Univerzitet u Beogradu
Elektrotehnički fakultet

Željko Tanjević, 2020/0465

Sažimanje novinskih članaka

projekat iz predmeta Principi modernih telekomunikacija

mentor:
prof. dr. Milan Bjelica

Beograd, maj 2023. godine

Sažetak

U ovom radu je opisana aplikacija za sažimanje novinskih članaka, rađena u programskom jeziku Python. Zasnovana je na obradi prirodnog jezika, pomoću biblioteke newspaper3k, dok je za grafički korisnički interfejs iskorištena je biblioteka tkinter.

Ključne riječi: Python, obrada prirodnog jezika, newspaper3k, tkinter

Sadržaj

Spisak slika

Spisak funkcija pomenutih u radu

Glava 1

Uvod

Ideja za projekat je nastala uočavanjem realnog problema obimnih i nekonciznih novinskih članaka o aktuelnim dešavanjima. Kroz razgovor sa prijateljima sam uvidio da ne postoji aplikacija na srpskom jeziku koja bi rezimirala zadate vijesti, time štedeći dragocjeno vrijeme čitaocima.

Nakon detaljnijeg istraživanja, naišao sam na izvanrednu Python biblioteku - newspaper3k. Njen autor je softverski inženjer Lucas Ou-Yang, koji je takođe odgovoran i za inicijalni algoritam za prikazivanje Instagram priča i pretragu, kao i za Snapchat-ovu stranicu za otkrivanje novih sadržaja. Inspirisan jednostavnošću biblioteke requests i uz pomoć lxml alata stvorio je veoma moćno oružje za obradu prirodnih jezika. Biblioteka u ovom trenutku podržava preko 30 svjetskih jezika među kojima su i srpski, hrtvatski i makedonski. Pored podržavanja prethodno pomenutih jezika, biblioteka obuhvata i sledeće karakteristike: okvir za višenitno preuzimanje članaka, identifikacija URL-ova online medijskih izvora, izvlačenje teksta i fotografija iz HTML-a, izvlačenje ključnih riječi, rezimea i autora iz zadatog teksta. Rad na jednoj od najpopularnijih biblioteka za pronalaženje informacija je i djelimično sponzorisan od strane kompanije Parse.ly, koju je najviše zaintrigiralo automatsko izdvajanje.

U nastavku ću biti prikazana osnovna primjena ove biblioteke, načini korištenja, kao i potencijalni prostor za napredak.

Glava 2

Obrada članaka - newspaper3k

Za prikaz rada ove biblioteke sam napravio aplikaciju za sažimanje novinskih članaka, te i klasu **MojClanak** u kojoj se nalazi polje klase **Article**, iz biblioteke newspaper3k, nad kojim se obavlja čitava manipulacija. Pored njega postoji i boolean polje iz klase tkinter „vise_clanaka", kao i polje „analiza" klase TextBlob iz istoimene biblioteke, o kojima će biti rečeno nešto više u nastavku.

Glavna funkcija u ovoj klasi jeste simbolično nazvana „obrada", koja izvršava sve neophodne radnje za pripremanje objekta klase **Article** za njegovu dalju manipulaciju:

```
1     def obrada(self, url: str) -> None:
2         if self.__vise_clanaka.get():
3             self.obrada_vise_clanaka(url)
4             return
5         self.__moj_clanak = Article(url)
6         self.__moj_clanak.download()
7         if self.__cirilica():
8             self.__moj_clanak.set_html(MojClanak.
                  transkripcija(self.__moj_clanak.html)
                  )
9         self.__moj_clanak.parse()
10        self.__moj_clanak.nlp()
11        self.__analiza = TextBlob(self.__moj_clanak.
              text)
```

Listing 2.1: Funkcija „obrada" iz klase MojClanak

2.1 Ekstrakcija veb podataka - funkcija „download”

Ova funkcija je zaslužna za preuzimanje HTML sadržaja sa zadatog linka. Kako autor navodi, nije je poželjno koristiti ukoliko vršimo asinhrono preuzimanje više članaka. Rješenje tog problema će biti obrađeno nešto kasnije.

```
1    def download(self, input_html=None, title=None,
        recursion_counter=0):
2        """Downloads the link's HTML content, don't
            use if you are batch async
3        downloading articles
4
5        recursion_counter (currently 1) stops
            refreshes that are potentially
6        infinite
7        """
8        if input_html is None:
9            parsed_url = urlparse(self.url)
10           if parsed_url.scheme == "file":
11               html = self._parse_scheme_file(
                    parsed_url.path)
12           else:
13               html = self._parse_scheme_http()
14
15           if html is None:
16               log.debug('Download failed on URL %s
                    because of %s' %
17                         (self.url, self.
                            download_exception_msg)
                            )
18               return
19       else:
20           html = input_html
21
22       if self.config.follow_meta_refresh:
23           meta_refresh_url = extract_meta_refresh(
                html)
24           if meta_refresh_url and
                recursion_counter < 1:
25               return self.download(
```

```
26                      input_html=network.get_html(
                            meta_refresh_url),
27                      recursion_counter=
                            recursion_counter + 1)
28
29          self.set_html(html)
30          self.set_title(title)
```

Listing 2.2: Funkcija „download” iz klase Article, iz biblioteke newspaper3k

Dakle, u linijama **8-19** se, ukoliko već nije zadat kao argument funkcije, preuzima HTML sadržaj i obrađuju potencijalne greške koje mogu nastati pri preuzimanju. Nakon toga se, u linijama **22-27** vodi računa o potencijalnom problemu koji može izazvati HTML tag - „meta refresh”. Ovaj tag može potencijalno kreirati beskonačne petlje, ukoliko se ne obradi na ovaj, ili neki drugi adekvatni način.

Kao što je na početku pomenuto, ukoliko želimo da asinhrono preuzimamo više članaka, potrebno je da to odradimo na neki drugačiji način. Autor je kao rješenje za ovaj problem ponudio funkciju „build”, koja vraća izvorni objekat, bez preuzimanja ili raščlanjivanja članaka:

```
1           def build(url='', dry=False, config=None, **
                kwargs) -> Source:
2               """Returns a constructed source object
                    without
3               downloading or parsing the articles
4               """
5               config = config or Configuration()
6               config = extend_config(config, kwargs)
7               url = url or ''
8               s = Source(url, config=config)
9               if not dry:
10                  s.build()
11              return s
```

Listing 2.3: Funkcija „build” iz biblioteke newspaper3k

Rješenje za obradu više članaka koje sam implementirao u klasi **MojClanak**, jeste funkcija koja za povratnu vrijednost ima listu URL-ova datih članaka.

```
1           @staticmethod
```

```
2         def obrada_vise_clanaka(url: str) -> List[
              str]:
3             return [article.url for article in
                  newspaper.build(url).articles]
```

Listing 2.4: Funkcija „obrada_vise_clanaka” iz klase MojClanak

Ukoliko pogledate implementaciju funkcije build, logično pitanje koje se postavlja jeste zašto kao povratnu vrijednost nisam vratio listu objekata klase **Article**, već URL-ove? Odgovor na ovo pitanje ću dati u glavi broj 3, **Grafički korisnički interfejs - tkinter**.

2.2 Raščlanjivanje

Nakon preuzimanja, a prije raščlanjivanja na podcjeline, u funkciji „obrada” vrši se provjera da li je određeni tekst pisan na srpskoj ćirilici, pošto je prilikom testiranja primjećeno da pravljenje rezimea ne radi kako treba u tom slučaju. Ukoliko jeste, vriši se transkripcija HTML sadržaja datog članka:

```
1     def __cirilica(self) -> bool:
2         if self.__moj_clanak is not None:
3             return any(char in MojClanak.
                  __MAPIRANA_AZBUKA.keys() for char in
                  self.__moj_clanak.html)
4         return False
```

Listing 2.5: Funkcija „cirilica” iz klase MojClanak

```
1         @staticmethod
2     def transkripcija(text: str) -> str:
3         return "".join(
4             MojClanak.__MAPIRANA_AZBUKA[char] if
                  char in MojClanak.__MAPIRANA_AZBUKA.
                  keys() else char for char in text)
```

Listing 2.6: Funkcija „transkripcija” iz klase MojClanak

Privatno polje „MAPIRANA_AZBUKA” predstavlja rječnik, u kome su ključevi mala i velika slova srpske azbuke, dok su njihove vrijednosti njihovi ekvivalenti u abecedi.

Na početku funkcije „parse” se poziva getter objekta klase **Parser** nad kojim se poziva funkcija fromstring koja između ostalog poziva istoimenu

funkciju iz biblioteke lxml, time pokušavajući da minimalno raščlani dio teksta, bez informacija o tome da li je to samo fragment ili dokument. Nakon toga se ažuriraju određeni podaci poput naslova članka, autora, jezika teksta, datuma, ključnih riječi i sl. Prije daljih obrada na tijelu dokumenta, vrši se čišćenje, odnosno poziva se funkcija „clean" nad objektom klase **DocumentCleaner**, gdje se kao parametar proslijeđuje sam dokument. Ova funkcija uklanja „class" atribute iz <body> taga. Nakon izvršenog čišćenja dokumenta, poziva se funkcija „calculate_best_node", koja je zadužena za identifikovanje najrelevantnijeg čvora. On sadrži primarni sadržaj iz datom HTML dokumenta, a određuje se korištenjem različitih mehanizama bodovanja. Zatim se iz najrelevantnijeg čvora izdvajaju video snimci, tekst članka, kao i slike. Na kraju funkcije se boolean atribut, koji određuje da li je raščlanjivanje uspješno izvršeno, postavlja na vrijednost **True**, te i oslobađaju resursi, što indikuje da je raščlanjivanje završeno.

Prilikom testiranja primjećeno je da postoje veb stranice koje nije moguće preuzeti, te dalje raščlaniti, niti korištenjem funkcije 2.4, niti funkcijom 2.2, jer se renderuje pomoću JavaScript-a. Rješenje za ovaj problem će biti razmatrano u nastavku.

2.3 Obrada prirodnog jezika

Ukoliko prvi put pozivamo funkciju „nlp" nad objektom klase Article, neophodno je da prije toga izvršimo komandu **nltk.download('punkt')**, koja preuzima paket „punkt" koji sadrži unaprijed obučene modele i podatke za tokenizaciju. Tokenizacija predstavlja proces dijeljenja teksta u pojedinačne riječi, sintagme, rečnice ili druge smislene grupacije koje se jednom riječju nazivaju tokeni.

Funkcija „nlp" u klasi **Article** izgleda ovako:

```
1     def nlp(self):
2         """Keyword extraction wrapper
3         """
4         self.throw_if_not_downloaded_verbose()
5         self.throw_if_not_parsed_verbose()
6
7         nlp.load_stopwords(self.config.get_language
              ())
8         text_keyws = list(nlp.keywords(self.text).
              keys())
```

```
9           title_keyws = list(nlp.keywords(self.title).
               keys())
10          keyws = list(set(title_keyws + text_keyws))
11          self.set_keywords(keyws)
12
13          max_sents = self.config.MAX_SUMMARY_SENT
14
15          summary_sents = nlp.summarize(title=self.
               title, text=self.text, max_sents=
               max_sents)
16          summary = '\n'.join(summary_sents)
17          self.set_summary(summary)
```

Listing 2.7: Funkcija „nlp" iz klase Article

Može se primjetiti da se na početku učitavaju stop riječi iz jezika koji smo dobili raščlanjivanjem preuzetog HTML fajla. Zatim se, pozivanjem funkcije „keywords" određuju ključne riječi u tekstu kao i u naslovu, od čega se kasnije pravi jedinstvena lista ključnih riječi. Potom se, uz pomoć funkcije „summarize" pravi sažetak datog teksta, u 5 rečenica, ukoliko se eksplicitno ne kaže drugačije.

Realizacija funkcije „summarize" u biblioteci newspaper3k izgleda ovako:

```
1           def summarize(url='', title='', text='',
               max_sents=5):
2               if not text or not title or max_sents <=
                    0:
3                   return []
4
5               summaries = []
6               sentences = split_sentences(text)
7               keys = keywords(text)
8               titleWords = split_words(title)
9
10              # Score sentences, and use the top 5 or
                   max_sents sentences
11              ranks = score(sentences, titleWords,
                   keys).most_common(max_sents)
12              for rank in ranks:
```

```
13                summaries.append(rank[0])
14            summaries.sort(key=lambda summary:
                  summary[0])
15            return [summary[1] for summary in
                  summaries]
```

Listing 2.8: Funkcija „summarize”

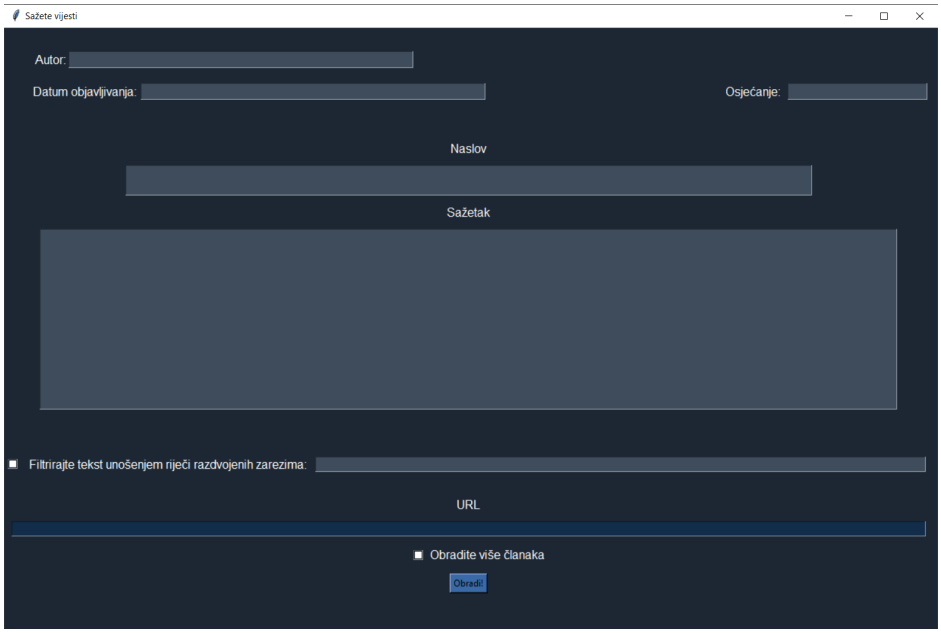
Samo rangiranje rečenica (funkcija „score”) se vrši na osnovu različitih parametara, poput povezanosti sa naslovom, učestanosti riječi iz te rečenice u ključnim riječima, dužini rečenice, kao i njenoj poziciji u tekstu.

Glava 3

Grafički korisnički interfejs - tkinter

Kao sponu između prethodno opisane biblioteke i korisnika korištena je biblioteka tkinter. Ona programeru daje oruđa za kreiranje korisničkih interfejsa za različite aplikacije i primjene.

Klasa koja koristi ovu biblioteku je, simbolično, nazvana GUI (Graphical User Interface). Nakon pozivanja funkcije „popuni_glavni_prozor", koja postavlja određene labele, tekstualna polja i dugmad na svoja mjesta, dobijamo ovakav izgled glavnog prozora:

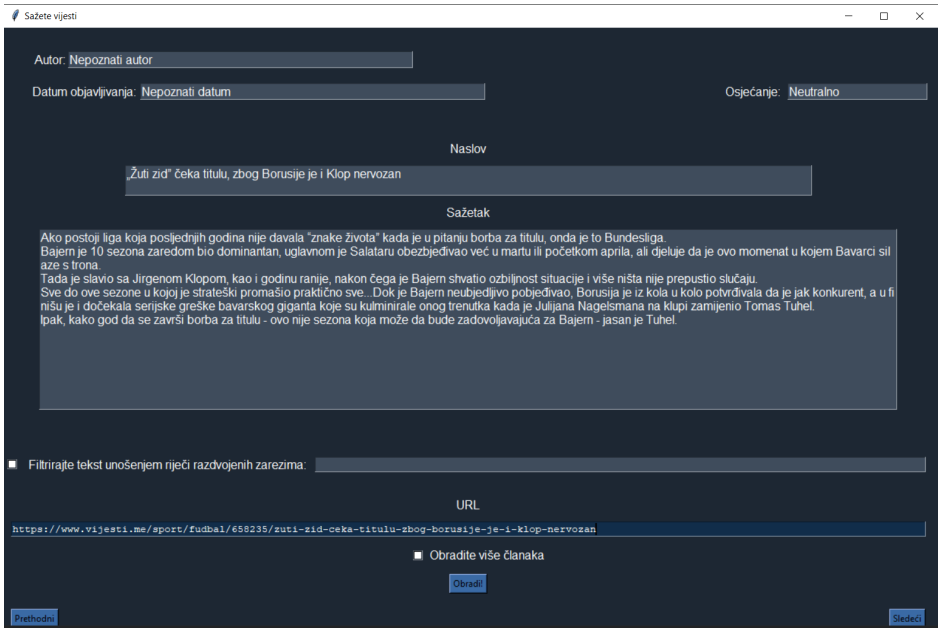


Slika 3.1: Prikaz grafičkog korisničkog interfejsa

Kao što se može primjetiti na slici 3.1, teget bojom su aktivna, polja u kojima može da se dodaje tekst, dok su sivom bojom označena polja koja će popuniti relevantne informacije iz članka koji želimo da obradimo. Izuzetak je polje za filtriranje, koje postaje teget pritiskom na polje za potvrdu, čime se omogućava unos riječi u njega, o čemu će biti riječi malo kasnije.

Unosom određenog URL-a, te pritiskom na dugme „Obradi!", započinje „lokalna_obrada" članka. Prvo se provjerava da li je unijeti string zapravo URL, pa ukoliko nije, u novom prozoru se ispisuje odgovarajuća poruka, kojom se korisnik obavještava o ishodu radnje. Pod uslovom da je unijet validan URL, nastavlja se sa obradom članka opisanom u Glavi 2, te ispisom informacija u odgovarajuća polja.

Pritiskom na polje za potvrdu „Obradite više članaka" poziva se funkcija 2.4, koja vraća listu URL-ova sa (feed-a) linkom zadatog članka ili sekcije. U funkciji „lokalna_obrada" u klasi GUI, uvijek se kreira nov objekat klase Article, pa se postavlja kao odgovarajuće polje u klasi MojClanak. Nakon toga se prikazuju relevantne informacije za taj članak. Pritiskom da dugme **Sledeći** ili **Prethodni**, prelazimo na sledeći, odnosno prethodni, članak u našoj listi preuzetoj sa inicijalno zadatog URL-a.



Slika 3.2: Prikaz jednog članka, prilikom obrade više njih

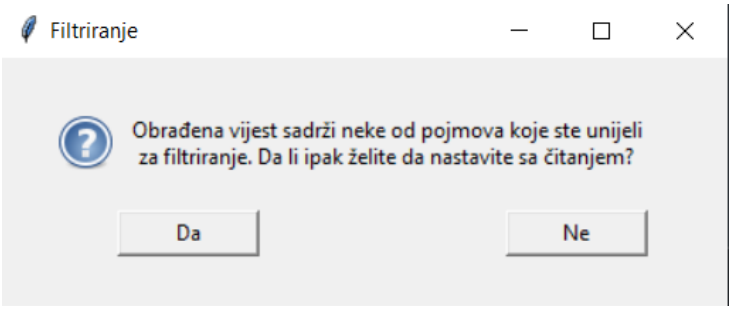
Glava 4

Specijalne opcije

Kako bi ova aplikacija bila potpunija, dodate su određene specijalne opcije. Naravno, pri daljem razvijanju aplikacije bi imalo prostora za napredak. Potencijalne ideje i načine izrade ću biti prokomentarisane u zaključku.

4.1 Filtriranje sadržaja

Jedna od opcija koje su dodate jeste filtriranje sadržaja. Svima nama nekada jednostavno bude previše informacija o određenoj temi, osobi ili događaju, te se trudimo da da izbjegavamo sadržaj vezan za istu, što više možemo. Kao što se može primjetiti na slici 3.1 unos u polje za filtriranje je onemogućen dok ne pritisnemo odgovarajuće polje za potvrdu. Pritiskom na njega, omogućava nam se upis, pa navodimo riječi, razdvojene zarezima, koje se kasnije pretražuju u samom tekstu članka preuzetog sa sajta. Ukoliko se pronađe poklapanje otvara se novi prozor sa ispisanom porukom da su pronađena poklapanja između teksta i unešenih riječi. Korisniku se tom prilikom postavlja pitanje da li želi da uprkos tome nastavi sa čitanjem članka ili ne.



Slika 4.1: Filtriranje teksta - prozor

Ukoliko se korisnik odluči da pritisne dugme „Da”, vijest se normalno ispisuje, prema prethodno opisanom formatu, dok ako se korisnik odluči za opciju „Ne” vraća se na početni prozor, bez ikakvog ispisivanja, gdje može ponovo unijeti URL vijesti koju želi da obradi.

4.2 Obrada više članaka

Druga opcija koja je dodata jeste obrada više članaka. Ako neki korisnik želi na brzinu da sažme i pročita više od jednog članka, ova opcija mu to omogućava. Tom prilikom (uz pomoć funkcije 2.4) se napravi lista vijesti, koje korisnik pritiskom na određenu dugmad može da obrađuje. Na slici 3.2 se može primjetiti da u donjem lijevom i donjem desnom uglu stoji opcija da se dohvati i obradi prethodni, odnosno sledeći članak u pomenutoj listi.

Glava 5

Zaključak

Kako se danas sve vrste korisničkih aplikacija trude da skrate vrijeme za koje njihov korisnik može da upije što veći broj različitih informacija. Mislim da je neophodno da postoji neka aplikacija na tržištu, koja bi rezimirala sadržaje novinskih članaka i dostavljala ih korisniku na korištenje, time štedeći njegovo dragocjeno vrijeme.

Za realizaciju jedne ovakve aplikacije može se koristiti opisana biblioteka - newspaper3k. Uz pomoć nje se mogu uspješno ekstrakovati, raščlaniti, pa i rezimirati veb podaci. Problem stranica koje se renderuju pomoću JavaScript-a se može prevazići kombinovanjem sa drugom Python bibliotekom pod nazivom BeautifulSoup, koja može uspješno da izvuče sadržaj koji možemo kasnije obraditi bibliotekom newspaper. Što se tiče grafičkog korisničkog interfejsa nije neophodno koristiti biblioteku tkinter. Moguće je korištenje njoj sličnih biblioteka poput PyQt, PySide, wxPython, Kivy i drugih.

Jedan od načina na koji se ova aplikacija može unaprijediti jeste dalje korištenje polja „analiza" iz klase **MojClanak**. Na primjer, može se realizovati filtracija samo pozitivnih, neutralnih ili samo negativnih članaka, čime se korisniku daje veći stepen slobode pri konzumiranju sadržaja. Polje „analiza" je objekat klase **TextBlob**, pa samim tim možemo da odredimo polaritet teksta koji mu je prosleđen u konstruktoru. Ukoliko korisnik želi da čita isključivo vijesti sa pozitivnom ili neutralnom konotacijom, neophodno je da polaritet bude veći ili jednak broju 0. Biblioteka **TextBlob** nam pruža jednostavan interfejs za mnoge uobičajene zadatke obrade prirodnog jezika i može biti korisno sredstvo za različite zadatke obrade i analize teksta. Čitalac se ohrabruje da detaljnije istraži pomenutu biblioteku.

Literatura

[1] Lucas Ou-Yang, *Newspaper3k: Article scraping & curation*, `https://newspaper.readthedocs.io/en/latest/`, pristupano maj 2023.

[2] Lucas Ou-Yang, *Lucas' Biography*, `https://codelucas.com/about/`, pristupano maj 2023.

[3] Lucas Ou-Yang, *Newspaper3k: Article scraping & curation*, GitHub repozitorijum, `https://github.com/codelucas/newspaper`, pristupano maj 2023.

[4] Python Software Foundation (PSF), *Graphical User Interfaces with Tk*, `https://docs.python.org/3/library/tk.html`, pristupano maj 2023.