Algoritmi za podudaranje znakovnih nizova

Markić, Ivan

SVEUČILIŠTE U SPLITU
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

Ivan Markić

ALGORITMI ZA PODUDARANJE ZNAKOVNIH NIZOVA: METODOLOGIJA IZGRADNJE DOMENSKIH MODELA VREDNOVANJA KLASIČNIH ALGORITAMA I NOVI ALGORITAM S PREDOBRADOM UZORKA

DOKTORSKA DISERTACIJA

Split, 2022.

S V E U Č I L I Š T E U S P L I T U
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

Ivan Markić

Algoritmi za podudaranje znakovnih nizova: metodologija izgradnje domenskih modela vrednovanja klasičnih algoritama i novi algoritam s predobradom uzorka

DOKTORSKA DISERTACIJA

Split, 2022.

Doktorska disertacija je izrađena na Zavodu za elektroniku i računarstvo, Fakulteta elektrotehnike, strojarstva i brodogradnje

Mentor: prof. dr. sc. Maja Štula
Rad br. 179.

PODACI ZA BIBLIOGRAFSKU KARTICU
Ključne riječi: Algoritmi za podudaranje znakovnih nizova, domenski model rangiranja algoritama, pristupi vrednovanja algoritama i metrike, online egzaktni algoritmi s predobradom uzorka, okvir za testiranje algoritama
Znanstveno područje: Tehničke znanosti
Znanstveno polje: Računarstvo
Znanstvena grana: Obradba informacija
Institucija na kojoj je rad izrađen: Sveučilište u Splitu, Fakultet elektrotehnike, strojarstva i brodogradnje
Mentor rada: prof. dr. sc. Maja Štula
Broj stranica: 144
Broj slika: 62
Broj tablica: 21
Broj korištenih bibliografskih jedinica: 154

Povjerenstvo za ocjenu doktorske disertacije:

1. Prof. dr. sc. Mario Čagalj (FESB, Split)
2. Izv. prof. dr. sc. Mirjana Domazet-Lošo (FER, Zagreb)
3. Izv. prof. dr. sc. Matko Šarić (FESB, Split)
4. Doc. dr. sc. Marin Bugarić (FESB, Split)
5. Izv. prof. dr. sc. Toni Perković (FESB, Split)

Povjerenstvo za obranu doktorske disertacije:

1. Prof. dr. sc. Mario Čagalj (FESB, Split) - predsjednik
2. Izv. prof. dr. sc. Mirjana Domazet-Lošo (FER, Zagreb) - član
3. Izv. prof. dr. sc. Matko Šarić (FESB, Split) - član
4. Izv. prof. dr. sc. Toni Perković (FESB, Split) - član
5. Doc. dr. sc. Marin Bugarić (FESB, Split) - član

Zamjenici:

1. Izv. prof. dr. sc. Josip Lörincz (FESB, Split)
2. Izv. prof. dr. sc. Ivo Stančić (FESB, Split)

Disertacija obranjena dana: 4. studenoga 2022.

Algoritmi za podudaranje znakovnih nizova: metodologija izgradnje domenskih modela vrednovanja klasičnih algoritama i novi algoritam s predobradom uzorka

Sažetak:

Odabir učinkovitog algoritma za podudaranje znakovnih nizova u tekstu izazovan je proces. Na izvršavanje algoritma mogu utjecati razni čimbenici poput jačine procesora, veličine radne memorije računala ili mogućnosti programskog jezika. U ovome radu predstavljena je metodologija izgradnje domenskih modela vrednovanja algoritama za podudaranje znakovnih nizova u tekstu temeljena na entropiji uzorka, izražavajući učinkovitost algoritma korištenjem metrika neovisnih o hardverskoj i softverskoj platformi. Razvijena metodologija je usmjerena prema algoritmima za egzaktnu usporedbu znakova prilikom podudaranja uzorka u tekstu bez prethodne spoznaje strukture teksta. Metodologija rangira algoritme prema njihovoj učinkovitosti koristeći svojstva uzorka koji se pretražuje, a ne ovisi o implementaciji algoritma ni o računalnoj arhitekturi te stoga pruža neovisan način odabira algoritma za određenu domenu primjene. Metodologija izgradnje domenskih modela vrednovanja algoritama eksperimentalno je potvrđena na dvije domene, DNA domeni i domeni prirodnog jezika. U radu je prezentiran i novi algoritam za egzaktno podudaranje znakovnih nizova. Heuristička tehnika prolaska kroz uzorak pomoću neparnih i parnih pozicija znakova u kombinaciji s podacima dobivenim predobradom uzorka oblikovana je pravilima algoritma. Definirana heuristička pravila koriste se za preskakanje nepotrebnih usporedbi znakova uzorka. Točnost i pouzdanost algoritma je eksperimentalno vrednovana.

Ključne riječi:

Algoritmi za podudaranje znakovnih nizova, domenski model rangiranja algoritama, pristupi vrednovanja algoritama i metrike, online egzaktni algoritmi s predobradom uzorka, okvir za testiranje algoritama

String matching algorithms: methodology for building domain models for classical algorithms evaluation and a novel algorithm with pattern preprocessing

Abstract:

Choosing an efficient algorithm for strings matching in a text is a challenging process. The algorithm execution can be affected by various factors such as the processor power, the size of the computer memory, or the programming language capabilities. This doctoral thesis presents the methodology for building domain models for strings matching algorithms evaluation based on pattern entropy, expressing the algorithm's efficiency using hardware and software platform independent metrics. The developed methodology is focused on algorithms for exact string comparison to match the pattern in the text without a prior knowledge of the text structure. The methodology ranks algorithms according to their efficiency using the properties of the pattern being searched, and does not depend on the implementation of the algorithm or the computer architecture and therefore provides an independent way of selecting an algorithm for a particular application domain. The methodology for building domain models for evaluating algorithms has been experimentally confirmed on two domains, the DNA domain, and the natural language domain. The doctoral thesis also presents a new algorithm for exact string matching. The heuristic technique of passing through a pattern using odd and even positions of characters, in combination with the data obtained by preprocessing the pattern, is shaped in the algorithm rules. Defined heuristic rules are used to skip unnecessary comparisons of pattern characters. The accuracy and reliability of the algorithm were experimentally evaluated.

Keywords:

String matching algorithms, domain algorithm ranking model, algorithm evaluation approaches and metrics, online exact algorithms with pattern preprocessing, algorithm testing framework

Zahvala

Zahvaljujem se svojoj obitelji i mentorici na razumijevanju i svesrdnoj podršci prilikom izrade ove doktorske disertacije.

Sadržaj

Popis tablica ix

Popis slika 11

UVOD 14

1.1. Motivacija 14

1.2. Pregled dosadašnjih istraživanja 18

1.3. Hipoteza 23

1.4. Opis i metodologija istraživanja 24

1.5. Očekivani znanstveni doprinos 27

1.6. Struktura 28

2. TEORIJSKE OSNOVE 29

2.1. Podudaranje znakovnih nizova 30

2.1.1. Pretraga prefiksa 31

2.1.2. Pretraga sufiksa 34

2.1.3. Pretraga faktora 39

2.2. Algoritamska složenost i izražavanje učinkovitosti algoritama 40

2.3. Odabir skupova podataka 43

2.4. Entropija 48

3. METODOLOGIJA IZGRADNJE DOMENSKIH MODELA VREDNOVANJA ALGORITAMA ZA EGZAKTNU USPOREDBU NIZOVA 52

3.1. Izbor reprezentativnih tekstova za izgradnju modela domene 54

3.2. Odabir algoritama 57

3.3. Rezultati pretraživanja uzoraka izraženi brojem usporedbi znakova 57

3.4. Izračun entropije uzorka 59

3.5. Podjela vrijednosti entropije u razrede 61

3.5.1. Razredi entropije za DNA domenu 62

3.5.2. Razredi entropije za domenu prirodnog jezika 64

3.6. Klasifikacija algoritama u izgrađenom modelu 66
3.7. Vrednovanje metodologije 78
3.7.1. Odabir skupa uzoraka za vrednovanje modela 80
3.7.2. Rezultati vrednovanja metodologije 85
4. NOVI ALGORITAM S PREDOBRADOM UZORKA 94
4.1. Vrednovanje novog algoritma 105
5. ZAKLJUČAK 119
5.1. Doprinosi disertacije 120
5.2. Daljnje istraživanje 121
6. Dodatak A 132
7. Dodatak B 135
8. Dodatak C 138

Popis tablica

Tablica 2.1. Tablica notacije..... 29

Tablica 3.1. Vrijednosti entropije za odabrane uzorke..... 59

Tablica 3.2. Ukupan broj opažanja vrijednosti entropije za DNA domenu, n = 91 62

Tablica 3.3. Diskretizacija podataka za domenu DNA 63

Tablica 3.4. Razredi entropije nakon diskretizacije za domenu DNA 63

Tablica 3.5. Ukupan broj opažanja vrijednosti entropije za domenu prirodnog jezika, n = 105 64

Tablica 3.6. Diskretizacija podataka za područje prirodnog jezika..... 65

Tablica 3.7. Razredi entropije nakon diskretizacije za područje prirodnog jezika..... 65

Tablica 3.8. Model rangiranja algoritama DNA domene 68

Tablica 3.9. Lista najučinkovitijih algoritama procijenjenih modelom za DNA domenu tekstova i uzoraka 72

Tablica 3.10. Model rangiranja algoritama za tekstove i uzorke prirodnog jezika 73

Tablica 3.11. Lista najučinkovitijih algoritama procijenjenih modelom za domenu tekstova i uzoraka prirodnog jezika..... 77

Tablica 3.12. Model rangiranja algoritama DNA domene i domene prirodnog jezika za odabrane razrede vrednovanja za gornji kvartil 79

Tablica 3.13. Izračun dvostruko skalirane Euklidove udaljenosti za 7. razred entropije domene DNA..... 87

Tablica 3.14. Dvostruka Euklidova udaljenost i koeficijent sličnosti za odabrane razrede DNA domene..... 87

Tablica 3.15. Dvostruka Euklidova udaljenost i koeficijent sličnosti za odabrane razrede domene prirodnog jezika 88

Tablica 3.16. Pearsonov koeficijent korelacije za 7. i 9. razred DNA domene..... 89

Tablica 3.17. Pearsonov koeficijent korelacije za 6. i 9. razred domene prirodnog jezika 90

Tablica 4.1. Tablica predobrade za uzorak iz primjera sa slike 4.10. 104

Tablica 4.2. Tablica pretraživanja za spremanje pozicija indeksa zadnjeg pročitanog znaka teksta za primjer sa slike 4.10. 104

Tablica 4.3. Vremenska i prostorna složenost algoritama korištenih u istraživanju 105

Popis slika

Slika 1.1. Klasifikacija algoritama za podudaranje znakovnih nizova 18
Slika 2.1. Prozor pretrage uzorka HACKING za domenu prirodnog jezika 31
Slika 2.2. Primjer pretrage uzorka pomoću naivnog algoritma 31
Slika 2.3. Pretraga prefiksa uzorka 32
Slika 2.4. Pomak Knuth-Morris-Pratt algoritma 33
Slika 2.5. Primjer pretrage Knuth-Morris-Pratt algoritma 34
Slika 2.6. Pretraga sufiksa uzorka 34
Slika 2.7. Prvi slučaj pomaka Boyer-Moore algoritma 35
Slika 2.8. Drugi slučaj pomaka Boyer-Moore algoritma 36
Slika 2.9. Treći slučaj pomaka Boyer-Moore algoritma 37
Slika 2.10. Primjer primjene pravila dobrog sufiksa Boyer-Moore algoritma 38
Slika 2.11. Primjer primjene pravila lošeg znaka Boyer-Moore algoritma 38
Slika 2.12. Pretraga faktora uzorka 39
Slika 2.13. Pomak prozora pretrage pretragom faktora uzorka 39
Slika 2.14. Big-O grafički prikaz složenosti 42
Slika 2.15. Odsječak teksta iz DNA domene 45
Slika 2.16. Odsječak teksta iz engleskog jezika 45
Slika 2.17. Primjer reprezentativnih uzoraka DNA domene 46
Slika 2.18. Primjer reprezentativnih uzoraka domene prirodnog jezika 47
Slika 3.1. Metodologija izgradnje modela temeljenog na entropiji za odabir algoritama za pretraživanje znakovnih nizova 54
Slika 3.2. Primjer rezultata pretrage 58
Slika 3.3. Izračunate entropije uzoraka 60
Slika 3.5. Blok dijagram primjene nove metodologije na primjeru DNA domene 67

Slika 3.6. Grafički prikaz rezultata modela za uzorke 1. razreda entropije DNA domene 71

Slika 3.7. Grafički prikaz rezultata modela za uzorke 8. razreda entropije DNA domene 71

Slika 3.8. Grafički prikaz rezultata modela za uzorke 6. razreda entropije domene prirodnog jezika 75

Slika 3.9. Grafički prikaz rezultata modela za uzorke 7. razreda entropije domene prirodnog jezika 76

Slika 3.10. Središnji granični teorem za uzorke 7. razreda entropije DNA domene 81

Slika 3.11. Središnji granični teorem za uzorke 9. razreda entropije DNA domene 82

Slika 3.12. Središnji granični teorem za uzorke 6. razreda entropije domene prirodnog jezika 83

Slika 3.13. Središnji granični teorem za uzorke 9. razreda entropije domene prirodnog jezika 84

Slika 3.14. Pearsonov linearni koeficijent korelacije domene DNA 91

Slika 3.15. Pearsonov linearni koeficijent korelacije domene prirodnog jezika 92

Slika 4.1. Pronađeni uzorak CIB algoritmom 95

Slika 4.2. Prvo pravilo pomaka uzorka CIB algoritma 96

Slika 4.3. Drugo a) pravilo pomaka uzorka CIB algoritma 97

Slika 4.4. Drugo b) pravilo pomaka uzorka CIB algoritma 98

Slika 4.5. Drugo c) pravilo pomaka uzorka CIB algoritma 98

Slika 4.6. Treće pravilo pomaka uzorka CIB algoritma 99

Slika 4.7. Četvrto a) pravilo pomaka uzorka CIB algoritma 100

Slika 4.8. Četvrto b) pravilo pomaka uzorka CIB algoritma 101

Slika 4.9. Grafički prikaz slijeda instrukcija CIB algoritma 102

Slika 4.10. Primjer pretrage uzorka CIB algoritmom 103

Slika 4.11. Srednja vrijednost broja usporedbi znakova (CC) za uzorke duljine 2, 4, 8, 16 i 32 znaka iz DNA domene 109

Slika 4.12. Srednja vrijednost broja usporedbi znakova (CC) za uzorke duljine 64, 128, 256, 512 i 1024 znaka iz DNA domene 109

Slika 4.13. Srednja vrijednost broja usporedbi znakova (CC) za uzorke duljine 2, 4, 8, 16 i 32 znaka iz domene prirodnog jezika 110

Slika 4.14. Srednja vrijednost broja usporedbi znakova (CC) za uzorke duljine 64, 128, 256, 512 i 1024 znaka iz domene prirodnog jezika 110

Slika 4.15. Srednja vrijednost vremena izvršavanja algoritma (ms) za uzorke duljine 2, 4, 8, 16 i 32 znaka iz DNA domene 111

Slika 4.16. Srednja vrijednost vremena izvršavanja algoritma (ms) za uzorke duljine 64, 128, 256, 512 i 1024 znaka iz DNA domene 112

Slika 4.17. Srednja vrijednost vremena izvršavanja algoritma (ms) za uzorke duljine 2, 4, 8, 16 i 32 znaka iz domene prirodnog jezika 113

Slika 4.18. Srednja vrijednost vremena izvršavanja algoritma (ms) za uzorke duljine 64, 128, 256, 512 i 1024 znaka iz domene prirodnog jezika 113

Slika 4.19. Srednja vrijednost potrošnje memorije za uzorke duljine 2, 4, 8, 16 i 32 znaka iz DNA domene 114

Slika 4.20. Srednja vrijednost potrošnje memorije za uzorke duljine 64, 128, 256, 512 i 1024 znaka iz DNA domene 115

Slika 4.21. Srednja vrijednost potrošnje memorije za uzorke duljine 2, 4, 8, 16 i 32 znaka iz domene prirodnog jezika 116

Slika 4.22. Srednja vrijednost potrošnje memorije za uzorke duljine 64, 128, 256, 512 i 1024 znaka iz domene prirodnog jezika 116

Slika 6.1. Orange Data Suite model 133

Slika 6.2. Orange Data Suite diskretizacija ulaznog skupa podataka vrijednosti entropije uzoraka 134

Slika 7.1. Karakteristike razreda entropije unutar IBM SPSS softverskog paketa 135

Slika 7.2. Vrijednosti 7. i 9. razreda entropije uzoraka korištenih u izgradnji i validaciji modela DNA domene 136

Slika 7.3. IBM SPSS Statistics Viewer 137

Slika 8.1. Postavke filtera Process Monitora za skupljanje empirijskih podataka eksperimentalne analize 139

Slika 8.2. Prikupljeni empirijski podaci pomoću Process Monitora za eksperimentalnu analizu 140

UVOD

1.1. Motivacija

Količina podataka na godišnjoj razini kontinuirano raste za 50%, a upravljanje informacijama postaje glavni izazov za organizacije širom svijeta [1]. Očekivanja od analize podataka su velika, a otkrivanje znanja iz unutarnjih ili vanjskih izvora podataka organizacijama pomaže u donošenju boljih i kvalitetnijih poslovnih odluka [2], [3]. Analize globalnih istraživačkih i savjetodavnih tvrtki pokazuju da su informacije postale jedno od najcjenjenijih dobara koje organizacije imaju. Osim što informacije mogu generirati dodatne prihode, novi tehnološki pristupi obradi velikih količina nestrukturiranih i strukturiranih podataka (engl. *Big Data*) stavljaju sve veće izazove na postojeću infrastrukturu informacijskih sustava [4]–[7].

Podaci se, zahvaljujući razvoju modernih baza podataka, spremaju u različitim formatima. Međutim, tekst i dalje ostaje glavni oblik pohrane i razmjene podataka. Kada se radi o podacima u formi teksta osnovni računalni zadatak svodi se na proces podudaranja znakovnih nizova (engl. *String Pattern Matching*) ili pretrage uzorka u određenom nizu. Primjena algoritama za provođenje procesa podudaranja znakovnih nizova nalazi se u mnogim aplikacijama kao što su programi za obradu teksta, sustavi web tražilica, multimedijalne aplikacije, forenzičke aplikacije, bioinformatički softver, aplikacije za informacijsku sigurnost, različite aplikacije za rudarenja teksta (engl. *text mining)*, itd. Primjerice *online* rječnici posjeduju baze od više desetaka milijuna riječi koje se lako pretražuju (poput Oxford English rječnika koji ima bazu od 50 milijuna riječi) [8]–[12].

Posljednjih desetljeća razvijeno je više različitih algoritama za podudaranje znakovnih nizova, a njihov razvoj i unaprjeđivanje ne prestaje zbog njihove važnosti u rješavanju mnogih problema u računalnoj znanosti. Podudaranje znakovnih nizova jedan je od najstarijih i najistraživanijih zadataka u računalnoj znanosti te predstavlja važan proces koji se primjenjuje u gotovo svim drugim područjima industrije. Sve veća količina tekstualnih podataka zahtijeva razvoj novih pristupa i alata za učinkovitiju pretragu korisnih informacija. Za područje obrade niza karaktera u svrhu podudaranja koristi se ukorijenjeni naziv „stringologija“ [13]–[21].

Različiti algoritmi za podudaranje znakovnih nizova imaju bolje ili lošije rezultate što ovisi između ostalog i o domeni primjene, a svako rješenje određenog problema predmet je višestrukih pristupa, od teorijskih do praktičnih. Teorijska rješenja dovode do značajnih algoritamskih postignuća, ali često su manje korisna u praksi jer je podudaranje znakovnih nizova računalno intenzivno. Višestruki čimbenici, kao što su kvaliteta implementacije algoritma, zahtjevi za točnošću (engl. *accuracy*), računalna snaga i arhitektura računala[1], svojstva programskog prevoditelja[2], itd., mogu utjecati na učinkovitost algoritma. Algoritam je učinkovit ako je njegova potrošnja resursa u procesu izvršavanja ispod prihvatljive ili poželjne razine [22]–[30].

U praksi se vrednovanje učinkovitosti algoritama najčešće provodi *ad hoc* pristupima. *Ad hoc* pristupi ogledaju se kroz pojedinačne, specifične implementacije različitih metrika poput vremena izvršavanja, zauzeća procesora i potrošnje memorije, a dobiveni rezultati su često subjektivni, neprenosivi ili čak i neponovljivi, što u konačnici dovodi u pitanje točnost istih i usporedivost s rezultatima strogo formalnih metrika [31]–[36].

Proučavanje složenosti i ograničenja dostupnih algoritama kompleksan je i dugotrajan proces. Veliki broj raspoloživih metrika i atributa kvalitete za kvantitativnu analizu[3] učinkovitosti algoritama usložnjava usporedbu algoritama otežavajući pri tome odabir najučinkovitijeg pa čak i samu standardnu definiciju učinkovitosti algoritama za pronalaženje tekstualnih uzoraka. Postojeće metrike rijetko su sintetizirane u formi gotovog alata zbog manjka ontologija[4] koje opisuju odnose između različitih konceptualnih rješenja. Primjena takvih alata je otežana u kontekstu prikupljanja podataka, kvantifikacije i analize rezultata, kao i dostupnosti drugim istraživačima. Osim manjka ontologija raspoložive metrike su ponekad teško primjenjive u praksi uslijed

[1] U računalnom inženjerstvu, arhitektura računala je skup pravila i metoda koji opisuju funkcionalnost, organizaciju i implementaciju računalnih sustava. Arhitektura sustava odnosi se na njegovu strukturu u smislu zasebno specificiranih komponenti tog sustava i njihovih međusobnih odnosa [30].

[2] Kompajler (prevodilac, prevoditelj, programski prevoditelj, engl. *compiler*) je računalni program koji čita program napisan u izvornom jeziku, te ga prevodi na strojni ili asemblerski kod [153].

[3] Kvantitativna analiza je istraživanje koje uglavnom uključuje brojeve kako bi se osigurala objektivnost, pouzdanost i mogućnost poopćavanja. Bitna karakteristika kvantitativnog istraživanja je da je proces prikupljanja podataka odvojen od analize prikupljenih podataka [77].

[4] Ontologija je obrazac podatka koji predstavlja koncepte unutar neke domene i odnose između tih koncepata. Koristi se za razumijevanje objekata unutar domene [154].

stroge formalne podloge s jakom matematičkom pozadinom. Sve navedeno komplicira definiranje svojstava, odnosno atributa kvalitete algoritama koji bi bili korisni pri odabiru određene metrike i prezentacije rezultata [37], [38].

State-of-the-art analiza pokazuje nepostojanje općeprihvaćenih metodologija što, uz već navedeni široki spektar korištenih metrika i atributa kvalitete učinkovitosti algoritama, otežava vrednovanje algoritama, ali i uspoređivanje različitih pristupa u vrednovanju te objektivizaciju rezultata vrednovanja. Korištenje određene metodologije za odabir najučinkovitijeg algoritma za podudaranje znakovnih nizova omogućilo bi jednostavniji i nedvosmislen odabir najučinkovitijeg algoritma za domenu primjene [39], [40].

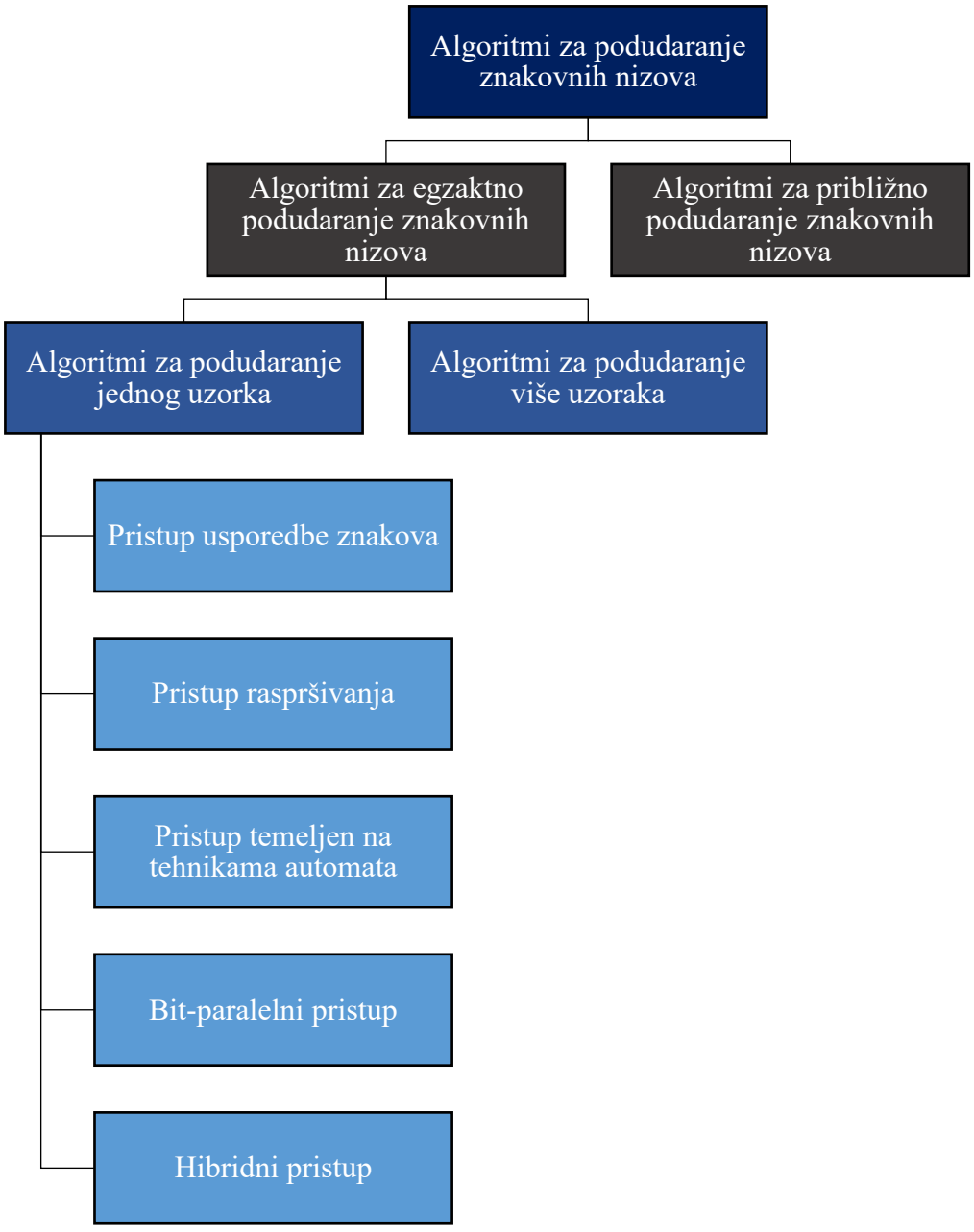
U ovoj doktorskoj disertaciji predstavljena je metodologija izgradnje domenskih modela vrednovanja algoritama za egzaktno podudaranje znakovnih nizova u svrhu odabira učinkovitog algoritma. Predstavljena metodologija temelji se na entropiji uzorka, a učinkovitost metodologije je izražena pomoću metrika koje nisu ovisne o hardverskoj infrastrukturi ili softverskoj platformi. Metodologijom se rangiraju algoritmi za egzaktno podudaranje znakovnih nizova prema učinkovitosti koristeći svojstva uzorka koji se pretražuje i za koja je eksperimentalno potvrđeno da su povezana s učinkovitošću korištenih algoritama. Razni čimbenici poput implementacije algoritma, računalne arhitekture, specifičnosti programskih jezika ne utječu na definiranu metodologiju rangiranja algoritma. Tako razvijeni modeli pružaju neovisan način odabira algoritma za određenu domenu primjene. U predloženoj metodologiji, učinkovitost algoritma se izražava pomoću metrike broja usporedbe znakova napravljene u procesu pretrage uzorka u tekstu. Obzirom na složenost pronalaženja uzoraka u tekstu te na različite tipove algoritama, razvoj modela za rangiranje algoritama ograničen je na algoritme za egzaktno podudaranje znakovnih nizova bez prethodnog poznavanja strukture teksta s pristupom usporedbe znakova uz mogućnost proširenja, uz određene prilagodbe, na ostale vrste algoritama za pretragu uzorka u tekstu.

U sklopu disertacije prezentiran je i novi algoritam, CIB (engl. *Character Index Based Algorithm*), za egzaktno podudaranje znakovnih nizova bez prethodnog poznavanja strukture teksta s pristupom usporedbe znakova. Razvijeni algoritam sastoji se od skupa heurističkih pravila za preskakanje nepotrebnih usporedbi znakova uzorka koja su ujedno i osnova novog algoritma. Pravila algoritma formalizirana su heurističkim tehnikama prolaska kroz uzorak pomoću neparnih i parnih pozicija znakova u

kombinaciji s podacima dobivenim predobradom uzorka. Novi algoritam funkcionira na način da se prilikom podudaranja znakovnih nizova vrši preskakanje što je više moguće usporedbi znakova prilikom pretrage uzorka. Preskakanjem usporedbi, u cilju što bržeg pronalaska znakova uzorka i teksta koji se ne podudaraju, ubrzava se pretraga uzroka i posljedično se smanjuju potrebni kapaciteti za izvršavanje algoritma. To smanjivanje čini algoritam učinkovitijim. Cilj CIB algoritma je korištenje do sada neiskorištenih svojstava uzoraka znakova u svrhu smanjenja broja usporedbi znakova prilikom pretrage uzorka u tekstu, pri čemu se smanjuje i potrošnja računalnih resursa.

1.2. Pregled dosadašnjih istraživanja

Algoritmi za podudaranje znakovnih nizova mogu se podijeliti u nekoliko kategorija prema svojstvima algoritama. Podjela algoritama prikazana je na slici 1.1 [27].



Slika 1.1. Klasifikacija algoritama za podudaranje znakovnih nizova

Osnovna podjela algoritama za podudaranje znakovnih nizova je na dvije grupe, egzaktne (engl. *exact*) i približne (engl. *approximate*) algoritme. Egzaktno podudaranje

znakovnih nizova podrazumijeva da svi uspoređeni znakovi uzorka i teksta moraju biti isti kako bi uzorak bio pronađen. Kod približnog podudaranja znakovnih nizova svi uspoređeni znakovi ne moraju biti isti, što za cilj ima pronaći podniz unutar teksta sličan traženom uzorku. Nakon osnovne podjele, algoritmi za podudaranje znakovnih nizova dalje se dijele na algoritme za podudaranje jednog uzorka i algoritme za podudaranje više uzoraka. Algoritmi za podudaranje znakovnih nizova koji pripadaju grupi algoritama za podudaranje jednog ili više uzoraka mogu se sukladno pristupu pretrage uzorka unutar teksta, podijeliti na pristup usporedbe znakova (engl. *character comparison based approach*, CC), pristup raspršivanja (engl. *hashing*), pristup temeljen na tehnikama automata, bit-paralelni i hibridni pristup [41].

Algoritmi za podudaranje znakovnih nizova koji pripadaju određenim pristupima pretrage uzorka unutar teksta imaju svoju softversku izvedbu algoritma koja koristi određene kompajlere i programske jezike u svrhu implementacije. Softverska izvedba algoritma je prilagodljiva u smislu izmjena i može se implementirati neograničen broj puta u različitim aplikacijama [42].

Softverska izvedba algoritama može se implementirati i na posebnom hardveru. U tome slučaju potrebno je imati posebne hardverske uređaje, kao što su grafičke jedinice za obradu (GPU, *Graphics Processing Unit*) i programibilna logička polja (FPGA, *Field Programmable Gate Array*). Takve izvedbe algoritama mogu se implementirati pomoću specifičnih programskih okruženja koja podržavaju paralelnu obradu. Prilagodba algoritma hardverskim specifičnostima brža, ali znatno skuplja. Osim nedostatka u cijeni, nakon implementacije algoritma za podudaranje znakovnih nizova na posebnom hardveru, nije moguća primjena na različite podatke ili aplikacije jer je potrebna promjena dizajna hardvera [42]–[44].

Pristup temeljen na usporedbi znakova funkcionira tako da uspoređuje pojedinačno znakove uzorka i znakove teksta da bi pronašao uzorak unutar teksta. Dva najpoznatija algoritma koji koriste pristup usporedbe znakova su *brute force* (naivni algoritam) i Boyer-Moore [45]. *Brute force* algoritam ne vrši predobradu uzorka, a svaki se znak uspoređuje s lijeva na desno pojedinačno bez obzira na vrijeme izvršavanja. Pristup temeljen na tehnikama raspršivanja štedi resurse u procesu izvršavanja. Umjesto usporedbi znakova vrši se usporedba vrijednosti raspršivanja (engl. *Hash value*) znakova [27], [46]. Sufiks automat je automat koji se sastoji od dva povezana, ali različita konstruktora automata za usporedbu: deterministički acikličkog konačnog automata

(struktura podataka koja predstavlja konačan skup znakova) i sufiks automata (konačnog automata u svojstvu sufiks indeksa) za podudaranje [47]. Ovaj pristup koristi usmjereni aciklički graf u kojem se čvorovi/vrhovi nazivaju stanjima, a bridovi između čvorova smatraju se prijelazom između stanja. Ovaj pristup koristi strukturu podataka sufiks automata koja prepoznaje sve sufikse uzorka. Algoritmi za podudaranje znakovnih nizova temeljeni na automatima dobri su za duge uzorke, ali nisu prikladni za uzorke kraćih duljina. Bit-paralelni pristup podrazumijeva paralelnu obradu. Paralelnu obradu prvi je put predložio Domolki 1968. kako bi se ubrzao postupak podudaranja. Taj koncept temelji se na paralelnom računarstvu. U tom pristupu broj operacija unutar algoritma se smanjuje na broj bitova u računalnoj riječi što algoritam čini bržim i učinkovitijim. Algoritmi koriste logičke operacije nad bitovima kao što su NOT, OR, AND i XOR za usporedbu nizova [27], [41], [48].

Dodatna podjela algoritama za podudaranje znakovnih nizova je na *online* i *offline* algoritamska rješenja pretrage. *Online* algoritamska rješenja pretrage ne zahtijevaju apriorno poznavanje strukture teksta, a eventualna predobrada se izvodi na uzorku. *Online* algoritamska rješenja za podudaranje znakovnih nizova sastoje se od dvije faze: faza predobrade uzorka i faze pretrage uzorka u tekstu. U fazi predobrade, formira se struktura uzorka koja je obično proporcionalna njegovoj duljini, a detalji formiranja strukture razlikuju se za različite izvedenice algoritama. Faza pretraživanja koristi dobivenu strukturu uzorka podataka i pokušava odrediti pojavljuje li se uzorak u tekstu. *Offline* algoritamska rješenja temelje se na aktivnostima predobrade teksta [49]–[53].

Još jedna podjela algoritma za podudaranje znakovnih nizova može se napraviti na dvije grupe sukladno razdoblju nastanka. Prva grupa algoritama za podudaranje znakovnih nizova pripada razdoblju nastanka prije 2000. godine, a druga nakon 2000. godine [51]. Ako se promatra razdoblje nastanka algoritama za podudaranje znakovnih nizova prije 2000. godine najčešće korišteni algoritmi za egzaktno podudaranje znakovnih nizova, uz osnovni naivni algoritam, su Boyer-Moore [45] i Knuth-Morris-Pratt [54] algoritmi. Boyer-Moore algoritam se smatra općenito najučinkovitijim algoritmom za podudaranje znakovnih nizova kod uobičajenih primjena pretrage teksta. Koristi se u gotovo svim uređivačima teksta i postao je standardni algoritam za usporedbu učinkovitosti ostalih algoritama. Ključna komponenta Boyer-Moore algoritma je tablica pomaka dobivena u procesu predobrade uzorka. Tablica pomaka sadrži informacije o broju znakova koji se mogu preskočiti nakon nepodudaranja znakova uzorka i teksta [55]. Jedan od temeljnih algoritama koji koristi koncepte sufiks automata je Knuth-Moriss-

Pratt [54]. Quick Search algoritam je varijanta Boyer-Moore algoritma [56], a Berry-Ravindran algoritam je poboljšana verzija Quick Search algoritma [57]. Postoje još i algoritmi temeljeni na sufiksnim automatima poput Reverse Factor i Turbo Reverse Factor algoritama [58]. Još jedna varijanta Boyer-Moore algoritma je i Horspool algoritam koji ima bolje rezultate u određenim praktičnim slučajevima [59]. Shift-Or algoritam brža je izvedenica Knuth-Morris-Pratt algoritma koja je pogodnija za složenije uzorke [60].

Algoritmi za usporedbu znakovnih nizova koji koriste tehniku raspršivanja su Karp-Rabin i Harrison algoritam. Usporedba se vrši s lijeva na desno [41], [61], [62]. U praksi se često koriste za otkrivanje plagijata. Aho-Corasick algoritam je uveo u korištenje teoriju konačnih automata, a predstavlja proširenje Knuth-Morris-Pratt algoritma [63]. Commentz-Walter algoritam koristi teoriju automata, a kombinacija je Boyer-Moore i Aho-Corasick algoritma [64].

U razdoblju nastanka algoritama za podudaranje znakovnih nizova nakon 2000. godine većina algoritama dobivena je poboljšanjem ili kombiniranjem ideja prethodno objavljenih algoritama. Primjerice, postoje varijante Boyer-Moore algoritma kao što su Fast-Search algoritam [65] i Backward-Fast-Search algoritam [66], [67]. Boyer-Moore-Horspool algoritam koji se temelji na teoriji vjerojatnosti preinačena je verzija Horspool algoritma [68]. Franek-Jennings-Smyth algoritam kombinacija je algoritama Knuth-Morris-Pratt i Quick Search algoritma [69].

Približna usporedba nizova podrazumijeva pronalaženje uzorka u tekstu u kojemu uzorak i tekst imaju određen broj razlika. Približno podudaranje postaje sve bitnije pri rješavanju problema pogrešnog unosa u sustavima za pretragu informacija (engl. *information retrieval*), greškama prijenosa u obradi signala, itd. [19], [52], [70]–[74].

Približno podudaranje formira se korištenjem funkcije udaljenosti pomoću koje se izražava koliko su dva niza slična. Postoji nekoliko funkcija udaljenosti. Najpoznatije funkcije udaljenosti su Hammingova i Levenshteinova udaljenost. Hammingova udaljenost između dva niza jednaka je broju pozicija koje se ne podudaraju između dva niza. Problem približne usporedbe nizova Hammingovom udaljenosti naziva se pretraga niza s k nepodudaranja (engl. *string searching with k mismatches*). Levenshteinova udaljenost ili udaljenost uređivanja (engl. *edit distance*) između dva niza je minimalan broj operacija umetanja, brisanja i zamjene znakova potreban za njihovo izjednačavanje.

Problem približne usporedbe nizova Levenshteinovom udaljenosti naziva se pretraga niza sa *k* razlika (engl. *string searching with k differences).* Razlozi za uvođenje približnog podudaranja nizova su: niska kvaliteta teksta, heterogenost baza podataka, pogreške u pravopisu u uzorku ili tekstu, pretraživanje stranih imena i pretraživanje s neizvjesnošću [70].

Algoritmi za približno podudaranje znakovnih nizova koriste specifične pristupe pretrage od kojih je najpoznatiji pristup dinamičkog programiranja. Dinamičko programiranje je naziv popularne tehnike u programiranju kojom se može značajno smanjiti složenost algoritma od eksponencijalne do polinomijalne. Slično metodi „podjeli pa vladaj“ (engl. *divide and conquer*) dinamičkim programiranjem rješavanje jednog problema svodi se na rješavanje podproblema. Za ovakve probleme kaže se da imaju optimalnu podstrukturu (engl. *optimal substructure*). Bitna karakteristika dinamičkog programiranja je da se svaki podproblem rješava najviše jednom, čime se izbjegava ponovno računanje numeričkih karakteristika istog stanja. Koristi se pojam tablične metode iz razloga što se rješenja podproblema čuvaju u pomoćnim tablicama. Dinamičko programiranje je klasično rješenje koje se koristi za računanje udaljenosti uređivanja između dva niza [70].

Postoji nekoliko okvira za testiranje algoritama podudaranja znakovnih nizova [75]. Jedan metodološki pristup za vrednovanje učinkovitosti algoritama za podudaranje znakovnih nizova definirali su Faro, Lecroq i Borz. Faro je 2010. predstavio okvir alata za istraživanje algoritama za podudaranje znakovnih nizova (SMART – *String Matching Algorithm Research Tool*), a šest godina kasnije i poboljšanu verziju. SMART je okvir dizajniran za razvoj, testiranje, usporedbu i procjenu algoritama za podudaranje znakovnih nizova [18]. Ovaj pristup uključuje veliki broj implementiranih algoritama za podudaranje znakovnih nizova čiji nastanak pripada do 2000. godine te skup tekstova (prirodnog jezika i DNA (engl. *Deoxyribonucleic acid*) nizova) nad kojima se algoritmi izvršavaju. SMART za vrednovanje algoritama koristi metrike u obliku vremena izvršavanja algoritma i zauzeća procesora. Hume i Sunday su predstavili svoju metodologiju za testiranje algoritama za podudaranje znakovnih nizova 1991. godine razvijenu u programskom jeziku C, a korištenu u 90-tim godinama prošlog stoljeća. U izgradnji metodologije korišteni su tekstovi iz domene prirodnog jezika te razne vrste prevoditelja za odvajanje izvedbe algoritma od hardvera i dostupnih programskih biblioteka. Za prikaz rezultata testa odabrane su metrike brzine pretraživanja na način da

se mjeri koliko se megabajta teksta pretraži po sekundi, zatim srednja vrijednost veličine pomaka uzorka i postotak usporedbi znakova uzorka u odnosu na tekst [76].

1.3. Hipoteza

Istraživanje je usmjereno prema pristupima i modelima vrednovanja *online* egzaktnih algoritama za podudaranje znakovnih nizova. Postavljene su dvije hipoteze:

1. Prema prvoj postavljenoj hipotezi postoje informacije koje se mogu dobiti iz uzoraka, a koje se mogu iskoristiti za odabir najučinkovitijeg algoritma za pretragu znakovnih nizova. Informacije koje je moguće dobiti iz uzoraka odnose se na svojstva uzoraka koja su vezana uz domenu tekstova i uzoraka koje pretražujemo. Za korištenje tih informacija potrebna je odgovarajuća metrika za razvoj modela pomoću kojega će se moći transparentno i jednostavno napraviti odabir najučinkovitijeg algoritma za podudaranje znakovnih nizova za određenu domenu tekstova. Postavljenom hipotezom, metrika usporedbe znakova, ali i dostupne informacije iz uzorka imaju odgovarajuća svojstva za razvoj takvog modela. Identifikacijom informacija koje se mogu dobiti iz uzoraka bez obzira na veličinu, poput entropije uzoraka te formalizacijom pristupa korištenja tih informacija moguće je postaviti novu metodologiju koja omogućava odabir najučinkovitijeg algoritma za podudaranje znakovnih nizova. Pomoću nove metodologije moguće je odabrati algoritam s kojim se može izvršiti egzaktno podudaranje s minimalnim brojem usporedbi znakova. Entropija je veličina koja se koristi u termodinamici i teoriji informacija. Iskorištavanje entropije u svrhu korištenja informacijskog sadržaja uzorka u izradi modela za odabir najučinkovitijeg algoritma predstavlja dodatnu mogućnost primjene iste. Informacijski sadržaj uzorka vezan je uz svojstva tekstova koji se pretražuju, kroz svojstva abecede, pravila korištenja abecede i sl., stoga je nova metodologija vezana uz domene primjene. U ovom radu ćemo se ograničiti na dvije domene na kojima ćemo primijeniti postavljenu metodologiju za razvoj modela odabira najučinkovitijeg algoritma za uzorak koji se pretražuje (s javno dostupnim standardnim kolekcijama tekstova); na domenu DNA nizova i na domenu prirodnog jezika pomoću kojih će se predložena metodologija predstaviti i vrednovati.

2. Dodatna postavljena hipoteza rada je kako uzorci koji su predmet pretrage sadrže svojstva, koja se mogu heuristički identificirati i definirati, a koja do sada nisu korištena u algoritmima za egzaktno podudaranje znakovnih nizova. Cilj je identificirati takva svojstva unutar uzorka koje je moguće dobiti njegovim analiziranjem u fazi predobrade te uključiti identificirana svojstva u novi algoritam za egzaktno podudaranje znakovnih nizova smanjujući pri tome broj potrebnih usporedbi znakova. Izgradnjom novog algoritma cilj je postići veću učinkovitost u procesu podudaranja na način da se smanji broj potrebnih usporedbi znakova čime se u konačnici smanjuje i potreba za računalnim resursima.

1.4. Opis i metodologija istraživanja

Provedeno istraživanje spada u primijenjena istraživanja u svrhu iznalaženja okvira za odabir najučinkovitijih algoritama za pronalaženje uzoraka u tekstu određene domene primjene neovisno o implementaciji algoritma i dostupnoj računalnoj arhitekturi.

Sukladno postavljenim hipotezama provedeno istraživanje može se podijeliti na teorijski i empirijski dio. Teorijski dio istraživanja obuhvaća analizu postojećih metrika i atributa učinkovitosti algoritama te prepoznaje pristupe kojima se mogu proizvesti objektivni, prenosivi rezultati vrednovanja algoritama za pronalaženje uzoraka neovisni o hardverskom i softverskom okruženju čime je omogućeno definiranje metodološkog pristupa u postavljanju modela za odabir najučinkovitijeg algoritma. Time su određene i granice nove metodologije koja je usmjerena prema algoritmima za egzaktno podudaranje znakovnih nizova koji koriste pristup usporedbe znakova s *online* algoritamskim rješenjem pretrage.

Izrađena metodologija podrazumijeva formalizirani pristup izgradnji modela neovisnog o arhitekturi računala, specifičnostima programskih jezika ili implementaciji algoritama i pruža platformski neovisan način odabira najučinkovitijeg algoritma za egzaktno podudaranje znakovnih nizova. Metodologija definira način odabira skupa tekstova za izgradnju modela, način odabira algoritama za vrednovanje, broj reprezentativnih uzoraka koji je dovoljan za potrebni stupanj pouzdanosti modela, primjenu statističkih mjera i kategorizaciju rezultata entropije reprezentativnih uzoraka za izgradnju modela i ostale detalje od odabira metrike pa do klasifikacije i rangiranja

algoritama izgrađenog modela. U empirijskom dijelu istraživanja razvijena metodologija je provjerena kroz izgradnju i vrednovanja modela rangiranja algoritama za dvije domene tekstova.

U sklopu teorijskog dijela istraživanja provedena je i detaljna analiza i identifikacija svojstava algoritama za egzaktnu usporedbu znakovnih nizova pri čemu su prikupljena određena heuristička saznanja oblikovana u formi heurističkih pravila koja su temelj izgradnje novog algoritma. U empirijskom dijelu istraživanja vrednovan je novi algoritam koji se temelji na heurističkim pravilima usporedbe neparnih i parnih pozicija znakova uzorka naspram teksta čiji se smjer usporedbe prvo kreće prema naprijed preko neparnih pozicija, a nakon toga prema nazad preko parnih pozicija.

Glavni ciljevi istraživanja mogu se predstaviti kroz dvije stavke i to na sljedeći način:

- Razvoj metodologije koja će olakšati odabir najučinkovitijeg algoritma za egzaktno podudaranje znakovnih nizova za zadani uzorak, tekst i domenu
- Razvoj novog algoritma za egzaktno podudaranje znakovnih nizova koji koristi pristup usporedbe znakova

Navedenim ciljevima u znanstvenom istraživanju nadograditi će se postojeće i stvoriti novo znanje kroz razvoj novih metoda i algoritama.

Faze istraživanja moguće je podijeliti na sljedeće cjeline [77]–[79]:

1. Analiza postojećeg stanja pregledom dostupne literature iz područja algoritama za podudaranje znakovnih nizova s posebnim naglaskom na problem egzaktnog podudaranja znakovnih nizova te modele vrednovanja istih,
2. Prepoznavanje i usporedba postojećih metrika i atributa kvalitete koji se koriste za analizu algoritama za podudaranje znakovnih nizova, te odabir platformski neovisnog, a empirijski primjenjivog pristupa pogodnog za mjerenje učinkovitosti algoritama,
3. Postavljanje metodologije za usporedbu učinkovitosti *online* algoritama za egzaktno podudaranje znakovnih nizova, s ciljem formalizacije pristupa u selekciji najučinkovitijeg algoritma za podudaranje znakovnih nizova konkretne domene pretraživanja znakovnih nizova,

4. Definiranje novog algoritma za egzaktno podudaranje znakovnih nizova korištenjem identificiranih svojstava uzoraka oblikovanih u heuristička pravila kojima se smanjuje broj nepotrebnih usporedbi znakova,
5. Empirijsko istraživanje u cilju vrednovanja ispravnosti i uporabljivosti nove definirane metodologije iz točke 3. kao i vrednovanja novog algoritma iz točke 4. na standardnim dostupnim kolekcijama tekstova, DNA nizovima, te domeni prirodnog jezika.

Za potrebe izgradnje i vrednovanja modela korišteni su skupovi tekstova i uzoraka kroz dvije odabrane domene s javno dostupnim standardnim kolekcijama tekstova, DNA nizovima znakova i nizovima znakova prirodnog jezika. Skup tekstova za izgradnju modela sastoji se od četiri javno dostupna teksta DNA nizova živućih organizama (Riba Kanglang iz porodice šarana, Golema želva, Escherichia coli, Rezus makaki iz porodice primata) i jednog javno dostupnog skupa tekstova iz domene prirodnog jezika (Biblija, Kralj James verzija) [80]–[84].

Saznanja korištena u kreiranju modela upotrjebljena su za izgradnju novog algoritma za egzaktnu usporedbu nizova koji koriste pristup usporedbe znakova s *online* algoritamskim rješenjem pretrage. Novi algoritam uspoređen je s ostalim uobičajeno korištenim, poznatim algoritmima iz iste grupe. Za provedbu empirijskog istraživanja vrednovanja novog algoritma, odabrani su skupovi tekstova iz DNA domene i domene prirodnog jezika. Iz DNA domene odabran je niz živućeg organizma Escherichia coli [82], a iz domene prirodnog jezika odabran je biblijski podskup (Biblija, Kralj James verzija) [84]. U svrhu što pouzdanijeg vrednovanja, vrednovanje se provodi korištenjem tri metrike. Korištene su empirijske metrike, vrijeme izvršavanja i potrošnja radne memorije prilikom izvršavanja algoritma, dok je metrika broja usporedbi znakova (CC) korištena kao formalna metrika. Također je provedena i teorijska (*a priori*) vremenska i prostorna analiza algoritama. Za kriterij učinkovitosti algoritma mjere se srednja vrijednost broja usporedbi znakova, srednja vrijednost brzine izražavanja izražene u milisekundama i srednja vrijednost potrošnje radne memorije za svaki uzorak i svaki tekst. Za odabrane algoritme u procesu usporedbe s uobičajeno korištenim algoritmima, njihove gotove implementacije preuzete su iz SMART alata [18].

Za provedbu postupka prikupljanja kvantitativnih podataka za izgradnju i vrednovanje algoritama modela korišten je alat Microsoft Visual Studio 2019. Nakon pokretanja algoritama podaci o broju usporedbi za svaki odabrani algoritam i uzorak

spremaju se u datoteku koja se zatim izdvaja i obrađuje pomoću specijalnih softverskih alata za obradu podataka. Za provedbu postupka obrade kvantitativnih podataka odabran je računalni alat Orange data Suite [85] zbog jednostavnosti analize podataka i vizualizacije. Za pripremu podataka za korištenje unutar odabranog alata izabrani su, zbog preglednosti, samo osnovni podaci za analizu poput naziva algoritma, duljine uzorka, broja potrebnih usporedbi za proces usporedbe znakovnih nizova i entropije uzorka. Dodatna analiza korištenjem opisne statistike i objašnjavanje rezultata napravljeno je pomoću alata Microsoft Excel.

1.5. Očekivani znanstveni doprinos

Korisnici algoritama za podudaranje znakovnih nizova često su u nedoumici odabira najučinkovitijeg algoritma na osnovu dostupnih informacija. Analizom i istraživanjem dostupne literature uvidjelo se da ne postoji općeprihvaćeni pristup vrednovanja algoritama za podudaranje znakovnih nizova [75][76]. Korištenje standardizirane metodologije za odabir najučinkovitijeg algoritma za pretragu nizova omogućilo bi jednostavan i razumljiv odabir najučinkovitijeg algoritma. Stečeni uvid u algoritme za pretragu nizova je pretočen u novi način pronalaska uzoraka u tekstu u formi novog algoritma.

Znanstveni doprinos u području podudaranja znakovnih nizova prikazan je kroz dvije stavke:

1. Nova metodologija za izgradnju modela koji podržava vrednovanje algoritama za egzaktno podudaranje znakovnih nizova koji koriste pristup usporedbe znakova s *online* algoritamskim rješenjem pretrage, određene domene tekstova na temelju svojstava uzoraka koji se pretražuju, a koji utječu na učinkovitost algoritama.
2. Novi algoritam za podudaranje znakovnih nizova koji koriste pristup usporedbe znakova s *online* algoritamskim rješenjem pretrage, temeljen na heurističkim pravilima o pozicijama znakova u uzorku koji se pretražuje. Za svaki pomak uzorka naspram teksta koriste se dostupne informacije o odgovarajućim indeksima znakova uzorka za preskakanje nepotrebnih usporedbi znakova.

1.6. Struktura

Ova doktorska disertacija je podijeljena u dvije tematske cjeline kroz četiri poglavlja.

Prvo poglavlje odnosi se na teorijske osnove, daje pregled osnovnih postavki vezanih uz istraživanje. Opisana je motivacija istraživanja i određuje se područje i pregled dosadašnjih istraživanja. Nakon toga predstavljene su hipoteze istraživačkog rada i opisane metodologije korištene u istraživanju.

Drugo poglavlje daje osvrt na teorijsku osnovu problematike podudaranja nizova. Opisane su osnovne tehnike pretrage, načini opisivanja algoritamske složenosti i izražavanja učinkovitosti. Zatim su opisani načini odabira skupova podataka korištenih u eksperimentalnom dijelu istraživanja i važnost entropije kao mjere iz teorije informacija u provedbi ovog istraživanja.

U trećem poglavlju opisana je osmišljena metodologija izgradnje domenskih modela vrednovanja algoritama za egzaktno podudaranje znakovnih nizova, načini izbora reprezentativnih tekstova i odabira algoritma. Napravljeno je eksperimentalno istraživanje koje uključuje prikupljanje rezultata pretrage izraženih metrikom broja usporedbi znakova, primjenu entropije na odabranu metriku, pripremljeni su podaci za analizu i isti su predstavljeni u formi modela za odabir najučinkovitijeg algoritma za pretragu uzoraka u znakovnim nizovima. Nakon izgradnje metodologije ista je vrednovana, a proces vrednovanja detaljno je opisan.

U četvrtom poglavlju predstavljen je novi algoritam za egzaktnu usporedbu nizova s opisom programske logike i osmišljenih pravila za pomak uzorka koji su osnova novog algoritma. Provedeno je eksperimentalno istraživanje i prikupljeni su rezultati istog. Nakon opisa novog algoritma opisan je i njegov način vrednovanja.

Zadnje poglavlje sadrži zaključak rada, kao i popis ostvarenih znanstvenih doprinosa te plan daljnjeg istraživanja.

2. TEORIJSKE OSNOVE

U ovome poglavlju opisane su teorijske osnove podudaranja znakovnih nizova. Napravljen je detaljan pregled načina kako se pretražuju uzorci unutar teksta, detaljno je objašnjena algoritamska složenost i načini ocjene učinkovitosti algoritma, zatim je opisan način odabira skupa adekvatno reprezentativnih podataka koji se koriste u analizi. Na kraju teorijskog dijela opisana je entropija te njeno korištenje u teoriji informacija kao mjere neodređenosti koja je osnova postavljene hipoteze nove metodologije za izgradnju modela vrednovanja algoritama na temelju svojstava uzoraka koji se pretražuju, a koji utječu na učinkovitost algoritama što je potvrđeno u eksperimentalnom dijelu disertacije.

Za potrebe ovoga izrade ovoga rada koristiti će se sljedeća tablica notacije (Tablica 2.1.):

Tablica 2.1. Tablica notacije

Oznaka	Opis
P	uzorak
T	tekst
∑	konačni skup znakova abecede
x	niz
y	niz
z	niz
m	duljina uzorka
n	duljina teksta
O	veliko O
Ω	veliko Ω
Θ	veliko Θ
o	malo o
bp	bazni parovi
z	z-vrijednost
ε	granica pogreške
N	veličina populacije
p	udio populacije
H	entropija

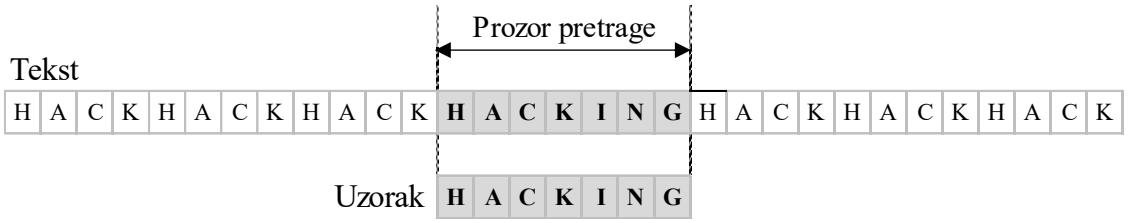
Algo	algoritam kojim je pretraga izvršena
Comp	broj usporedbi znakova u procesu pretrage
Pattern	traženi uzorak
PattEnt	vrijednost entropije
PattEntRound	zaokružena vrijednost entropije
PatEntClass	predstavlja razred entropije
CC	broj usporedbi znakova
r	Pearsonov koeficijent korelacije

2.1. Podudaranje znakovnih nizova

Podudaranje znakovnih nizova rješenje je problema pronalaska svih pojavljivanja datog uzorka u tekstu. Uzorak (engl. *pattern*) se označava znakom *P*, a tekst se označava znakom *T*. Zadatak podudaranja nizova je pronaći sva pojavljivanja danog uzorka $P=p_1p_2...p_m$ u tekstu $T=t_1t_2...t_n$ gdje su *P* i *T* nizovi konačnog skupa znakova ∑. Za nizove *x*, *y* i *z* kaže se da je niz *x* prefiks niza *xy*, sufiks niza *yx* i faktor niza *xyz*. Duljina uzorka *P* označava se malim slovom *m*, a duljina teksta *T* malim slovom *n*, gdje su $m, n > 0$ i $m \leq n$ [10], [49].

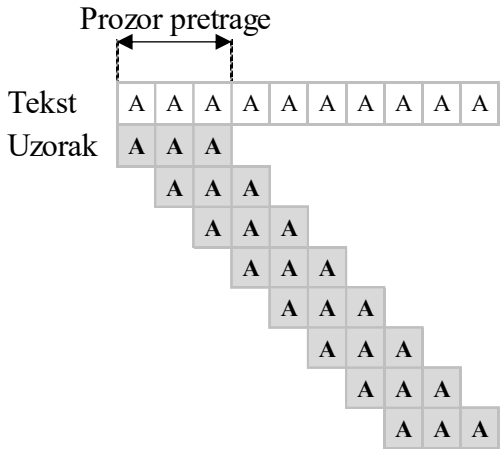
Proces usporedbe vrši se tako da se prvo poravna lijevi kraj prozora pretrage (engl. *search window*) i teksta. Prozor pretrage jednak je duljini uzorka (*m*) od lijevog do desnog kraja teksta. Nakon poravnanja prozora pretrage i teksta uzorak se pretražuje na način da se uspoređuju znakovi teksta *T* sa znakovima uzorka *P*. Kada usporedimo dva znaka, oni se podudaraju ako su jednaki. Inače se ne podudaraju. Ova se aktivnost naziva pokušaj. Nakon potpunog podudaranja (ili nepodudaranja) uzorka s tekstom, prozor se pomiče udesno. Postupak se ponavlja sve dok desni kraj prozora uzorka ne dosegne desni kraj teksta. Većina algoritama za egzaktnu usporedbu znakovnih nizova u stanju su učinkovito preskočiti dio usporedbi znakova prepoznajući strukturu uzorka ili teksta u stvarnom vremenu [10].

Algoritmi za pretragu nizova, ovisno o načinu kako se uzorak pretražuje unutar teksta, generalno se mogu podijeliti na tri načina: način pretrage prefiksa (engl. *prefix searching*), način pretrage sufiksa (engl. *sufix searching*) i način pretrage faktora (engl. *factor searching*) [33], [49]. Prozor pretrage uzorka „HACKING“ na primjeru teksta prirodnog jezika prikazan je na slici 2.1.



Slika 2.1. Prozor pretrage uzorka HACKING za domenu prirodnog jezika

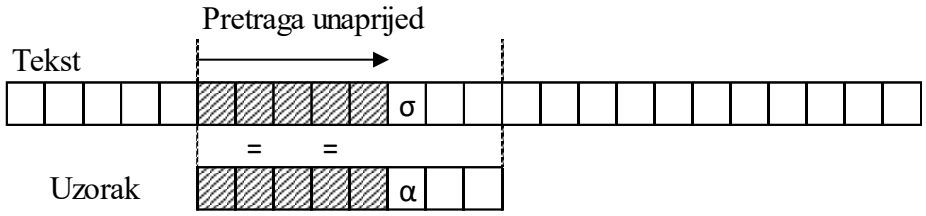
Osnovni algoritam za podudaranje znakovnih nizova je naivni (engl. *Naive, brute-force*) algoritam. Naivni algoritam vrši poravnanje lijevog kraja uzorka s lijevim krajem prozora pretrage teksta nakon čega uspoređuje znak po znak s lijeva na desno. Usporedba znakova izvršava se sve dok uzorak nije pronađen ili dok uzorak nema više znakova za usporedbu. Algoritam nema fazu predobrade. Faza pretrage ima *O(mn)* vremensku složenost neovisno o veličini abecede (∑) (vremenska složenost *O*, jedna od metrika učinkovitosti algoritama, opisana je u narednim poglavljima) [50], [51]. Primjer naivnog algoritma za pretraga uzorka AAA u tekstu AAAAAAAAAA (DNA domena) prikazan je na slici 2.2. Za dani uzorak (*m=3*) i tekst (*n=10*) za pronalazak svih uzoraka u tekstu potrebne su ukupno 24 usporedbe znakova.



Slika 2.2. Primjer pretrage uzorka pomoću naivnog algoritma

2.1.1. Pretraga prefiksa

Prvi način pretrage prefiksa (engl. *prefix searching*) pretraživanje unutar prozora vrši prema naprijed tj. s lijeva na desno, što je uobičajeni način zapisa većine pisama prirodnih jezika, očitavajući sve znakove teksta jedan iza drugog (Slika 2.3.). Za svaku poziciju prozora traži se najdulji prefiks prozora koji je ujedno prefiks uzorka [49].

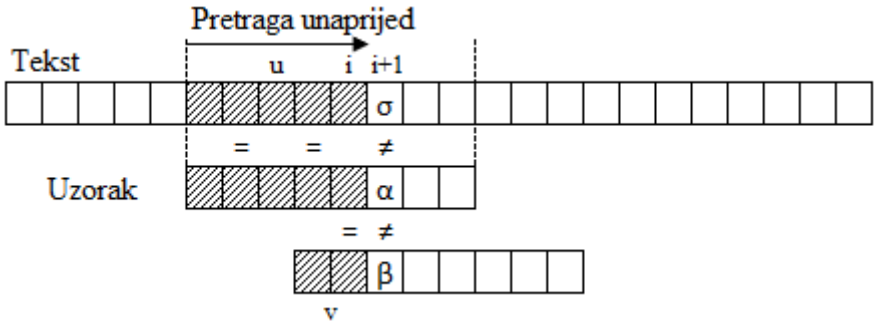


Slika 2.3. Pretraga prefiksa uzorka

Uzorak je pronađen ako se očitavajući tekst došlo do pozicije gdje je poznata duljina najdužeg sufiksa teksta koji odgovara prefiksu uzorka te kada je duljina sufiksa teksta jednaka duljini uzorka. Osnovni problem je kako pronaći učinkoviti način izračuna duljine najdužeg sufiksa teksta koji je jednak duljini uzorka nakon očitanja idućeg znaka teksta. Dva su načina rješenja tog problema. Prvi način rješenja problema je nalaženje efektivnog mehanizma za izračun najduljeg sufiksa očitavajući tekst koji je ujedno i prefiks uzorka. Drugi način je da se očitavajući tekst najprije sačuvaju svi skupovi prefiksa uzorka koji su ujedno i sufiksi očitanog teksta. Nakon očitanja pojedinog znaka teksta treba ažurirati skup prefiksa [49]. Najpoznatiji algoritam iz prvog predloženog rješenja problema je Knuth-Morris-Pratt (KMP) algoritam [54] koji je poboljšana verzija Moris-Pratt algoritma, dok su najpoznatiji algoritmi iz drugog predloženog rješenja Shift-And i Shift-Or algoritmi koji pripadaju bit-paralelnom pristupu usporedbe znakova [60], [86].

Knuth-Morris-Pratt (KMP) algoritam za egzaktno podudaranje znakovnih nizova u procesu pretrage koristi prvi način pretrage prefiksa. Algoritam ima fazu predobrade uzorka čiji je cilj izraditi tablicu s podacima sljedećih pozicija pomaka uzorka u slučaju nepodudaranja. Knuth-Morris-Pratt ima linearno vrijeme izvršavanja [10], [54], [87].

Knuth-Morris-Pratt algoritam za svaki znak teksta očitava duljinu najduljeg prefiksa uzorka koji je ujedno i sufiks teksta. Niz $v\beta$ je mogući prefiks uzorka (p) koji ujedno može biti i najdulji prefiks uzorka, a koji je ujedno sufiks od $t_1 \dots t_{i+1}$ niza znakova teksta. Niz v je sufiks od u i naziva se granica od u, a označava se $b(u)$. Podniz prozora usporedbe teksta sa znakovima uzorka je u. Nakon pomaka uzorka znak β mora biti jednak t_{i+1} (σ na slici). Znak β se razlikuje od α koji se razlikuje od σ (odnosno t_{i+1}) (Slika 2.4.) [14], [49].



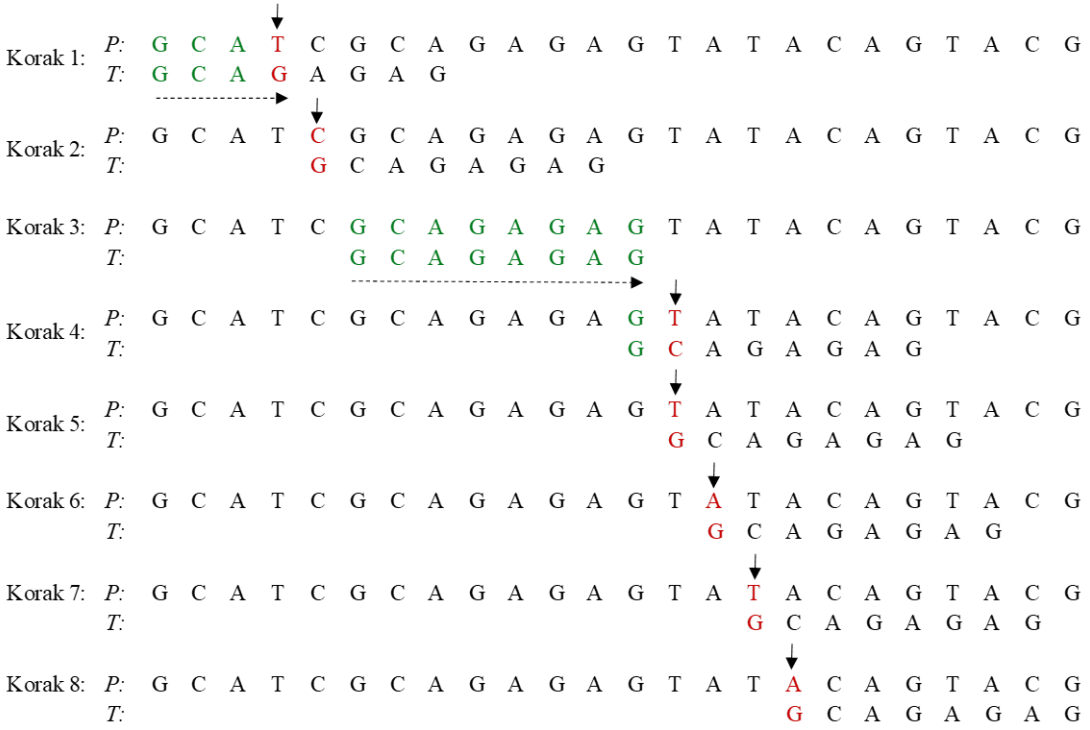
Slika 2.4. Pomak Knuth-Morris-Pratt algoritma

Originalna ideja algoritma koju su osmislili Morris i Pratt [88] imala je za cilj izračunati najdulju granicu *b(u)* za svaki prefiks *u* uzorka. Nakon izračuna najdulje granice, u je najdulji prefiks uzorka takav da je ujedno i sufiks od $t_1 \dots t_i$ niza znakova teksta.

Nakon očitavanja znaka teksta $\sigma = t_{i+1}$ vrijedi sljedeće: ako je $\sigma = p_{|u|+1}$ (označeno α na slici 2.4.) tada je novi najdulji prefiks $up_{|u|+1}$, ako je $\sigma \neq p_{|u|+1}$, tada se uspoređuje σ sa $p_{|b(u)|+1}$, ako je $\sigma = p_{|b(u)|+1}$ tada je $b(u)p_{|b(u)|+1}$ novi najduži prefiks uzorka koji je ujedno i sufiks od $t_1 \dots t_{i+1}$ niza znakova teksta, ako je $\sigma \neq p_{|b(u)|+1}$, tada se uspoređuje $\sigma = p_{|b(b(u))|+1}$. Ovo se ponavlja sve dok σ slijedi granicu ili dok više nema granica tj. dok novi najdulji prefiks nije prazan niz ε [49].

Knuth je predložio poboljšanje na način da faza predobrade ima za cilj izračunati za svaki prefiks *u* uzorka najdulju granicu *v* takvu da zadovoljava uvjet $p_{|u|+1} \neq p_{|v|+1}$. Knuth-Morris-Pratt dobar je za kratke uzorke s manjom abecedom poput DNA domene tekstova. Knuth-Morris-Pratt algoritam ima vremensku i prostornu složenost faze predobrade *O(m)*, dok je vremenska složenost faze pretrage *O(n+m)* [14], [49].

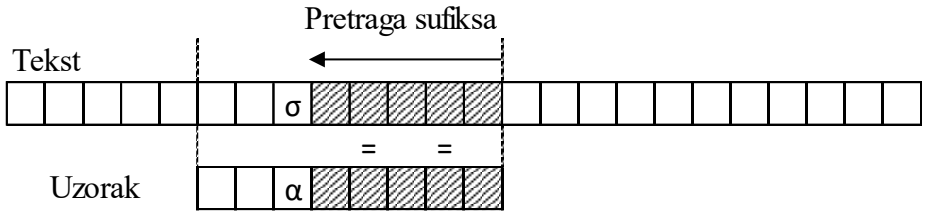
Primjer pretrage Knuth-Morris-Pratt algoritmom prikazan je na slici 2.5. [89].



Slika 2.5. Primjer pretrage Knuth-Morris-Pratt algoritma

2.1.2. Pretraga sufiksa

Drugi način pretrage sufiksa (engl. *sufix searching*) pretraživanje unutar prozora vrši unatrag očitavajući najdulji sufiks prozora koji je ujedno i sufiks od uzorka (Slika 2.6.). Ovaj način usporedbe omogućuje u prosjeku manji broj usporedbi znakova zbog učinkovitijeg preskakanja nepotrebnih usporedbi nego algoritmi koji pripadaju načinu pretrage prefiksa [49].



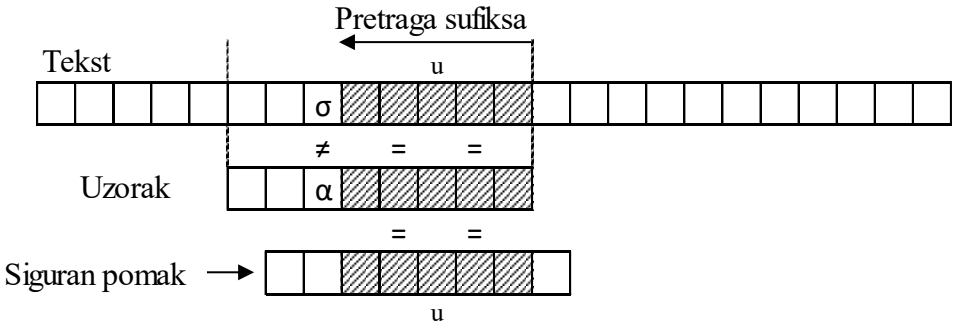
Slika 2.6. Pretraga sufiksa uzorka

Najpoznatiji algoritam iz ove grupe je Boyer-Moore [45], zatim njegova pojednostavljena verzija Horspool [59] i Sunday algoritam [56].

Boyer-Moore (BM) algoritam se smatra najefikasnijim algoritmom za uspoređivanje nizova i postao je *de facto* standardni algoritam koji se koristi u raznim sustavima za pretragu. Njegova osnovna verzija se često primjenjuje u programima za uređivanje teksta, pogotovo za korištenje naredbi pretrage i zamjene (engl. *search and substitute*). Algoritam ima fazu predobrade uzorka u kojoj se izrađuje tablica s podacima o preskakanju znakova u slučaju nepodudaranja za svaki znak uzorka. U fazi pretrage najprije se napravi poravnanje uzorka i teksta nakon čega se vrši usporedba s desna na lijevo. Usporedba se obavlja znak po znak. U prvoj iteraciji usporedbi, ako se znakovi na poziciji podudaraju, usporedba se nastavlja sve dok se ne dogodi nepodudaranje znakova. Kada se dogodi nepodudaranje, uzorak se pomiče na desno relativno od teksta i usporedba se nastavlja s desnog kraja uzorka. Boyer-Moore algoritam dobar je za duge uzorke s velikom abecedom poput prirodnog jezika [45].

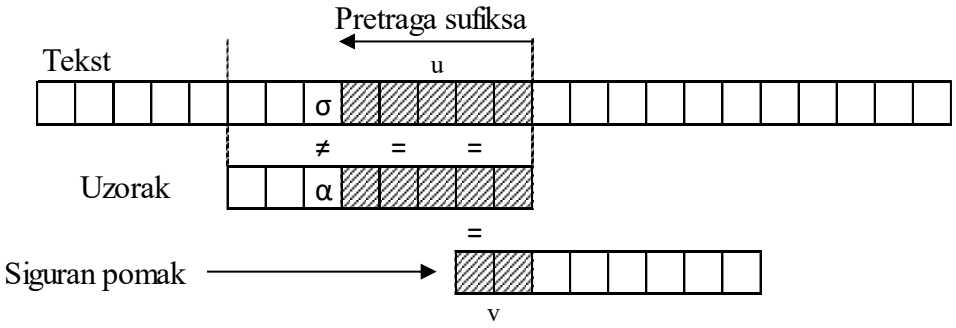
Boyer-Moore algoritam u svome radu izračunava tri funkcije pomaka d_1, d_2 i d_3 iz kojih proizlaze tri slučaja. Svrha funkcija pomaka je brža pretraga i preskakanje nepotrebnih usporedbi znakova. Za svaku situaciju potrebno je pročitati sufiks u prozora pretrage koji je ujedno i sufiks uzorka. Za slučaj da se dogodi nepodudaranje znaka teksta σ sa znakom uzorka α kao na slici 2.6. primjenjuje se određeni slučaj.

U prvom slučaju, sufiks u pojavljuje se na drugoj poziciji kao faktor uzorka. Tada se vrši siguran pomak prozora pretrage tako da se sufiks u teksta podudara sa sljedećom pojavom sufiksa u uzorka (Slika 2.7.). Ideja je izračunati za svaki sufiks uzorka udaljenost do pozicije njegovog pojavljivanja unatrag u uzorku. Ova se funkcija naziva d_1. Ako se sufiks u uzorka ne pojavljuje ponovno u uzorku tada su u i d_1 povezani s veličinom m cijelog uzorka. Uzorak se pomiče na sljedeću poziciju u [49].



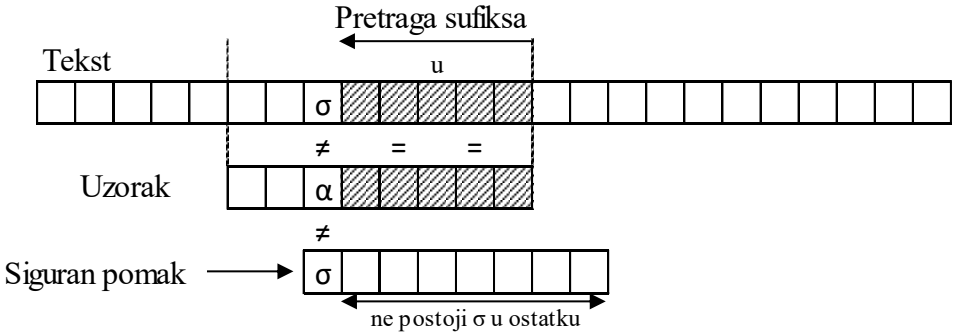
Slika 2.7. Prvi slučaj pomaka Boyer-Moore algoritma

U drugom slučaju sufiks *u* se ne pojavljuje niti na jednoj poziciji kao faktor uzorka. Ovo ne mora značiti da su se stvorili preduvjeti za siguran pomak prozora pretrage jer se može dogoditi situacija kao na slici 2.8. Sufiks *v* od *u* može biti i prefiks uzorka. Za rješenje ove situacije izračunava se druga funkcija d_2 za sve sufikse uzorka. Uzorak se pomiče za najdulji prefiks uzorka koji je također sufiks od *u*. Funkcijom d_2 označava se za svaki sufiks *u* uzorka duljina najduljeg prefiksa *v* uzorka koji je također sufiks od *u* [49].



Slika 2.8. Drugi slučaj pomaka Boyer-Moore algoritma

U trećem slučaju se prilikom pretrage unatrag dogodi nepodudaranje na znaku σ. Ako se prozor pretrage pomakne pomoću prve funkcije d_1, a nepodudarni znak se ne poravna sa znakom σ u uzorku napraviti će se nepotrebna usporedba novog prozora pretrage (Slika 2.9.). U ovome slučaju izračunava se funkcija d_3 kojom se osigurava da znak σ nakon pomaka odgovara znaku σ uzorka prilikom sljedeće usporedbe. Uzorak se pomiče na sljedeću poziciju σ uzorka. Funkcija označava udaljenost krajnje desne pozicije svakog znaka σ korištene abecede do kraja uzorka. Ako znak σ ne postoji u uzorku udaljenost se poistovjećuje s *m* [49].

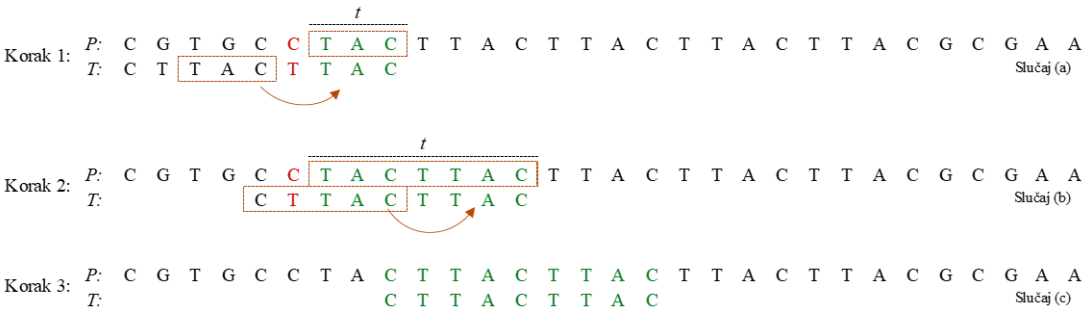


Slika 2.9. Treći slučaj pomaka Boyer-Moore algoritma

Da bi se pomakao prozor nakon očitanja u i nepodudaranja na σ, Boyer-Moore uspoređuje dva pomaka. Prvi pomak je maksimum između pomaka dobivenih funkcijama $d_1(u)$ i $d_3(\sigma)$, budući da se želi poravnati sufiks u sa sljedećom pojavom u uzorku, znajući pri tome da znak σ teksta mora odgovarati znaku σ uzorka. Drugi pomak je minimum između rezultata prethodnog maksimuma ($d_1(u)$ i $d_3(\sigma)$) i razlike $m\text{-}d_2(u)$, to je maksimalni sigurni pomak koji se može napraviti. Ako se prilikom usporedbe znakova prozora teksta i uzorka dođe do početka prozora znači da je uzorak pronađen. Ako je uzorak pronađen koristi se samo drugi slučaj ili funkcija d_2 za pomak prozora pretrage [49].

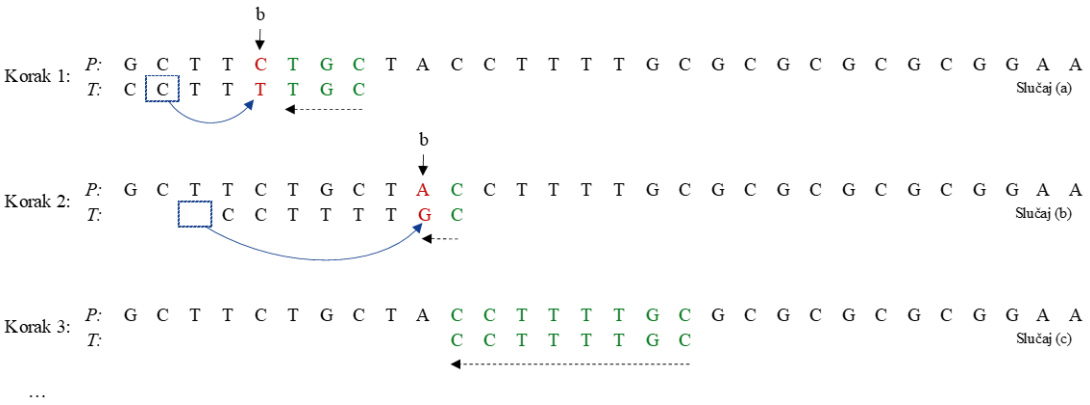
Za siguran pomak uzorka na novu poziciju usporedbe uslijed nepodudaranja znakova Boyer-Moore algoritam koristi dvije različite tehnike pomaka. Prva tehnika koju Boyer-Moore koristi je pravilo pomaka dobrog sufiksa (engl. *good suffix shift rule*), a druga je pravilo pomaka lošeg znaka (engl. *bad character shift rule*) [10], [76].

Primjena pravila dobrog sufiksa opisuje se primjerom sa slike 2.10. Pravilo pomaka dobrog sufiksa promatra podniz t teksta koji se podudara sa sufiksom uzorka, a radi na principu funkcija pomaka d_1 i d_2. Moguće je preskočiti poravnanja sve dok se ne dogodi slučaj a) kada se t pojavljuje u cijelosti lijevo unutar uzorka, ili slučaj b) kada se prefiks uzorka podudara sa sufiksom od t ili slučaj c) kada se uzorak premješta iza t [90].



Slika 2.10. Primjer primjene pravila dobrog sufiksa Boyer-Moore algoritma

Primjena pravila lošeg znaka najbolje se opisuje sljedećim primjerom sa slike 2.11. U primjeru se vrši usporedba uzorka i teksta sve dok se ne dogodi nepodudaranje. Radi na principu funkcije pomaka d_3. Neka je b nepodudarni znak teksta. Moguće je napraviti preskakanja poravnanja sve dok se ne dogodi slučaj a) do podudaranja sa suprotnim znakom uzorka, ili slučaj b) kada se uzorak premješta iza b, ili slučaj c) ako nema nepodudaranja nema preskakanja usporedbe. Ključ pravila lošeg znaka je pomicati uzorak za onoliko znakova koliko je moguće. Nakon pomicanja, usporedba uzorka i teksta počinje ponovno s desnog kraja uzorka [10], [41], [90].

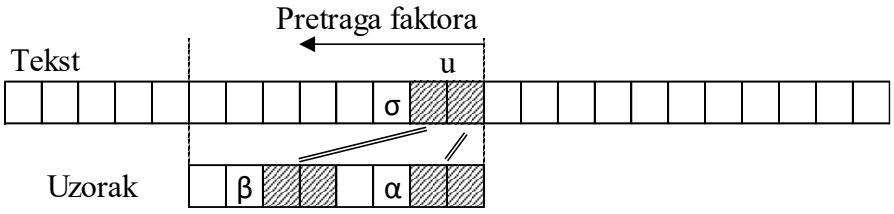


Slika 2.11. Primjer primjene pravila lošeg znaka Boyer-Moore algoritma

Pravilo pomaka dobrog sufiksa i pravilo pomaka lošeg znaka su heurističke tehnike koje u slučajevima nepodudaranja znakova uzorka i teksta pomiču uzorak za najveći mogući iznos dobiven primjenom određenog pravila. Nijedna od metoda ne koristi više od $m+n$ usporedbi znakova i kao takve se izvršavaju u linearnom vremenu. Pravilo pomaka lošeg znaka u praksi je manje efektivno pogotovo za manje nizove. Zbog toga je uvedeno pravilo pomaka dobrog sufiksa [10], [76].

2.1.3. Pretraga faktora

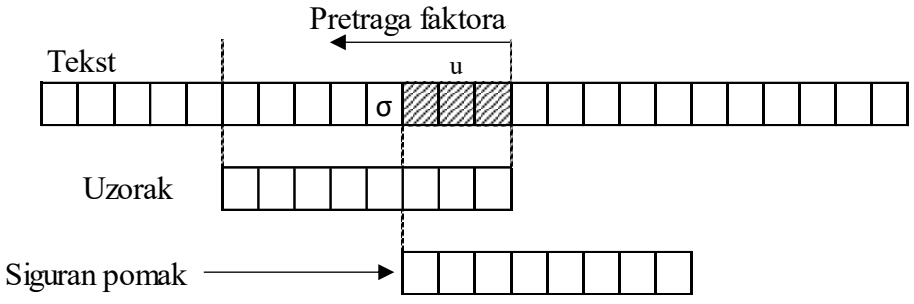
Pretraživanje unutar prozora vrši se unatrag očitavajući najdulji sufiks prozora koji je ujedno i faktor uzorka (Slika 2.12.). Ovakvim načinom pretrage teži se prema sublinearnim i optimalnim algoritmima koji su u praksi uspješniji od algoritama iz ostalih načina pretrage za dovoljno dug uzorak. Najveća prepreka u realizaciji ovakvih algoritama je način prepoznavanja faktora unutar uzorka [49].



Slika 2.12. Pretraga faktora uzorka

Ovaj način karakterizira jednostavnost pomaka prozora pretrage. Ako se pretpostavi da je unatrag očitan faktor *u* uzorka i da se nepodudaranje desilo na idućem znaku *σ* to znači da niz *σu* nije više faktor uzorka tako da ga uzorak više ne može niti sadržavati. Stoga se prozor pretrage može sigurno pomaknuti odmah iza *σ* kao što je prikazano na slici 2.13. [49].

Najpoznatiji algoritmi u ovoj grupi su Backward Dawg Matching algoritam (BDM) [58], Backward Nondeterministic Dawg Matching algoritam (BNDM) [91] i Backward Oracle Matching algoritam (BOM) [92].



Slika 2.13. Pomak prozora pretrage pretragom faktora uzorka

2.2. Algoritamska složenost i izražavanje učinkovitosti algoritama

Algoritamska složenost je mjera koja uspoređuje algoritme na način da se promatra izvršavanje logike algoritma ne uzimajući pri tome u obzir detalje poput programske implementacije algoritma u određenom programskom jeziku, hardverske specifikacije računala na kojem se algoritam izvršava, broja instrukcija procesora i sl. Složenost se odnosi na sami algoritam, način na koji obrađuje podatke kako bi riješio zadani problem. To je problem dizajna softvera na "razini ideje". Usporedba dva algoritma na način da se analizira vrijeme izvršavanja (obično u milisekundama) pojedinog algoritma na računalu koristi se u slučajevima kada je jednak broj koraka izvršavanja dva algoritma odnosno kada je jednaka njihova složenost [23].

Složenost se može promatrati kao maksimalni broj jednostavnih instrukcija koje neki program može izvršiti. Neke programske instrukcije mogu ostati neprebrojane, a fokus mogu biti one instrukcije koje se izvode najveći broj puta. Takve se instrukcije nazivaju dominantnim. Broj dominantnih instrukcija ovisi o specifičnim ulaznim podacima [8].

Primjeri računalnih instrukcija koje su obuhvaćene analizom algoritamske složenosti su računske instrukcije poput zbrajanja i množenja, pretrage podataka u memoriji računala ili izvođenje regularnih izraza za pronalaženje uzorka u nizu. Moguće je imati neučinkovit algoritam koji se izvodi na najboljem hardveru kako bi brzo dao rezultat. Međutim, s velikim skupovima ulaznih podataka ograničenja hardvera će postati očita. Stoga je poželjno optimizirati algoritam prije razmišljanja o nadogradnji hardvera. Ako se poveća volumen ulaznih podataka, bolji hardver ne može kompenzirati algoritamsku neučinkovitost. Zbog toga je algoritamska složenost definirana u terminima asimptotskog ponašanja [93], [94].

Broj programskih instrukcija za izvršavanje algoritma, odnosno vrijeme izvršavanja može varirati od slučaja do slučaja. Primjerice, ako je potrebno uzlazno sortirati polje koje sadrži cjelobrojne vrijednosti, za početno polje poredano silazno potrebna je zamjena svakog elementa polja. U takvom slučaju potrebno je mnogo više koraka nego u svim ostalim slučajevima sortiranja elemenata polja. Takav slučaj naziva se najgori mogući slučaj (engl. *worst-case scenario*). Analiza kojom se algoritam tretira na najgori mogući način kako bi se uvidjela njegova izvodivost u slučaju, primjerice jako veliki skup ulaznih podataka, naziva se analiza najgoreg mogućeg slučaja (engl. *worst-*

case analysis). Za potrebe mjerenja broja koraka algoritma definira se funkcija rasta *f(n)* kao broj koraka potrebnih za izvršavanje algoritma. Analiza složenosti vodi računa o tome što se događa s funkcijom rasta broja koraka ako se poveća broj ulaznih podataka *(n)*. U slučaju kada se algoritam brže izvršava nad velikim skupom podataka u odnosu na drugi algoritam pretpostavka je da će se izvršavati brže i s manjim skupom podataka. Za određivanje asimptotskog ponašanja funkcije rasta izbacuju se svi uvjeti koji rastu sporo, dok se zadržavaju oni koji rastu brzo i brzo postaju sve veći [93], [94].

Postoje dva glavna pristupa za izražavanje učinkovitosti algoritama obzirom na atribut brzine koji je najvažniji aspekt učinkovitosti algoritma te se obično podrazumijeva pod pojmom učinkovitosti algoritma. Prvi pristup je formalni (teorijski) i on analizira (*a priori*) složenost algoritma kroz vremensku (vremenska složenost) i prostornu analizu (prostorna složenost). Vremenska složenost (engl. time complexity) je proces određivanja ukupnog potrebnog vremena za izvršavanje algoritma. Prostorna složenost (engl. space complexity) je proces određivanja potrebne količine memorije za izvršavanje algoritma s obzirom na veličinu ulaza. Drugi je pristup empirijski i analizira (*a posteriori*) upotrebu računalnih resursa kroz prostornu i vremensku složenost (potrebna radna memorija i vrijeme izvršavanja). Empirijske metrike, kao što su: vrijeme izvršavanja algoritma obično prikazano u milisekundama, korištenje procesora i memorije, privremeno korištenje diska, dugoročno korištenje diska, potrošnja energije itd., obično se koriste za analizu prostorne složenosti algoritma izražavajući tako njegove performanse [17], [22], [40], [95]–[100].

Najčešće korištena asimptotska notacija koja predstavlja algoritamsku složenost iz formalnog pristupa je *Big O* notacija (veliko *O*). *Big O* ili drugim nazivom Landau notacija, predstavlja složenost algoritma prikazanog kao funkcija ulazne veličine koja opisuje gornju granicu složenosti algoritma. Gornja granica složenosti algoritma je najgora moguća izvedenica trenutnog algoritma za koju se smatra da trenutna izvedenica algoritma nikada neće biti lošija od najgore moguće izvedenice. Koristi se za usporedbu dvije asimptotske funkcije gdje jedna funkcija asimptotski nije viša od druge (operator ≤). *Big O* notacija je korisna zbog toga što daje informaciju da algoritam nikada neće biti sporiji od određene granice i kao takva predstavlja vrijednu informaciju da li je algoritam dovoljno dobar [8], [22], [23].

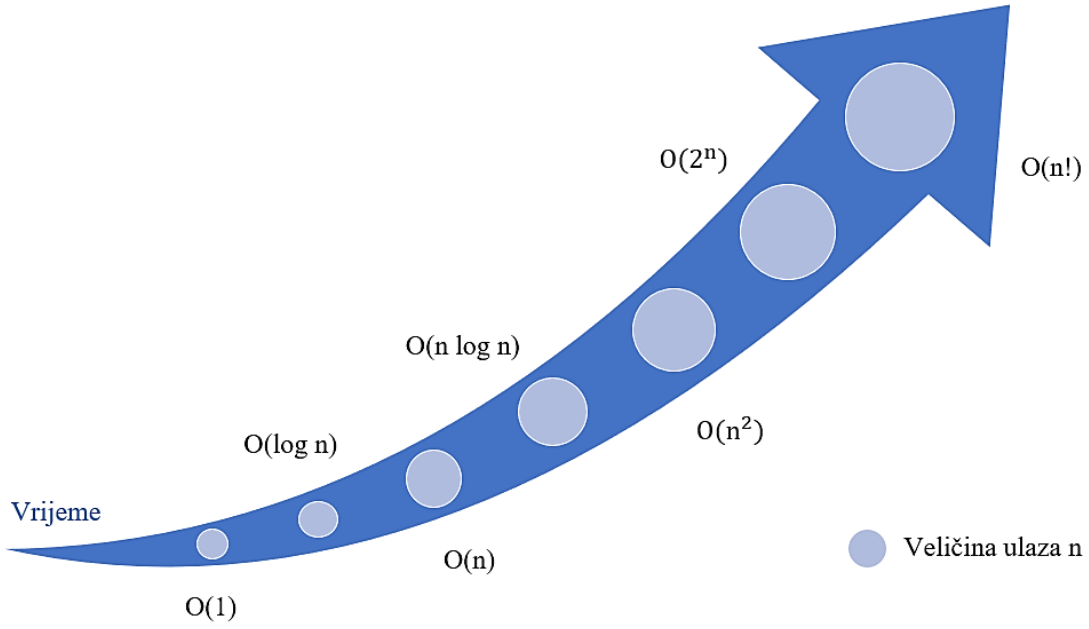
Uz *Big O* notaciju lako je usporediti različite algoritme jer notacija jasno izražava kako se algoritam skalira kada se veličina ulaznih podataka povećava. To se često naziva

redoslijed rasta. Redoslijedi rasta kod *Big O* notacije su: *O(1)* je konstantno vrijeme izvođenja, *O(n)* je linearni rast, *O(log(n))* je logaritamski rast, $O(n^2)$ je kvadratni rast, $O(2^n)$ je eksponencijalni rast i *O(n!)* je faktorijelski rast složenosti [8], [22], [23].

Redoslijedi rasta *Big O* notacije mogu se usporediti od najboljeg prema najgorem prema sljedećem algebarskom izrazu prikazanom jednadžbom (2.1):

$$O(1) < O(\log n) < O(\sqrt{n}) < O(n) < O(\mathrm{n}\log n) < \\ < O(n^2) < O(n^3) < O(2^n) < O(10^n) < O(n!) \tag{2.1}$$

Redoslijede rasta složenosti algoritama pomoću *Big O* notacije, gdje za povećanje ulaznih podataka raste algoritamska složenost, moguće je prikazati i pomoću grafičkog prikaza sa slike 2.14.



Slika 2.14. Big-O grafički prikaz složenosti

Moguće su i ostale notacije poput Ω (veliko omega) −asimptotska notacija za označavanje donje granice funkcije, tj. složenosti kojom algoritam nikada neće biti bolji od određene vrijednosti za najbolji mogući slučaj, Θ −predstavlja trenutnu vrijednost gornje i donje granice vremena izvršavanja algoritma koja se koristi za analizu složenosti algoritma u prosječnom slučaju, o (malo o) −asimptotska notacija za označavanje gornje granice stope rasta vremena izvođenja algoritma koja nije asimptotski čvrsta, a koja

koristi se za usporedbu dvije asimptotske funkcije gdje je jedna funkcija asimptotski manja od druge (operator <) [8], [23], [93], [94].

Metodologija razvijena u ovoj doktorskoj disertaciji usmjerena je prema procjeni brzine algoritama korištenjem metrike usporedbi znakova ili CC (*character comparison*) metrike [76]. Iako su vremenska i prostorna složenost neovisne metrike jer ne ovise o računalnoj arhitekturi, programskom jeziku i drugim faktorima koji mogu utjecati na brzinu algoritama u disertaciji je prednost dana metrici usporedbe znakova koja je u odnosu na vremensku i prostornu složenost jednostavnija za praktičnu implementaciju prilikom izgradnje modela za odabir algoritma.

Metrika usporedbe znakova izražava ukupan broj uspoređenih znakova uzorka sa znakovima teksta te je kvantitativna mjera neovisna o suptilnostima implementacije algoritma, o programskom jeziku, računalnim resursima i operativnim sustavima [76]. To znači da je neovisna o platformi poput formalnih pristupa, a obuhvaća djelomično i složenost prostora te se može programski implementirati i koristiti poput empirijskih pristupa. Stoga ova metrika spada u kombinirani formalni i empirijski pristup ocjene složenosti algoritama za pretragu znakovnih nizova.

2.3. Odabir skupova podataka

Odabir skupova podataka koji se koriste u analizi različitih problema izazovan je zadatak. U strojnom učenju (engl. *machine learning*) zajednički zadatak je proučavanje i izgradnja modela koji mogu učiti i predviđati podatke. Takvi modeli funkcioniraju na način da se odluke donose na osnovu podataka. Ulazni podaci koji se koriste za izgradnju modela obično se dijele u tri skupa podataka: trening skup, skup vrednovanja i testni skup. Ovakvom podjelom osigurava se da izrađeni model točno odražava svoje ponašanje na nekorištenim podacima [101], [102].

Model se izgrađuje na osnovu podataka iz trening skupa. Sukcesivno se model vrednuje drugim skupom podataka vrednovanja. Skup testnih podataka upotrebljava se za nepristranu procjenu konačnog modela. U literaturi se često radi zamjena tako što se skup podataka vrednovanja koristi umjesto testnog skupa podataka. Ako je napravljena podjela podataka na samo dva skupa onda se testni skup podataka može promatrati kao skup podataka vrednovanja [103], [104].

Iako naša metodologija ne koristi algoritme strojnog učenja iskorištena je postojeća tehnika prema kojoj su ulazni podaci podijeljeni na dva skupa. Ulazni skup podataka izgradnje metodologije sastoji se od trening skup podataka pomoću kojih je model izgrađen i skupa vrednovanja pomoću kojih je model vrednovan [102].

Za potrebe izgradnje modela domena i novog algoritma za egzaktnu usporedbu nizova definiran je skup reprezentativnih tekstova. Prilikom odabira reprezentativnih tekstova iskorištene su dvije javno dostupne standardne kolekcije tekstova, domena DNA nizova i domena prirodnog jezika.

Biološki nizovi poput DNA mogu biti vrlo slični jedni drugima zbog svoje evolucijske veze. U svrhu zaobilaženja ove osobnosti bioloških nizova i mogućnosti precjenjivanja točnosti modela osmišljena je strategija podjele podataka u zasebne skupove analogno tehnikama strojnog učenja. Podjela ulaznih podataka napravljena je na skup podataka kojima je model izrađen i skup podataka kojima je model vrednovan.

Zahvaljujući napretku istraživanja genoma, posebno krajem 90-ih godina prošlog stoljeća kada je pokrenut projekt GenBank[5] [105], proizvedena je velika količina bioloških podataka. Domena DNA nizova, zbog dostupnosti i velike količine DNA podataka, pruža okruženje u kojem je moguće provesti istraživanje o pristupima odabira najučinkovitijih algoritama za pronalaženje uzoraka kao i razvoj novog algoritma te izvršiti testove na stvarnim podacima [9], [50], [106].

Biološki nizovi iz DNA domene imaju određene karakteristike. Duljina DNA niza izražena je u baznim parovima (*bp,* engl. *base pair*) i varira od nekoliko tisuća do nekoliko milijuna, pa čak i milijardi *bp*. Konačni skup znakova za DNA domenu je $\sum = \{A, C, G, T\}$ s veličinom od 4 znaka. Odsječak teksta DNA domene prikazan je na slici 2.15. [14], [49].

[5] Baza podataka sekvenci GenBank je zbirka otvorenog pristupa svih javno dostupnih sekvenci nukleotida i proteina. Podatke u Genbak reproducira i održava Nacionalni centar za biotehnološke informacije u sklopu međunarodne suradnje pri razmjeni podataka o nukleotidnim sekvencama.

```
GAAACGCCGTAGCGCCGATGGTAGTGTGGGGTCTCCCCATGCGAGAGTAGGGAACTGCC
AGGTATCAAATTAAGCAGTAAGCCGGTCATAAAACCGGTGGTTGTAAAAGAATTCGGTG
GAGCGGTAGTTCAGTCGGTTAGAATACCTGCCTGTCACGCAGGGGGTCGCGGGTTCGAG
TCCCGTCCGTTCCGCCACTTACTAAGAAGCCTCGAGTTAACGCTCGAGGTTTTTTTTCG
TTTGTATTTCTATTATTGCCAAAATCGCAAAAATCCTCTGCGTTTTACGCCATTTTTCC
GCAACAGTCTGAAGCCCATAATCACCTCAGTTAACGAAAATAGCATTAAAAGAGGCATA
TTATGGCTATCCCTGCATTTGGTTTAGGCACTTTCCGTCTGAAAGACGACGTTGTTATT
TCATCTGTGAAAACGGCGCTTGAACTTGATTATCGCGCAATTGATACCGCACAAATCTA
TGATAACGAAGCCGCAGTAGGTCAGGCGATTGCAGAAAGTGGCGTGCCACGTCATGAAC
```

Slika 2.15. Odsječak teksta iz DNA domene

Još jedna od visoko dostupnih domena za provođenje istraživanja algoritama za pronalaženje uzoraka znakovnih nizova je domena prirodnog jezika (engl. *natural language*) koja je ujedno i domena u kojoj se najčešće uobičajeno koriste algoritmi za pronalaženje nizova znakova. Ova domena je također izvor velikih skupova tekstualnih podataka te je istraživanje pristupa odabira najučinkovitijih algoritama provedeno i na ovoj domeni. Svi skupovi tekstualnih podataka koji pripadaju domeni prirodnog jezika mogu biti predmet istraživanja. Kada je u pitanju prirodni jezik, to je jezik kojim se govori, piše ili ga simbolički koriste ljudi u općenitu svrhu komunikacije [107].

Domena prirodnog jezika također ima svoje karakteristike. Konačni skup znakova za domenu engleskog prirodnog jezika, koja je korištena u istraživanju, je $\sum$ = *{abeceda koja se sastoji od 26 slova, od kojih svaka ima obrazac velikih i malih slova}*. Nizovi koji se pretražuju u prirodnom jeziku su jednostavniji nego DNA nizovi zato što u sebi sadrže manji broj ponavljanja znakova. Odsječak teksta engleskog jezika domene prirodnog jezika prikazan je na slici 2.16 [14], [49].

```
In the beginning God created the heaven and the earth. And
the earth was without form, and void; and darkness was upon
the face of the deep. And the Spirit of God moved upon the
face of the waters. And God said, Let there be light: and
there was light. And God saw the light, that it was good:
and God divided the light from the darkness.
```

Slika 2.16. Odsječak teksta iz engleskog jezika

Nakon odabira reprezentativnih tekstova odabrani su i reprezentativni uzorci. U postupku uzorkovanja određeni uzorci odabrani su iz populacije tako da se pomoću njih

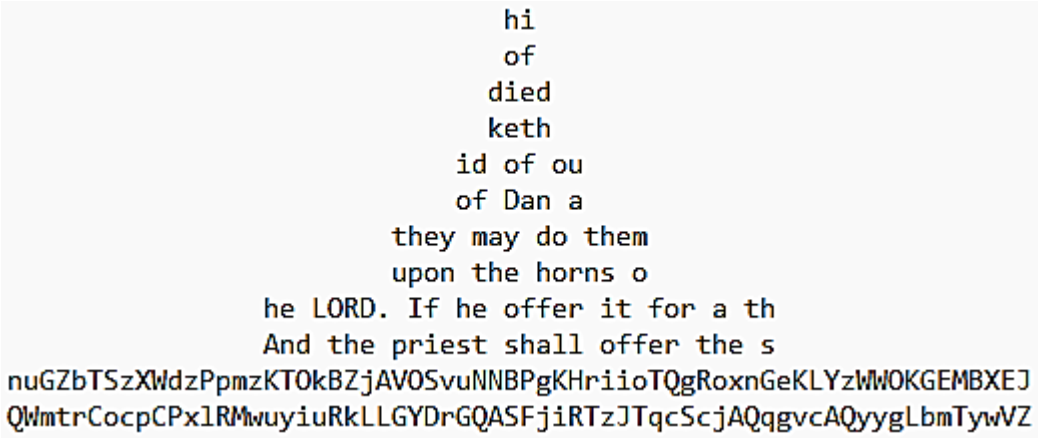
mogu izvesti zaključci o cjelokupnoj populaciji s određenim stupnjem pouzdanosti. Uzorci su odabrani na dva načina. Prvi način odabira su namjerni uzorci tj. uzorci koji su podnizovi reprezentativnih tekstova, a drugi način odabira su uzorci stvoreni od nasumično generiranih znakova abecede promatrane domene.

U istraživanjima provedenima u sklopu disertacije vodilo se računa o maksimalnoj preciznosti i izbjegavanju pristranosti. Duljina uzoraka kreće se od 2 znaka do 32 znaka sa slijedom duljina uzoraka od 2, 4, 8, 16 i 32 znaka. Duljina uzoraka do 32 znaka s navedenim slijedom duljina, uobičajena je korištena duljina uzoraka u mnogim empirijskim analizama učinkovitosti algoritama i dovoljna za vrednovanje modela [18], [51], [76]. Rad sa duljinama većim od 32 znaka kao i onim uzorcima neparnih duljina je moguć.

Primjer reprezentativnih uzoraka domene DNA prikazan je na slici 2.17. Na slici je prikazan samo dio uzoraka iz skupa uzoraka kojima je model izgrađen. Prikazani su jedan redak jedan uzorak.

Slika 2.17. Primjer reprezentativnih uzoraka DNA domene

Primjer reprezentativnih uzoraka domene prirodnog jezika korištenih u pretrazi prikazan je na slici 2.18.



Slika 2.18. Primjer reprezentativnih uzoraka domene prirodnog jezika

U cilju pravilnog odabira skupa uzoraka za izgradnju modela za bilo koju domenu primjene metodologije potrebno je osigurati određenu razinu pouzdanosti da se zaključci o populaciji donose na temelju uzorka. U pravilu je teško ili vrlo zahtjevno ispitati cjelokupnu populaciju prilikom provođenja istraživanja. Stoga su iz populacije odabrani uzorci iz kojih se mogu izvesti odgovarajući zaključci. Izračunavanje veličine reprezentativnih uzoraka nije trivijalno i važan je aspekt svake empirijske studije. Međutim, postoje određene metode za izračun veličine uzorka uz određeni stupanj pouzdanosti i moguću granicu pogreške [108]–[110].

Da bi se postigao određeni stupanj pouzdanosti izgrađenog modela, skup reprezentativnih uzoraka mora sadržavati određeni broj uzoraka koji mogu objektivno odražavati modeliranu domenu.

Veličina skupa reprezentativnih uzoraka bez obzira na strukturu skupa uzoraka (podnizovi ili nasumično generirani) određuje se jednadžbom (2.2) za neograničenu populaciju [108]–[110]:

$$n = \frac{z^2 \times p(1-p)}{\varepsilon^2} \tag{2.2.}$$

Veličina skupa reprezentativnih uzoraka određuje se jednadžbom (2.3) za konačnu populaciju [108]–[110]:

$$n' = \frac{n}{1 + \frac{z^2 \times p(1-p)}{\varepsilon^2 N}} \tag{2.3}$$

gdje je n veličina uzorka dobivena jednadžbom (2.2), z je z-vrijednost, ε je granica pogreške, N je veličina populacije, a p je udio populacije.

Najčešće korištene razine pouzdanosti su (0,9) 90%, (0,95) 95% i (0,99) 99%. U provedenom istraživanju korištena je razina pouzdanosti 0,95. Svaka od razina pouzdanosti ima odgovarajuće z-vrijednosti koje pružaju tablice na temelju odabrane razine pouzdanosti (npr. razina pouzdanosti 0,95 ima z-vrijednosti 1,96). Granica pogreške je način izražavanja pogreške uzorkovanja u mjerenju ili istraživanju tj. maksimalna udaljenost za procjenu uzorka koja odstupa od stvarne vrijednosti. Veličina populacije je ukupan broj uzoraka u studiji. U teoriji, imamo posla s neograničenom populacijom jer uzorci i tekstovi mogu imati neograničen broj znakova. Međutim, u praksi moramo ograničiti populaciju na konačan broj [50], [111], [112].

2.4. Entropija

Pojam entropije proizišao je iz područja termodinamike, a prvi je put korišten u Klausijevom (Rudolf Clausius) teoremu prema kojem toplina ne može spontano sama od sebe prelaziti s hladnijeg na toplije tijelo. Definirani teorem u to vrijeme imao je veliki efekt tako da je promijenio sveukupni pogled na znanost [113]–[118].

Primjena entropije široko je rasprostranjena u raznim znanstvenim disciplinama poput fizike i prirodnih znanosti. Posebno se može istaknuti primjena entropije u teoriji informacija. Informacija je pojam koji se susreće gotovo svugdje. Statističku strukturu informacije ili matematičku definiciju pojma informacije postavio je Claude Shannon u svome radu „Matematička teorija komunikacija“ iz 1948. godine. Prema Shannonu, mjera informacijskog sadržaja slijeda znakova količinski je pojam kojemu je jedinica bit (engl. *binary digit*). Budući da vjerojatnost može poprimiti samo vrijednosti između 0 i 1, brojčana će vrijednost informacije uvijek biti pozitivna [119]–[128].

Prema Shannonu entropija je mjera neodređenosti pridružena slučajnoj varijabli. Za Shannonovu mjeru informacije nije bitno da li je neki niz nastao nasumično ili ima neko određeno značenje. Ako se promatra neki nasumični niz, iz konteksta teorije informacija može imati maksimum informacijskog sadržaja, međutim prema Shannonu

vrijednost informacijskog sadržaja takvog niza može biti manja od niza smislenog teksta jednake duljine. Shannonova definicija sadrži samo statističku ovisnost slijeda znakova, a potpuno zanemaruje značenje. Ona dopušta kvantitativno opisivanje svih svojstava jezika koja se u svojoj biti temelje na učestalosti slova ili općenito na učestalosti znakova u nekom skupu. Smisao rečenice i gramatička ispravnost potpuno se zanemaruju na ovoj razini [111], [116].

Najjednostavniji način opisa entropije je pomoću događaja koji mogu imati različito, ali konačno stanje neizvjesnosti. Za definiciju Shannonove entropije pretpostavimo da su definirani događaji $A_1, A_2,..., A_n$ i oni čine kompletan skup. Sljedeći izraz je valjan, pri čemu je $\sum_{i=1}^{n} p_i = 1$, gdje je $p_i=p(A_i)$. Konačni sustav α sadrži sve događaje A_i, $i=1, 2,..., n$ s odgovarajućim vrijednostima vjerojatnosti p_i. Sljedeći izraz označiti će sustav α (jednadžba (2.4)) [117], [121], [129]:

$$\alpha = \begin{pmatrix} A_1 & A_2 & \dots & A_n \\ p_1 & p_2 & \dots & p_n \end{pmatrix} \tag{2.4}$$

Svaki konačni sustav opisuje neko stanje neizvjesnosti jer je nemoguće znati koje je stanje sustava u određenom vremenu. Cilj je kvantitativno izraziti takvu neizvjesnost na neki način. To znači da treba definirati određenu funkciju koja će pridružiti određeni broj sustavu α. Na taj način sustav će imati mjeru za svoju neizvjesnost [117], [121].

Shannonova mjera informacije omogućila je kvantitativno izražavanje količine informacija sadržanih u nizu kao i razmatranje međusobnih odnosa koji se dotad nisu mogli egzaktno matematički opisati. Funkcija koja kvantitativno mjeri neizvjesnost sustava naziva se *entropija sustava,* a definirana je sljedećom jednadžbom (2.5) [117], [121], [129], [130]:

$$H(p_1, p_2, \dots, p_n) = -\sum_{i=1}^{n} p_i \, log p_i \tag{2.5}$$

Entropija sustava označena je s $H(\alpha)$, n je broj znakova, a p_i je vjerojatnost njihove pojave. Ako je $p_i = 0$, slijedi da je $p_i log\, p_i = 0$. Baza logaritma teorije informacija obično je 2. Entropija je nula samo ako je jedna od vjerojatnosti $p_i=1,...,n$ jednaka 1. U tom slučaju nema neizvjesnosti jer je moguće precizno predvidjeti stanje sustava. U svakom drugom slučaju, entropija je pozitivan broj. Ako sustav α sadrži rezultate ispitivanja,

razina neizvjesnosti prije provođenja testa jednaka je entropiji sustava α. Kada se test izvrši, razina neizvjesnosti je nula [119], [121], [129], [130].

Jednadžba (2.5) odnosi se samo na onaj aspekt značenja kojom informacija ima neku novu vrijednost. Nova vrijednost može imati veće i manje efekte neizvjesnosti. Veći efekt neizvjesnosti nastaje ako u tekstu postoje rjeđi znakovi što znači da informacija postaje mjera neizvjesnosti nekog događaja. Neki znakovi teksta mogu imati veći efekt iznenađenja; primjerice znak „f“ ima manju vjerojatnost pojave nego znak „a“ implicirajući tako da informacijski sadržaj raste smanjenjem vjerojatnosti pojave nekog znaka [121], [130].

Primjenom jednadžbe (2.5), napravljen je izračun entropije na primjeru za uzorak „*TCGTAACT“*. Za izračunavanje entropije uzorka potrebno je izračunati vjerojatnost pojave svakog pojedinog znaka. Nakon brojanja pojedinog znaka u uzorku, *A = 2, C = 2, G = 1, T = 3*, vjerojatnosti su:

$$P(A) = \frac{2}{8} = 0.25 \qquad P(C) = \frac{2}{8} = 0.25$$

$$P(G) = \frac{1}{8} = 0.125 \qquad P(T) = \frac{3}{8} = 0.375$$

$$H = -\left(\frac{2}{8} \times \log_2 \frac{2}{8}\right) - \left(\frac{2}{8} \times \log_2 \frac{2}{8}\right) - \left(\frac{1}{8} \times \log_2 \frac{1}{8}\right) - \left(\frac{3}{8} \times \log_2 \frac{3}{8}\right)$$

$$H = -(-0.5) - (-0.5) - (-0.375) - (-0.53064) = 1.90563906222957$$

$$H \approx 1.91$$

Za uzorak „*TCGTAACT“* iz navedenog primjera izračunata entropija je 1,90563906222957. Entropija za drugi uzorak iz engleskog teksta *”e name of the LORD. And the LORD”* u skladu s jednadžbom (2.5) iznosi 3,698391111. Vrijednosti entropija u razvijenoj metodologiji zaokružuju se na dvije decimale (tj. entropija za uzorak „*TCGTAACT“* je 1,91, a entropija za uzorak engleskog teksta u gornjem primjeru je 3,70).

Budući da je entropija značajka upotrijebljenog jezika kojeg karakterizira skup znakova, uspoređujući dva jezika s istim brojem znakova u skupu, određeni jezik može

imati veću entropiju jer je distribucija vjerojatnosti pojave znakova sličnija normalnoj distribuciji. Primjer je prirodni jezik s latiničnom abecedom. Entropija ima najveće vrijednosti ako postoji normalna distribucija znakova. Moguća je situacija da neki znakovi nisu učestali tj. da su manje vjerojatni. U tom slučaju za reprezentativni statistički skup (tekst koji uključuje i razmake) promatra se srednja vrijednost informacijskog sadržaja po znaku ili srednja vrijednost cijelog skupa prema jednadžbi (2.5) [111].

Metodologija predstavljena u ovoj doktorskoj disertaciji temeljena je na iskorištavanju informacijskog sadržaja uzorka kroz svojstva abecede, pravila korištenja abecede i sl., tzv. svojstva tekstova koji se pretražuju. Budući da entropija izražava informacijski sadržaj uzorka koji ovisi o vjerojatnosti pojave pojedinog znaka unutar istog, vrijednost entropije uzoraka iskorištena je kao temelj izgradnje nove metodologije za odabir najučinkovitijeg algoritma za podudaranje znakovnih nizova.

3. METODOLOGIJA IZGRADNJE DOMENSKIH MODELA VREDNOVANJA ALGORITAMA ZA EGZAKTNU USPOREDBU NIZOVA

U sklopu ove doktorske disertacije provedeno je *State-of-the-art* istraživanje koje je pokazalo da postoji nedostatak sustavnih pristupa odabira najučinkovitijih algoritama za određenu domenu primjene. Kako bi se pokušao pronaći odgovarajući sustavni način odabira najučinkovitijeg algoritma osmišljen je novi pristup pomoću kojeg se može jasno i jednostavno odabrati najučinkovitiji algoritam za određenu domenu primjene [18], [31], [51], [75], [76].

Osmišljeni pristup, definiran u formi metodologije, koristi svojstva uzorka i formalne metrike što omogućuje transparentnost pri odabiru najučinkovitijega algoritma u smislu neovisnosti o računalnim resursima, programskim jezicima i operativnim sustavima. Razvijena metodologija je jednostavno primjenjiva s jasno definiranim atributima kvalitete i načinima prezentacije rezultata.

Razvijena metodologija za odabir najučinkovitijeg algoritma za egzaktno podudaranje znakovnih nizova koji koriste pristup usporedbe znakova s *online* algoritamskim rješenjem pretrage, koja je platformski neovisna, definira sve detalje izgradnje modela. Pod detaljima izgradnje modela podrazumijevaju se koraci od odabira skupa tekstova do klasifikacije i rangiranja algoritama izgrađenog modela.

U ovom radu istraživanje je oblikovano kroz četiri faze:

1. Formalizacija općenite i sustavne metodologije za odabir najučinkovitijeg algoritma
2. Eksperimentalno vrednovanje metodologije izgradnjom dva domenska modela
3. Formalizacija novog algoritma temeljenog na indeksima znakova predstavljena u sljedećem poglavlju
4. Eksperimentalno vrednovanje novog algoritma u sljedećem poglavlju

U prvoj fazi osmišljena je metodologija odabira najučinkovitijeg algoritma. Postavljenom metodologijom izgrađena su dva modela za dvije različite domene

pronalaženja uzoraka u tekstovima: domenu prirodnog jezika i domenu DNA tekstova. Izgrađeni modeli su vrednovani reprezentativnim skupovima podataka različitim od skupova podataka kojima su modeli izgrađeni kako bi se provjerilo jesu li modeli adekvatno rangirali algoritme prema njihovoj učinkovitosti za promatrane domene. Eksperimentalni rezultati vrednovanja na dvije domene predstavljeni su tablično i grafički te je potvrđena prva postavljena hipoteza. Također su u prvoj fazi izgradnje metodologije prikupljena dodatna saznanja o svojstvima algoritama i mogućim poboljšanjima istih kroz identifikaciju dodatnih informacija koje se mogu dobiti predobradom uzorka, koja do sada nisu uključena u postojeće algoritme, a mogu biti iskorištena u pretrazi na način da se preskaču nepotrebne usporedbe čime se ubrzava pretraga i smanjuju potrebni računalni resursi. Ta saznanja dobivena iz prve i druge faze sintezom su objedinjena u formi novog algoritma za egzaktnu usporedbu znakova temeljenog na indeksima koji je opisan u sljedećem poglavlju. Četvrta faza je faza eksperimentalnog vrednovanja novog algoritma. Provođenjem eksperimentalnog vrednovanja novog algoritma, novi algoritam uspoređen je s postojećim, najvažnijim algoritmima za egzaktnu usporedbu znakova. Dobiveni rezultati, koji su detaljno predstavljeni u sljedećem poglavlju ovoga rada, potvrđuju drugu postavljenu hipotezu. Također su prikupljena i dodatna saznanja koja se u budućem istraživačkom radu mogu iskoristiti za unaprjeđenje predstavljenog novog algoritma.

Izgradnja metodologije sastoji se od šest koraka prikazanih na slici 3.1. Prvi korak predložene metodologije je odabir reprezentativnih tekstova za izgradnju modela domene. U drugom koraku odabiru se algoritmi koje se žele rangirati prema učinkovitosti. Nakon odabira reprezentativnih tekstova za izgradnju modela domene i algoritama, faza pretraživanja reprezentativnih uzoraka započinje u trećem koraku. U četvrtom koraku izračunava se entropija uzoraka. U petom koraku primjenjuje se diskretizacija entropije. Posljednji korak je klasifikacija algoritama u izgrađenom modelu i predstavljanje rangiranja algoritama prema predloženom pristupu.

Slika 3.1. Metodologija izgradnje modela temeljenog na entropiji za odabir algoritama za pretraživanje znakovnih nizova

Na temelju rezultata koji su dobiveni u eksperimentalnom dijelu istraživanja, objavljena su tri znanstvena rada vezana uz područje istraživanja [131]–[133].

U narednim poglavljima svaki korak postavljene metodologije detaljno je opisan.

3.1. Izbor reprezentativnih tekstova za izgradnju modela domene

Nakon formalnog opisa koncepta izgradnje metodologije u nastavku je opisan glavni dio razvijene metodologije, proces obrade podataka. Prije same primjene specifičnih tehnika analize podataka potrebno je navesti osnovne karakteristike podataka koji su nam na raspolaganju.

Prvi korak u izgradnji metodologije za odabir učinkovitog algoritma za egzaktnu pretragu znakovnih nizova, shematski prikazan na slici 3.1., podrazumijeva odabir reprezentativnih tekstova za izgradnju modela domene.

Prvi kriterij za odabir domenskih tekstova je javna dostupnost tekstova. Kako su istraživanja u području različitih domena tekstova raznolika i raširena, uobičajeno je postojanje javno dostupnih tekstova koji su unaprijed pripremljeni u obliku pogodnom za korištenje u različitim istraživanjima, poput dodatnih strukturalnih informacija (veličina teksta, specijalni karakteri, itd.) i optimizacije zapisa (npr. eliminiranje dupliciranih karaktera delimitera, itd.). Ukoliko se radi o domeni za koju još ne postoje javno dostupni, unaprijed pripremljeni tekstovi potrebno ih je pripremiti u sklopu razvoja modela. U provedenom eksperimentalnom istraživanju su odabrani reprezentativni tekstovi koji su javno dostupni i unaprijed pripremljeni. Dio njih spada u prethodno korištene tekstove SMART okvira za testiranje algoritama podudaranja znakovnih nizova. Tekstovi korišteni u postojećim okvirima za testiranje su različitih tipova, uključujući tekstove sekvenci genoma i prirodnog jezika [18]. Budući da SMART zbirka sadrži samo jednu sekvencu nukleotida, za tekstove DNA domene koji nisu dio postojeće zbirke odabrane su dodatne sekvence iz javno dostupne zbirke tekstova − GenBank [105] kako bi reprezentativni tekstovi izgradnje i vrednovanja modela za DNA domenu bili raznolikiji i time reprezentativniji. Dodatno odabrane sekvence imaju isti format kao i postojeća sekvenca pa nije bila potrebna nikakva prethodna priprema dodatnih reprezentativnih tekstova odnosno DNA sekvenci. Veličine dodatno odabranih DNA sekvenci izražene su u megabajtima i kreću se od 4 Mb do 25 Mb. Analogno mjeri baznih parova (*bp*) kreću se od 4 miliona do 25 miliona bp. Različiti raspon veličina DNA tekstova odabran je kako bi rezultati usporedbe bili pouzdaniji. Osim odabranih reprezentativnih tekstova DNA domene mogli su se odabrati bilo koji tekstovi domene iz navedenog javnog repozitorija ili bilo kojeg drugog repozitorija.

Za reprezentativne tekstove DNA domene odabrani su javno dostupni tekstovi četiri različite vrste genoma živućih organizama:

- Riba Kanglang iz porodice šarana (RJVU01051648.1 Anabarilius grahami AG-KIZ scaffold371_cov124, 14.747.523 bp, 14,3 Mb veličina datoteke) [80]
- Golema želva (NW_006571126.1 Chelonia mydas, CheMyd_1.0 scaffold1, 7.392.783 bp, 7,1 Mb) [81]
- Escherichia coli (NZ_LN874954.1, 49.02.341 bp, 4,8 Mb) [82]

- Rezus makaki iz porodice primata (ML143108.1 Macaca mulatta AG07107 kromosom 19 genoma, ScNM3vo_33 × 44 M, 24.310.526 bp, 24,3 Mb) [83]

Za domenu prirodnog jezika odabrani su javno dostupni reprezentativni tekstovi na engleskom jeziku u *Canterbury Corpusu* [134]. *Canterbury Corpus* sadrži zbirku tekstova prirodnog jezika različitih veličina koji se koriste za usporedbu metoda kompresije. Provedena istraživanja su vršena na alfabetskim jezicima. Budući da javno dostupna zbirka sadrži tekst engleskog jezika, u provedenom istraživanju je korišten biblijski podskup kao tekst za pretraživanje. Biblijski podskup iz postojećeg korpusa je reprezentativniji za prirodni engleski tekst od ostalih engleskih tekstova i javno je objavljen [11], [17], [121]. Osim odabranih reprezentativnih tekstova domene prirodnog jezika mogli su se odabrati bilo koji tekstovi domene iz navedene javne zbirke ili bilo kojeg drugog dostupnog repozitorija.

Reprezentativni tekst domene prirodnog jezika je:

- Biblija (Kralj James verzija, veličina 4,04 Mb) [84]

Reprezentativni uzorci sastoje se od dva skupa uzoraka. Prvi skup su namjerni uzorci, a drugi skup su uzorci stvoreni od nasumično generiranih znakova abecede. Odabir reprezentativnih uzoraka opisan je u poglavlju 2.3. Veličina skupa reprezentativnih uzorka treba biti dostatna kako bi zaključci zasnovani na odabranim uzorcima bili sigurniji i pouzdaniji.

Ciljana pouzdanost kojom se pomoću odabranih uzoraka iz populacije mogu izvesti zaključci o cjelokupnoj populaciji je 95%, sa stvarnom vrijednosti unutar ±1% od ispitane vrijednosti. Metodologija nije ograničena manjom razinom pouzdanosti, ali kako želimo što pouzdanije modele ovo je preporučena razina pouzdanosti. S ciljanom razinom pouzdanosti može se zaključiti da je model objektivan jer je izrađen s odgovarajućom veličinom broja uzoraka.

Da bi se osigurala ciljana razina pouzdanosti primjenom jednadžbe (2.3) potrebno je ukupno 4.269 (ili više) uzoraka za DNA domenu (N=7.682) i 1.685 (ili više) uzoraka za domenu prirodnog jezika (N=2.043) [108]–[110]. Korištenjem više uzorka od potrebnih za ciljanu razinu pouzdanosti postiže se bolja objektivnost modela.

3.2. Odabir algoritama

U drugom koraku izgradnje metodologije (Slika 3.1.) odabiru se algoritmi koji će se rangirati prema učinkovitosti.

U provedenim eksperimentima izgradnje dva domenska modela odabrano je sedam najčešće korištenih algoritama za egzaktno podudaranje znakovnih nizova koji koriste pristup usporedbe znakova s *online* algoritamskim rješenjem pretrage. Odabrani algoritmi su: *Brute force* (BF, *näive*), Boyer-Moore (BM), Knuth-Morris-Pratt (KMP), Apostolico-Crochemore (AC), Quick Search (QS), Morris-Pratt (MP) i Horspool (HOR) [54], [56], [59], [75], [76], [89], [135], [136].

Sedam algoritama za pretraživanje nizova odabrano je kao polazna točka za izgradnju modela. Učinkovitost bilo kojeg drugog algoritma za egzaktno podudaranje znakovnih nizova može se rangirati osmišljenom metodologijom.

3.3. Rezultati pretraživanja uzoraka izraženi brojem usporedbi znakova

Nakon odabira reprezentativnih tekstova i uzoraka, te ciljanih algoritama, u trećem koraku proveden je proces pretraživanja reprezentativnih uzoraka odnosno skupljanja rezultata pretrage.

Proces pretraživanja znači da se svi reprezentativni uzorci pretražuju u odabranim reprezentativnim tekstovima pomoću algoritama odabranih u drugom koraku. Metrika korištena u procesu pretraživanja je broj usporedbi znakova CC (*character comparison*). Ova metrika u apsolutnom iznosu izražava ukupan broj uspoređenih znakova uzorka sa znakovima teksta neovisno o implementacijskoj platformi algoritama. Prikupljeni podaci svrstavaju se u grupu primarnih podataka jer se prvi put koriste u ovome istraživanju.

Na slici 3.2. a) i b) su prikazani rezultati pretrage u formatu tekstualne datoteke, odvojeni s točka zarez, u kojoj svaki redak predstavlja određeni rezultat pretrage za DNA domenu odnosno za domenu prirodnog jezika. Prikazan je samo dio rezultata iz skupa svih rezultata pretraživanja svih reprezentativnih uzoraka u svim odabranim reprezentativnim tekstovima domena modela.

```
Algo;m;Comp;Pattern
...
BF;8;56.418.198;AAGTGCTA
BM;8;18.109.833;AAGTGCTA
KMP;8;41.355.651;AAGTGCTA
AC;8;34.238.355;AAGTGCTA
QS;8;24.950.853;AAGTGCTA
MP;8;48.466.462;AAGTGCTA
HOR;8;19.306.915;AAGTGCTA
...
```

a) jednog uzoraka DNA domene

```
Algo;m;Comp;Pattern;PattEnt;PattEntRound
...
BF;8;56.418.198;AAGTGCTA;1.90563906222957;1.91
BM;8;18.109.833;AAGTGCTA;1.90563906222957;1.91
KMP;8;41.355.651;AAGTGCTA;1.90563906222957;1.91
AC;8;34.238.355;AAGTGCTA;1.90563906222957;1.91
QS;8;24.950.853;AAGTGCTA;1.90563906222957;1.91
MP;8;48.466.462;AAGTGCTA;1.90563906222957;1.91
HOR;8;19.306.915;AAGTGCTA;1.90563906222957;1.91
...
```

b) jednog uzoraka domene prirodnog jezika

Slika 3.2. Primjer rezultata pretrage

Oznake korištene u datoteci rezultata pretrage reprezentativnih uzoraka za svaki odabrani algoritam na slici 3.2. a) i b) su:

Algo – algoritam kojim je pretraga izvršena,
m – veličina uzorka,
Comp – broj usporedbi znakova u procesu pretrage i
Pattern – traženi uzorak.

Nakon napravljene pretrage reprezentativnih uzoraka u odabranim tekstovima u idućem koraku izgradnje modela potrebno je izračunati entropiju uzoraka korištenih u pretrazi. Vrijednost entropije uzorka povezuje se s pojedinačnim uzorkom u datoteci s rezultatima čiji su isječci prikazani na slici 3.2. a) i b). Metodologija izgradnje modela za odabir učinkovitog algoritma za promatranu domenu ni na koji način ne ograničava moguće različite načine implementacije metodologije. U eksperimentima koji su

provedeni u sklopu doktorskog rada koraci su realizirani na opisane načine. Stoga su vrlo jednostavno lako ponovljivi, a dobiveni rezultati u takvom obliku su pregledni i lako preoblikovani u prikaze korištene u radu za prezentiranje rezultata.

3.4. Izračun entropije uzorka

U narednim koracima izgradnje metodologije napravljena je obrada i analiza prikupljenih podataka iz prethodnog koraka pretrage reprezentativnih uzoraka u svrhu pripreme i obrade podataka za daljnju analizu kako bi se provelo zaključivanje o učinkovitosti analiziranih algoritama. Za analizu i obradu podataka korištena je deskriptivna statistika i entropija.

Specifičnost četvrtog koraka je izračun entropije uzoraka. Izračun entropije napravljen je na cijelom skupu uzoraka korištenih u izgradnji modela. Za izračun entropije uzoraka korištena je Shannonova entropija izražena jednadžbama (2.5) i (2.6). Primjer izračuna entropije datog uzroka prikazan je opisanom primjeru u poglavlju 2.4.

U tablici 3.1. prikazane su vrijednosti entropije za dio nasumično odabranih uzoraka iz skupa uzoraka korištenih u prethodnom koraku metodologije, pretraživanju uzoraka s rezultatima izraženim brojem usporedbi znakova.

Tablica 3.1. Vrijednosti entropije za odabrane uzorke

Uzorak	Entropija	Entropija zaokruženo
CA	1,00	1,00
TAAA	0,811278124459133	0,81
AATAAATC	1,2987949406954	1,30
AATATCACTTTGATGC	1,88285606369205	1,88
AGTGAGTGTTTATGCAGTTTGGCCTAACGCAT	1,94236882041153	1,94
Joseph remembere	3,0243974703477	3.02
e: The LORD lift	3,5778195311147	3.58

Nakon izračuna entropije za sve uzorke korištene u pretrazi, inicijalna datoteka s rezultatima nadopunjuje se dodatnim podacima u provedenom eksperimentalnom izgrađivanju modela razvijenom metodologijom. Izgled datoteka s rezultatima nakon izračuna entropije uzoraka za DNA domenu i za domenu prirodnog jezika prikazan je na

slici 3.3. a) i b). Metodologijom se zahtjeva izračun entropije reprezentativnih uzoraka zbog potrebe analize dobivenih rezultata u sljedećim koracima izgradnje metodologije.

```
Algo;m;Occ;Comp;Pattern;PattEnt;PattEntRound
...
BF;16;1;4.417.460;e: The LORD lift;3.57781953111478;3.58
BM;16;1;385.628;e: The LORD lift;3.57781953111478;3.58
KMP;16;1;4.402.044;e: The LORD lift;3.57781953111478;3.58
AC;16;1;4.016.960;e: The LORD lift;3.57781953111478;3.58
QS;16;1;409.425;e: The LORD lift;3.57781953111478;3.58
MP;16;1;4.402.067;e: The LORD lift;3.57781953111478;3.58
HOR;16;1;395.739;e: The LORD lift;3.57781953111478;3.58
...
```

b) domene prirodnog jezika

```
Algo;m;Comp;Pattern;PattEnt;PattEntRound
...
BF;8;56.418.198;AAGTGCTA;1.90563906222957;1.91
BM;8;18.109.833;AAGTGCTA;1.90563906222957;1.91
KMP;8;41.355.651;AAGTGCTA;1.90563906222957;1.91
AC;8;34.238.355;AAGTGCTA;1.90563906222957;1.91
QS;8;24.950.853;AAGTGCTA;1.90563906222957;1.91
MP;8;48.466.462;AAGTGCTA;1.90563906222957;1.91
HOR;8;19.306.915;AAGTGCTA;1.90563906222957;1.91
...
```

a) DNA domene

Slika 3.3. Izračunate entropije uzoraka

Rezultati izračuna entropije (Slika 3.3. a) i b)) za svaki pretraženi uzorak spremljeni su u inicijalnu datoteku s rezultatima i označeni su oznakama *PattEnt* – vrijednost entropije, *PattEntRound* – zaokružena vrijednost na dvije decimale varijable *PattEnt*.

Nakon izračuna entropije za uzorke korištene u izgradnji modela prelazi se na idući korak. U idućem koraku svaka vrijednost entropije uzorka klasificira se u određene grupe ili razrede. Razredi nisu poznati unaprijed, a podjela u razrede napravljena je sukladno distribuciji frekvencija entropije uzoraka.

3.5. Podjela vrijednosti entropije u razrede

U petom koraku izgradnje metodologije vrijednosti entropije uzoraka smještaju se u odgovarajuće razrede provedbom procesa diskretizacije. Diskretizacija je proces transformacije numeričkih podataka u nominalne tako što se numeričke vrijednosti sukladno distribuciji frekvencija (engl. *frequency distribution*) smještaju u odgovarajuće razrede kojih ima konačan broj, a što za cilj ima pretvoriti veliki skup broja vrijednosti podataka u manje skupove tako da procjena i upravljanje podacima postane jednostavno [111], [137].

Najčešći pristupi diskretizacije su nenadgledani i nadgledani pristup. U nenadgledanom pristupu razredi mogu biti jednake širine opsega (engl. *equal-width binning*) i jednakog pojavljivanja u opsezima (eng. *equal-frequency binning*). Nadgledani pristup uzima u obzir razrede [111], [137].

Maksimalan broj razreda, kada je nadgledani pristup diskretizacije u primjeni, u fazi diskretizacije određen je jednadžbom (3.1) (n je ukupan broj opažanja u podacima) [111], [137]:

$$broj\ razreda = C = 2 \times \sqrt[3]{n} \tag{3.1}$$

Također, raspon intervala podataka treba izračunati pronalaženjem minimalnih i maksimalnih vrijednosti. Raspon će se koristiti za određivanje intervala razreda ili širine razreda. Koristi se sljedeća jednadžba (3.2) [111], [137]:

$$h = \frac{\max(vrijednosti) - \min\ (vrijednosti)}{C} \tag{3.2}$$

Izračunate entropije podijeljene u razrede prikazuju broj opažanja ili rezultata u uzorku određenog intervala pri čemu razredi ne moraju biti predstavljeni istim brojem uzoraka.

3.5.1. Razredi entropije za DNA domenu

Za DNA domenu modela ukupan broj opažanja u podacima je *n=91.* Podaci broja opažanja (*n*) su prebrojane različite vrijednosti izračunatih entropija zaokruženih na dvije decimale. Detaljan prikaz ukupnog broja vrijednosti opažanja prikazan je u tablici 3.2.

Tablica 3.2. Ukupan broj opažanja vrijednosti entropije za DNA domenu, n = 91

Brojač	Entropija	Brojač	Entropija	Brojač	Entropija
1	0	31	1,4	61	1,7
2	0,34	32	1,41	62	1,71
3	0,53	33	1,42	63	1,72
4	0,54	34	1,43	64	1,73
5	0,7	35	1,44	65	1,74
6	0,81	36	1,45	66	1,75
7	0,9	37	1,46	67	1,76
8	0,95	38	1,47	68	1,77
9	0,99	39	1,48	69	1,78
10	1	40	1,49	70	1,79
11	1,01	41	1,5	71	1,8
12	1,05	42	1,51	72	1,81
13	1,06	43	1,52	73	1,82
14	1,12	44	1,53	74	1,83
15	1,16	45	1,54	75	1,84
16	1,19	46	1,55	76	1,85
17	1,2	47	1,56	77	1,86
18	1,24	48	1,57	78	1,87
19	1,25	49	1,58	79	1,88
20	1,26	50	1,59	80	1,89
21	1,27	51	1,6	81	1,9
22	1,3	52	1,61	82	1,91
23	1,32	53	1,62	83	1,92
24	1,33	54	1,63	84	1,93
25	1,34	55	1,64	85	1,94
26	1,35	56	1,65	86	1,95
27	1,36	57	1,66	87	1,96
28	1,37	58	1,67	88	1,97
29	1,38	59	1,68	89	1,98
30	1,39	60	1,69	90	1,99
				91	2

Broj razreda za DNA domenu, primjenom jednadžbe (3.1) je 9, a širina razreda nakon primjene jednadžbe (3.2) je 0,22222.

Nakon definiranja broja razreda i širine razreda napravljena je podjela vrijednosti entropije u razrede. U tablici 3.3. prikazan je dio ukupne klasifikacije entropije određenih uzoraka u razrede nakon diskretizacije za DNA domenu. U tablici kolona „*PatEntClass*“ predstavlja razred entropije nakon procesa diskretizacije.

Tablica 3.3. Diskretizacija podataka za domenu DNA

Uzorak	PattEnt	PattEntRound	PattEntClass
AAAAAAAA	0,0000000000000	0,00	< 0,22222
TTTTTTTTCTTTTTTT	0,3372900666170	0,34	0,22222–0,44444
AACAAAAA	0,5435644431996	0,54	0,44444–0,66667
AAAAAAACAAACAACA	0,6962122601251	0,70	0,66667–0,88889
TGGTAAAAAAAAAAAA	1,0612781244591	1,06	0,88889–1,11111
AAAAAGCG	1,2987949406954	1,30	1,11111–1,33333
CAAG	1,5000000000000	1,50	1,33333–1,55556
CCTACTAAACACCGTA	1,7640976555739	1,76	1,55556–1,77778
GCATACCTTTCGCAGC	1,9362781244591	1,94	≥ 1,77778

U tablici 3.4. prikazan je ukupan broj uzoraka za svaki razred entropije DNA domene nakon procesa diskretizacije.

Tablica 3.4. Razredi entropije nakon diskretizacije za domenu DNA

Broj razreda	Razred entropije	Broj uzoraka
1	< 0,22222	9
2	0,22222–0,44444	1
3	0,44444–0,66667	23
4	0,66667–0,88889	72
5	0,88889–1,11111	195
6	1,11111–1,33333	278
7	1,33333–1,55556	961
8	1,55556–1,77778	1.451
9	≥ 1,77778	4.692
	Ukupno	7.682

3.5.2. Razredi entropije za domenu prirodnog jezika

Za domenu prirodnog jezika ukupan broj opažanja u podacima je n=105. Detaljan prikaz ukupnog broja vrijednosti opažanja prikazan je u tablici 3.5.

Tablica 3.5. Ukupan broj opažanja vrijednosti entropije za domenu prirodnog jezika, n = 105

Brojač	Entropija	Brojač	Entropija	Brojač	Entropija	Brojač	Entropija
1	0	31	3,23	61	3,57	91	3,88
2	1	32	3,25	62	3,58	92	3,89
3	1,5	33	3,26	63	3,59	93	3,9
4	2	34	3,27	64	3,6	94	3,91
5	2,16	35	3,28	65	3,61	95	3,93
6	2,25	36	3,3	66	3,62	96	3,94
7	2,41	37	3,32	67	3,63	97	3,95
8	2,5	38	3,33	68	3,64	98	3,97
9	2,62	39	3,34	69	3,65	99	3,99
10	2,66	40	3,35	70	3,66	100	4,01
11	2,75	41	3,36	71	3,67	101	4,02
12	2,77	42	3,37	72	3,68	102	4,03
13	2,78	43	3,38	73	3,69	103	4,07
14	2,83	44	3,39	74	3,7	104	4,09
15	2,85	45	3,4	75	3,71	105	4,14
16	2,86	46	3,41	76	3,72		
17	2,88	47	3,42	77	3,73		
18	2,9	48	3,43	78	3,74		
19	2,91	49	3,44	79	3,75		
20	2,95	50	3,45	80	3,76		
21	2,98	51	3,46	81	3,77		
22	3	52	3,48	82	3,78		
23	3,02	53	3,49	83	3,79		
24	3,03	54	3,5	84	3,8		
25	3,08	55	3,51	85	3,81		
26	3,11	56	3,52	86	3,82		
27	3,12	57	3,53	87	3,83		
28	3,15	58	3,54	88	3,84		
29	3,16	59	3,55	89	3,86		
30	3,2	60	3,56	90	3,87		

Broj razreda za domenu prirodnog jezika primjenom jednadžbe (3.1) je 9, širina razreda nakon primjene jednadžbe (3.2) je 0,46004.

U tablici 3.6. prikazan je dio ukupne klasifikacije entropije određenih uzoraka u razrede nakon diskretizacije za domenu prirodnog jezika.

Tablica 3.6. Diskretizacija podataka za područje prirodnog jezika.

Uzorak	PattEnt	PattEntRound	PattEntClass
Mm	0,0000000000000	0,00	< 0,46004
We	1,0000000000000	1,00	0,92007–1,38011
Full	1,5000000000000	1,50	1,38011–1,84014
off from	2,1556390622295	2,16	1,84014–2,30018
ine enemies, eve	2,6556390622295	2,66	2,30018–2,76021
Joseph remembere	3,0243974703477	3,02	2,76021–3,22025
e: The LORD lift	3,5778195311147	3,58	3,22025–3,68028
they have, and deliver our lives	3,7508072359050	3,75	≥ 3,68028

U tablici 3.7. prikazan je ukupan broj uzoraka za svaki razred entropije domene prirodnog jezika nakon diskretizacije.

Tablica 3.7. Razredi entropije nakon diskretizacije za područje prirodnog jezika

Broj razreda	Razred entropije	Broj uzoraka
1	< 0,46004	3
2	0,46004–0,92007	0
3	0,92007–1,38011	151
4	1,38011–1,84014	53
5	1,84014–2,30018	383
6	2,30018–2,76021	393
7	2,76021–3,22025	283
8	3,22025–3,68028	556
9	≥ 3,68028	221
	Ukupno	2.043

Razredi entropije koje sadrže mali broj uzoraka najmanje utječu na model zbog toga što su takvi uzorci rijetki i javljaju se u manje od 0,5% slučajeva. Stoga učinkovitost

algoritama pri pronalaženju takvih, rijetkih uzoraka nije jednako važna kao učinkovitost algoritama pri pronalaženju čestih uzoraka. Primjeri takvih uzoraka za DNA domenu su TTTTTTTTTTTTTTTTT, AAAGAAA, a za domenu prirodnog jezika LL.

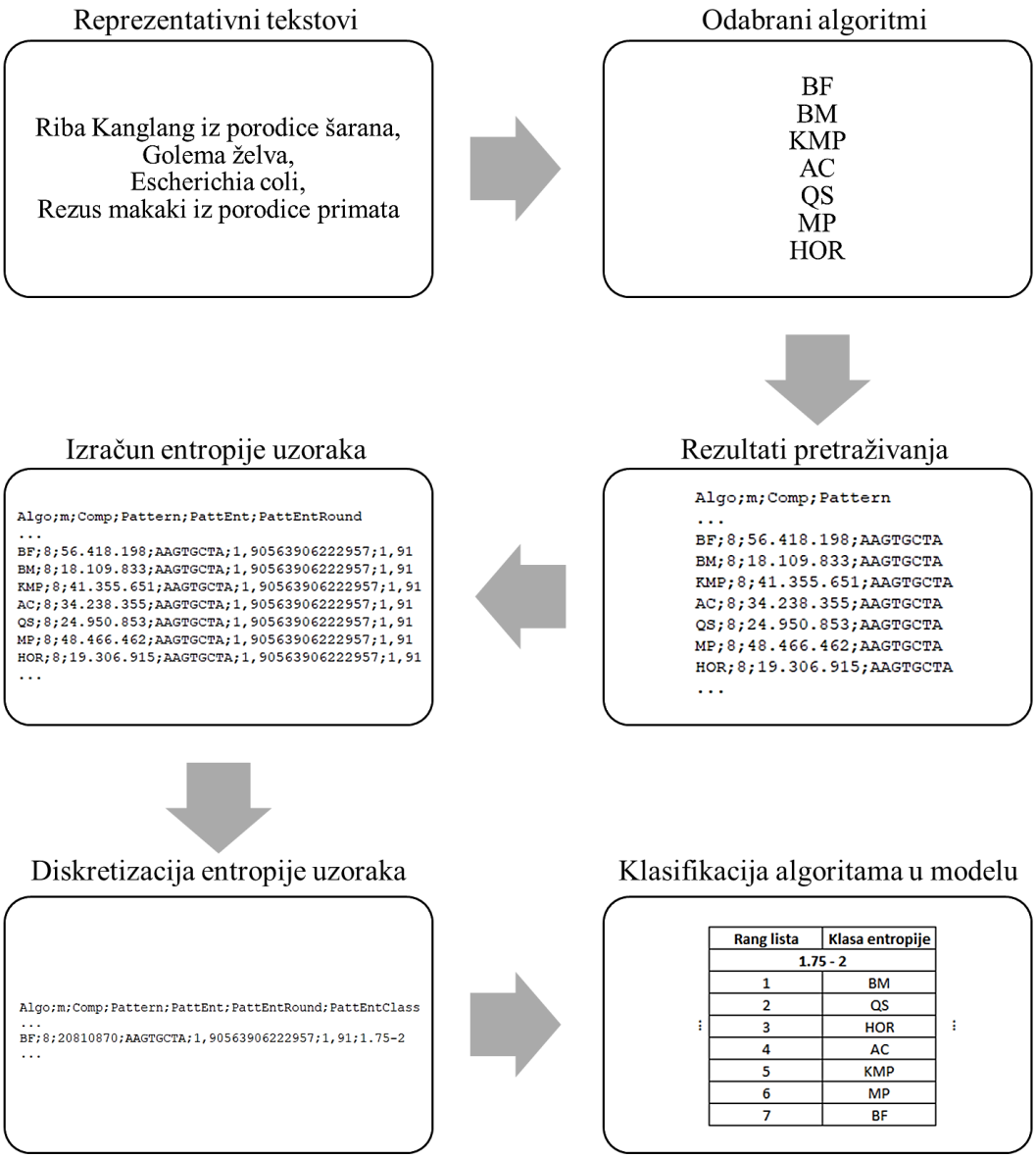
Vrijednost entropije ovisi i o načinu odabira uzroka. Entropija je veća za uzorke koji su stvoreni od nasumično generiranih znakova abecede promatrane domene od entropije namjernih uzoraka koji su podnizovi reprezentativnog teksta, osim ako reprezentativni tekst nije nasumično generiran od znakova abecede. Domene koje imaju veći broj znakova abecede imaju veću vjerojatnost da uzorci imaju manji broj ponavljajućih znakova što dovodi do povećavanja vrijednost entropije.

3.6. Klasifikacija algoritama u izgrađenom modelu

Posljednji korak u izgradnji metodologije je kvantitativna i kvalitativna analiza obrađenih podataka. Ovom analizom provodi se klasifikacija algoritama u izgrađenom modelu čime su sirovi podaci pretvoreni u informacije o učinkovitosti algoritama. Izrađeni model primjenjiv je za bilo koji uzorak kojeg treba pronaći u bilo kojem dostupnom tekstu ili korpusu tekstova određene domene.

Rezultati broja potrebnih usporedbi znakova za podudaranje svakog uzorka u kombinaciji s određenim razredom entropije uzorka integrirani su u novi model za rangiranje algoritama. Algoritmi koji su učinkovitiji izvode manji broj usporedbi znakova pri pronalaženju uzorka u tekstu. Model predlaže odabir učinkovitog algoritma za podudaranje znakovnih nizova za određeni uzorak na temelju razreda entropije kojoj pripada promatrani uzorak. Pomoću modela moguće je utvrditi povezanost entropije uzorka i učinkovitosti algoritama metrikom broja usporedbi znakova [111], [137].

Primjena predložene metodologije u radu na DNA domeni je prikazana na slici 3.5. blok dijagramom.



Slika 3.5. Blok dijagram primjene nove metodologije na primjeru DNA domene

Tablica 3.8. prikazuje izgrađeni model rangiranja algoritama prema entropijama uzoraka po kvartilima za domenu DNA tekstova i uzorka. Gornji kvartil je 25% najboljih algoritama s najmanjim brojem usporedbi karaktera za pronalaženje uzoraka. Donji kvartil je 25% najlošijih algoritama s najvećim brojem usporedbi karaktera za pronalaženje uzoraka.

Tablica 3.8. Model rangiranja algoritama DNA domene

Algo/Razred	1	2	3	4	5	6	7	8	9
Gornji kvartil									
AC	20,59%	0,00%	20,00%	12,61%	5,74%	8,99%	8,51%	5,43%	2,53%
BF	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
BM	47,92%	100,00%	56,67%	51,82%	57,20%	61,11%	58,26%	61,95%	62,83%
HOR	38,24%	0,00%	46,67%	49,55%	49,88%	51,06%	49,92%	53,92%	54,60%
KMP	14,71%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
MP	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
QS	55,88%	100,00%	53,33%	51,35%	55,36%	53,97%	54,08%	53,68%	54,85%
Interkvartil									
AC	52,94%	100,00%	70,00%	60,36%	70,32%	67,46%	64,56%	68,14%	65,75%
BF	50,00%	0,00%	30,00%	47,75%	48,38%	44,18%	46,31%	45,94%	52,37%
BM	43,75%	0,00%	43,33%	48,18%	40,33%	38,89%	41,74%	38,05%	37,16%
HOR	50,00%	100,00%	53,33%	50,00%	47,13%	48,94%	49,92%	46,08%	45,13%
KMP	58,82%	100,00%	56,67%	47,75%	55,61%	60,05%	56,28%	59,35%	52,37%
MP	50,00%	0,00%	46,67%	47,75%	48,38%	44,18%	46,31%	46,13%	52,37%
QS	44,12%	0,00%	46,67%	48,65%	41,90%	46,03%	45,92%	46,32%	44,90%
Donji kvartil									
AC	26,47%	0,00%	10,00%	27,03%	23,94%	23,54%	26,93%	26,43%	31,72%
BF	50,00%	100,00%	70,00%	52,25%	51,62%	55,82%	53,69%	54,06%	47,63%
BM	8,33%	0,00%	0,00%	0,00%	2,47%	0,00%	0,00%	0,00%	0,01%
HOR	11,76%	0,00%	0,00%	0,45%	2,99%	0,00%	0,17%	0,00%	0,27%
KMP	26,47%	0,00%	43,33%	52,25%	44,39%	39,95%	43,72%	40,65%	47,63%
MP	50,00%	100,00%	53,33%	52,25%	51,62%	55,82%	53,69%	53,87%	47,63%
QS	0,00%	0,00%	0,00%	0,00%	2,74%	0,00%	0,00%	0,00%	0,25%

Razredi entropije korištene u tablici 3.8. definirane su u tablici 3.4.

Za svaki razred entropije uzoraka rezultati broja usporedbi prilikom pretraživanja uzoraka za određeni algoritam izraženi su u postotcima i grupirani u kvartile. Na ovaj način je predstavljeno koliko često je određeni algoritam među 25% najučinkovitijih algoritama za određeni tip uzorka identificiran entropijom uzorka. Za najučinkovitije algoritme koji su predloženi modelom smatraju se oni algoritmi koji će za dani uzorak napraviti najmanji broj usporedbi znakova za pronalaženje uzorka.

Prema podacima modela za DNA domenu, prikazanim u tablici 3.8., ako pretraživani uzorak pripada 1. razredu entropije (< 0,22222), 55,88% rezultata pretraživanja izraženih kao broj usporedbi znakova (CC) s QS algoritmom nalazi se u gornjem kvartilu pa u ovome slučaju izgrađeni model preporučuje QS algoritam kao najučinkovitiji algoritam razreda. Kao drugi najučinkovitiji algoritam 1. razreda entropije preporučuje se BM algoritam prema kojem je 47,92% rezultata u gornjem kvartilu.

Ako pretraživani uzorak pripada 2. razredu entropije (0,22222–0,44444), 100% rezultata pretraživanja izraženih kao broj usporedbi znakova (CC) s BM i QS algoritmom nalazi se u gornjem kvartilu pa u ovome slučaju izgrađeni model preporučuje BM ili QS algoritme kao najučinkovitije algoritme razreda.

Ako pretraživani uzorak pripada 3. razredu entropije (0,44444–0,66667), 56,67% rezultata pretraživanja izraženih kao broj usporedbi znakova (CC) s BM algoritmom nalazi se u gornjem kvartilu pa u ovome slučaju izgrađeni model preporučuje BM algoritam kao najučinkovitiji algoritam razreda. Kao drugi najučinkovitiji algoritam 3. razreda entropije preporučuje se QS algoritam prema kojem je 53,33% rezultata u gornjem kvartilu.

Ako pretraživani uzorak pripada 4. razredu entropije (0,66667–0,88889), 51,82% rezultata pretraživanja izraženih kao broj usporedbi znakova (CC) s BM algoritmom nalazi se u gornjem kvartilu pa u ovome slučaju izgrađeni model preporučuje BM algoritam kao najučinkovitiji algoritam razreda. Kao drugi najučinkovitiji algoritam 4. razreda entropije preporučuje se QS algoritam prema kojem je 51,35% rezultata u gornjem kvartilu.

Ako pretraživani uzorak pripada 5. razredu entropije (0,88889–1,11111), 57,20% rezultata pretraživanja izraženih kao broj usporedbi znakova (CC) s BM algoritmom nalazi se u gornjem kvartilu pa u ovome slučaju izgrađeni model preporučuje BM algoritam kao najučinkovitiji algoritam razreda. Kao drugi najučinkovitiji algoritam 5. razreda entropije preporučuje se QS algoritam prema kojem je 55,36% rezultata u gornjem kvartilu.

Ako pretraživani uzorak pripada 6. razredu entropije (1,11111–1,33333), 61,11% rezultata pretraživanja izraženih kao broj usporedbi znakova (CC) s BM algoritmom nalazi se u gornjem kvartilu pa u ovome slučaju izgrađeni model preporučuje BM

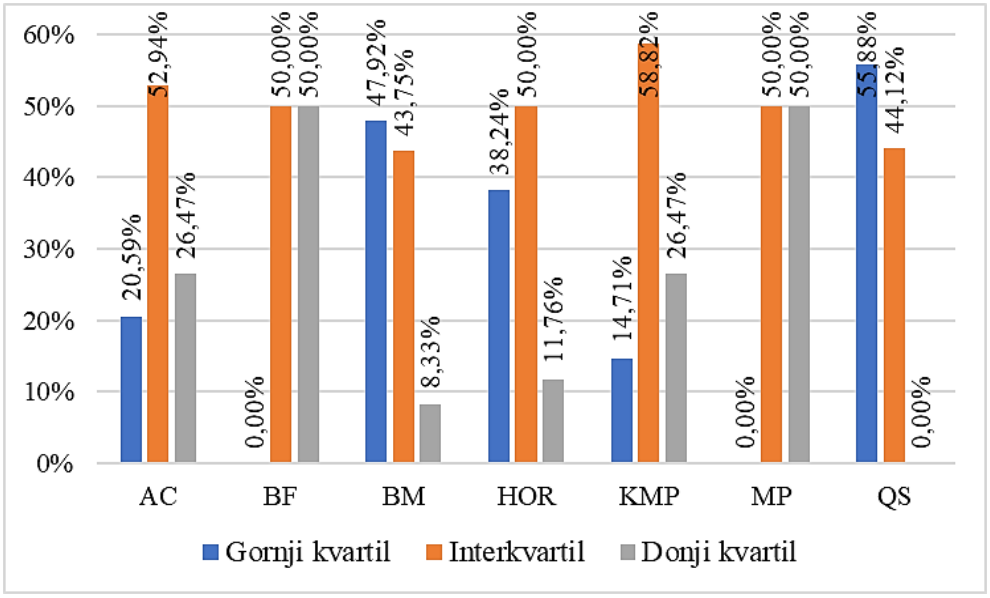
algoritam kao najučinkovitiji algoritam razreda. Kao drugi najučinkovitiji algoritam 6. razreda entropije preporučuje se QS algoritam prema kojem je 53,97% rezultata u gornjem kvartilu.

Ako pretraživani uzorak pripada 7. razredu entropije (1,33333–1,55556), 58,26% rezultata pretraživanja izraženih kao broj usporedbi znakova (CC) s BM algoritmom nalazi se u gornjem kvartilu pa u ovome slučaju izgrađeni model preporučuje BM algoritam kao najučinkovitiji algoritam razreda. Kao drugi najučinkovitiji algoritam 7. razreda entropije preporučuje se QS algoritam prema kojem je 54,08% rezultata u gornjem kvartilu.

Ako pretraživani uzorak pripada 8. razredu entropije (1,55556–1,77778), 61,95% rezultata pretraživanja izraženih kao broj usporedbi znakova (CC) s BM algoritmom nalazi se u gornjem kvartilu pa u ovome slučaju izgrađeni model preporučuje BM algoritam kao najučinkovitiji algoritam razreda. Kao drugi najučinkovitiji algoritam 8. razreda entropije preporučuje se HOR algoritam prema kojem je 53,92% rezultata u gornjem kvartilu.

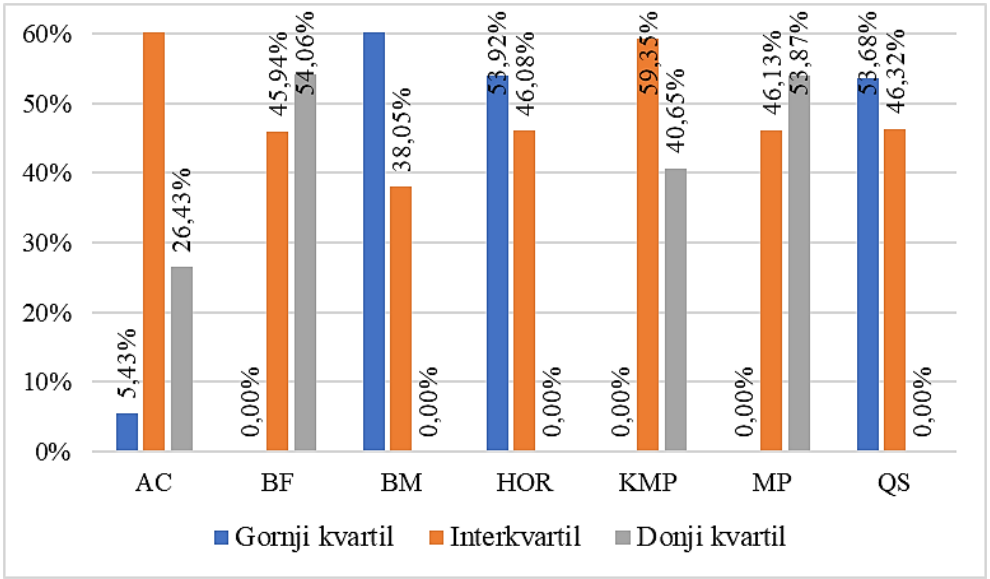
Ako pretraživani uzorak pripada 9. razredu entropije (≥ 1,77778), 62,83% rezultata pretraživanja izraženih kao broj usporedbi znakova (CC) s BM algoritmom nalazi se u gornjem kvartilu pa u ovome slučaju izgrađeni model preporučuje BM algoritam kao najučinkovitiji algoritam razreda. Kao drugi najučinkovitiji algoritam 9. razreda entropije preporučuje se QS algoritam prema kojem je 54,85% rezultata u gornjem kvartilu.

Prethodno tablično prezentirani rezultati modela DNA domene mogu se i preglednije prikazati grafički kao na slici 3.6. koja prikazuje rezultate modela za uzorke 1. razreda entropije. Prezentacija rezultata modela nije ograničena predloženom metodologijom tako da se prilikom izgradnje modela može odabrati korisnicima najprihvatljiviji način prezentacije rezultata modela.



Slika 3.6. Grafički prikaz rezultata modela za uzorke 1. razreda entropije DNA domene

Grafički prikaz rezultata modela za 8. razred entropije DNA domene prikazan je na slici 3.7.



Slika 3.7. Grafički prikaz rezultata modela za uzorke 8. razreda entropije DNA domene

U tablici 3.9. prikazana je skraćena lista najučinkovitijih algoritama ocijenjenih modelom u kontekstu lakše interpretacije uporabljivosti izrađenog modela za DNA

domenu (Tablica 3.8.) koja za cilj ima pojednostaviti odabir najučinkovitijih algoritama za određeni uzorak.

Tablica 3.9. Lista najučinkovitijih algoritama procijenjenih modelom za DNA domenu tekstova i uzoraka

Razred	Gornji kvartil Najučinkovitiji algoritmi	Interkvartil	Donji kvartil Najneučinkovitiji algoritmi
1	QS (55,88%) BM (47,92%)	KMP (58,82%) AC (52,94%)	BF (50,00%) KMP (26,47%)
2	BM (100,00%) QS (100,00%)	HOR (100,00%) KMP (100,00%)	BF (100,00%) MP (100,00%)
3	BM (56,67%) QS (53,33%)	AC (70,00%) KMP (56,67%)	BF (70,00%) MP (53,33%)
4	BM (51,82%) QS (51,35%)	AC (60,36%) HOR (50%)	BF (52,25%) AC (27,03%)
5	BM (57,20%) QS (55,36%)	AC (70,32%) KMP (55,61%)	BF (51,62%) KMP (44,39%)
6	BM (61,11%) QS (53,97%)	AC (67,46%) KMP (60,05%)	BF (55,82%) KMP (39,95%)
7	BM (58,26%) QS (54,08%)	AC (64,56%) KMP (56,28%)	BF (53,69%) KMP (43,72%)
8	BM (61,95%) HOR (53,92%)	AC (68,14%) KMP (59,35%)	BF (54,06%) MP (53,87%)
9	BM (62,83%) QS (54,85%)	AC (65,75%) KMP (52,37%)	BF (47,63%) AC (31,72%)

Tablica 3.10. prikazuje izgrađeni model rangiranja algoritama prema entropijama uzoraka po kvartilima za tekstove i uzorke domene prirodnog jezika.

Tablica 3.10. Model rangiranja algoritama za tekstove i uzorke prirodnog jezika

Algo/Razred	1	2	3	4	5	6	7	8	9
Gornji kvartil									
AC	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
BF	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
BM	66,67%	0,00%	40,56%	60,38%	57,18%	64,05%	65,02%	58,81%	66,06%
HOR	33,33%	0,00%	35,06%	30,19%	38,72%	41,01%	51,24%	56,47%	52,04%
KMP	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
MP	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
QS	100,00%	0,00%	100,00%	84,91%	79,23%	70,13%	59,01%	59,71%	57,01%
Interkvartil									
AC	100,00%	0,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%
BF	0,00%	0,00%	36,67%	33,96%	33,08%	32,41%	31,45%	33,45%	32,13%
BM	33,33%	0,00%	59,44%	39,62%	42,82%	35,95%	34,98%	41,19%	33,94%
HOR	66,67%	0,00%	64,94%	69,81%	61,28%	58,99%	48,76%	43,53%	47,96%
KMP	100,00%	0,00%	44,44%	49,06%	45,90%	46,58%	47,70%	46,04%	47,06%
MP	33,33%	0,00%	44,44%	41,51%	45,90%	46,08%	45,94%	45,50%	46,15%
QS	0,00%	0,00%	0,00%	15,09%	20,77%	29,87%	40,99%	40,29%	42,99%
Donji kvartil									
AC	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
BF	100,00%	0,00%	63,33%	66,04%	66,92%	67,59%	68,55%	66,55%	67,87%
BM	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
HOR	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
KMP	0,00%	0,00%	55,56%	50,94%	54,10%	53,42%	52,30%	53,96%	52,94%
MP	66,67%	0,00%	55,56%	58,49%	54,10%	53,92%	54,06%	54,50%	53,85%
QS	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%

Razredi entropije korištene u tablici 3.10. definirani su u tablici 3.7.

Prema podacima modela za domenu prirodnog jezika, prikazanim u tablice 3.10., ako pretraživani uzorak pripada 1. razredu entropije (< 0,46004), 100,00% rezultata pretraživanja izraženih kao broj usporedbi znakova (CC) s QS algoritmom nalazi se u gornjem kvartilu pa u ovome slučaju izgrađeni model preporučuje QS algoritam kao

najučinkovitiji algoritam razreda. Kao drugi najučinkovitiji algoritam 1. razreda entropije preporučuje se BM algoritam prema kojem je 66,67% rezultata u gornjem kvartilu.

Ako pretraživani uzorak pripada 3. razredu entropije (0,92007–1,38011), 100,00% rezultata pretraživanja izraženih kao broj usporedbi znakova (CC) s QS algoritmom nalazi se u gornjem kvartilu pa u ovome slučaju izgrađeni model preporučuje QS algoritam kao najučinkovitiji algoritam razreda. Kao drugi najučinkovitiji algoritam 3. razreda entropije preporučuje se BM algoritam prema kojem je 40,56% rezultata u gornjem kvartilu.

Ako pretraživani uzorak pripada 4. razredu entropije (1,38011–1,84014), 84,91% rezultata pretraživanja izraženih kao broj usporedbi znakova (CC) s QS algoritmom nalazi se u gornjem kvartilu pa u ovome slučaju izgrađeni model preporučuje QS algoritam kao najučinkovitiji algoritam razreda. Kao drugi najučinkovitiji algoritam 4. razreda entropije preporučuje se BM algoritam prema kojem je 60,38% rezultata u gornjem kvartilu.

Ako pretraživani uzorak pripada 5. razredu entropije (1,84014–2,30018), 79,23% rezultata pretraživanja izraženih kao broj usporedbi znakova (CC) s QS algoritmom nalazi se u gornjem kvartilu pa u ovome slučaju izgrađeni model preporučuje QS algoritam kao najučinkovitiji algoritam razreda. Kao drugi najučinkovitiji algoritam 5. razreda entropije preporučuje se BM algoritam prema kojem je 57,18% rezultata u gornjem kvartilu.

Ako pretraživani uzorak pripada 6. razredu entropije (2,30018–2,76021), 70,13% rezultata pretraživanja izraženih kao broj usporedbi znakova (CC) s QS algoritmom nalazi se u gornjem kvartilu pa u ovome slučaju izgrađeni model preporučuje QS algoritam kao najučinkovitiji algoritam razreda. Kao drugi najučinkovitiji algoritam 6. razreda entropije preporučuje se BM algoritam prema kojem je 64,05% rezultata u gornjem kvartilu.

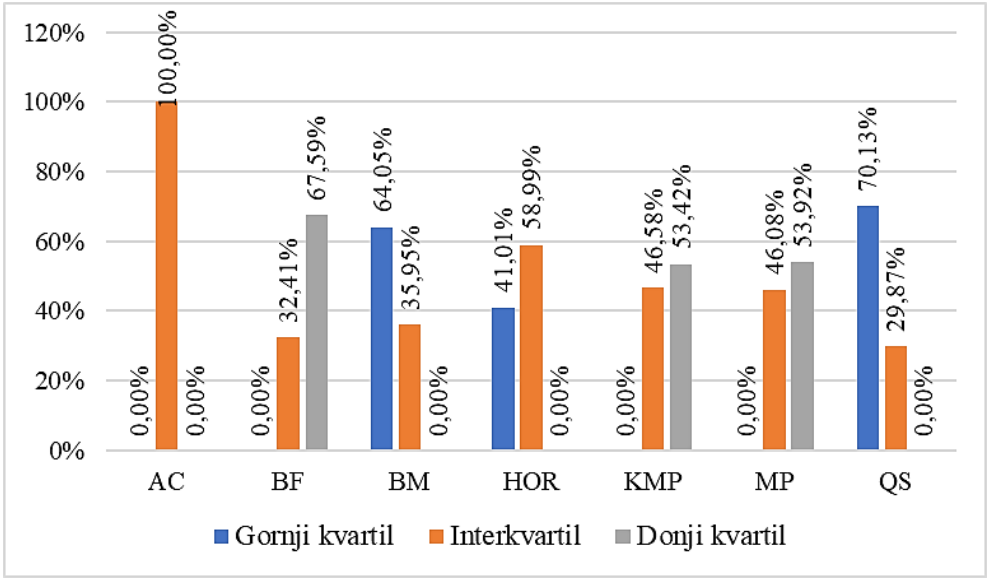
Ako pretraživani uzorak pripada 7. razredu entropije (2,76021–3,22025), 65,02% rezultata pretraživanja izraženih kao broj usporedbi znakova (CC) s BM algoritmom nalazi se u gornjem kvartilu pa u ovome slučaju izgrađeni model preporučuje BM algoritam kao najučinkovitiji algoritam razreda. Kao drugi najučinkovitiji algoritam 7.

razreda entropije preporučuje se QS algoritam prema kojem je 59,01% rezultata u gornjem kvartilu.

Ako pretraživani uzorak pripada 8. razredu entropije (3,22025–3,68028), 59,71% rezultata pretraživanja izraženih kao broj usporedbi znakova (CC) s QS algoritmom nalazi se u gornjem kvartilu pa u ovome slučaju izgrađeni model preporučuje QS algoritam kao najučinkovitiji algoritam razreda. Kao drugi najučinkovitiji algoritam 8. razreda entropije preporučuje se BM algoritam prema kojem je 58,81% rezultata u gornjem kvartilu.

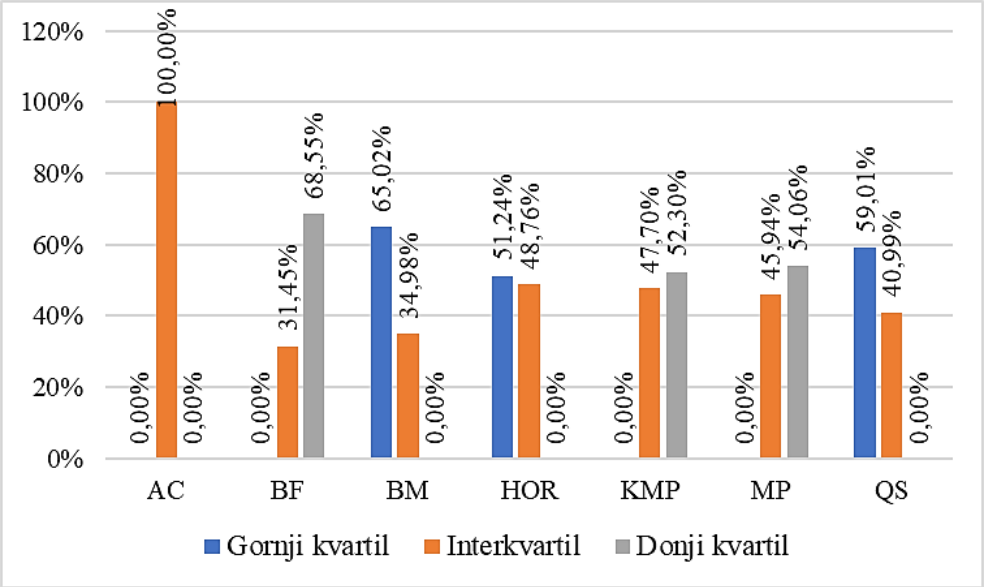
Ako pretraživani uzorak pripada 9. razredu entropije (≥ 3,68028), 66,06% rezultata pretraživanja izraženih kao broj usporedbi znakova (CC) s BM algoritmom nalazi se u gornjem kvartilu pa u ovome slučaju izgrađeni model preporučuje BM algoritam kao najučinkovitiji algoritam razreda. Kao drugi najučinkovitiji algoritam 9. razreda entropije preporučuje se QS algoritam prema kojem je 57,01% rezultata u gornjem kvartilu.

Grafički prikaz rezultata modela domene prirodnog jezika za uzorke 6. razreda entropije prikazan je na slici 3.8.



Slika 3.8. Grafički prikaz rezultata modela za uzorke 6. razreda entropije domene prirodnog jezika

Grafički prikaz rezultata modela za uzorke 7. razreda entropije domene prirodnog jezika prikazan je na slici 3.9.



Slika 3.9. Grafički prikaz rezultata modela za uzorke 7. razreda entropije domene prirodnog jezika

Grafički prikazi postavljenog modela za ostale razrede entropije DNA domene i domene prirodnog jezika bili bi prezentirani na isti način.

U tablici 3.11. prikazana je skraćena lista najučinkovitijih algoritama ocijenjenih modelom u kontekstu lakše interpretacije uporabljivosti izrađenog modela za domenu prirodnog jezika (Tablica 3.10.) koja za cilj ima pojednostaviti odabir najučinkovitijih algoritama za određeni uzorak.

Tablica 3.11. Lista najučinkovitijih algoritama procijenjenih modelom za domenu tekstova i uzoraka prirodnog jezika

Razred	Gornji kvartil Najučinkovitiji algoritmi	Interkvartil	Donji kvartil Najneučinkovitiji algoritmi
1	QS (100,0%) BM (66,67%)	KMP (100,00%) HOR (66,67%)	BF (100,00%) MP (66,67%)
3	QS (100,00%) BM (40,56%)	AC (100,00%) HOR (64,94%)	BF (63,33%) KMP (55,56%)
4	QS (84,91%) BM (60,38%)	HOR (69,81%) KMP (49,06%)	BF (66,04%) MP (58,49%)
5	QS (79,23%) BM (57,18%)	HOR (61,28%) KMP (45,90%)	BF (66,92%) KMP (54,10%)
6	QS (70,13%) BM (64,05%)	HOR (58,99%) KMP (46,58%)	BF (67,59%) MP (53,92%)
7	BM (65,02%) QS (59,01%)	HOR (48,76%) KMP (47,70%)	BF (68,55%) MP (54,06%)
8	QS (59,71%) BM (58,81%)	KMP (46,04%) MP (45,50%)	BF (66,55%) MP (54,50%)
9	BM (66,06%) QS (57,01%)	HOR (47,96%) KMP (47,06%)	BF (67,87%) MP (53,85%)

U tablici 3.8. i tablici 3.10. korištena je apsolutna mjera disperzije − interkvartilni raspon. Budući da vrijednost interkvartilnog raspona predstavlja disperziju podatka tj. udaljenost između gornjeg i donjeg kvartila, izrađeni model se fokusira samo na gornji kvartil u kojemu se nalazi lista najučinkovitijih algoritama. Utjecaj interkvartilnog raspona spada u domenu budućih istraživanja u svrhu poboljšanja predloženoga modela. [138], [139].

U slučaju da uzorak za kojeg je potrebno napraviti pretragu unutar domene najučinkovitijim algoritmom ne pripada niti jednom razredu entropije, relevantan je prvi najbliži razred entropije.

U sljedećem koraku model će se detaljno interpretirati u smislu pouzdanosti njegovih rezultata.

3.7. Vrednovanje metodologije

Nakon izgradnje modela napravljeno je vrednovanje modela za DNA domenu i domenu prirodnog jezika na način da su za provjeru točnosti izgrađenih modela odabrani uzorci dva razreda entropije postavljenog modela.

Glavni preduvjet za ispravno vrednovanje metodologije je odabir skupa uzorka nad kojima će se vršiti vrednovanje. Za potrebe vrednovanja modela odabrani su reprezentativni tekstovi koji se razlikuju od tekstova pomoću kojih je model izgrađen. Za potvrdu valjanosti postavljenog modela DNA domene odabran je reprezentativni tekst DNA sekvence − *Homo sapiens* izolirani HG00514 kromosom 9 HS_NIOH_CHR9_SCAFFOLD_1, 43.213.237 bp, 39 Mb [140]. Za domenu prirodnog jezika valjanost modela vrednovana je tekstom iz domene prirodnog jezika preuzetim iz *Canterbury Corpusa* [134].

Vrednovanjem modela za svaki algoritam korišten u analizi usporediti će se eksperimentalni rezultati broja usporedbi znakova dobivenih prilikom pretraživanja uzoraka u procesu vrednovanja izgrađenog modela za odabrane razrede s rezultatima broja usporedbi znakova dobivenih prilikom pretraživanja uzoraka izgrađenog modela za odabrane razrede

Za potrebe vrednovanja izrađenog modela DNA domene odabrani su uzorci koji pripadaju 7. razredu entropije i uzorci koji pripadaju 9. razredu entropije. Za vrednovanje izrađenog modela za domenu prirodnog jezika odabrani su uzorci koji pripadaju 6. razredu entropije te 9. razredu entropije. Odabrani razredi imaju najveći broj uzoraka od dostupnih razreda postavljenog modela. Osim odabranih razreda mogao se izabrati bilo koji razred za potrebe vrednovanja izrađenog modela.

Model rangiranja algoritama DNA domene i domene prirodnog jezika za odabrane razrede prikazan je u tablici 3.12.

Tablica 3.12. Model rangiranja algoritama DNA domene i domene prirodnog jezika za odabrane razrede vrednovanja za gornji kvartil

	DNA domena		*Domena prirodnog jezika*	
Algo/Razred	7	9	6	9
Gornji kvartil				
AC	8,51%	2,53%	0,00%	0,00%
BF	0,00%	0,00%	0,00%	0,00%
BM	58,26%	62,83%	64,05%	66,06%
HOR	49,92%	54,60%	41,01%	52,04%
KMP	0,00%	0,00%	0,00%	0,00%
MP	0,00%	0,00%	0,00%	0,00%
QS	54,08%	54,85%	70,13%	57,01%

Prema podacima modela koji se nalaze u tablici 3.12., a koji su dio modela prikazanog u tablici 3.8., kada pretraživani uzorak pripada 7. razredu entropije DNA domene, 58,26% rezultata pretraživanja s BM algoritmom nalazi se u gornjem kvartilu što znači da za uzorke iz DNA domene (DNA tekstova) koji se nalaze u 7. razredu entropije (1,33333–1,55556) izgrađeni model preporučuje BM algoritam kao najučinkovitiji algoritam razreda. Ako pretraživani uzorak pripada 9. razredu entropije (≥ 1,77778) DNA domene, 62,83% rezultata pretraživanja s BM algoritmom nalazi se u gornjem kvartilu čineći ga tako najučinkovitijim algoritmom razreda. Izgrađeni model za uzorke iz DNA domene koji se nalaze u 7. i 9. razredu entropije kao drugi najučinkovitiji algoritam preporučuje QS algoritam.

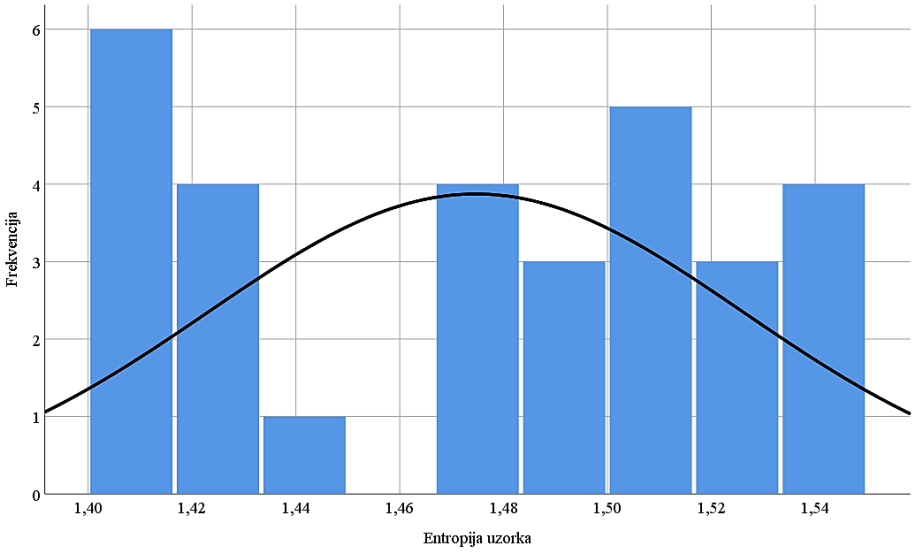
Prema podacima modela koji se nalaze tablici 3.12., kada pretraživani uzorak pripada 6. razredu entropije (2,30018–2,76021) domene prirodnog jezika, 70,13% rezultata pretraživanja s QS algoritmom nalazi se u gornjem kvartilu što znači da za izgrađeni model preporučuje QS algoritam kao najučinkovitiji algoritam razreda, a ako pretraživani uzorak pripada 9. razredu entropije (≥ 3,68028) domene prirodnog jezika, 66,06% rezultata pretraživanja s BM algoritmom nalazi se u gornjem kvartilu čineći ga najučinkovitijim algoritmom razreda.

3.7.1. Odabir skupa uzoraka za vrednovanje modela

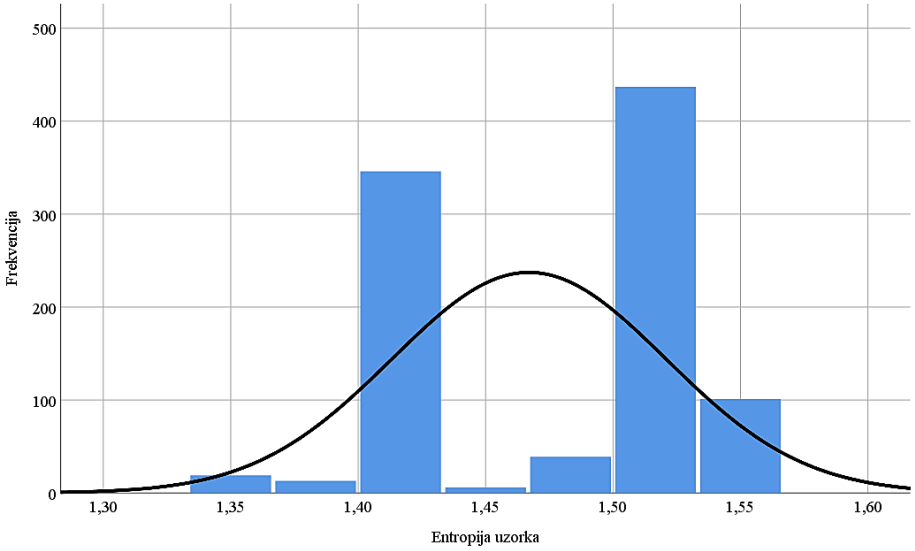
Iz odabranih reprezentativnih tekstova pomoću kojih će se model vrednovati odabrani su uzorci za vrednovanje pripadajućih razreda entropije postavljenog modela. Odabrani uzorci za vrednovanje modela nisu dio skupa uzoraka kojima je model izgrađen. U cilju postizanja različitosti reprezentativnih uzoraka na način da se razlikuju uzorci korišteni u izgradnji modela od uzoraka korištenih u vrednovanju modela utvrđeno je da se manje od 10% skupa uzoraka pomoću kojih je model vrednovan nalazi u skupu uzoraka kojima je model izrađen. Uzorci koji se ponavljaju u skupu uzoraka kojima je model vrednovan nalaze se u skupu uzoraka DNA domene. Radi se većinom o kratkim uzorcima duljine do 4 znaka.

Prije faze provjere valjanosti modela izvršena je provjera jesu li odabrani uzorci dovoljno reprezentativni za provjeru valjanosti modela. Provjera je obavljena središnjim graničnim teoremom koji dopušta rad s približno normalnom distribucijom. Prema središnjem graničnom teoremu ako se iz određene populacije uzme dovoljan broj uzoraka, srednja vrijednost odabranih uzorka i srednja vrijednost populacije bit će približno ista. Broj uzoraka u skupu prema središnjem graničnom teoremu smatra se dovoljnim ako je njegova veličina veća ili jednaka od 30. To znači, ako je broj uzoraka u skupu prema središnjem graničnom teoremu, srednja vrijednost skupa odabranih uzorka imat će funkciju raspodjele blizu normalne. Pomoću središnjeg graničnog teorema uspoređene su distribucije vrijednosti uzoraka kojima je model izgrađen i distribucije vrijednosti uzoraka kojima je model vrednovan [111].

Na slici 3.10. prikazana je distribucija vrijednosti entropija skupa uzoraka 7. razreda entropije (1,33333–1,55556) koji se koriste u fazama vrednovanja i izgradnje modela DNA domene. Skup uzoraka prikazanog razreda entropije DNA domene pomoću kojih se vrednuje model ima srednju vrijednost 1,473 i standardnu devijaciju 0,051, a skup uzoraka prikazanog razreda entropije DNA domene pomoću kojih je model izgrađen ima srednju vrijednost 1,467 i standardnu devijaciju 0,054 .



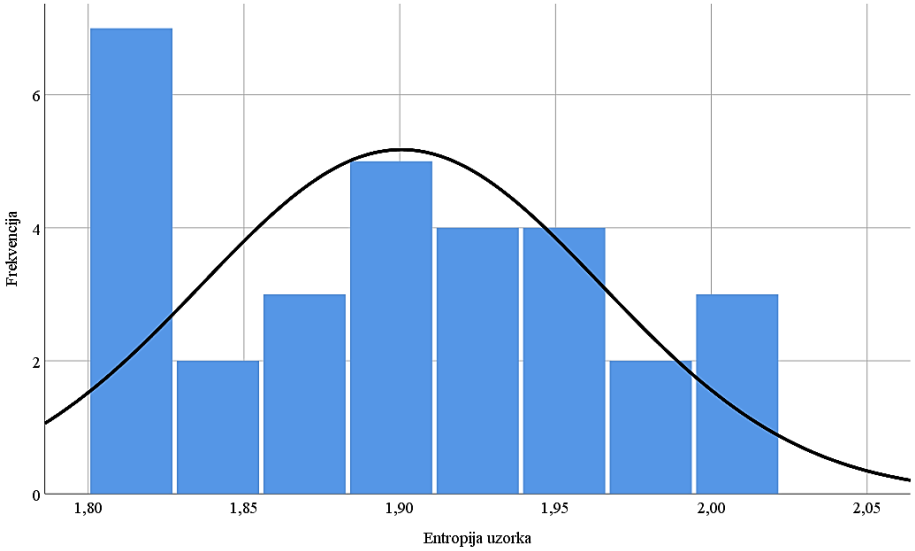
korištene u fazi vrednovanja modela



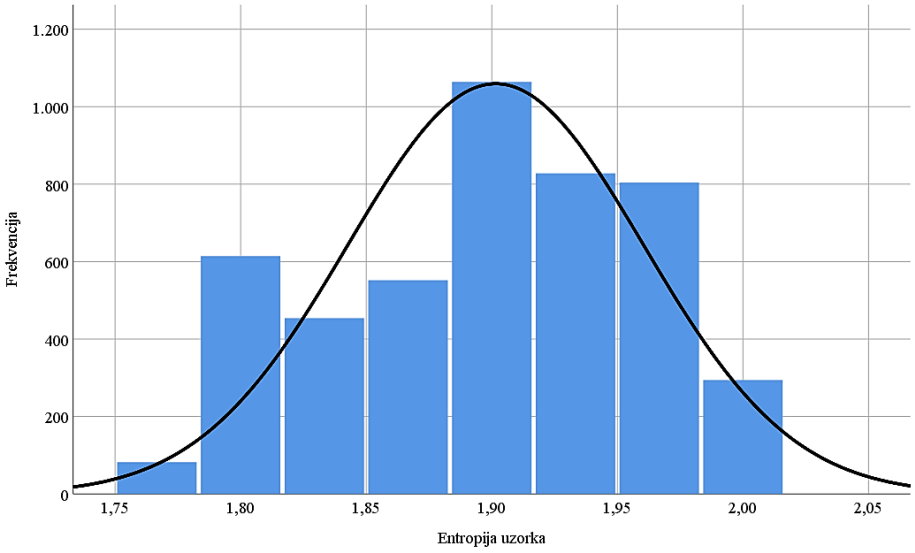
korištene u fazi izgradnje modela

Slika 3.10. Središnji granični teorem za uzorke 7. razreda entropije DNA domene

Na slici 3.11. prikazana je distribucija vrijednosti entropija skupa uzoraka 9. razreda entropije (≥ 1,77778) koji se koriste u fazama vrednovanja i izgradnje modela DNA domene. Skup uzoraka prikazanog razreda entropije DNA domene pomoću kojih se vrednuje model ima srednju vrijednost 1,900 i standardnu devijaciju 0,064. Skup uzoraka prikazanog razreda entropije DNA domene pomoću kojih je model izgrađen ima srednju vrijednost 1,901 i standardnu devijaciju 0,059.



korištene u fazi vrednovanja modela

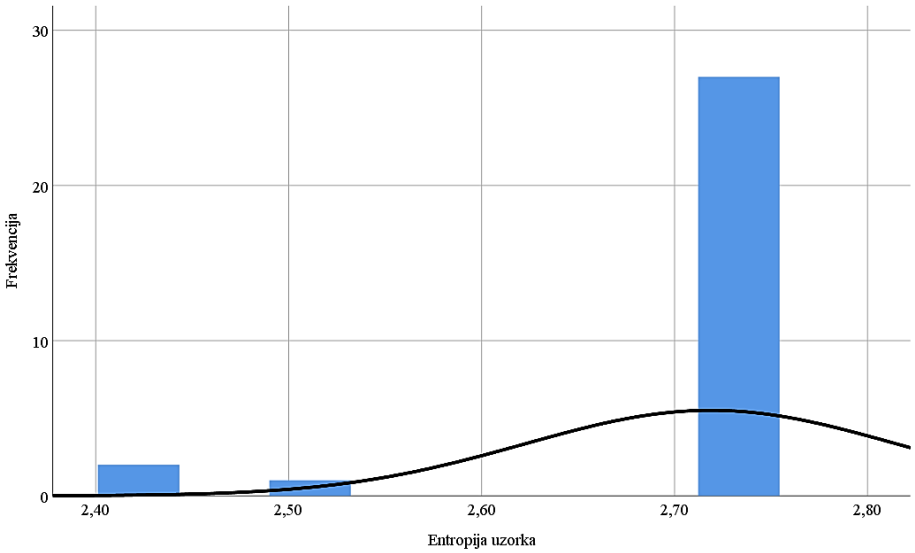


korištene u fazi izgradnje modela

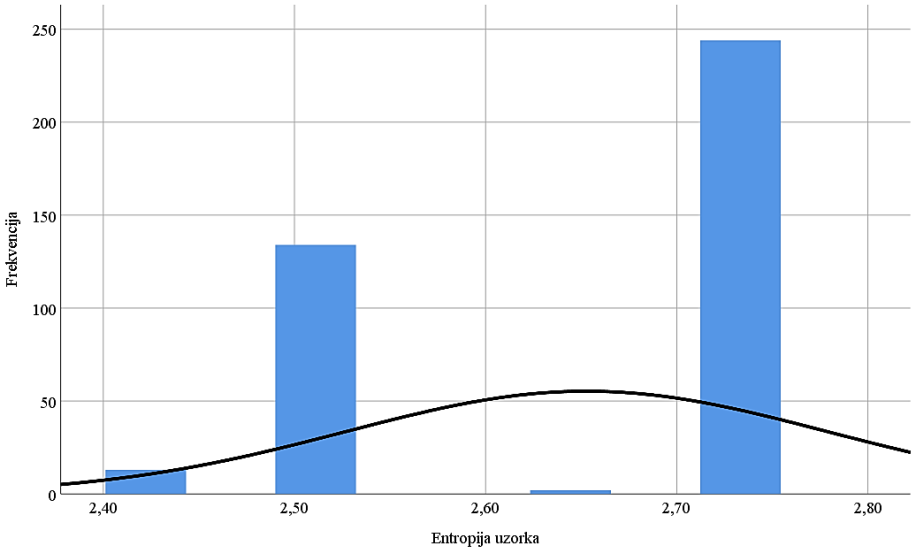
Slika 3.11. Središnji granični teorem za uzorke 9. razreda entropije DNA domene

Na slici 3.12. prikazana je distribucija vrijednosti entropija skupa uzoraka 6. razreda entropije (2,30018–2,76021) koji se koriste u fazama vrednovanja i izgradnje modela domene prirodnog jezika. Skup uzoraka prikazanog razreda entropije domene prirodnog jezika pomoću kojih se vrednuje model ima srednju vrijednost 2,719 i standardnu devijaciju 0,096, a skup uzoraka prikazanog razreda entropije domene

prirodnog jezika pomoću kojih je model izgrađen ima srednju vrijednost 2,653 i standardnu devijaciju 0,126.



korištene u fazi vrednovanja modela

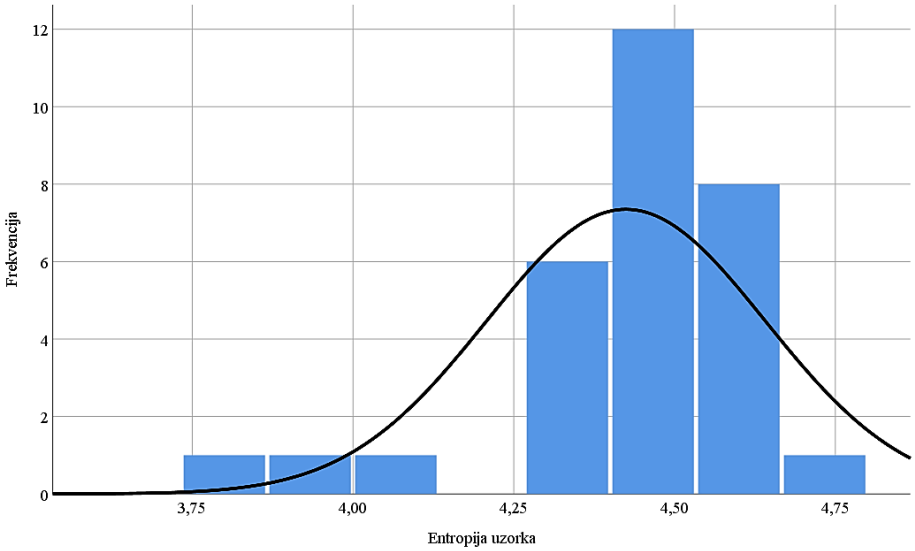


korištene u fazi izgradnje modela

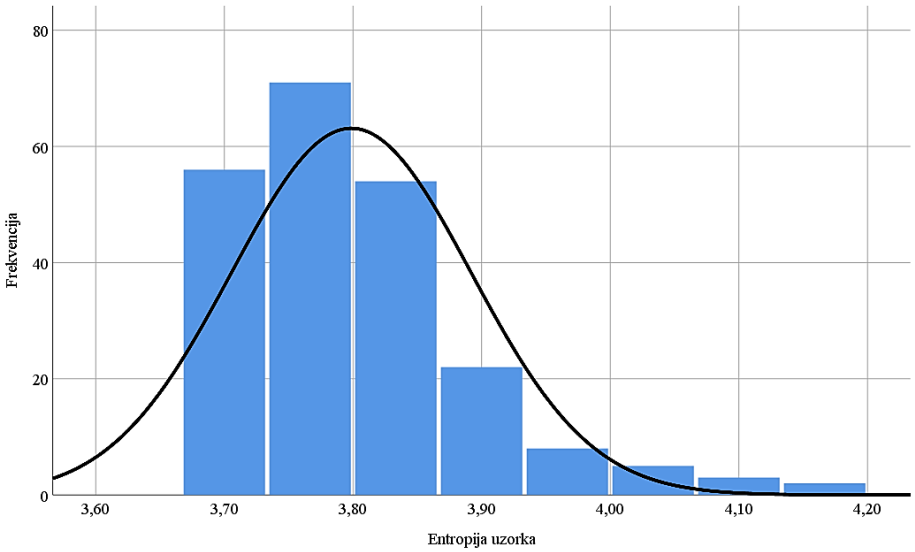
Slika 3.12. Središnji granični teorem za uzorke 6. razreda entropije domene prirodnog jezika

Na slici 3.13. prikazana je distribucija vrijednosti entropija skupa uzoraka 9. razreda entropije (≥ 3,68028) koji se koriste u fazama vrednovanja i izgradnje modela domene prirodnog jezika. Skup uzoraka prikazanog razreda entropije domene prirodnog

jezika pomoću kojih se vrednuje model ima srednju vrijednost 4,424 i standardnu devijaciju 0,217, a skup uzoraka prikazanog razreda entropije domene prirodnog jezika pomoću kojih je model izgrađen ima srednju vrijednost 3,799 i standardnu devijaciju 0,093.



korištene u fazi vrednovanja modela



korištene u fazi izgradnje modela

Slika 3.13. Središnji granični teorem za uzorke 9. razreda entropije domene prirodnog jezika

Nakon provjere da odabrani uzorci korišteni u vrednovanju modela predstavljaju korištene domene idući korak je osnova procesa vrednovanja modela. Osnova procesa vrednovanja izgrađenog modela jeste provjeriti razlikuju li se eksperimentalni rezultati vrednovanja od eksperimentalnih rezultata izgrađenog modela.

3.7.2. Rezultati vrednovanja metodologije

Za potrebe usporedbe dva skupa podataka, eksperimentalnih rezultata vrednovanja izrađenog modela i eksperimentalnih rezultata izgradnje modela za odabrane razrede entropije, korištena je dvostruka Euklidova udaljenost i Pearsonov koeficijent korelacije [111], [138].

Usporedba dva ili više skupova podataka podrazumijeva provjeru sličnosti i udaljenosti između istih. Najjednostavnija ilustracija udaljenosti dvije varijable može se prikazati jednostavnim trigonometrijskim jednadžbama [111].

Za skupove podataka s više varijabli potrebno je proširiti osnovni pristup izračuna udaljenosti dvije varijable. Izračun udaljenosti između više varijabli promatranih skupova podataka naziva se Euklidova udaljenost (d). Euklidova udaljenost koristi se samo za numeričke varijable [111].

Euklidova udaljenost između dvije varijable p_{1i} i p_{2i} iz dva promatrana skupa računa se pomoću jednadžbe (3.3):

$$d = \sqrt{\sum_{i=1}^{v} (p_{1i} - p_{2i})^2} \tag{3.3}$$

Euklidova udaljenost nema ograničenu vrijednost za maksimalnu udaljenost. Vrijednosti 0 znači apsolutni identitet dok je maksimalna vrijednost nepoznata dok se posebno ne izračuna. Zbog toga nije moguće točno odrediti prikazuje li koeficijent malu ili veliku udaljenost. Kako bi se premostio ovaj nedostatak Euklidove udaljenosti koristi se dvostruko skalirana Euklidova udaljenost [111], [141].

Dvostruko skalirana Euklidova udaljenost normalizira izvornu Euklidovu udaljenost u raspon od 0 – 1, pri čemu 1 predstavlja maksimalnu razliku između dviju

varijabli dok 0 predstavlja minimalnu razliku odnosno identičnost promatranih podataka. U nastavku su opisani koraci izračuna dvostruko skalirane Euklidove udaljenosti [141].

Prvi korak u usporedbi dvaju skupova podataka na koje je primijenjena metoda dvostruko skalirane Euklidove udaljenosti je izračunavanje maksimalne moguće kvadratne razlike (*md*) po varijabli *i* od *v* varijabli, pri čemu je *v* broj promatranih varijabli u skupu podataka. Vrijednost Md_i je definirana s Md_i = *(Maksimum za varijablu i-Minimum za varijablu i)*^{2}, pri čemu se 0 (0%) koristi za minimalne i 1 (100%) za maksimalne vrijednosti dvostruko skalirane Euklidove udaljenosti.

U drugom koraku cilj je proizvesti skaliranu varijablu Euklidove udaljenosti (Jednadžba 3.4), gdje se zbroj kvadratnih odstupanja po varijabli dijeli s maksimalnim mogućim odstupanjem za tu varijablu:

$$d_1 = \sqrt{\sum_{i=1}^{v} \left(\frac{(p_{1i} - p_{2i})^2}{md_i} \right)} \tag{3.4}$$

Posljednji korak je dijeljenje skalirane Euklidove udaljenosti s korijenom od *v*, gdje je *v* broj promatranih varijabli (Jednadžba 3.5):

$$d_2 = \frac{\sqrt{\sum_{i=1}^{v} \left(\frac{(p_{1i} - p_{2i})^2}{md_i} \right)}}{\sqrt{v}} \tag{3.5}$$

Nakon izračuna dvostruko skalirane Euklidove udaljenosti (d_2) moguće je izraziti koeficijent sličnosti (Jednadžba 3.6) na način da se dvostruko skalirana Euklidova udaljenost oduzima od 1 [141]–[144]:

$$koeficijent\ sličnosti = 1 - d_2 \tag{3.6}$$

U tablici 3.13. prikazan je izračun skalirane Euklidove udaljenosti za uzorke iz 7. razreda entropije domene DNA. Kolona „Rang lista algoritama izrađenog modela (p_1)“ odnosi se na skup rangiranih podataka izrađenog modela, a kolona „Rang lista algoritma nakon vrednovanja modela (p_2)“ odnosi se na podatke rangiranja pojedinog algoritma

dobivene vrednovanjem modela. Skalirana Euklidova udaljenost izračunava se pomoću jednadžbe (3.4) za sve vrijednosti varijable dva skupa podataka p_1 i p_2 (jedan redak u tablici 3.13. sadrži dvije varijable). U tablici je prikazan samo dio rezultata iz skupa rezultata za 7. razred entropije DNA domene. Na isti način izračunata je skalirana Euklidova udaljenost za ostale razrede korištene u vrednovanju metodologije.

Tablica 3.13. Izračun dvostruko skalirane Euklidove udaljenosti za 7. razred entropije domene DNA

Rang lista algoritama izrađenog modela (p_1)	Rang lista algoritma nakon vrednovanja modela (p_2)	Skalirana Euklidova udaljenost (d_1)
0,58263 (BM)	0,96667 (BM)	0,14749
0,49915 (HOR)	0,26667 (HOR)	0,05405
	…	
0,43718 (KMP)	0,16667 (KMP)	0,07318

Primjenom jednadžbe (3.5) na stupac „Skalirana Euklidova udaljenost (d_1)" u tablici 3.13. dobije se dvostruko skalirana Euklidova udaljenost s iznosom od 0,252.

Primjenom jednadžbe (3.6) oduzimanjem dvostruke Euklidove udaljenosti od 1, (1-0,252) dobije se koeficijent sličnosti od 0,748. Izraženo u postocima sličnost rangiranja algoritama izgrađenih modela za 7. razred iznosi 75%. U tablici 3.14. prikazana je dvostruka Euklidova udaljenost i odgovarajući koeficijent sličnosti usporedbe skupa rezultata vrednovanja izrađenog modela i skupa rezultata izrađenog modela za 7. i 9. razred entropije uzoraka DNA domene. Slične vrijednosti dobiju se i za ostale razrede. Visoki stupanj sličnosti modela validira izgrađeni model DNA domene.

Tablica 3.14. Dvostruka Euklidova udaljenost i koeficijent sličnosti za odabrane razrede DNA domene

Razred	Dvostruko skalirana Euklidova vrijednost (d_2)	Koeficijent sličnosti
7	0,252	0,748 (75%)
9	0,227	0,773 (77%)

U tablici 3.15. prikazana je dvostruka Euklidova udaljenost i odgovarajući koeficijent sličnosti usporedbe skupa rezultata vrednovanja izrađenog modela i skupa rezultata izrađenog modela za 6. i 9. razred entropije domene prirodnog jezika.

Tablica 3.15. Dvostruka Euklidova udaljenost i koeficijent sličnosti za odabrane razrede domene prirodnog jezika

Razred	Dvostruko skalirana Euklidova vrijednost (d_2)	Koeficijent sličnosti
6	0,162	0,838 (84%)
9	0,159	0,841 (84%)

Pretvarajući dvostruko skaliranu Euklidovu udaljenost u kontekst sličnosti, zaključuje se da podaci rangiranja algoritama u izgrađenom modelu imaju visoku razinu sličnosti naspram rezultata vrednovanja izrađenog modela. Tako rezultati za uzorke 7. razreda entropije iz izgrađenog modela za DNA domenu imaju koeficijent sličnosti s rezultatima vrednovanja od 75%, a za uzorke 9. razreda entropije iz izgrađenog modela imaju koeficijent sličnosti s rezultatima vrednovanja od 77%. Visok postotak sličnosti imaju i rezultati za uzorke 6. i 9. razred entropije izgrađenog modela za domenu prirodnog jezika s rezultatima vrednovanja od 84%.

Rezultati broja usporedbi znakova za svaki algoritam korišten u analizi dobivenih prilikom pretraživanja uzoraka u procesu vrednovanja izgrađenog modela za odabrane razrede imaju visok stupanj sličnosti s rezultatima broja usporedbi znakova izgrađenog modela za odabrane razrede. Drugim riječima, udio napravljenih usporedbi znakova uzorka za određeni algoritam unutar kvartila je sličan udjelu usporedbi znakova istog algoritma u izgrađenom modelu.

Nakon izračuna koeficijenta sličnosti napravljena je i dodatna analiza korelacije skupa rezultata izgradnje modela i skupa rezultata vrednovanja izgrađenog modela. Za dodatnu analizu koeficijenta korelacije iskorišten je Pearsonov koeficijent korelacije [138], [139], [144]–[147]. Utvrđivanjem korelacije između vrijednosti dvije varijable može se dobiti prava informacija o njihovoj međusobnoj povezanosti. Povezanost znači da je vrijednost jedne varijable moguće s određenom vjerojatnošću predvidjeti na osnovi saznanja o vrijednosti druge varijable.

Pearsonov koeficijent korelacije označava se malim slovom „r“ i izračunava se jednadžbom (3.7) [111], [138]:

$$r = \frac{\sum xy}{NS_X S_Y} \tag{3.7}$$

Gdje su x i y odstupanja rezultata od aritmetičkih sredina varijabli X i Y, N je broj ispitanika, S_x je pogreška aritmetičke sredine za varijablu X, a S_y pogreška aritmetičke sredine za varijablu Y.

Pearsonov koeficijent korelacije koristi se kada postoji linearna povezanost i neprekidna normalna distribucija između promatranih varijabli nekog skupa. Pearsonov koeficijent korelacije može imati vrijednosti od +1 (savršena pozitivna korelacija) do –1 (savršena negativna korelacija) [146].

U tablici 3.16. prikazani su Pearsonovi koeficijenti korelacije za uzorke 7. i 9. razreda entropije DNA domene, primjenom jednadžbe (3.7) za skupove podataka rezultata vrednovanja izrađenog modela i skupove podataka rezultata izgrađenog modela za odabrane razrede.

Prilikom tumačenja rezultata Pearsonovog koeficijenta korelacije vrijede sljedeći uvjeti za dobiveni koeficijent korelacije u kontekstu veličine [146]:

- do 0,2 vrlo niska korelacija
- od 0,2 do 0,4 niska korelacija
- od 0,4 do 0,6 srednje visoka korelacija
- od 0,6 do 0,8 visoka korelacija i
- od 0,8 do 1 vrlo visoka korelacija

Tablica 3.16. Pearsonov koeficijent korelacije za 7. i 9. razred DNA domene

Razred	DNA domenu (r)
7	0,775
9	0,809

Vrijednosti korelacije za odabrani 7. i 9. razred entropije uzoraka DNA domene pokazuju visoku i vrlo visoku korelaciju između rezultata vrednovanja i rezultata izgrađenog modela.

U tablici 3.17. prikazani su Pearsonovi koeficijenti korelacije za 6. i 9. razred entropije uzoraka domene prirodnog jezika, primjenom jednadžbe (3.7).

Tablica 3.17. Pearsonov koeficijent korelacije za 6. i 9. razred domene prirodnog jezika

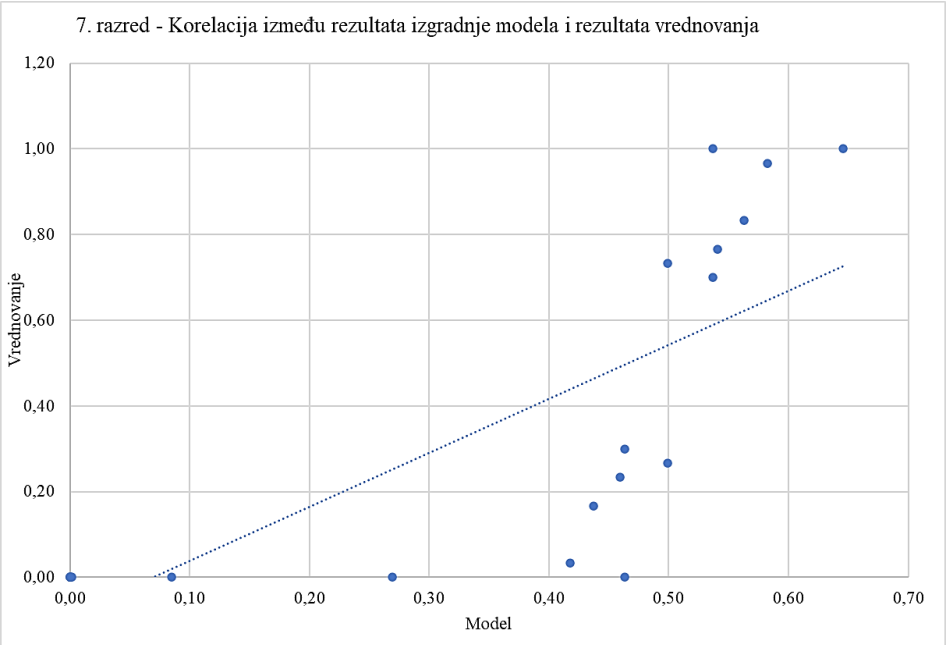
Razred	Domena prirodnog jezika (r)
6	0,911
9	0,934

Vrijednosti korelacije za odabrani 6. i 9. razred entropije uzoraka domene prirodnog jezika pokazuju vrlo visoku korelaciju između rezultata vrednovanja i rezultata izgrađenog modela.

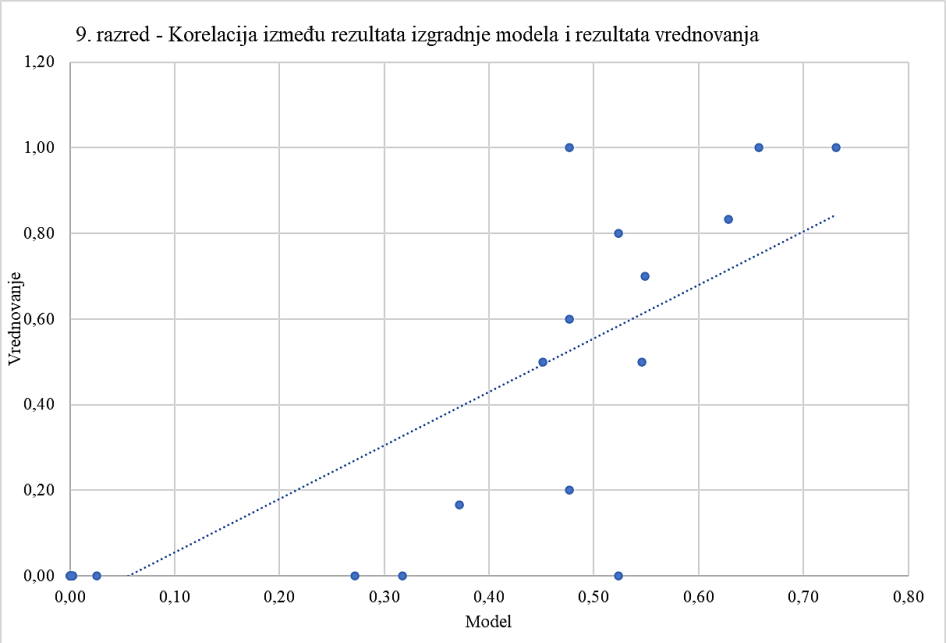
Korelaciju ćemo dodatno analizirati i grafički. Za grafički prikaz korišten je točkasti grafikon (engl. *scatter plot*) [138].

Pozitivna korelacija na točkastom grafikonu lako se uočava na način da su kvadratići raspoređeni tako da se rasprostiru od lijevog donjeg do desnog gornjeg kuta koordinatnog sustava. Kod pozitivne korelacije bolji rezultati u jednoj varijabli povezani su s boljim rezultatima druge varijable. Rast vrijednosti jedne varijable prati rast vrijednosti druge varijable. Negativna korelacija na točkastom dijagramu uočava se na način da su kvadratići raspoređeni tako da se rasprostiru od desnog donjeg do lijevog gornjeg kuta koordinatnog sustava. Rastom vrijednosti u jednoj varijabli moguće je da u drugoj varijabli vrijednosti padaju ili rastu. Opadanje vrijednosti u jednoj varijabli povezano je s porastom vrijednosti odgovarajuće druge varijable [111], [144].

Pearsonov linearni koeficijent korelacije između rezultata vrednovanja izrađenog modela i rezultata izgrađenog modela za 7. razred entropije uzoraka DNA domene prikazan je na slici 3.14. a), a za 9. razred entropije DNA domene na slici 3.14. b).



za 7. razred entropije

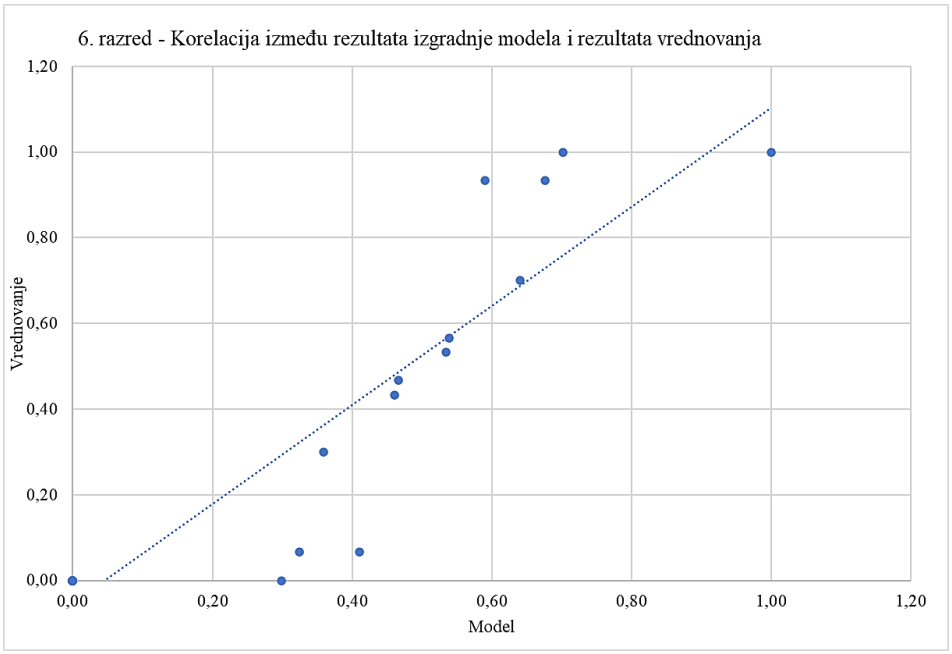


za 9. razred entropije

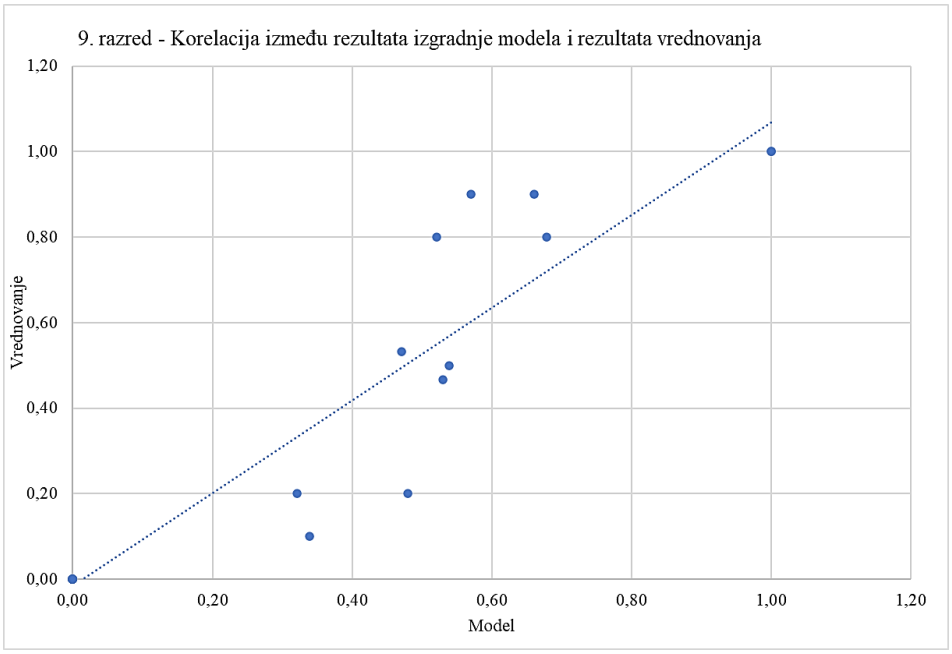
Slika 3.14. Pearsonov linearni koeficijent korelacije domene DNA

Pearsonov linearni koeficijent korelacije između rezultata vrednovanja izrađenog modela i rezultata izgrađenog modela za 6. razred entropije uzoraka domene prirodnog

jezika prikazan je na slici 3.15. a), a za 9. razred entropije domene prirodnog jezika na slici 3.15. b).



za 6. razred entropije



za 9. razred entropije

Slika 3.15. Pearsonov linearni koeficijent korelacije domene prirodnog jezika

Nakon što je dokazan visok koeficijent sličnosti primjenom dvostruko skalirane Euklidove udaljenosti između rezultata broja usporedbi znakova dobivenih prilikom pretraživanja uzoraka u procesu vrednovanja izgrađenog modela za odabrane razrede s rezultatima broja usporedbi znakova izgrađenog modela za odabrane razrede, primjenom Pearsonovog linearnog koeficijenta korelacije dokazana je i snažna pozitivna korelacijska veza između ova dva skupa rezultata.

Rezultati broja usporedbi znakova prilikom vrednovanja izgrađenih modela pokazuju da se predložena metodologija može koristiti u praksi za razvoj modela za odabir učinkovitog algoritma za egzaktno podudaranje znakovnih nizova unutar određene domene primjene.

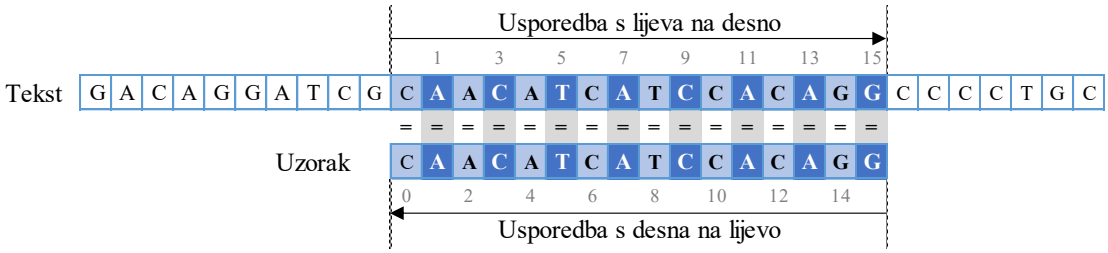
4. NOVI ALGORITAM S PREDOBRADOM UZORKA

Tijekom razvoja metodologije analizirani su postojeći algoritmi za egzaktno podudaranje znakovnih nizova u tekstu te su identificirana dodatna svojstva uzorka koje je moguće izdvojiti analiziranjem uzorka u fazi predobrade i koristiti za preskakanje usporedbi karaktera pri traženju uzorka u tekstu. Sintezom stečenih saznanja opisanih u prethodnim poglavljima u jednu cjelinu osmišljena je ideja novog algoritma za egzaktno podudaranje znakovnih nizova koji koriste pristup usporedbe znakova s *online* algoritamskim rješenjem pretrage. Novi algoritam je nazvan Algoritam temeljen na indeksima znakova *(CIB − Character Index-Based algorithm)*.

Ideja našeg novog algoritma leži u usporedbi *neparnih* i *parnih* indeksa pozicija prozora uzorka s tekstom u cilju pronalaska svih pojavnosti uzorka u tekstu. Novi algoritam ima dvije faze izvršavanja. Prva faza je faza predobrade uzorka u kojoj se izdvajaju potrebne informacije iz uzorka koje omogućavaju preskakanje nepotrebnih usporedbi u fazi pretrage. To su informacije o prvoj i drugoj poziciji svakog znaka abecede u uzorku. Druga faza je faza pretrage.

U fazi predobrade algoritma svojstva uzoraka koja su heuristički identificirana i definirana u formi pravila pomaka predstavljaju temelj novog algoritma. Novi algoritam ima za cilj smanjiti broj usporedbi znakova za pronalazak uzorka u tekstu preskačući nepotrebne usporedbe istih te tako povećati učinkovitost pretrage.

U fazi pretrage prvo se neparne pozicije uspoređuju s lijeva na desno, a ako se sve neparne pozicije podudare tj. budu jednake, algoritam nastavlja uspoređivati parne pozicije s desna na lijevo, u suprotnom smjeru. Ako se svi znakovi neparnih i parnih pozicija podudaraju tada je uzorak pronađen. Na slici 4.1. prikazan je primjer pronađenog uzorka s navedenim rednim brojevima usporedbe svakog znaka uzorka i teksta.



Slika 4.1. Pronađeni uzorak CIB algoritmom

Predobrada uzorka pronalazi prvu i drugu poziciju svakog znaka. Kako bismo predstavili ponašanje algoritma koristimo termin *tablica predobrade* u koju se spremaju podaci predobrade uzorka, a koja sadrži broj redaka koji je jednak veličini abecede (jedan redak za svaki znak u abecedi), koja je nakon postupka predobrade uzorka ispunjena s dvije varijable u svakom retku (Slika 4.2, prikaz tablice predobrade). Varijable su indeksi prve i druge pozicije znaka abecede u uzorku, dok se kao početni indeks znaka koristi vrijednost *0*, a ne vrijednost *1*. Ako znak ne postoji u uzorku, za vrijednosti varijabli indeksa prve i druge pozicije znaka abecede u uzorku, u tablici predobrade nalazi se vrijednost *-1*. Ako znak ne postoji na drugom mjestu, vrijednost varijable indeksa druge pozicije znaka abecede u uzorku je -1.

Promatranjem prve dvije pozicije znaka abecede u uzorku moguće je smanjiti broj potrebnih usporedbi znakova prilikom pronalaženja uzorka. Na ovaj će način znakovi koji se ne javljaju često u riječima (primjerice u domeni prirodnog jezika) vjerojatno nedostajati barem na drugoj poziciji pojavnosti promatranog znaka čime se stječu uvjeti za siguran pomak uzorka uslijed nepodudaranja znakova.

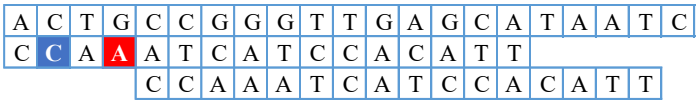
U fazi pretraživanja uzorka potrebno je pamtiti indeks pozicije zadnjeg pročitanog znaka teksta naspram prozora pretrage. Kako bismo predstavili ponašanje algoritma koristimo termin *tablica pretraživanja* za spremanje indeksa pozicije zadnjeg pročitanog znaka teksta naspram prozora pretrage na svakoj usporedbi znakova (Tablica 4.2., prikaz tablice pretraživanja). Početne vrijednosti tablice pretraživanja teksta za svaki znak abecede domene su *-1*, a kao početni indeks znaka koristi se vrijednost *0*, a ne vrijednost *1*. Nakon svakog pomaka uzorka, tablica pretraživanja inicijalizira se na početne vrijednosti.

U slučaju da se nepodudaranje dogodi u bilo kojem koraku, primjenjuju se odgovarajuća heuristička pravila pomaka uzorka formalizirana na osnovu informacija

dobivenih u fazi predobrade uzorka. Primjenom heurističkih pravila pomaka (detaljno opisanih u nastavku) cilj je sigurno pomaknuti uzorak na novu poziciju nakon nepodudaranja znaka teksta s znakom uzorka s obzirom na indeks prve i/ili druge pozicije znaka abecede u uzorku za nepodudarni znak teksta. Siguran pomak uzorka podrazumijeva da se uzorak pozicionirao nakon pomaka tako da se ne dogodi preskakanje pokušaja i uzorak ne bude pronađen. Nakon sigurnog pomaka uzorka naspram teksta proces usporedbe kreće iz početka. Pravila pomaka uzorka u slučaju nepodudaranja znakova uzorka i teksta čine temelj novog algoritma.

Kada se znakovi na bilo kojoj poziciji ne podudaraju glavni cilj je napraviti siguran pomak s maksimalnom duljinom. U tu svrhu prvo se u tablici predobrade pronađu vrijednosti indeksa prve i druge pozicije znaka abecede u uzorku nepodudarnog znaka teksta. Nakon pronalaska navedenih vrijednosti u tablici predobrade za pomak uzorka koriste se sljedeća pravila pomaka:

Pravilo 1. – Ako nepodudarni znak teksta ne postoji u uzorku cilj je napraviti pomak uzorka prema kojem se uzorak pomiče odmah iza nepodudarnog znaka iz teksta. U ovome slučaju tablica predobrade ima vrijednost *-1* za varijable indeksa prve i druge pozicije znaka abecede u uzorku nepodudarnog znaka, uzorak se pomiče točno iza nepodudarnog znaka teksta. Ilustracija prvog pravila pomaka prikazana je na slici 4.2. na primjeru traženja DNA uzorka CCAAATCATCCACATT. Pretraživanje kreće s lijeva na desno po parnim pozicijama znakova u uzorku. Prvi parni znak je C i podudara se znakom u tekstu.



Tablica predobrade

Znak abecede	Prva pozicija	Druga pozicija
A	2	3
C	0	1
G	***-1***	***-1***
T	5	8

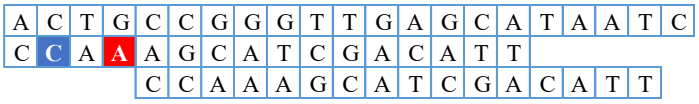
Slika 4.2. Prvo pravilo pomaka uzorka CIB algoritma

Primjer primjene pravila je usporedba sljedeće parne pozicije znaka uzorka, četvrtog znaka uzorka (u ovom primjeru slovo A) i odgovarajućeg znaka teksta (slovo G) gdje se javlja nepodudarnost. Algoritam u tablici predobrade za nepodudarni znak u tekstu (za ovaj primjer slovo G) dohvaća indeks prve i druge pozicije znaka abecede u uzorku

koje u ovom primjeru iznose *-1*. Primjenom pravila uzorak pomiče tako da se prvi znak uzorka poravna s prvim sljedećim znakom nakon nepodudarnog znaka iz teksta.

Pravilo 2. - Ako nepodudarni znak iz teksta postoji u uzorku cilj je napraviti pomak uzorka prema kojem se uzorak pomiče ovisno o indeksu prve ili druge pozicije znaka abecede u uzorku za nepodudarni znak teksta u tablici predobrade. U ovome slučaju vrijednosti varijabli indeksa prve i/ili druge pozicije znaka abecede u uzorku za nepodudarni znak teksta u tablici predobrade razlikuju se od *-1*, te vrijedi sljedeće:

a) Ako su vrijednosti varijabli indeksa prve i/ili druge pozicije znaka abecede u uzorku za nepodudarni znak teksta veće od trenutnog indeksa usporedbe znaka uzorka ili indeks druge pozicije znaka abecede u uzorku ne postoji (vrijednost varijable indeksa druge pozicije znaka abecede u uzorku *-1*), uzorak se pomiče točno iza nepodudarnog znaka teksta. Ilustracija pravila pomaka CIB algoritma prikazana je na slici 4.3. za uzorak CCAAAGCATCGACATT.



Tablica predobrade

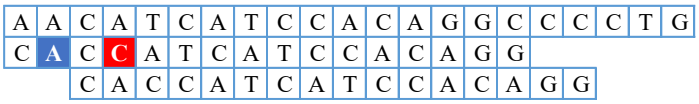
Znak abecede	Prva pozicija	Druga pozicija
A	2	3
C	0	1
G	***5***	***10***
T	8	14

Slika 4.3. Drugo a) pravilo pomaka uzorka CIB algoritma

Primjer primjene pravila je usporedba četvrtog znaka uzorka (u ovom primjeru slovo A) i odgovarajućeg znaka teksta (slovo G) nakon čega se dogodi nepodudarnost, a vrijednosti varijabli indeksa prve pozicije znaka abecede u uzorku iz tablice predobrade za nepodudarni znak teksta, slovo G, iznosi *5*, a druge pozicije iznosi *10*. Primjenom pravila cijeli uzorak se pomiče tako da se prvi znak uzorka poravna s prvim sljedećim znakom nakon nepodudarnog znaka iz teksta.

b) Ako su vrijednosti varijabli indeksa prve i druge pozicije znaka abecede u uzorku za nepodudarni znak u tablici predobrade takve da je indeks prve pozicije manji, a indeks druge pozicije veći od trenutnog indeksa usporedbe znaka uzorka (ili indeks druge pozicije znaka abecede u uzorku ne postoji, pa je vrijednost varijable

indeksa druge pozicije znaka abecede u uzorku -*1*), uzorak se pomiče za vrijednost iznosa trenutnog indeksa usporedbe uzorka umanjenog za vrijednost varijable indeksa prve pozicije znaka u uzorku. Usporedba na sljedećem poravnanju je poznata i nije potrebno raditi usporedbu zapamćene pozicije. Ilustracija pravila pomaka CIB algoritma prikazana je na slici 4.4. na primjeru traženja DNA uzorka CACCATCATCCACAGG.



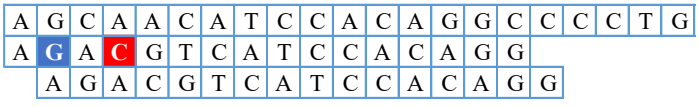
Tablica predobrade

Znak abecede	Prva pozicija	Druga pozicija
A	***1***	***4***
C	0	2
G	14	15
T	5	8

Slika 4.4. Drugo b) pravilo pomaka uzorka CIB algoritma

Primjer primjene pravila je usporedba četvrtog znaka uzorka (u ovom primjeru slovo C) i odgovarajućeg znaka teksta (slovo A) nakon čega se dogodi nepodudarnost, a kako rezultat iz tablice predobrade za varijable indeksa prve pozicije nepodudarnog znaka, u ovom slučaju znaka A, teksta iznosi *1*, a druge iznosi *4* potrebno je pomaknuti uzorak za dva mjesta.

c) Ako su vrijednosti varijabli indeksa prve i druge pozicije znaka abecede u uzorku za nepodudarni znak teksta u tablici predobrade takve da su vrijednosti indeksa pozicija manje od trenutnog indeksa usporedbe znaka uzorka, uzorak se pomiče za jedno mjesto. Ilustracija pravila pomaka uzorka CIB algoritma prikazana je na slici 4.5. na primjeru traženja DNA uzorka AGACGTCATCCACAGG.



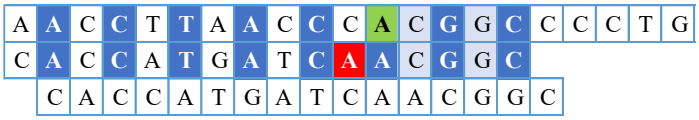
Tablica predobrade

Znak abecede	Prva pozicija	Druga pozicija
A	**0**	**2**
C	3	6
G	1	4
T	5	8

Slika 4.5. Drugo c) pravilo pomaka uzorka CIB algoritma

Primjer primjene pravila je usporedba četvrtog znaka uzorka (u ovom primjeru slovo C) i odgovarajućeg znaka teksta (slovo A), te se dogodila nepodudarnost, a kako rezultat iz tablice predobrade za vrijednosti varijabli indeksa prve pozicije znaka abecede za nepodudarni znak teksta, slovo A, iznosi *0*, a druge pozicije iznosi *2* potrebno je pomaknuti uzorak za jedno mjesto.

Pravilo 3. - Kada su sve neparne pozicije jednake nastavlja se pretraga parnih pozicija u suprotnom smjeru s desna na lijevo po uzorku. U slučaju nepodudaranja znakova uzorka i teksta cilj je pomaknuti uzorak na način da se u obzir uzme pozicija pojave znaka uzorka u tekstu prilikom provjere po neparnim i parnim pozicijama unutar prozora pretrage. To znači da se u slučaju nepodudaranja znakova uzorka i teksta vrši dodatna provjera postoji li nepodudarni znak uzorka u tablici za spremanje indeksa pozicije zadnjeg pročitanog znaka teksta i ako postoji uzorak se pomiče na poziciju zadnjeg pročitanog znaka teksta tj. razliku spremljene vrijednosti indeksa pozicije zadnjeg znaka teksta i vrijednosti trenutnog indeksa usporedbe uzorka. Ako postoji pozicija nepodudarnog znaka uzorka u tablici za spremanje indeksa pozicije zadnjeg pročitanog znaka teksta, ista se uzima u obzir kod sljedećeg poravnanja uzorka. Usporedba na sljedećem poravnanju je poznata i nije potrebno raditi usporedbu zapamćene pozicije. Ilustracija pravila pomaka CIB algoritma prikazana je na slici 4.6. na primjeru traženja DNA uzorka CACCATGATCAACGGC.



Tablica predobrade

Znak abecede	Prva pozicija	Druga pozicija
A	*1*	*4*
C	**0**	**2**
G	6	13
T	5	8

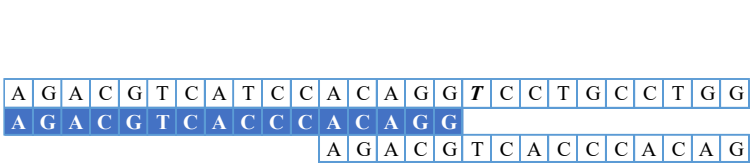
Slika 4.6. Treće pravilo pomaka uzorka CIB algoritma

Primjer primjene pravila je nepodudaranje na parnoj poziciji (u ovom primjeru slovo A) i provjera postojanja za znak uzorka u tablici pretraživanja u koju se sprema indeks pozicije zadnjeg pročitanog znaka teksta. Za nepodudarni znak uzorka (u ovom primjeru slovo A) koje se u fazi pretrage po neparnim pozicijama nalazilo na *1*, *7* i *11* indeksu, vrijednost indeksa zadnje pozicije u tablici pretraživanja iznosi *11*, a vrijednost

trenutnog indeksa usporedbe uzorka iznosi *10*. Potrebno je pomaknuti uzorak za jedno mjesto.

Pravilo 4. - Nakon pronalaska uzorka cilj je sigurno pomaknuti uzorak za maksimalno mogućih mjesta naspram teksta. U ovome slučaju uzorak se pomiče na način da se pročita sljedeći znak teksta i za njega se u tablici predobrade pronađu vrijednosti varijabli indeksa prve i druge pozicije znaka abecede u uzorku. Novi granični indeks teksta se poravna sa vrijednostima indeksa prve pozicije znaka abecede u uzorku ili ako vrijednost indeksa druge pozicije iznosi *-1* uzorak se pomakne za određeni broj mjesta. Nakon pronalaska vrijednosti varijabli indeksa prve i/ili druge pozicije znaka abecede u uzorku uzorak se pomiče na sljedeći način:

a) Ako su vrijednosti varijabli indeksa prve i druge pozicije znaka abecede u uzorku za sljedeći znak teksta u tablici predobrade takve da indeks prve pozicije ima vrijednost različitu od *-1*, a indeks druge pozicije znaka abecede u uzorku ne postoji (to znači da je vrijednost varijable indeksa druge pozicije znaka abecede u uzorku -1), uzorak se pomiče za vrijednost iznosa duljine uzorka umanjenu za vrijednost iznosa indeksa prve pozicije znaka abecede u uzorku. Ilustracija pravila pomaka uzorka CIB algoritma prikazana je na slici 4.7. na primjeru traženja DNA uzorka AGACGTCACCCACAG.



Tablica predobrade

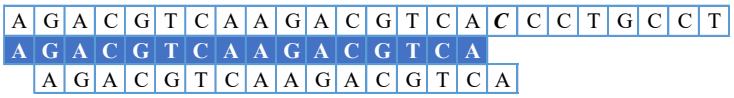
Znak abecede	Prva pozicija	Druga pozicija
A	0	2
C	3	6
G	1	4
T	***5***	***-1***

Slika 4.7. Četvrto a) pravilo pomaka uzorka CIB algoritma

Primjer pravila je odabir sljedećeg znaka teksta nakon pronalaska uzorka (u ovom primjeru slovo T). Rezultat iz tablice predobrade za vrijednosti varijabli indeksa prve pozicije znaka abecede za nepodudarni znak teksta, u ovom slučaju T, u uzorku iznosi *5*, a druge pozicije iznosi *-1* stoga je potrebno pomaknuti uzorak za jedanaest mjesta (duljina uzorka *16* minus pozicija znaka T *5*).

b) Ako su vrijednosti varijabli indeksa prve i druge pozicije znaka abecede u uzorku za sljedeći znak teksta u tablici predobrade takve da indeksi prve i druge pozicije imaju

vrijednosti različite od *-1*, uzorak se pomiče za jedno mjesto. Ilustracija pravila pomaka uzorka CIB algoritma prikazana je na slici 4.8. na primjeru traženja DNA uzorka AGACGTCAAGACGTCA.



Tablica predobrade

Znak abecede	Prva pozicija	Druga pozicija
A	0	2
C	***3***	***6***
G	1	4
T	5	13

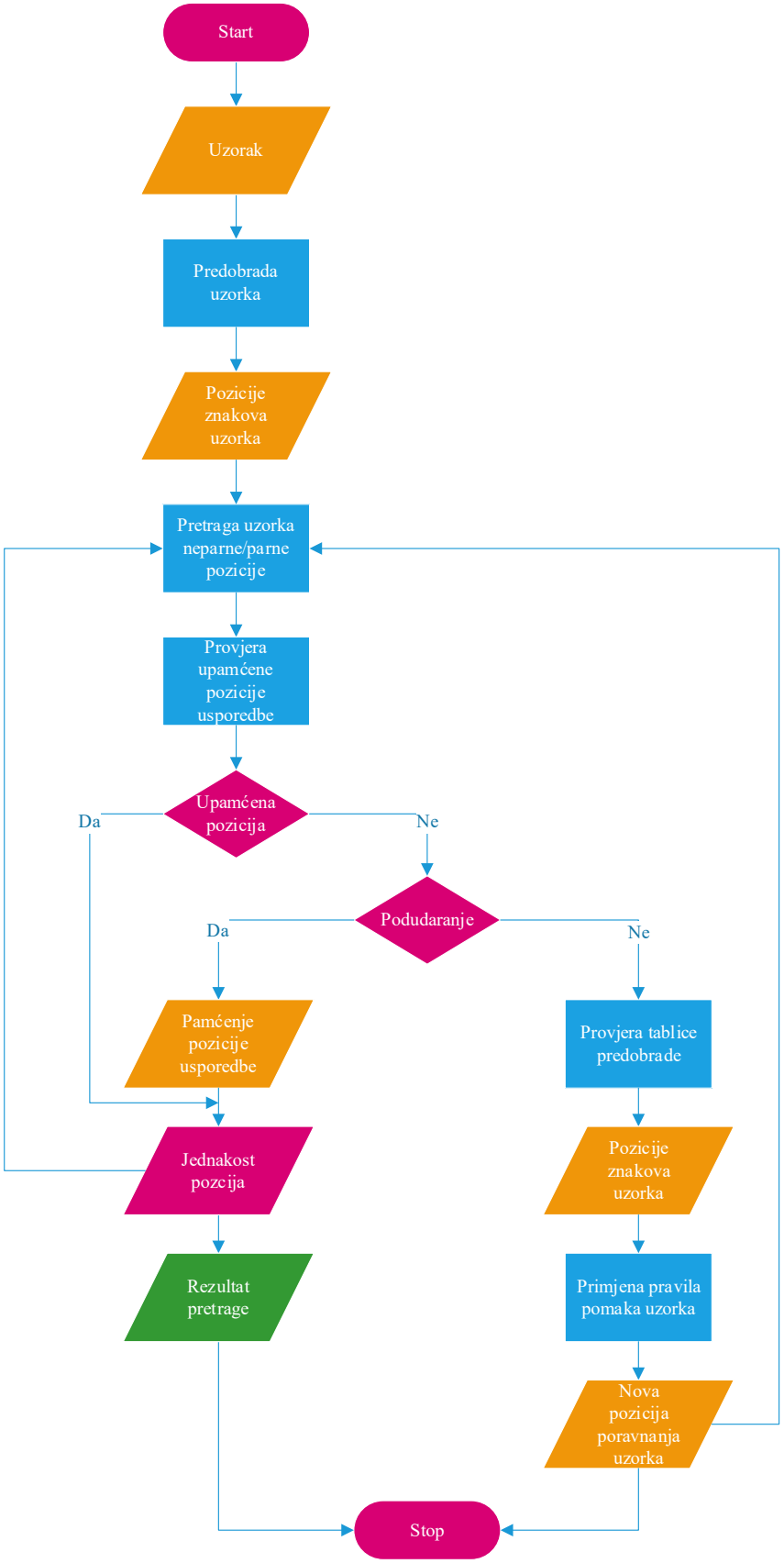
Slika 4.8. Četvrto b) pravilo pomaka uzorka CIB algoritma

Primjer pravila je odabir sljedećeg znaka teksta nakon pronalaska uzorka (u ovom primjeru slovo C). Ako rezultat iz tablice predobrade za vrijednosti varijabli indeksa prve pozicije znaka abecede za nepodudarni znak teksta iznosi *3*, a druge iznosi *6* potrebno je pomaknuti uzorak za jedno mjesto.

c) Ako vrijednosti varijabli indeksa prve i druge pozicije iznose *-1*, potrebno je uzeti sljedeći znak teksta i ponoviti proces pronalaska vrijednosti indeksa prve i druge pozicije idućeg znaka teksta u tablici predobrade.

Proces pretrage uzorka u tekstu se ponavlja sve dok se ne dođe do kraja teksta.

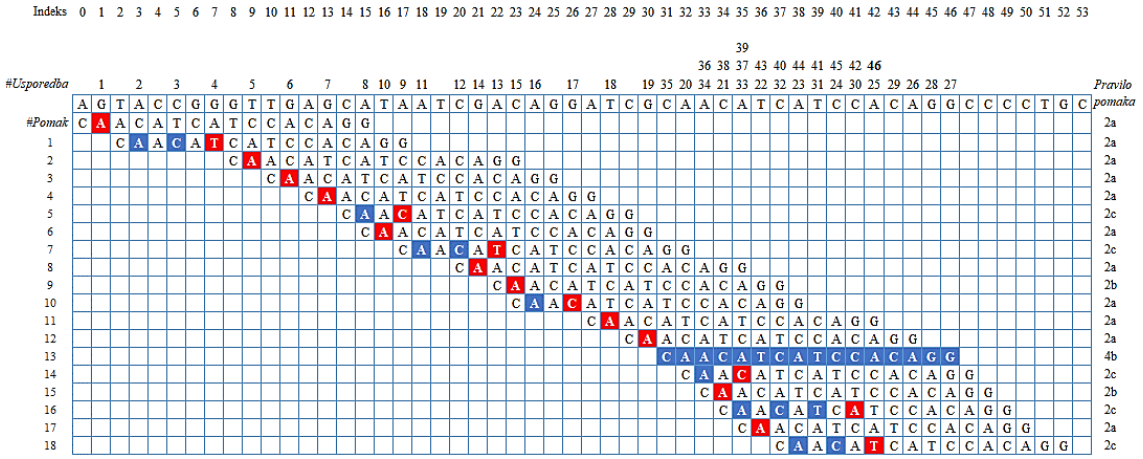
Grafički prikaz slijeda instrukcija eksperimentalne implementacije algoritma prikazan je na slici 4.9.



Slika 4.9. Grafički prikaz slijeda instrukcija CIB algoritma

Na sljedećem primjeru sa slike 4.10. prezentirano je kako algoritam pronalazi uzorak u *54* znaka dugom tekstu. Na prikazanom primjeru napravi se ukupno 46 usporedbi znakova tijekom pretraživanja uzorka „CAACATCATCCACAGG“. Ukupan broj pomaka uzorka koje je potrebno napraviti kako bi se napravila pretraga za navedeni primjer je *18*.

Na slici je svaki pomak uzorka duž teksta prikazan kao novi red označen oznakom *#Pomak*. Svaka usporedba znakova navedena je na vrhu slike rednim brojem usporedbe oznakom *#Usporedba*. Za svaki pomak uzorka naspram teksta navedeno je i primijenjeno pravilo pomaka nakon nepodudaranja znakova uzorka i teksta. Crvenom bojom su označene nepodudarne usporedbe znakova koje su pokrivene prethodno definiranim pravilima algoritma, a plavom su označene podudarne usporedbe znakova uzorka i teksta.



Slika 4.10. Primjer pretrage uzorka CIB algoritmom

Za navedeni primjer, tablica predobrade uzorka "CAACATCATCCACAGG" prikazana je u tablici 4.1. Vrijednosti varijable indeksa prve pozicije znaka abecede u uzorku za znak A iznosi *1*, a druge iznosi *2*, za znak C vrijednost indeksa prve pozicije iznosi *0*, a druge *3*, itd.

Tablica 4.1. Tablica predobrade za uzorak iz primjera sa slike 4.10.

Znak abecede	Indeks prve pozicije	Indeks druge pozicije
A	1	2
C	0	3
G	14	15
T	5	8

Podaci u tablici 4.2. predstavljaju poziciju znaka iz primjera pretraživanja niza prikazanog na slici 4.10. Tablica pretraživanja za spremanje indeksa pozicije zadnjeg pročitanog znaka teksta naspram prozora pretrage uvijek ima samo jedan stupac koji sadrži indeks zadnje pročitanog znaka teksta. Tablica na slici sadrži sve vrijednosti koje su se spremile u tablicu pretraživanja kroz postupak usporedbe znakova u različitim stupcima za svaki pomak uzorka. Na primjeru sa slike 4.10. postoji više stupaca za neke pomake uzorka (1, 5, 7, 10, 13, 14, 16 i 18) koji odgovaraju usporedbi znakova nakon poravnanja. U tablici na slici je prikazano kako se vrijednosti u tablici pretraživanja mijenjaju tijekom pretraživanja.

Tablica 4.2. Tablica pretraživanja za spremanje pozicija indeksa zadnjeg pročitanog znaka teksta za primjer sa slike 4.10.

#Pomak	**0**	**1**			**2**	**3**	**4**	**5**		**6**	**7**			**8**	**9**	**10**		**11**	**12**	**13**			
#Usporedba	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
A	-1	1	1	1	-1	-1	-1	1	3	-1	1	1	5	-1	-1	1	1	-1	-1	1	1	1	7
C	-1	-1	3	3	-1	-1	-1	-1	-1	-1	-1	3	3	-1	1	-1	-1	-1	-1	-1	3	3	3
G	1	-1	-1	5	-1	1	1	-1	-1	-1	-1	-1	-1	1	-1	-1	3	-1	1	-1	-1	-1	-1
T	-1	-1	-1	-1	1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	5	5

#Pomak	**13**												**14**	**15**	**16**				**17**	**18**			
#Usporedba	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
A	7	11	13	13	13	13	13	13	13	4	2	2	1	3	-1	1	1	1	1	-1	1	1	5
C	9	9	9	9	9	12	10	10	6	6	6	0	-1	-1	1	-1	3	3	7	-1	-1	3	3
G	-1	-1	-1	15	14	14	14	14	14	14	14	14	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
T	5	5	5	5	5	5	5	8	8	8	8	8	-1	-1	-1	-1	-1	5	5	1	-1	-1	-1

4.1. Vrednovanje novog algoritma

Novi algoritam implementiran je u programskom jeziku C pomoću Microsoft Visual Studio razvojnog okruženja. To je programski jezik opće namjene za rješavanje zadataka, izradu drivera, operacijskih sustava, tekst procesora ili igara. Zbog tih karakteristika C je među popularnijim programskim jezicima [94]. Microsoft Visual Studio je razvojno okruženje namijenjeno razvoju raznih vrsta aplikacija na Windows platformi [148].

Nakon izgradnje novog algoritma napravljeno je njegovo vrednovanje. Prilikom vrednovanja algoritma postavljena hipoteza kako postoje svojstva uzorka, koja se mogu heuristički identificirati i definirati, a koja do sada nisu korištena u algoritmima za egzaktno podudaranje znakovnih nizova, je potvrđena validacijom točnosti novog algoritma, pri čemu je svaki traženi uzorak koji je postojao u tekstovima korištenim u vrednovanju pronađen. Također je vrednovanjem novog algoritma ocijenjena kompetitivnost istoga u odnosu na druge uobičajene algoritme koji pripadaju istoj grupi algoritama koji koriste pristup usporedbe znakova s *online* algoritamskim rješenjem pretrage kako bi valorizirali koliko postavljena pravila algoritma optimiziraju broj potrebnih usporedbi znakova odnosno tako izražene performanse algoritma [149]–[151].

Vremenske i prostorne složenosti usporednih algoritama koji se koriste u ovom istraživanju prikazani su u tablici 4.3. [18], [33], [41]:

Tablica 4.3. Vremenska i prostorna složenost algoritama korištenih u istraživanju

	PRETRAGA	PREDOBRADA	
Algoritam	Vremenska složenost	Vremenska složenost	Prostorna složenost
BF (*brute force, näive*)	$O(mn)$	-	-
BM (Boyer-Moore)	$O(mn)$	$O(m+\sigma)$	$O(m+\sigma)$
KMP (Knuth-Morris-Pratt)	$O(n+m)$	$O(m)$	$O(m)$
AC (Apostolico-Crochemore)	$O(n)$	$O(m)$	$O(m)$
QS (Quick Search)	$O(mn)$	$O(m+\sigma)$	$O(\sigma)$
MP (Morris-Pratt)	$O(n+m)$	$O(m)$	$O(m)$
HOR (Horspool)	$O(mn)$	$O(m+\sigma)$	$O(\sigma)$
CIB (Character Index-Based)	$O(mn)$	$O(m+\sigma)$	$O(m+\sigma)$

U teorijskoj analizi vremenske složenosti prikazanoj u tablici 4.3. prikazano je ukupno vrijeme potrebno za izvršavanje algoritma s obzirom na veličinu ulaza koja se sastoji od veličine teksta (n) i veličine uzorka (m). U analizi prostorne složenosti algoritma prikazan je zauzeti memorijski prostor s obzirom na veličinu ulaza koji se sastoji od veličine uzorka (m) i pomoćnog prostora algoritma (σ). Prostorna složenost sastoji se od ukupnog i pomoćnog prostora. Pomoćni prostor je dodatni ili privremeni prostor koji koristi algoritam i označen je sa σ, a veličina mu varira u ovisnosti od veličine abecede i izvedbe algoritma. Glavna je razlika između ukupnog i pomoćnog prostora u tome što složenost prostora kvantificira ukupni prostor koji koristi algoritam, a pomoćni prostor kvantificira dodatni prostor koji se koristi u algoritmu osim zadanog ulaza i služi za rješenje određenog problema [8], [10], [89]. U tablici 4.3. za prostornu složenost nekih algoritama kao zadani ulaz nije prikazan pomoćni prostor jer ne utječe na prostornu složenost algoritma. Analiza izvedbe preuzetih gotovih algoritama nije predmet ovog istraživanja.

Napravljeno vrednovanje novog algoritma temeljeno je na procjeni performansi algoritma pomoću tri pristupa. Prvi pristup je formalna metrika koja mjeri broj usporedbi znakova (CC) u procesu pronalaženja uzorka u nizu. Drugi pristup je empirijska metrika koja mjeri potrebno vrijeme za izvršavanje algoritma. Treći pristup je empirijska metrika koja mjeri potrebnu količinu radne memorije za izvršavanje algoritma.

Potrebna količina radne memorije je mjera (RAM – engl. *Random Access Memory*) koju određeni program koristi tijekom svog izvršavanja. Ovo istraživanje mjeri trenutnu veličinu iskorištene memorije u bajtovima, koja se ne može dijeliti s drugim procesima (engl. *private bytes*) [99].

Da bi se napravilo vrednovanje novog algoritma potrebno je odabrati reprezentativne tekstove i uzorke. Odabrani reprezentativni tekstovi, što uključuje tekst iz DNA domene i domene prirodnog jezika dio su prethodno analiziranih postojećih okvira za testiranje algoritama podudaranja znakovnih nizova [18].

Za reprezentativni tekst DNA domene odabran je javno dostupni tekst živućeg organizama Escherichia coli iz DNA domene (veličina 4.897.531 bajtova) [82].

Za domenu prirodnog jezika odabrani reprezentativni tekst na engleskom jeziku javno je dostupan iz *Canterbury Corpusa* [134]. Reprezentativni tekstovi domene prirodnog jezika je Biblija (Kralj James verzija, veličina 4.047.392 bajtova) [84].

Nakon odabira reprezentativnih tekstova odabrani su i uzorci potrebni za vrednovanje novog algoritma. Analiza je napravljena na 511 uzoraka za DNA domenu i 550 uzoraka za domenu prirodnog jezika. Duljina uzoraka kreće se od 2 znaka do 1024 znaka sa slijedom duljina uzoraka od 2, 4, 8, 16, 32, 64, 128, 256, 512 i 1024 znaka. Prema zakonu velikih brojeva, odnosno primjenom središnjeg graničnog teorema, definiran je potreban broj uzoraka za svaku duljinu uzorka prema kojima će varijabla imati normalnu raspodjelu [111]. Za svaku duljinu uzorka iz slijeda prema središnjem graničnom teoremu odabran je skup od 55 uzoraka (za duljinu uzorka od 2 znaka DNA domene postoji samo 16 uzoraka) koji se sastoje od skupa namjernih uzoraka (podnizovi reprezentativnog teksta) i skupa nasumično generiranih uzoraka (nasumično generiranih znakova abecede promatrane domene). Za svaku duljinu uzorka i reprezentativne tekstove srednja vrijednost broja usporedbi znakova, srednja vrijednost vremena izvršavanja (iskazana u milisekundama, *ms*) i srednja vrijednost potrošnje memorije mjere se kao kriterij učinkovitosti algoritma.

Za provedbu vrednovanja novog algoritma za usporedbu se koristi sedam uobičajenih, poznatih algoritama za pretragu znakovnih nizova iz iste grupe. Novi algoritam uspoređen je sa sljedećim algoritmima: BF (*brute force, näive*), BM (Boyer-Moore), KMP (Knuth-Morris-Pratt), AC (Apostolico-Crochemore), QS (Quick Search), MP (Morris-Pratt) i HOR (Horspool) [41], [45], [54], [56], [59], [75], [76], [89], [135], [136]. Gotove implementacije algoritama korištenih u ovom eksperimentu preuzete su iz SMART alata. Sve preuzete gotove implementacije postojećih algoritama za usporedbu su napravljene u C programskom jeziku kao i implementacija razvijenog algoritma. Odabrani algoritmi odgovaraju njihovoj objavljenoj verziji [10], [18].

Nakon odabira algoritama slijedi proces pretraživanja odabranih uzoraka u reprezentativnim tekstovima. Prikupljeni podaci svrstavaju se u grupu primarnih podataka jer se prvi put koriste u ovome istraživanju.

Nakon napravljene pretrage odabranih uzoraka, rezultati pretrage broja usporedbi znakova i vremena izvršavanja za DNA domenu i domenu prirodnog jezika spremljeni su u tekstualne datoteke.

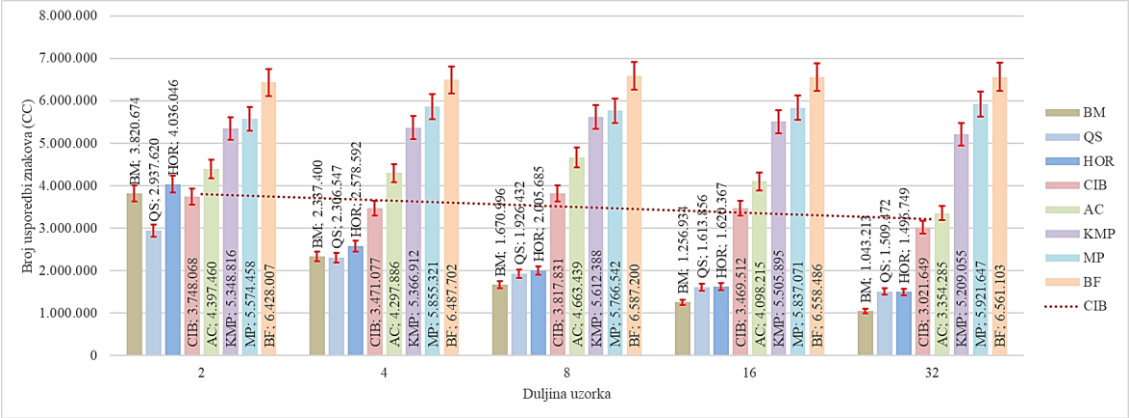
Za svaki uzorak izmjerena je i potrebna količina memorije za izvršavanje pretrage pomoću odabranih algoritama. Inicijalna datoteka s rezultatima usporedbe znakova dopunjena je podacima o potrebnoj količini memorije. Potrebna količina memorije izražena je u bajtovima (engl. *bytes*). Rezultati vrednovanja novog algoritma koji se odnose na potrebnu količinu memorije za izvršavanje algoritma pretrage prikupljeni su pomoću alata Process Monitor proizvođača Microsoft [152].

Nad tako prikupljenim rezultatima napravljena je obrada i analiza. Za analizu i obradu podataka korištena je deskriptivna statistika. Podaci su analizirani pomoću Microsoft Excel alata. Microsoft Excel je program za proračunske tablice koji je razvila kompanija Microsoft. Excel sadrži alate za analizu koji omogućuje kreiranje tablica i generiranje grafikona iz podataka u proračunskim tablicama [148].

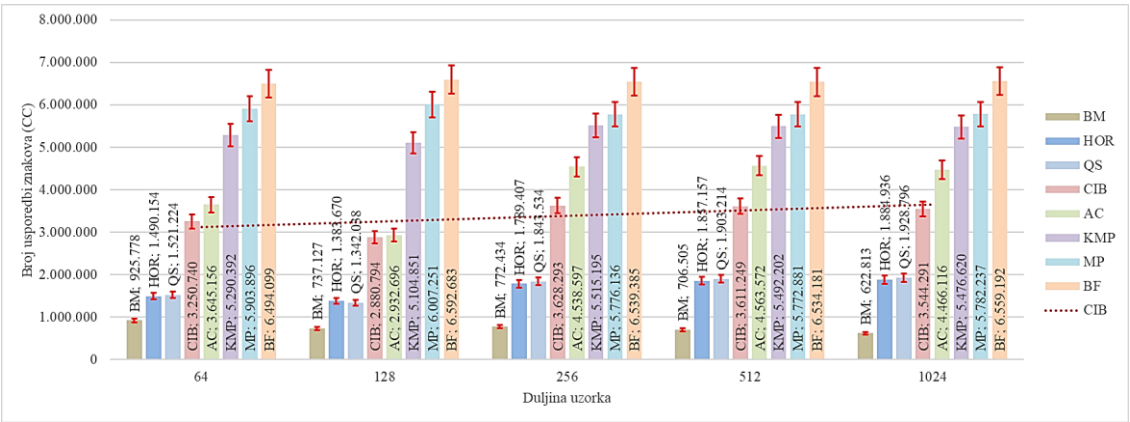
Za srednju vrijednost broja usporedbi znakova, srednju vrijednost vremena izvršavanja i srednju vrijednost potrošnje memorije prikazan je interval pouzdanosti. Interval pouzdanosti predstavlja raspon mogućih vrijednosti unutar kojega se s izvjesnom vjerojatnosti nalazi korištena statistička mjera populacije prema kojem možemo tvrditi da je populacija u određenom postotku između određenog ranga, tj. koliko su rezultati istraživanja bliski stvarnosti, pouzdani, dosljedni i ponovljivi [111]. Drugim riječima, s određenim intervalom pouzdanosti izrada nove studije s uzorcima cijele populacije, rezultati bi bili između dobivenog intervala. Interval pouzdanosti prikazan na rezultatima eksperimentalnog vrednovanja novog algoritma iznosi 95%.

Rezultati eksperimentalnog vrednovanja novog algoritma za duljine uzoraka od 2, 4, 8, 16 i 32 znaka primjenjujući metriku broja usporedbi znakova za DNA domenu prikazani su na slici 4.11. Rezultati eksperimentalnog vrednovanja novog algoritma za duljine uzoraka od 64, 128, 256, 512 i 1024 znaka primjenjujući metriku broja usporedbi znakova za DNA domenu prikazani su na slici 4.12. Rezultati su razdvojeni u dvije slike zbog preglednosti rezultata.

Vertikalna os predstavlja mjerne podatke metrike broja usporedbi, dok su na horizontalnoj osi prikazani algoritmi grupirani po duljini uzorka. Korištena legenda na slikama: BF (*Brute Force, Näive*), BM (Boyer-Moore), KMP (Knuth-Morris-Pratt), AC (Apostolico-Crochemore), QS (Quick Search), MP (Morris-Pratt), HOR (Horspool), CIB (Character Index-Based).



Slika 4.11. Srednja vrijednost broja usporedbi znakova (CC) za uzorke duljine 2, 4, 8, 16 i 32 znaka iz DNA domene

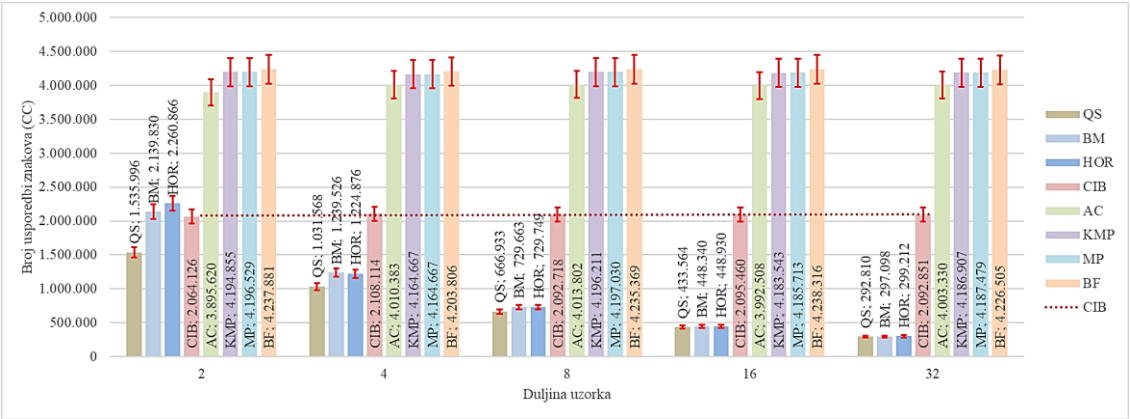


Slika 4.12. Srednja vrijednost broja usporedbi znakova (CC) za uzorke duljine 64, 128, 256, 512 i 1024 znaka iz DNA domene

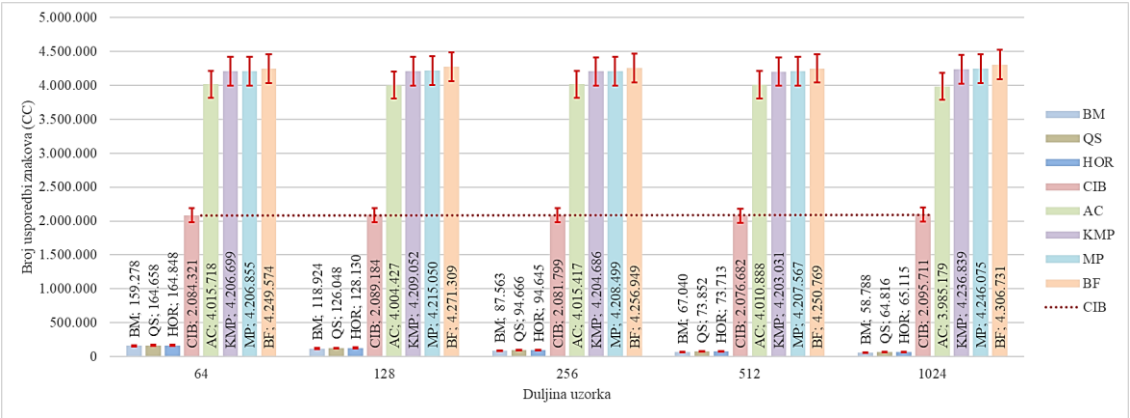
Analizirajući rezultate istraživanja prosječnog broja usporedbi znakova za DNA domenu CIB u pojedinim slučajevima ima manji broj usporedbi znakova od BM i HOR kao i QS koji u nekim slučajevima ima manji broj usporedbi od BM. Za sve ostale duljine uzoraka BM ima najmanji broj usporedbi znakova, a slijede ga QS i HOR. CIB je za sve ostale duljine uzoraka srednje rangiran po broju usporedbi znakova. Za sve duljine uzorka CIB ima manji broj usporedbi znakova od AC, KMP, MP i BF. Promatrajući rezultate prosječnog broja usporedbi znakova CIB algoritma za tekst i uzorke DNA domene može se zaključiti da je CIB algoritam različito efikasan za različite duljine uzorka. Isprekidanom crvenom linijom prikazan je trend vrijednosti rezultata istraživanja CIB algoritma. CIB za uzorke iz DNA domene ima manji prosječni broj usporedbi znakova

za uzorke većih duljina (npr. 32, 64, 128 znakova) nego za uzorke kraćih duljina (npr. 4, 8, 16 znakova).

Rezultati eksperimentalnog vrednovanja novog algoritma za duljine uzoraka od 2, 4, 8, 16 i 32 znaka primjenjujući metriku broja usporedbi znakova za domenu prirodnog jezika prikazani su na slici 4.13. Rezultati eksperimentalnog vrednovanja novog algoritma za duljine uzoraka od 64, 128, 256, 512 i 1024 znaka primjenjujući metriku broja usporedbi znakova za domenu prirodnog jezika prikazani su na slici 4.14. Rezultati su razdvojeni u dvije slike zbog preglednosti rezultata.



Slika 4.13. Srednja vrijednost broja usporedbi znakova (CC) za uzorke duljine 2, 4, 8, 16 i 32 znaka iz domene prirodnog jezika

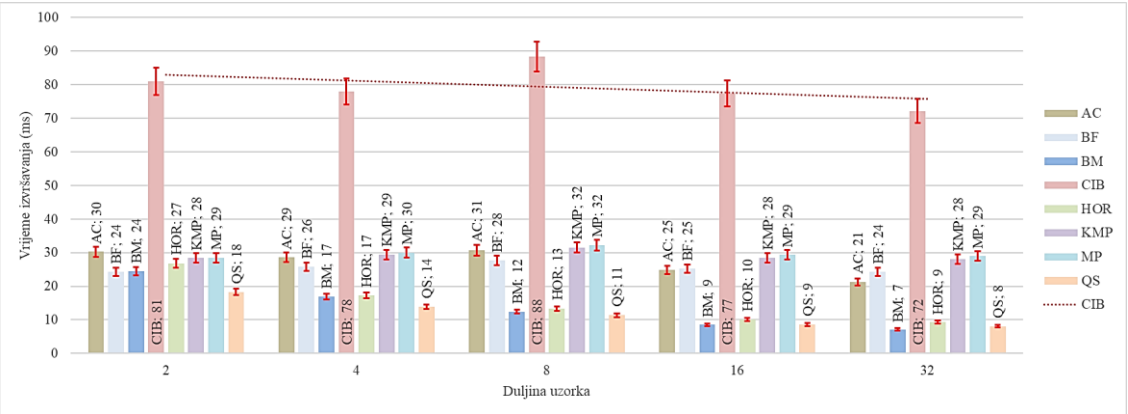


Slika 4.14. Srednja vrijednost broja usporedbi znakova (CC) za uzorke duljine 64, 128, 256, 512 i 1024 znaka iz domene prirodnog jezika

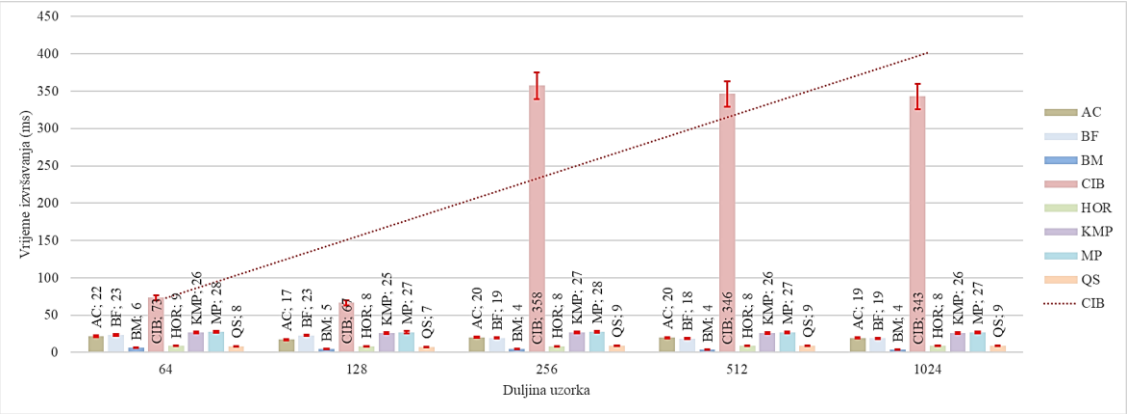
Analizirajući rezultate istraživanja prosječnog broja usporedbi znakova za domenu prirodnog jezika CIB u pojedinim slučajevima ima manji broj usporedbi znakova od BM i HOR. Algoritam QS ima također u pojedinim slučajevima manji broj usporedbi znakova od BM za uzorke duljine. Za sve ostale duljine uzoraka BM ima najmanji broj usporedbi znakova, a slijede ga QS i HOR. Za sve duljine uzorka CIB ima manji broj usporedbi znakova od AC, KMP, MP i BF. Promatrajući rezultate prosječnog broja usporedbi znakova CIB algoritma za tekst i uzorke domene prirodnog jezika može se zaključiti da je CIB algoritam kao i kod DNA domene različito efikasan za različite duljine uzorka. CIB za uzorke iz domene prirodnog jezika ima manji prosječni broj usporedbi znakova za uzorke većih duljina (npr. 64, 128 , 256, 512 znakova) nego za uzorke kraćih duljina (npr. 4, 8, 16, 32 znakova).

Rezultati eksperimentalnog vrednovanja novog algoritma za duljine uzoraka od 2, 4, 8, 16 i 32 znaka primjenjujući metriku vremena izvršavanja algoritma za DNA domenu prikazani su na slici 4.15. Rezultati eksperimentalnog vrednovanja novog algoritma za duljine uzoraka od 64, 128, 256, 512 i 1024 znaka primjenjujući metriku vremena izvršavanja algoritma za DNA domenu prikazani su na slici 4.16. Rezultati su razdvojeni u dvije slike zbog preglednosti rezultata.

Vertikalna os predstavlja mjerne podatke metrike vremena izvršavanja algoritma prikazanim u milisekundama, dok su na horizontalnoj osi prikazani algoritmi grupirani po duljini uzorka.



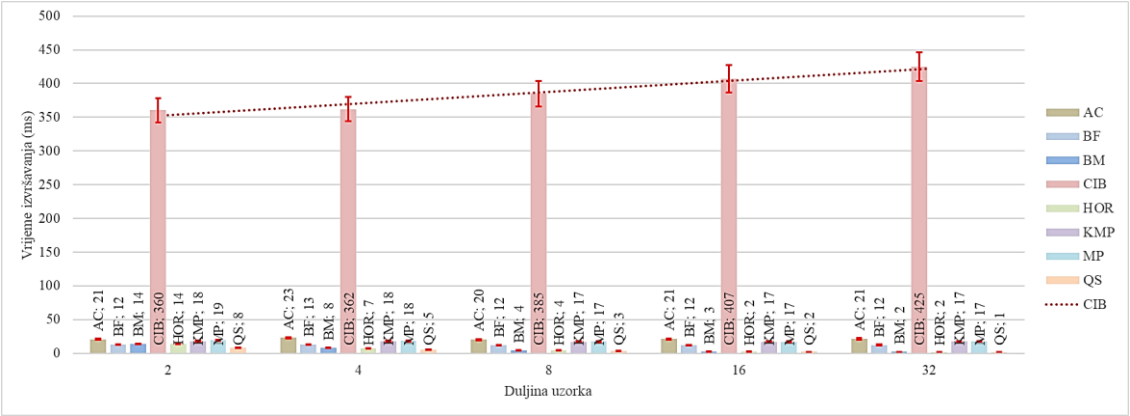
Slika 4.15. Srednja vrijednost vremena izvršavanja algoritma (ms) za uzorke duljine 2, 4, 8, 16 i 32 znaka iz DNA domene



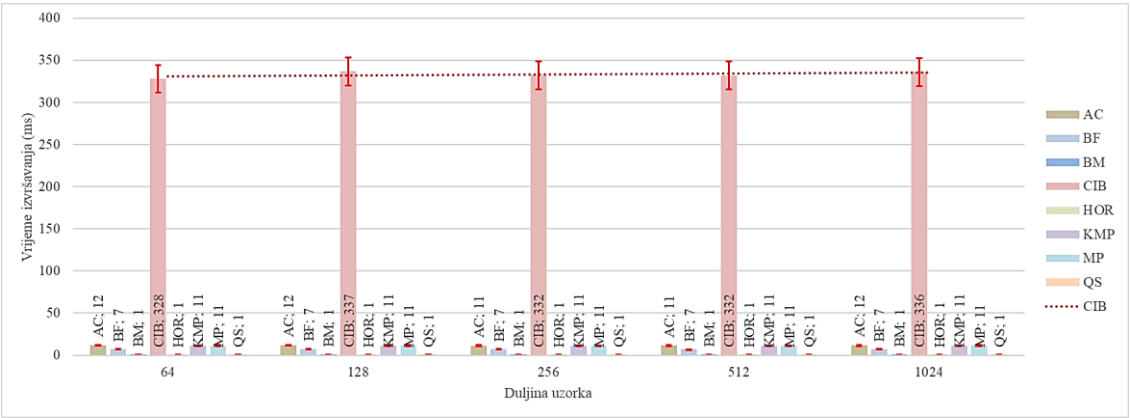
Slika 4.16. Srednja vrijednost vremena izvršavanja algoritma (ms) za uzorke duljine 64, 128, 256, 512 i 1024 znaka iz DNA domene

Analizirajući rezultate istraživanja prosječnog vremena izvršavanja algoritma za DNA domenu BM, QS i HOR za sve duljine uzorka imaju najmanje vrijeme izvršavanja čineći ih time najefikasnijima. Promatrajući rezultate prosječnog vremena izvršavanja CIB algoritma za tekstove i uzorke DNA domene može se zaključiti da je CIB algoritam efikasniji za kraće duljine uzoraka. Za uzorke većih duljina CIB algoritam postaje manje efikasan. Komparativno nekompetitivni rezultati prosječnog vremena izvršavanja CIB algoritma u odnosu na slične algoritme su posljedica neoptimizacije razvijenog koda prema pravilima CIB algoritma. Optimizacijom implementacije CIB algoritma, koja nije realizirana u sklopu dosadašnjeg istraživanja, moguće je postići vremena izvršavanja algoritma komparativna sa sličnim algoritmima.

Rezultati eksperimenta vrednovanja novog algoritma za duljine uzoraka od 2, 4, 8, 16 i 32 znaka primjenjujući metriku vremena izvršavanja algoritma za domenu prirodnog jezika prikazani su na slici 4.17. Rezultati eksperimentalnog vrednovanja novog algoritma za duljine uzoraka od 64, 128, 256, 512 i 1024 znaka primjenjujući metriku vremena izvršavanja algoritma za domenu prirodnog jezika prikazani su na slici 4.18. Rezultati su razdvojeni u dvije slike zbog preglednosti rezultata.



Slika 4.17. Srednja vrijednost vremena izvršavanja algoritma (ms) za uzorke duljine 2, 4, 8, 16 i 32 znaka iz domene prirodnog jezika



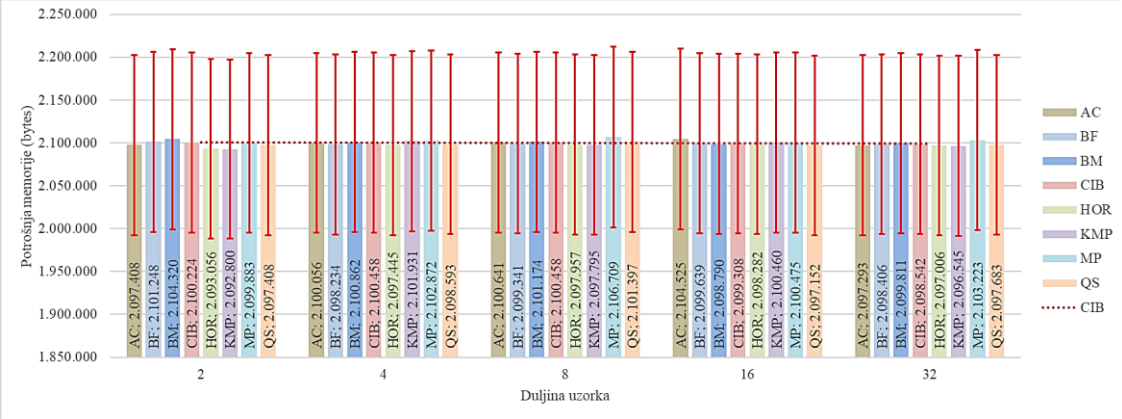
Slika 4.18. Srednja vrijednost vremena izvršavanja algoritma (ms) za uzorke duljine 64, 128, 256, 512 i 1024 znaka iz domene prirodnog jezika

Analizirajući rezultate istraživanja prosječnog vremena izvršavanja algoritma za domenu prirodnog jezika BF ima manje prosječno vrijeme izvršavanja od BM za pojedine duljine uzoraka. QS ima najmanje prosječno vrijeme izvršavanja za skoro sve duljine uzoraka, a slijede ga BM i HOR. Promatrajući rezultate prosječnog vremena izvršavanja CIB algoritma za tekstove i uzorke domene prirodnog jezika može se zaključiti da je CIB algoritam efikasniji za uzorke većih duljina. Za uzorke manjih duljina CIB algoritam postaje manje efikasan. Kao i u primjeru rezultata za DNA domenu, komparativno nekompetitivni rezultati prosječnog vremena izvršavanja CIB algoritma u odnosu na slične algoritme su posljedica neoptimizacije razvijenog koda što se može unaprijediti optimizacijom implementacije CIB algoritma.

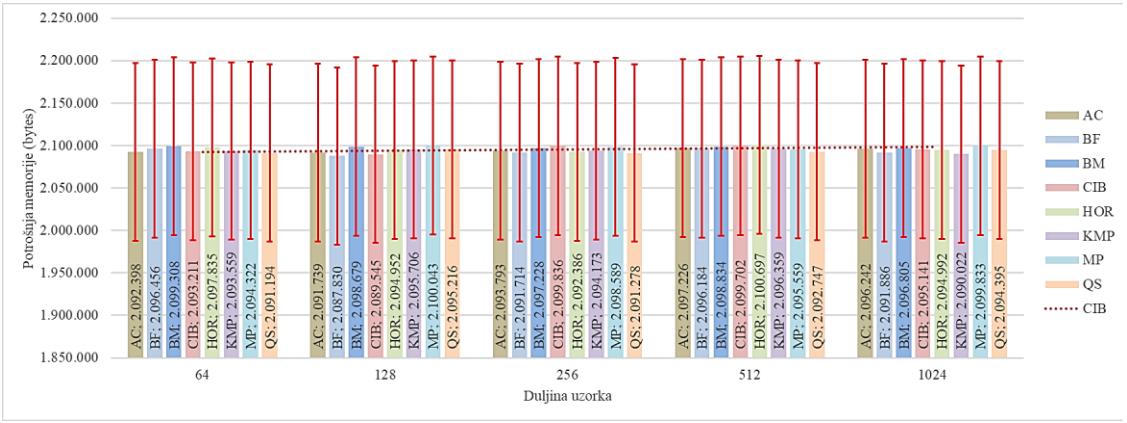
Rezultati eksperimentalnog vrednovanja novog algoritma za duljine uzoraka od 2, 4, 8, 16 i 32 znaka primjenjujući metriku potrošnje memorije za DNA domenu prikazani su na slici 4.19., a rezultati eksperimentalnog vrednovanja novog algoritma za duljine uzoraka od 64, 128, 256, 512 i 1024 znaka primjenjujući metriku potrošnje memorije za DNA domenu prikazani su na slici 4.20. Rezultati su razdvojeni u više slika kako bi se osigurala preglednost rezultata.

Empirijska analiza potrošnje memorije uključuje veličinu uzorka i pomoćni prostora algoritma. Veličina ulaznog teksta (n) nije iskazana u empirijskoj analizi potrošnje memorije.

Vertikalna os predstavlja mjerne podatke metrike potrošnje memorije, dok su na horizontalnoj osi prikazani algoritmi grupirani po duljini uzorka.



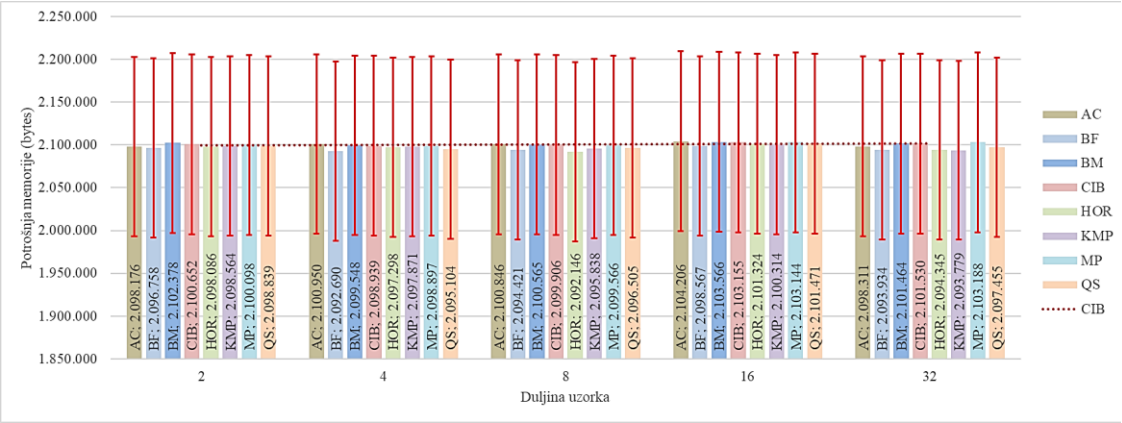
Slika 4.19. Srednja vrijednost potrošnje memorije za uzorke duljine 2, 4, 8, 16 i 32 znaka iz DNA domene



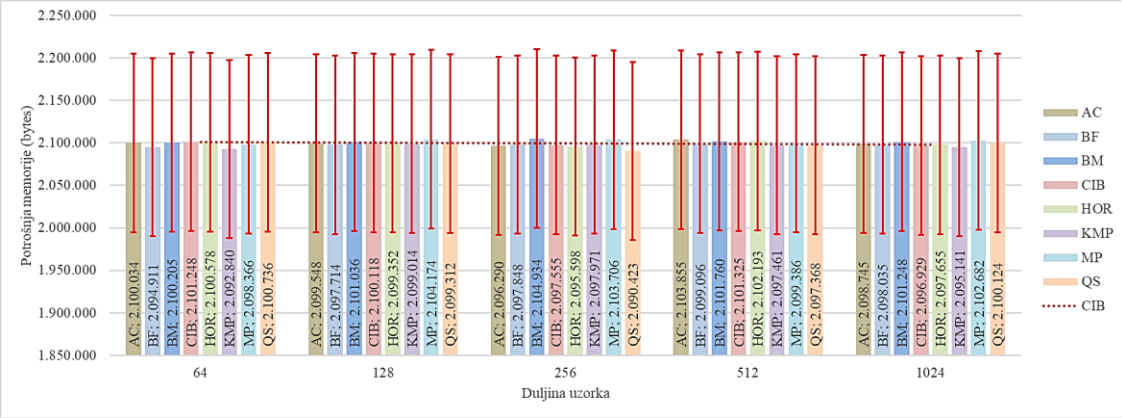
Slika 4.20. Srednja vrijednost potrošnje memorije za uzorke duljine 64, 128, 256, 512 i 1024 znaka iz DNA domene

Analizirajući rezultate istraživanja prosječne potrošnje memorije za DNA domenu CIB ima manju vrijednost prosječne potrošnje memorije od BM za različite duljine uzoraka. BM za pojedine slučajeve duljine uzoraka ima najveću prosječnu vrijednost potrošnje memorije. HOR ima najmanju prosječnu vrijednost prosječne potrošnje memorije za uzorke kraćih duljina, a slijede ga KMP i QS. CIB u pojedinim slučajevima kraćih duljina uzoraka ima manju vrijednost prosječne potrošnje memorije od BM, HOR i QS.

Rezultati eksperimentalnog vrednovanja novog algoritma za duljine uzoraka od 2, 4, 8, 16 i 32 znaka primjenjujući metriku potrošnje memorije za domenu prirodnog jezika prikazani su na slici 4.21. Rezultati eksperimentalnog vrednovanja novog algoritma za duljine uzoraka od 64, 128, 256, 512 i 1024 znaka primjenjujući metriku potrošnje memorije za domenu prirodnog jezika prikazani su na slici 4.22. Rezultati su razdvojeni u dvije slike zbog preglednosti rezultata.



Slika 4.21. Srednja vrijednost potrošnje memorije za uzorke duljine 2, 4, 8, 16 i 32 znaka iz domene prirodnog jezika



Slika 4.22. Srednja vrijednost potrošnje memorije za uzorke duljine 64, 128, 256, 512 i 1024 znaka iz domene prirodnog jezika

Analizirajući rezultate istraživanja prosječne potrošnje memorije za domenu prirodnog jezika BF spada u grupu algoritama koji imaju jednu od najmanjih prosječnih vrijednosti potrošnje memorije, a slijede ga HOR, KMP i QS. CIB u pojedinim slučajevima kraćih duljina uzoraka ima manju vrijednost prosječne potrošnje memorije od BM, HOR i QS.

Analizirajući rezultate istraživanja empirijskih rezultata potrošnje memorije nisu uočena značajnija odstupanja između potrošnje memorije pojedinih algoritama koji su odabrani za provođenje eksperimenta.

Iz rezultata istraživanja prosječne potrošnje memorije CIB algoritma za tekstove i uzorke DNA domene i domene prirodnog jezika može se zaključiti da CIB algoritam ne odstupa značajno od ostalih algoritama. Empirijskom analizom uočeno je kako CIB ima nešto manju vrijednost prosječne potrošnje memorije za uzorke većih duljina dok za pojedine duljine uzoraka ima manju potrošnju memorije od algoritama koji imaju bolje prosječno vrijeme izvršavanja.

Različita efikasnost CIB algoritma koja je prikazana u analizi prosječnog broja usporedbi (CC) znakova za DNA domenu i domenu prirodnog jezika ima veze s različitim vjerojatnostima pojave različitih karaktera. Za DNA domenu vjerojatnost pojave znakova teoretski je ista za sva slova (iako to ovisi o vrsti), a kod domene prirodnih jezika postoje rijetki znakovi koji se često pojavljuju.

Vrijeme izvršavanja ovisi o nizu stvari kao što su programski jezik, kompajler ili čak računalo koji pokreće algoritam. Vremena izvršavanja koja su rezultat napravljene analize CIB algoritma razlog su za potrebu dodatne optimizacije koda CIB algoritma. CIB algoritam koristi jednostavne *if* izraze za primjenu pravila pomaka koja su osnova algoritma, osnovne tipove podatka, višedimenzionalna polja za spremanje specifičnih informacija koje služe za preskakanje nepotrebnih usporedbi kao i manje napredne načine pretrage istih. Prolaženje kroz višedimenzionalna polja putem petlji koja se koriste u CIB algoritmu zahtjeva dodatnu optimizaciju jer je moguće da se nedovoljno velika polja spremaju ili vade iz predmemorije. Moderni procesori rade vrlo brzo, ali imaju ograničenu količinu predmemorije, koja je lako dostupna. Kada ne mogu pronaći ono što im je potrebno u predmemoriji, moraju čitati iz memorije, što je relativno sporo. CIB algoritam u pojedinim slučajevima koristi kompleksnije dizajne obrazaca, a gdje bi jednostavniji dizajni obrazaca bili dovoljni. Koriste se i pojedine funkcije za rješavanje podproblema algoritma koje koriste više parametara nego je potrebno čime se onemogućuje jasna dodjela odgovornosti. Vrijeme izvršenja pojedinih instrukcija CIB algoritma dijelom ovisi i o ponašanju drugih instrukcija na procesoru. Moderni programski jezici koriste mnogo optimizacija kako bi većinu vremena ubrzali većinu instrukcija. Optimizacija CIB algoritma na način da se optimizira broj instrukcija dovesti će do postizanja bržeg vremena izvršavanja. Prikazani rezultati nakon provedenog vrednovanja novog algoritma pokazuju da svojstva uzorka (a vjerojatno i teksta) utječu na učinkovitost algoritma.

Provođenjem eksperimentalnog istraživanja novog CIB algoritma utvrđena je točnost istog jer je u procesu pretrage svaki uzorak pronađen.

5. ZAKLJUČAK

U doktorskoj disertaciji je opisana važnost područja pretrage znakovnih nizova u znanosti i industriji. Proučavanje složenosti, prednosti i ograničenja algoritama zahtjevan je zadatak. Razvoj novih algoritama pretraživanja nizova rezultirao je povećanim interesom za istraživanje područja podudaranja nizova.

Ovaj se rad usredotočuje na algoritme koji koriste egzaktne tehnike podudaranja niza s pristupom usporedbe znakova i bez prethodnog poznavanja strukture teksta. Predložena metodologija rangiranja algoritama temelji se na svojstvima pretraživanog niza i svojstvima tekstova koji se pretražuju. Traženi nizovi klasificiraju se prema entropiji uzorka. Ova metodologija izražava učinkovitost algoritama koristeći metrike neovisne o platformi, tako da ne ovise o implementaciji algoritma, računalnoj arhitekturi ili karakteristikama programskih jezika. Za druge vrste algoritama trenutno nije moguće koristiti razvijenu metodologiju. Metrika usporedbe korištenih znakova neovisna je o platformi u kontekstu formalnih pristupa, ali broj usporedbi izravno utječe na vrijeme potrebno za izvršavanje algoritma i korištenje računalnih resursa.

U radu se detaljno razmatraju dostupni mjerni podaci za procjenu svojstava algoritama pretraživanja nizova i predlaže metodologija za izgradnju modela domene za odabir optimalnog algoritma za pretraživanje nizova. Metodologija se temelji na prikazivanju točnih rezultata podudaranja nizova kako bi se izrazila učinkovitost algoritma bez obzira na duljinu uzorka upita i veličinu skupa podataka. Razmotrili smo broj uspoređenih znakova svakog algoritma izraženih pretraženom entropijom niza za osnovnu analizu.

Pokazani su visoki stupnjevi sličnosti i snažna korelacija između rezultata validacije i podataka izgrađenog modela, što ovu metodologiju čini korisnim alatom koji može pomoći istraživačima da odaberu učinkovit algoritam podudaranja nizova prema potrebama i odaberu prikladno programsko okruženje za razvoj novih algoritama. Sve što je potrebno je uzorak iz određene domene po kojoj je model izgrađen, a model će predložiti korištenje najoptimalnijeg algoritma za korištenje. Definirani model u konačnici pomaže pri odabiru algoritma koji će najvjerojatnije pokrenuti najmanji broj usporedbe znakova u podudaranju uzoraka, a samim time potrošiti najmanje računalnih resursa za izvršavanje.

Ovo istraživanje ne ocjenjuje logiku algoritama ili razvojna programska okruženja. Primarni cilj istraživanja za usporedbu rezultata algoritama je izgradnja modela odabira algoritma.

U radu je predstavljen i novi algoritam za egzaktno podudaranje znakovnih nizova temeljen na usporedbi znakova po indeksima *(CIB – Character Index-Based algorithm)* i koji višestruko iskorištava indekse znakova u uzorku koji se pretražuje.

Razvijeni algoritam uspoređen je s često korištenim algoritmima za egzaktno podudaranje znakovnih nizova na dvije domene tekstova, domeni DNA i domeni prirodnog jezika. Heuristička tehnika prolaska kroz uzorke pomoću neparnih i parnih pozicija u kombinaciji s prethodno obrađenim informacijama formalizirana je pravilima algoritma. Za svaki pomak uzorka kroz tekst, informacije o podudaranju indeksa znakova dobivene tijekom trenutne pozicije uzorka i teksta koriste se za preskakanje nepotrebnih usporedbi znakova.

Analiza memorijskih zahtjeva postojećih algoritama za podudaranje znakovnih nizova tijekom izvođenja zanimljiva je tema. Preskakanje što veće usporedbe znakova ubrzava pretraživanje uzoraka i posljedično smanjuje kapacitet obrade i korištenja memorije, čineći algoritme učinkovitijima.

5.1. Doprinosi disertacije

Na kraju potrebno je naglasiti još jedanput temeljne znanstvene doprinose ove disertacije. Znanstveni doprinos u području pronalaženja tekstualnih podataka prikazan je kroz dvije stavke:

1. Nova metodologija za izgradnju modela koji podržava vrednovanje algoritama za egzaktno podudaranje znakovnih nizova koji koriste pristup usporedbe znakova s online algoritamskim rješenjem pretrage, određene domene tekstova na temelju svojstava uzoraka koji se pretražuju, a koji utječu na učinkovitost algoritama.

i

2. Novi algoritam za podudaranje znakovnih nizova koji koriste pristup usporedbe znakova s online algoritamskim rješenjem pretrage, temeljen na heurističkim pravilima o pozicijama znakova u uzorku koji se pretražuje. Za svaki pomak uzorka naspram teksta koriste se dostupne informacije o odgovarajućim indeksima znakova uzorka za preskakanje nepotrebnih usporedbi znakova.

5.2. Daljnje istraživanje

Tijekom rada na ovoj disertaciji identificirani su različiti novi izazovi i smjernice za proširenje algoritama za podudaranje znakovnih nizova, na primjer, utjecaj svojstava uzorka na učinkovitost algoritma.

Unaprjeđenje preciznosti i razumljivosti odabira najučinkovitijeg algoritma je ključ odgovarajuće primjene odabranog algoritma u procesu pretrage uzorka u znakovnim nizovima. Težnja da rad odabranog algoritma bude optimalan tj. da rezultat pretrage bude najbrži uz minimalno iskorištavanje resursa računala fokus je unaprjeđenja metodologije.

Pretraga uzoraka u znakovnim nizovima može biti iznimno zahtjevna. U nastojanju da pretraga zahtjeva što manje resursa potrebno je iz uzorka pokušati dobiti što je više moguće informacija kako bi pretraga bila jednostavnija. Za potrebe izgradnje novog algoritma definiran je skup pravila kojima je moguće preskočiti nepotrebne usporedbe znakova na osnovu informacija iz predobrade uzorka. Koristeći definirana pravila preskakanja nepotrebnih usporedbi znakova u procesu pretrage uzorka u znakovnim nizovima moguće je ista unaprijediti. Unaprjeđenje pravila preskakanja nepotrebnih usporedbi znakova ima za cilj troškove kapaciteta obrade uzorka i procese usporedbe znakova svesti na najmanju moguću mjeru.

Daljnja istraživanja usmjerena su na pronalaženje dodatnih karakteristika niza, osim entropije uzoraka, koje mogu poboljšati razvijenu preciznost metodologije za odabir učinkovitijih algoritama pretraživanja nizova. Dodatne karakteristike uvelike će pomoći pri odabiru najučinkovitijeg algoritma za pretragu znakovnih nizova.

U nastavku istraživanja cilj je identificirati dodatna heuristička pravila za preskakanje još više znakova usporedbe primjenjujući metode paralelizacije za istovremeno pretraživanje s obje strane uzorka. Osim primjene metoda paralelizacije

moguće je novi algoritam pokretati na raznim vrstama procesora poput grafičkih procesorskih jedinica, ali i distribuiranih računalnih sustava.

Optimizacija algoritma izazovan je zadatak. Optimizacijom koda CIB algoritma na način da se algoritam prilagodi za upotrebu na određenoj platformi u cilju povećanja brzine izvršavanja, smanjenja zauzeća resursa (kao što su radna memorija i procesorska snaga) ima za cilj povećanje efikasnosti.

Iako su postojeći algoritmi, poput BM-a i QS-a, najučinkovitiji, vrijedi nastaviti istraživanje algoritama za egzaktno podudaranje znakovnih nizova u kontekstu poboljšanja performansi istih istražujući svojstva uzoraka nizova.

Literatura

[1] „Global Research and Advisory Company | Gartner“. https://www.gartner.com (pristupljeno 15. siječanj 2020.).

[2] S. Chaudhuri, U. Dayal, i V. Narasayya, „An overview of business intelligence technology“, *Commun. ACM*, sv. 54, izd. 8, str. 88–98, kol. 2011, doi: 10.1145/1978542.1978562.

[3] H. G. Miller i P. Mork, „From data to decisions: A value chain for big data“, *IT Prof.*, sv. 15, izd. 1, str. 57–59, 2013, doi: 10.1109/MITP.2013.11.

[4] C. Ji *i ostali*, „Big data processing: Big challenges“, *J. Interconnect. Networks*, sv. 13, izd. 3–4, tra. 2012, doi: 10.1142/S0219265912500090.

[5] R. C. Joseph i N. A. Johnson, „Big data and transformational government“, *IT Prof.*, sv. 15, izd. 6, str. 43–48, stu. 2013, doi: 10.1109/MITP.2013.61.

[6] K. Kambatla, G. Kollias, V. Kumar, i A. Grama, „Trends in big data analytics“, *J. Parallel Distrib. Comput.*, sv. 74, izd. 7, str. 2561–2573, srp. 2014, doi: 10.1016/j.jpdc.2014.01.003.

[7] J. J. Berman, *Principles of Big Data: Preparing, Sharing, and Analyzing Complex Information*. 2013.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, i C. Stein, *Introduction to algorithms*, Drugo izda. The MIT Press, 2001.

[9] N. C. Jones i P. a Pevzner, *An introduction to Bioinformatics Algorithms*, sv. 101. 2004. doi: 10.1198/jasa.2006.s110.

[10] D. Gusfield, „Algorithms on strings, trees, and sequences: computer science and computational biology“, *Theory and Practice*, sv. 28, izd. 4. str. 554, 1997. doi: 10.1145/270563.571472.

[11] P. Compeau i P. Pevzner, *Bioinformatics Algorithms: An Active Learning Approach*, sv. 1. 2015.

[12] J. Yu, Y. Xue, i J. Li, „Memory Efficient String Matching Algorithm for Network Intrusion Management System“, *Tsinghua Sci. Technol.*, sv. 12, izd. 5, str. 585–593, lis. 2007, doi: 10.1016/S1007-0214(07)70137-2.

[13] S. Vijayarani i R. Janani, „String matching algorithms for reteriving information from desktop – Comparative analysis“, u *Proceedings of the International Conference on Inventive Computation Technologies, ICICT 2016*, 2016, sv. 2016. doi: 10.1109/INVENTIVE.2016.7830233.

[14] S. Faro i T. Lecroq, „The Exact Online String Matching Problem: A Review of the Most Recent Results“, *Acm Comput. Surv.*, sv. 45, izd. 2, str. 13, 2013, doi: Artn 13\rDoi 10.1145/2431211.2431212.

[15] W. M. Szeto i M. H. Wong, „Stream segregation algorithm for pattern matching in polyphonic music databases“, *Multimed. Tools Appl.*, sv. 30, izd. 1, str. 109–127, srp. 2006, doi: 10.1007/S11042-006-0011-9.

[16] S. Zerdoumi *i ostali*, „Image pattern recognition in big data: taxonomy and open challenges: survey“, *Multimed. Tools Appl. 2017 778*, sv. 77, izd. 8, str. 10091–10121, 2017, doi: 10.1007/S11042-017-5045-7.

[17] C. Pizzi, M. Ornamenti, S. Spangaro, S. E. Rombo, i L. Parida, „Efficient algorithms for sequence analysis with entropic profiles“, *IEEE/ACM Trans. Comput. Biol. Bioinforma.*, sv. 15, izd. 1, str. 117–128, 2018, doi: 10.1109/TCBB.2016.2620143.

[18] S. Faro, T. Lecroq, i S. Borz, „The String Matching Algorithms Research Tool“, *Proc. Prague Stringology Conf.*, str. 99–113, 2016.

[19] K. Al-Khamaiseh i S. Alshagarin, „A Survey of String Matching Algorithms“, *J. Eng. Res. Appl. www.ijera.com ISSN*, sv. 4, izd. 2, str. 2248–9622144, 2014.

[20] N. Singla i D. Garg, „String Matching Algorithms and their Applicability in various Applications“, *Int. J. Soft Comput. Eng.*, izd. 6, str. 2231–2307, 2012.

[21] M. Góngora-Blandón i M. Vargas-Lombardo, „State of the Art for String Analysis and Pattern Search Using CPU and GPU Based Programming“, *J. Inf. Secur.*, sv. 03, izd. 04, str. 314–318, 2012, doi: 10.4236/jis.2012.34038.

[22] R. Sedgewick i P. Flajolet, *An introduction to the analysis of algorithms*, Drugo izda. Addison-Wesley/Pearson Education, 2013.

[23] S. S. Skiena, *The algorithm design manual*, Drugo izda. Springer, 2008.

[24] S. Shrivastav, S. Kumar, i K. Kumar, „Towards an ontology based framework for searching multimedia contents on the web“, *Multimed. Tools Appl.*, sv. 76, izd. 18, str. 18657–18686, ruj. 2017, doi: 10.1007/s11042-017-4350-5.

[25] Y. H. Kim, H. J. Kwon, J. G. Kang, i H. Chang, „The study on content based multimedia data retrieval system“, *Multimed. Tools Appl.*, sv. 57, izd. 2, str. 393–405, ožu. 2012, doi: 10.1007/s11042-011-0758-5.

[26] V. SaiKrishna, P. A. Rasool, i D. N. Khare, „String Matching and its Application in Diversified Fields“, *IJCSI Int. J. Comput. Sci. Issues*, sv. 9, izd. 1, str. 219–226, 2012.

[27] A. A. AbdulRazzaq, N. A. Rashid, A. A. Hasan, i M. A. Abu-Hashem, „The exact string matching algorithms efficiency review“, *Glob. J. Technol.*, sv. 4, izd. 2, 2013.

[28] L. Colussi, „Correctness and efficiency of pattern matching algorithms“, *Inf. Comput.*, sv. 95, izd. 2, str. 225–251, pros. 1991, doi: 10.1016/0890-5401(91)90046-5.

[29] P. Kalsi, H. Peltola, i J. Tarhio, „Comparison of exact string matching algorithms for biological sequences“, *Commun. Comput. Inf. Sci.*, sv. 13, str. 417–426, 2008, doi: 10.1007/978-3-540-70600-7_31.

[30] A. Clements, *Principles of computer hardware*, Fourth Edi. Oxford University Press, 2006.

[31] K. M. Alhendawi i A. S. Baharudin, „String Matching Algoritms (SMAs): Survey & Empirical Analysis“, *J. Comput. Sci. Manag.*, sv. 2, izd. 5, str. 2637–2644, 2013.

[32] E. J. Smith, „String pattern matching algorithms: An empirical analysis“, The University of Montana, 1991.

[33] P. Pandiselvam, T. Marimuthu, i R. Lawrance, „A comparative study on string matching algorithms of biological sequences“, *Int. Conf. Intell. Comput.*, str. 1–5, 2014.

[34] S. Azzam, A. Wesam, H. B. Aladdin, i H. A. Adel, „An Efficient Pattern Matching Algorithm“, *J. Appl. Sci.*, sv. 7, izd. 18, str. 2691–2695, pros. 2007, doi: 10.3923/jas.2007.2691.2695.

[35] N. Tuck, T. Sherwood, B. Calder, i G. Varghese, „Deterministic memory-efficient string matching algorithms for intrusion detection“, u *Proceedings - IEEE INFOCOM*, 2004, sv. 4, str. 2628–2639. doi: 10.1109/INFCOM.2004.1354682.

[36] Q. X. Yang, S. S. Yuan, L. Zhao, L. Chun, i S. Peng, „Faster algorithm of string comparison“, *Pattern Anal. Appl.*, sv. 6, izd. 2, str. 122–133, lip. 2003, doi: 10.1007/s10044-002-0180-8.

[37] N. S. Gill i P. S. Grover, „Component-based measurement“, *ACM SIGSOFT Softw. Eng. Notes*, sv. 28, izd. 6, str. 4–4, stu. 2003, doi: 10.1145/966221.966237.

[38] M. Goulão, M. Goulão, i O. B. E. Abreu, „Software Components Evaluation: an Overview“, *Proc. 5ª Conferência da APSI*, 2004.

[39] A. L. Baroni i F. B. Abreu, „A formal library for aiding metrics extraction“, 2003.

[40] M. Goulão i F. Brito e Abreu, „Formal definition of metrics upon the CORBA component model“, *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, sv. 3712 LNCS, str. 88–105, 2005, doi: 10.1007/11558569_8.

[41] S. I. Hakak, A. Kamsin, P. Shivakumara, G. A. Gilkar, W. Z. Khan, i M. Imran, „Exact String Matching Algorithms: Survey, Issues, and Future Research Directions“, *IEEE Access*, sv. 7, str. 69614–69637, 2019, doi: 10.1109/ACCESS.2019.2914071.

[42] J. A. Joseph, R. Korah, i S. Salivahanan, „Efficient string matching FPGA for speed up network Intrusion detection“, *Appl. Math. Inf. Sci.*, sv. 12, izd. 2, str. 397–404, ožu. 2018, doi: 10.18576/amis/120214.

[43] S. Arudchutha, T. Nishanthy, i R. G. Ragel, „String matching with multicore CPUs: Performing better with the Aho-Corasick algorithm“, *2013 IEEE 8th Int. Conf. Ind. Inf. Syst. ICIIS 2013 - Conf. Proc.*, str. 231–236, 2013, doi: 10.1109/ICIINFS.2013.6731987.

[44] X. Wang i D. Pao, „Memory-based architecture for multicharacter Aho-Corasick string matching“, *IEEE Trans. Very Large Scale Integr. Syst.*, sv. 26, izd. 1, str. 143–154, sij. 2018, doi: 10.1109/TVLSI.2017.2753843.

[45] R. S. Boyer i J. S. Moore, „A fast string searching algorithm“, *Commun. ACM*, sv. 20, izd. 10, str. 762–772, lis. 1977, doi: 10.1145/359842.359859.

[46] A. A. Karcioglu i H. Bulut, „Improving hash-q exact string matching algorithm with perfect hashing for DNA sequences“, *Comput. Biol. Med.*, sv. 131, str. 104292, tra. 2021, doi: 10.1016/j.compbiomed.2021.104292.

[47] W. Yang, „Mealy machines are a better model of lexical analyzers“, *Comput. Lang.*, sv. 22, izd. 1, str. 27–38, tra. 1996, doi: 10.1016/0096-0551(96)00003-3.

[48] B. Melichar, J. Holub, i J. Polcar, *Text searching algorithms*, sv. I, izd. March. 2005.

[49] G. Navarro i M. Raffinot, „Flexible Pattern Matching in Strings: Practical Online Search Algorithms for Texts and Biological Sequences“, *Computer*, sv. 35, izd. 9. 2002. doi: 10.1109/MC.2002.1033033.

[50] N. Jiji i T. Mahalakshmi, „Survey of Exact String Matching Algorithm for Detecting Patterns in Protein Sequence“, *Adv. Comput. Sci. Technol.*, sv. 10, izd. 8, str. 2707–2720, 2017.

[51] P. D. Michailidis i K. G. Margaritis, „On-line string matching algorithms: survey and experimental results“, *Int. J. Comput. Math.*, sv. 76, izd. 4, str. 411–434, sij. 2001, doi: 10.1080/00207160108805036.

[52] M. Ghodsi, „Approximate String Matching using Backtracking over Suffix Arrays“, *Comput. Sci. Dep. Univ. Maryl.*, 2009.

[53] A. V. Aho, „Algorithms for Finding Patterns in Strings“, *Algorithms Complex.*, str. 255–300, sij. 1990, doi: 10.1016/B978-0-444-88071-0.50010-2.

[54] V. R. Knuth, D.E., Morris, J.H., & Pratt, J. H. Morris, Jr., i V. R. Pratt, „Fast Pattern Matching in Strings“, *SIAM J. Comput.*, sv. 6, izd. 2, str. 323–350, 1977, doi: 10.1137/0206024.

[55] M. Hernández, „A taxonomy of some right-to-left string-matching algorithms“, *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, sv. 5979 LNCS, str. 79–95, 2010, doi: 10.1007/978-3-642-11999-6_6.

[56] D. M. Sunday, „A very fast substring search algorithm“, *Commun. ACM*, sv. 33, izd. 8, str. 132–142, kol. 1990, doi: 10.1145/79173.79184.

[57] T. Berry i S. Ravindran, „A Fast String Matching Algorithm and Experimental Results.“, 1999.

[58] M. Crochemore *i ostali*, „Speeding up two string-matching algorithms“, *Algorithmica*, sv. 12, izd. 4–5, str. 247–267, stu. 1994, doi: 10.1007/BF01185427.

[59] R. N. Horspool, „Practical fast searching in strings“, *Softw. Pract. Exp.*, sv. 10, izd. 6, str. 501–506, lip. 1980, doi: 10.1002/spe.4380100608.

[60] R. Baeza-Yates i G. H. Gonnet, „A new approach to text searching“, *Commun. ACM*, sv. 35, izd. 10, str. 74–82, lis. 1992, doi: 10.1145/135239.135243.

[61] R. M. Karp i M. O. Rabin, „Efficient randomized pattern-matching algorithms“, *IBM J. Res. Dev.*, sv. 31, izd. 2, str. 249–260, 1987, doi: 10.1147/RD.312.0249.

[62] M. C. Harrison, „Implementation of the substring test by hashing“, *Commun. ACM*, sv. 14, izd. 12, str. 777–779, pros. 1971, doi: 10.1145/362919.362934.

[63] A. V. Aho i M. J. Corasick, „Efficient string matching: an aid to bibliographic search“, *Commun. ACM*, sv. 18, izd. 6, str. 333–340, lip. 1975, doi: 10.1145/360825.360855.

[64] B. Commentz-Walter, „A string matching algorithm fast on the average“, 1979, str. 118–132. doi: 10.1007/3-540-09510-1_10.

[65] D. Cantone i S. Faro, „Fast-Search: A new efficient variant of the Boyer-Moore string matching algorithm“, u *International Workshop on Experimental and Efficient Algorithms*, 2003, str. 47–58.

[66] D. Cantone i S. Faro, „Forward-Fast-Search: Another Fast Variant of the Boyer-

Moore String Matching Algorithm.“, u *Stringology*, 2003, str. 10–24.

[67] M. Crochemore i T. Lecroq, „A fast implementation of the Boyer-Moore string matching algorithm“, *Submitt. Publ.*, 2008.

[68] M. E. Nebel, „Fast string matching by using probabilities: On an optimal mismatch variant of Horspool’s algorithm“, *Theor. Comput. Sci.*, sv. 359, izd. 1–3, str. 329–343, 2006.

[69] F. Franek, C. G. Jennings, i W. F. Smyth, „A simple fast hybrid pattern-matching algorithm“, *J. Discret. Algorithms*, sv. 5, izd. 4, str. 682–695, pros. 2007, doi: 10.1016/J.JDA.2006.11.004.

[70] P. D. Michailidis i K. G. Margaritis, „On-Line Approximate String Searching Algorithms: Survey and Experimental Results“, *Int. J. Comput. Math.*, sv. 79, izd. 8, str. 867–888, 2002, doi: 10.1080/00207160212111.

[71] G. M. Landau i U. Vishkin, „Efficient string matching with k mismatches“, *Theor. Comput. Sci.*, sv. 43, str. 239–249, 1986, doi: 10.1016/0304-3975(86)90178-7.

[72] G. Davies i S. Bowsher, „Algorithms for pattern matching“, *Softw. Pract. Exp.*, sv. 16, izd. 6, str. 575–601, lip. 1986, doi: 10.1002/spe.4380160608.

[73] Z. Galil i R. Giancarlo, „Data structures and algorithms for approximate string matching“, *J. Complex.*, sv. 4, izd. 1, str. 33–72, ožu. 1988, doi: 10.1016/0885-064X(88)90008-8.

[74] P. Jokinen, J. Tarhio, i E. Ukkonen, „A Comparison of Approximate String Matching Algorithms“, *Softw. Pract. Exp.*, sv. 26, izd. 12, str. 1439–1458, pros. 1996, doi: 10.1002/(SICI)1097-024X(199612)26:12<1439::AID-SPE71>3.0.CO;2-1.

[75] S. Faro, „Evaluation and improvement of fast algorithms for exact matching on genome sequences“, *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, sv. 9702, str. 145–157, 2016, doi: 10.1007/978-3-319-38827-4_12.

[76] A. Hume i D. Sunday, „Fast string searching“, *Softw. Pract. Exp.*, sv. 21, izd. 11, str. 1221–1248, 1991, doi: 10.1002/spe.4380211105.

[77] A. Tkalac Verčič, D. Sinčić, i N. Pološki Vokić, *Priručnik za metodologiju istraživačkog rada u društvenim djelatnostima*. 2011.

[78] G. Dodig Crnković, „Scientific Methods in Computer Science“, *Department of Computer Science Mälardalen University*.

[79] D. Taboršak, „Metodologija izrade Znanstveno-Istraživačkog rada“.

[80] „Anabarilius grahami isolate AG-KIZ scaffold371_cov124, whole genome sh - Nucleotide - NCBI“. https://www.ncbi.nlm.nih.gov/nuccore/RJVU01051648.1 (pristupljeno 05. ožujak 2019.).

[81] „Chelonia mydas unplaced genomic scaffold, CheMyd_1.0 scaffold1, whole - Nucleotide - NCBI“. https://www.ncbi.nlm.nih.gov/nuccore/NW_006571126.1 (pristupljeno 22. lipanj 2020.).

[82] „Escherichia coli strain LM33 isolate patient, whole genome shotgun seq - Nucleotide - NCBI“. https://www.ncbi.nlm.nih.gov/nuccore/NZ_LN874954.1 (pristupljeno 22. lipanj 2020.).

[83] „Macaca mulatta isolate AG07107 chromosome 19 genomic scaffold ScNM3vo_ - Nucleotide - NCBI“. https://www.ncbi.nlm.nih.gov/nuccore/ML143108.1 (pristupljeno 22. lipanj 2020.).

[84] „The Canterbury Corpus - The King James version of the bible“. https://corpus.canterbury.ac.nz/descriptions/large/bible.html (pristupljeno 19. kolovoz 2020.).

[85] „Orange – Data Mining Fruitful & Fun“. https://orange.biolab.si/

[86] S. Wu i U. Manber, „Fast text searching: allowing errors“, *Commun. Acm*, sv. 35, izd. 10, str. 83–91, 1992, doi: 10.1145/135239.135244.

[87] G. A. Stephen, *String Searching Algorithms*, sv. 3. WORLD SCIENTIFIC, 1994. doi: 10.1142/2418.

[88] J. H. Morris (Jr) i V. R. Pratt, „A linear pattern-matching algorithm“, 1970.

[89] T. Lecroq i C. Charras, *Handbook od Exact String Matching*. Rouen: Laboratoire d'Informatique de Rouen Université de Rouen, Faculté des Sciences et des Techniques 76821 Mont-Saint-Aignan Cedex, France, 2001.

[90] Ben Langmead, „Teaching Materials“, *Department of Computer Science*. http://www.langmead-lab.org/teaching-materials/

[91] G. Navarro i M. Raffinot, „Fast and flexible string matching by combining bit-parallelism and suffix automata“, *ACM J. Exp. Algorithmics*, sv. 5, str. 4, pros. 2000, doi: 10.1145/351827.384246.

[92] C. Allauzen, M. Crochemore, i M. Raffinot, „Efficient Experimental String Matching by Weak Factor Recognition“, u *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, sv. 2089, Springer, Berlin, Heidelberg, 2001, str. 51–72. doi: 10.1007/3-540-48194-X_5.

[93] D. Zindros, „A Gentle Introduction to Algorithm Complexity Analysis“, *Analysis*, 2012. http://discrete.gr/complexity/

[94] R. Sedgewick, *Algorithms in C*, Treće izda. Addison-Wesley, 2002.

[95] P. Jain i S. Pandey, „Comparative Study on Text Pattern Matching for Heterogeneous System“, *Int. J. Comput. Sci. Eng. Technol.*, sv. 3, izd. 11, str. 537–543, 2012.

[96] T. M. Cover i J. A. Thomas, *Elements of Information Theory*. John Wiley and Sons, 2005. doi: 10.1002/047174882X.

[97] G. Barabucci, P. Ciancarini, A. Di Iorio, i F. Vitali, „Measuring the quality of diff algorithms: A formalization“, *Comput. Stand. Interfaces*, sv. 46, str. 52–65, 2016, doi: 10.1016/j.csi.2015.12.005.

[98] N. Ivkovic, D. Jakobovic, i M. Golub, „Measuring Performance of Optimization Algorithms in Evolutionary Computation“, *Int. J. Mach. Learn. Comput.*, sv. 6, izd. 3, str. 167–171, 2016, doi: 10.18178/ijmlc.2016.6.3.593.

[99] A. V. Aho, J. E. Hopcroft, i J. D. Ullman, *The design and analysis of computer algorithms*. Addison-Wesley Pub. Co, 1974.

[100] J. Hromkovič, *Theoretical computer science: introduction to Automata, computability, complexity, algorithmics, randomization, communication, and*

cryptography. Springer-Verlag Berlin Heidelberg, 2004.

[101] D. Ron, „Guest Editor's Introduction“, *Mach. Learn. 1998 301*, sv. 30, izd. 1, str. 5–6, 1998, doi: 10.1023/A:1007411609915.

[102] S. Petti i S. R. Eddy, „Constructing benchmark test sets for biological sequence analysis using independent set algorithms“, *PLOS Comput. Biol.*, sv. 18, izd. 3, str. e1009492, ožu. 2022, doi: 10.1371/JOURNAL.PCBI.1009492.

[103] L. Prechelt, „Early Stopping — But When?“, *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, sv. 7700 LECTU, str. 53–67, 2012, doi: 10.1007/978-3-642-35289-8_5.

[104] G. James, D. Witten, T. Hastie, i R. Tibshirani, „An Introduction to Statistical Learning“, New York, NY: Springer US, 2021. doi: 10.1007/978-1-0716-1418-1.

[105] „GenBank“. https://www.ncbi.nlm.nih.gov/genbank/ (pristupljeno 16. veljača 2020.).

[106] M. I. Khalil, „Locating All Common Subsequences in Two DNA Sequences“, *Int. J. Inf. Technol. Comput. Sci.*, sv. 8, izd. 5, str. 81–87, 2016, doi: 10.5815/ijitcs.2016.05.09.

[107] T. Winograd, „Understanding natural language“, *Cogn. Psychol.*, sv. 3, izd. 1, str. 1–191, sij. 1972, doi: 10.1016/0010-0285(72)90002-3.

[108] J. Bartlett, J. Kotrlik, i C. Higgins, „Organizational research: Determining appropriate sample size in survey research“, *Inf. Technol. Learn. Perform. J.*, sv. 19, izd. 1, 2001.

[109] H. Taherdoost, „Determining Sample Size; How to Calculate Survey Sample Size“, *Int. J. Econ. Manag. Syst.*, sv. 2, 2017.

[110] G. D. Israel, „Determining Sample Size“, 1992.

[111] G. J. Myatt i W. P. Johnson, *Making sense of data I a practical guide to exploratory data analysis and data mining*, Drugo izda. New Jersey: John Wiley & Sons, Inc., 2014.

[112] L. Cohen, L. Manion, i K. Morrison, „Research Methods in Education“, *Res. Methods Educ.*, ožu. 2013, doi: 10.4324/9780203720967/RESEARCH-METHODS-EDUCATION-LOUIS-COHEN-LAWRENCE-MANION-KEITH-MORRISON.

[113] „termodinamika | Hrvatska enciklopedija“. https://www.enciklopedija.hr/ (pristupljeno 15. veljača 2021.).

[114] M. Knežević, „Pojam entropije i njegovo značenje u teoriji i praksi socijalnog rada“, *Ljetop. Soc. rada*, str. 111–123, 1996.

[115] I. Pavlić, *Statistička teorija i primjena*. Zagreb: Tehnička knjiga, 1971.

[116] F. Provost i T. Fawcett, *Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking*, 1st editio. O'Reilly Media, 2013.

[117] A. Wehrl, „General properties of entropy“, *Rev. Mod. Phys.*, sv. 50, izd. 2, str. 221, tra. 1978, doi: 10.1103/RevModPhys.50.221.

[118] W. Ebeling, „Entropy, predictability and historicity of nonlinear processes“, str. 217–228, ožu. 1993, doi: 10.1142/9789814503648_0021.

[119] C. E. Shannon, „A Mathematical Theory of Communication“, *Bell Syst. Tech. J.*, sv. 27, izd. 3, str. 379–423, srp. 1948, doi: 10.1002/j.1538-7305.1948.tb01338.x.

[120] R. Mohammed, „Information Analysis of DNA Sequences“, *arXiv*, sv. cs.CE, str. 1–22, 2010.

[121] A. O. Schmitt i H. Herzel, „Estimating the entropy of DNA sequences“, *J. Theor. Biol.*, sv. 188, izd. 3, str. 369–377, 1997, doi: 10.1006/jtbi.1997.0493.

[122] W. Ebeling i G. Nicolis, „Word frequency and entropy of symbolic sequences: a dynamical perspective“, *Chaos, Solitons and Fractals*, sv. 2, izd. 6, str. 635–650, 1992, doi: 10.1016/0960-0779(92)90058-U.

[123] H. Herzel, W. Ebeling, i A. O. Schmitt, „Entropies of biosequences: The role of repeats“, *Phys. Rev. E*, sv. 50, izd. 6, str. 5061–5071, pros. 1994, doi: 10.1103/PhysRevE.50.5061.

[124] A. Lesne, J. L. Blanc, i L. Pezard, „Entropy estimation of very short symbolic sequences“, *Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys.*, sv. 79, izd. 4, str. 1–10, 2009, doi: 10.1103/PhysRevE.79.046208.

[125] A. Muchnik i N. Vereshchagin, „Shannon entropy vs. kolmogorov complexity“, *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, sv. 3967 LNCS, str. 281–291, 2006, doi: 10.1007/11753728_29.

[126] P. Grunwald i P. Vitanyi, „Shannon Information and Kolmogorov Complexity“, lis. 2004.

[127] A. Teixeira, A. Matos, A. Souto, i L. Antunes, „Entropy Measures vs. Kolmogorov Complexity“, *Entropy*, sv. 13, str. 595–611, 2011, doi: 10.3390/e13030595.

[128] S. Vajapeyam, „Understanding Shannon’s entropy metric for information“, *arXiv:1405.2061*, 2014.

[129] R. W. (Richard W. Hamming, *Coding and information theory*, Drugo izda. Prentice Hall, 1986.

[130] P. C. Rhodes i G. R. Garside, „Use of maximum entropy method as a methodology for probabilistic reasoning“, *Knowledge-Based Syst.*, sv. 8, izd. 5, str. 249–258, lis. 1995, doi: 10.1016/0950-7051(95)98902-I.

[131] I. Markić, M. Štula, M. Zorić, i D. Stipaničev, „Entropy-Based Approach in Selection Exact String-Matching Algorithms“, *Entropy*, sv. 23, izd. 1, str. 31, pros. 2020, doi: 10.3390/e23010031.

[132] I. Markić, M. Štula, i M. Jukić, *Pattern Searching in Genome*, sv. 10, izd. 1. International Journal of Advancements in Computing Technology, 2018.

[133] I. Markić, M. Štula, i M. Zorić, „String pattern searching algorithm based on characters indices“, u *Proceedings of 4th International Conference on Smart and Sustainable Technologies (SpliTech)*, 2019, str. 100–103.

[134] „The Canterbury Corpus“. http://corpus.canterbury.ac.nz/

[135] A. Apostolico i M. Crochemore, „Optimal canonization of all substrings of a string“, *Inf. Comput.*, sv. 95, izd. 1, str. 76–95, stu. 1991, doi: 10.1016/0890-5401(91)90016-U.

[136] S. Hakak, A. Kamsin, P. Shivakumara, M. Y. Idna Idris, i G. A. Gilkar, „A new split based searching for exact pattern matching for natural texts“, *PLoS One*, sv.

13, izd. 7, str. e0200912, srp. 2018, doi: 10.1371/journal.pone.0200912.

[137] S. Manikandan, „Frequency distribution“, *J. Pharmacol. Pharmacother.*, sv. 2, izd. 1, str. 54, 2011, doi: 10.4103/0976-500x.77120.

[138] S. M. Dowdy, S. Wearden, i D. M. Chilko, „Statistics for research.“, str. 627, 2004.

[139] F. Croxton, *Applied general statistics*, 3. ed. London: I. Pitnam and Sons, 1968.

[140] „National Center for Biotechnology Information“. https://www.ncbi.nlm.nih.gov/ (pristupljeno 22. lipanj 2020.).

[141] P. Barrett, „Euclidean Distance raw, normalized, and double-scaled coefficients“, 2005. https://www.pbarrett.net/techpapers/euclid.pdf

[142] C. Wheelan, *Naked Statistics: Stripping the Dread from the Data*. New York, United States: WW Norton & Co, 2013.

[143] H. Anton, *Elementary Linear Algebra*, 11th Editi. Wiley, 2019.

[144] J. Lee Rodgers, W. Alan Nice Wander, J. L. Rodgers, i ; W Alan Nicewander, „Thirteen Ways to Look at the Correlation Coefficient“, *Am. Stat.*, sv. 42, izd. 1, str. 59–66, 2012, doi: 10.1080/00031305.1988.10475524.

[145] J. Benesty, J. Chen, Y. Huang, i I. Cohen, „Pearson Correlation Coefficient“, *Springer Top. Signal Process.*, sv. 2, str. 1–4, 2009, doi: 10.1007/978-3-642-00296-0_5.

[146] D. P. Francis, A. J. S. Coats, i D. G. Gibson, „How high can a correlation coefficient be? Effects of limited reproducibility of common cardiological measures“, *Int. J. Cardiol.*, sv. 69, izd. 2, str. 185–189, svi. 1999, doi: 10.1016/S0167-5273(99)00028-5.

[147] A. Shah, „Correlation Vs. Regression: A Review“, *Int. J. Soc. Impact*, 2020, doi: 10.25215/2455/0502015.

[148] „Microsoft Docs“. https://docs.microsoft.com./en-us/ (pristupljeno 04. siječanj 2021.).

[149] H. P. Kriegel, E. Schubert, i A. Zimek, „The (black) art of runtime evaluation: Are we comparing algorithms or implementations?“, *Knowl. Inf. Syst.*, sv. 52, izd. 2, str. 341–378, kol. 2017, doi: 10.1007/s10115-016-1004-2.

[150] D. M. W. Powers, „Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness and Correlation“, *Hum. Commun. Sci. SummerFest*, izd. December, str. 24, 2007.

[151] C. Goutte i E. Gaussier, „A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation“, *Lect. Notes Comput. Sci.*, sv. 3408, str. 345–359, 2005, doi: 10.1007/978-3-540-31865-1_25/COVER.

[152] Microsoft, „Sysinternals - Windows Sysinternals“. https://docs.microsoft.com/en-us/sysinternals/

[153] M. Kiš, *Školski informatički rječnik*. Naklada Ljevak, 2000.

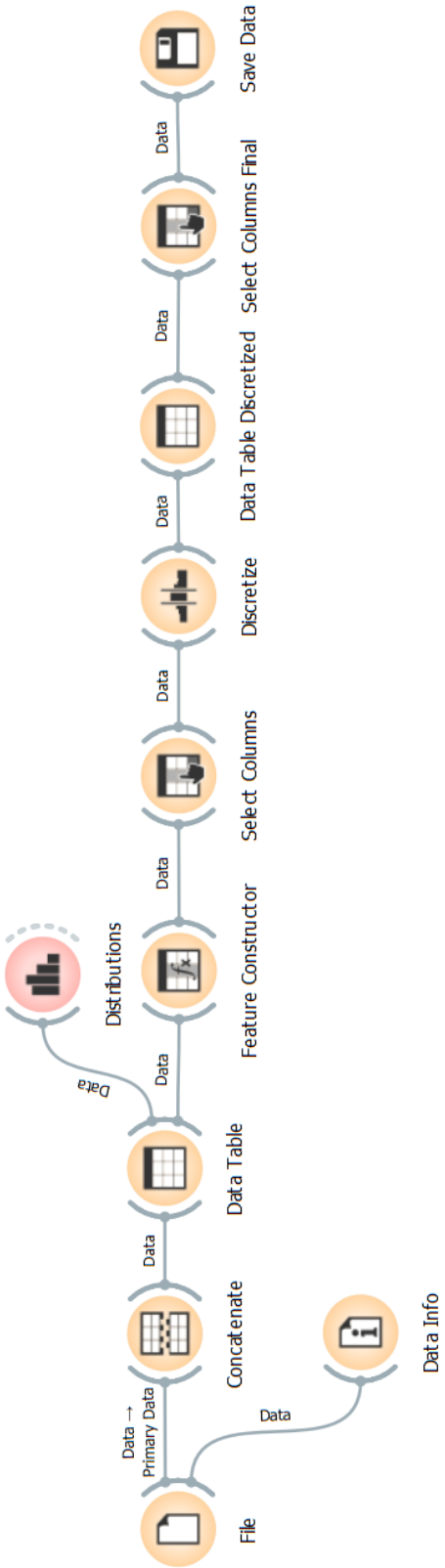
[154] M. A. Musen, „The protégé project: a look back and a look forward“, *AI Matters*, sv. 1, izd. 4, str. 4–12, lip. 2015, doi: 10.1145/2757001.2757003.

6. Dodatak A

Nakon pretrage reprezentativnih uzoraka u odabranim domenskim tekstovima i prikupljanja rezultata pretrage potrebno je vrijednosti entropije raspodijeliti u razrede. Za raspodjelu vrijednosti entropije u razrede primijenjena je diskretizacija entropije.

Za diskretizaciju vrijednosti entropije uzoraka koji su korišteni u izgradnji modela za odabir najučinkovitijeg algoritma za određenu domenu vrijednosti entropije uzoraka korišten je Orange Data Suite alat. To je alat za vizualizaciju podataka otvorenog koda, strojno učenje i rudarenje podataka. Baziran je na tehnikama vizualnog programiranja čime se omogućuje brza i kvalitativna analiza podataka i interaktivna vizualizacija podataka [85].

Na slici 6.1. prikazan je Orange Data Suite model pomoću kojega se vrijednost entropije razvrstava u razrede. Model se sastoji od dodataka (engl. *widget*) koji su sastavni dio Orange Data Suite. Neki od dodataka su: dodatak datoteke koji služi za učitavanje izvornih podataka spremljenih u datoteci, dodatka tablice podataka u kojeg se spremaju učitani podaci, komunikacijskog kanala koji prosljeđuje skup podataka iz dodatka datoteka u tablicu podataka, dodatka za diskretizaciju i distribuciju.

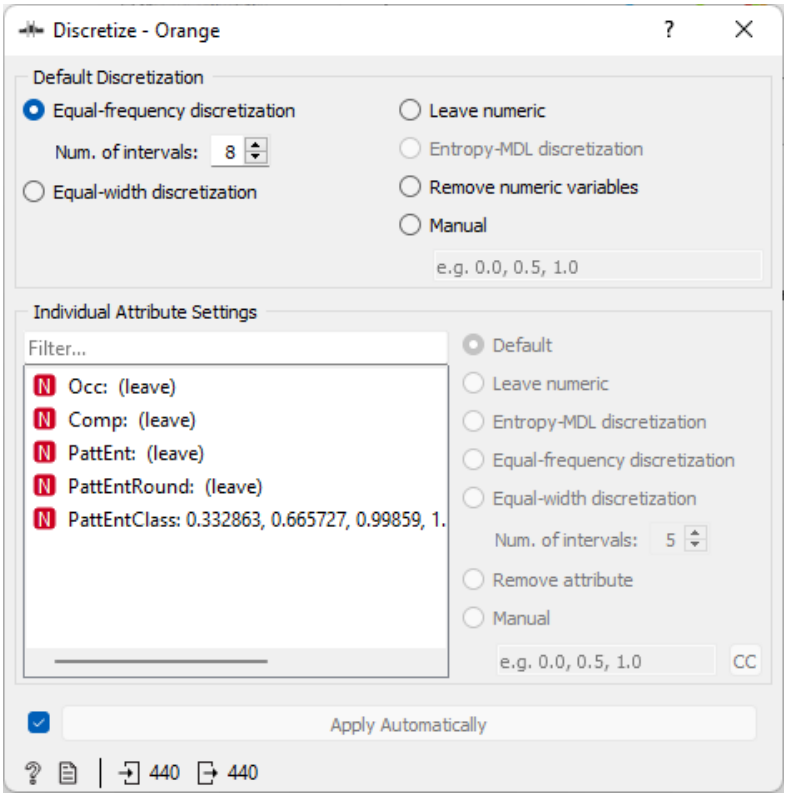


Slika 6.1. Orange Data Suite model

Dodatak za prihvat datoteke (engl. *File*) sa rezultatima prikazan na slici 6.1. čita ulaznu podatkovnu datoteku (tablicu podataka s instancama podataka) i šalje skup podataka u svoj izlazni kanal. Dodatak čita podatke iz programa Excel (.xlsx), jednostavne datoteke razdvojene tabulatorima (.txt), datoteke razdvojene zarezima (.csv).

Konstruktor značajki (engl. *Feature Constructor*) prikazan na slici 6.1. omogućuje ručno dodavanje značajki (stupaca) u skup podataka. Nova značajka može biti izračun postojeće ili kombinacija nekoliko (zbrajanje, oduzimanje, itd.). Moguće je odabrati koja će to biti vrsta značajke (diskretna, kontinuirana ili niz) i koji su njezini parametri (naziv, vrijednost, izraz).

Postavke Orange Data Suite dodatka za distribuciju sa slike 6.2. prikazuju raspodjelu vrijednosti diskretnih ili kontinuiranih atributa iz ulaznog skupa podataka. Izlazne vrijednosti su skup podataka s diskretiziranim vrijednostima. Broj razreda opisan je u poglavlju 3.5. Podjela vrijednosti entropije u razrede.



Slika 6.2. Orange Data Suite diskretizacija ulaznog skupa podataka vrijednosti entropije uzoraka

7. Dodatak B

Za potrebe analize podataka u ovoj doktorskoj disertaciji korišten je IBM SPSS programski paket proizvoda. IBM SPSS je programski paket proizvoda koji nudi mogućnosti napredne statističke analize, biblioteke algoritama strojnog učenja, analizu teksta, integraciju s velikim podacima. SPSS je široko korišten program za statističku analizu u svim granama znanosti Također ga koriste istraživači tržišta, zdravstveni istraživači, anketne tvrtke, obrazovni istraživači, marketinške organizacije, eksperti za rudarenje podataka i drugi.

Unutar SPSS programskog paketa proizvoda, SPSS Statistics je statistička softverska platforma. SPSS Statistics nudi korisničko sučelje i robustan skup značajki koje omogućuju brzo izvlačenje korisnih uvida iz podataka. Napredni statistički postupci pomažu osigurati visoku točnost i kvalitetno donošenje odluka. Uključeni su svi vidovi životnog ciklusa analitike, od pripreme podataka i upravljanja do analize i izvješćivanja.

Prije korištenja alata za analizu podataka napravljena je diskretizacija vrijednosti entropije uzoraka i priprema okruženja za korištenje podataka. Priprema okruženja unutar IBM SPSS programskog paketa podrazumijeva kreiranje varijabli, a zatim i popunjavanje varijabli podacima koji će se koristiti u analizi.

Karakteristike jedne varijable prikazane su na slici 7.1.:

Name	Type	Width	Decimals	Label	Values	Missing	Columns	Align	Measure
PatEntClass7Gen	Numeric	24	16		None	None	15	Right	Scale

Slika 7.1. Karakteristike razreda entropije unutar IBM SPSS softverskog paketa

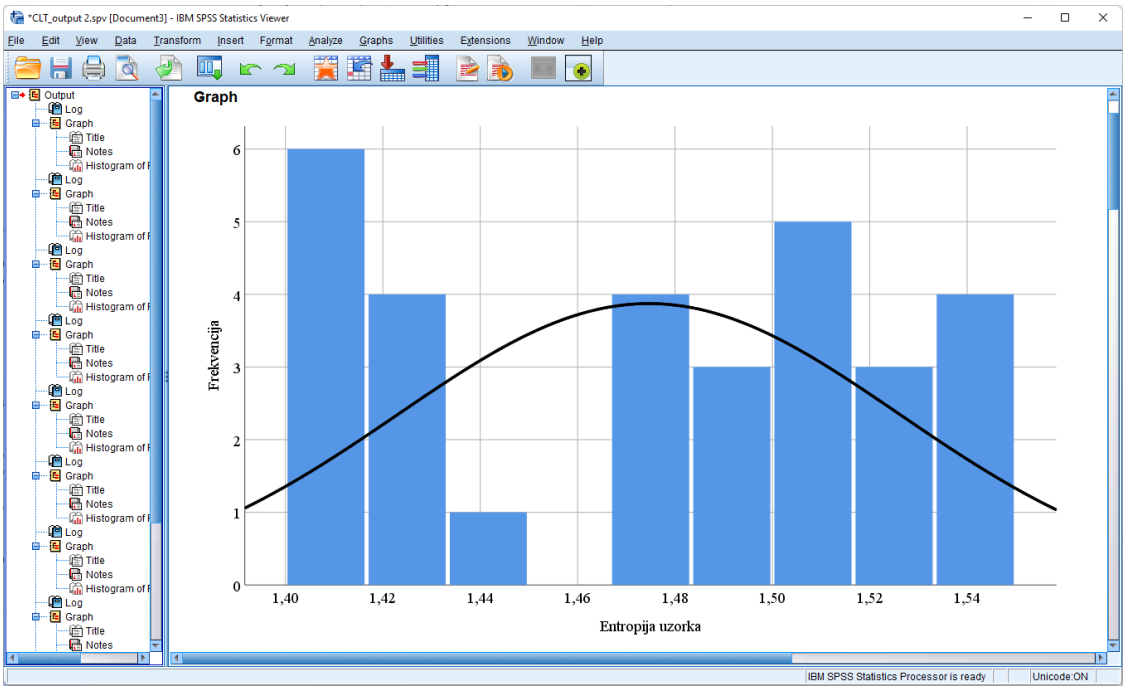
Termin varijable identičan je terminu odabranih razreda entropije korištenih u analizi. Nakon pripreme podatka za analizu napravljena je analiza distribucije istih. Distribucija podataka odabranih razreda prikazana je pomoću središnjeg graničnog teorema, grafički prikazana pomoću histograma. Na osnovu izgleda histograma doneseni su zaključci o statističkoj prirodi populacije.

Na slici 7.2. prikazan je isječak dijela podataka iz SPSS Statistics za koje je napravljena analiza središnjeg graničnog teorema.

PatEntClass7Gen	PatEntClass7SampGen	PatEntClass9Gen	PatEntClass9SampGen
1,3663146570364000	1,4056390620000000	1,9086089620000000	1,8112781240000000
1,5106924110431000	1,4056390620000000	1,8825874750000000	1,8112781240000000
1,5000000000000000	1,4056390620000000	1,9362781240000000	1,8112781240000000
1,4056390622295700	1,5487949410000000	1,8825874750000000	1,8112781240000000
1,5000000000000000	1,5000000000000000	1,8299395500000000	1,9056390620000000
1,4913146570364000	1,5000000000000000	1,9423688200000000	1,9056390620000000
1,5000000000000000	1,5000000000000000	1,8424558340000000	1,9362781240000000
1,4237949406954000	1,4185644430000000	1,9288146570000000	1,8802408150000000
1,5487949406954000	1,4197367180000000	1,9620310230000000	1,9237949410000000
1,5000000000000000	1,5461796920000000	1,8795524280000000	1,8496017530000000
1,4056390622295700	1,4185644430000000	1,9772170010000000	1,9772170010000000
1,4056390622295700	1,5016144720000000	1,9836796920000000	1,8828560640000000
1,5000000000000000	1,5016144720000000	1,9237949410000000	1,8050365330000000
1,4056390622295700	1,5016144720000000	1,9362781240000000	1,9575547990000000
1,4056390622295700	1,5271036270000000	1,8050365330000000	1,9575547990000000
1,5487949406954000	1,4777930390000000	1,8704744890000000	1,9132265380000000
1,4056390622295700	1,5167028040000000	1,8919340030000000	1,9575547990000000
1,4002047062527900	1,5008828850000000	1,8825874750000000	1,8186764450000000
1,4056390622295700	1,5164336140000000	1,8726915990000000	1,8726915990000000
1,5000000000000000	1,4677244230000000	1,9620310230000000	2,0000000000000000
1,4735958170526900	1,4056390620000000	1,9772170010000000	1,9287289080000000
1,5000000000000000	1,4480191160000000	1,8186764450000000	1,8050365330000000
1,5000000000000000	1,4772170010000000	1,9772170010000000	2,0000000000000000
1,5000000000000000	1,5461796920000000	1,8112781240000000	1,9544340030000000
1,5000000000000000	1,4689514570000000	1,9715667060000000	1,9715667060000000
1,4056390622295700	1,4056390620000000	1,8919340030000000	1,9056390620000000
1,5487949406954000	1,5487949410000000	1,9544340030000000	1,9056390620000000
1,4056390622295700	1,5306390620000000	1,9480191160000000	1,9056390620000000
1,5000000000000000	1,4056390620000000	1,9544340030000000	1,8496017530000000
1,5306390622295700	1,4216744280000000	1,8058437680000000	2,0000000000000000

Slika 7.2. Vrijednosti 7. i 9. razreda entropije uzoraka korištenih u izgradnji i validaciji modela DNA domene

Na slici 7.3. prikazan je primjer rezultata analize pomoću Output Viewer modula unutar programskog paketa IBM SPSS. Pomoću Output Viewera prikazani su rezultati traženih statističkih analiza. Omogućena je jednostavnija interpretacija podataka, selektivni prikaz rezultata, te grafičko i tablično prikazivanje podataka. Output Viewer također služi i kao poveznica između SPSS-ovih rezultata i drugih programa poput Worda ili Excela.



Slika 7.3. IBM SPSS Statistics Viewer

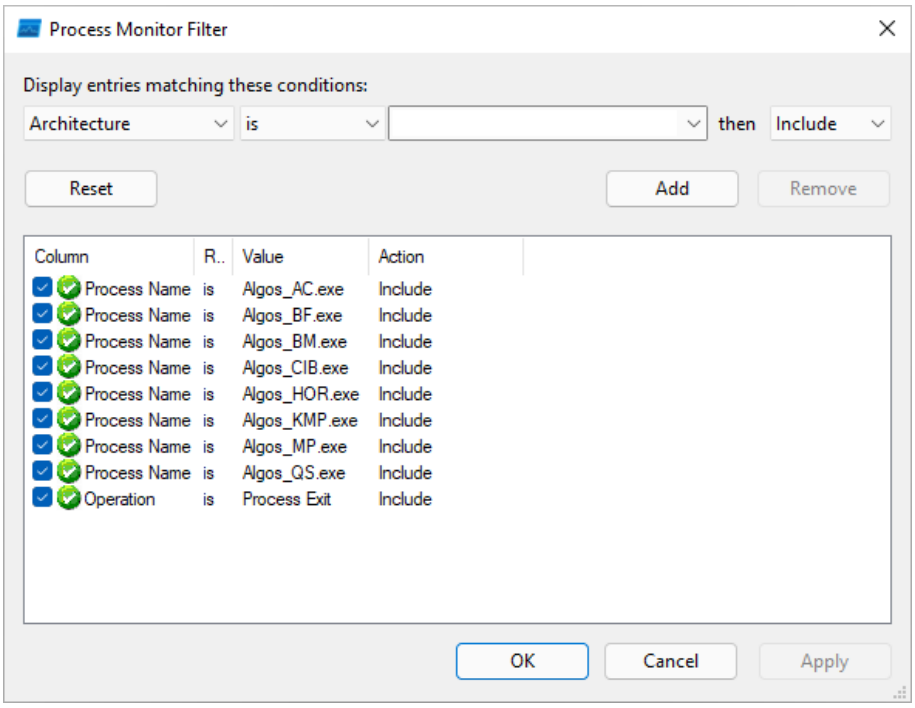
8. Dodatak C

Za potrebe mjerenja iskorištenja radne memorije novog algoritma upotrjebljen je alat Microsoft Process Monitor. Process Monitor napredni je alat za praćenje Windows datotečnog sustava, registra i aktivnosti procesa u stvarnom vremenu. Kombinira značajke dva naslijeđena uslužna programa Sysinternals, Filemon i Regmon, i dodaje opsežan popis poboljšanja uključujući filtriranje, opsežna svojstva događaja kao što su ID-ovi sesija i imena korisnika, pouzdane informacije o procesu, pune nizove niti s integriranim simbolom podrške za svaku operaciju, istovremeno prijavljivanje u datoteku i još mnogo toga.

Prije pokretanja eksperimentalne analize potrebno je podesiti alat na način da se konfiguriraju filteri za hvatanje događaja. Nakon pokretanja algoritama i pretrage uzoraka u odabranim reprezentativnim tekstovima potrebno je napraviti izdvajanje podataka o stanju sustava Windows.

Fokus analize je podatak o potrošnji memorije (kolona privatni bajtovi, engl. *private bytes*). Privatni bajtovi su trenutna veličina, u bajtovima, memorije koju je proces alocirao i koja se ne može dijeliti s drugim procesima.

Na slici 8.1. prikazane su postavke filtera Process Monitor alata.



Slika 8.1. Postavke filtera Process Monitora za skupljanje empirijskih podataka eksperimentalne analize

Rezultati analize zauzeća memorije pomoću Process Monitor alata prikazani su na slici 8.2. Rezultati su izvezeni u CSV format i analizirani pomoću Microsoft Excel alata.

Slika 8.2. Prikupljeni empirijski podaci pomoću Process Monitora za eksperimentalnu analizu

Životopis

Ivan Markić je rođen u Ljubuškom, Bosna i Hercegovina, 1984. godine. Osnovnu i srednju školu (Opća gimnazija) završio je u Ljubuškom. Titulu diplomiranog inženjera računarstva dobio je 2007. godine na Fakultetu strojarstva i računarstva Sveučilišta u Mostaru. Tijekom studija dobio je Dekanovu nagradu za najbolje studente. Od 2009. do 2017. zaposlen je u Croatia osiguranju kao razvojni programer, a kasnije i kao voditelj sektora informatike. Od 2017. zaposlen je u Central osiguranju d.d. Sarajevo, a od 2021. zaposlen je i u ASA osiguranju d.d. Sarajevo kao voditelj sektora informatike. U 2012. godini upisao je poslijediplomski studij na Fakultetu elektrotehnike, strojarstva i brodogradnje, Sveučilišta u Splitu. Fokus istraživanja su algoritmi za pretragu nizova, obrada i analiza podataka.

Curriculum Vitae

Ivan Markić was born in Ljubuški, Bosnia and Herzegovina, in 1984. He finished primary and secondary school (General Gymnasium) in Ljubuški. He received the Bachelor of Science in Computer Science in 2007 at the Faculty of Mechanical Engineering and Computing, University of Mostar. During his studies, he received the Dean's Award for Best Students. From 2009 to 2017, he was employed by Croatia insurance Ltd. as a software developer and later as the head of the IT department. Since 2017, he has been employed by Central insurance Ltd. Sarajevo, and since 2021 he has been employed by ASA insurance Ltd. Sarajevo as the head of the IT department. In 2012, he enrolled in postgraduate studies at the Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, University of Split. His research interests include string search algorithms, data processing, and analysis.