnet. Sie sind silbrig glal˜nzende, reaktive Metalle, die in ihrer Valenzschale ein einzelnes
Die Arbeit ist bereits seit dem Altertum Gegenst	Das deutsche Arbeitsrecht ist ein Rechtsgebiet, das die Rechtsbeziehungen zwischen einzelnen Arbeitnehmern und Arbeitgebern sowie zwischen den Koalitionen und Vertretungsorganen der Arbeitnehmer und dem Arbeitgeber regelt.
Mit "Ampelkoalition" wird in Deutschland ul˜blic	Mit Ampelkoalition bezeichnet man die Zusammenarbeit dreier politischer Parteien zur Bildung einer stabilen Regierungsmehrheit. Ausgehend von der ursprul˜nglichen Bedeutung der Koalition einer sozialdemokratischen oder sozialistischen, einer libera
Im Mittelalter trat Ergotismus als Folge des Verz	Ergotismus oder Mutterkornvergiftung ist eine Mykotoxikose und bezeichnet die Symptomatik einer Vergiftung durch Mutterkornalkaloide wie zum Beispiel Ergotamin oder Ergometrin.
Arnims Vater war der wohlhabende Kol˜niglich F	Achim von Arnim war ein deutscher Schriftsteller. Neben Clemens Brentano und Joseph von Eichendorff gilt er als ein wichtiger Vertreter der Heidelberger Romantik.
Es gibt drei klassische Aggregatzustal˜nde: Ful˜r	Als Aggregatzustal˜nde werden die unterschiedlichen Zustal˜nde eines Stoffes bezeichnet, die sich durch blosse Al˜nderungen von Temperatur oder Druck ineinander umwandeln kol˜nnen. Es gibt die drei klassischen Aggregatzustal˜nde "fest", "ful˜ssig"
Bol˜rsenkurse sind dadurch gekennzeichnet, das	Aussetzung des Handels ist eine Massnahme der Geschal˜ftsful˜hrung einer Bol˜rse, die den Bol˜rsenhandel mit Handelsobjekten allgemein oder einem bestimmten Handelsobjekt ful˜r einen unbestimmten Zeitraum untersagt.
Ein Axiomensystem als Produkt der Axiomatisier	Ein Axiomensystem ist ein System von "grundlegenden Aussagen", Axiomen, die ohne Beweis angenommen und aus denen alle Sal˜tze des Systems logisch abgeleitet werden. Die Ableitung erfolgt dabei durch die Regeln eines formalen logischen Kalkul˜
Aktiva stehen auf der Aktivseite der Bilanz und s	Unter Aktiva versteht man die Summe des einem Unternehmen zur Verful˜gung stehenden Vermol˜gens, das auf der linken Seite einer Bilanz zu finden ist. Gegensatz sind die Passiva.
Abu Nidal wurde im Mai 1937 am Hafen von Jaffa	Abu Nidal , eigentlicher Name Sabri Chalil al-Banna , war palal˜stinensischer Terrorist und der Grul˜nder der Abu-Nidal-Organisation, einer Abspaltung der PLO im Jahr 1974. Die Terrororganisation Abu Nidals ful˜hrte ul˜ber 100 Anschlal˜ge in mehr als 20 Lal
Die Arbeitgeberverbal˜nde entstanden in Reakti	Die Bundesvereinigung der Deutschen Arbeitgeberverbal˜nde e. V. ist der arbeits- und sozialpolitische Spitzenverband der gesamten deutschen Wirtschaft und hat ihren Sitz in Berlin . Die BDA vertritt als einzige Vereinigung die Interessen aller Branchen d
Die boolesche Algebra ist nach George Boole be	In der Mathematik ist eine boolesche Algebra eine spezielle algebraische Struktur, die die Eigenschaften der logischen Operatoren UND, ODER, NICHT sowie die Eigenschaften der mengentheoretischen Verknul˜pfungen Durchschnitt, Vereinigung, Kompler
Der aus zwei alten Familien aus Devonshire stan	Charles Babbage war ein englischer Mathematiker, Philosoph, Erfinder und Politischer Ol˜konom. Die von ihm entworfene mechanische Rechenmaschine Analytical Engine gilt als Vorlal˜ufer des modernen Computers.
Die Geschichte des Cha-Cha-Chas wurde nur mul	Der Cha-Cha-Cha [] ist ein moderner, paarweise getanzter Gesellschaftstanz kubanischen Ursprungs. Der Cha-Cha-Cha in seiner weltweit verbreiteten westlichen Variante gehol˜rt zu den lateinamerikanischen Tal˜nzen des Tanzsports und wird als Bestandt
Clive Staples Lewis wurde 1898 im nordirischen	C. S. Lewis war ein irischer Schriftsteller und Literaturwissenschaftler. Er lehrte am Magdalen College der University of Oxford und hatte den Lehrstuhl ful˜r Englische Literatur des Mittelalters und der Renaissance an der University of Cambridge inne. Vor al
Im Jahr 1977 erreichten die Aktivital˜ten der sog	Als Deutscher Herbst wird die Zeit und ihre politische Atmosphal˜re in der Bundesrepublik Deutschland im September und Oktober 1977 bezeichnet, die gepral˜gt war durch Anschlal˜ge der terroristischen Vereinigung Rote Armee Fraktion . Die Entful˜hrun
Beim Dixieland-Jazz treten im Vergleich zum tra	Die Stilrichtung des Dixieland entwickelte sich in den 1910er-Jahren aus der Nachahmung des New Orleans Jazz durch weisse Musiker und verbreitete sich von New Orleans aus nach Chicago und New York.
Das im Osten angrenzende Delpartement ist Alp	Das Delpartement Var [] ist das franzol˜sische Delpartement mit der Ordnungsnummer 83. Es liegt im Sul˜dosten des Landes in der Region Provence-Alpes-Col,te dâ€™Azur und ist nach dem gleichnamigen Fluss Var benannt, der jedoch seit 1859 nicht mehr
Die Vertreter der Doldenblul˜tler sind fast aussc	Die Doldenblul˜tler oder Doldengewal˜chse sind eine Pflanzenfamilie in der Ordnung der Doldenblul˜tlerartigen . Die meisten Arten sind krautige Pflanzen mit mehrfach geteilten Blal˜ttern und Doppeldolden als Blul˜tenstand, wodurch sie leicht der Fami
Einsteinium wurde zusammen mit Fermium nacl	Einsteinium ist ein ausschliesslich kul˜nstlich erzeugtes chemisches Element mit dem Elementsymbol "Es" und der Ordnungszahl 99. Im Periodensystem steht es in der Gruppe der Actinoide und zal˜hlt zu den Transuranen. Einsteinium ist ein radioaktives M
Eiben-Arten sind immergrul˜ne Stral˜ucher oder	Die Eiben bilden eine Pflanzengattung in der Familie der Eibengewal˜chse . Die etwa zehn Arten sind hauptsal˜chlich in den gemal˜ssigten Gebieten der Nordhalbkugel verbreitet; in Europa ist die Europal˜ische Eibe als einzige Art heimisch.
Edsger Dijkstra wurde als Sohn eines Chemikers	Edsger Wybe Dijkstra war ein niederlal˜ndischer Informatiker. Er war der Wegbereiter der strukturierten Programmierung. 1972 erhielt er den Turing Award ful˜r grundlegende Beitral˜ge zur Entwicklung von Programmiersprachen.
Die Tochter eines Binnenschiffers und einer Fris	Edelgard Bulmahn ist eine deutsche Politikerin . Sie war von 1998 bis 2005 Bundesministerin ful˜r Bildung und Forschung und von 2005 bis 2009 Vorsitzende des Ausschusses ful˜r Wirtschaft und Technologie des Deutschen Bundestages. Zu den von ihr als Bil
Fermium wurde zusammen mit Einsteinium nacl	Fermium ist ein ausschliesslich kul˜nstlich erzeugtes chemisches Element mit dem Elementsymbol Fm und der Ordnungszahl 100. Im Periodensystem steht es in der Gruppe der Actinoide und zal˜hlt auch zu den Transuranen. Fermium ist ein radioaktives M
Mul˜nteferings Vater war Landwirt, seine Muttei	Franz Mul˜ntefering ist ein deutscher Politiker . In den Jahren 1975 bis 1992 und 1998 bis 2013 war Mul˜ntefering Abgeordneter des Deutschen Bundestages . Von 1998 bis 1999 war er Bundesminister ful˜r Verkehr, Bau- und Wohnungswesen im ersten Kabin
Kuhn wurde als Sohn eines einfachen Beamten,	Fritz Kuhn ist ein deutscher Politiker und Oberbul˜rgermeister der baden-wul˜rttembergischen Landeshauptstadt Stuttgart. Er war von 2000 bis 2002 Bundesvorsitzender von Bul˜ndnis 90/Die Grul˜nen und von 2002 bis 2013 Bundestagsabgeordneter, von 20
Der aus zwei zusammengekuppelten Wagen bes	Der Verbrennungstriebwagen 877 war der erste Dieselschnelltriebwagen der Deutschen Reichsbahn und zugleich der erste Stromlinienzug in planmal˜ssigem Einsatz. Mit ihm wurde ab 1933 zwischen Berlin und Hamburg die damals weltweit schnellste Zug
Der Frankenwald liegt zwischen dem Thul˜ringer	Der Frankenwald ist ein 300 bis hohes und 925 km2 grosses deutsches Mittelgebirge im Nordosten Frankens . Kleine Teile gehol˜ren zu Thul˜ringen und bilden die sul˜dol˜stliche Fortsetzung des Thul˜ringer Waldes. Der Frankenwald ist der mittlere Teil des
Die Wul˜ste Gobi ist ein riesiges Trockengebiet,	Die Wul˜ste Gobi, oder kurz die Gobi , ist ein weitral˜umiges Trockengebiet in Zentralasien, in der Mongolei und China. Sie besteht aus zusammenhal˜ngenden, vielgestaltigen Wul˜sten- und Halbwul˜stenlandschaften. Charakteristisch sind insbesondere d
Als Eiffels deutscher Stammvater gilt dem bisher	Alexandre Gustave Eiffel [] war ein franzol˜sischer Ingenieur mit deutschen Vorfahren.
Vorgal˜nger der grul˜nen Sauce sind in Europa bi	Grul˜ne Sauce ist in unterschiedlichen Varianten in zahlreichen Lal˜ndern bekannt. In Deutschland ist es eine kalte Kral˜utersauce, die meist zu gekochtem Fleisch oder Fisch, kaltem Braten, Pellkartoffeln oder Salzkartoffeln gereicht wird. Sie wird mit Kral˜
Roddenberry wurde 1921 in El Paso in Texas gebo	Eugene Wesley "Gene" Roddenberry war ein amerikanischer Drehbuchautor, Fernseh- und Filmproduzent und der Schol˜pfer von "Star Trek ".

Slika 0-1: Primjer izgleda korpusa

Dodatne primjere ovog skupa podataka moguće je vidjeti pod naslovom 8.3 Primjer SwissText korpusa gdje su dodatno prikazani svi ispisi i rezultati koji se argumentiraju pod sljedećim podnaslovima.

5.2 Statistički opis korpusa

Kako bismo dobili uvid u ovaj skup podataka, računa se prosječna duljina svakog teksta, koliko riječi sadrži najkraći, a koliko najdulji tekst, ukupan broj riječi u svakom tekstu, napraviti isto i za sažetke te odrediti omjere tekstova i njihovih definiranih sažetaka.

5.2.1 Opis i rezultati prosječne duljine rečenica, riječi, prosječan broj rečenica unutar teksta

Kako bismo dobili uvid u naš skup podataka računa se broj riječi i rečenica najdužeg teksta, dužina riječi i rečenica teksta. Dužina riječi i rečenica je izražena u dužini znakova svake riječi i rečenice. Kako bismo izračunali broj rečenica potrebno je pročitati svaki redak okvira podataka unutar stupca source te cijeli tekst rastaviti na rečenice. Definira se funkcija sentence_count() koja pomoću tokenizacije teksta rastavlja tekst na rečenice te kao rezultat vraća broj rečenica koji se potom pohranjuje o novi stupac. Funkciju pozivamo na stupcima summary i source našeg okvira podataka kako bismo saznali koliki je broj rečenica u svim

tekstovima. Rezultati se pohranjuju unutar novih stupaca no_sent_source i no_sent_summary.

```
# -- counting the number of sentences of summaries and original texts -- #
def sentence_count(row):
    tokens = sent_tokenize(row, language='german')
    return len(tokens)

df['no_sent_source'] = df['source'].apply(sentence_count)
df['no_sent_summary'] = df['summary'].apply(sentence_count)
df.head()
```

Slika 0-2: Funkcija prepbrojavanja rečenica

Postupak za prebrojavanje riječi je isti, samo što ovoga puta radimo prepoznavanje i tokenizaciju riječi funkcijom word_token unutar definirane funkcije word_count(). Rezultate funkcije pohranjujemo u nove stupce no_word_source i no_word_summary. Kako bi se što točnije prebrojale riječi potrebno je prethodno ukloniti sve interpunkcijske znakove pošto funkcija word_token također prepoznaje interpunkcijske znakove kao riječi što daje netočne rezultate.

```
# -- counting the number of sentences of summaries and original texts -- #
def word_count(row):
    row = row.translate(str.maketrans('', '', string.punctuation))
    tokens = word_tokenize(row, language='german')
    return len(tokens)

df['no_word_source'] = df['source'].apply(word_count)
df['no_word_summary'] = df['summary'].apply(word_count)
df.head()
```

Slika 0-3: Fukcija prepobrajavanje riječi

Sljedeće što računamo jest prosječna dužina rečenice i riječi. U tu svrhu dovoljno je samo malo preformulirati prethodne formule. Tijekom računanja prosječne duljine rečenica ponovno se teskt razdvaja na rečenice te je rezultat ovoga puta predstavljen kao količnik zbroja dužine rečenica i broja rečenica. Potom se funkcija poziva za stupce source i summary našeg okvira podataka. Također se odmah računa prosječna dužina rečenica unutar stupaca source i summary koji su ponovno predstavljeni kao kvocijent zbroja svih vrijednosti stupca i broja redaka, tj. broja tekstova stupca.

```
# -- average sentance length in characters -- #
def avg_s_len(row):
    tokens = sent_tokenize(row, language='german')
    return float(sum(len(s) for s in tokens)/len(tokens))

df['avg_source_s_len'] = df['source'].apply(avg_s_len)
df['avg_summary_s_len'] = df['summary'].apply(avg_s_len)

avg_s_len_source = float(sum(idx for idx in df['avg_source_s_len'])/len(df))
avg_s_len_summary = float(sum(idx for idx in df['avg_summary_s_len'])/len(df


print(avg_s_len_source, avg_s_len_summary)
df.head()
```

```
126.19653690882299 114.67222942718325
```

Slika 0-4: Funkcija prosječne duljine rečenica teksta i izračun prosječne duljine rečenica unutar stupca

Funkcija za računanje prosječne dužine riječi u tekstu slična je prethodnojfunkciji. Kao i u funkciji za prebrojavanje riječi unutar teksta uklanjaju se interpunkcijski znakovi kako ne bi bili uračunati u riječi, rastavljamo teksta na listu riječi te kao rezultat funkcije se dobiva kvocijent sume dužine svake riječi i ukupnog broja riječi unutar teksta. Funkcija se poziva zasebno za stupce source i summary te se računa ukupna prosječna dužina riječi svakog od stupaca u obliku sume vrijednosti stupca awg_w_len_<source ili summary> te broja redaka stupca.

```
# --- avg word length -- #
def avg_w_len(row):
    row = row.translate(str.maketrans('', '', string.punctuation))
    tokens = word_tokenize(row, language='german')
    return float(sum(len(w) for w in tokens)/len(tokens))

df['avg_source_w_len'] = df['source'].apply(avg_w_len)
df['avg_summary_w_len'] = df['summary'].apply(avg_w_len)
df.head()
```

Slika 0-5: Funkcija računanja prosječne duljine riječi teksta

Svi rezultati na temelju ovih izračuna predstavljaju se unutar Tablica 2:Statistički opis skupa podataka; prosječna duljina riječi, prosječna duljina rečenica itd.

	Tekstovi	*Sažetci*
Ukupni prosjek dužine riječi	6.037844064022607	6.38804764156304
Ukupni prosjek dužine rečenica	126.19653690882299	114.67222942718325

Ukupni prosječni broj riječi svih tekstova	559.93383	32.85926
Ukupni prosječni broj rečenica svih tekstova	32.0396	2.22468
Broj rečenica najdužeg teksta	84	6
Broj riječi najdužeg teksta	2000	150
Prosječna dužina riječi najdužeg teksta	4.139175	132.0
Prosječna dužina rečenice najkraćeg teksta	37.039474	29.0
Broj rečenica najkraćeg teksta	2	1
Broj riječi najkraćeg teksta	313	6
Prosječna dužina riječi najkraćeg teksta	5.600639	4.166667
Prosječna dužina rečenica najkraćeg teksta	1016.5	15.5

Tablica 2:Statistički opis skupa podataka; prosječna duljina riječi, prosječna duljina rečenica itd.

Pretraga sažetka s najmanjim brojem riječi i rečenica rezultira sa višestrukim rezultatima koji imaju 1 rečenicu sastavljenu od 5 riječi. Svi tekstovi tih primjera su međusobno različiti te im je zajednička samo navedena duljina sažetaka. Pošto su rečenice predstavljene u formatu string, njihova dužina je predstavljena u obliku broja znakova unutar stringa, tj. rečenice. Kako bi se zaista pronašao tekst koji ima najkraći sažetak, promatraju se također značajke prosječne duljine rečenica, broj riječi sažetka te prosječna duljina riječi sažetka. Dva sažetka međusobno različitih tekstova zadovoljavaju uvijete ove pretrage. Budući kako je riječ o dva rezultata s istim najmanjim brojem riječi, koji također imaju i jednaku vrijednost prosječne dužine rečenica sažetka, jednak broj riječi sažetka te jednaku vrijednost prosječne dužine riječi sažetka, promatra se kao jedinstveni rezultat koji je uvršten u Tablica 2:Statistički opis skupa podataka; prosječna duljina riječi, prosječna duljina rečenica itd.

5.2.2 Ngrami i njihov prikaz

Po izračunu navedenoga, generalan uvid u ngrame dobivamo sljedećim načinom; Učitava se nltk.collocations te se definira finder za pronalaženje svih ngrama. Definira se mjerenje ngrama korištenjem odgovarajuće mjere te se postavlja finder koji prolazi kroz zadani stupac. Potom, po izboru, filtriramo bigrame po frekvenciji pojavljivanja te se ispisuje top n bigrama. Primjer kraćeg koda za to izgledao bi ovako:

```
import nltk
from nltk.collocations import *
bigram_measures = nltk.collocations.BigramAssocMeasures()
finder = BigramCollocationFinder.from_documents(df['word_token'])

top 10 ngrams with  highest PMI
#finder.apply_freq_filter(1)

finder.nbest(bigram_measures.pmi, 10)
```

Slika 0-6:Kraći primjer ispisa top 10 bigrama frekvencije 1

Rezultati pokretanja navedenog algoritma na stupcu word_token koji zapravo sadrži tekstove iz stupca source sa odvojenim i tokeniziranim riječima :

```
[("'Abd", 'or-Rahmān'),
 ("'Active", "Brake'-Differenzial"),
 ("'Alleen", 'Gezamenlijk'),
 ("'CHESEBROUGH", 'VASELINE'),
 ("'De", 'Bloempot'),
 ("'Designer", 'Collaborations'),
 ("'Etezad", 'od-Doleh'),
 ("'Eyn", 'ol-Molk'),
 ("'Fair", 'Ellender'),
 ("'Fakhr", 'ol-Zakerin')]
```

U slučaju ispisa trigrama koraci su isti te su rezultati sljedeći:

Prema rezultatima znači da bi generalna funkcija, koja bi radila izračun kolokacija, ovisno o izboru korisnika je definirana sljedeći način:

```
def finder(df, col, repetition, top_n):
    print("You can pick following:\n1: bigrams\n 2: trigrams")
    option = int(input(" \nyour option : "))
    def bigram():
        bigram_measures = nltk.collocations.BigramAssocMeasures()
        find = BigramCollocationFinder.from_documents(df[col])
        find.apply_freq_filter(repetition)
        finder.nbest(bigram_measures.pmi, top_n)
    def trigram():
        trigram_measures = nltk.collocations.TrigramAssocMeasures()
        find = TrigramCollocationFinder.from_documents(df[col])
        find.apply_freq_filter(repetition)
        finder.nbest(trigram_measures.pmi, top_n)
    def default():
        print("Incorrect option")
    switch ={
        1: bigram,
        2: trigram,
    }
    switch.get(option, default)()
```

Slika 0-7:Funkcija za računanje kolokacija

gdje sa df definiramo koji okvir podataka želimo, col predstavlja stupac unutar definiranog okvira, repetition je frekvencija pojavljivanja ili PMI, a uz pomoć `top_n` definiramo ispis prvih n rezultata. Ostale rezultate bigrama i trigrama je moguće vidjeti u 8.7 Ispis rezultata dobivenih ngrama korpusa

Za drugi način kojim bismo grafički mogli prikazati ngrame potrebno je procesirati podatke tako što se definira funkcija `cleanSource()` koja kao parametar uzima okvir podataka, čisti podatke od posebnih znakova i stop-riječi te pretvaramo sve znakove u mala slova.

```
def cleanSource(documents):
    cleanedSource = []
    for document in documents:
        s = re.sub(r'[^a-zA-Z0-9\s]', '', document)
        s = re.sub('\s+',' ', s)
        s = str(s).lower()
        tokens = [token for token in s.split(" ") if token != ""]
        tokens = [word for word in tokens if word not in stopwords.words('german')]
        review = ' '.join(tokens)
        cleanedSource.append(review)
    return(cleanedSource)
```

Slika 0-8: Funkcija za procesiranje podataka prethodno računanju ngrama

Potom je potrebno dokumentirati ngrame, a za to nam služi `documentNgrams()` funkcija koja kao parametre uzima tekstove te varijabla `size` definira je li riječ o monogramu (jednoj riječi), bigramu ili trigramu. Zatim se za svaku stavku unutar toka podataka razdvajaju nizovi te se petlja vrti dokle god je duljina niza manja od definirane veličine, a kada više ne zadovoljava uvjet se pridodaje listi ngrama. Nakon toga se podatci pohranjuju u novi okvir podataka.

```
def documentNgrams(documents, size):
    ngrams_all = []
    for document in documents:
        tokens = document.split()
        if len(tokens) <= size:
            continue
        else:
            output = list(ngrams(tokens, size))
        for ngram in output:
            ngrams_all.append(" ".join(ngram))
    cnt_ngram = Counter()
    for word in ngrams_all:
        cnt_ngram[word] += 1
    df = pd.DataFrame.from_dict(cnt_ngram, orient='index').reset_index()
    df = df.rename(columns={'index':'words', 0:'count'})
    df = df.sort_values(by='count', ascending=False)
    df = df.head(15)
    df = df.sort_values(by='count')
    return(df)
```

Slika 0-9: Funkcija spremanja ngrama u tok podataka

Posljednji korak koji nam ostaje jest samo prikazivanje ngrama koje smo prethodno definirali. Definira se funkcija plotNgrams() koja u određene varijable pohranjuje rezultate koje smo dobili pozivanjem prethodne funkcije. Potom se određuje veličina slike grafa, definiraju se parametri za prikaz grafa u kojem imamo po graf za svaki ngram.

```python
def plotNgrams(documents):
    unigrams = documentNgrams(documents, 1)
    bigrams = documentNgrams(documents, 2)
    trigrams = documentNgrams(documents, 3)

    # Set plot figure size
    fig = plt.figure(figsize = (20, 7))
    plt.subplots_adjust(wspace=.5)

    ax = fig.add_subplot(131)
    ax.barh(np.arange(len(unigrams['words'])), unigrams['count'], align='center', alpha=.5)
    ax.set_title('Unigrams')
    plt.yticks(np.arange(len(unigrams['words'])), unigrams['words'])
    plt.xlabel('Count')

    ax2 = fig.add_subplot(132)
    ax2.barh(np.arange(len(bigrams['words'])), bigrams['count'], align='center', alpha=.5)
    ax2.set_title('Bigrams')
    plt.yticks(np.arange(len(bigrams['words'])), bigrams['words'])
    plt.xlabel('Count')

    ax3 = fig.add_subplot(133)
    ax3.barh(np.arange(len(trigrams['words'])), trigrams['count'], align='center', alpha=.5)
    ax3.set_title('Trigrams')
    plt.yticks(np.arange(len(trigrams['words'])), trigrams['words'])
    plt.xlabel('Count')

    plt.show()
```

Slika 0-10: Funkcija prikazivanja grafa

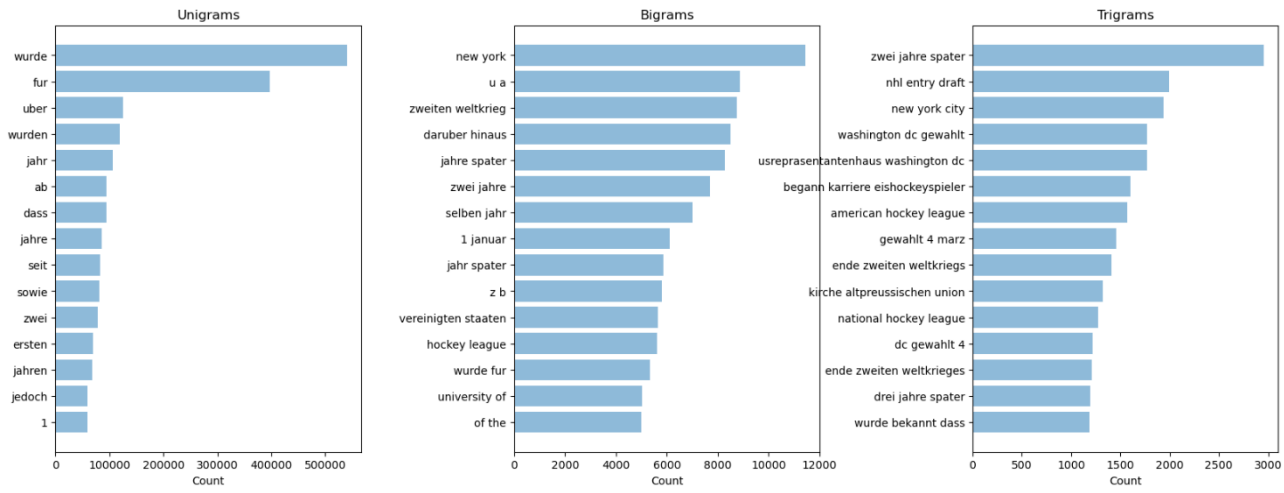
Jedino što preostaje jest definirati i pozvati funkciju textTrends() koja će izvršiti operacije nad našim originalnim skupom podataka.

```python
def textTrends(documents):
    cleanedSource = cleanSource(documents)
    plotNgrams(cleanedSource)

textTrends(df['source'])
```

Slika 0-11: Funkcija koja poziva sve prethodne korake te ih primjenjuje na skup podataka

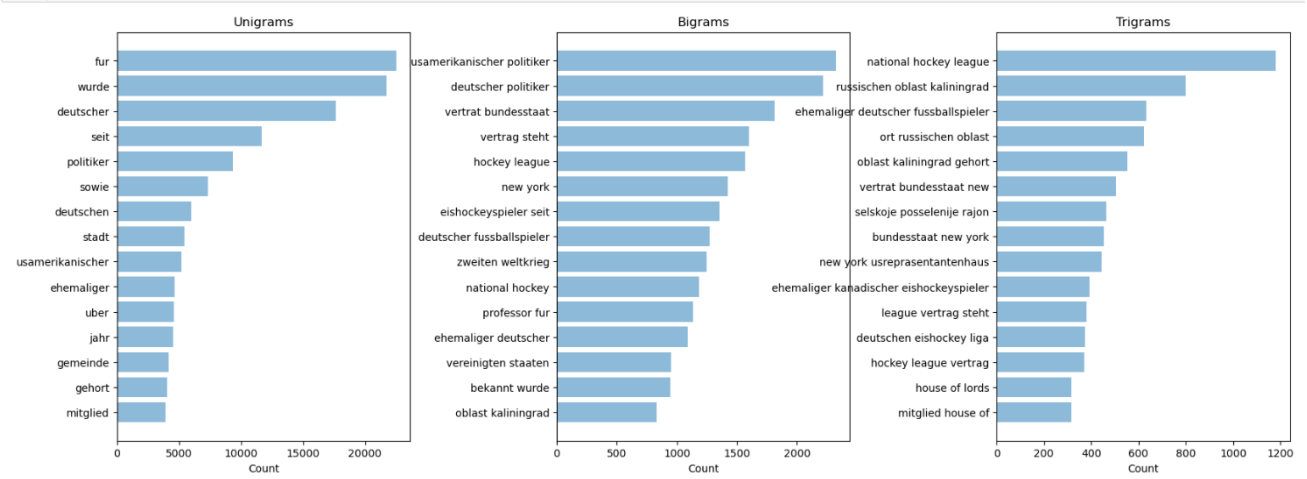
Rezultati ovog algoritma su sljedeći:



Slika 0-12: Prikaz najčešćih unigrama, bigrama i treigrama u stupcu source skupa podataka

Ovdje vidimo kako se najčešće pojavljuje glagol werden kod monograma, New York kod bigrama te zwei Jahre später (na hrv. dvije godine kasnije).

Isti algoritam ćemo provjeriti i za prethodno definirane sažetke koji su dostupni unutar stupca summary našeg skupa podataka:



Slika 0-13: Ispis unigrama, bigrama i trigrama stupca summary skupa podataka

Ovdje je moguće vidjeti, pošto koristimo listu stop-riječi koja je definirana unutar nltk.corpusa, kako postoji nekoliko riječi poput für koje nisu prepoznate kao stop-riječi, te stoga imaju najveći broj pojavljivanja u monogramima. U bigramima vidimo kako se kombinacija amerikanischer Politiker pojavljuje načešće, dok je New York pao skroz na šestu poziciju. U trigramima se pak National Hockey League pojavljuje najčešće, a isti se u stupcu source pojavljuje tek na jednaestoj poziciji.

5.3 Priprema podataka prethodno procesiranju

Prethodno samoj obradi i stvaranja algoritma potrebno pročistiti podatke od stop-riječi. Pošto govorimo o njemačkom jeziku, bitno je ukloniti sve određene i neodređene članove, određene i neodređene zamjenice, veznike, čestice, sve interpunkcijske znakove. Također je potrebno ukloniti posebne znakove te ih zamijeniti standardnima. Detaljnije pojašnjenje je dostupno pogledati (Beli 2018). Svaki od koraka opisuju se unutar zadanih naslova

5.3.1 Učitavanje podataka, čišćenje skupa podataka od stop-riječi i tokenizacija

Kako bismo učitali podatke u program potreban nam je Pandas. Naredbom `import pandas as pd` učitava se biblioteka koja nam je potrebna za učitavanje skupa podataka. Unutar `functions.py` definirana je funkcija `read()` u kojoj se definiraju putanje train.csv i test.csv.

Svaka od navedenih datoteka se pohranjuje unutar vlastitih varijabli. Funkciju pozivamo unutar `main.py`.

Sada imamo oba skupa podataka pohranjena unutar odgovarajućih varijabli, može se nastaviti sa tokenizacijom corpusa. Unutar datoteke `functions.py` definirana je funkcija `identify_tokens()` koja prolazi red po red unutar našeg skupa podataka. Kako bismo odradili ovaj dio potrebno je učitati nltk biblioteku koja ima definiranu funkciju `word_tokenize` koju ćemo primijeniti na stupac `source` našeg skupa podataka. Potom se for petljom prolazi kroz listu tokena. Potom se stvara novi stupac unutar dana skupa pod nazivom `words` te se ova funkcija primjenjuje na cjelokupni skup podataka. Proces prolaska kroz cijeli skup podataka zna biti dugotrajan radi same veličine korpusa.

5.3.2 Korjenovanje lematizacija i uklanjanje stop-riječi

Ovdje je riječ o više vrsta korjenovatelja: Porter korjenovatelj i Snowball korjenovatelj. Konkretan popis korjenovatelja, rezultate i izvore moguće je detaljnije pročitati u (Beli 2018).

Porter korjenovatelj definiran je funkcijom `stemmer`. Stupac `words` koji smo dobili u prethodnom koraku, stvara listu `stemmed_list` te prolazi kroz sve riječi, prepoznajući njihov korijen. Potom se na isti način poziva funkcija `stemmer`, čiji se rezultati pohranjuju u novi stupac `stemmed_words`. Kako bismo odredili korištenje Porter korjenovatelja, postavljena je varijabla stemming na `PorterStemmer()`.

Kada govorimo o Snowball korjenvatelju proces je sličan. Funkcija `stemmer` je koncipirana na switch modelu u kojem korisnik sam bira korjenovatelja. Snowball korjenovatelj je dostupan unutar nltk paketa, te kako bi se koristio za njemački jezik, potrebno je učitati verziju GermanStemmer. Ostatak procesa je isti kao i za Porterov korjenovatelj.

Slika 0-14: Primjena algoritama korjenovanja.

Lematizacija se izvršava na dva moguća načina: pomoću Treetaggera, Hanover lematizatora te WordNet lemastizatora. Funkcija pickLemma() je također definirana kao funkcija višestrukog odabira te se unutar nje definiraju ove tri mogućnosti: Hanover, Treetag ili WordNet.

WordNet lematizator se koristi pomoću biblioteke nltk kao i Porterov korjenovatelj. Postavlja se varijabla lemmatizer kao `WordNetLemmatizer()`, definira se lista u koju pohranjujemo riječi koje su prethodno definirane pod stupcem `word` te se prolazi riječ po riječ, pri tome primijejujući definirani lematizator.

Hanover lematizator učitava model `morphmodel_ger.pgz`, koji je potreban za za obradu njemačkih teksotva, te primjenjujemo lematizator na prethodno tagirane rečenice.

Treetager lematizator prvenstevno postavlja jezik koji će se obrađivati parametrom `TAGLANG='de'` te se potom se primjenjuje lematizator (Beli 2018), a rečenice. U slučaju odabira bilo čega drugoga, ispisuje se `default()`. Kako bi se pozvale određene funkcije na temelju odabira korisnika, definira se rječnik u kojem, pri unosu bilo kojeg broja se pokreće određena funkcija vezana uz lematizatore.

Slika 0-15: Primjena lematizatora

Kako bi se moglo odrediti koje riječi i rečenice su ključni za temu potrebno je ukloniti stop-riječi. Njihovo pojavljivanje unutar tekstova predstavlja veliki šum kada se žele prepoznati riječi koje su ključne za opdređenu temu. Kako bi se uklonile stop-riječi njemačkog jezika, definira se varijabla german_stopwords u koju se pohranjuje popis svih riječi koje spadaju u tu vrstu. Popis tih riječi za njemački jezik moguće je pronaći unutar nltk.corpus biblioteke pozivanjem `stopwords.words('german')`. Nakon svog prethodnog procesiranja teksta i korpusa, moguće je odabrati ili listu korjenovanih ili lematiziranih tekstova. Potom sa for petljom se spremaju sve riječi u listu koje ne pripadaju i nije ih moguće naći unutar liste stop-riječi.

```
def cl_StopWords(df):
    german_stopwords = stopwords.words('german')
    my_list = df['stemmed_words']
    words = ' '.join([w for w in my_list if not w in german_stopwords])
    return words
```

5.4 Klasifikacija teksta

U ovom diplomskom radu primijenjeno je više vrsta klasifikatora:

- Random Forest,
- Naivni Bayes,
- Support Vector Machine.

Svaki od navedenih klasifikatora se primjenjuje za sve vrste vektorizacije rečenica. Rezultati klasifikacija su predstavljeni u 6.3 Usporedba rezultata algoritama klasifikacije

5.4.1 Računanje značajki skupa podataka

Prethodno računanju značajki potrebno je svaki od tekstova rastaviti na rečenice te rečenice predstaviti kao zasebne retke unutar okvira podataka. Stoga se koristi već predstavljeni način obilježavanja rečenica okvira podataka predstavljen u dijelu rada algoritama sažimanja, pomoću biblioteke nltk.tokenize odakle se preuzima metoda sent_tokenize() u obliku lambda funkcije. Ukratko kod je sljedeći:

```
import numpy as np
import pandas as pd

# Getting s as pandas series which has split on full stop and new sentence a new line
s = df["source"].apply(lambda x : sent_tokenize(x)).apply(pd.Series,1).stack()
s.index = s.index.droplevel(-1) # to line up with df's index
s.name = 'source' # needs a name to join

x=s.to_frame(name='source1')

# Joining should ideally get me proper output. But I am getting original dataframe instead of split one.
df = df.join(x)
```

Slika 0-16: Rastavljenje tekstova skupa podataka na rečenice te predstavljanje rečenica kao zasebne retke okvira podataka.

Iz modula pandas se koristi funkcija pd.Series kako bi se svaka rečenica predstavila kao novi redak okvira podataka. Funkcija droplevel() omogućuje poravnavanje dobivenih rezultata s postojećim skupom podataka, jer i dalje želimo znati koja rečenica je pripadala kojem tekstu. Definira se ima novoga stupca u source1 te se dosadašnji rezultat pokretanja spaja s postojećim okvirom podataka.

Potrebno je napraviti još jedan korak kako bismo dobili popis riječi koje predstavljaju ključne riječi tekstova originalnog skupa podataka. Kako bismo izračunali ključne riječi potrebno je napraviti rječnik IDF vrijednosti tekstova. Prethodno s uklanjaju sve stop-riječi iz teksta, postavljamo ograničenje na maksimalno 10 000 značajki, uklanjamo 85% izraza koji se pojavljuju najčešće u tekstovima te računamo njihove vrijednosti vektora.

```
from nltk.corpus import stopwords
german_stop_words = stopwords.words('german')

cv = CountVectorizer(max_df=0.85,stop_words=german_stop_words, max_features=10000)
word_count_vector = cv.fit_transform(df['source'])

```

Slika 0-17: Uklanjanje stop-riječi i stvaranje rječnika IDF vrijednosti

Potom za matricu koju dobijemo kao rezultat prethodne isječka koda računamo IDF vrijedosti vektora.

```
tfidf_transformer=TfidfTransformer(smooth_idf=True,use_idf=True)
tfidf_transformer.fit(word_count_vector)
```

Slika 0-18:Računanje IDF vrijednosti vektora

Sada je potrebno izračunati TF-IDF vrijednosti i pronaći ključne riječi. Definiraju se funkcije za sortiranje TF IDf matrica te izvlačenje top n ključnih riječi. Funkcija sort_coo() uzima matrice te ih sortira po silaznoj vrijednosti. Funkcija extract_topn_from_vector() kao parametre uzima značajke tj. naš popis riječi koje su kandidati za ključne riječi, sortiranu matricu te se definira top n rezultata. Potom se poziva funkcija za sortiranje i definiramo prazna polja score_vals i feature_vals. Iteriramo po listi kandidata, u score_vals pohranjujemo njihove TF-IDF vrijednosti, a u feature_vals pohranjujemo riječi koje su odabrane. Na kraju se stvara uređena n-torka koja se prikazuje kao rezultat ove funkcije.

```
def sort_coo(coo_matrix):
    tuples = zip(coo_matrix.col, coo_matrix.data)
    return sorted(tuples, key=lambda x: (x[1], x[0]), reverse=True)

def extract_topn_from_vector(feature_names, sorted_items, topn=10):
    """get the feature names and tf-idf score of top n items"""

    #use only topn items from vector
    sorted_items = sorted_items[:topn]

    score_vals = []
    feature_vals = []

    for idx, score in sorted_items:
        fname = feature_names[idx]

        #keep track of feature name and its corresponding score
        score_vals.append(round(score, 3))
        feature_vals.append(feature_names[idx])

    #create a tuples of feature,score
    #results = zip(feature_vals,score_vals)
    results= {}
    for idx in range(len(feature_vals)):
        results[feature_vals[idx]]=score_vals[idx]

    return results
```

Slika 0-19: funkcije sortiranja i definiranja top n vrijednosti IDf vrijednosti

Sljedeći korak je utvrđivanje TF-IDF vrijednosti za svaki dokument, tj. tekst. Definira se funkcije keyw() u kojoj se postavlja polje kljuc, računa TF-IDF pozivanjem `tfidf_transformer.transform(...)` kako bi se generirali vektore. Potom se sortiraju riječi od one s najvećom vrijednosti vektora prema najmanjoj uz pomoć prethodno definirane funkcije sort_coo(), poziva se funkcija extract_topn_from_vector() te iteriramo po listi sortiranih vrijednosti i riječi te se pohranjuju vrijednosti u polje kljuc. Funkcija se potom poziva na stupcu source te vrijednosti spremamo unutar novog stupca keywords.

```
#doc = df['source'][0]
def keyw(df):
    kljuc = []
    tf_idf_vector=tfidf_transformer.transform(cv.transform([df]))
    sorted_items=sort_coo(tf_idf_vector.tocoo())
    keywords = extract_topn_from_vector(feature_names,sorted_items,10)
    for k in keywords:
        kljuc.append(k)
    return kljuc

df['keywords'] = df['source'].apply(keyw)
```

Slika 0-20: Finalna fukncija izvlačenja, sortiranja i ispisa top n ključnih riječi

Sada kada smo dobili rezultate za svako od tekstova, možemo prijeći na definiranje potrebnih značajki s popisa.

Značajke koje računamo su sljedeće:

- Broj riječi u rečenicama koji započinju velikim slovom,
- Broj riječi u rečenicama koji su ključne riječi,
- Broj rečenica koje se nalaze unutar source i summary stupca okvira podataka.

Kako bismo dobili broj riječi u rečenici koje započinju velikim slovom definira se funkcija capital_lWods() koja kao argument prima okvir podataka. Tekstovi se rastavljaju na rečenice te se iterira po svakom slovu riječi svake rečenice. Funkcija se primjenjuje na stupcu source te se rezultati sprema u stupac noCap_LetterWords_inSentence okvira podataka. Rezultate je moguće vidjeti unutar naslova 8.9.1 Rezultati određivanja i računanja značajki.

```
#@title Default title text
def CapWords(df):
    return sum(1 for c in df if c.isupper())
```

Slika 0-21: Funkcija prebrojavanja riječi sa vlikim početnim slovom

Način dobivanja dužine rečenica unutar svakog od tekstova pojašnjeno je unutar naslova 5.2.1 Opis i rezultati prosječne duljine rečenica, riječi, prosječan broj rečenica unutar teksta. Postupak je isti u ovom slučaju.

Kako bi se izračunao broj riječi u rečenici koji su dio sažetka, definira se funkcija W_sourceSummary() koja kao parametre uzima tekstove iz stupca te se sastoji od sljedećih koraka:

1. Definira se prazna lista u koju ćemo kasnije pohraniti rezultate.

2. Sažetci iz stupca summary se spremaju u varijablu text2 iz kojih se potom uklanjaju svi interpunkcijski znakovi.
3. Tekstovi stupca source se razbijaju na rečenice, a sažetci na riječi pošto sažetci variraju od 1 do više rečenice te nema potrebe raditi duplo računanje što konzumira više vremena. Također uklanjanjem interpunkcijskih znakova iz jednog teksta je dovoljno tijekom primjene sljedećeg koraka.
4. Iteriramo po rečenicama tekstova za sažimanje, razdvajamo rečenice na listu riječi, te, kako bi se mogla koristiti funkcija intersection(), pretvaramo liste u skupove. Traži se presjek dvaju lista kako bismo dobili riječi korištene unutar oba teksta, a rezultat se ponovno pretvara nazad u listu te se uzima dužina liste koju potom pohranjujemo unutar liste f definirane na početku funkcije te se vraća kao rezultat računanja.
5. Pozivamo funkciju nad stupcima source i summary te se definira lambda funkcija koja će iterirati po stupcima i kao parametre W_sourceSummary() će dobivati tekstove u string formatu, a na kraju obrade, svi rezultati se pohranjuju u stupac no_words_inSent_SS.

```
def W_sourceSummary(df, df2):
    text2 = df2
    text2 = text2.translate(str.maketrans('', '', string.punctuation))
    tokens = sent_tokenize(df, language = "german")
    tok2 = word_tokenize(text2, language = "german")
    for s in tokens:
      #print (s)
      return len(set(s.split()).intersection(set(tok2)))
```

Slika 0-22: Funkcija broja riječi u rečenici koje su također nalaze unutar sažetaka

Na kraju potrebno je dodatno izračunati broj riječi u rečenici koje su ključne riječi. Kako bi se izračunalo navedeno koristimo malo prerađenu prethodnu funkciju te je potrebno listu ključnih riječi pretvoriti u stringove. Definiraju se funkcije W_sourceKeywords() i listToString(). Ovoga puta se sve riječi iz source stupca prebacuju u mala slova dok je ostatak koraka jednak kao i u funkciji W_sourceSummary().

```
def W_sourceKeywords(df, df2):
    df.lower()
    text2 = df2
    text2 = text2.translate(str.maketrans('', '', string.punctuation))
    tokens = sent_tokenize(df, language = "german")
    tok2 = word_tokenize(text2, language = "german")
    for s in tokens:
        return len(set(s.split()).intersection(set(tok2)))

def listToString(df):
    string = ' '.join([str(e) for e in df])
    return string
```

```
[ ] df['no_words_inSent_SS'] = df.apply(lambda x: W_sourceSummary(x['source1'], x['summary']), axis=1)
    df['keywords'] = df['keywords'].apply(listToString)
    df['no_words_inSent_SK'] = df.apply(lambda x: W_sourceKeywords(x['source1'], x['keywords']), axis=1)
    df['noCap_LetterWords_inSentence'] = df['source1'].apply(CapWords)
```

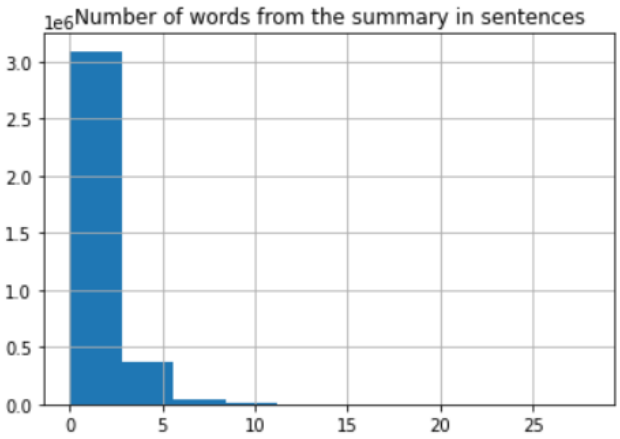
Slika 0-23:Funkcije W_sourceKeywords() i listToString() za određivanje broja riječi u rečenici koje su ključne riječi

Nakon navedenog potrebno je označiti rečenice s „Yes“ i „No“ kako bi se te iste oznake mogle iskoristiti u daljnjim koracima klasifikacije tekstova. Navedeno je moguće napraviti na temelju značajki pomoću biblioteka matplotlib.pyplot i numpy. Promatranjem skupa podataka moguće je vidjeti kako stupci koji govore o broju riječi s velikim početnim slovom i broju riječi sadržanih unutar rečenice i sažetaka daju najbolji odnos između navedene dvije klase za učenjenje modela.

Princip provjere stanja podataka u trenutnom skupu pomoću histograma je sljedeći:

```
1 import matplotlib.pyplot as plt
2 plt.title("Number of words from the summary in sentences")
3 df['no_words_inSent_SS'].hist()
```

```
<AxesSubplot:title={'center':'Number of words from the summary in sentences'}>
```

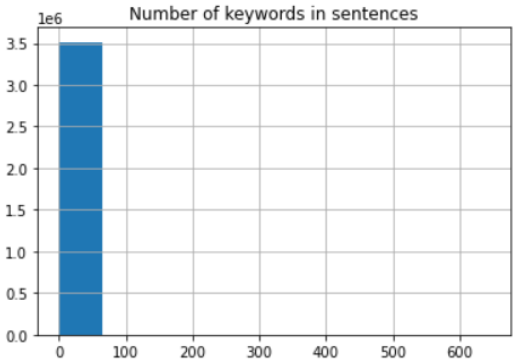


Slika 0-24: Prikaz pregleda podataka u potrazi za najboljim čimbenicima označavanja skupa podataka

Isto je moguće napraviti i za stupac koji predstavlja broj riječi u rečenici koje započinju velikim početnim slovom:

```
1 import matplotlib.pyplot as plt
2 plt.title("Number of keywords in sentences")
3 df['noCap_LetterWords_inSentence'].hist()
```

```
<AxesSubplot:title={'center':'Number of keywords in sentences'}>
```



Slika 0-25: Prikaz podataka na temelju stupca o broju riječi u rečenici s velikim početnim slovom.

Sada je moguće kombinirati dva navedena stupca kako bi se stvorio novi stupac unutar okvira podataka koji će sadržavati navedene oznake:

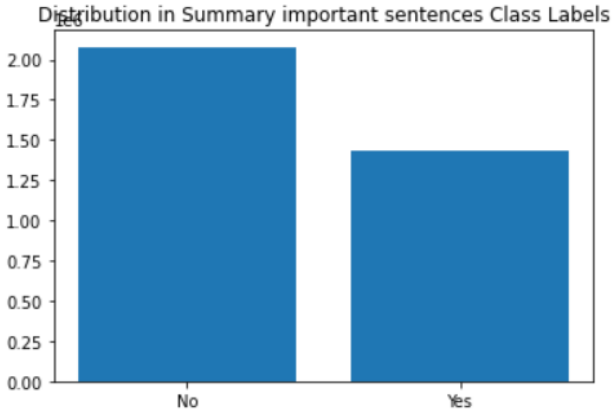
```
1 df['sim_sent'] = np.where((df.noCap_LetterWords_inSentence>8.0) | (df.no_words_inSent_SS>=2.0), 'Yes', 'No')
```

0-26: Obilježavanje seata podataka.

Kada se promotre dobiveni rezultati na prethodne načine kako bi se utvrdio balans pozitivnih i negativnih oznaka:

```
1 from collections import Counter
2 plt.title("Distribution in Summary important sentences Class Labels")
3 plt.bar(dict(Counter(df['sim_sent'])).keys(), dict(Counter(df['sim_sent'])).values())
```

```
<BarContainer object of 2 artists>
```



Slika 0-27: Balans pozitivno i negativno označenih rečenica.

Kroz kasnija testiranja navedeni način označavanja rečenica u klasu sažetka pokazao se najboljim. Bilo kakvom drugom kombinacijom lako se dolazi do bilo kakvog pretjeranog prilagođenog modela. Više o tome u kasnijim dijelovima.

Sada imamo sve ključne značajke za početak klasifikacije po više vrsta klasifikatora te, radi jednostavnosti skupa podataka i kasnijeg procesiranja i obrade, mogu se izbaciti stupci koji sadrže originalne tekstove i sažetke. Rezultat se pohranjuje u novu csv datoteku kako bi se očuvali rezultati te ne bi utjecalo na originalni skup podataka.

Sve što preostaje je izračunati TF-IDF, prosječni word2vec rečenice i doc2vec vektor rečenica te centroid dokumenta. Potom se primjenjuju 4 različita algoritma sličnosti.

5.4.2 Vektori rečenica

U ovome radu se primjenjuju tri vrste vektorizacije tekstova pomoću:

- TF-IDF* vektor rečenica,
- Prosječnih word2vec vektor rečenica,
- Doc2vec vektor rečenica.

Nakon što su primijenjeni prethodni koraci računanja vrijednosti navedenih značajki tekstova, trenutni skup podataka sadrži preko 3,5 milijuna rečenica s odgovarajućim značajkama. Prethodno vektorizaciji potrebno je lematizirati te ukloniti stop-riječi. Za lematizaciju tekstova koristi se nltk-ov WordNet lematizator (Beli 2018) te skup stop riječi iz nltk.corpus biblioteke za njemački jezik. Sve navedene vektorizacije se provode na stupcu source1 koji predstavlja stupac rečenica.

```
nltk.download('wordnet') # first-time use only
german_stop_words = stopwords.words('german')
lemmer = nltk.stem.WordNetLemmatizer()
def LemTokens(tokens):
    return [lemmer.lemmatize(token) for token in tokens]
remove_punct_dict = dict((ord(punct), None) for punct in string.punctuation)
def LemNormalize(text):
    return LemTokens(nltk.word_tokenize(text.lower().translate(remove_punct_dict)))
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Dorian\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

```
stopword_list = nltk.corpus.stopwords.words('german')

tv = TfidfVectorizer(min_df = 0.05, max_df = 0.8, tokenizer=LemNormalize, stop_words = stopword_list)
X = tv.fit_transform(df['source1'])
vocab = tv.get_feature_names()
```

Slika 0-28: Lematizacija te računanje TF-IDF vrijednosti.

Na Slika 0-28: Lematizacija te računanje TF-IDF vrijednosti. se prikazuje postupak lematizacije i vektorizacije rečenica te stvaranje vokabulara koji je bitan u sljedećem koraku.

Kako bi se uz izračunate TF-IDF feature sačuvale vrijednosti koje su ostalim stupcima potrebno je X pretvoriti u polje, a potom i u pandas okvir podataka gdje će obilježja dobivena ovim načinom biti svrstana u zasebne stupce imenovanima na temelju vrijednosti spremljenih unutar vocab varijable. Potom se stupac po stupac dodaju prethodno izračunate značajke:

```
X1 = pd.DataFrame(X.toarray(), columns = vocab)
X1['no_words_inSent_SS'] = df['no_words_inSent_SS']
X1['no_words_inSent_SK'] = df['no_words_inSent_SK']
X1['noCap_LetterWords_inSentence'] = df['noCap_LetterWords_inSentence']
```

Slika 0-29: Pretvaranje izračunatih značajki u okvir podataka te dodavanje svih stupaca iz originalnog skupa podataka koji predstavljaju obilježja izračunana prethodno vektorizaciji.

Dobiveni okvir podataka se potom pretvara u sparse matricu te iz sparse matrice u dense matricu kako bi se mogla napraviti raspodjela skupa podataka na 70% za učenje i 30% za evaluaciju. Svi ovi koraci su potrebni jer train_test_split() funkcija ne prima sparse matricu kao ulazni oblik podataka[11]:

```
X_sparse = sparse.csr_matrix(X1.values)
X = X_sparse.toarray()
y = df.sim_sent.values
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.3)
```

Slika 0-30: Pretvaranje okvira podataka u sparse matricu pa potom u dense matricu, definiranje skupa značajki i skupa oznaka (stupac sim_sent koji sadrži oznake "Yes" i "No") te podjela skupa podataka na 70% za učenje i 30% za učenje.

5.4.3 Klasifikacija primjenom slučajnih šuma

Kada su izračunati prethodni vektori i značajke može se započeti s klasifikacijom. Ovdje će nam biti potrebni pročišćeni tekstovi od stop-riječi kao i u svakom od klasifikatora. Kako bi se postigla klasifikacija slučajnih šuma potrebne su nam biblioteke sklearn.ensamble iz kojega uzimamo klasifikator slučajnih šuma.

Prvenstveno se definiraju X i y pri čemu X predstavlja tekst na kojemu će se vršiti klasifikacija dok lista sadržana od niza jedinica i nula predstavlja koje od navedenih rečenica pripadaju pozitivnoj klasi sažetka ili negativnoj klasi. Potom se skup dijeli na omjer od 70/30 pri čemu se koristi 70% za učenje, a 30% za evaluaciju. Sljedeći korak nakon ovoga je sama klasifikacija.

[11] Tijekom detaljnijeg istraživanja iako je riječ o sparse.csr_matrix() formatu podataka, train_test_split() argument stratify nije kompatibilan niti sa jednim formatom sparse matrica ili pak ne radi dobro. Stoga je bilo preporučeno napraviti sve navedene korake pretvorbe kako bi se moglo uključiti u podjelu skupa na učenje i testiranje. Primjer jednog takvog problema: https://stackoverflow.com/questions/57860726/how-to-split-sparse-matrix-into-train-and-test-sets

Kada je riješen prvi način vektorizacije, prelazi se na word2vec vektorizaciju rečenica. Ponovo se uzima originalni skup podataka. Ovoga puta, radi ograničenja hardware-a, ponajviše RAM memorije, uzima se 60% skupa podataka te se taj skup dijeli na 70% za učenje i 30% za učenje. Definiraju se funkcije sentence_to_wordlist(), koja pretvara rečenice u liste riječi te uklanja stop-riječi, i text_to_sentences(), koja prepoznaje rečenice te poziva unutar sebe navedenu funkciju kako bi pretvorila rečenice u liste riječi s uklonjenim stop riječima.

```
stopword_list = nltk.corpus.stopwords.words('german')
def sentence_to_wordlist(sentence, remove_stopwords=False):
    sentence_text = re.sub("[^a-zA-Z]"," ", sentence)

    # convert to lower case and split at whitespace
    words = sentence_text.lower().split()

    # remove stop words (false by default)
    if remove_stopwords:
        stops = set(stopword_list)
        words = [w for w in words if not w in stops]

    return words
```

```
def text_to_sentences(sentence, remove_stopwords=False):
    # use the NLTK tokenizer to split the paragraph into sentences
    raw_sentences = sent_tokenize(sentence, language = "german")

    # each sentence is furthermore split into words
    sentences = []
    for raw_sentence in raw_sentences:
        # If a sentence is empty, skip it
        if len(raw_sentence) > 0:
            sentences.append(sentence_to_wordlist(raw_sentence, remove_stopwords))

    return sentences
```

Slika 0-31: Funkcije za stvaranje liste riječi rečenica.

Inicijalizira se prazna lista koja će predstavljati skup rečenica dobivenih pozivom funkcije text_to_sentences() na skupu podataka bez parametra uklanjanja stop riječi:

```
train_sentences = []  # Initialize an empty list of sentences
for sent in train_df['source1']:
    train_sentences += text_to_sentences(sent)
```

Sada je sve spremno za učenje word2vec modela. Postavljaju se parametri broja značajki, broja najmanje frekvencije pojavljivanja riječi, broj niti za paralelizaciju, veličina prozora itd. Za svaki slučaj se provjerava postoji li već stvoreni model, izgradi se novi temeljem navedenih parametara te zamijeni stari model s novim.

```
model_name = 'train_model'
# Set values for various word2vec parameters
num_features = 300    # Word vector dimensionality
min_word_count = 40   # Minimum word count
num_workers = 3       # Number of threads to run in parallel
context = 10          # Context window size
downsampling = 1e-3   # Downsample setting for frequent words
if not os.path.exists(model_name):
    # Initialize and train the model (this will take some time)
    model = word2vec.Word2Vec(train_sentences, workers=num_workers, \
                vector_size=num_features, min_count = min_word_count, \
                window = context, sample = downsampling)

    # If you don't plan to train the model any further, calling
    # init_sims will make the model much more memory-efficient.
    model.init_sims(replace=True)

    # It can be helpful to create a meaningful model name and
    # save the model for later use. You can load it later using Word2Vec.load()
    model.save(model_name)
else:
    model = Word2Vec.load(model_name)
```

Slika 0-32: Treniranje word2vec modela.

Kada je model napravljen potrebno je generirati prosječne word2vec vektore. Ti vektori se pohranjuju unutar varijable trainDataVecs koja poziva funkciju get_avg_feature() s modelom, brojem značajki te listom rečenica u kojima su uklonjene stop riječi ako parametrima.

```
def make_feature_vec(words, model, num_features):
    """
    Average the word vectors for a set of words
    """
    feature_vec = np.zeros((num_features,),dtype="float32")  # pre-initialize (for speed)
    nwords = 0.
    index2word_set = set(model.wv.index_to_key)  # words known to the model

    for word in words:
        if word in index2word_set:
            nwords = nwords + 1.
            feature_vec = np.add(feature_vec,model.wv[word])

    feature_vec = np.divide(feature_vec, nwords)
    return feature_vec


def get_avg_feature_vecs(sents, model, num_features):
    """
    Calculate average feature vectors for all sentences
    """
    counter = 0
    review_feature_vecs = np.zeros((len(reviews),num_features), dtype='float32')  # pre-initialize (for speed)

    for review in reviews:
        #print(review)
        review_feature_vecs[counter] = make_feature_vec(review, model, num_features)
        counter = counter + 1
    return review_feature_vecs
```

Slika 0-33: Funkcija get_avg_feature_vecs() poziva funkciju make_feature_vec() koja računa prosječne vektore riječi na skupu riječi. Potom se unutar funkcije get_avg_feature_vecs() računa isto za sve rečenice.

Pošto prethodni korak može rezultirati vrijednostima None, potrebno je ih je ukloniti iz skupa prethodno klasifikaciji. For petljom se prolazi kroz polje vektora te se sve rečenice sa nan vrijednostima vektora spremaju u listu nan_indices. Potom, ako navedena lista nije prazna, iz trainDocVecs se uklanjaju se vrijednosti koje su unutar nan_indices te se također brišu rečenice unutar skupa za učenje koji korespondiraju obrisanim vektorima unutar trainDocVecs. Cijeli ovaj postupak se također ponavlja i za testni dio skupa podataka.

```
clean_train_sent = []
for sent in train_df['source1']:
    clean_train_sent.append(sentence_to_wordlist(sent, remove_stopwords=True))
trainDataVecs = get_avg_feature_vecs(clean_train_sent, model, num_features)
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:14: RuntimeWarning: invalid value encountered in true_divide
```

```
nan_indices = list({x for x,y in np.argwhere(np.isnan(trainDataVecs))})
if len(nan_indices) > 0:
    print('Removing {:d} instances from train set.'.format(len(nan_indices)))
    trainDataVecs = np.delete(trainDataVecs, nan_indices, axis=0)
    train_df.drop(train_df.iloc[nan_indices, :].index, axis=0, inplace=True)
    assert trainDataVecs.shape[0] == len(train_df)
```

```
Removing 38622 instances from train set.
```

Slika 0-34: Proces stvaranja prosječnih word2vec vektora te uklanjanje nan vrijednosti vektora iz skupa.

Kada je napravljeno isto i za testni dio skupa podataka, sve je spremno za klasifikaciju.

Na kraju ostaje još samo pretvaranje rečenica u doc2vec vektore. Kao i za word2vec, ovdje se također uzima 60% inicijalnog skupa podataka. Navedeni manji dio se dijeli na 70% za učenje te 30% za učenje pomoću train_test_split() funckije. Potom se rečenice rastavljaju na riječi pomoću funkcije word_tokenize() te se sve riječi prebacuju u mala pisana slova i dodaju u listu tokens koja je rezultat funkcije tokenize_text(text). Pomoću Taggeddocument() funkcije unutar koje se poziva funkcija tokenize_text() sa stupcem rečenica kao parametar izvora riječi, dok stupac oznaka služi kao parametar oznake modela. Ovaj korak se radi i za skup za učenje i testiranje.

```
nltk.download('punkt')
def tokenize_text(text):
    tokens = []
    for word in nltk.word_tokenize(text, language = "german"):
        if len(word) < 2:
            continue
        tokens.append(word.lower())
    return tokens

train_tagged = train.apply(
    lambda r: TaggedDocument(words=tokenize_text(r['source1']), tags=[r.sim_sent]), axis=1)
test_tagged = test.apply(
    lambda r: TaggedDocument(words=tokenize_text(r['source1']), tags=[r.sim_sent]), axis=1)
```

Slika 0-35: Označavanje skupova za učenje i testiranje.

Kada se izvrši prethodni korak, uči se doc2vec model u 30 epoha koristeći train_tagged i test_tagged skupove.

```
model_dbow = Doc2Vec(dm=0, vector_size=200, negative=5, hs=0, min_count=2, sample = 0, workers=cores)
model_dbow.build_vocab([x for x in tqdm(train_tagged.values)])
```

```
100%|██████████| 1476085/1476085 [00:00<00:00, 2264532.92it/s]
```

```
for epoch in range(30):
    model_dbow.train(utils.shuffle([x for x in tqdm(train_tagged.values)]), total_examples=len(train_tagged.values), epochs=
    model_dbow.alpha -= 0.002
    model_dbow.min_alpha = model_dbow.alpha
```

Slika 0-36: Treniranje doc2vec modela u 30 epoha.

U posljednjem koraku prethodno klasifikaciji se definira funkcija vec_for_learning() koja stvara završni skup značajki vektora za klasifikator koje se potom spremaju u odgovarajuće varijable X_train, X_test, y_train i y_test.

```
def vec_for_learning(model, tagged_docs):
    sents = tagged_docs.values
    targets, regressors = zip(*[(doc.tags[0], model.infer_vector(doc.words, steps=20)) for doc in sents])
    return targets, regressors
```

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
y_train, X_train = vec_for_learning(model_dbow, train_tagged)
y_test, X_test = vec_for_learning(model_dbow, test_tagged)
```

Slika 0-37: Završna pripema podataka ped klasifikaciju.

5.4.4 Klasifikacija algoritmom Naîvnog Bayesa

Kada su riješeni svi prethodni koraci vektorizacije može se prijeći na klasifikaciju. Rezultati klasifikatora su prikazani pod naslovom 6.3 Usporedba rezultata algoritama klasifikacije. U ovome radu se koristi GaussianNB iz biblioteke sklearn.naive_bayes. Te je princip klasifikacije za sve vektore isti s razlikom u nazivima skupova. Svaki od klasifikatora je napravljen na način prezentiran na Slika 0-38: Algoritam klasifikacije Naîvnog Bayesa.

```
from sklearn.naive_bayes import GaussianNB
print("n============ Fitting Naive Bayes...")
gnb_classifier = GaussianNB()
gnb_classifier.fit(X_train, y_train)
y_pred = gnb_classifier.predict(X_test)
print("\n============ Naive Bayes")
print("---- confusion_matrix:")
print(confusion_matrix(y_test,y_pred))
print("---- classification_report:")
print(classification_report(y_test,y_pred))
print("---- accuracy_score:")
print(accuracy_score(y_test,y_pred))
```

Slika 0-38: Algoritam klasifikacije Naîvnog Bayesa.

5.4.5 SVM klasifikacija

Kao i u prethodnom naslovu princip korištenja ovog klasifikatora je jednak za sve vrste vektorizacija. Radi bržeg izračuna koristi se linearni SVM klasifikator s razlikom u nazivima skupova podataka za učenje i učenje.

```
from sklearn import svm
svm_classifier = svm.LinearSVC()
print("n============ Support Vector Machine...")
svm_classifier.fit(X_train, y_train)

y_pred = svm_classifier.predict(X_test)
print("\n============ Support Vector Machine")
print("---- confusion_matrix:")
print(confusion_matrix(y_test,y_pred))
print("---- classification_report:")
print(classification_report(y_test,y_pred))
print("---- accuracy_score:")
print(accuracy_score(y_test,y_pred))
```

Slika 0-39: Prikaz učenja SVM klasifikatora na skupu podataka.

5.4.6 Klasifikator slučajnih šuma

Za razliku od prethodnih klasifikatora, algoritam slučajnih šuma se kod određenih vektorizacija primjenjuje s različitim vrijednostima i skupovima parametara u svrhu boljih rezultata. Ostali princip primjene navedenog klasifikatora je isti.

```
from sklearn.ensemble import RandomForestClassifier
print("n============ Fitting Random Forest...")
forest_classifier = RandomForestClassifier()
forest_classifier.fit(X_train, y_train)

y_pred = forest_classifier.predict(X_test)
print("\n============ Random Forest")
print("---- confusion_matrix:")
print(confusion_matrix(y_test,y_pred))
print("---- classification_report:")
print(classification_report(y_test,y_pred))
print("---- accuracy_score:")
print(accuracy_score(y_test,y_pred))
```

Slika 0-40: Primjena klasifikatora slučajnih šuma na doc2vec vektorima rečenica.

5.5 Opis metoda ekstraktivnog sažimanja

Ovdje se opisuju metode sažimanja teksta. Kao što je navedeni unutar naslova 5 Skup podataka, opis pristupa i statistički opis korpusa koristi se nekoliko pristupa ovom zadatku. Ovdje se opisuju te testiraju rezultati i načini rada svakog od navedenih algoritama.

Kako bismo stvorili usmjereni graf rečenica te usporedili njihovu sličnost na temelju različitih mjera sličnosti, potrebno je proći kroz sve korake u 5.3 Priprema podataka prethodno procesiranju. Kad smo očistili tekst, generiraju se TF-IDF matrice na temelju rečenica. Osim TF-IDF rezultata, primjenjivat ćemo iste algoritme na temelju doc2vec i word2vec rezultata. Tijekom primjene doc2vec potrebno je uzeti u obzir prosječnu vrijednost vektora riječi unutar rečenice prilikom izračuna, te primijeniti navedene algoritme izračuna sličnosti rečenica.

5.5.1 Stvaranje grafa rečenica na temelju TF-IDF rezultata

Kako bi se izračunala vrijednost matrica, primjenjuje se sklearn biblioteka. Definiramo varijablu u koju spremamo definirani način izračuna TF-IDF vektora, koji potom pozivamo na prethodno procesiran tekstove.

Potom je potrebno definirati mjere sličnosti. Za sličnost kosinusa koristimo onu definiranu unutar sklearn.metrics dostupna je funkcija izračuna sličnosti kosinusa. Po primjeni TF-IDF rezultata formiramo matricu kosinusne sličnosti rečenica.

Jaccardova sličnost također se koristi navedenim vrijednostima. Kako bismo dobili Jaccardovu sličnost između rečenica, koristi se biblioteka sklearn.metric.pariwise iz koje koristimo Jaccardovu mjeru sličnosti. Kod za računanje Jaccardove sličnosti je stoga sljedeći:

```
from sklearn.metrics.pairwise import pairwise_distances
jaccard = pairwise_distances(dist_df.values, metric='jaccard')
sns.heatmap(pairwise_distances(dist_df.values, Y=None, metric='jaccard'));
```

Mihalcea sličnost rečenica ne uzima TF-IDF rezultate već se primjenjuje formula 33) pod naslovom 6.1 Grafovi algoritama sličnosti rečenica.

5.5.2 LexRank

Način funkcioniranja i detaljan opis rada ovog algoritma sažimanja tekstova opisan je unutar 2.1.3 LexRank i uloga u sažimanju tekstova. Kako bismo primijenili algoritam učitavamo seaborn za kasniji prikaz rezultata te iz lexrank biblioteke preuzimamo STOPWORDS i LexRank algoritam. Primjena LexRank algoritma na tekstu je sljedeća:

```
documents = []
documents = df['source']
lxr = LexRank(documents, stopwords=STOPWORDS['de'])
sentances = df['source'][0]
tokens = sent_tokenize(sentances, language='german')
#print(tokens)
lexrank = lxr.get_summary(tokens, summary_size=5, threshold=.1, fast_power_method = True)
print(lexrank)
```

Slika 0-41: Primjena LexRanka na njemačkim tekstovima

Definira se skup dokumenata koji je u ovom slučaju stupac source našeg okvira podataka. U trećem retku čistimo naše dokumente od stop-riječi te, u našem slučaju, uzimamo prvi tekst radi konzistentnosti primjera i uspoređivanja rezultata kroz cijeli rad. Potom rastavljamo tekst na rečenice te se postavlja njemački jezik za bolje prepoznavanje teksta.

Kada smo napravili prethodno koristimo funkciju get_summary() koje može primat više parametara:

- Prvi parametar su definirane i razdvojene rečenice našeg teksta.
- Drugim parametrom, summary_size, se definira veličina generiranoga sažetka. Tako smo dobili rezultate vidljive u 8.8.1 Ispis Lexrank rezultat
- Parametrom threshold se definira prag na temelju kojeg se određuje koje rečenice imaju dovoljno visoki indeks kako bi se mogle iskoristiti za generiranje sažetka
- Posljednji parametar, fast_power_method, se može definirati s True ili False. Kad se ovaj parametar postavi na True, program koristi više radne memorije kako bi brže napravio izračune i definirao sažetak

Svi rezultati ovog algoritma su vidljivi unutar 8.8.1 Ispis Lexrank rezultat.

Osim navedene biblioteke također je moguće primijeniti LexRank algoritam iz sumy biblioteke čiji su rezultati također prikazani unutar privitka pod naslovom 8.8.2 Ispis rezultata LexRank algoritma iz sumy biblioteke. Njegova primjena je puno lakša i brže za obradu. Dovoljno je primijeniti parser koji će prepoznati i procesirati njemačke tekstove te se potom poziva LexRankSumarizer kao algoritam sažimanja tekstova. Sve navedeno je definirano unutar funkcije lexRank()

```
def lexRank(df, size):
    parser = PlaintextParser.from_string(df, Tokenizer("german"))
    summarizer = LexRankSummarizer()
    summary = summarizer(parser.document, size)
    for s in summarizer(parser.document, size):
        print(s)
```

Slika 0-42: Funkcija lexRank() koja je implementacija LexRank algoritma sumy biblbioteke

5.5.3 TextRank

TextRank je još jedan od algoritama sažimanja koristi se biblioteka sumy u kojoj su također sadržani i sumarizator na temelju Kullback-Leiblerove divergencije, LexRank, LSA (*Latent Semantic Analysis*) te mnogi drugi.

Prvenstveno definiramo sve potrebne alate te ih pozivamo iz biblioteke, nakon čega učitavamo okvir podataka.

```
[18] from sumy.summarizers.text_rank import TextRankSummarizer
     from sumy.parsers.plaintext import PlaintextParser
     from sumy.nlp.tokenizers import Tokenizer
     textR = TextRankSummarizer()
     import nltk, pandas as pd
     nltk.download('punkt')

     [nltk_data] Downloading package punkt to /root/nltk_data...
     [nltk_data]   Unzipping tokenizers/punkt.zip.
     True

     df = pd.read_csv('/content/data_test.csv', encoding='utf-8')
     df.head()
```

	source
0	Das Kelvin wurde durch die Generalkonferenz fu...
1	Campbell wurde als Tochter eines schottischen ...
2	Roland Emmerich wurde in Stuttgart-Obertürkhe...
3	Das Stasi-Unterlagen-Gesetz ist die gesetzlich...
4	Der Enzkreis hat Anteil am Nordschwarzwald, Kr...

Slika 0-43: Definiranje alata i učitavanja skupa podataka

Potom se definira jezik na kojem će se raditi, a to je njemački. Također se definira funkcija textRank() koja kao argumente prima okvir podataka i veličinu sažetka. Pomoću PlaintextParser metode označavamo tekst u odnosu na jezik. U summary spremamo sažetak koji se dobije kao rezultat ovog algoritma. Potom se funkcija poziva na sve tekstove okvira podataka te u zasebne stupce pohranjujemo sve sažetke ovisno o tome želimo li dobiti kao rezultat top 1, 2, 5 ili 10 rečenica teksta.

```
 LANGUAGE = "german"
# stopword_list = nltk.corpus.stopwords.words('german')
def textRank(df, size):
    parser = PlaintextParser.from_string(df, Tokenizer(LANGUAGE))
    summary = textR(parser.document,size)
    return summary

df['TextRank2']=df.apply(lambda x: textRank(x['source'], 2), axis=1)
df['TextRank5']=df.apply(lambda x: textRank(x['source'], 5), axis=1)
df['TextRank10']=df.apply(lambda x: textRank(x['source'], 10), axis=1)
```

```
[31] df['TextRank1']=df.apply(lambda x: textRank(x['source'], 1), axis=1)
```

Slika 0-44: Pozivanje TextRank funkcije na tekstove te izvlačenje top 1.2.5 i 10 rečenica

Po izlazu prezentiranom pod naslovom 8.8.3 Ispis TextRank rezultat potrebno je očistiti podatke od do određenih oznaka kao što su „(<Sentence:“, „>,“, „>)“ ili „<Sentence:“ prethodno evaluaciji. Kako bi se postiglo željeno, koriste se regularni izrazi za čišćenje podataka u svrhu dobivanja čistih rečenica.

Također kako bis e mogao izračunati LexRank rezultat na temelju svih drugih sličnosti definira se funkcija lex_Score() koja kao argumente prima okvir podataka, broj rečenica od kojih želimo formirati sažetak te prag na temelju kojeg se filtrira graf. Po dodavanju rubova i vrhova u graf, definira se lista u koju ćemo pohranjivati rezultate. Koristi se pagerank() funkcija biblioteke nx budući da je LexRank indeks prilagođena PageRank funkcija. Postavlja se parametar weights koji koristi težinske vrijednosti na bridovima koje su prethodno izračunate na temelju mjera sličnosti. Rezultati se spremaju u listu u obliku parova rečenica i njihovih odgovarajućih Lexrank rezultata, odabire se top n rečenica te na kraju stvara tekst od odabranih rečenica. Slika 0-45: Primjer LexRank rezultata sa Jaccardovom sličnosti rečenica. Postupak je također isti i za druge algoritme sličnosti rečenica prikazuje primjer funkcije koja računa rezultate na temelju Jaccardove sličnosti između rečenica. Primjer ove funkcije je isti i za ostale algoritme sličnosti.

```
def lex_Score(df, top_n, thresh):
    sentences = sent_tokenize(df, language="german")
    g=nx.DiGraph()
    edges = [(i,j,jaccard(x,y))
             for i,x in enumerate(sentences)
             for j,y in enumerate(sentences) if i < j]

    g.add_weighted_edges_from((i,j,sim) for i,j,sim in edges)
    lst,res = [],[]
    pr = nx.pagerank(g, weight='weight')

    for s, i in enumerate(sentences):
        lst.append((i, pr[s]))
    top = nlargest(top_n, lst, key=itemgetter(1))
    top_s = [x[0] for x in top]
    string = ' '.join([str(e) for e in top_s])
    return (string)
```

Slika 0-45: Primjer LexRank rezultata sa Jaccardovom sličnosti rečenica. Postupak je također isti i za druge algoritme sličnosti rečenica.

5.5.4 Algoritam sažimanja KL divergencije- sumy paket

Primjena ovog algoritma sažimanja je slična onoj LexRanka iz iste biblioteke. Ponovno se uzima manji skup od 50 tekstova kojima se uklanjaju stop riječi pomoću funkcije remove_stopwords(). Potom se definira funkcija Kldiv() koja kao parametre uzima očišćene tekstove te jezik za obilježavanje. Potom se parsira dokument, sprema u listu te kao rezultat vraća sažeti tekst.

```
stopword_list = nltk.corpus.stopwords.words('german')
LANGUAGE = "german"
def remove_stopwords(text):
    filtered_words = [word for word in nltk.word_tokenize(text, language="german") if word not in stopword_list]
    filtered_text = ' '.join(filtered_words)
    return filtered_text

def KLdiv(df, size):
    lst = []
    parser = PlaintextParser.from_string(df, Tokenizer(LANGUAGE))
    summarizer = KLSummarizer()
    summary = summarizer(parser.document, size)
    for s in summarizer(parser.document, size):
        lst.append(s)
    string = ' '.join([str(e) for e in lst])
    return string
```

Slika 0-46: Funkcije za uklanjanje stop riječi te stvaranje sažetaka na temelju algoritma Kullback-Leiblerove divergencije.

Po definiciji funkcija i pozivanju funkcija za stvaranje sažetaka koji sadrže top jednu, dvije, pet i deset rečenica, primjenjuje se Rouge mjera za utvrđivanje točnosti dobivenih sažetaka u usporedbi s referenciranim sažetcima skupa podataka. Rezultati se pohranjuju unutar definirane tekstualne datoteke.

```
test_df['source'] = test_df['source'].apply(remove_stopwords)
test_df['kl1'] = test_df.apply(lambda x: KLdiv(x['source'], 1), axis=1)
test_df['kl2'] = test_df.apply(lambda x: KLdiv(x['source'], 2), axis=1)
test_df['kl5'] = test_df.apply(lambda x: KLdiv(x['source'], 5), axis=1)
test_df['kl10'] = test_df.apply(lambda x: KLdiv(x['source'], 10), axis=1)
test_df.to_csv('G:/Extractive-Summarisation-of-German-Wikipedia/dataset/kl-divergenceR2.csv')
```

```
from rouge import Rouge
import sys

sys.stdout = open("G:/Extractive-Summarisation-of-German-Wikipedia/dataset/kl_test.txt", "w")

hy, rf = test_df['kl1'], test_df['summary']
rouge= Rouge()
scoresT1= rouge.get_scores(hy, rf, avg=True)
print("KL1 scores: ", scoresT1, '\n\n')

hy2 = test_df['kl2']
scoresT2= rouge.get_scores(hy2, rf, avg=True)
print("KL2 scores: ", scoresT2, '\n\n')

hy5 = test_df['kl5']
scoresT5= rouge.get_scores(hy5, rf, avg=True)
print("KL5 scores: ", scoresT5, '\n\n')

hy10 = test_df['kl10']
scoresT10= rouge.get_scores(hy10, rf, avg=True)
print("KL10 scores: ", scoresT10, '\n\n')

sys.stdout.close()
```

Slika 0-47: Formiranje sažetaka sa top 1, 2, 5 ili 10 rečenica te evaluacija dobivenih rezultata.

5.5.5 SBKE

Selectivity Based Keyword Extraction (Beliga, Martinčić-Ipšić i Meštrović 2016) primjenjuje stupanj selektivnosti čvora koji je prikazan pod naslovom 2.1.2 Metode rangiranja rečenica na bazi grafova u formuli na stranici 20. Kako bismo primijenili funkciju na grafove koje smo prethodno stvorili definira se funkcija sbke_score() koja ima dva parametra. Parametrom top_n se definira koliko top rečenica se želi predstaviti kao izlaz, te okvir podataka iz kojeg prosljeđujemo podatke.

```
def sbke_score(df, top_n):
    sentences = sent_tokenize(df, language="german")
    '''g=nx.DiGraph()
    edges = [(i,j,mihalcea(x,y))
             for i,x in enumerate(sentences)
             for j,y in enumerate(sentences) if i < j]

    g.add_weighted_edges_from((i,j,sim) for i,j,sim in edges)'''
    lst =[]
    for i in g.nodes():
        no_nei = len(g.edges(i))
        if no_nei !=0 :
            w_sum = g.degree(i,weight='weight')
            w_sum += w_sum
            alpha = 1-w_sum
            sbke = no_nei*(w_sum/no_nei)**alpha
            lst.append((sentences[i], sbke))
        else:
            lst.append((sentences[i], 0))

    top = nlargest(top_n, lst, key=itemgetter(1))
    top_s = [x[0] for x in top]
    string = ' '.join([str(e) for e in top_s])
    return string

test_df = df.head(50)
test_df['sbke_miha1'] = test_df.apply(lambda x: sbke_score(x['source'], 1, 0
test_df['sbke_miha2'] = test_df.apply(lambda x: sbke_score(x['source'], 2, 0
test_df['sbke_miha5'] = test_df.apply(lambda x: sbke_score(x['source'], 5, 0
test_df['sbke_miha10'] = test_df.apply(lambda x: sbke_score(x['source'], 10,
df.to_csv('G:/Extractive-Summarisation-of-German-Wikipedia/dataset/sbke_miha
from rouge import Rouge
import sys
```

Slika 0-48: SBKE funkcija, prvi dio

U ovom slučaju se grafovi definiraju unutar funkcije radi jednostavnosti primjene. Dodajemo veze na temelju mjera sličnosti se stvara graf.

Potrebno je proći kroz sve vrhove grafa. Sprema se broj incidentnih vrhova trenutnog vrha te se potom zbrajaju težinske vrijednosti ulaznih i izlaznih veza tog vrha. Alpha je ovdje predstavljena kao rezultat oduzimanja sume težinskih vrijednosti od 1. Količnik sume težinskih vrijednosti bridova i broja incidentnih vrhova se potencira s brojem alpha te se dodatno množi brojem susjednih vrhova. Postupak se ponavlja za sve vrhove unutar grafa. Rezultate ovog postupka se pohranjuje u listu uređenih parova, pri čemu je prvi element rečenica, a drugi stupanj odabira.

Potom ugrađenom funkcijom nlargest() koristimo parametar top_n kako bi se definiralo koliko rečenica se želi izvući iz grafa. Prolazi se po listi uređenih parova kako bi se na kraju stvorio jedinstveni tekst.

```
test_df = df.head(50)
test_df['sbke_miha1'] = test_df.apply(lambda x: sbke_score(x['source'], 1, 0.15), axis=1)
test_df['sbke_miha2'] = test_df.apply(lambda x: sbke_score(x['source'], 2, 0.15), axis=1)
test_df['sbke_miha5'] = test_df.apply(lambda x: sbke_score(x['source'], 5, 0.15), axis=1)
test_df['sbke_miha10'] = test_df.apply(lambda x: sbke_score(x['source'], 10, 0.15), axis=1)
df.to_csv('G:/Extractive-Summarisation-of-German-Wikipedia/dataset/sbke_miha.csv', encoding='utf-8')
from rouge import Rouge
import sys

sys.stdout = open("G:/Extractive-Summarisation-of-German-Wikipedia/dataset/sbke_miha_res.txt", "w")

hy, rf = test_df['sbke_miha1'], test_df['summary']
rouge= Rouge()
scoresT1= rouge.get_scores(hy, rf, avg=True)
print("sbke_miha1 scores: ", scoresT1, '\n\n')

hy2 = test_df['sbke_miha2']
scoresT2= rouge.get_scores(hy2, rf, avg=True)
print("sbke_miha2 scores: ", scoresT2, '\n\n')

hy5 = test_df['sbke_miha5']
scoresT5= rouge.get_scores(hy5, rf, avg=True)
print("sbke_miha5 scores: ", scoresT5, '\n\n')

hy10 = test_df['sbke_miha10']
scoresT10= rouge.get_scores(hy10, rf, avg=True)
print("sbke_miha10 scores: ", scoresT10, '\n\n')

sys.stdout.close()
```

Slika 0-49: SBKE, drugio dio

Kada je definirana funkcija, radi bržeg procesiranja uzima se top 50 tekstova te se poziva funkcija koja će potom pohraniti rezultate unutar okvira podataka. Posljednji korak je analiza dobivenih sažetaka čiji se rezultati pohranjuju u posebne .txt datoteke za uvid. Sažetci se evaluiraju na temelju priloženih sažetaka unutar skupa podataka.

6 Usporedba rezultata

6.1 Grafovi algoritama sličnosti rečenica

Primijenjeni su svi algoritmi sličnosti kako bismo dobili grafove sličnosti rečenica tekstova. Svaki od algoritama drukčije primjenjuje način izračuna sličnosti. Ovaj dio rada prikazuje rezultate korištenja navedenih algoritama sličnosti koji se također koriste i za konstrukciju grafa.

6.1.1 Rezultati algoritma sličnosti kosinusa rečenica i graf

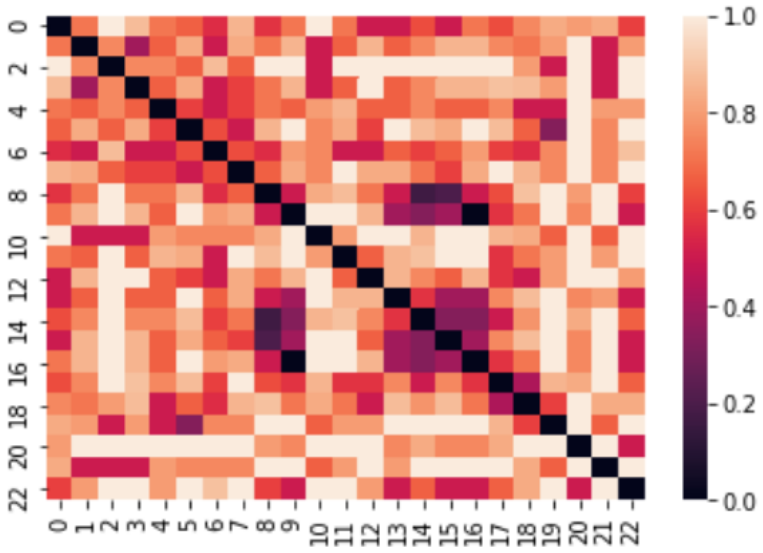
Kako bi se izračunala sličnost kosinusa rečenica potrebno je prvenstveno pretvoriti rečenice u vektore kako bismo mogli utvrditi njihovu sličnost. Koristi se biblioteka sklearn.metrics.pariwise u kojoj su definirane Jaccard sličnost rečenica te još mnoge druge.

Nakon promjene rečenice u vektor primjenjuje se sljedeći dio koda kako bismo dobili te grafički prikazali ovu sličnost među rečenicama:

```
from sklearn.metrics.pairwise import pairwise_distances
cosine = pairwise_distances(dist_df.values, metric='cosine')
sns.heatmap(pairwise_distances(dist_df.values, Y=None, metric='jaccard'));
```

Slika 0-1:Primjena sličnosti kosinusa rečenca na primjeru rečenica prvoga teksta

Graf koji smo dobili na temelju ovog izračuna je sljedeći



Slika 0-2: Grafički prikaz sličnosti rečenica

S crnom bojom je prikazana dijagonala matrice u kojima je sličnost između rečenica postavljena na 0, jer se ne vrši računanje sličnosti rečenica samih sa sobom.

Kako bismo pak izračunali sličnost kosinusa između rečenica postupak je malo drugačiji. Također se uklanjaju stop-riječi, te stvaraju skup riječi koje su sadržane u obje rečenice te u dvije zasebne liste spremamo vrijednosti 1 ili 0 u odnosu na to jesu li riječi sadržane u obje rečenice. Potom se prolazi po listi vektora, računa se umnožak elemenata dvije liste vektora u kojima je svaka rečenica predstavljena zasebnim vektorom. Za tu vrijednost uvećavamo brojač te se na kraju definira formula u kojoj se ta ukupna vrijednost brojača dijeli s umnoškom sume elemenata prve liste i sume elemenata druge liste. Dobiveni umnožak u nazivniku potom potenciramo na 0,5. Funkcija koju smo primijenili izgleda ovako:

```
def cosine_sim(sent1, sent2):
    s1 = sent1.translate(str.maketrans('', '', string.punctuation))
    s2 = sent2.translate(str.maketrans('', '', string.punctuation))
    s1 = word_tokenize(sent1, language='german')
    s2 = word_tokenize(sent2, language='german')
    sw = stopwords.words('german')
    l1, l2 = [],[]

    X_set = {w for w in s1 if not w in sw}
    Y_set = {w for w in s2 if not w in sw}

    rvector = X_set.union(Y_set)
    for w in rvector:
        if w in X_set: l1.append(1) # create a vector
        else: l1.append(0)
        if w in Y_set: l2.append(1)
        else: l2.append(0)
    c = 0

    # cosine formula
    for i in range(len(rvector)):
            c+= l1[i]*l2[i]
    return c / float((sum(l1)*sum(l2))**0.5)
```

Slika 0-3:Funkcija računanja sličnosti kosinusa rečenica teksta

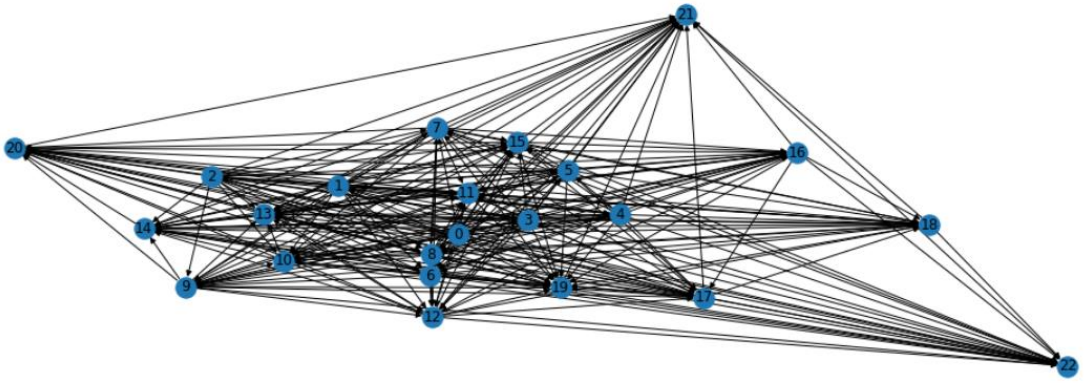
Sve što preostaje je pozvati ovaj algoritam unutar grafa za određivanje težine rubova. Algoritam za navedeno je isti kao i kada se primjenjuje Mihalcea sličnost na tekstovima. Kako bi se postiglo željeno, koristi se NetworkX python biblioteka gdje su rečenice čvorovi, a vrijednosti koje se dobiju kao rezultat funkcije cosine_sim() su rubovi. Prolazi se kroz sve vrhove grafa te se promatra sličnost između svakog od vrhova.

```
sentences = sent_tokenize(df['source'][0], language="german")
thresh = 0.1
g=nx.DiGraph()
edges = [(i,j,cosine_sim(x,y))
        for i,x in enumerate(sentences)
        for j,y in enumerate(sentences) if i < j]

g.add_weighted_edges_from((i,j,sim) for i,j,sim in edges)
plt.figure(figsize=(15,5))
nx.draw(g, with_labels=True, connectionstyle='arc, rad = 0.15')
plt.show()  # pause before exiting
```

Slika 0-4: Kreiranje grafa na temelju algoritma sličnosti kosinusa

Rezultat navedenog koda je sljedeći graf, na kojemu se kasnije primjenjuju još neki algoritmi iz kasnijih naslova:



Slika 0-5: Prikaz grafa rečenica prvog teksta na temelju mjere sličnosti kosinusa pod uvjetom zadovoljavanja praga u vrijednosti 0.15

6.1.2 Rezultati algoritma Mihalcea sličnosti rečenica i graf

Kako bi se izračunala ova mjera sličnosti potrebno definirati funkciju mihalcea() koja kao parametre prima dvije rečenice koje se potom razbijaju na skup riječi. Računa se suma logaritama duljine prve rečenice i logaritam duljine druge rečenice. Također gledamo duljinu presjeka dviju rečenica te se taj broj dijeli s prethodnom sumom logaritama duljine rečenica. U tom slučaju, funkcija mihalcea() izgleda ovako:

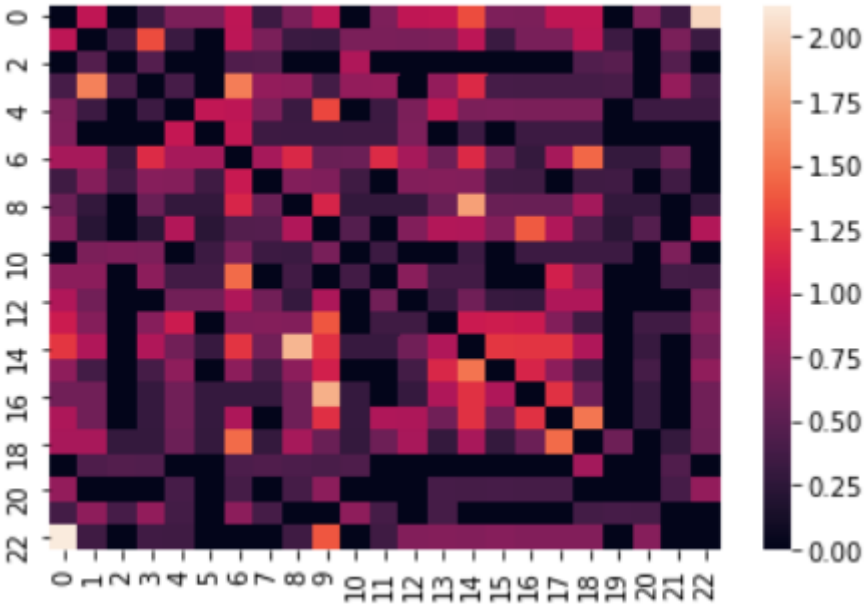
```
import math
def mihalcea(sent1, sent2):
    s1 = word_tokenize(sent1, language="german")
    s2 = word_tokenize(sent2, language="german")
    return len(set(s1).intersection(set(s2)))/(math.log(len(s1))+math.log(len(s2)))
```

Slika 0-6: Funkcija za računanje sličnosti između rečenica na temelju algoritma prezentiranog od strane (Mihalcea 2004)

Kada se primjeni Mihalcea sličnost rečenica koju računamo kao:

$$sim(s1,s2)=broj\ riječi(s1\cap s2)\log duljina(s1)+\log duljina(s2) \qquad 33).$$

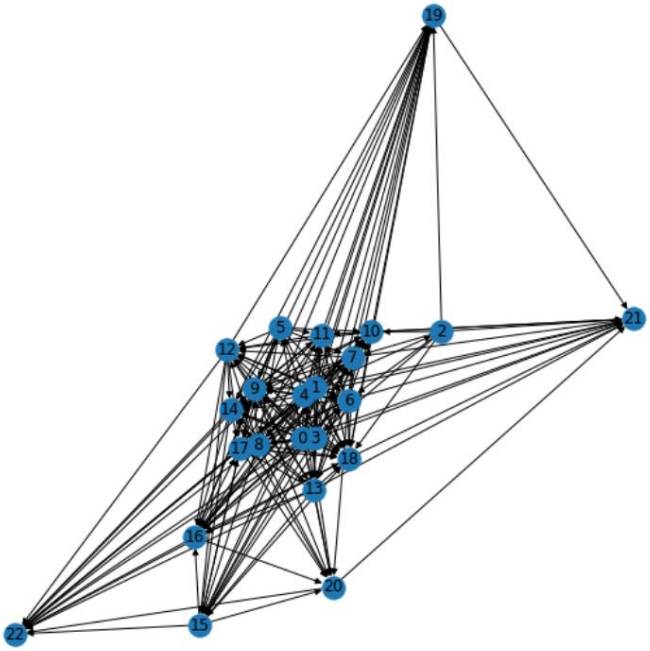
Tijekom računanja vrijednosti odnosa rečenica dobivamo sljedeći prikaz rezultata:



Slika 0-7: Prikaz rezultata Mihalcea algoritma sličnosti

Detaljniji ispis rezultata je vidljiv pod naslovom

8.10.3 Rezultati algoritma Mihalcea sličnosti. Ovdje primjećujemo kako dobar dio rečenica zapravo ima relativno malu sličnost. Jednu od najvećih imaju prva rečenica i zadnja rečenica, deveta rečenica i šesnaesta rečenica te sedma rečenica i četrnaesta rečenica. Tijekom stvaranja grafa na temelju ove sličnosti dobvamo sljedeći rezultat:



Slika 0-8:Primjer grafa prvog teksta pri čemu su čvorovi rečenice, a veze između čvorova imaju vrijednosti mihalcea sličnosti između rečenica

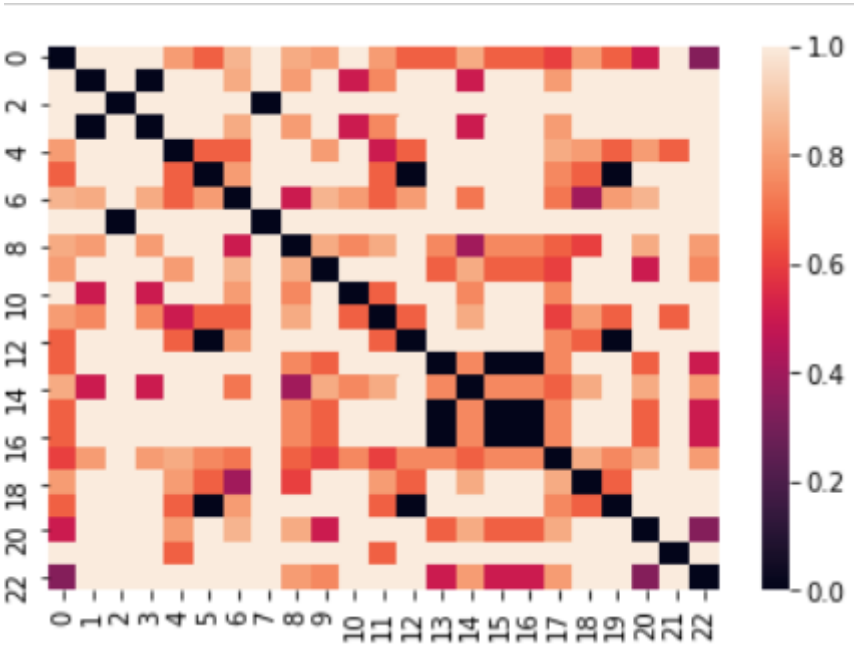
Funkcija koja se primjenjuje ovdje ista je onoj pod naslovom 6.1.1 Rezultati algoritma sličnosti kosinusa rečenica i graf samo što se ovdje poziva funkcija mihalcea() umjesto funkcije cosine_sim().

6.1.3 rezultati algoritma Jaccardove sličnosti rečenica i graf

Kako bi se izračunala Jaccardova sličnost između rečenica, potrebno je rečenice pretvoriti u skupove, tj. skupove riječi kako bi se na njima mogla izvršiti operacija presjeka skupova. Kako bi se dobio što točniji rezultat uklanjaju se svi interpunkcijski znakove te se rečenice prikazuju kao skup riječi te se potom računa Jaccardova sličnost koristeći gotovu formulu unutar nltk biblioteke:

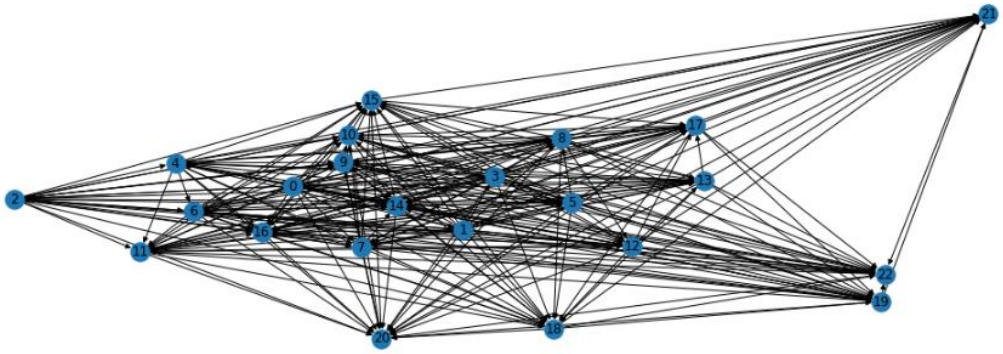
```
def jaccard(sent1, sent2):
    s1 = sent1.translate(str.maketrans('', '', string.punctuation))
    s2 = sent2.translate(str.maketrans('', '', string.punctuation))
    return nltk.jaccard_distance(set(s1), set(s2))
```

Tijekom pokretanja Jaccardove sličnosti rečenica dobiva se sljedeći odnos među rečenicama:



Slika 0-9: Prikaz odnosa na temelju Jaccardove sličnosti rečenica prvog teksta okvira podataka.

Ova slika točnije opisuje isječak rezultata vidljiv pod naslovom 8.10.2 Rezultati algoritma Jaccardove sličnosti gdje se prikazuje koliko su rečenice ovoga teksta međusobno slične. Što je boja tamnija, sličnost među rečenicama je manja. Kako bi se prikazao graf ovog teksta primjenjuje se biblioteka NetworkX.



Slika 0-10: Graf prvog teksta skupa podataka s algoritmom Jaccardove sličnosti rečenica

6.2 Usporedba rezultata algoritama sažimanja tekstova

Prikazuju se rezultati sažimanja tekstova. TextRank je pokrenut isključivo koristeći algoritam sličnosti kosinusa rečenica te je se iz priloženog okvira podataka koriste tekstovi unutar stupca source te se uspoređuju s prethodno dostupnim sažetcima unutar okvira podataka pod stupcem summary. Računa se ROUGE-1 mjera te je unutar tablica prikazane prosječne vrijednosti za svaki od navedenih tekstova. Važno je napomenuti kako su za neke od tekstova skupa podataka ponuđeni sažetci koji se sastoje od različitog broja riječi. Neki od njih imaju samo jednu rečenicu koja je predstavljena kao referencirani sažetak. Razlika u duljinama sažetaka ima određeni utjecaj na generalni prosječni rezultat sljedećih algoritama. Prilikom spominjanja top 1, 2, 3, 5 ili 10 rečenica, misli se na top *n* rečenica u odnosu na algoritme rangiranja rečenica i veza između rečenica koje predstavljaju sličnosti između dva vrha grafa. Algoritmi rangiranja rečenica su Lexrank, SBKE te PageRank algoritam unutar TextRank sažimanja rečenica.

6.2.1 LexRank rezultati

Primjenom računanja LexRank stupnja svake rečenice kao potencijalnog kandidata za rečenicu sažetka dobivaju se rezultati prezentirani unutar sljedećih tablica. Kako je navedeno potrebno je uzeti u obzir različitu duljinu referenciranih članaka, a svi rezultati u tablici prikazuju prosječan rezultat u odnosu na n količinu tekstova

top 1

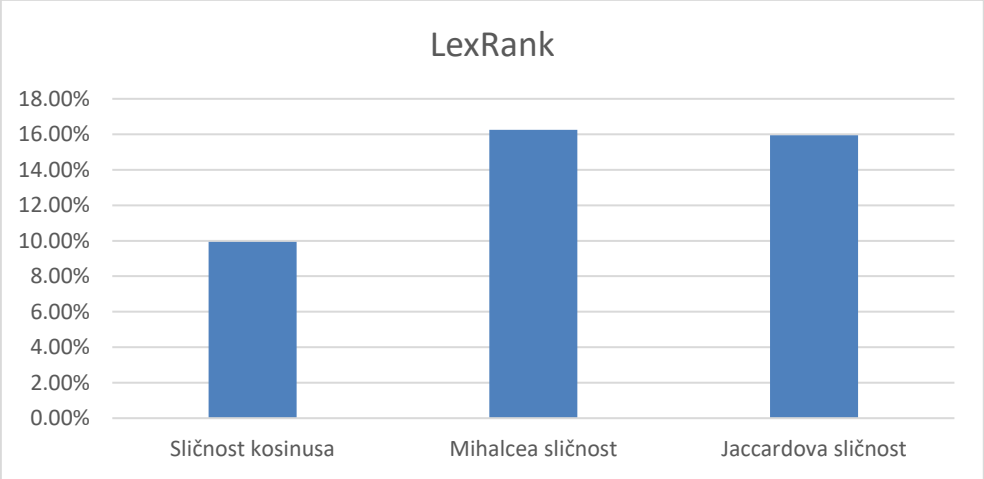
Algoritmi sličnosti	**F1-score**	**precision**	**recall**
Sličnost kosinusa	6,29%	17,02%	4,42%
Mihalcea sličnost	7,66%	15,9%	5,71%
Jaccardova sličnost	7.62%	15,84%	5,68%

top 2

Sličnost kosinusa	8,68%	14,56%	7,5%
Mihalcea sličnost	1,22%	16,15%	11,68%
Jaccardova sličnost	11,77%	16,01%	10,99%
	top 5		
Sličnost kosinusa	**9,94%**	**9,88%**	**12,45%**
Mihalcea sličnost	**16,26%**	**13,63%**	**25,09%**
Jaccardova sličnost	**15,95%**	**13,60%**	**24,16%**
	top 10		
Sličnost kosinusa	9,52%	7,33%	17,52%
Mihalcea sličnost	15,97%	10,8%	39,42%
Jaccardova sličnost	15,95%	10,94%	37,17%

Tablica 3: Rezultati LexRank sumarizacije na SwissText korpusu:mjera ROUGE-1.

Iz tablice je vidljivo kako uvjerljivo top 5 rečenica daju najbolje rezultate u slučaju LexRank algoritma na temelju izgrađenih grafova. Ovdje su prikazane prosječne vrijednosti u odnosu na broj procesiranih tekstova. Također je vidljivo kako algoritam Mihalcea sličnosti rečenica daje najbolje vrijednosti u ovoj kategoriji ali i generalno. Tek kod manjeg broja rečenica algoritam sličnosti kosinusa daje bolje rezultate ali za neznatan iznos. Također je moguće vidjeti kako je najbolji F1 rezultat ovdje onaj koji koristi Mihalcea algoritam sličnosti s prosječnim rezultatom 16,26%.



Slika 0-11: Prikaz rezultata F1 na sažetcima sastavljenih od top 5 rečenica pomoću LexRank algoritma rangiranja rečenica.

Na Slika 0-11: Prikaz rezultata F1 na sažetcima sastavljenih od top 5 rečenica pomoću LexRank algoritma rangiranja rečenica. moguće je vidjeti rezultate sažetaka za top 5 rečenica radi lakšeg i razumljivijeg prikaza rezultata.

6.2.2 SBKE rezultati

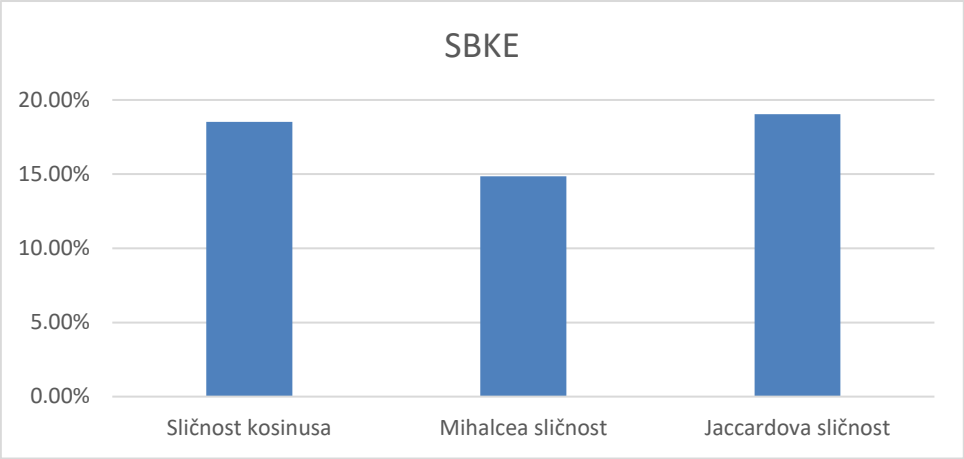
Podebljanim slovima su označeni najbolji rezultati a sa žutom bojom najbolji rezultat od svih u tablici. Također za SBKE (Beliga, Martinčić-Ipšić i Meštrović 2016) rezultati su sljedeći:

Algoritmi sličnosti	**F1-score**	**precision**	**recall**
		top 1	
Sličnost kosinusa	12,8%	25,17%	9,67%
Mihalcea sličnost	5,59%	14,84%	4,28%
Jaccardova sličnost	11.65%	21,86%	8,96%
		top 2	
Sličnost kosinusa	16,46%	21,29%	16,6%
Mihalcea sličnost	8,72%	14,44%	7,67%
Jaccardova sličnost	16,37%	20,96%	16,87%
		top 5	
Sličnost kosinusa	**18,53%**	**15,75%**	**29,99%**
Mihalcea sličnost	**14,86%**	**14,93%**	**18,99%**
Jaccardova sličnost	**19,04%**	**15,45%**	**31,55%**
		top 10	
Sličnost kosinusa	16,6%	11,39%	40,91%
Mihalcea sličnost	16,26%	11,92%	32,84%
Jaccardova sličnost	16,47%	10,98%	42,76%

Tablica 4: Rezultati primjene SBKE sumarizacije: ROUGE-1 mjera

U tablici su vidljivi rezultati za ROUGE-1 mjeru. Također je evaluacija napravljena na temelju 50 tekstova SwissText skupa podataka te su prikazani prosječni rezultati. Jaccardova sličnost ovdje daje uvjerljivo najbolje rezultate u odnosu na druge mjere sličnosti. Promotrimo li ostale rezultate možemo vidjeti kako većinom prevladava algoritam sličnosti kosinusa u top 1, 2 i 10 rečenica po stupcu F1.

Radi lakšeg prikaza rezultata, na Slika 0-12: Prikaz F1 rezultata za sažetke sastavljene od top 5 rečenica pomoću SBKE algoritam rangiranja rečenica. služi kako bi se predstavili neki od rezultata za opisani pristup rangiranja rečenica.



Slika 0-12: Prikaz F1 rezultata za sažetke sastavljene od top 5 rečenica pomoću SBKE algoritam rangiranja rečenica.

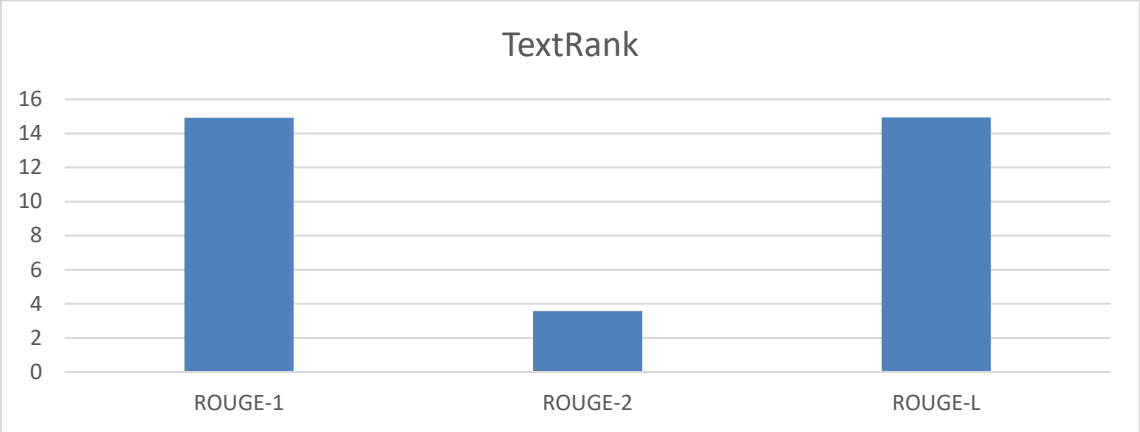
6.2.3 TextRank rezultati

Prikazuju se rezultati primjene TextRank algoritma na skupu podataka. Vrijednosti prikazane unutar tablice su prosječne vrijednosti za sažetke koji su formirani u zasebna stupce okvira podataka. Uzima se prosječna vrijednost na svih 100.000 tekstova skupa podataka. Kako bi se izračunale vrijednosti koristi se python biblioteka rouge.

	top 1		
Algoritmi sličnosti	**F1-score**	**precision**	**recall**
ROUGE-1	16,44%	16,94%	19,04%
ROUGE-2	2,84%	2,91%	3,31%
ROUGE-L	13,87%	14,35%	15,62%
	top 2		
ROUGE-1	**17,14%**	**13,63%**	**28,29%**
ROUGE-2	**3,35%**	**2,63%**	**5,71%**
ROUGE-L	**15.21%**	**12,35%**	**23,45%**
	top 5		
ROUGE-1	14,92%	9,54%	42,84%
ROUGE-2	3.57%	2,25%	1,07%
ROUGE-L	14,93%	9,87%	36,70%
	top 10		
ROUGE-1	12,27%	7,14%	5,45%
ROUGE-2	3,43%	1,99%	1,60%
ROUGE-L	13,70%	8,26%	48,18%

Tablica 5: Rezultati TextRank algoritma

Na Slika 0-13: Prikaz ROUGE rezultata za sažetke sastavljene od top 5 rečenica pomoću TextRank algoritma za sažimanje. se preglednije prikazuju rezultati sažimanja tekstova pomoću TextRank algoritma.



Slika 0-13: Prikaz ROUGE rezultata za sažetke sastavljene od top 5 rečenica pomoću TextRank algoritma za sažimanje.

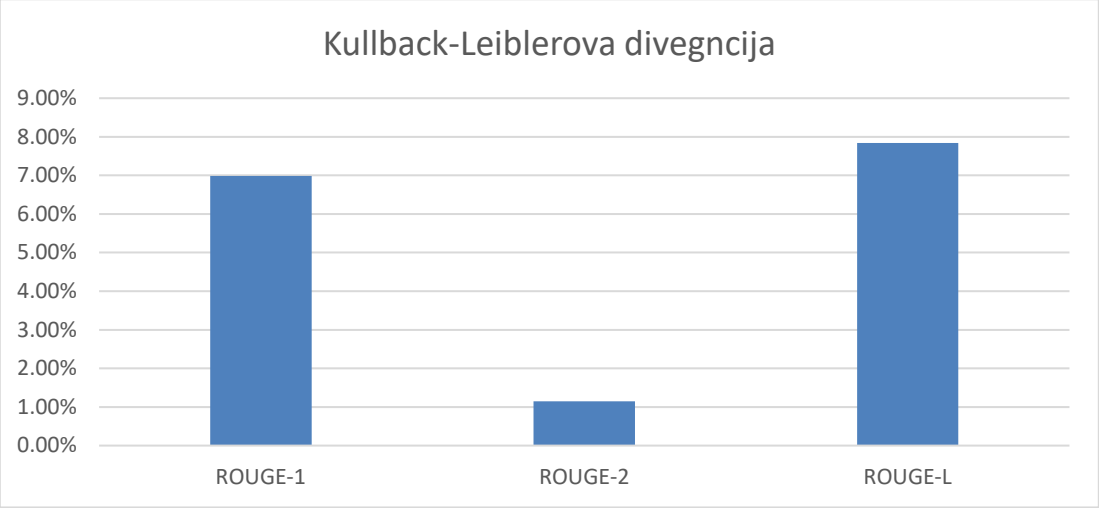
6.2.4 Rezultati algoritma sažimanja pomoću KL-divergencije

Prikazuju se rezultati primjene algoritma sažimanja tekstova pomoću KL divergencije na skupu podataka. Vrijednosti prikazane unutar tablice su prosječne vrijednosti za sažetke koji su formirani u zasebna stupce okvira podataka. Uzima se prosječna vrijednost prvih 50 tekstova skupa podataka. Kako bi se izračunale vrijednosti koristi se python biblioteka rouge.

	top 1		
Algoritmi sličnosti	**F1-score**	**precision**	**recall**
ROUGE-1	7,39%	16,30%	5,71%
ROUGE-2	1,29%	3,83%	0,93%
ROUGE-L	7,81%	16,25%	5,96%
	top 2		
ROUGE-1	**8,26%**	**10,06%**	**8,72%**
ROUGE-2	**1,46%**	**2,04%**	**1,39%**
ROUGE-L	**8,96%**	**10,76%**	**9,09%**
	top 5		
ROUGE-1	6,99%	5,69%	11,87%
ROUGE-2	1,15%	2,25%	1,98%
ROUGE-L	7,84%	6,36%	12,56%
	top 10		

ROUGE-1	6,24%	4,34%	14,64%
ROUGE-2	0,93%	0,97%	2,17%
ROUGE-L	7,12%	4,95%	15,62%

Tablica 6: Rezultati algoritma sažimanja tekstova na temelju KL divergencije.



Slika 0-14: Prikaz ROUGE rezultata za sažteke sastavljene od top 5 rečenica na temelju algoritma sažimanja pomoću Kullback-Leiblerove divergencije.

6.3 Usporedba rezultata algoritama klasifikacije

Ovdje su prikazani rezultati klasifikatora za sve tri vrste vektorizacije u odgovarajućim podnaslovima i tablicama. Algoritmi primjena navedenih klasifikatora su dostupni pod odgovarajućim naslovima.

6.3.1 Rezultati klasifikacije Naîvnog Bayesa

U priloženoj tablici Tablica 7: Rezultati algoritma Naîvnog Bayesa. prikazuju se rezultati klasifikacije algoritama vektorizacije na skupu podataka, čiji je proces opisan pod naslovom 5.4.2 Vektori rečenica. Rezultati klasifikacija su sljedeći:

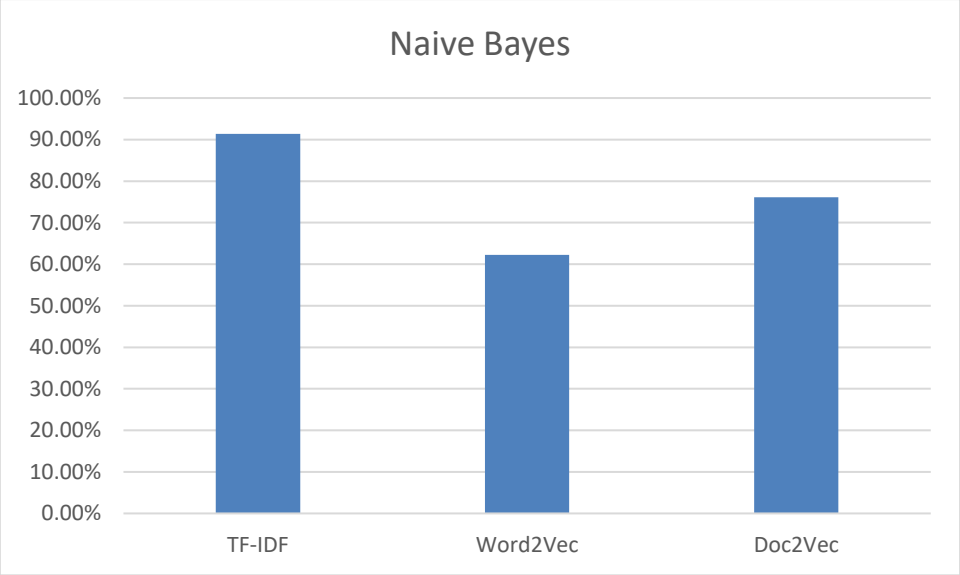
	Precision		Recall		F1		accuracy
	Yes	**No**	**Yes**	**No**	**Yes**	**No**	
TF-IDF	**0,89**	**0,93**	**0,91**	**0,92**	**0,90**	**0,93**	**91,36%**
word2vec	0,53	0,77	0,79	0,50	0,63	0,61	62,28%
Doc2vec	0,80	0,75	0,55	0,91	0,65	0,82	76,1%

Tablica 7: Rezultati algoritma Naîvnog Bayesa.

Iz priloženog je moguće vidjeti kako je najbolji rezultat ostvaren s TF-IDF vektorizacijom konstanto rezultatom od 91,36% dok je doc2vec dao drugi najbolji rezultat od 76,1%. Word2vec je generalno kod svih klasifikacija dao najlošije rezultate. Kada govorimo o

klasifikaciji TF-IDF vektora, tijekom testiranja drugih mogućnosti i označavanja rečenica na temelju drugih stupaca moguće je dobiti rezultat oko 96% međutim tada dolazi do situacije gdje je model previše prilagođen navedenom skupu podataka. U suprotnome također je moguće dobiti i lošije rezultate od oko 80% točnosti te stoga ova sredina potvrđuje najbolji mogući rezultat ove klasifikacije.

Radi lakšeg čitanja rezultata, Slika 0-15: Prikaz točnosti Naîve Bayes klasifikatora na različitim načinima vektorizacije. jasnije prikazuje rezultate iz Tablica 7: Rezultati algoritma Naîvnog Bayesa. na temelju stupca accuracy.



Slika 0-15: Prikaz točnosti Naîve Bayes klasifikatora na različitim načinima vektorizacije.

Također se primjenjuje ROUGE mjera točnosti na klasifikatoru te su rezultati sljedeći:

	F1	precision	recall
	ROUGE-1		
TF-IDF	**91,36153%**	**91,36154%**	**91,36154%**
Word2vec	62,3727%	62,3727%	62,3727%
Doc2vec	76,502345%	76,502345%	76,502344%
	ROUGE-L		
TF-IDF	**91,36153%**	**91,36154%**	**91,36154%**
Word2vec	62,3727%	62,3727%	62,3727%
Doc2vec	76,502345%	76,502345%	76,502345%

Tablica 8: ROUGE mjere klasifikatora Naîvnoh Bayesa.

Tijekom računanja mjera ROUGE-2 mjera svuda pokazuje rezultate 0.0 za sve vrijednosti klasifikacije.

6.3.2 rezultati klasifikacije SVM

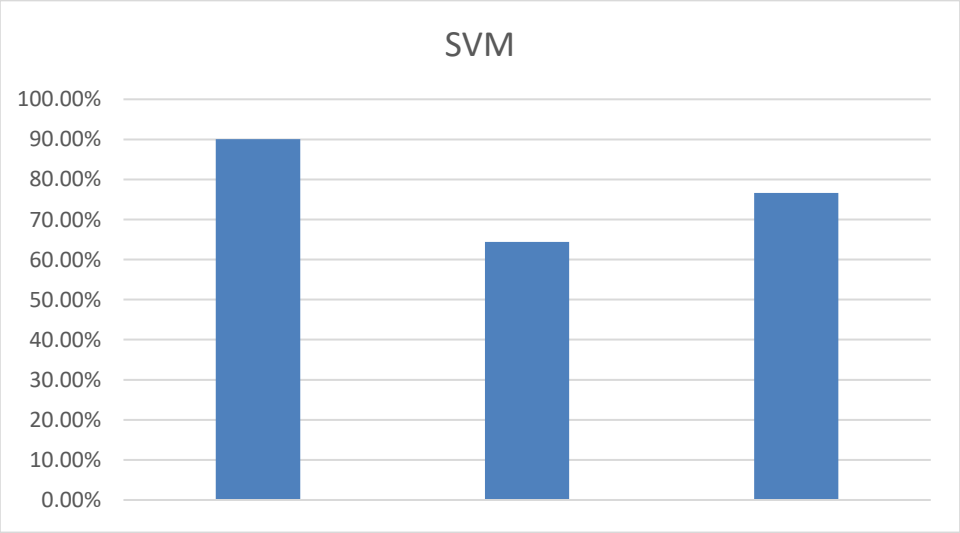
U Tablica 9: Prikaz rezultata SVM klasifikacije. su prikazani rezultati algoritma SVM klasifikacije svih vrsta vektorizacija zadanog skupa podataka.

	Precission		Recall		F1		accuracy
	Yes	**No**	**Yes**	**No**	**Yes**	**No**	
Tf-Idf	**0,88**	**0,92**	**0,89**	**0,92**	**0,88**	**0,92**	**90,04%**
word2vec	0,60	0,66	0,41	0,81	0,49	0,73	64,43%
Doc2vec	0,78	0,76	0,60	0,88	0,68	0,82	76,61%

Tablica 9: Prikaz rezultata SVM klasifikacije.

Iz priloženog je moguće vidjeti kako je i dalje konstantno TF-IDF vektorizacija daje najbolje rezultate klasifikacije. Također je moguće vidjeti koliko su rezultati između svih klasifikatora relativno blizu gledajući u odnosu na vektorizaciju, međutim vidljiva je jedina iznimka kod klasifikacije algoritma slučajnih šuma.

Na Slika 0-16: Prikaz rezultata SVM klasifikatora na različitim pristupima vektorizacije rečenica. vidljivi su rezultati na temelju stupca accuracy iz Tablica 9: Prikaz rezultata SVM klasifikacije. radi lakšeg prikaza podataka i čitljivosti rezultata.



Slika 0-16: Prikaz rezultata SVM klasifikatora na različitim pristupima vektorizacije rečenica.

Također, rezultati ROUGE mjera su sljedeći:

	F1	precision	recall
	ROUGE-1		
TF-IDF	**90,456192%**	**90,456193%**	**90,456193%**
Word2vec	64,4509%	64,4509%	64,4509%
Doc2vec	76.9471%	76.9471%	76.9471%
	ROUGE-L		
TF-IDF	**90,456192%**	**90,456193%**	**90,456193%**
Word2vec	64,4509%	64,4509%	64,4509%
Doc2vec	76.9471%	76.9471%	76.9471%

Tablica 10: Rezultati ROUGE mjera SVM klasifikatora.

6.3.3 Rezultati klasifikacije algoritma slučajnih šuma

Posljednji u nizu algoritama klasifikacije je dao rezultate vidljive u Tablica 11: Rezultati klasifikacije algoritma slučajnih šuma.

	Precission		Recall		F1		accuracy
	Yes	**No**	**Yes**	**No**	**Yes**	**No**	
Tf-Idf	1,00	1,00	1,00	1,00	1,00	1,00	100%
word2vec	0,77	0,74	0,60	0,86	0,67	0,80	74,8%
Doc2vec	**0,76**	**0,77**	**0,62**	**0,87**	**0,68**	**0,81**	**76,51%**

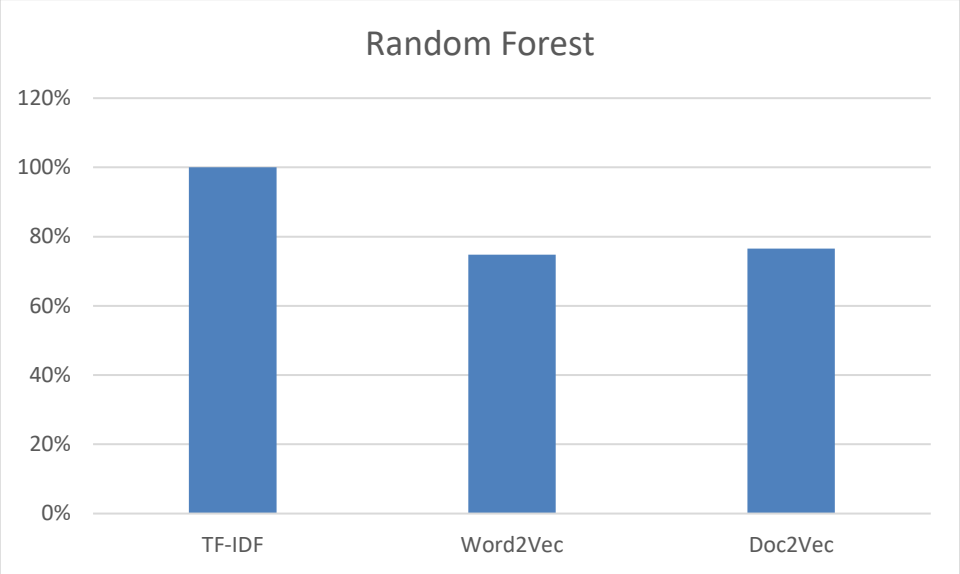
Tablica 11: Rezultati klasifikacije algoritma slučajnih šuma.

Ovdje je moguće primijetiti dvije anomalije:

1. Kod TF-IDF vektorizacije algoritam slučajnih šuma vraća 100% točnosti u svim dijelovima mjerenja. Tijekom prethodnog testiranja u kojima su rečenica označene na temelju vrijednosti drugih stupaca od onih koji su prikazani u ovome radu i unatoč smanjenju točnosti ostalih algoritama, točnost algoritma je uvijek bila 100% osim u situaciji kada je odnos između pozitivno označenih i negativno označenih rečenica (one koje neće biti odabrane u klasu sažetka) bio gotovo 50/50 pri čemu su svi algoritmi pokazivali točnosti u rasponu između 50% i 52% pa tako i algoritam slučajnih šuma. Također je isprobano smanjivanje broja razina, smanjivanje količine skupa podataka, međutim rezultat je bio uvijek isti.
2. Kod word2vec vektorizacije moguće je primijetiti veće odstupanje rezultata točnosti od ostalih algoritam klasifikacija, a korišten je jednak skup podataka, jednake duljine.

Radi navedena dva razloga i usporedbe samo točnosti word2vec i doc2vec vektorizacije, druga opcija je dala bolje rezultate.

Prilaže se i grafički prikaz rezultata algoritma klasifikacije slučajnih šuma na temelju različitih vrsta vektorizacije rečenica. Podatci prikazani unutar grafa predstavljaju stupac accuracy.



Slika 0-17: Rezultati algoritma klasifikacije slučajnih šuma na temelju različitih algoritama vektorizacije rečenica.

Također, rezultati ROUGE mjera za ovaj klasifikator su sljedeći:

	F1	precision	recall
	ROUGE-1		
TF-IDF	**91,36153%**	**91,36154%**	**91,36154%**
Word2vec	75,1743%	75,1743%	75,1743%
Doc2vec	76,8232%	76,8232%	76,8232%
	ROUGE-L		
TF-IDF	**91,36153%**	**91,36154%**	**91,36154%**
Word2vec	75,1743%	75,1743%	75,1743%
Doc2vec	76,8232%	76,8232%	76,8232%

Tablica 12: Rezultati ROUGE mjera klasifikatora slučajnih šuma.

7 Zaključak

Prije zaključivanja važno je primijetiti sljedeće vezano uz skup podataka. Evaluacija SBKE (Beliga, Martinčić-Ipšić i Meštrović 2016) i LexRank (Corley i Mihalcea 2005) rezultata je prosječna na n količinu tekstova. Skup podataka je sastavljen tako da neki od tekstova imaju samo jednu rečenicu kao sažetak te su moguća odstupanja pri stvaranju sažetaka kod takvih slučajeva veća te umanjuju prosječan postotak točnosti algoritama.

Promatrajući TextRank algoritam moguće je vidjeti relativno opadanje rezultata u odnosu na referencirane sažetke, pri tom uzimajući u obzir varijaciju duljina priloženih sažetaka unutar skupa podataka. Tijekom promatranja možemo vidjeti kako prezentirani sažetci ovog algoritam, a u sebi sadrže jednu ili dvije najbitnije rečenice, pokazuju veću točnost u odnosu na sažetke s 5 ili 10 rečenica koji imaju manji postotak točnosti. Njegov najbolji rezultat je 17,14% po F1 mjeri Rouge-1.

LexRank (Mihalcea, Corley i Strapparava 2006) algoritam rangiranja rečenica ovdje pokazuje kako je najbolji rezultat navedenog algoritma 16.26% koji se dobije na temelju Mihalcea sličnosti rečenica. Najtočnijima su se pokazali rezultati za top 5 rečenica pri čemu u svim ostalim kategorijama prevladava algoritam sličnosti kosinusa.

SBKE (Beliga, Martinčić-Ipšić i Meštrović 2016) pokazao je najbolje rezultate u odnosu na sve navedene pristupe sažimanja tekstova. Najbolji F1 rezultat iznosi 19,04% i to s primjenom Jaccardove sličnosti rečenica. Također prevladavaju rezultati za top 5 rečenica.

Algoritam sažimanja na temelju KL divergencije nije pokazao najbolje rezultate te je najlošiji od navedenih s najboljim rezultatom oko 8,96% i to u kategoriji top 2 rečenice.

Algoritmi klasifikacije su pokazali dosta zanimljive rezultate, međutim svakako postoji još mjesta za napredak. Jedna od mogućnosti kako dobiti bolje doc2vec rezultate jest učenje i Bag-of-Word doc2vec modela i doc2vec modela distribuirane memorije te na kraju spojiti ta dva modela u jedan i izgraditi vektore. Kao što je očekivano, izuzev algoritma slučajnih šuma, TF-IDF je pokazao najbolje rezultate.

8 Privitak

Radi jednostavnosti prikaza i primjera, koristi se isti tekst kroz sve podnaslove unutar privitka radi lakšeg praćenja i konzistentnosti podataka. Podnaslove klasifikacije pak obuhvaćaju cjelokupni skup podataka i rezultate vezane uz taj skup podataka.

8.1 Popis slika

8.2 Popis tablica

8.3 Primjer SwissText korpusa

Ovdje je moguće vidjeti nekoliko primjera tekstova koji se nalaze unutar swissText korpusa.

Prikazan je tekst prethodno obradi i procesiranju koji je moguće pronaći unutar stupca source:

```
Minghella war der Sohn italienisch-schottischer Eltern, die auf der Isl
e of Wight eine Fabrik für Eiscreme betrieben. Nach seinem Schulabschlu
ss studierte er an der Universität Hull, wo er eine Zeit lang als Dozen
t tätig war. 1978 drehte er einen ersten Kurzfilm. Seit 1981 war er als
Autor und Story Editor tätig. Er wurde mit Theaterstücken, Rundfunkhörs
pielen, der Fernsehserie "Inspector Morse" und vielen Drehbüchern für F
ilm und Fernsehen bekannt. Er entwickelte die Drehbücher für die 1988 e
rfolgreich ausgestrahlte Fernsehserie The Storyteller von Muppets-Erfin
der Jim Henson. Auch als Produzent war er erfolgreich, darunter für die
Filme "Der stille Amerikaner", "Die Dolmetscherin" und "Der Vorleser",
für den er 2008 posthum für den Oscar nominiert wurde. Gemeinsam mit se
inem Freund und Kollegen Sydney Pollack gründete er die Produktionsfirm
a Mirage Enterprises. Der Regisseur Minghella galt als ein guter Schaus
pielerführer: Unter seiner Regie brachten es zahlreiche Darsteller zu O
scar-Nominierungen, zwei Schauspielerinnen erhielten die Auszeichnung a
ls "Beste Nebendarstellerin": Juliette Binoche und Renée Zellweger . Ge
gen Ende seines Lebens kehrte Minghella zu seinen Anfängen im Radio und
auf der Bühne zurück: 2006 wurde sein Hörspiel "Eyes Down Looking" mit
Jude Law zu Ehren von Samuel Beckett auf BBC Radio 3 ausgestrahlt, ein
Jahr zuvor hatte seine Inszenierung der Puccini-Oper Madame Butterfly i
n der English National Opera in London Premiere und wurde auch in der N
ationaloper von Vilnius und in der Metropolitan Opera in New York gezei
gt. Am Ende des Films "Abbitte" von Joe Wright hat er einen Kurzauftrit
t als Talkshow-Moderator neben Vanessa Redgrave. Seine letzte Arbeit al
s Drehbuchautor war das Skript für den Musical-Film "Nine" . Zu seinen
letzten Regiearbeiten zählt der Pilotfilm zur Krimiserie "Eine Detektiv
in für Botswana" , den die BBC fünf Tage nach seinem Tod erstmals ausst
rahlte. Minghella war mit der aus Hongkong stammenden Choreographin, Pr
oduzentin und Schauspielerin Carolyn Choa verheiratet. Der Ehe entstamm
en zwei Kinder, die in der Filmbranche tätig sind: Tochter Hannah Mingh
ella in der Produktion und Sohn Max Minghella als Schauspieler . Die Ta
nte Edana Minghella und der Onkel Dominic Minghella sind Drehbuchautore
n. Minghella starb im Alter von 54 Jahren in einem Londoner Krankenhaus
an inneren Blutungen infolge der Operation eines Tonsillenkarzinoms und
eines Karzinoms im Nacken. 1984 erhielt Minghella den Londoner Kritiker
preis als meistversprechender junger Dramatiker, 1986 den Kritikerpreis
für sein Stück "Made in Bangkok" als bestes Stück der Saison. 1997 erhi
elt er für "Der englische Patient" den Oscar in der Rubrik "Beste Regie
", 1999 eine Oscar-Nominierung in der Kategorie "Bestes adaptiertes Dre
hbuch" für "Der talentierte Mr. Ripley", bei dem er auch Regie führte.
2001 wurde Minghella zum Commander of the British Empire ernannt. Von 2
003 bis 2007 war er Präsident des British Film Institute. Seit 1997 trä
gt das Anthony Minghella Theatre auf der Isle of Wight seinen Namen.
```

Njegov sažetak, koji je dostupan unutar stupca summary, pri tome izgleda ovako:

Anthony Minghella, CBE war ein britischer Filmregisseur, Filmproduzent, Drehbuchautor, Dramatiker, Hörspiel-Autor, Theater- und Opern-Regisseur .

8.4 Primijer lematizacije SwissText korpusa

Na prethodni tekst, primijenjena su 3 različita algoritma za lematizaciju. U sljedećim naslovima se predstavljaju rezultati svakoga.

8.4.1 Wordnet lematizator

Rezultati WordNet lematizatora:

Minghella war der Sohn italienisch-schottischer Eltern , die auf der Is le of Wight eine Fabrik für Eiscreme betrieben . Nach seinem Schulabsch luss studierte er an der Universität Hull , wo er eine Zeit lang al Doz ent tätig war . 1978 drehte er einen ersten Kurzfilm . Seit 1981 war er al Autor und Story Editor tätig . Er wurde mit Theaterstücken , Rundfun khörspielen , der Fernsehserie `` Inspector Morse '' und vielen Drehbüc hern für Film und Fernsehen bekannt . Er entwickelte die Drehbücher für die 1988 erfolgreich ausgestrahlte Fernsehserie The Storyteller von Mup pets-Erfinder Jim Henson . Auch al Produzent war er erfolgreich , darun ter für die Filme `` Der stille Amerikaner '' , `` Die Dolmetscherin '' und `` Der Vorleser '' , für den er 2008 posthum für den Oscar nominier t wurde . Gemeinsam mit seinem Freund und Kollegen Sydney Pollack gründ ete er die Produktionsfirma Mirage Enterprises . Der Regisseur Minghell a galt al ein guter Schauspielerführer : Unter seiner Regie brachten e zahlreiche Darsteller zu Oscar-Nominierungen , zwei Schauspielerinnen e rhielten die Auszeichnung al `` Beste Nebendarstellerin '' : Juliette B inoche und Renée Zellweger . Gegen Ende seine Lebens kehrte Minghella z u seinen Anfängen im Radio und auf der Bühne zurück : 2006 wurde sein H örspiel `` Eyes Down Looking '' mit Jude Law zu Ehren von Samuel Becket t auf BBC Radio 3 ausgestrahlt , ein Jahr zuvor hatte seine Inszenierun g der Puccini-Oper Madame Butterfly in der English National Opera in Lo ndon Premiere und wurde auch in der Nationaloper von Vilnius und in der Metropolitan Opera in New York gezeigt . Am Ende de Films `` Abbitte '' von Joe Wright hat er einen Kurzauftritt al Talkshow-Moderator neben Va nessa Redgrave . Seine letzte Arbeit al Drehbuchautor war da Skript für den Musical-Film `` Nine '' . Zu seinen letzten Regiearbeiten zählt der Pilotfilm zur Krimiserie `` Eine Detektivin für Botswana '' , den die B BC fünf Tage nach seinem Tod erstmals ausstrahlte . Minghella war mit d er au Hongkong stammenden Choreographin , Produzentin und Schauspieleri n Carolyn Choa verheiratet . Der Ehe entstammen zwei Kinder , die in de r Filmbranche tätig sind : Tochter Hannah Minghella in der Produktion u nd Sohn Max Minghella al Schauspieler . Die Tante Edana Minghella und d er Onkel Dominic Minghella sind Drehbuchautoren . Minghella starb im Al ter von 54 Jahren in einem Londoner Krankenhaus an inneren Blutungen in folge der Operation eines Tonsillenkarzinoms und eines Karzinoms im Nac ken . 1984 erhielt Minghella den Londoner Kritikerpreis al meistversprе chender junger Dramatiker , 1986 den Kritikerpreis für sein Stück `` Ma de in Bangkok '' al bestes Stück der Saison . 1997 erhielt er für `` De r englische Patient '' den Oscar in der Rubrik `` Beste Regie '' , 1999 eine Oscar-Nominierung in der Kategorie `` Bestes adaptiertes Drehbuch '' für `` Der talentierte Mr. Ripley '' , bei dem er auch Regie führte . 2001 wurde Minghella zum Commander of the British Empire ernannt . Vo n 2003 bi 2007 war er Präsident de British Film Institute . Seit 1997 t rägt da Anthony Minghella Theatre auf der Isle of Wight seinen Namen .

8.4.2 TreeTagger lematizator

Rezultati TreeTagger lematizatora su sljedeći:

```
[('Minghella', 'Minghella', 'NE'), ('war', 'sein', 'VAFIN'), ('der', 'd
er', 'ART'), ('Sohn', 'Sohn', 'NN'), ('italienisch-schottischer', 'ital
ienisch-schottisch', 'ADJA'), ('Eltern', 'Eltern', 'NN'), (',', '--', '
$,'), ('die', 'die', 'PRELS'), ('auf', 'auf', 'APPR'), ('der', 'der', '
ART'), ('Isle', 'Isle', 'NE'), ('of', 'of', 'FM'), ('Wight', 'wight', '
FM'), ('eine', 'eine', 'ART'), ('Fabrik', 'Fabrik', 'NN'), ('für', 'Für
', 'NE'), ('Eiscreme', 'Eiscreme', 'NN'), ('betrieben', 'betreiben', 'V
VPP'), ('.', '--', '$.'), ('Nach', 'nach', 'APPR'), ('seinem', 'sein',
'PPOSAT'), ('Schulabschluss', 'Schulabschlus', 'NN'), ('studierte', 'st
udieren', 'VVFIN'), ('er', 'er', 'PPER'), ('an', 'an', 'APPR'), ('der',
'der', 'ART'), ('Universität', 'universität', 'FM'), ('Hull', 'hull', '
FM'), (',', '--', '$,'), ('wo', 'wo', 'PWAV'), ('er', 'er', 'PPER'), ('
eine', 'eine', 'ART'), ('Zeit', 'Zeit', 'NN'), ('lang', 'lang', 'ADJD')
, ('als', 'als', 'APPR'), ('Dozent', 'Dozent', 'NN'), ('tätig', 'tätig'
, 'ADJD'), ('war', 'sein', 'VAFIN'), ('.', '--', '$.'), ('1978', '1978'
, 'CARD'), ('drehte', 'drehen', 'VVFIN'), ('er', 'er', 'PPER'), ('einen
', 'einen', 'ART'), ('ersten', 'erster', 'ADJA'), ('Kurzfilm', 'Kurzfil
m', 'NN'), ('.', '--', '$.'), ('Seit', 'seit', 'APPR'), ('1981', '1981'
, 'CARD'), ('war', 'sein', 'VAFIN'), ('er', 'er', 'PPER'), ('als', 'als
', 'APPR'), ('Autor', 'Autor', 'NN'), ('und', 'und', 'KON'), ('Story',
'Story', 'NN'), ('Editor', 'Editor', 'NE'), ('tätig', 'tätig', 'ADJD'),
('.', '--', '$.'), ('Er', 'er', 'PPER'), ('wurde', 'werden', 'VAFIN'),
('mit', 'mit', 'APPR'), ('Theaterstücken', 'Theaterstücken', 'NN'), (',
', '--', '$,'), ('Rundfunkhörspielen', 'Rundfunkhörspielen', 'NN'), (',
', '--', '$,'), ('der', 'der', 'ART'), ('Fernsehserie', 'Fernsehserie',
'NN'), ('``', '``', '$('), ('Inspector', 'Inspector', 'NE'), ('Morse',
'morsen', 'VVIMP'), ("''", "''", '$('), ('und', 'und', 'KON'), ('vielen
', 'vielen', 'PIAT'), ('Drehbüchern', 'Drehbüchern', 'NN'), ('für', 'Fü
r', 'NE'), ('Film', 'Film', 'NN'), ('und', 'und', 'KON'), ('Fernsehen',
'Fernsehen', 'NN'), ('bekannt', 'bekannt', 'PTKVZ'), ('.', '--', '$.'),
('Er', 'er', 'PPER'), ('entwickelte', 'entwickeln', 'VVFIN'), ('die', '
die', 'ART'), ('Drehbücher', 'Drehbücher', 'NN'), ('für', 'Für', 'NE'),
('die', 'die', 'ART'), ('1988', '1988', 'CARD'), ('erfolgreich', 'erfol
greich', 'ADJD'), ('ausgestrahlte', 'ausgestrahlt', 'ADJA'), ('Fernsehs
erie', 'Fernsehserie', 'NN'), ('The', 'The', 'NE'), ('Storyteller', 'St
oryteller', 'NE'), ('von', 'von', 'APPR'), ('Muppets-Erfinder', 'Muppet
s-erfinder', 'NN'), ('Jim', 'Jim', 'NE'), ('Henson', 'Henson', 'NE'), (
'.', '--', '$.'), ('Auch', 'auch', 'ADV'), ('als', 'als', 'APPR'), ('Pr
oduzent', 'Produzent', 'NN'), ('war', 'sein', 'VAFIN'), ('er', 'er', 'P
PER'), ('erfolgreich', 'erfolgreich', 'ADJD'), (',', '--', '$,'), ('dar
unter', 'darunter', 'PROAV'), ('für', 'Für', 'NE'), ('die', 'die', 'ART
'), ('Filme', 'Film', 'NN'), ('``', '``', '$('), ('Der', 'der', 'ART'),
('stille', 'still', 'ADJA'), ('Amerikaner', 'Amerikaner', 'NN'), ("''",
"''", '$('), (',', '--', '$,'), ('``', '``', '$('), ('Die', 'die', 'ART
'), ('Dolmetscherin', 'Dolmetscherin', 'NN'), ("''", "''", '$('), ('und
', 'und', 'KON'), ('``', '``', '$('), ('Der', 'der', 'ART'), ('Vorleser
', 'Vorleser', 'NN'), ("''", "''", '$('), (',', '--', '$,'), ('für', 'F
```

ür', 'NE'), ('den', 'den', 'ART'), ('er', 'er', 'PPER'), ('2008', '2008
', 'CARD'), ('posthum', 'posthum', 'ADJD'), ('für', 'Für', 'NE'), ('den
', 'den', 'ART'), ('Oscar', 'Oscar', 'NN'), ('nominiert', 'nominieren',
'VVPP'), ('wurde', 'werden', 'VAFIN'), ('.', '--', '$.'), ('Gemeinsam',
'gemeinsam', 'ADJD'), ('mit', 'mit', 'APPR'), ('seinem', 'sein', 'PPOSA
T'), ('Freund', 'Freund', 'NN'), ('und', 'und', 'KON'), ('Kollegen', 'K
ollege', 'NN'), ('Sydney', 'Sydney', 'NE'), ('Pollack', 'Pollack', 'NE'
), ('gründete', 'gründen', 'VVFIN'), ('er', 'er', 'PPER'), ('die', 'die
', 'ART'), ('Produktionsfirma', 'Produktionsfirma', 'NN'), ('Mirage', '
Mirage', 'NE'), ('Enterprises', 'Enterprise', 'NE'), ('.', '--', '$.'),
('Der', 'der', 'ART'), ('Regisseur', 'Regisseur', 'NN'), ('Minghella',
'Minghella', 'NE'), ('galt', 'gelten', 'VVFIN'), ('als', 'als', 'APPR')
, ('ein', 'ein', 'ART'), ('guter', 'gut', 'ADJA'), ('Schauspielerführer
', 'Schauspielerführer', 'NN'), (':', '--', '$('), ('Unter', 'unter', '
APPR'), ('seiner', 'sein', 'PPOSAT'), ('Regie', 'Regie', 'NN'), ('brach
ten', 'bringen', 'VVFIN'), ('es', 'es', 'PPER'), ('zahlreiche', 'zahlre
ich', 'ADJA'), ('Darsteller', 'Darsteller', 'NN'), ('zu', 'zu', 'APPR')
, ('Oscar-Nominierungen', 'Oscar-nominierung', 'NN'), (',', '--', '$,')
, ('zwei', 'zwei', 'CARD'), ('Schauspielerinnen', 'Schauspielerin', 'NN
'), ('erhielten', 'erhalten', 'VVFIN'), ('die', 'die', 'ART'), ('Auszei
chnung', 'Auszeichnung', 'NN'), ('als', 'als', 'APPR'), ('``', '``', '$
('), ('Beste', 'gut', 'ADJA'), ('Nebendarstellerin', 'Nebendarstellerin
', 'NN'), ("''", "''", '$('), (':', '--', '$.'), ('Juliette', 'Juliette
', 'NE'), ('Binoche', 'Binoche', 'NE'), ('und', 'und', 'KON'), ('Renée'
, 'Renée', 'NE'), ('Zellweger', 'Zellweger', 'NE'), ('.', '--', '$.'),
('Gegen', 'gegen', 'APPR'), ('Ende', 'Ende', 'NN'), ('seines', 'sein',
'PPOSAT'), ('Lebens', 'Leben', 'NN'), ('kehrte', 'kehren', 'VVFIN'), ('
Minghella', 'Minghella', 'NE'), ('zu', 'zu', 'APPR'), ('seinen', 'sein'
, 'PPOSAT'), ('Anfängen', 'Anfängen', 'NN'), ('im', 'im', 'APPRART'), (
'Radio', 'Radio', 'NN'), ('und', 'und', 'KON'), ('auf', 'auf', 'APPR'),
('der', 'der', 'ART'), ('Bühne', 'bühne', 'FM'), ('zurück', 'zurück', '
FM'), (':', '--', '$('), ('2006', '2006', 'CARD'), ('wurde', 'werden',
'VAFIN'), ('sein', 'sein', 'PPOSAT'), ('Hörspiel', 'Hörspiel', 'NN'), (
'``', '``', '$('), ('Eyes', 'Eyes', 'NE'), ('Down', 'Dow', 'NE'), ('Loo
king', 'Looking', 'NE'), ("''", "''", '$('), ('mit', 'mit', 'APPR'), ('
Jude', 'Jude', 'NE'), ('Law', 'law', 'FM'), ('zu', 'zu', 'APPR'), ('Ehr
en', 'Ehre', 'NN'), ('von', 'von', 'APPR'), ('Samuel', 'Samuel', 'NE'),
('Beckett', 'Beckett', 'NE'), ('auf', 'auf', 'APPR'), ('BBC', 'Bbc', 'N
E'), ('Radio', 'Radio', 'NE'), ('3', '3', 'CARD'), ('ausgestrahlt', 'au
sstrahlen', 'VVPP'), (',', '--', '$,'), ('ein', 'ein', 'ART'), ('Jahr',
'Jahr', 'NN'), ('zuvor', 'zuvor', 'ADV'), ('hatte', 'haben', 'VAFIN'),
('seine', 'sein', 'PPOSAT'), ('Inszenierung', 'Inszenierung', 'NN'), ('
der', 'der', 'ART'), ('Puccini-Oper', 'puccini-oper', 'FM'), ('Madame',
'madame', 'FM'), ('Butterfly', 'butterfly', 'FM'), ('in', 'in', 'APPR')
, ('der', 'der', 'ART'), ('English', 'english', 'FM'), ('National', 'na
tional', 'FM'), ('Opera', 'Opera', 'NN'), ('in', 'in', 'APPR'), ('Londo
n', 'London', 'NE'), ('Premiere', 'Premiere', 'NE'), ('und', 'und', 'KO
N'), ('wurde', 'werden', 'VAFIN'), ('auch', 'auch', 'ADV'), ('in', 'in'
, 'APPR'), ('der', 'der', 'ART'), ('Nationaloper', 'Nationaloper', 'NN'
), ('von', 'von', 'APPR'), ('Vilnius', 'Vilnius', 'NE'), ('und', 'und',
'KON'), ('in', 'in', 'APPR'), ('der', 'der', 'ART'), ('Metropolitan', '

Metropolitan', 'NN'), ('Opera', 'Opera', 'NN'), ('in', 'in', 'APPR'), ('New', 'New', 'NE'), ('York', 'York', 'NE'), ('gezeigt', 'zeigen', 'VVP P'), ('.', '--', '$.'), ('Am', 'am', 'APPRART'), ('Ende', 'Ende', 'NN') , ('des', 'des', 'ART'), ('Films', 'Film', 'NN'), ('``', '``', '$('), ('Abbitte', 'Abbitte', 'NN'), ("''", "''", '$('), ('von', 'von', 'APPR') , ('Joe', 'Joe', 'NE'), ('Wright', 'Wright', 'NE'), ('hat', 'haben', 'V AFIN'), ('er', 'er', 'PPER'), ('einen', 'einen', 'ART'), ('Kurzauftritt ', 'Kurzauftritt', 'NN'), ('als', 'als', 'APPR'), ('Talkshow-Moderator' , 'Talkshow-moderator', 'NN'), ('neben', 'neben', 'APPR'), ('Vanessa', 'Vanessa', 'NE'), ('Redgrave', 'Redgrave', 'NE'), ('.', '--', '$.'), (' Seine', 'sein', 'PPOSAT'), ('letzte', 'letzter', 'ADJA'), ('Arbeit', 'A rbeit', 'NN'), ('als', 'als', 'APPR'), ('Drehbuchautor', 'Drehbuchautor ', 'NN'), ('war', 'sein', 'VAFIN'), ('das', 'das', 'ART'), ('Skript', ' Skript', 'NN'), ('für', 'Für', 'NE'), ('den', 'den', 'ART'), ('Musical-Film', 'Musical-film', 'NN'), ('``', '``', '$('), ('Nine', 'Nin', 'NE') , ("''", "''", '$('), ('.', '--', '$.'), ('Zu', 'zu', 'APPR'), ('seinen ', 'sein', 'PPOSAT'), ('letzten', 'letzter', 'ADJA'), ('Regiearbeiten', 'Regiearbeit', 'NN'), ('zählt', 'zählen', 'VVFIN'), ('der', 'der', 'ART '), ('Pilotfilm', 'Pilotfilm', 'NN'), ('zur', 'zur', 'APPRART'), ('Krim iserie', 'Krimiserie', 'NN'), ('``', '``', '$('), ('Eine', 'eine', 'ART '), ('Detektivin', 'Detektivin', 'NN'), ('für', 'Für', 'NE'), ('Botswan a', 'Botswana', 'NE'), ("''", "''", '$('), (',', '--', '$,'), ('den', ' den', 'PRELS'), ('die', 'die', 'ART'), ('BBC', 'Bbc', 'NE'), ('fünf', ' Fünf', 'NE'), ('Tage', 'Tag', 'NN'), ('nach', 'nach', 'APPR'), ('seinem ', 'sein', 'PPOSAT'), ('Tod', 'Tod', 'NN'), ('erstmals', 'erstmals', 'A DV'), ('ausstrahlte', 'ausstrahlen', 'VVFIN'), ('.', '--', '$.'), ('Min ghella', 'Minghella', 'NE'), ('war', 'sein', 'VAFIN'), ('mit', 'mit', ' APPR'), ('der', 'der', 'ART'), ('aus', 'aus', 'APPR'), ('Hongkong', 'Ho ngkong', 'NE'), ('stammenden', 'stammend', 'ADJA'), ('Choreographin', ' Choreographin', 'NN'), (',', '--', '$,'), ('Produzentin', 'Produzent', 'NN'), ('und', 'und', 'KON'), ('Schauspielerin', 'Schauspielerin', 'NN'), ('Carolyn', 'Carolyn', 'NE'), ('Choa', 'Choa', 'NE'), ('verheiratet' , 'verheiraten', 'VVPP'), ('.', '--', '$.'), ('Der', 'der', 'ART'), ('E he', 'Ehe', 'NN'), ('entstammen', 'entstammen', 'VVFIN'), ('zwei', 'zwe i', 'CARD'), ('Kinder', 'Kind', 'NN'), (',', '--', '$,'), ('die', 'die' , 'PRELS'), ('in', 'in', 'APPR'), ('der', 'der', 'ART'), ('Filmbranche' , 'Filmbranche', 'NN'), ('tätig', 'tätig', 'ADJD'), ('sind', 'sein', 'V AFIN'), (':', '--', '$.'), ('Tochter', 'Tochter', 'NN'), ('Hannah', 'Ha nnah', 'NE'), ('Minghella', 'Minghella', 'NE'), ('in', 'in', 'APPR'), ('der', 'der', 'ART'), ('Produktion', 'Produktion', 'NN'), ('und', 'und' , 'KON'), ('Sohn', 'Sohn', 'NN'), ('Max', 'Max', 'NE'), ('Minghella', ' Minghella', 'NE'), ('als', 'als', 'APPR'), ('Schauspieler', 'Schauspiel er', 'NN'), ('.', '--', '$.'), ('Die', 'die', 'ART'), ('Tante', 'Tante' , 'NN'), ('Edana', 'Edana', 'NE'), ('Minghella', 'Minghella', 'NE'), (' und', 'und', 'KON'), ('der', 'der', 'ART'), ('Onkel', 'Onkel', 'NN'), ('Dominic', 'Dominic', 'NE'), ('Minghella', 'Minghella', 'NE'), ('sind', 'sein', 'VAFIN'), ('Drehbuchautoren', 'Drehbuchautor', 'NN'), ('.', '-- ', '$.'), ('Minghella', 'Minghella', 'NE'), ('starb', 'sterben', 'VVFIN '), ('im', 'im', 'APPRART'), ('Alter', 'Alter', 'NN'), ('von', 'von', ' APPR'), ('54', '54', 'CARD'), ('Jahren', 'Jahr', 'NN'), ('in', 'in', 'A PPR'), ('einem', 'einem', 'ART'), ('Londoner', 'london', 'ADJA'), ('Kra

nkenhaus', 'Krankenhaus', 'NN'), ('an', 'an', 'APPR'), ('inneren', 'inn
erer', 'ADJA'), ('Blutungen', 'Blutung', 'NN'), ('infolge', 'infolge',
'APPR'), ('der', 'der', 'ART'), ('Operation', 'Operation', 'NN'), ('ein
es', 'eines', 'ART'), ('Tonsillenkarzinoms', 'Tonsillenkarzinom', 'NN')
, ('und', 'und', 'KON'), ('eines', 'eines', 'ART'), ('Karzinoms', 'Karz
inoms', 'NE'), ('im', 'im', 'APPRART'), ('Nacken', 'Nacken', 'NN'), ('.
', '--', '$.'), ('1984', '1984', 'CARD'), ('erhielt', 'erhalten', 'VVFI
N'), ('Minghella', 'Minghella', 'NE'), ('den', 'den', 'ART'), ('Londone
r', 'london', 'ADJA'), ('Kritikerpreis', 'Kritikerpreis', 'NN'), ('als'
, 'als', 'APPR'), ('meistversprechender', 'meistversprechend', 'ADJA'),
('junger', 'jung', 'ADJA'), ('Dramatiker', 'Dramatiker', 'NN'), (',', '
--', '$,'), ('1986', '1986', 'CARD'), ('den', 'den', 'ART'), ('Kritiker
preis', 'Kritikerpreis', 'NN'), ('für', 'Für', 'NE'), ('sein', 'sein',
'PPOSAT'), ('Stück', 'Stück', 'NN'), ('``', '``', '$('), ('Made', 'made
', 'FM'), ('in', 'in', 'APPR'), ('Bangkok', 'Bangkok', 'NE'), ("''", "'
'", '$('), ('als', 'als', 'APPR'), ('bestes', 'gut', 'ADJA'), ('Stück',
'Stück', 'NN'), ('der', 'der', 'ART'), ('Saison', 'Saison', 'NN'), ('.'
, '--', '$.'), ('1997', '1997', 'CARD'), ('erhielt', 'erhalten', 'VVFIN
'), ('er', 'er', 'PPER'), ('für', 'Für', 'NE'), ('``', '``', '$('), ('D
er', 'der', 'ART'), ('englische', 'englisch', 'ADJA'), ('Patient', 'Pat
ient', 'NN'), ("''", "''", '$('), ('den', 'den', 'ART'), ('Oscar', 'Osc
ar', 'NN'), ('in', 'in', 'APPR'), ('der', 'der', 'ART'), ('Rubrik', 'Ru
brik', 'NN'), ('``', '``', '$('), ('Beste', 'gut', 'ADJA'), ('Regie', '
Regie', 'NN'), ("''", "''", '$('), (',', '--', '$,'), ('1999', '1999',
'CARD'), ('eine', 'eine', 'ART'), ('Oscar-Nominierung', 'Oscar-nominier
ung', 'NN'), ('in', 'in', 'APPR'), ('der', 'der', 'ART'), ('Kategorie',
'Kategorie', 'NN'), ('``', '``', '$('), ('Bestes', 'Gute', 'NN'), ('ada
ptiertes', 'adaptiert', 'ADJA'), ('Drehbuch', 'Drehbuch', 'NN'), ("''",
"''", '$('), ('für', 'Für', 'NE'), ('``', '``', '$('), ('Der', 'der', '
ART'), ('talentierte', 'talentiert', 'ADJA'), ('Mr', 'mr', 'XY'), ('.',
'--', '$.'), ('Ripley', 'Ripley', 'NE'), ("''", "''", '$('), (',', '--'
, '$,'), ('bei', 'bei', 'APPR'), ('dem', 'dem', 'PRELS'), ('er', 'er',
'PPER'), ('auch', 'auch', 'ADV'), ('Regie', 'Regie', 'NN'), ('führte',
'führte', 'FM'), ('.', '--', '$.'), ('2001', '2001', 'CARD'), ('wurde',
'werden', 'VAFIN'), ('Minghella', 'Minghella', 'NE'), ('zum', 'zum', 'A
PPRART'), ('Commander', 'Commander', 'NN'), ('of', 'of', 'FM'), ('the',
'the', 'FM'), ('British', 'british', 'FM'), ('Empire', 'empire', 'FM'),
('ernannt', 'ernennen', 'VVPP'), ('.', '--', '$.'), ('Von', 'von', 'APP
R'), ('2003', '2003', 'CARD'), ('bis', 'bis', 'KON'), ('2007', '2007',
'CARD'), ('war', 'sein', 'VAFIN'), ('er', 'er', 'PPER'), ('Präsident',
'Präsident', 'NN'), ('des', 'des', 'ART'), ('British', 'British', 'NE')
, ('Film', 'Film', 'NN'), ('Institute', 'Institute', 'NE'), ('.', '--',
'$.'), ('Seit', 'seit', 'APPR'), ('1997', '1997', 'CARD'), ('trägt', 't
rägen', 'VVFIN'), ('das', 'das', 'ART'), ('Anthony', 'Anthony', 'NE'),
('Minghella', 'Minghella', 'NE'), ('Theatre', 'Theatre', 'NE'), ('auf',
'auf', 'APPR'), ('der', 'der', 'ART'), ('Isle', 'Isle', 'NE'), ('of', '
of', 'FM'), ('Wight', 'wight', 'FM'), ('seinen', 'sein', 'PPOSAT'), ('N
amen', 'Name', 'NN'), ('.', '--', '$.')]

8.5 Rezultati korjenovatelja

U algoritmu se primjenjuju dva korjenovatelja: Porterov korjenovatelj i Snowball korjenovatelj. Primjeri i rezultati primjene svakog od korjenovatelja.

8.5.1 Primjena Porterovog korjenovatelja

Primjer teksta nakon korjenovanja:

```
['minghella', 'war', 'der', 'sohn', 'italienisch-schottisch', 'eltern',
',', 'die', 'auf', 'der', 'isl', 'of', 'wight', 'ein', 'fabrik', 'für',
'eiscrem', 'betrieben', '.', 'nach', 'seinem', 'schulabschluss', 'studi
ert', 'er', 'an', 'der', 'universität', 'hull', ',', 'wo', 'er', 'ein',
'zeit', 'lang', 'al', 'dozent', 'tätig', 'war', '.', '1978', 'dreht', '
er', 'einen', 'ersten', 'kurzfilm', '.', 'seit', '1981', 'war', 'er', '
al', 'autor', 'und', 'stori', 'editor', 'tätig', '.', 'er', 'wurd', 'mi
t', 'theaterstücken', ',', 'rundfunkhörspielen', ',', 'der', 'fernsehse
ri', '``', 'inspector', 'mors', "''", 'und', 'vielen', 'drehbüchern', '
für', 'film', 'und', 'fernsehen', 'bekannt', '.', 'er', 'entwickelt', '
die', 'drehbüch', 'für', 'die', '1988', 'erfolgreich', 'ausgestrahlt',
'fernsehseri', 'the', 'storytel', 'von', 'muppets-erfind', 'jim', 'hens
on', '.', 'auch', 'al', 'produz', 'war', 'er', 'erfolgreich', ',', 'dar
unt', 'für', 'die', 'film', '``', 'der', 'still', 'amerikan', "''", ','
, '``', 'die', 'dolmetscherin', "''", 'und', '``', 'der', 'vorles', "''
", ',', 'für', 'den', 'er', '2008', 'posthum', 'für', 'den', 'oscar', '
nominiert', 'wurd', '.', 'gemeinsam', 'mit', 'seinem', 'freund', 'und',
'kollegen', 'sydney', 'pollack', 'gründet', 'er', 'die', 'produktionsfi
rma', 'mirag', 'enterpris', '.', 'der', 'regisseur', 'minghella', 'galt
', 'al', 'ein', 'guter', 'schauspielerführ', ':', 'unter', 'seiner', 'r
egi', 'brachten', 'es', 'zahlreich', 'darstel', 'zu', 'oscar-nominierun
gen', ',', 'zwei', 'schauspielerinnen', 'erhielten', 'die', 'auszeichnu
ng', 'al', '``', 'best', 'nebendarstellerin', "''", ':', 'juliett', 'bi
noch', 'und', 'rené', 'zellweg', '.', 'gegen', 'end', 'sein', 'leben',
'kehrt', 'minghella', 'zu', 'seinen', 'anfängen', 'im', 'radio', 'und',
'auf', 'der', 'bühn', 'zurück', ':', '2006', 'wurd', 'sein', 'hörspiel'
, '``', 'eye', 'down', 'look', "''", 'mit', 'jude', 'law', 'zu', 'ehren
', 'von', 'samuel', 'beckett', 'auf', 'bbc', 'radio', '3', 'ausgestrahl
t', ',', 'ein', 'jahr', 'zuvor', 'hatt', 'sein', 'inszenierung', 'der',
'puccini-op', 'madam', 'butterfli', 'in', 'der', 'english', 'nation', '
opera', 'in', 'london', 'premier', 'und', 'wurd', 'auch', 'in', 'der',
'nationalop', 'von', 'vilniu', 'und', 'in', 'der', 'metropolitan', 'ope
```

```
ra', 'in', 'new', 'york', 'gezeigt', '.', 'am', 'end', 'de', 'film', '`
`', 'abbitt', "'", 'von', 'joe', 'wright', 'hat', 'er', 'einen', 'kurz
auftritt', 'al', 'talkshow-moder', 'neben', 'vanessa', 'redgrav', '.',
'sein', 'letzt', 'arbeit', 'al', 'drehbuchautor', 'war', 'da', 'skript'
, 'für', 'den', 'musical-film', '``', 'nine', "'", '.', 'zu', 'seinen'
, 'letzten', 'regiearbeiten', 'zählt', 'der', 'pilotfilm', 'zur', 'krim
iseri', '``', 'ein', 'detektivin', 'für', 'botswana', "'", ',', 'den',
'die', 'bbc', 'fünf', 'tage', 'nach', 'seinem', 'tod', 'erstmal', 'auss
trahlt', '.', 'minghella', 'war', 'mit', 'der', 'au', 'hongkong', 'stam
menden', 'choreographin', ',', 'produzentin', 'und', 'schauspielerin',
'carolyn', 'choa', 'verheiratet', '.', 'der', 'ehe', 'entstammen', 'zwe
i', 'kinder', ',', 'die', 'in', 'der', 'filmbranch', 'tätig', 'sind', '
:', 'tochter', 'hannah', 'minghella', 'in', 'der', 'produkt', 'und', 's
ohn', 'max', 'minghella', 'al', 'schauspiel', '.', 'die', 'tant', 'edan
a', 'minghella', 'und', 'der', 'onkel', 'domin', 'minghella', 'sind', '
drehbuchautoren', '.', 'minghella', 'starb', 'im', 'alter', 'von', '54'
, 'jahren', 'in', 'einem', 'london', 'krankenhau', 'an', 'inneren', 'bl
utungen', 'infolg', 'der', 'oper', 'ein', 'tonsillenkarzinom', 'und', '
ein', 'karzinom', 'im', 'nacken', '.', '1984', 'erhielt', 'minghella',
'den', 'london', 'kritikerprei', 'al', 'meistversprechend', 'junger', '
dramatik', ',', '1986', 'den', 'kritikerprei', 'für', 'sein', 'stück',
'``', 'made', 'in', 'bangkok', "'", 'al', 'best', 'stück', 'der', 'sai
son', '.', '1997', 'erhielt', 'er', 'für', '``', 'der', 'englisch', 'pa
tient', "'", 'den', 'oscar', 'in', 'der', 'rubrik', '``', 'best', 'reg
i', "'", ',', '1999', 'ein', 'oscar-nominierung', 'in', 'der', 'kategor
ri', '``', 'best', 'adaptiert', 'drehbuch', "'", 'für', '``', 'der', '
talentiert', 'mr', '.', 'ripley', "'", ',', 'bei', 'dem', 'er', 'auch'
, 'regi', 'führt', '.', '2001', 'wurd', 'minghella', 'zum', 'command',
'of', 'the', 'british', 'empir', 'ernannt', '.', 'von', '2003', 'bi', '
2007', 'war', 'er', 'präsid', 'de', 'british', 'film', 'institut', '.',
'seit', '1997', 'trägt', 'da', 'anthoni', 'minghella', 'theatr', 'auf',
'der', 'isl', 'of', 'wight', 'seinen', 'namen', '.']
```

8.5.2 Primjena Snowball korjenovatelja

Primjer teksta nakon primjene Snowball korjenovatelja:

```
['minghella', 'war', 'der', 'sohn', 'italienisch-schott', 'elt', ',', '
die', 'auf', 'der', 'isl', 'of', 'wight', 'ein', 'fabrik', 'für', 'eisc
rem', 'betrieb', '.', 'nach', 'sein', 'schulabschluss', 'studiert', 'er
', 'an', 'der', 'universität', 'hull', ',', 'wo', 'er', 'ein', 'zeit',
'lang', 'als', 'dozent', 'tätig', 'war', '.', '1978', 'dreht', 'er', 'e
in', 'erst', 'kurzfilm', '.', 'seit', '1981', 'war', 'er', 'als', 'auto
```

```
r', 'und', 'story', 'editor', 'tätig', '.', 'er', 'wurd', 'mit', 'theat
erstück', ',', 'rundfunkhörspiel', ',', 'der', 'fernsehseri', '``', 'in
spector', 'mors', "'", 'und', 'viel', 'drehbüch', 'für', 'film', 'und'
, 'fernseh', 'bekannt', '.', 'er', 'entwickelt', 'die', 'drehbüch', 'fü
r', 'die', '1988', 'erfolgreich', 'ausgestrahlt', 'fernsehseri', 'the',
'storytell', 'von', 'muppets-erfind', 'jim', 'henson', '.', 'auch', 'al
s', 'produzent', 'war', 'er', 'erfolgreich', ',', 'darunt', 'für', 'die
', 'film', '``', 'der', 'still', 'amerikan', "'", ',', '``', 'die', 'd
olmetscherin', "'", 'und', '``', 'der', 'vorles', "'", ',', 'für', 'd
en', 'er', '2008', 'posthum', 'für', 'den', 'oscar', 'nominiert', 'wurd
', '.', 'gemeinsam', 'mit', 'sein', 'freund', 'und', 'kolleg', 'sydney'
, 'pollack', 'gründet', 'er', 'die', 'produktionsfirma', 'mirag', 'ente
rpris', '.', 'der', 'regisseur', 'minghella', 'galt', 'als', 'ein', 'gu
t', 'schauspielerführ', ':', 'unt', 'sein', 'regi', 'bracht', 'es', 'za
hlreich', 'darstell', 'zu', 'oscar-nominier', ',', 'zwei', 'schauspiele
rinn', 'erhielt', 'die', 'auszeichn', 'als', '``', 'best', 'nebendarste
llerin', "'", ':', 'juliett', 'binoch', 'und', 'rené', 'zellweg', '.',
'geg', 'end', 'sein', 'leb', 'kehrt', 'minghella', 'zu', 'sein', 'anfän
g', 'im', 'radio', 'und', 'auf', 'der', 'bühn', 'zurück', ':', '2006',
'wurd', 'sein', 'hörspiel', '``', 'eyes', 'down', 'looking', "'", 'mit
', 'jud', 'law', 'zu', 'ehr', 'von', 'samuel', 'beckett', 'auf', 'bbc',
'radio', '3', 'ausgestrahlt', ',', 'ein', 'jahr', 'zuvor', 'hatt', 'sei
n', 'inszenier', 'der', 'puccini-op', 'madam', 'butterfly', 'in', 'der'
, 'english', 'national', 'opera', 'in', 'london', 'premi', 'und', 'wurd
', 'auch', 'in', 'der', 'nationalop', 'von', 'vilnius', 'und', 'in', 'd
er', 'metropolitan', 'opera', 'in', 'new', 'york', 'gezeigt', '.', 'am'
, 'end', 'des', 'film', '``', 'abbitt', "'", 'von', 'joe', 'wright', '
hat', 'er', 'ein', 'kurzauftritt', 'als', 'talkshow-moderator', 'neb',
'vanessa', 'redgrav', '.', 'sein', 'letzt', 'arbeit', 'als', 'drehbucha
utor', 'war', 'das', 'skript', 'für', 'den', 'musical-film', '``', 'nin
', "'", '.', 'zu', 'sein', 'letzt', 'regiearbeit', 'zählt', 'der', 'pi
lotfilm', 'zur', 'krimiseri', '``', 'ein', 'detektivin', 'für', 'botswa
na', "'", ',', 'den', 'die', 'bbc', 'fünf', 'tag', 'nach', 'sein', 'to
d', 'erstmal', 'ausstrahlt', '.', 'minghella', 'war', 'mit', 'der', 'au
s', 'hongkong', 'stammend', 'choreographin', ',', 'produzentin', 'und',
'schauspielerin', 'carolyn', 'choa', 'verheiratet', '.', 'der', 'ehe',
'entstamm', 'zwei', 'kind', ',', 'die', 'in', 'der', 'filmbranch', 'tät
ig', 'sind', ':', 'tocht', 'hannah', 'minghella', 'in', 'der', 'produkt
ion', 'und', 'sohn', 'max', 'minghella', 'als', 'schauspiel', '.', 'die
', 'tant', 'edana', 'minghella', 'und', 'der', 'onkel', 'dominic', 'min
ghella', 'sind', 'drehbuchautor', '.', 'minghella', 'starb', 'im', 'alt
', 'von', '54', 'jahr', 'in', 'ein', 'london', 'krankenhaus', 'an', 'in
n', 'blutung', 'infolg', 'der', 'operation', 'ein', 'tonsillenkarzinom'
, 'und', 'ein', 'karzinom', 'im', 'nack', '.', '1984', 'erhielt', 'ming
hella', 'den', 'london', 'kritikerpreis', 'als', 'meistversprech', 'jun
g', 'dramat', ',', '1986', 'den', 'kritikerpreis', 'für', 'sein', 'stüc
k', '``', 'mad', 'in', 'bangkok', "'", 'als', 'best', 'stück', 'der',
'saison', '.', '1997', 'erhielt', 'er', 'für', '``', 'der', 'englisch',
'patient', "'", 'den', 'oscar', 'in', 'der', 'rubrik', '``', 'best', '
regi', "'", ',', '1999', 'ein', 'oscar-nominier', 'in', 'der', 'katego
ri', '``', 'best', 'adaptiert', 'drehbuch', "'", 'für', '``', 'der', '
talentiert', 'mr', '.', 'ripley', "'", ',', 'bei', 'dem', 'er', 'auch'
, 'regi', 'führt', '.', '2001', 'wurd', 'minghella', 'zum', 'command',
'of', 'the', 'british', 'empir', 'ernannt', '.', 'von', '2003', 'bis',
'2007', 'war', 'er', 'präsident', 'des', 'british', 'film', 'institut',
'.', 'seit', '1997', 'trägt', 'das', 'anthony', 'minghella', 'theatr',
'auf', 'der', 'isl', 'of', 'wight', 'sein', 'nam', '.']
```

8.6 Čišćenje teksta od stop-riječi

Ovaj korak je moguće napraviti čak i nakon koraka lematizacije. Ovdje se prikazuju rezultati čišćenja riječi nakon lematizacije i nakon korjenovanja.

8.6.1 Rezultati čišćenja teksta nakon koraka lematizacije

Primjer teksta očišćenog od stop-riječi nakon koraka lematizacije i tokenizacije rečenica:

Minghella Sohn italienisch-schottischer Eltern , Isle of Wight Fabrik f ür Eiscreme betrieben . Nach Schulabschluss studierte Universität Hull , Zeit lang Dozent tätig . 1978 drehte ersten Kurzfilm . Seit 1981 Auto r Story Editor tätig . Er wurde Theaterstücken , Rundfunkhörspielen , F ernsehserie `` Inspector Morse '' vielen Drehbüchern für Film Fernsehen bekannt . Er entwickelte Drehbücher für 1988 erfolgreich ausgestrahlte Fernsehserie The Storyteller Muppets-Erfinder Jim Henson . Auch Produze nt erfolgreich , darunter für Filme `` Der stille Amerikaner '' , `` Di e Dolmetscherin '' `` Der Vorleser '' , für 2008 posthum für Oscar nomi niert wurde . Gemeinsam Freund Kollegen Sydney Pollack gründete Produkt ionsfirma Mirage Enterprises . Der Regisseur Minghella galt guter Schau spielerführer : Unter Regie brachten zahlreiche Darsteller Oscar-Nomini erungen , zwei Schauspielerinnen erhielten Auszeichnung `` Beste Nebend arstellerin '' : Juliette Binoche Renée Zellweger . Gegen Ende Lebens k ehrte Minghella Anfängen Radio Bühne zurück : 2006 wurde Hörspiel `` Ey es Down Looking '' Jude Law Ehren Samuel Beckett BBC Radio 3 ausgestrah lt , Jahr zuvor Inszenierung Puccini-Oper Madame Butterfly English Nati onal Opera London Premiere wurde Nationaloper Vilnius Metropolitan Oper a New York gezeigt . Am Ende Films `` Abbitte '' Joe Wright Kurzauftrit t Talkshow-Moderator neben Vanessa Redgrave . Seine letzte Arbeit Drehb uchautor Skript für Musical-Film `` Nine '' . Zu letzten Regiearbeiten zählt Pilotfilm Krimiserie `` Eine Detektivin für Botswana '' , BBC fün f Tage Tod erstmals ausstrahlte . Minghella Hongkong stammenden Choreog raphin , Produzentin Schauspielerin Carolyn Choa verheiratet . Der Ehe entstammen zwei Kinder , Filmbranche tätig : Tochter Hannah Minghella P roduktion Sohn Max Minghella Schauspieler . Die Tante Edana Minghella O nkel Dominic Minghella Drehbuchautoren . Minghella starb Alter 54 Jahre n Londoner Krankenhaus inneren Blutungen infolge Operation Tonsillenkar zinoms Karzinoms Nacken . 1984 erhielt Minghella Londoner Kritikerpreis meistversprechender junger Dramatiker , 1986 Kritikerpreis für Stück `` Made Bangkok '' bestes Stück Saison . 1997 erhielt für `` Der englische Patient '' Oscar Rubrik `` Beste Regie '' , 1999 Oscar-Nominierung Kate

```
gorie `` Bestes adaptiertes Drehbuch '' für `` Der talentierte Mr. Ripl
ey '' , Regie führte . 2001 wurde Minghella Commander of the British Em
pire ernannt . Von 2003 2007 Präsident British Film Institute . Seit 19
97 trägt Anthony Minghella Theatre Isle of Wight Namen .
```

8.7 Ispis rezultata dobivenih ngrama korpusa

Ovdje se prilažu svi rezultati ispisa ngrama. Pri tome se definiraju različiti parametri poput frekvencije pojavljivanja određenih ngrama, dohvaćanje i prikaz top n kolokacija u odnosu na ..definiranu frekvenciju itd.

8.7.1 Bigrami

Prvi ispis, koji je također vidljiv pod naslovom 5.2.2 Ngrami i njihov prikaz uključuje filter prethodnog definiranja željene frekvencije rezultata, u ovom slučaju frekvecnije 1, te se prikazuje top 40 rezultata:

```
[("'Abd", 'or-Rahmān'),
 ("'Active", "Brake'-Differenzial"),
 ("'Alleen", 'Gezamenlijk'),
 ("'CHESEBROUGH", 'VASELINE'),
 ("'De", 'Bloempot'),
 ("'Designer", 'Collaborations'),
 ("'Etezad", 'od-Doleh'),
 ("'Eyn", 'ol-Molk'),
 ("'Fair", 'Ellender'),
 ("'Fakhr", 'ol-Zakerin'),
 ("'Glen", "Affric'-Anlage"),
 ("'Joyce", 'Dixey'),
 ("'Juan", 'Buscón'),
 ("'Kissengen", 'tablets'),
 ("'Leidse", 'Instrumentmakersschool'),
 ("'Louise", 'Tegern'),
 ("'Lux", 'Aurumque'),
 ("'Magna", 'Glossatura'),
 ("'Miscellanae", 'observationis'),
 ("'Otto", 'Froebel'),
 ("'Samisch-finnische", 'Bootsterminologie'),
 ("'Seison", 'Toguchi'),
 ("'bona", 'consilia'),
 ("'corporate", 'accumulation'),
```

```
 ("'crème", "renversée'/Un-Safe"),
 ("'de", 'Neufvillei'),
 ("'français", 'parlé'),
 ("'für", 'pathologisch-chemische'),
 ("'ihr", 'allertiefstes'),
 ("'jeux", "d'esprit"),
 ("'königlich-bairische", 'Land-Commissär'),
 ("'les", 'museiques'),
 ("'manum", 'pressum'),
 ("'me", 'Paoktzis'),
 ("'nem", 'Pushkick'),
 ("'network", 'lock'),
 ("'posti", 'tappa'),
 ("'self", 'fulfilling'),
 ("'true", 'patriot'),
 ('+XPE', 'ADIVVA')]
```

Ispis top 40 bigrama koji su se pojavili barem 3 puta

```
[('536к', 'Kaliningrad-Mosyr'),
 ('AGM-88', 'HARM'),
 ('Abdelmalik', 'Ladjali'),
 ('Abdurrahim', 'Kuzu'),
 ('Ahrue', 'Luster'),
 ('Al-Abidyn', 'S-Latef'),
 ('Alasdair', 'MacColla'),
 ('Aleksandre', 'Iaschwili'),
 ('Alephis', 'tigneresi'),
 ('Alexis-Armand', 'Charost'),
 ('Aloizs', 'Tumiņš'),
 ('Anvar', 'Yunusow'),
 ('Apti', 'Auchadow'),
 ('Ariada-Akpars', 'Wolschsk'),
 ('Arnaldur', 'Indriðason'),
 ('Arutik', 'Rubenjan'),
 ('Astrit', 'Ajdarevic'),
 ('Aviron', 'Bayonnais'),
 ('Államvédelmi', 'Hatóság'),
 ('Äbdulchakim', 'Schäpijew'),
 ('Badain', 'Jaran'),
 ('Banderia', 'Prutenorum'),
 ('Baroner', 'Kritzli'),
 ('Basílio', 'Krevey'),
 ('Bathys', 'Ryax'),
 ('Batrachochytrium', 'dendrobatidis'),
 ('Bärenthorener', 'Kiefernwirtschaft'),
 ('Beirami', 'Baher'),
 ('Benegal', 'Narsing'),
 ('Benthem', 'Crouwel'),
 ('Bereza', 'Kartuska'),
 ('Berkant', 'Göktan'),
```

```
 ('Bjarke', 'Ingels'),
 ('Blutgerinnungsvorstufe', 'Fibrinogen'),
 ('Blutjunge', 'Verführerinnen'),
 ('Bosansko', 'Grahovo'),
 ('Branka', 'Musulin'),
 ('Bristle', 'Blasting'),
 ('Budvanska', 'Rivijera'),
 ('Bugsprietsegel', 'setzbar')]
```

8.7.2 Trigrami

Prikaz top 40 trigrama sa frekvencijom 1:

```
[("'Abd", 'or-Rahmān', 'Dschāmi'),
 ("'Alleen", 'Gezamenlijk', 'Oefenen'),
 ("'Lux", 'Aurumque', 'reimagined'),
 ('+XPE', 'ADIVVA', 'HLOTARIVM'),
 ('-//Apple', 'Computer//DTD', 'PLIST'),
 ('-127', 'mm/L', '38-Mk.12-Kanone'),
 ('-76', 'mm-/L', '50-Mk.17-Kanonen'),
 ('-Resko', '-Świdwin', '-Buślary'),
 ('.22-kalibrigen', 'achtschüssigen', 'Iver-Johnson-Revolver'),
 ('0000ULOBNN', '000100014459392326983661684 7', '00010001'),
 ('08-09', 'F/W', 'SFAA'),
 ('10,5cm', 'hrubý', 'kanón'),
 ('100-Meter-Sprinterin', 'Jabou', 'Jawo'),
 ('1135/36', 'Volodar', 'Gļebovič'),
 ('171n', 'Herlasgrün-Falkenstein-Klingenthal', '-Falkenau'),
 ('20-bar-Heissdampfsammelschiene', 'hängenenden', 'Drehstromturbosätze
'),
 ('260-tägigen', 'Ritualkalender', 'tonalpohualli'),
 ('28-cm-Doppeltürme', 'Drh', 'LC/1907'),
 ('3-bus', 'triticis', 'siliginis'),
 ('400-Meter-Läuferin', 'Tonique', 'Williams-Darling'),
 ('7.5-minute', 'topographic', 'quadrangle'),
 ('A/L', 'Namdal', 'Kornsilo'),
 ('ABELS', 'KALLWASS', 'STITZ'),
 ('ADDERE', 'PRAETEREA', 'DEXTRA'),
 ('ADMIRANDAE', 'COMMISERATIONIS', 'PRODIGIO'),
 ('AFFERENT', 'FILIOS', 'TVOS'),
 ('AGNETE', 'ILSABEIN', 'SCHULTE'),
 ('ANDREA', 'UGOLINI', 'NINI'),
 ('APOSTOLO', 'SCOTORVM', 'PATRONO'),
 ('ARCHIVIS', 'FATEOR', 'VOLVI'),
 ('ARCUM', 'TRIUMPHIS', 'INSIGNEM'),
 ('ARDINOLT', 'REINHART', 'OBELOTE'),
 ('ARMEN', 'MOSE', '33,27'),
 ('ARMIS', 'ARCUM', 'TRIUMPHIS'),
 ('ARTIVM', 'LIBERALIVM', 'MVSEVM'),
 ('ASPANGBAHHOF', 'ZEHNTAUSENDE', 'ÖSTERREICHISCHE'),
 ('ASSISTENTE', 'MIHI', 'DULCI'),
 ('Aafje', 'Talea', 'Vrijer'),
 ('Activator-like', 'Effector', 'Nucleases'),
 ('Adeliche', 'Creichgauerischen', 'Fräulein-Stifft')]
```

8.8 Sumarizacija i ispis rezultata

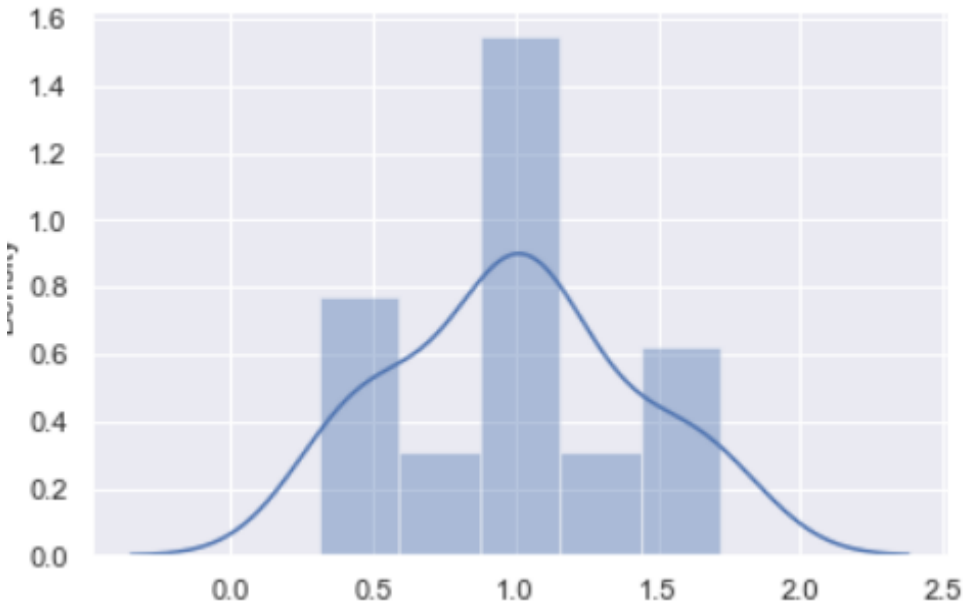
Ovdje je se prikazuju rezultati sumarizacija LexRanka, TextRanka, SBKE te algoritma sažimanja tekstova na temelju KL divergencije iz sumy paketa.

8.8.1 Ispis Lexrank rezultata

Tijekom pokretanja LexRank algoritma, moguće je definirati nekoliko parametara. Sljedeći rezultati dobiveni su kada je prag postavljen na 0.1 te veličina sažetka na top 2 rečenice. Nakon primjene čišćenja. Rezultat je sljedeći:

```
Der Ehe entstammen zwei Kinder, die in der Filmbranche tätig si
nd: Tochter Hannah Minghella in der Produktion und Sohn Max Min
ghella als Schauspieler .', 'Minghella war der Sohn italienisch
-schottischer Eltern, die auf der Isle of Wight eine Fabrik für
Eiscreme betrieben.
```

Kada želimo vidjeti gustoću u odnosu na LexRank vrijednostima za rečenice, dobiva se sljedeći graf sa pragom 0.1:



Slika 0-1: Prikaz rezultata LexRank rezultata

Rezultat za top 5 rečenica s istim postavkama LexRanka:

```
Der Ehe entstammen zwei Kinder, die in der Filmbranche tätig si
nd: Tochter Hannah Minghella in der Produktion und Sohn Max Min
ghella als Schauspieler .', 'Minghella war der Sohn italienisch
```

-schottischer Eltern, die auf der Isle of Wight eine Fabrik für Eiscreme betrieben.', 'Die Tante Edana Minghella und der Onkel Dominic Minghella sind Drehbuchautoren.', '1997 erhielt er für "Der englische Patient" den Oscar in der Rubrik "Beste Regie", 1999 eine Oscar-Nominierung in der Kategorie "Bestes adaptierte s Drehbuch" für "Der talentierte Mr.', '2001 wurde Minghella zu m Commander of the British Empire ernannt.

Rezultat za top 10 rečenica s istim postavkama LexRanka:

Der Ehe entstammen zwei Kinder, die in der Filmbranche tätig si nd: Tochter Hannah Minghella in der Produktion und Sohn Max Min ghella als Schauspieler .', 'Minghella war der Sohn italienisch -schottischer Eltern, die auf der Isle of Wight eine Fabrik für Eiscreme betrieben.', 'Die Tante Edana Minghella und der Onkel Dominic Minghella sind Drehbuchautoren.', '1997 erhielt er für "Der englische Patient" den Oscar in der Rubrik "Beste Regie", 1999 eine Oscar-Nominierung in der Kategorie "Bestes adaptierte s Drehbuch" für "Der talentierte Mr.', '2001 wurde Minghella zu m Commander of the British Empire ernannt.', 'Auch als Produzen t war er erfolgreich, darunter für die Filme "Der stille Amerik aner", "Die Dolmetscherin" und "Der Vorleser", für den er 2008 posthum für den Oscar nominiert wurde.', 'Seit 1997 trägt das A nthony Minghella Theatre auf der Isle of Wight seinen Namen.', '1984 erhielt Minghella den Londoner Kritikerpreis als meistver sprechender junger Dramatiker, 1986 den Kritikerpreis für sein Stück "Made in Bangkok" als bestes Stück der Saison.', 'Er wurd e mit Theaterstücken, Rundfunkhörspielen, der Fernsehserie "Ins pector Morse" und vielen Drehbüchern für Film und Fernsehen bek annt.', 'Seine letzte Arbeit als Drehbuchautor war das Skript f ür den Musical-Film "Nine" .

Rezultat za najbitniju rečenicu u prvom tekstu s istim postavkama:

Der Ehe entstammen zwei Kinder, die in der Filmbranche tätig si
nd: Tochter Hannah Minghella in der Produktion und Sohn Max Min
ghella als Schauspieler .

Kada uklonimo prag, rezultati za top 1, 2, 5 i 10 rečenica su sljedeći:

- Top 1:

Minghella war der Sohn italienisch-schottischer Eltern, die auf
der Isle of Wight eine Fabrik für Eiscreme betrieben.

- Top 2:

Minghella war der Sohn italienisch-schottischer Eltern, die auf
der Isle of Wight eine Fabrik für Eiscreme betrieben.', 'Die Ta
nte Edana Minghella und der Onkel Dominic Minghella sind Drehbu
chautoren.

- Top 5:

Minghella war der Sohn italienisch-schottischer Eltern, die auf
der Isle of Wight eine Fabrik für Eiscreme betrieben.', 'Die Ta
nte Edana Minghella und der Onkel Dominic Minghella sind Drehbu
chautoren.', 'Der Ehe entstammen zwei Kinder, die in der Filmbr
anche tätig sind: Tochter Hannah Minghella in der Produktion un
d Sohn Max Minghella als Schauspieler .', '2001 wurde Minghella
zum Commander of the British Empire ernannt.', 'Auch als Produz
ent war er erfolgreich, darunter für die Filme "Der stille Amer
ikaner", "Die Dolmetscherin" und "Der Vorleser", für den er 200
8 posthum für den Oscar nominiert wurde.

- Top 10:

Minghella war der Sohn italienisch-schottischer Eltern, die auf
der Isle of Wight eine Fabrik für Eiscreme betrieben.', 'Die Ta
nte Edana Minghella und der Onkel Dominic Minghella sind Drehbu
chautoren.', 'Der Ehe entstammen zwei Kinder, die in der Filmbr
anche tätig sind: Tochter Hannah Minghella in der Produktion un
d Sohn Max Minghella als Schauspieler .', '2001 wurde Minghella

zum Commander of the British Empire ernannt.', 'Auch als Produz ent war er erfolgreich, darunter für die Filme "Der stille Amer ikaner", "Die Dolmetscherin" und "Der Vorleser", für den er 200 8 posthum für den Oscar nominiert wurde.', '1997 erhielt er für "Der englische Patient" den Oscar in der Rubrik "Beste Regie", 1999 eine Oscar-Nominierung in der Kategorie "Bestes adaptierte s Drehbuch" für "Der talentierte Mr.', 'Seit 1997 trägt das Ant hony Minghella Theatre auf der Isle of Wight seinen Namen.', '1 984 erhielt Minghella den Londoner Kritikerpreis als meistversp rechender junger Dramatiker, 1986 den Kritikerpreis für sein St ück "Made in Bangkok" als bestes Stück der Saison.', 'Er wurde mit Theaterstücken, Rundfunkhörspielen, der Fernsehserie "Inspe ctor Morse" und vielen Drehbüchern für Film und Fernsehen bekan nt.', 'Am Ende des Films "Abbitte" von Joe Wright hat er einen Kurzauftritt als Talkshow-Moderator neben Vanessa Redgrave.

Postavljanjem praga na 0.5 dobivaju se sljedeći rezultati za top 1, 2, 5 i 10 rečenica

- Top 1:
- Seit 1997 trägt das Anthony Minghella Theatre auf der Isl e of Wight seinen Namen.

- Top 2:

Seit 1997 trägt das Anthony Minghella Theatre auf der Isle of W ight seinen Namen.', 'Am Ende des Films "Abbitte" von Joe Wrigh t hat er einen Kurzauftritt als Talkshow-Moderator neben Vaness a Redgrave.

- Top 5:

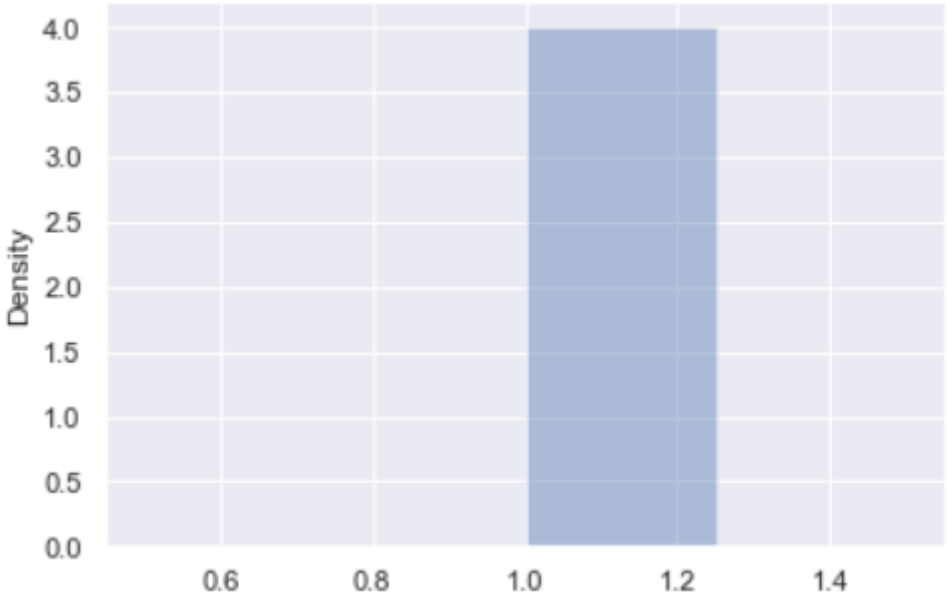
Seit 1997 trägt das Anthony Minghella Theatre auf der Isle of W ight seinen Namen.', 'Am Ende des Films "Abbitte" von Joe Wrigh t hat er einen Kurzauftritt als Talkshow-Moderator neben Vaness a Redgrave.', 'Nach seinem Schulabschluss studierte er an der U niversität Hull, wo er eine Zeit lang als Dozent tätig war.', '

1978 drehte er einen ersten Kurzfilm.', 'Seit 1981 war er als A utor und Story Editor tätig.

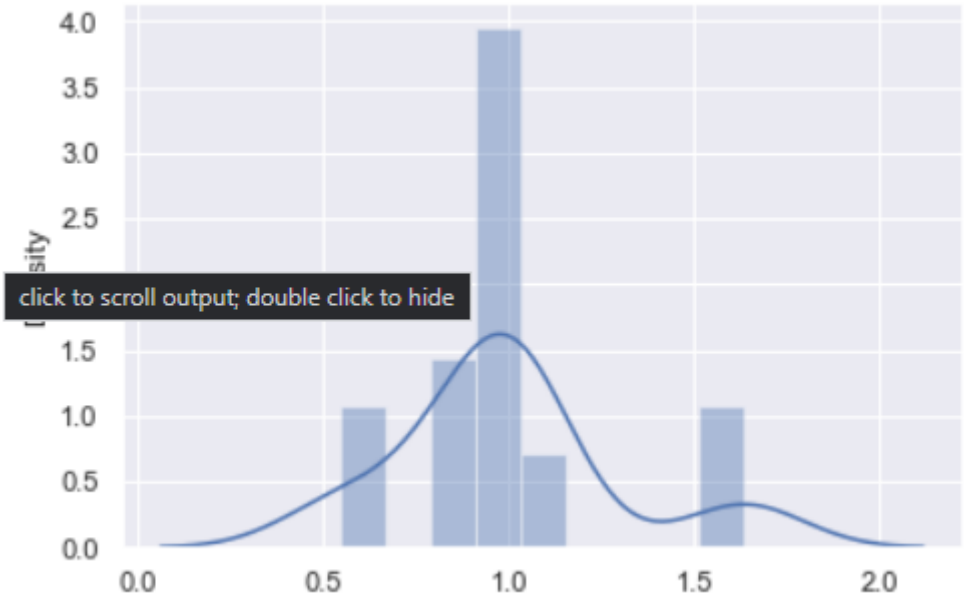
- Top 10:

Minghella war der Sohn italienisch-schottischer Eltern, die auf der Isle of Wight eine Fabrik für Eiscreme betrieben.', 'Die Ta nte Edana Minghella und der Onkel Dominic Minghella sind Drehbu chautoren.', 'Der Ehe entstammen zwei Kinder, die in der Filmbr anche tätig sind: Tochter Hannah Minghella in der Produktion un d Sohn Max Minghella als Schauspieler .', '2001 wurde Minghella zum Commander of the British Empire ernannt.', 'Auch als Produz ent war er erfolgreich, darunter für die Filme "Der stille Amer ikaner", "Die Dolmetscherin" und "Der Vorleser", für den er 200 8 posthum für den Oscar nominiert wurde.', '1997 erhielt er für "Der englische Patient" den Oscar in der Rubrik "Beste Regie", 1999 eine Oscar-Nominierung in der Kategorie "Bestes adaptierte s Drehbuch" für "Der talentierte Mr.', 'Seit 1997 trägt das Ant hony Minghella Theatre auf der Isle of Wight seinen Namen.', '1 984 erhielt Minghella den Londoner Kritikerpreis als meistversp rechender junger Dramatiker, 1986 den Kritikerpreis für sein St ück "Made in Bangkok" als bestes Stück der Saison.', 'Er wurde mit Theaterstücken, Rundfunkhörspielen, der Fernsehserie "Inspe ctor Morse" und vielen Drehbüchern für Film und Fernsehen bekan nt.', 'Am Ende des Films "Abbitte" von Joe Wright hat er einen Kurzauftritt als Talkshow-Moderator neben Vanessa Redgrave.

Prilikom pregleda gustoće u odnosu na LexRank vrijednosti rečenica te smanjivanjem praga na 0.2 dobivaju se sljedeći rezultati:



Slika 0-2: Prikaz gustoće u odnosu na LexRank vrijednosti rečenica s pragom 0.5



Slika 0-3: Prikaz gustoće u odnosu na LexRank vrijednosti rečenica s pragom 0.2

8.8.2 Ispis rezultata LexRank algoritma iz sumy biblioteke

Ovdje su prikazani rezultati LexRank algoritma koji je primijenjen na temelju sumy biblioteke. Prikazani rezultati su pročišćeni za evaluaciju, jer je standardan izlaz algoritma isti onome TextRank algoritma.

Za top 1 rezultati prvog teksta je sljedeći:

Von 2003 bis 2007 war er Präsident des British Film Institute.

Za top 2 rečenice, rezultati su sljedeći:

Der Ehe entstammen zwei Kinder, die in der Filmbranche tätig si nd: Tochter Hannah Minghella in der Produktion und Sohn Max Min ghella als Schauspieler .Von 2003 bis 2007 war er Präsident des British Film Institute.

Za top 5 rečenica rezultati su sljedeći:

Auch als Produzent war er erfolgreich, darunter für die Filme " Der stille Amerikaner", "Die Dolmetscherin" und "Der Vorleser", für den er 2008 posthum für den Oscar nominiert wurde.Der Ehe e ntstammen zwei Kinder, die in der Filmbranche tätig sind: Tocht er Hannah Minghella in der Produktion und Sohn Max Minghella al s Schauspieler .1997 erhielt er für "Der englische Patient" den Oscar in der Rubrik "Beste Regie", 1999 eine Oscar-Nominierung in der Kategorie "Bestes adaptiertes Drehbuch" für "Der talenti erte Mr.Von 2003 bis 2007 war er Präsident des British Film Ins titute.Seit 1997 trägt das Anthony Minghella Theatre auf der Is le of Wight seinen Namen.

Za top 10 rečenica, rezultati su sljedeći:

Nach seinem Schulabschluss studierte er an der Universität Hull , wo er eine Zeit lang als Dozent tätig war.Auch als Produzent war er erfolgreich, darunter für die Filme "Der stille Amerikan er", "Die Dolmetscherin" und "Der Vorleser", für den er 2008 po sthum für den Oscar nominiert wurde.Gemeinsam mit seinem Freund und Kollegen Sydney Pollack gründete er die Produktionsfirma Mi rage Enterprises.Der Regisseur Minghella galt als ein guter Sch auspielerführer: Unter seiner Regie brachten es zahlreiche Dars teller zu Oscar-Nominierungen, zwei Schauspielerinnen erhielten

die Auszeichnung als "Beste Nebendarstellerin": Juliette Binoch e und Renée Zellweger .Gegen Ende seines Lebens kehrte Minghell a zu seinen Anfängen im Radio und auf der Bühne zurück: 2006 wu rde sein Hörspiel "Eyes Down Looking" mit Jude Law zu Ehren von Samuel Beckett auf BBC Radio 3 ausgestrahlt, ein Jahr zuvor hat te seine Inszenierung der Puccini-Oper Madame Butterfly in der English National Opera in London Premiere und wurde auch in der Nationaloper von Vilnius und in der Metropolitan Opera in New Y ork gezeigt.Am Ende des Films "Abbitte" von Joe Wright hat er e inen Kurzauftritt als Talkshow-Moderator neben Vanessa Redgrave .Der Ehe entstammen zwei Kinder, die in der Filmbranche tätig s ind: Tochter Hannah Minghella in der Produktion und Sohn Max Mi nghella als Schauspieler .1997 erhielt er für "Der englische Pa tient" den Oscar in der Rubrik "Beste Regie", 1999 eine Oscar-N ominierung in der Kategorie "Bestes adaptiertes Drehbuch" für " Der talentierte Mr.Von 2003 bis 2007 war er Präsident des Briti sh Film Institute.Seit 1997 trägt das Anthony Minghella Theatre auf der Isle of Wight seinen Namen.

8.8.3 Ispis TextRank rezultata

Ovdje su prikazani rezultati TexRank algoritma. Algoritam sličnosti koji je korišten je sličnost kosinusa rečenica. Rezultati prikazuju top 1, top 2, top 5 i top 10 rezultata TexRank algoritma.

Prvi tekst iz skupa podataka dana_test daje sljedeće rezultate za top 1 rečenice:

(<Sentence: Die Celsius-Skala der Temperatur wurde so definiert, dass die Temperatur in Grad Celsius gemessen gegenüber der Temperatur in Kelvin um 273,15 verschoben ist: Durch diese Festlegung wurde erreicht, dass die Differenz zwischen zwei Temperaturwerten in Kelvin und Grad Celsius gemessen zahlenmässig gleich gross sind und gleichwertig verwendet werden können.>,)

Prvi tekst iz skupa podataka dana_test daje sljedeće rezultate za top 2 rečenice:

(<Sentence: Die Celsius-Skala der Temperatur wurde so definiert, dass die Temperatur in Grad Celsius gemessen gegenüber der Temperatur in Kelvin um 273,15 verschoben ist: Durch diese Festlegung wurde erreicht, dass die Differenz zwischen zwei Temperaturwerten in Kelvin und Grad

```
Celsius gemessen zahlenmässig gleich gross sind und gleichwertig
verwendet werden können.>,
 <Sentence: In der zukünftigen Definition wird das Kelvin wohl durch die
Festlegung der Boltzmann-Konstante dadurch festgelegt, dass ein Kelvin
der Änderung der thermodynamischen Temperatur um die Energie entspricht,
die dem Zahlenwert von k entspricht.>)
```

Prvi tekst iz skupa podataka dana_test daje sljedeće rezultate za top 5 rečenica:

```
(<Sentence: Das Kelvin wurde durch die Generalkonferenz für Mass und
Gewicht zum ersten Mal 1954 – damals als "Grad Kelvin" – und in der
heute gültigen Form erneut 1968 definiert und als SI-Basiseinheit
festgelegt: 2007 wurde der Definition hinzugefügt, dass es sich um
Wasser mit der Isotopenzusammensetzung des Vienna Standard Mean Ocean
Water handeln soll.>,
 <Sentence: Die Celsius-Skala der Temperatur wurde so definiert, dass
die Temperatur in Grad Celsius gemessen gegenüber der Temperatur in
Kelvin um 273,15 verschoben ist: Durch diese Festlegung wurde erreicht,
dass die Differenz zwischen zwei Temperaturwerten in Kelvin und Grad
Celsius gemessen zahlenmässig gleich gross sind und gleichwertig
verwendet werden können.>,
 <Sentence: Wenn der Zahlenwert einer Temperatur formula_4 auf der
Kelvin-Skala formula_5-mal so gross ist wie der einer anderen
Temperatur formula_6, so ist der Energiegehalt bei formula_4 formula_5-
mal so hoch wie der bei formula_6 .>,
 <Sentence: In atomistischer Sicht kann man sagen, dass bei der Kelvin-
Skala die mittlere kinetische Energie der Teilchen proportional zur
Temperatur ist, das heisst eine doppelte kinetische Energie entspricht
einer doppelten Temperatur .>,
 <Sentence: In der zukünftigen Definition wird das Kelvin wohl durch die
Festlegung der Boltzmann-Konstante dadurch festgelegt, dass ein Kelvin
der Änderung der thermodynamischen Temperatur um die Energie entspricht,
die dem Zahlenwert von k entspricht.>)
```

Prvi tekst iz skupa podataka dana_test daje sljedeće rezultate za top 10 rečenica:

```
(<Sentence: Das Kelvin wurde durch die Generalkonferenz für Mass und
Gewicht zum ersten Mal 1954 – damals als "Grad Kelvin" – und in der
heute gültigen Form erneut 1968 definiert und als SI-Basiseinheit
festgelegt: 2007 wurde der Definition hinzugefügt, dass es sich um
Wasser mit der Isotopenzusammensetzung des Vienna Standard Mean Ocean
Water handeln soll.>,
 <Sentence: Die Celsius-Skala der Temperatur wurde so definiert, dass
die Temperatur in Grad Celsius gemessen gegenüber der Temperatur in
Kelvin um 273,15 verschoben ist: Durch diese Festlegung wurde erreicht,
dass die Differenz zwischen zwei Temperaturwerten in Kelvin und Grad
Celsius gemessen zahlenmässig gleich gross sind und gleichwertig
verwendet werden können.>,
 <Sentence: Gefrier- und Siedepunkt von Wasser bei Normalbedingungen
liegen mit dieser Definition weiterhin bei 0 °C und 100 °C , werden
aber nicht mehr als Fixpunkte für die Definition der Temperaturskala
verwendet.>,
 <Sentence: Wenn der Zahlenwert einer Temperatur formula_4 auf der
Kelvin-Skala formula_5-mal so gross ist wie der einer anderen
```

```
Temperatur formula_6, so ist der Energiegehalt bei formula_4 formula_5-
mal so hoch wie der bei formula_6 .>,
 <Sentence: In atomistischer Sicht kann man sagen, dass bei der Kelvin-
Skala die mittlere kinetische Energie der Teilchen proportional zur
Temperatur ist, das heisst eine doppelte kinetische Energie entspricht
einer doppelten Temperatur .>,
 <Sentence: Ein weiterer Zusammenhang leitet sich aus der Maxwell-
Boltzmann-Verteilung ab: eine Verdopplung der Temperatur auf der
Kelvin-Skala führt bei idealen Gasen zu einer Erhöhung der
Teilchengeschwindigkeit im quadratischen Mittel um den Faktor
formula_10.>,
 <Sentence: Die stetig verringerten Unsicherheiten bei der Messung der
Temperatur des Wassertripelpunktes machten es im 21. Jahrhundert
möglich, den Einfluss der Isotopenzusammensetzung auf den Tripelpunkt
des Wassers zu bestimmen .>,
 <Sentence: In der zukünftigen Definition wird das Kelvin wohl durch
die Festlegung der Boltzmann-Konstante dadurch festgelegt, dass ein
Kelvin der Änderung der thermodynamischen Temperatur um die Energie
entspricht, die dem Zahlenwert von k entspricht.>,
 <Sentence: Die Farbtemperatur gibt die spektrale
Strahldichteverteilung eines schwarzen Strahlers an, der die Temperatur
= Farbtemperatur hat.>,
 <Sentence: Die Wahrscheinlichkeit zur Überwindung der Barriere gibt die
Boltzmannverteilung an: wobei formula_13 die Boltzmannkonstante ist.>)
```

8.8.4 Rezultati SBKE na grafu sa Mihalcea sličnosti rečenica

Ovdje su prikazani rezultati vezani uz primjenu SBKE jedinice rangiranja rečenica na temelju grafa generiranog sa sličnostima između rečeneica na temelju Mihalcea sličnosti rečenica. Prikazani su rezultati za top 1, 2 5 i 10 rečenica.

Rezultat za top 1 rečenicu s pragom od 0,12:

```
1978 drehte er einen ersten Kurzfilm.
```

Rezultat za top 2 rečenice teksta s pragom 0,12:

```
1978 drehte er einen ersten Kurzfilm. Er entwickelte die Drehbü
cher für die 1988 erfolgreich ausgestrahlte Fernsehserie The St
oryteller von Muppets-Erfinder Jim Henson.
```

Rezultati za top 5 rečenica s pragom 0,12:

```
1978 drehte er einen ersten Kurzfilm. Er entwickelte die Drehbü
cher für die 1988 erfolgreich ausgestrahlte Fernsehserie The St
oryteller von Muppets-Erfinder Jim Henson. Seit 1981 war er als
Autor und Story Editor tätig. Gemeinsam mit seinem Freund und K
ollegen Sydney Pollack gründete er die Produktionsfirma Mirage
```

Enterprises. Nach seinem Schulabschluss studierte er an der Uni versität Hull, wo er eine Zeit lang als Dozent tätig war.

Rezultati za top 10 rečenica s pragom 0,12:

1978 drehte er einen ersten Kurzfilm. Er entwickelte die Drehbücher für die 1988 erfolgreich ausgestrahlte Fernsehserie The Storyteller von Mup pets-Erfinder Jim Henson. Seit 1981 war er als Autor und Story Editor t ätig. Gemeinsam mit seinem Freund und Kollegen Sydney Pollack gründete er die Produktionsfirma Mirage Enterprises. Nach seinem Schulabschluss studierte er an der Universität Hull, wo er eine Zeit lang als Dozent t ätig war. Minghella war der Sohn italienisch-schottischer Eltern, die a uf der Isle of Wight eine Fabrik für Eiscreme betrieben. Am Ende des Fi lms "Abbitte" von Joe Wright hat er einen Kurzauftritt als Talkshow-Mod erator neben Vanessa Redgrave. Er wurde mit Theaterstücken, Rundfunkhör spielen, der Fernsehserie "Inspector Morse" und vielen Drehbüchern für Film und Fernsehen bekannt. Seine letzte Arbeit als Drehbuchautor war d as Skript für den Musical-Film "Nine". Der Regisseur Minghella galt als ein guter Schauspielerführer: Unter seiner Regie brachten es zahlreiche Darsteller zu Oscar-Nominierungen, zwei Schauspielerinnen erhielten die Auszeichnung als "Beste Nebendarstellerin": Juliette Binoche und Renée Zellweger.

8.8.5 Ispis primjera algoritma sažimanja KL divergencije – sumy

Ovdje se prikazuju ispisi generiranih sažetaka algoritma na temelju top 1, 2, 5 i 10 rečenica unutar prvog teksta skupa podataka.

Top 1 rečnica:

Gegen Ende seines Lebens kehrte Minghella zu seinen AnfaÌˆngen im Radio und auf der BuÌˆhne zuruÌˆck: 2006 wurde sein HoÌˆrspiel "Eyes Down Looking" mit Jude Law zu Ehren von Samuel Beckett auf BBC Radio 3 ausgestrahlt, ein Jahr zuvor hatte seine Inszenierung der Puccini-Oper Madame Butterfly in der English National Opera in London Premiere und wurde auch in der Nationaloper von Vilnius und in der Metropolitan Opera in New York gezeigt.

Top 2 rečenice:

Gegen Ende seines Lebens kehrte Minghella zu seinen AnfaÌˆngen im Radio und auf der BuÌˆhne zuruÌˆck: 2006 wurde sein HoÌˆrspiel "Eyes Down Looking" mit Jude Law zu Ehren von Samuel Beckett auf BBC Radio 3 ausgestrahlt, ein Jahr zuvor hatte seine Inszenierung der Puccini-Oper Madame Butterfly in der English National Opera in London Premiere und wurde auch in der Nationaloper von Vilnius und in der Metropolitan Opera in New York gezeigt. Minghella

starb im Alter von 54 Jahren in einem Londoner Krankenhaus an inneren Blutungen infolge der Operation eines Tonsillenkarzinoms und eines Karzinoms im Nacken.

Top 5 rečenica:

Er wurde mit TheaterstuÌˆcken, RundfunkhoÌˆrspielen, der Fernsehserie "Inspector Morse" und vielen DrehbuÌˆchern fuÌˆr Film und Fernsehen bekannt. Gegen Ende seines Lebens kehrte Minghella zu seinen AnfaÌˆngen im Radio und auf der BuÌˆhne zuruÌˆck: 2006 wurde sein HoÌˆrspiel "Eyes Down Looking" mit Jude Law zu Ehren von Samuel Beckett auf BBC Radio 3 ausgestrahlt, ein Jahr zuvor hatte seine Inszenierung der Puccini-Oper Madame Butterfly in der English National Opera in London Premiere und wurde auch in der Nationaloper von Vilnius und in der Metropolitan Opera in New York gezeigt. Minghella starb im Alter von 54 Jahren in einem Londoner Krankenhaus an inneren Blutungen infolge der Operation eines Tonsillenkarzinoms und eines Karzinoms im Nacken. 2001 wurde Minghella zum Commander of the British Empire ernannt. Seit 1997 traÌˆgt das Anthony Minghella Theatre auf der Isle of Wight seinen Namen.

Top 10 rečenica:

1978 drehte er einen ersten Kurzfilm. Er wurde mit TheaterstuÌˆcken, RundfunkhoÌˆrspielen, der Fernsehserie "Inspector Morse" und vielen DrehbuÌˆchern fuÌˆr Film und Fernsehen bekannt. Gegen Ende seines Lebens kehrte Minghella zu seinen AnfaÌˆngen im Radio und auf der BuÌˆhne zuruÌˆck: 2006 wurde sein HoÌˆrspiel "Eyes Down Looking" mit Jude Law zu Ehren von Samuel Beckett auf BBC Radio 3 ausgestrahlt, ein Jahr zuvor hatte seine Inszenierung der Puccini-Oper Madame Butterfly in der English National Opera in London Premiere und wurde auch in der Nationaloper von Vilnius und in der Metropolitan Opera in New York gezeigt. Seine letzte Arbeit als Drehbuchautor war das Skript fuÌˆr den Musical-Film "Nine" . Minghella starb im Alter

von 54 Jahren in einem Londoner Krankenhaus an inneren Blutungen infolge der Operation eines Tonsillenkarzinoms und eines Karzinoms im Nacken. 1997 erhielt er fuÌˆr "Der englische Patient" den Oscar in der Rubrik "Beste Regie", 1999 eine Oscar-Nominierung in der Kategorie "Bestes adaptiertes Drehbuch" fuÌˆr "Der talentierte Mr. Ripley", bei dem er auch Regie fuÌˆhrte. 2001 wurde Minghella zum Commander of the British Empire ernannt. Von 2003 bis 2007 war er PraÌˆsident des British Film Institute. Seit 1997 traÌˆgt das Anthony Minghella Theatre auf der Isle of Wight seinen Namen.

8.9 Računanje značajki i rezultati klasifikacije

Ovdje se prikazuju rezultati svih navedenih klasifikatora te rezultati pripreme i računanja značajki. Svaki od navedenih rezultata klasifikacije biti će prikazani pod odgovarajućim naslovom, kao i rezultati računanja značajki.

8.9.1 Rezultati određivanja i računanja značajki

Ovdje se prikazuju rezultati dobiveni tijekom procesa računanja i stvaranja značajki:

- Broj riječi u rečenicama koji započinju velikim slovom
- Broj riječi u rečenicama koji su ključne riječi
- Broj rečenica koje se nalaze unutar source i summary stupca okvira podataka

Broj riječi u rečenicama koji započinju velikim početnim slovom predstavljen je stupcem noCap_LetterWords_inSentence, dok top 10 ključnih riječi svakog teksta predstavljen stupcem keywords.

Finalni izgled skupa podataka sa svim izračunanim značajkama te oznakama izgleda ovako:

	source1	no_words_inSent_SS	no_words_inSent_SK	noCap_LetterWords_inSentence	sim_sent
0	Minghella Sohn italienisch-schottischer Eltern...	1	0	7	No
1	Nach Schulabschluss studierte Universität Hul...	0	0	6	No
2	1978 drehte ersten Kurzfilm .	0	0	1	No
3	Seit 1981 Autor Story Editor tätig .	0	0	4	No
4	Er wurde Theaterstücken , Rundfunkhörspielen...	0	0	9	Yes
...	...	...	...	...	...
3513930	Carl Sigmans Text erschien 1960er Jahren briti...	0	0	7	No
3513931	Camillo Felgen alias Heinz Helmer verfasste de...	5	1	12	Yes
3513932	Zahlreiche Interpreten sangen 1964 Lied franzo...	1	1	11	Yes
3513933	Von Letzterer stammt spanischsprachige Fassung...	2	1	6	Yes
3513934	In 1970er Jahren nahmen mehrere prominente Sä...	5	1	13	Yes

Slika 0-4: Prikaz finalnog izgleda skupa podataka prije klasifikacije rečenica sa svim izračunanim vrijednostima značajki i oznaka klasa.

8.10 Rezultati algoritama sličnosti rečenica

8.10.1 Rezultati algoritma sličnosti kosinusa rečenica

Matrica sličnosti kosinusa rečenica prvog teksta je sljedeća:

```
[[1.         0.09744106 0.         0.03869878 0.04781903 0.07081717
  0.16238519 0.02678708 0.05236823 0.10528038 0.         0.06032723
  0.11526368 0.08639868 0.18210037 0.12524267 0.03669045 0.0504767
  0.15099939 0.03452039 0.09164033 0.03421487 0.29627416]
 [0.09744106 1.         0.05666065 0.24378722 0.04506231 0.03343207
  0.11653198 0.08641943 0.04296718 0.02636967 0.0545237  0.05934736
  0.128356   0.04462207 0.13541677 0.01736579 0.05336891 0.04294381
  0.09585254 0.05265513 0.         0.07369767 0.01424906]
 [0.         0.05666065 1.         0.03914789 0.02717718 0.02776222
  0.04601168 0.03041633 0.         0.         0.11580107 0.
  0.         0.         0.         0.         0.         0.
  0.02175773 0.04372518 0.         0.03461194 0.         ]
 [0.03869878 0.24378722 0.03914789 1.         0.07260379 0.02309883
  0.11504768 0.05328216 0.06535386 0.0320622  0.05297722 0.08200837
  0.         0.07112823 0.16326919 0.03519102 0.02007082 0.04652881
  0.01810295 0.0363804  0.         0.07304019 0.08640722]
 [0.04781903 0.04506231 0.02717718 0.07260379 1.         0.14179973
  0.15548369 0.09684431 0.03500175 0.11615417 0.01552663 0.07732742
  0.0434896  0.0984333  0.06407051 0.06551943 0.0373683  0.02337285
  0.08129923 0.05624395 0.04950831 0.07434519 0.01366907]
 [0.07081717 0.03343207 0.02776222 0.02309883 0.14179973 1.
  0.19633362 0.0670742  0.03301445 0.02723821 0.04736631 0.0245635
  0.07189798 0.         0.04006099 0.06179941 0.02557652 0.01478983
  0.04434119 0.0574547  0.0726898  0.06098882 0.         ]
 [0.16238519 0.11653198 0.04601168 0.11504768 0.15548369 0.19633362
  1.         0.08689474 0.10800343 0.10326853 0.04427635 0.17516607
  0.17880369 0.06565365 0.11805139 0.10009616 0.02788078 0.15853638
  0.26385776 0.17957939 0.04190945 0.05984672 0.0231421 ]
 [0.02678708 0.08641943 0.03041633 0.05328216 0.09684431 0.0670742
  0.08689474 1.         0.03269197 0.04219931 0.0173772  0.
  0.06202162 0.07127133 0.03966969 0.06119578 0.01559422 0.
  0.01406527 0.02826611 0.         0.02237486 0.         ]
 [0.05236823 0.04296718 0.         0.06535386 0.03500175 0.03301445
  0.10800343 0.03269197 1.         0.0854937  0.03196651 0.04034858
```

```
 0.04936601 0.04167783 0.13998556 0.0904027  0.0281052  0.06508864
 0.1368595  0.05164323 0.01788465 0.         0.02535727]
[0.10528038 0.02636967 0.         0.0320622  0.11615417 0.02723821
 0.10326853 0.04219931 0.0854937  1.         0.04833378 0.02771843
 0.08830424 0.08385285 0.22479035 0.08784184 0.15473375 0.12476986
 0.18774301 0.0293585  0.0525034  0.03395865 0.10103177]
[0.         0.0545237  0.11580107 0.05297722 0.01552663 0.04736631
 0.04427635 0.0173772  0.03196651 0.04833378 1.         0.02653114
 0.         0.         0.01939469 0.         0.02476468 0.03194912
 0.01243044 0.02498069 0.         0.12446431 0.         ]
[0.06032723 0.05934736 0.         0.08200837 0.07732742 0.0245635
 0.17516607 0.         0.04034858 0.02771843 0.02653114 1.
 0.07515809 0.03585095 0.02448025 0.         0.         0.11692764
 0.06966747 0.03868724 0.         0.10620245 0.07488936]
[0.11526368 0.128356   0.         0.         0.0434896  0.07189798
 0.17880369 0.06202162 0.04936601 0.08830424 0.         0.07515809
 1.         0.01216897 0.04491629 0.04051325 0.00820607 0.07317936
 0.13253954 0.0535277  0.         0.         0.05438647]
[0.08639868 0.04462207 0.         0.07112823 0.0984333  0.
 0.06565365 0.07127133 0.04167783 0.08385285 0.         0.03585095
 0.01216897 1.         0.09514838 0.10710629 0.04449576 0.02612319
 0.0451199  0.         0.02831474 0.03657196 0.04014528]
[0.18210037 0.13541677 0.         0.16326919 0.06407051 0.04006099
 0.11805139 0.03966969 0.13998556 0.22479035 0.01939469 0.02448025
 0.04491629 0.09514838 1.         0.25141751 0.10986343 0.12042466
 0.19487762 0.         0.04340384 0.         0.07401393]
[0.12524267 0.01736579 0.         0.03519102 0.06551943 0.06179941
 0.10009616 0.06119578 0.0904027  0.08784184 0.         0.
 0.04051325 0.10710629 0.25141751 1.         0.07222649 0.04925124
 0.05334779 0.         0.06695621 0.         0.07568772]
[0.03669045 0.05336891 0.         0.02007082 0.0373683  0.02557652
 0.02788078 0.01559422 0.0281052  0.15473375 0.02476468 0.
 0.00820607 0.04449576 0.10986343 0.07222649 1.         0.06777942
 0.07051586 0.         0.01909387 0.03188698 0.02707173]
[0.0504767  0.04294381 0.         0.04652881 0.02337285 0.01478983
 0.15853638 0.         0.06508864 0.12476986 0.03194912 0.11692764
 0.07317936 0.02612319 0.12042466 0.04925124 0.06777942 1.
 0.18922601 0.02329382 0.01787492 0.         0.02534348]
[0.15099939 0.09585254 0.02175773 0.01810295 0.08129923 0.04434119
 0.26385776 0.01406527 0.1368595  0.18774301 0.01243044 0.06966747
 0.13253954 0.0451199  0.19487762 0.05334779 0.07051586 0.18922601
 1.         0.12480858 0.         0.01600542 0.09179683]
[0.03452039 0.05265513 0.04372518 0.0363804  0.05624395 0.0574547
 0.17957939 0.02826611 0.05164323 0.0293585  0.02498069 0.03868724
 0.0535277  0.         0.         0.         0.         0.02329382
 0.12480858 1.         0.         0.03216512 0.         ]
[0.09164033 0.         0.         0.         0.04950831 0.0726898
 0.04190945 0.         0.01788465 0.0525034  0.         0.
 0.         0.02831474 0.04340384 0.06695621 0.01909387 0.01787492
 0.         0.         1.         0.09062445 0.09502983]
[0.03421487 0.07369767 0.03461194 0.07304019 0.07434519 0.06098882
 0.05984672 0.02237486 0.         0.03395865 0.12446431 0.10620245
 0.         0.03657196 0.         0.         0.03188698 0.
 0.01600542 0.03216512 0.09062445 1.         0.         ]
[0.29627416 0.01424906 0.         0.08640722 0.01366907 0.
 0.0231421  0.         0.02535727 0.10103177 0.         0.07488936
 0.05438647 0.04014528 0.07401393 0.07568772 0.02707173 0.02534348
 0.09179683 0.         0.09502983 0.         1.        ]]
```

8.10.2 Rezultati algoritma Jaccardove sličnosti

Ovdje su prikazani rezultati algoritma Jaccardove sličnosti rečenica prvog teksta. Prilaže se rezultat sličnosti među rečenicama u kojemu su vidljivi svi međusobni odnosi svake rečenice sa svakom rečenicom unutar prvog teksta okvira podataka. Uzima se prvih nekoliko rezultata radi jednostavnijeg uvida:

```
jaccard distance 0.0
       al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_0   0   0    0     0   1          1   1      0    0      0
doc_0   0   0    0     0   1          1   1      0    0      0

jaccard distance 1.0
       al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_1   1   0    0     0   0          0   0      0    1      0
doc_0   0   0    0     0   1          1   1      0    0      0

jaccard distance 1.0
       al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_2   0   0    0     0   0          0   0      0    0      0
doc_0   0   0    0     0   1          1   1      0    0      0

jaccard distance 1.0
       al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_3   1   0    0     0   0          0   0      0    1      0
doc_0   0   0    0     0   1          1   1      0    0      0

jaccard distance 0.8
       al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_4   0   0    0     1   1          0   0      0    0      1
doc_0   0   0    0     0   1          1   1      0    0      0

jaccard distance 0.6666666666666666
       al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_5   0   0    0     0   1          0   0      0    0      0
doc_0   0   0    0     0   1          1   1      0    0      0

jaccard distance 0.8571428571428571
       al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_6   1   0    2     0   3          0   0      1    0      1
doc_0   0   0    0     0   1          1   1      0    0      0

jaccard distance 1.0
       al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_7   0   0    0     0   0          0   0      0    0      0
```

```
doc_0   0   0    0     0   1          1   1      0    0      0

jaccard distance 0.8333333333333334
       al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_8   2   0    1     0   0          1   0      1    0      0
doc_0   0   0    0     0   1          1   1      0    0      0

jaccard distance 0.8
       al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_9   0   1    0     0   0          1   0      0    0      2
doc_0   0   0    0     0   1          1   1      0    0      0

jaccard distance 1.0
        al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_10   1   0    0     0   0          0   0      0    0      0
doc_0    0   0    0     0   1          1   1      0    0      0

jaccard distance 0.8
        al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_11   1   0    0     1   1          0   0      0    0      0
doc_0    0   0    0     0   1          1   1      0    0      0

jaccard distance 0.6666666666666666
        al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_12   0   0    0     0   2          0   0      0    0      0
doc_0    0   0    0     0   1          1   1      0    0      0

jaccard distance 0.6666666666666666
        al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_13   0   0    0     0   0          1   0      0    0      0
doc_0    0   0    0     0   1          1   1      0    0      0

jaccard distance 0.8333333333333334
        al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_14   1   0    1     0   0          2   0      0    1      0
doc_0    0   0    0     0   1          1   1      0    0      0

jaccard distance 0.6666666666666666
        al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_15   0   0    0     0   0          2   0      0    0      0
doc_0    0   0    0     0   1          1   1      0    0      0

jaccard distance 0.6666666666666666
        al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_16   0   0    0     0   0          1   0      0    0      0
doc_0    0   0    0     0   1          1   1      0    0      0

jaccard distance 0.6
```

```
        al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_17   2   2    0     0   1          1   0      0    0      0
doc_0    0   0    0     0   1          1   1      0    0      0

jaccard distance 0.8
        al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_18   0   0    2     0   2          0   0      2    0      0
doc_0    0   0    0     0   1          1   1      0    0      0

jaccard distance 0.6666666666666666
        al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_19   0   0    0     0   1          0   0      0    0      0
doc_0    0   0    0     0   1          1   1      0    0      0

jaccard distance 0.5
        al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_20   0   0    0     0   0          1   1      0    0      1
doc_0    0   0    0     0   1          1   1      0    0      0

jaccard distance 1.0
        al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_21   0   0    0     1   0          0   0      0    0      0
doc_0    0   0    0     0   1          1   1      0    0      0

jaccard distance 0.3333333333333333
        al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_22   0   0    0     0   0          1   1      0    0      0
doc_0    0   0    0     0   1          1   1      0    0      0

jaccard distance 1.0
       al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_0   0   0    0     0   1          1   1      0    0      0
doc_1   1   0    0     0   0          0   0      0    1      0

jaccard distance 0.0
       al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_1   1   0    0     0   0          0   0      0    1      0
doc_1   1   0    0     0   0          0   0      0    1      0

jaccard distance 1.0
       al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_2   0   0    0     0   0          0   0      0    0      0
doc_1   1   0    0     0   0          0   0      0    1      0

jaccard distance 0.0
       al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_3   1   0    0     0   0          0   0      0    1      0
doc_1   1   0    0     0   0          0   0      0    1      0
```

```
jaccard distance 1.0
       al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_4   0   0    0     1   1          0   0      0    0      1
doc_1   1   0    0     0   0          0   0      0    1      0

jaccard distance 1.0
       al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_5   0   0    0     0   1          0   0      0    0      0
doc_1   1   0    0     0   0          0   0      0    1      0

jaccard distance 0.8333333333333334
       al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_6   1   0    2     0   3          0   0      1    0      1
doc_1   1   0    0     0   0          0   0      0    1      0

jaccard distance 1.0
       al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_7   0   0    0     0   0          0   0      0    0      0
doc_1   1   0    0     0   0          0   0      0    1      0

jaccard distance 0.8
       al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_8   2   0    1     0   0          1   0      1    0      0
doc_1   1   0    0     0   0          0   0      0    1      0

jaccard distance 1.0
       al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_9   0   1    0     0   0          1   0      0    0      2
doc_1   1   0    0     0   0          0   0      0    1      0

jaccard distance 0.5
        al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_10   1   0    0     0   0          0   0      0    0      0
doc_1    1   0    0     0   0          0   0      0    1      0

jaccard distance 0.75
        al  ck  der  film  fu  minghella  of  oscar  tig  wurde
doc_11   1   0    0     1   1          0   0      0    0      0
doc_1    1   0    0     0   0          0   0      0    1      0
```

8.10.3 Rezultati algoritma Mihalcea sličnosti

Rečenice prvog teksta su predstavljene redom brojevima od 0 do n koliko ih je sadržano u tekstu. Udaljenost ovdje je reprezentirana predstavljena uređenim trojkama pri čemu je prvi element uređene trojke prva rečenica koja se uspoređuje, drugi element je druga rečenica koja

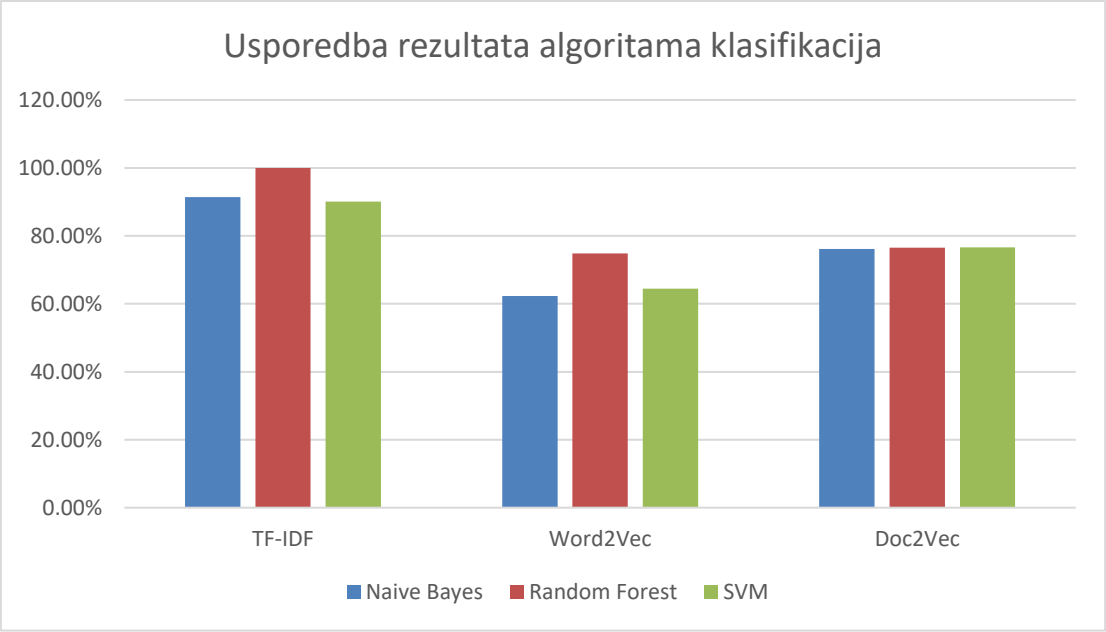
se uspoređuje te treći element je sličnost između dvije rečenice. Kao rezultat pokretanja mihalcea() funkcije dobiva se sljedeća lista kao izlaz:

```
[(0, 1, 0.8417265744165442), (0, 2, 0.2044843780639489), (0, 3, 0.37436
818919827736), (0, 4, 0.6627474617756755), (0, 5, 0.5192420436830061),
(0, 6, 0.7596446222083343), (0, 7, 0.35382641809938753), (0, 8, 0.61274
90480907971), (0, 9, 0.687268944356234), (0, 10, 0.16834531488330884),
(0, 11, 0.5307396271490813), (0, 12, 0.8012466314080338), (0, 13, 0.874
5803345976697), (0, 14, 0.9673463349582323), (0, 15, 0.5525527761579856
), (0, 16, 0.48675105773213695), (0, 17, 0.7966041099021299), (0, 18, 0
.7659363101134964), (0, 19, 0.3811684437682579), (0, 20, 0.561552283797
4161), (0, 21, 0.368368517438657), (0, 22, 1.2383924633478565), (1, 2,
0.404723739388645), (1, 3, 0.9270198872707541), (1, 4, 0.49287186311260
73), (1, 5, 0.17155761360662464), (1, 6, 0.7537705373699886), (1, 7, 0.
5259667621144383), (1, 8, 0.45597894096575103), (1, 9, 0.40947438480196
274), (1, 10, 0.5007123010430011), (1, 11, 0.5259667621144383), (1, 12,
0.6357714444954519), (1, 13, 0.6934426851274016), (1, 14, 0.79951021216
71913), (1, 15, 0.3649209568399075), (1, 16, 0.48273357748200746), (1,
17, 0.6321175663898954), (1, 18, 0.7599649016095851), (1, 19, 0.5662174
974532644), (1, 20, 0.18540397745415083), (1, 21, 0.5473814352598613),
(1, 22, 0.35064450807629216), (2, 3, 0.4604257801093024), (2, 4, 0.1985
327397206032), (2, 5, 0.209243389805969), (2, 6, 0.3581984995413015), (
2, 7, 0.4297415210966088), (2, 8, 0.18085049725270286), (2, 9, 0.159320
821980426), (2, 10, 0.6070856090829674), (2, 11, 0.2148707605483044), (
2, 12, 0.19077608994233797), (2, 13, 0.2119318086698044), (2, 14, 0.192
1596300948879), (2, 15, 0.22569202144874392), (2, 16, 0.193618865340964
7), (2, 17, 0.1894615908449756), (2, 18, 0.3617009945054057), (2, 19, 0
.47075491110477363), (2, 20, 0.2302128900546512), (2, 21, 0.45138404289
748785), (2, 22, 0.2148707605483044), (3, 4, 0.36436922766846275), (3,
5, 0.19116406736002547), (3, 6, 0.8284343272195943), (3, 7, 0.587550340
1543793), (3, 8, 0.5015536282966785), (3, 9, 0.297237388925098), (3, 10
, 0.5562119323624525), (3, 11, 0.5875503401543793), (3, 12, 0.175631725
4872274), (3, 13, 0.5802164754554788), (3, 14, 0.7072146024589432), (3,
15, 0.4096008872996888), (3, 16, 0.3560764655036198), (3, 17, 0.34903406
14910062), (3, 18, 0.334369085531119), (3, 19, 0.4254884276413387), (3,
20, 0.20851619571212315), (3, 21, 0.6144013309495321), (3, 22, 0.391700
2267695862), (4, 5, 0.6751902917064592), (4, 6, 1.0403308612378428), (4
, 7, 0.5173223086360709), (4, 8, 0.7491128939396555), (4, 9, 1.07790914
4985287), (4, 10, 0.4928718631126073), (4, 11, 0.6897630781814278), (4,
12, 0.9394259230498108), (4, 13, 0.8527139327602735), (4, 14, 0.6300066
836512455), (4, 15, 0.538024997091025), (4, 16, 0.4754419736864149), (4
, 17, 0.9341070075716525), (4, 18, 0.8989354727275867), (4, 19, 0.55621
19323624524), (4, 20, 0.3643692276684 6275), (4, 21, 0.35868333139401665
), (4, 22, 0.3448815390907139), (5, 6, 0.6181616348430732), (5, 7, 0.36
09285109255383), (5, 8, 0.3116851244442047), (5, 9, 0.2791757236791252)
, (5, 10, 0.3431152272132493), (5, 11, 0.3609285109255383), (5, 12, 0.4
8947225054146926), (5, 13, 0.178386605194704), (5, 14, 0.32833660390264
29), (5, 15, 0.18803637397379358), (5, 16, 0.33046439736370276), (5, 17
, 0.3243900165870136), (5, 18, 0.3116851244442047), (5, 19, 0.194711691
32843705), (5, 20, 0.19116406736002547), (5, 21, 0.18803637397379358),
(5, 22, 0.18046425546276915), (6, 7, 0.6303544313840402), (6, 8, 1.1078
636706192722), (6, 9, 0.752981855206788), (6, 10, 0.7537705373699886),
(6, 11, 1.1031202549220704), (6, 12, 1.0096023239923788), (6, 13, 0.624
0079363908237), (6, 14, 0.8701096414949027), (6, 15, 0.4899964921703108
), (6, 16, 0.29169563026728534), (6, 17, 1.004334323874154), (6, 18, 1.
2463466294466814), (6, 19, 0.6733812595332352), (6, 20, 0.3313737308878
377), (6, 21, 0.4899964921703108), (6, 22, 0.15758860784601006), (7, 8,
```

```
0.4768285833715889), (7, 9, 0.42621000949379806), (7, 10, 0.35064450807
629216), (7, 11, 0.18463468653442752), (7, 12, 0.4996762874565183), (7,
13, 0.5473814352598613), (7, 14, 0.5028371126588549), (7, 15, 0.3851370
3414599854), (7, 16, 0.3374430507882949), (7, 17, 0.1655559322908945),
(7, 18, 0.31788572224772593), (7, 19, 0.39915098238271013), (7, 20, 0.1
958501133847931), (7, 21, 0.38513703414599854), (7, 22, 0.1846346865344
2752), (8, 9, 1.1371889198141762), (8, 10, 0.6079719212876681), (8, 11,
0.6357714444954519), (8, 12, 0.7268122397097266), (8, 13, 0.62931596088
82498), (8, 14, 1.3154787828902732), (8, 15, 0.4943621630314899), (8, 1
6, 0.44102115569280914), (8, 17, 0.8675881915416176), (8, 18, 0.9766935
964646827), (8, 19, 0.6795666503051123), (8, 20, 0.334369085531119), (8
, 21, 0.1647873876771633), (8, 22, 0.31788572224772593), (9, 10, 0.6824
573080032712), (9, 11, 0.42621000949379806), (9, 12, 0.9178437765151833
), (9, 13, 0.8446752022765888), (9, 14, 0.9224083438751282), (9, 15, 0.
5868854584199328), (9, 16, 0.9272003155221238), (9, 17, 1.0439860215645
835), (9, 18, 0.7581259465427842), (9, 19, 0.6030164298959908), (9, 20,
0.44585608338764693), (9, 21, 0.1467213646049832), (9, 22, 0.7103500158
229968), (10, 11, 0.7012890161525843), (10, 12, 0.4768285833715889), (1
0, 13, 0.1733606712818504), (10, 14, 0.3198040848668765), (10, 15, 0.18
246047841995375), (10, 16, 0.321822384988005), (10, 17, 0.6321175663898
954), (10, 18, 0.6079719212876681), (10, 19, 0.5662174974532644), (10,
20, 0.18540397745415083), (10, 21, 0.5473814352598613), (10, 22, 0.1753
2225403814608), (11, 12, 0.8327938124275306), (11, 13, 0.36492095683990
75), (11, 14, 0.33522474177256995), (11, 15, 0.19256851707299927), (11,
16, 0.16872152539414745), (11, 17, 0.993335593745367), (11, 18, 0.79471
43056193149), (11, 19, 0.39915098238271013), (11, 20, 0.195850113384793
1), (11, 21, 0.38513703414599854), (11, 22, 0.36926937306885504), (12,
13, 0.4943621630314899), (12, 14, 0.6103205134100028), (12, 15, 0.34597
62562611936), (12, 16, 0.3069974218206291), (12, 17, 1.0561189350985716
), (12, 18, 1.0175371355936174), (12, 19, 0.535865288820912), (12, 20,
0.1756317254872274), (12, 21, 0.1729881281305968), (12, 22, 0.499676287
4565183), (13, 14, 0.829093176682112), (13, 15, 0.76081854893906), (13,
16, 0.6676164013906681), (13, 17, 0.6552228466577641), (13, 18, 0.47198
697066618733), (13, 19, 0.3940751731490904), (13, 20, 0.386810983636985
85), (13, 21, 0.38040927446953), (13, 22, 0.5473814352598613), (14, 15,
0.8706246255640889), (14, 16, 0.7719656221860525), (14, 17, 0.910428818
975089), (14, 18, 0.7308215460501518), (14, 19, 0.3596681365184985), (1
4, 20, 0.3536073012294716), (14, 21, 0.17412492511128178), (14, 22, 0.50
28371126588549), (15, 16, 0.7012890161525843), (15, 17, 0.5157199024573
869), (15, 18, 0.3295747753543266), (15, 19, 0.20887764485602905), (15,
20, 0.409600887299688), (15, 21, 0.20121480219092233), (15, 22, 0.57770
55512189978), (16, 17, 0.7632328935509646), (16, 18, 0.4410211556928091
4), (16, 19, 0.18111148749870565), (16, 20, 0.3560764655036198), (16, 2
1, 0.17532225403814608), (16, 22, 0.5061645761824424), (17, 18, 1.30138
22873124263), (17, 19, 0.5324067465292174), (17, 20, 0.3490340614910062
), (17, 21, 0.17190663415246227), (17, 22, 0.4966677968726835), (18, 19
, 0.8494583128813904), (18, 20, 0.1671845427655595), (18, 21, 0.3295747
753543266), (18, 22, 0.4768285833715889), (19, 20, 0.21274421382066935)
, (19, 21, 0.4177552897120581), (19, 22, 0.19957549119135506), (20, 21,
0.409600887299688), (20, 22, 0.5875503401543793), (21, 22, 0.1925685170
7299927)]
```

8.11 Usporedni prikaz rezultata algoritama klasifikacije

Ovdje je moguće usporediti rezultate algoritama klasifikacija u odnosu na različite načine vektorizacije rečenica.



Slika 0-5: Prikaz rezultata između svih primijenjenih algoritama klasifikacije na temelju TF-IDF, Word2Vec i Doc2Vec vektorima rečenica.

9 Bibliografija

9.1 Knjige i znanstveni članci

Aggarwal, Charu C., i ChengXiang Zhai. *Mining Text Data.* Uredio Charu C. Aggarwal i ChengXiang Zhai. Springer, 2012.

Allahyari, Mehdi, i dr. »Text Summarization Techniques: A Brief Survey.« *International Journal of Advanced Computer Science and Applications*, 1. 1 2017.

Baayen, R. Harald. *Word Frequency Distribution.* Uredio Harald Baayen, i dr. 18 svez. Dordrecht: Kluwer Academic Publishers, 2001.

Beli, Dorian. »Usporedba jezičnih alata za njemački jezik, završni rad.« Rijeka: Odijel za Informatiku, Sveučilište u Rijeci, 2018.

Beliga, Slobodan, Sanda Martinčić-Ipšić, i Ana Meštrović. »Selectivity-Based Keyword.« U *International Journal on Semantic Web and Information Systems*, uredio Miltiadis D. Lytras, 1-26. IGI Global, 2016.

Bishop, Christpher M. *Pattern Recognition and Machine Learning.* Uredio Bernhard Schölkopf, Michael Jordan i Jon Kleinberg. New York: Springer, 2021.

Carenini, Giuseppe, Gabriel Murray, i Raymond Ng. *Methods for Mining and SummarizingText Conversations; Synthesis Lectures on Data Management.* Uredio Tamer T. Özsu. Morgan & Claypool Publishers, 2011.

Chen, Vincent, Torres Eduardo Montaño, i Liezl Puzon. »An Examination of the CNN/DailyMail.« 2017.

Connor, Richard. »A Tale of Four Metrics.« *Lecture Notes in Computer Science*, 2016: 210-217.

Corley, Courtney, i Rada Mihalcea. »Measuring the Semantic Similarity of Texts.« *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailmen*, lipanj 2005: 13-18.

Dunning, T. »Accurate methods for the statistics of surprise and coincidence.« *Computational Linguistics*, 1994: 61-74.

Erkan, G., i D. R. Radev. »LexRank: Graph-based centrality as salience in text summarization.« *Journal of Artificial Intelligence Research*, 2004.

Ermakova, Liana, Valère Jean Cossu, i Josiane Mothe. »A survey on evaluation of summarization methods.« *Information Processing and Management*, 2019: 1749-1814.

Gagniuc, Paul. *Markov Chains: From Theory to Implementation and Experimentation.* New York: John Wiley & Sons, 2017.

Han, Jiawei, Micheline Kamber, i Jian Pei. *Data Mining: Concepts and Techniques.* Uredio Morgan Kaufmann. Svez. 3. Waltham: Elsevier, 2011.

Herings, P. Jean-Jacques, Gerard van der Laan, i Dolf Talman. »Measuring the Power of Nodes in Digraphs.« *SSRN Electronic Journal*, 3. listopad 2001.

Jezek, Karel , i Josef Steinberger. »Evaluation Measures for Text Summarization.« *Computing and Informatics*, 2009: 1001-1026.

Jiang, Jay J., i David W. Conrath. »Semantic similarity based on.« *Proceedings of the 10th Research on Computational Linguistics International Conference*, kolovoz 1998: 19-33.

Jing, Hongyan, Regina Barzilay, Kathleen McKeown, i Michael Elhadad. »Summarization Evaluation Methods: Experiments and Analysis.« *AAAI Technical Report SS-98-06*, 2000: 51-59.

Kleinberg, Jon M. »Authoritative Sources in a Hyperlinked Environment.« *Journal of the ACM*, rujan 1999: 604-632.

Kullback, Solomon. *Information Theory and Statistics.* John Wiley & Sons, 1959.

Kullback, Solomon, i Richard Arthur Leibler. »On Information and Sufficiency.« *Annals of Mathematical Statistics*, ožujak 1951, 22. izd.: 79-86.

Lealcock, C., i M. Chodorow. »Combining local context and wordnet similarity for word sense identification.« *WordNet, An Electronic Lexical Database*, siječanj 1998: 147-165.

Lesk, Michael E. »Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone.« *Proceedings of the SIGDOC Conference 1986*, siječanj 1986: 24-26.

Levandowsky, Michael, i David Winter. »Distance between Sets.« *Nature*, 1971: 34-35.

Lin, Chin-Yew, i Franz Josef Och. »Automatic Evaluation of Machine Translation Quality Using Longest Common Subsequence and Skip-Bigram Statistics.« *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004)*, 2004.

Lin, Dekang. »An information-theoretic definition of similarity.« *Proceedings of the 15th International Conference on Machine Learning*, lipanj 1998: 296-304.

Lipkus, Alan H. »A proof of the triangle inequality for the Tanimoto distance.« *Journal of Mathematical Chemistry*, 1999: 263-265.

MacKay, David J.C. »2.6 Gibbs' inequality.« U *Information Theory, Inference, and Learning Algorithms*, autor David J.C. MacKay, 30-35. Cambridge University Press, 2003.

Malik, Usman. *Random Forest Algorithm with Python and Scikit-Learn.* 25.. siječanj 2019. https://stackabuse.com/random-forest-algorithm-with-python-and-scikit-learn/ (pokušaj pristupa 8.. travnja 2021.).

Mihalcea, Rada. »Graph-based ranking algorithms for sentence extraction, applied to text summarization.« *ACLdemo '04: Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, 2004: 20-25.

Mihalcea, Rada, Courtney Corley, i Carlo Strapparava. »Corpus-based and knowledge-based measures of text semantic similarity.« *Proceedings of the 21st national conference on Artificial intelligence*, 2006: 775-780.

Mihalcea, Rada, i Paul Tarau. »TextRank: Bringing Order into Texts.« *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 2004: 404-411.

Moulton, Ryan, i Yunjiang Jiang. »Maximally Consistent Sampling and the Jaccard Index of Probability Distributions.« *International Conference on Data Mining, Workshop on High Dimensional Data Mining*, 2018: 347-356.

Nenkova, Ani. »Summarization Evaluation for Text and Speech: Issues and Approaches.« *Ninth International Conference on Spoken Language Processing*, 2006.

Nenkova, Ani, i Kathleen McKeown. »Automatic Summarization.« U *Foundations and Trends in Information Retrieva*, uredio Maarten de Rijke, Yiqun Liu i Diane Kelly, 103-233. Now Publishers Inc, 2011.

Opsahl, Tore, Filip Agneessens, i John Skvoretz. »Node centrality in weighted networks: Generalizing degree and shortest paths.« *Social Networks*, 2010: 245-251.

Paice, C. D. »The automatic generation of literature abstracts: An approach based on the identification of self-indicating phrases.« *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1981.: 172-191.

Parida, Shantipriya, i Petr Motlicek. »Abstract Text Summarization: A Low Resource Challenge.« *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019: 5994-5998.

Saggion, Horacio, i Thierry Poibeau. »Automatic Text Summarization: Past, Present and.« U *Multi-source, Multilingual Information Extraction*, autor Thierry Poibeau, Horacio Saggion, Jakub Piskorski i Roman Yangarber, uredio Thierry Poibeau, Horacio Saggion, Jakub Piskorski i Roman Yangarber, 3-13. Springer, 2016.

—. »Automatic Text Summarization: Past, Present and Future.« *Multi-source, Multilingual Information Extraction*, 2012: 3-21.

Shostenko, Luka. *lexrank 0.1.0*. 3.. ožujka 2018. https://pypi.org/project/lexrank/ (pokušaj pristupa 8.. ožujka 2021.).

Steinberger, Josef, i Karel Jezek. »Evaluation Measures for Text Summarization.« *Computing and Informatics*, Siječanj 2009.: 1001-1026.

Wu, Zhibiao, i Martha Palmer. »Verbs semantics and lexical selection.« *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, lipanj 1994: 133-138.

9.2 Web stranice i članci

Count Bayesie. 10.. svibnja 2017. https://www.countbayesie.com/blog/2017/5/9/kullback-leibler-divergence-explained (pokušaj pristupa 2.. svibanj 2021).

Joshi, Prateek. *An Introduction to Text Summarization using the TextRank Algorithm (with Python implementation).* 1.. listopada 2018. https://www.analyticsvidhya.com/blog/2018/11/introduction-text-summarization-textrank-python/ (pokušaj pristupa 3.. svibnja 2021.).

pytextrank 3.1.1. 25.. ožujka 2021. https://pypi.org/project/pytextrank/ (pokušaj pristupa 12.. svibnja 2021.).

Python | Lemmatization with NLTK. 6.. studenog 2018. https://www.geeksforgeeks.org/python-lemmatization-with-nltk/ (pokušaj pristupa 23.. veljače 2021.).

Random Forest in Python, Will. *Random Forest in Python.* 27.. prosinca 2017. https://towardsdatascience.com/random-forest-in-python-24d0893d51c0 (pokušaj pristupa 4.. svibnja 2021.).

Saba, Gulsanober. *Python Switch Case Statement Tutorial – Three Ways To Implement Python Switch Case Statement.* 11.. travnja 2019. https://www.simplifiedpython.net/python-switch-case-statement/ (pokušaj pristupa 16.. svibnja 2021.).

SwissText. *SwissText2019.* 2019. https://www.swisstext.org/swisstext.org/2019/shared-task/german-text-summarization-challenge.html (pokušaj pristupa 4. kolovoz 2021.).

Vashisht, Ashutosh. *LexRank method for Text Summarization.* n.d. https://iq.opengenus.org/lexrank-text-summarization/ (pokušaj pristupa 18.. svibnja 2021.).

Wartena, Christian. *Vorverarbeitung von Texten mit Python und NLTK*. 14.. prosinca 2020. https://textmining.wp.hs-hannover.de/Preprocessing.html (pokušaj pristupa 12.. travnja 2021.).