Izborni projekt - Programiranje II

RJEČNIK RIJEČI HRVATSKE WIKIPEDIJE

Luka Illich

Sadržaj

UVOD

Cilj ovog projekta bio je prebrojavanje riječi sadržanih u člancima hrvatske wikipedije (https://hr.wikipedia.org/). Zbog opsežnosti, projekt sam podijelio u šest glavnih zadataka koje ću detaljno objasniti u narednim poglavljima. Za rješavanje problema koristio sam programske jezke C i Python te Bash ljusku za paralelizaciju pojedinih programa.

LISTA ČLANAKA

Istražujući na koji način izvršiti projekt, naišao sam na Python modul naziva *wikipediaapi*. Uz razne funkcionalnosti sadržavao je i mogućnost da prema zadanom naslovu samostalno preuzima članke izravno u tekstualnom formatu (bez HTML oznaka). Na poveznici https://dumps.wikimedia.org/hrwiki/20211220/ pronašao sam datoteku koja sadrži sve naslove hrvatske wikipedije (datoteka *hrwiki-20211220-all-titles*). Izvadak datoteke:

```
      page_namespace    page_title
      0     !
      0     !!!
      0     !Kuerschneria
      0     !Kung
      0     "Dear_Boss"_pismo
      0     "Deklaracija"_i_borba_za_hrvatsku_samostalnost
      0     "Enuma_Eliš"
(...)
      2   분당선M/common.css
      2   분당선M/common.js
      2   애콜라이트
      3   !Dilan
      3   !Silent
      3   !W!
```

Pošto su se u datoteci nalazili i naslovi koji se nisu odnosili na članke (članci --> *name_space* = 0) korištenjem jednostavne Python skripte (*extract_list_of_articles.py*) izradio sam novu datoteku u kojoj su se nalazili isključivo naslovi članaka. Datoteka *list_of_all_articles.txt* sadrži 258 984 naslova. Izvod:

```
      !
      !!!
      !Kuerschneria
      !Kung
      "Dear_Boss"_pismo
      "Deklaracija"_i_borba_za_hrvatsku_samostalnost
      "Enuma_Eliš"
      "From_Hell"_pismo
      "Glass_cockpit"_pilotska_kabina
      "Hello,_World!"
```

Listu sam naknadno podijelio u 8 datoteka kako bih paralelno mogao preuzimati više članaka i kako bi spriječio gašenje predugo otvorene veze sa poslužiteljem.

TEKST ČLANAKA

Korištenjem već spomenutog Python modula i njegove funkcionalnosti, izradio sam novu skriptu *download_text_of_articles.py* koja je iz zadane datoteke čitala naslove te slala upit za preuzimanje članka danog naslova.

```
import wikipediaapi
from datetime import datetime

wiki_wiki = wikipediaapi.Wikipedia(language='hr',
extract_format=wikipediaapi.ExtractFormat.WIKI)

input_file = input('Koristi listu: ')
output_file = input('Zapisi u: ')
log_file = input('Log-file: ')
(...)
f_read = open(input_file,'r', encoding='utf-8')
f_write = open(output_file, 'w', encoding='utf-8')
f_log = open(log_file, 'w', encoding='utf-8')
(...)
page = wiki_wiki.page(name)
f_write.write(page.text + '\n')
```

postavljanje modula za čitanje s hrvatske wikipedije

unos naziva datoteki za čitanje, pisanje i zapis log-fileova

varijabla *page* sprema stranicu članka te zapisuje samo tekstualni dio u izlaznu datoteku

Ovu skriptu sam paralelno pokretao na više računala sa zasebnim *lista*.txt* datotekama. (* zamjenjuje broj) Od dobivenih 8 novih text*.*txt* datoteka stvorio sam novu datoteku *TEXT_final.txt* koja je sadržavala sav tekst spomenutih *.txt* datoteka, odnosno sav tekst hrvatske wikipedije (korišteno: *merge_files.py*).

LISTA RIJEČI

Korištenjem C programskog jezika, izradio sam program koji je iz datoteke *TEXT_final.txt* čitao zasebno svaku riječ, sva slova u riječi prebacivao u mala slova te djelio riječ na mjestima na kojima bi se slučajno našao neki od znakova koji nije slovo ili broj (npr.: *. , ! ? +* - ....). Izvršavanjem ovog programa dobivena je datoteka *lista_rijeci_final.txt* s 89 937 040 linija (riječi).

```
FILE *text = fopen("TEXT_final.txt", "r");
FILE *output = fopen("lista_rijeci.txt", "w");
char buffer[SIZE];
if (text == NULL) {
      printf("Error! Could not open file\n");
      exit(-1);
}
while (fscanf(text, "%s", buffer) != EOF) {
      toLower(buffer);      //funkcija za pretvaranje u mala slova
      clearString(buffer); //funkcija za brisanje neželjenih znakova
      char *token = strtok(buffer, " ");
      while(token != NULL) {
            fprintf(output, "%s\n", token);
            token = strtok(NULL, " ");
      }
}
fclose(text);
fclose(output);
```

Ako nakon brisanja znakova postoji praznina unutar riječi, ovaj dio koda dijeli riječ na mjestima praznine („ “).

PODJELA LISTE

Kako bi ponovno mogao paralelizirati izvršavanje idućeg programa listu riječi sam podijelio u 90 datoteka (po milijun riječi). Datoteka *divide_list.c*:

```
FILE *input = fopen("lista_rijeci_final.txt", "r");
if (input == NULL) {
	printf("Error! Could not open file\n");
	exit(-1);
}
for (int i = 0; i < 90; ++i) {
	char dest[] = "lista";
	char num[8];
	sprintf(num, "%d", i);
	strcat(dest, num);
	strcat(dest, ".txt");
	FILE *output = fopen(dest, "w");
	char buffer[SIZE];
	int j = 0;
	while (fscanf(input, "%s", buffer) != EOF) {
		fprintf(output, "%s\n", buffer);
		j++;
		if (j >= LAST) break; //LAST = 1000000
	}
	fclose(output);
}
fclose(input);
```

Ovaj dio koda stvara novu datoteku različitog imena u svakoj novoj iteraciji.

Ovaj dio koda čita liniju po liniju iz ulazne datoteke i zapisuje liniju u trenutnu izlaznu datoteku sve dok ne zapiše 1000 000tu liniju.

PREBROJAVANJE RIJEČI LISTE

Novi C program je iz pojedine liste prebrojavao različite riječi i bilježio rezultate u zasebnu datoteku (i tako 90 datoteka). Pošto bi za obradu jedne datoteke od milijun riječi bilo potrebno otprilike 7 minuta, odnosno 10ak sati za sve riječi, napravio sam jednostavnu Bash scriptu koja pokrece po 5 instanci tog C programa te tako dobio obradu više listi odjednom, a koristio sam i više računala za još bržu izvedbu.

Bash:

```
#!/usr/bin/env bash
for i in {1..5}
do
	gnome-terminal -e "./statistic"
done
```

C: (datoteka *statistic.c*)

Za pohranu ulaznih podataka koristio sam strukturu u koju sam upisivao riječ i broj ponavljanja te riječi.

SPAJANJE I SORTIRANJE

Zadnji C program je iz 90 datoteka u memoriju učitao sve rezultate svih datoteka, pomoću *qsort()* funkcije sortirao polje struktura, spojio ponovljenje riječi te u datoteku *out.txt* zapisao završni popis riječi zajedno s njihovim brojem ponavljanja i postotkom u odnosu na ukupan broj.

```
typedef struct identity{
      char word[SIZE];
      int counter;
} identity;

int cmpF(const void *a, const void * b) {
      const identity *ia = a;
      const identity *ib = b;
      return strcmp(ia->word, ib->word);
}
(...)

qsort(id, inscribed, sizeof(identity), cmpF);
```

struktura naziva *identity*, sadrži riječ i brojač

funkcija koja uspoređuje dvije riječi

Funkcija *qsort()* kao argumente prima pokazivač na polje elemenata koje treba sortirati, broj elemenata, veličinu pojedinog elementa te naziv funkcije koja se koristi za usporedbu dva elementa.

REZULTATI

Analizom rezultata utvrdio sam da je pronađeno 1 551 295 različitih riječi. Naravno, to je puno veći broj od pretpostavki da hrvatski jezik ima oko 400 tis. riječi. Razlika je prisutna zbog brojeva, posebnih znakova (npr.: $\delta°$, π, $L\tau$, 1, …) te riječi iz drugih pisama (poput ćirilice, grčkog alfabeta, itd.) kao i raznih stranih imena. Njih nisam mogao predvidjeti, a također su ušli u analizu riječi.

Jedna od najčešćih riječi: „je“ se pojavljuje 3 099 282 puta, odnosno u 3.4460573752 % slučajeva.

Ispis s dna datoteke *out.txt*:

```
Pronađeno 1551295 različitih riječi!
Ukupan zbroj riječi: 89937040
```

Ukupan zbroj riječi predstavlja zbroj *countera* svih elemenata u strukturi *identity* te odgovata broju linija iz datoteke *lista_rijeci_final.txt*.

LITERATURA

- https://www.wikipedia.org/
- https://hr.wikipedia.org/
- https://dumps.wikimedia.org/
- https://dumps.wikimedia.org/hrwiki/20211220/
- https://stackoverflow.com/
- https://pypi.org/project/Wikipedia-API/
- https://en.cppreference.com/w/c/algorithm/qsort