PD.TVZ.HR
POLYTECHNIC & DESIGN Vol. 8, No. 2, 2020.
DOI: 10.19279/TVZ.PD.2020-8-2-08

AUTOMATSKO SAŽIMANJE HRVATSKOG TEKSTA

AUTOMATIC SUMMARIZATION OF CROATIAN TEXT

Zvonimir Petrović[1], Vjeran Bušelić[2]

[1]Tehničko veleučilište u Zagrebu, Vrbik 8, Zagreb, Hrvatska, Student
[2]Tehničko veleučilište u Zagrebu, Vrbik 8, Zagreb, Hrvatska

SAŽETAK

Prototip aplikacije predstavljene u radu nastao je tijekom izrade završnog rada prvog autora. U ovom članku su ukratko navedene osnovne trenutno dostupne metode automatskog sažimanja teksta uz isticanje izazova korištenja tekstova na hrvatskom jeziku. Dan je vrlo sažeti prikaz izrade prototipa odgovarajuće internetske aplikacije. Uz objašnjenje arhitekture, pojašnjeni su moduli i pokazan krajnji rezultat. U završnom dijelu diskutirani su uočeni izazovi i ponuđene smjernice za daljnji razvoj.

Ključne riječi: *automatsko sažimanje teksta, ekstrakcijski algoritam, hrvatski jezik, internetska aplikacija, prototip*

ABSTRACT

The prototype of the application presented in the paper was created during final batchelor exam of the first author. This article briefly lists the basic currently available methods of automatic text summarization while highlighting the challenges of using texts in the Croatian language. A very concise overview of the web application prototype is given. With an explanation of the architecture, the modules are explained and the end result is shown. In the final part, the identified challenges and guidelines for further development are discussed.

Keywords: *automatic text summarization, extraction algorithm, Croatian language, web application, prototype*

1. UVOD
1. INTRODUCTION

Automatski sustavi sažimanja teksta osmišljeni su kako bi izdvajanje relevantnih informacija iz nekog izvora bilo što brže i jednostavnije. Takvi sustavi zadržavaju najvažnije informacije izvornog dokumenta u bitno manjem opsegu teksta što skraćuje vrijeme čitanja i poglavito razumijevanja ključnih poruka. Postupak automatskog sažimanja teksta predstavlja rješenje u vidu bržeg i učinkovitijeg pronalaska ključnih informacija, uz istovremeno smanjenja onih redundantnih. Nadalje, povećava se efektivnost pretraživanja informacija jer je potrebno indeksirati bitno manje podataka te je u potpunosti nepristran, za razliku od ljudi koji mogu utjecati na prijevod kako bi spriječili širenje određenih ideja, informacija i sl.

Svrha sažimanja teksta počiva na smanjivanju količine teksta potrebnog za obradu ili čitanje uz očuvanje najbitnijih poruka i informacija unutar originalnog teksta ili više različitih tekstova. Razvoj softverskih rješenja suočava autore s nizom problema od kojih je najveći problem prepoznavanja svih ključnih informacija iz izvornog teksta koje se trebaju pojaviti u sažetku.

Koliko je potrebno programerskog znanja, koja stručna literatura postoji, te kolika bi bila iskoristljivost i kompleksnost izrade programa za sažimanje teksta na hrvatskom jeziku, pitanja su kojima se bavio rad „Izazovi strojnog sažimanja tekstova na hrvatskom jeziku“, obranjen na završnom ispitu studija Informatike, Tehničkog veleučilišta u Zagrebu [1]. U ovom su članku, temeljenom na tom radu, sažete najosnovnije stručne podloge, dan prikaz složenosti programskog rješenja te izložena osnovna razmišljanja o daljnjim koracima.

Generalno se u praksi koriste dvije metode automatskog sažimanja teksta: ekstrakcijska i apstrakcijska. Ekstrakcijska metoda koristi postojeće riječi, fraze ili rečenice iz izvornog teksta kako bi se pomoću njih izgradila sažeta verziju teksta. Za razliku od toga, apstrakcijska metoda prepoznaje semantiku samog teksta i kreira sažetak na način sličniji ljudskom pristupu izrade sažetka [2].

1.1. EKSTRAKCIJSKA METODA
1.1. EXTRACTION METHOD

Ekstrakcijska metoda sažimanja teksta je način sažimanja u kojem se najprije odrede važne sekcije teksta koje će sačinjavati sažetak. Razlikuju se dva tipa konstrukcije tih sekcija; prikaz temeljen na temi teksta i prikaz temeljen na indikatorima teksta. Prikazom teme oblikuje se sažetak temeljen na temi teksta koji se sažima. Prilikom sažimanja teksta na taj način razlikuju se tehnički pristupi sažimanju koji se dijele na: pristupe temeljene na frekvenciji, pristupe temeljene na ključnim riječima, te latentno-semantičkoj analizi [3].

Prikaz temeljen na indikatorima u tekstu stvara listu rečenica poredanih po važnosti njezinih indikatora kao što su njezina dužina, fraze unutar rečenice, lokaciju unutar teksta i sl. Nakon odabira načina ekstrakcije svaka rečenica dobiva numeričku reprezentaciju težine koju ista ima u tekstu. U pristupu temeljenom na temi ta težina objašnjava koliko dobro rečenica opisuje najbitnije dijelove teme kojom se tekst bavi.

Težina određene rečenice u pristupu temeljenom na indikatorima ovisi o zadanim parametrima (utezima) koje rečenica zadovoljava. Nakon tog procesa, algoritam za sažimanje uzima zadani broj rečenica koje imaju najveću težinu, koje će se pojaviti u generiranom sažetku [4].

1.2. APSTRAKCIJSKA METODA
1.2. ABSTRACTION METHOD

Apstrakcijskom metodom sažimanja teksta dobiva se sažetak koji koristi rečenice i izraze različite od izvornog teksta. Korištenjem ideja i koncepata originalnog teksta generira se sažetak koji je puno bliži izgledu sažetka kakvog bi napisao čovjek. Budući da je svaki jezik vrlo kompleksan te samo izražavanje široko, za ovakav pristup automatskom sažimanja teksta potreban je mnogo složeniji pristup obradi jezika, te vrlo jaki računalni resursi.

Iako apstrakcijska metoda nudi bolje rezultate, sama izrada takvih algoritama je puno složenija i računalno zahtjevna, što je glavni razlog zašto ekstrakcijska metoda dominira kao najčešće korišteni oblik sažimanja teksta, pa će biti prikazana i u ovom radu.

U radu je korišten algoritam za sažimanje teksta otvorenog koda (*engl. Open Source*), baziran na „The Tokenizer“ algoritmu [5]. Najveći izazov je svakako bio prilagoditi algoritam na rad s hrvatskim dijakritičkim znakovima kako bi se mogli sažimati hrvatski tekstovi [6].

2. PROTOTIP INTERNETSKE APLIKACIJE SAZMI_ME
2. WEB PROTOTYPE APPLICATION SAZMI_ME

Prototip aplikacije predstavljene u radu nastao je za potrebe završnog rada prvog autora. Cilj rada nije bio napraviti najbolju inačicu softvera za sažimanje tekstova na hrvatskom jeziku, već nakon izučene literature, napraviti radni prototip i ukazati na težinu izazova koji se pojavljuju u radu. Korišteno je osnovno informatičko razvojno mrežno okruženje, a sama izrada koda je opisana i dokumentirana u ovom paragrafu, kako bi se na razvoj budućih inačica potaknuli i manje informatički obrazovani stručnjaci.

2.1. ALGORITAM ZA SAŽIMANJE TEKSTA
2.1. TEXT SUMMARIZATION ALGORITHM

Ekstrakcijski algoritam originalno je predviđen za engleski jezik. Prilikom testiranja s hrvatskim tekstovima uočen je slijedeći niz problema:

- Algoritam je prepoznavao sve točke kao krajeve rečenica. Tako i one unutar datuma, na krajevima titula te skraćenica.

- S druge strane, ukoliko na kraju rečenice točka nije postojala, obje rečenice je spojio u jednu.
- Nije postojao način prepoznavanja hrvatskih dijakritičkih znakova.
- Bilo je potrebno uključiti prepoznavanje prefiksa i sufiksa hrvatskog jezika.

Problematika je riješena kroz nekoliko koraka, kako slijedi:

- Dodavanjem konstanti hrvatske abecede s velikim i malim slovima te dijakritičkim znakovima.
- Dodavanjem konstanti za prefikse i sufikse u hrvatskom jeziku te zamjenom značenja točke u tekstu nakon tih riječi kako algoritam ne bi pogrešno prepoznao kraj rečenice (npr. prof., dr., gosp., gđa., …). Svaka kratica morala je biti unesena ručno u algoritam.
- Ukoliko u tekstu rečenica ne završava točkom, a između nje i iduće rečenice je prijelaz u drugi red, te prva riječ iduće rečenice počinje velikim slovom, prijelaz u novi red se označava kao kraj prve rečenice i početak iduće.
- Prepoznavanje datuma je riješeno tako da ukoliko se ispred točke nalaze brojevi, provjerava pojava datuma u hrvatskom obliku (dd. mm. gggg.). Prepoznavanje datuma omogućeno je i u slučaju kad je mjesec naveden imenom, jer su uvedene varijable s imenima svih mjeseci.

Proširenjem mogućnosti izvornog algoritma nije narušena prvotna ideja da se koristi za sažimanje tekstova na engleskom jeziku, te algoritam jednako dobro radi i s engleskim i hrvatskim jezikom.

U ovoj inačici korisniku nije omogućeno određivanje postotka sažimanja teksta, tako da postotak sažimanja ovisi isključivo o tekstu koji se obrađuje. Testiranjem algoritma utvrđen je prosječni postotak sažimanja između 65% i 80%, što najčešće zadovoljava postotak sažimanja.

Budući da se sažimanje temelji na odabiru najbitnije rečenice iz svakog paragrafa, može se pojaviti problem ukoliko tekst nije podijeljen na paragrafe, ili ako se u svakom paragrafu nalazi samo jedna rečenica.

Ako je u svakom paragrafu po jedna rečenica, algoritam će vratiti sažetak koji se sastoji od svih rečenica, a ako tekst nije odijeljen u paragrafe, kao rezultat dobiva se samo jedna rečenica. Rješenje ovog problema svodi se na predradnju provjere formatiranja ponuđenog teksta te je ukoliko je potrebno, tekst potrebno preformatirati.

2.2. MOGUĆA POBOLJŠANJA
2.2. POSSIBLE IMPROVEMENTS

Obzirom na princip odabira ključnih rečenica iz originalnog teksta, a koje se biraju na temelju učestalosti riječi, postoji mogućnost poboljšanja algoritma unošenjem ključnih riječi. Taj bi dodatak značajno poboljšao kvalitetu sažetka, jer bi najteži dio sažimanja, povezan sa shvaćanjem ključnih informacija u tekstu bio prepušten osobi koja koristi algoritam. Ovo je poglavito korisno npr. u sažimanju stručnih preglednih članaka, gdje informacije o ključnim riječima već postoje.

Bitno je napomenuti kako je algoritam tehnološki ograničen limitom od 31.000 riječi. Korištena metoda unutar *NodeJS* servera *child process* ograničena je (tom) količinom podataka koje se u jednom trenutku može prenijeti Python algoritmu. U slučaju potrebe, ovaj problem je rješiv bilo da se Python kod prevede u JavaScript kod, pa da se sažimanje vrši direktno u *NodeJS* serveru, ili dodavanjem metode koja bi vršila funkcionalnost čitanja teksta iz datoteke.

2.3. GLAVNA FUNKCIJA
2.3. MAIN FUNCTION

Centalno mjesto *sazmi_me* aplikacije predstavlja glavna funkcija u kojoj se algoritmu predaju podaci koji dolaze iz klijentskog sloja. Postavljaju se naslov (*engl.* title) i sadržaj (*engl.* content). Nakon postavljanja varijabli koje algoritam koristi instancira se novi objekt tipa *SummaryTool*. Zatim se puni *sentences_dic* varijabla te se zajedno s naslovom i sadržajem predaje metodi *st.get_summary* i sažetak se sprema u varijablu *summar*y. Zatim se sažetak ispisuje, a nakon toga ispisuju se osnovni statistički podaci o dužini originalnog teksta, dužini dobivenog sažetka i omjeru sažetosti. Sys.stdout.flush() funkcija se koristi kako bi se sažetak proslijedio serveru.

2.4. FUNKCIJA RAZDVAJANJA U REČENICE - SPLIT_CONTENT_TO_SENTENCES

2.4. SPLIT CONTENT TO SENTENCES FUNCTION

Ova funkcija dijeli sadržaj na zasebne rečenice. To se postiže zamjenom točaka koje se pojavljuju na krajevima datuma, prefiksa i sufiksa, pretvaranjem prelazaka u novi red u krajeve rečenica, te općenitom standardizacijom izgleda rečenice. Nakon toga, svim točkama koje su preostale dodaje se ključna riječ <stop> kako bi algoritam znao da je na tom mjestu kraj rečenice. Takav sadržaj se prosljeđuje *get_sentences_ranks* funkciji.

2.5. FUNKCIJA PREKLAPANJA REČENICA

2.5. SENTENCES INTERSECTION FUNCTION

Funkcija uspoređuje dvije rečenice i vraća rezultat njihovih preklapanja. Svaka rečenica je podijeljena na pojedine riječi, funkcija prebroji koliko zajedničkih riječi imaju, te vraća rezultat za svaku rečenicu koji je potom normaliziran na način da ga dijelimo s prosječnom dužinom obje rečenice.

2.6. FUNKCIJA ODREĐIVANJA VAŽNOSTI REČENICA

2.6. SENTENCES RANK FUNCTION

Funkcija uzima tekst koji joj je proslijeđen i daje rezultat za svaku rečenicu u tekstu. Izračun je napravljen u dva koraka: tekst se dijeli u rečenice i sva preklapanja između rečenica se spremaju u matricu. Na taj je način tekst pretvoren u dvodimenzionalni niz u koji se računa i upisuje njena težina u odnosu na cjelinu.

```
def main():

    title = sys.argv[1]

    content = sys.argv[2]

    # Create a SummaryTool object
    st = SummaryTool()

    # Build the sentences dictionary
    sentences_dic = st.get_sentences_ranks(content)

    # Build the summary with the sentences dictionary
    summary = st.get_summary(title, content, sentences_dic)

    # Print the summary
    print (summary)

    # Print the ratio between the summary length and the original length
    print ("")
    print ("_Broj riječi u izvornom tekstu: %s" % (len(title) + len(content)))
    print ("Broj riječi u sažetku: %s" % len(summary))
    print ("Postotak sažetosti: %s" % (100 - (100 * (len(summary) / (len(title) + len(content))))))
    sys.stdout.flush()

if __name__ == '__main__':
    main()
```

Slika 1 *Glavna funkcija; Izvor: Autor*

Figure 1 *Main function; Source: Author*

```
def split_content_to_sentences(self, content):
    content = " " + content + "  "
    content = content.replace("\n",". ")
    content = re.sub(prefixes, "\\1<prd>", content)
    content = re.sub(months, "<prd>\\1", content)
    if "dr." in content: content = content.replace("dr.","dr<prd>")
    if "prof." in content: content = content.replace("prof.","prof<prd>")
    if "mag." in content: content = content.replace("mag.","mag<prd>")
    content = re.sub("\s" + alphabets + "[.] "," \\1<prd> ", content)
    content = re.sub(acronyms+" "+starters,"\\1<stop> \\2", content)
    content = re.sub(alphabets + "[.]" + alphabets + "[.]" + alphabets + "[.]","\\1<prd>\\2<prd>\\3<prd>", content)
    content = re.sub(alphabets + "[.]" + alphabets + "[.]","\\1<prd>\\2<prd>", content)
    content = re.sub(" " + suffixes + "[.] " + starters," \\1<stop> \\2", content)
    content = re.sub(" " + suffixes + "[.]"," \\1<prd>", content)
    content = re.sub(" " + alphabets + "[.]"," \\1<prd>", content)
    content = re.sub(dates + " " + lowercase,"\\1<prd> \\2", content)
    content = re.sub(digits + "[.]" + digits, "\\1<prd>\\2", content)

    if "”" in content:
        content = content.replace(".”","”.")
    if "\"" in content:
        content = content.replace(".\"","\".")
    if "!" in content:
        content = content.replace("!\"","\"!")
    if "?" in content:
        content = content.replace("?\"","\"?")
    content = content.replace(".",".<stop>")
    content = content.replace("?","?<stop>")
    content = content.replace("!","!<stop>")
    content = content.replace("<prd>",".")
    sentences = content.split("<stop>")
    sentences = sentences[:-1]
    content = [s.strip() for s in sentences]
    return content
```

Slika 2 *Funkcija razdvajanja u rečenice; Izvor: Autor*

Figure 2 *Split Content to Sentences function; Source: Author*

```
def sentences_intersection(self, sent1, sent2):

    # split the sentence into words/tokens
    s1 = set(sent1.split(" "))
    s2 = set(sent2.split(" "))

    # If there is no intersection, just return 0
    if (len(s1) + len(s2)) == 0:
        return 0

    # We normalize the result by the average number of words
    return len(s1.intersection(s2)) / ((len(s1) + len(s2)) / 2)
```

Slika 3 *Funkcija preklapanja rečenica; Izvor: Autor*

Figure 3 *Sentences Intersection function; Source: Author*

3. REZULTAT RADA APLIKACIJE SAZMI_ME

3. APPLICATION SAZMI_ME RESULTS

Aplikacija ima vrlo jednostavno sučelje kroz kojeg se unosi ili prenosi proizvoljan tekst. Algoritam sažima tekst te se rezultat prikazuje uz informaciju o naslovu i sadržaju (sažeti tekst), kao i osnovnim statističkim pokazateljima: broj riječi u izvornom tekstu, broj riječi u sažetku i postotak sažetosti. Iz priloženog primjera vidljiv je potpuno funkcionalan rad automatskog sažimanja teksta na hrvatskom jeziku korištenjem ekstrakcijskog algoritma.

```
def get_sentences_ranks(self, content):

    # Split the content into sentences
    sentences = self.split_content_to_sentences(content)

    # Calculate the intersection of every two sentences
    n = len(sentences)
    values = [[0 for x in range(n)] for x in range(n)]
    for i in range(0, n):
        for j in range(0, n):
            values[i][j] = self.sentences_intersection(sentences[i], sentences[j])

    # Build the sentences dictionary
    # The score of a sentences is the sum of all its intersection
    sentences_dic = {}
    for i in range(0, n):
        score = 0
        for j in range(0, n):
            if i == j:
                continue
            score += values[i][j]
        sentences_dic[self.format_sentence(sentences[i])] = score
    return sentences_dic
```

Slika 4 *Funkcija određivanja važnosti rečenica; Izvor: Autor*

Figure 4 *Sentences Rank function; Source: Author*

4. ZAKLJUČAK
4. CONCLUSION

U radu je prikazan prototip internetske aplikacije izgrađene na ekstrakcijskom algoritmu za sažimanje teksta. Prvenstveno zbog ograničenosti hardvera i nepostojanja dobrih rješenja u apstrakcijskom pristupu primijenjen je ekstrakcijski način sažimanja teksta. Algoritam je prilagođen za rad s hrvatskim jezikom uz očuvanje mogućnosti sažimanja tekstova na engleskom jeziku.

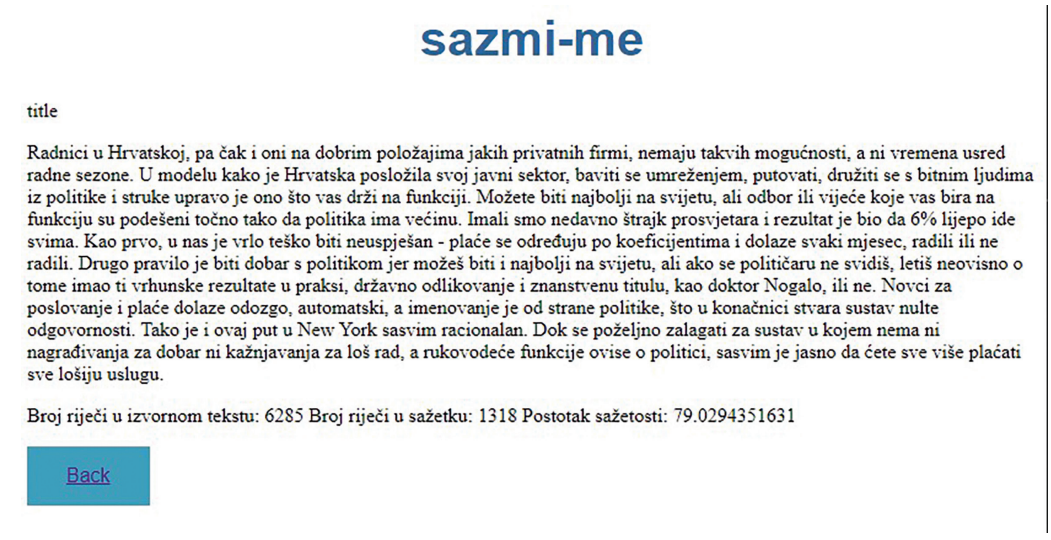
Upotreba algoritama za sažimanje teksta u ovom ili još bolje, primjenom apstrakcijskih algoritama ima veliki potencijal u današnjem svijetu koji je preopterećen količinom informacija. Skraćivanje teksta ovakvim algoritmima, zadržavajući puni smisao, predstavlja olakšanje stručnjacima, ali i običnim korisnicima koji žele saznati više, u što kraćem vremenu.

Glavna poboljšanja koja bi se mogla riješiti i implementirati u neposrednoj budućnosti odnose se na format teksta koji se unosi. Nužno je dodati još metoda kojima bi se pobrinulo da je tekst formatiran na adekvatan (dogovoreni) način.

U aplikaciji je korišten postojeći algoritam. Daljnji razvoj algoritma predstavlja jako velik izazov prvenstveno zbog jezične kompleksnosti, neovisno radi li se o engleskom ili hrvatskom jeziku. Problematičnost obrade jezika i njegova kompliciranost čini srž problema u razvoju ovakvih i sličnih algoritama, ali napretkom tehnologije i konstantnim povećanjem količine dostupnih informacija postoji jasna potražnja za rješenjima u ovom području. Može se zaključiti da će tehnologije sažimanja teksta imati veliku ulogu u budućnosti, a po razvoju i stavljanju na slobodnu upotrebu odgovarajućih algoritama, nije potrebno veliko informatičko znanje da se oblikuju proizvodi dostupni najširoj publici kako je i pokazano u ovom radu.

Slika 5 *Primjer internetskog sučelja i unosa proizvoljnog teksta; Izvor: Autor*

Figure 5 *Example of web interface with data entry; Source: Author*



Slika 6 *Primjer rezultata; Izvor: Autor*

Figure 6 *Example results; Source: Author*

5. REFERENCE
5. REFERENCES

[1.] Zvonimir Petrović (2020): Izazovi strojnog sažimanja tekstova na hrvatskom jeziku / završni rad, Zagreb: Tehničko veleučilište u Zagrebu

[2.] Pai, Ms. Anusha: Text Summarizer Using Abstractive and Extractive Method, International Journal of Engineering & Technology (IJERT), Vol.3 Issue 5, 05.2014. (971-975)

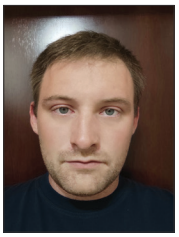
[3.] Brajković, Emil; Volarić, Tomislav; Vasić, Daniel: "Pregled tehnika automatskog sažimanja teksta", Fakultet prirodoslovno-matematičkih i odgojnih znanosti, Matice hrvatske b.b., 05.2018

[4.] Tafseer Nayeem, Mir: Methods of sentence extraction, abstraction and ordering for automatic text summarization, Islamic University of Technology, Canada, 2017, 107 (4-8)

[5.] Babluki, Shlomi, "Build your own summary tool!", 28.3.2013., The Tokenizer, https://thetokenizer.com/2013/04/28/build-your-own-summary-tool/, preuzeto 23.08.2019.

[6.] Tadić, Marko; Brozović-Rončević, Dunja; Kapetanović, Amir: "Hrvatski jezik u digitalnom dobu", META-NET, eBook, Germany, 2012, 93 (34)

AUTORI · *AUTHORS*



● **Zvonimir Petrović**
Diplomirao na Tehničkom Veleučilištu u Zagrebu 2020. godine i stekao naziv bacc. ing.techn.inf. Nakon studija zapošljava se u informatičkoj tvrtki Libusoft Cicom d.o.o. u Zagrebu u odjelu poslovne inteligencije. Privatni interesi autora počivaju na razvoju informatičkih rješenja u jezičnom području.

Korespondencija · *Correspondence*
zpetrovic@tvz.hr

● **Vjeran Bušelić -** nepromjenjena biografija nalazi se u časopisu Polytechnic & Design Vol. 7, No. 2, 2019.

Korespondencija · *Correspondence*
vjeran.buselic@tvz.hr