

ŽILINSKÁ UNIVERZITA V ŽILINE

FAKULTA RIADENIA A INFORMATIKY

BAKALÁRSKA PRÁCA

EMANUEL ZAYMUS

Automatická rekonštrukcia diakritiky pre slovenčinu

Vedúci práce: Ing. Štefan Toth, PhD.

Registračné číslo: 28360720191584

Žilina, 2019

ŽILINSKÁ UNIVERZITA V ŽILINE, FAKULTA RIADENIA A INFORMATIKY.

ZADANIE TÉMY BAKALÁRSKEJ PRÁCE.

Študijný odbor : Informatika

Meno a priezvisko

Emanuel Zaymus

Osobné číslo

556771

Názov práce v slovenskom aj anglickom jazyku

Automatická rekonštrukcia diakritiky pre slovenčinu

Automatic diacritics restoration in Slovak text

Zadanie úlohy, ciele, pokyny pre vypracovanie

(Ak je málo miesta, použite opačnú stranu)

Cieľ bakalárskej práce:


Navrhnuť a vytvoriť algoritmus spolu s aplikáciou, ktorá skontroluje a automaticky správne opraví slová v texte bez diakritiky v slovenskom jazyku.


Obsah:

1. Analyzovať a porovnať existujúce algoritmy na rekonštrukciu diakritiky
2. Navrhnuť a implementovať vlastný algoritmus pre rekonštrukciu diakritiky v slovenskom texte
3. Otestovať algoritmus na vybraných vzorkách textu a porovnať ho s existujúcimi algoritmami
4. Navrhnuť, implementovať a otestovať aplikáciu na obnovu diakritiky v čistom texte a vo Worde

Meno a pracovisko vedúceho BP: Ing. Štefan Toth, PhD., KST, ŽU

Meno a pracovisko tutora BP:

4.2.19, 
vedúci katedry
(dátum a podpis)

Zadanie zaregistrované dňa 24.10.2018 pod číslom 584/2018 podpis 

Čestné vyhlásenie

Čestne prehlasujem, že som prácu vypracoval samostatne s využitím dostupnej literatúry a vlastných vedomostí. Všetky zdroje použité v bakalárskej práci som uviedol v súlade s predpismi.

Súhlasím so zverejnením práce a jej výsledkov.

V Žiline, dňa 23.4. 2019

.....
Meno Priezvisko

Pod'akovanie

Chcem sa úprimne poďakovať vedúcemu práce Ing. Štefanovi Tothovi, PhD., za jeho príjemný prístup a výbornú spoluprácu a vedenie pri tvorbe tejto bakalárskej práce.

ABSTRAKT V ŠTÁTNOM JAZYKU

ZAYMUS, Emanuel: *Automatická rekonštrukcia diakritiky pre slovenčinu*. [Bakalárska práca]. – Žilinská univerzita v Žiline. Fakulta riadenia a informatiky; Katedra softvérových technológií. – Vedúci: Ing. Štefan Toth, PhD. – Stupeň odbornej kvalifikácie: Študent v odbore Informatika. – FRI UNIZA v Žiline, 2019. 74 s.

Písanie bez diakritiky v slovenskom jazyku sa stalo veľmi bežným. Pre mnohých ľudí je písanie bez diakritiky jednoduchšie a pohodlnejšie. V tejto bakalárskej práci sa pozrieme na problémy dopĺňania diakritiky a aké programy súčasnosti existujú. Predstavíme vlastný algoritmus, ktorý následne implementujeme dvoma spôsobmi za pomoci pokročilej údajovej štruktúry Trie. Taktiež práca popisuje vytvorenie webovej aplikácie s príjemným používateľským rozhraním za pomoci ASP.NET Core framework-u. Otestujeme naše algoritmy na vytvorenej testovacej množine textov a porovnáme ich s existujúcimi programami. Zhodnotíme výsledky a vyvodíme záver.

Kľúčové slová: C#, diakritika, n-gram, rekonštrukcia diakritiky, slovenčina

ABSTRAKT V CUDZOM JAZYKU

Writing without diacritics became very common. For many people it is easier and more comfortable. In this bachelor thesis we will see the problems with diacritics reconstruction and what programs already exist. We will introduce our own algorithm which we will implement in two different ways using advanced data structure Trie. We will create web application with nice user interface using ASP.NET Core. We will test our algorithms on the set of test texts and compare them with the existing programs. We will evaluate the results and draw conclusions.

Key words: C#, Diacritics, Diacritics Reconstruction, N-gram, Slovak

Obsah

Zoznam obrázkov	8
Zoznam tabuliek	9
Zoznam grafov	10
Zoznam diagramov	11
Zoznam skratiek	12
Úvod	13
1 Uvedenie do problematiky a súčasný stav	14
1.1 Súčasný programy pre dopĺňanie diakritiky	15
1.2 Naše ciele	17
2 Údaje a ich čistenie	18
2.1 Zdroj údajov a ich štruktúra	18
2.2 Čistenie	22
2.2.1 Odstraňovanie n-gramov s príliš malým absolútnym výskytom	23
2.2.2 Odstránenie číslíc a neznámych znakov	24
2.2.3 Odstraňovanie chybných slov od danej frekvencie	25
2.2.4 Odstraňovanie príliš dlhých slov	27
2.3 Zhodnotenie vyčistenia dát	28
3 Vlastný algoritmus a jeho implementácia	31
3.1 Návrh algoritmu	31
3.2 Použité softvérové technológie a údajová štruktúra	34
3.3 Spôsoby implementácie	36
3.3.1 Údajová štruktúra v operačnej pamäti (in-memory)	36
3.3.2 Údajová štruktúra na pevnom disku	39
3.3.3 Pomocné procesy algoritmu	44
3.4 Popísanie výsledného softvéru	46
4 Testovanie a porovnanie algoritmov	48
4.1 Automatizácia testovania	48
4.2 Porovnanie	49
4.3 Zhrnutie	54
5 Výsledný softvér	56
5.1 Knižnica - DLL	56
5.2 Webová aplikácia	57
5.3 Testovač diakritiky	59
Záver	60
6 Zoznam použitej literatúry	61

Zoznam príloh	63
Prílohy	64
Príloha A: Parametre počítača, s ktorým sa vyvíjalo a testovalo	65
Príloha B: Zoznam testovacích textov	66
Príloha C: Podrobné výsledky testovania	69
Príloha D: Obsah DVD	74

Zoznam obrázkov

Obrázok 1 – Údajová štruktúra Trie	35
Obrázok 2 – Uloženie n-gramov do binárneho súboru	40
Obrázok 3 – Snímky obrazoviek – Pád aplikácie ŠSTATISTICKÝ DIAKRITIKOVAČ	53
Obrázok 4 – Snímky obrazovky – Vytvorená webová aplikácia.....	58

Zoznam tabuliek

Tabuľka 1 – Prvých 60 záznamov zo súboru všetkých slov	20
Tabuľka 2 – Informácie o veľkosti súborov a počte jednotlivých n-gramov	20
Tabuľka 3 – Prvých 20 záznamov zo súboru 2-gramov	21
Tabuľka 4 – Prvých 20 záznamov zo súboru 3-gramov	21
Tabuľka 5 – Prvých 20 záznamov zo súboru 4-gramov	21
Tabuľka 6 – Niekoľko posledných vybraných záznamov zo súboru s 1-gramami	22
Tabuľka 7 – Hranice pre absolútne početnosti n-gramov	23
Tabuľka 8 – Výsledky čistení, odstraňovanie n-gramov s malým absolútnym výskytom ..	23
Tabuľka 9 – Množina akceptovaných znakov	24
Tabuľka 10 – Výsledky čistenia, odstránenie číslic a neznámych znakov	24
Tabuľka 11 – Vzorky dát – odstránenie číslic a neznámych znakov	25
Tabuľka 12 – Príklady 1-gramov s 3 rovnakými znakmi sa sebou	26
Tabuľka 13 – Výsledky čistenia, odstraňovanie chybných slov od danej frekvencie	26
Tabuľka 14 – Príklady odstránených anomálií – chybné slová od danej frekvencie	27
Tabuľka 15 – Výsledky čistenia, odstraňovanie príliš dlhých slov	28
Tabuľka 16 – Príklady odstránených 1-gramov, odstraňovanie príliš dlhých slov	28
Tabuľka 17 – Vyčistenie n-gramov vzhľadom na pôvodné súbory	29
Tabuľka 18 – Percentuálne využitie pôvodných súborov	29
Tabuľka 19 – Percentuálne využitie n-gramov pre doplnenie diakritiky	37
Tabuľka 20 – Percentuálne využitie n-gramov väčších ako 1 pre doplnenie diakritiky	38
Tabuľka 21 – Maximálne početnosti skupín n-gramov pre Trie štruktúru v RAM	38
Tabuľka 22 – Maximálne početnosti skupín n-gramov v binárnom súbore	41
Tabuľka 23 – Štatistika testovania – Algoritmus s údajovou štruktúrou v RAM	49
Tabuľka 24 – Štatistika testovania – Algoritmus s údajovou štruktúrou na pevnom disku	49
Tabuľka 25 – Štatistika testovania – DIAKRITIK JÚĽŠ	50
Tabuľka 26 – Štatistika testovania – ŠTATISTICKÝ DIAKRITIKOVAČ	51
Tabuľka 27 – Štatistika testovania – DIAKRITIKA BRM	54

Zoznam grafov

Graf 1 – Porovnanie početností pôvodných a vyčistených n-gramov	30
Graf 2 – Percentuálne porovnanie početností vyčistených n-gramov	30
Graf 3 – Porovnanie programov – Zistené úspešností programov	55
Graf 4 – Porovnanie programov – Uvedené úspešností autormi programov	55

Zoznam diagramov

Diagram 1 – Diagram aktivít – Základný cyklus algoritmu	31
Diagram 2 – Diagram aktivít – Určenie diakritiky slova pomocou jeho okolia.....	33
Diagram 3 – Diagram aktivít – Implementácia určenia diakritiky	36
Diagram 4 – Diagram aktivít – Implementácia určenia diakritiky s Cache pamäťou	43
Diagram 5 – Diagram tried – Hlavná časť programu	46

Zoznam skratiek

API	Application Programming Interface (Aplikačné programovacie rozhranie)
ASCII	American Standard Code for Information Interchange.
ASP.NET	Active Server Pages - open-source framework pre vytváranie web-aplikácií
DLL	Dynamic-link Library
FIFO	First-in First-out
JSON	JavaScript Object Notation
JÚLŠ	Jazykovedný ústav Ľudovíta Štúra
MVC	Model-View-Controller
N-gram	Skupina slov, ktoré sa v jazyku vyskytujú spolu
RAM	Random-access memory
SAV	Slovenská akadémia vied
SMS	Short Message Service
SNK	Slovenský národný korpus
SSD	Solid-state drive
UML	Unified Modeling Language
UNICODE	Množina univerzálnych kódov znakov (Universal Coded Character Set)
URL	Uniform Resource Location
UTF-8	Unicode Transformation Format 8-bit

Úvod

V súčasnosti sa vyskytuje mnoho slovenských textov napísaných bez diakritiky a to najmä na internete. Ľudia zvyknú písať kvôli pohodliu správy bez diakritiky, a to najmä v súkromných listoch.

V minulosti bol problém s kódovaním písmen s diakritikou. Využívalo sa totiž len ASCII kódovanie, ktoré obsahovalo len 128 znakov (od 0 po 127), a to zahrňovalo: číslice, riadiace kódy, netlačiteľné znaky, špeciálne znaky a písmená len anglickej abecedy (to znamená bez diakritiky). Táto sada znakov však nepostačovala a preto bola skoro rozšírená o ďalších 128 znakov, a jej presný názov bol EASCII (Extended ASCII). Postupne vznikali rôzne verzie prídavných 128 znakov, až vzniklo viac ako dvesto verzií EASCII. U nás sa najviac využívala znaková sada Latin 2, ktorá bola zahrnutá vo verzií Windows 1250, obsahujúca latinské písmená s diakritickými znamienkami stredoeurópskych jazykov. Pri takýchto verziách sa často muselo manuálne meniť kódovanie, aby bol text čitateľný. Dnes máme k dispozícii už iné štandardy ako UNICODE či UTF-8 kódovanie. [1]

Často sa ignoruje diakritika aj pri písaní SMS správ. Každá SMS správa je obmedzená na maximum 160 7-bitových znakov. Písmená bez diakritiky patria do množiny 7-bitových znakov, no písmená s diakritikou sú neštandardné a patria do množín 8-bitových alebo 16-bitových znakov. Preto je každé písmeno s diakritikou započítané ako niekoľko písmen bez diakritiky. [2]

Preto sa chceme v tejto práci pozrieť, čo môžeme spraviť pre doplnenie správnej diakritiky do textu bez diakritiky. Spravíme malý náhľad do existujúcich programov na dopĺňanie diakritiky a predstavíme náš vlastný algoritmus a jeho implementáciu. Následne otestujeme navrhnutý algoritmus a porovnáme ho s existujúcimi programami.

1 Uvedenie do problematiky a súčasný stav

Diakritické znamienko – je rozlišovacia značka, ktorá sa píše v okolí písmena (v slovenčine zvyčajne nad písmeno) a môže meniť výslovnosť, prízvuk a niekedy aj význam celého slova. V písanej podobe slovenského jazyka poznáme štyri diakritické znamienka: dĺžeň, mäkčeň, dve bodky a vokáň.

Písmenká, pri ktorých sa používajú diakritické znamienka sú:

- ´ - dĺžeň: á, é, í, í, ó, í, ú, ý
- ˇ - mäkčeň: č, ď, ľ, ň, š, ť, ž
- ˆ - dve bodky: ä
- ^ - vokáň: ô

Dopĺňanie diakritiky do jednotlivých slov však nie je vždy úplne jednoznačné, pretože v jazyku sa často vyskytujú slová s viacerými správnymi možnosťami doplnenia diakritiky. Príklady takýchto slov sú:

- | | |
|--------------------|----------------------|
| • bol, bôľ | • máme, mame |
| • podľa, podlá | • večnú, vecnú |
| • bude, búde | • sľubne, sľubné |
| • boli, bolí, bôli | • vinica, viniča |
| • nás, náš | • kocka, kočka |
| • majú, máju | • koníckom, kónickom |
| • určite, určité | • platom, plátom |

Pri dopĺňaní diakritiky do slova „**uplne**“, by sme sa museli pozrieť na kontext, z ktorého slovo pochádza. Ak by pochádzalo z vety: „*Bol uplne vycerpany a preto ostal doma.*“, doplnili by sme diakritiku slova „**úplne**“, no ak by veta bola: „*Zamestnanci dostali uplne zadanie svojej prace.*“, určili by sme slovo „**úplné**“. Z toho nám vyplýva, že pre určenie správnej diakritiky je nevyhnutné pozrieť sa na slovo aj s jeho okolím, v ktorom sa nachádza.

1.1 Súčasné programy pre dopĺňanie diakritiky

V súčasnosti existujú rôzne dopĺňače diakritiky, pracujúce na rôznych princípoch. Spomenieme niektoré z nich.

- **DIAKRITIK JÚLŠ – nástroj pre rekonštrukciu diakritiky [3]**

Tento program vznikol na Jazykovednom ústave Ľudovíta Štúra Slovenskej akadémie vied v oddelení Slovenského národného korpusu. Bol sprístupnený 18.8. 2014 a je založený na využití jazykového modelu postaveného na veľkom korpuse textov slovenského jazyka. Pri jeho používaní je možnosť zvolenia akou metódou sa bude rekonštruovať diakritika textu. Na výber sú možnosti: first (výber prvej možnosti, ktorú nájde), random (výber náhodnej diakritiky pre každé slovo), naive (výber diakritiky, ktorá sa v jazyku vyskytuje najčastejšie), n-gram (výber diakritiky na základe okolia slova v dĺžke n slov, kde n môže byť 2, 3, 4, 5 alebo 6) a ďalšie. Chybovosť rekonštruovaného textu uvádzajú 0,21 %.

- **Dopĺňovač diakritiky ŠTATISTICKÝ DIAKRITIKOVAČ [4]**

Nástroj vznikol na Slovenskej technickej univerzite v Bratislave na Fakulte informatiky a informačných technológií ako bakalárska práca [5] J. Gederu. Funguje na základe vytvárania všetkých možných kombinácií zo správnych diakritických foriem n slov stojacich pri sebe. Po vytvorení všetkých kombinácií je im priradená pravdepodobnosť výskytu pomocou príslušných n-gramov a nakoniec vybraná možnosť s najlepšou pravdepodobnosťou. Viac nájdete na odkaze [5]. Algoritmus rekonštruje iba správne slová bez chýb a preklepov. Tiež sa neberú do úvahy vlastné podstatné mená ani skratky. Uvedená úspešnosť algoritmu je 98 %. Veľmi podobný algoritmus využíva aj JÚLŠ.

- **Dopĺňač diakritiky DIAKRITIKA BRM [6]**

Voľne dostupná webová aplikácia vytvorená R. Hraškom. Funguje na princípe dopĺňania slova s najčastejšou diakritickou formou. Je to najjednoduchší spôsob, nie sú použité žiadne sofistikovanejšie algoritmy alebo technológie, ani komplikovanejšie jazykové modely.

Ďalšie práce a články týkajúce sa tejto problematiky:

- *Automatické doplňování diakritiky pro slovenštinu (V. Žák)* [7] – Používa vytvorený model obsahujúci 1-gramy a 2-gramy s najväčšou pravdepodobnosťou výskytu v jazyku pre každé slovo. Dopĺňanie je založené na porovnaní pravdepodobností 1-gramu a 2-gramu a rozhodnutí podľa väčšej pravdepodobnosti. Uvedená úspešnosť: 98,39 % (testované na množine 819 slov).
- *Diacritics Restoration for Slovak Texts Using Deep Neural Networks (M. Šuppa)* [8] – Využitie neurónových sietí na problém rekonštrukcie diakritiky za pomoci datasetu slovenskej Wikipédie. Uvedená úspešnosť: 98,7 %.
- *Doplňování diakritiky pro mobilní zařízení (J. Železník)* [9] – Rekonštrukcia diakritiky pre zariadenia s obmedzenou výpočtovou kapacitou a pamäťou. Využitie Viterbiho algoritmu a Skrytého Markovovho modelu. Rozdeľuje slová na k-tice písmen, ktorým samostatne prideluje diakritiku podľa pravdepodobnosti a následne ich spája. Môžu vzniknúť aj neexistujúce slová. Úspešnosť najlepších výsledkov: 84,32 % pre slovenský text (5 807 slov), 73,43 % pre český text (8 501 slov).
- *Obnovení diakritiky v českém textu (J. Vrána)* [10] – Prehľadáva históriu použitých doplnení diakritiky zoradených podľa pravdepodobnosti a vyberá tú s najväčšou pravdepodobnosťou. Následne sa pozerá, či obmena niektorého variantu slova nezlepší celkovú úspešnosť. Využíva, taktiež, históriu desiatich najpravdepodobnejších viet. Po každej rekonštrukcii aktualizuje históriu použitých slov. Využíva aj veľké písmená, konce viet a čiastočne doplnenú diakritiku vstupného textu. Uvedená úspešnosť: 97,4 % (438 647 slov).
- *Unsupervised Spelling Correction for the Slovak Text (D. Hladek, J. Staš, J. Juhár)* [11] – Odborná práca zameraná na opravu pravopisných chýb všetkého druhu (nie len diakritiky). Hlavná myšlienka algoritmu je založená na hľadaní najlepšej postupnosti správnych slov pre nesprávne napísané slová zo vstupného textu. Využíva štatistické metódy, Skrytý Markovov model či Viterbiho algoritmus.

1.2 Naše ciele

Naším cieľom je navrhnuť a implementovať vlastný algoritmus pre doplnenie diakritiky. Vytvoríme verziu programu ako DLL knižnicu a aj ako webovú stránku s intuitívnym používateľským rozhraním. Našu vytvorenú aplikáciu porovnáme s inými programami pre doplnenie diakritiky a pozrieme sa na štatistické výsledky. Naša aplikácia bude tiež uspôsobená, aby mohla doplniť diakritiku priamo do súboru z programu Word.

2 Údaje a ich čistenie

V tejto kapitole sa pozrieme na dáta, ktoré budeme využívať a popíšeme ich štruktúru. Keďže náš algoritmus, ktorým popíšeme v nasledujúcej kapitole, bude založený na prehľadávaní veľkého množstva dát, je nevyhnutné najprv uviesť ich pôvod, štruktúru a problémy, ktoré prinášajú. Uvidíme, že vyčistenie týchto dát je takmer nevyhnutné.

2.1 Zdroj údajov a ich štruktúra

V našej aplikácii budeme využívať textové súbory voľne stiahnuteľné z internetovej stránky Slovenského národného korpusu (SNK) [12] Jazykovedného ústavu Ľudovíta Štúra (JÚLŠ) Slovenskej akadémie vied (SAV). SNK je elektronická databáza, ktorá obsahuje primárne slovenské slová z najrôznejších slovenských textov od roku 1955. Zahrňuje texty z mnohých žánrov, štýlov, vedeckých oblastí či regiónov. Slová v korpuse sú spolu s doplňujúcimi informáciami, ktoré sú získané pomocou špecializovaných vyhľadávacích nástrojov IRSTLM Toolkit [13]. Budeme využívať aktuálnu verziu korpusu z označením **prim-8.0**, ktorá bola sprístupnená 31.1.2018 s takmer 1,5 mld. tokenov (token je skupina/reťazec znakov nasledujúcich za sebou, pričom celý reťazec je oddelených od ostatných separátormi, t. j. zvyčajne medzerami, tabulátormi alebo novými riadkami). Využitie na vedecko-výskumné účely je bezplatné. K dispozícii sú aj staršie verzie SNK ako napríklad prim-7.0 (v rozsahu 1,25 mld. tokenov), prim-6.1 (830 mil. tokenov), a ďalšie.

Štruktúra korpusu prim-8.0

SNK prim-8.0 obsahuje niekoľko podkorpusov [14]:

- prim-8.0-public-all – všetky verejne prístupné texty SNK (71,10 % publicistické, 15,22 % umelecké, 8,51 % odborné, 5,17 % iné texty), 1 477 447 216 tokenov, 1 160 286 731 slov
- prim-8.0-public-sane – bez textov s nesprávnou diakritikou, pred roka 1955, z oblastí mimo Slovenska a z lingvistických časopisov (73,75 % publicistické, 16,33 % umelecké, 8,91 % odborné, 1,01 % iné texty), 1 368 990 447 tokenov, 1 076 309 519 slov
- prim-8.0-public-vyv – vyvážený podkorpus (33,33 % publicistické, 33,33 % umelecké, 33,33 % odborné texty), 377 138 077 tokenov, 297 524 160 slov
- prim-8.0-public-inf – podkorpus publicistických (informatívnych) textov, 1 009 613 215 tokenov, 791 376 893 slov

- prim-8.0-public-prf – podkorpus vedeckých, odborných a populárno-náučných textov, 121 926 591 tokenov, 96 084 340 slov
- prim-8.0-public-img – podkorpus umeleckých textov, 223 552 510 tokenov, 177 545 076 slov
- prim-8.0-public-sk – podkorpus pôvodných slovenských textov (81,24 % publicistické, 7,91 % umelecké, 9,53 % odborné, 1,32 % iné texty), 1 042 623 207 tokenov, 821 878 724 slov
- prim-8.0-public-img-sk – podkorpus pôvodných slovenských umeleckých textov, 82 503 983 tokenov, 65 627 003 slov
- r1955az1989-5.0 – osobitný korpus textov z rokov 1955 – 1989 (5,11 % publicistické, 75,73 % umelecké, 13,82 % odborné, 5,34 % iné texty), 83 631 422 tokenov, 66 825 217 slov
- a ďalšie štatistické a frekvenčné podkorporusy

Pre naše účely, sme vybrali najväčší podkorpus zo SNK prim-8.0, **prim-8.0-public-all**, ktorý sa následne delí na:

- prim-8.0-public-all-lemma-frequency.txt (základy slov s ich frekvenciou)
- prim-8.0-public-all-word_frequency.txt (slová s ich frekvenciami)
- prim-8.0-public-all-word_frequency_non_case_sensitive.txt (slová s ich frekvenciami bez ohľadu na veľké-malé písmená)
- prim-8.0-public-all-2-gramy.txt (2-gramy)
- prim-8.0-public-all-3-gramy.txt (3-gramy)
- prim-8.0-public-all-4-gramy.txt (4-gramy)

Vybrali sme slová s ignorovanými veľkými písmenami a ich frekvenciou a všetky n-gramy. (Čo je to n-gram popíšeme v nasledujúcom texte.)

Štruktúra súboru slov s ignorovanými veľkými písmenami a ich frekvenciou

Tento textový súbor (prim-8.0-public-all-word_frequency_non_case_sensitive.txt) obsahuje všetky jedinečné tokeny, kde sú zanedbané rozdiely veľkých písmen. Formát uloženia dát v súbore je: jedinečný token a jeho absolútna frekvencia oddelené tabulátorom, pričom každý záznam je na novom riadku. Jednotlivé riadky sú zoradené zostupne podľa frekvencie. Súbor má 5 033 773 riadkov, čo zodpovedá počtu jedinečných tokenov v súbore

a jeho veľkosť na disku je približne 65 572 KB (64,04 MB). Ako príklad uvidíme prvých 60 záznamov zo súboru v tabuľke Tabuľka 1.

Tabuľka 1 – Prvých 60 záznamov zo súboru všetkých slov

.	99073674	ako	6974924	ktoré	2751345
,	97115374	si	6685445	ich	2731889
a	33198366	som	5388442	vo	2718608
v	29173775	"	5070180	aby	2662891
sa	25144617	za	4630505	jeho	2614116
na	23760809	–	4457886	ktorý	2587551
je	11651884	po	4432395	pri	2585966
že	9991553	by	4244123	nie	2582616
:	9906823	ale	4240497	však	2577888
-	9653336	pre	4152559	len	2571234
s	9110005	?	4096243	2	2569101
z	8916864	už	3935317	bol	2149314
to	8775872	čo	3676155	podľa	2120039
aj	8637275	k	3656235	ho	2102575
o	8357444	sme	3339783	bude	2088075
do	7646978	1	3269632	3	2032384
)	7605351	keď	3259931	alebo	2015854
(7382987	od	3168030	jej	2013265
„	7309268	sú	2866139	i	1937747
“	7153758	tak	2810258	či	1906995

Štruktúra súborov n-gramov s ich frekvenciami

Ide o súbory: prim-8.0-public-all-2-gramy.txt, prim-8.0-public-all-3-gramy.txt, prim-8.0-public-all-4-gramy.txt. **N-gram** je skupina n slov, ktoré sa v jazyku vyskytujú spolu. Súbory obsahujú 2-gramy [bi-gram], 3-gramy [tri-gram] a 4-gramy [tetra-gram], to znamená dvojice, trojice a štvorice slov, ktoré sa vyskytujú spolu v slovenskom jazyku. (Analogicky 1-gram [uni-gram] bude samostatne stojace slovo.) Taktiež sú vytvorené s ignorovaním veľkých písmen. Formát uloženia dát v súbore je: absolútna frekvencia n-gramu a samotný n-gram oddelené medzerou, jednotlivé tokeny (slová) sú medzi sebou oddelené tabulátormi a každý záznam je na novom riadku. Jednotlivé riadky sú zoradené zostupne podľa frekvencie.

Tabuľka 2 – Informácie o veľkosti súborov a počte jednotlivých n-gramov

N-gramy	Počet riadkov / n-gramov	Veľkosť na disku (GB)
2-gramy	125 298 237	2,99
3-gramy	349 836 625	10,0
4-gramy	440 534 052	15,0

Pre názornosť uvedieme prvých 20 záznamov z každého súboru.

Tabuľka 3 – Prvých 20 záznamov zo súboru 2-gramov

2-gramy zo súboru: prim-8.0-public-all-2-gramy.txt			
1220137	je to	580999	keď sa
1150516	nie je	555947	aj v
840345	sa v	545514	ale aj
804801	sa na	536956	o tom
784260	v roku	524299	na slovensku
766982	som sa	479809	aby sa
725750	že sa	479393	sme sa
685646	a v	460229	v tomto
629275	by sa	458086	ako sa
587774	na to	452134	čo sa

Tabuľka 4 – Prvých 20 záznamov zo súboru 3-gramov

3-gramy zo súboru: prim-8.0-public-all-3-gramy.txt			
121232	na druhej strane	60480	a tak sa
106889	v tomto roku	57825	v banskej bystrici
98736	v súvislosti s	57443	by som sa
96029	že je to	55786	čo sa týka
92650	nie je to	55151	a ja som
91226	to nie je	52773	v spolupráci s
84797	v porovnaní s	52265	v prvom rade
75047	na rozdiel od	51020	že by sa
68647	v tom čase	48048	sa s ním
62899	v tomto prípade	47355	bez ohľadu na

Tabuľka 5 – Prvých 20 záznamov zo súboru 4-gramov

4-gramy zo súboru: prim-8.0-public-all-4-gramy.txt			
19952	bez ohľadu na to		
18566	sociálnych vecí a rodiny		
17830	informovala o tom agentúra		
16208	že to nie je		
15694	s láskou a úctou		
11900	v spišskej novej vsi		
11477	z času na čas		
10588	čo nás navždy opustil		
10433	zo dňa na deň		
10408	v znení neskorších predpisov		
9682	pre deti a mládež		
9355	strednej a východnej európy		
9191	s odvolaním sa na		
8930	a ja som sa		
8871	ale na druhej strane		
8738	a to aj napriek		
8634	v ten istý deň		

Keďže všetky súbory n-gramov sú vytvorené z rovnakej počiatočnej množiny údajov, je zrejmé, že budú obsahovať rovnaké znečistenie. Tieto znaky a chybné slová sa v jazyku síce vyskytujú, no pre nás by boli zaťažujúcim elementom pri prehľadávaní dát. Pri niektorých znakoch resp. písmenách z cudzích jazykov, ani neexistuje diakritická forma, ako pri písmenách slovenskej abecedy.

2.2.1 Odstraňovanie n-gramov s príliš malým absolútnym výskytom

Ešte predtým ako začneme čistiť súbory od záznamov resp. n-gramov s nežiadanými znakmi a číslicami, umelo zmenšíme veľkosť jednotlivých súborov na základe odstránenia n-gramov s príliš malým absolútnym výskytom. Hranicu frekvencie od akej už nebudeme považovať n-gramy za relevantné si určíme samy.

Tabuľka 7 – Hranice pre absolútne početnosti n-gramov

N-gramy	Hranica (absolútna početnosť, ktorú už nepovažujeme za relevantnú)
1-gramy	0
2-gramy	1
3-gramy	2
4-gramy	3

Tabuľka hovorí o tom, že napríklad pre 2-gramy, ktoré sa nenachádzajú v podkorpuse 2-gramov SNK aspoň 2-krát nepovažujeme za relevantné (hranica je 1). Toto čistenie nám odstráni drvivú väčšinu nezmyselných tokenov, no taktiež aj mnoho správnych n-gramov. Prihliadnuc však na to, že sme odstránili slovné spojenia, ktoré sa v celom SNK vyskytli iba 1x pre 2-gramy, 2x pre 3-gramy a 3x pre 4-gramy, predpokladáme, že podstatné jadro sme nepoškodili. Všimnime si, že pri tomto čistení sme 1-gramy vynechali (hranica je 0). 1-gramy považujeme všetky za relevantné.

Tabuľka 8 – Výsledky čistení, odstraňovanie n-gramov s malým absolútnym výskytom

N-gramy	Pôvodné n-gramy		Nové n-gramy		Rozdiel	
	Veľkosť (GB)	Početnosť	Veľkosť (GB)	Početnosť	Veľkosť (GB)	Početnosť
2	2.99	125 298 237	1.05	43 901 500	1.94	81 396 737
3	10.00	349 836 625	0.99	36 168 928	9.01	313 667 697
4	15.00	440 534 052	0.42	12 130 054	14.58	428 403 998

2.2.2 Odstránenie číslíc a neznámych znakov

Pri ďalšom čistení budeme používať už doteraz vyčistené súbory (z predchádzajúcej podkapitoly).

Ako sme už uviedli v úvode tejto kapitoly, n-gramy obsahujú množstvo tokenov, ktoré nemôžeme považovať za validné. Preto určíme množinu znakov, ktoré považujeme za validné, a budeme kontrolovať všetky n-gramy, či neobsahujú aj iné znaky ako nami akceptované. Vybraná množina znakov obsahuje všetky písmená zo slovenskej abecedy, českej abecedy a nemeckej abecedy. Prečo aj česká a nemecká abeceda? Pri hlbšom študovaní n-gramov sme zistili, že obsahujú množstvo v slovenčine často používaných českých a nemeckých výrazov. Takýchto n-gramov je dokonca niekoľko 10 000 a majú veľmi vysoké absolútne frekvencie. Preto aby sme nestratili vzácne informácie, začlenili sme české a nemecké písmená z abecedy do našej množiny.

Tabuľka 9 – Množina akceptovaných znakov

Množina akceptovaných znakov									
a	á	ä	b	c	č	d	d'	e	é
f	g	h	i	í	j	k	l	í	l'
m	n	ň	o	ó	ô	p	q	r	í
s	š	t	t'	u	ú	v	w	x	y
ý	z	ž	č	ř	ů	ö	ü	ß	ß
Modrá	písmená slovenskej abecedy								
Zelená	prídavok českej abecedy								
Červená	prídavok nemeckej abecedy								

Môžeme si všimnúť, že tabuľka neobsahuje veľké písmená. Pripomenieme, že súbory, ktoré používame, sú vytvorené bez prihliadania na veľké písmená, a preto môžeme našu množinu obmedziť len na malé písmená. Teraz sa pozrime na výsledky čistenia:

Tabuľka 10 – Výsledky čistenia, odstránenie číslíc a neznámych znakov

N-gramy	Pôvodné n-gramy		Nové n-gramy		Rozdiel	
	Veľkosť (MB)	Početnosť	Veľkosť (MB)	Početnosť	Veľkosť (MB)	Početnosť
1	64.00	5 033 773	60.55	4 731 821	3.45	301 952
2	1 081.23	43 901 500	1 044.91	42 036 569	36.32	1 864 931
3	1 023.47	36 168 928	984.69	34 520 363	38.78	1 648 565
4	428.15	12 130 054	404.97	12 078 445	23.18	51 609

Pozrime sa ako vyzerajú odstránené 1-gramy. Rozdelili sme ich do troch skupín: čísla, znaky s číslami, neakceptované znaky. Vzorky z týchto 1-gramov s najväčšími frekvenciami máme v tabuľke Tabuľka 11.

Tabuľka 11 – Vzorky dát – odstránenie číslíc a neznámych znakov

Čísla		Znaky s číslami		Neakceptované znaky (začiatok)		Neakceptované znaky (pokračovanie)	
1	3269632	m2	24551	.	99073674	ñp	527
2	2569101	3d	16366	,	97115374	ń	512
3	2032384	ta3	14498	:	9906823	¹⁸	510
0	1709480	f1	10084	-	9653336	¹⁹	508
4	1510520	d1	9846)	7605351	budućnost	500
5	1472258	v4	8001	(7382987	szűcs	499
0	1217408	o2	7874	„	7309268	²⁰	491
6	1201646	k1	7776	“	7153758	győri	485
10	1149307	m3	6649	–	4457886	ascânio	485
30	1041962	1x	6456	?	4096243	◆	484
7	942557	mp3	5882	!	1799939	iñigo	475
15	935302	3x3	5812	/	1716921	françoisa	474
20	928473	co2	5632	—	1481971	>	468
8	903919	2x	5612	;	851421	✓	465
9	866755	c1	5338	%	629177	²¹	464
12	798941	k2	5256	*	370808	(462
11	740449	5a	5245	–	341969	²²	453
17	689320	a4	5087	...	332755	vă	448
18	685891	r1	4882	-	254970	○	447
19	680376	2d	4775	+	247219	athénaïs	445
16	654843	11m	4449	”	215904	popović	436
14	629854	a1	4339	→	179507	◆	435
13	602071	3x	4049	•	163922	²³	434
21	477494	1b	3851	&	150311	hegedűsová	431
22	462772	c2	3713	=	141254	κ	430
25	457141	3b	3669	'	129403	stanisław	427
50	415175	2b	3370]	117136	ξ	426
23	382809	1a	3323	[115450		418
24	346662	u2	3209	@	108027	<	418
40	321037	g20	3023	\	100433	²⁴	415

2.2.3 Odstraňovanie chybných slov od danej frekvencie

Po odstránení n-gramov s príliš malou frekvenciou, n-gramov obsahujúcich číslice a nepoznané znaky, dáta stále obsahujú množstvo chybných dát, ktoré potrebujeme odstrániť. Sú to n-gramy obsahujúce nezmyselné slová a preklepy so zdvojenými či strojenými písmenami. Keďže v slovenských slovách sú bežné 2 rovnaké znaky za sebou,

budeme za chybné slová považovať také, čo majú 3 rovnaké znaky za sebou, pričom nebudeme hľadiť na diakritické znamienka. Uvedieme príklady takýchto 1-gramov v tabuľke Tabuľka 12.

Tabuľka 12 – Príklady 1-gramov s 3 rovnakými znakmi sa sebou

www	202270	pssst	434	iiia	226
iii	89284	xxviii	432	zzz	218
viii	8702	šss	371	xxxvi	210
ppp	8299	ááá	355	góóól	210
xiii	4535	xxxi	351	láááska	193
xxx	4462	jááánošíík	349	sss	190
aaa	3519	brrr	347	tááák	180
xxiii	3469	ooo	331	xxxviii	175
mmm	1933	nooo	301	xxxv	174
yyy	1923	xxxii	294	pomóóóc	174
xviii	1885	fff	292	vvv	171
bbb	1642	oops	283	nieee	171
hmmm	1376	xxiii	281	yyyy	170
azzz	1369	iee	278	supeer	170
sss	1019	ttt	277	niééé	168
sssr	872	xxxx	273	kkkk	167
sššr	569	lll	271	xxxvii	164
ddd	518	bzzz	253	booom	159
eee	464	kkk	252	hmmmm	158
ccc	457	xxxiv	233	áááá	158

V tabuľke však vidíme aj slová, ktoré by sme úplne nechceli vylúčiť. Sú to napríklad: „ááá“, „brrr“, „góóól“, „tááák“, „pomóóóc“, „niééé“ a iné. Preto sme na základe štúdia týchto slov určili hranicu, od ktorej budeme chybné slová odstraňovať. Táto hranica je určená najvyššou frekvenciou, od ktorej budeme jednotlivé n-gramy odstraňovať smerom nadol. Pre nás sa bude rovnať hodnote 11. Pozrime sa na výsledky vyčistenia:

Tabuľka 13 – Výsledky čistenia, odstraňovanie chybných slov od danej frekvencie

N-gramy	Pôvodné n-gramy		Nové n-gramy		Rozdiel	
	Veľkosť (MB)	Početnosť	Veľkosť (MB)	Početnosť	Veľkosť (MB)	Početnosť
1	60.55	4 731 821	60.32	4 716 624	0.23	15 197
2	1 044.91	42 036 569	1 044.66	42 024 550	0.25	12 019
3	984.69	34 520 363	984.52	34 513 840	0.17	6 523
4	404.97	12 078 445	404.86	12 075 320	0.11	3 125

Odstránili sme aj takéto anomálie:

Tabuľka 14 – Príklady odstránených anomálií – chybné slová od danej frekvencie

[illegible]

2.2.4 Odstraňovanie príliš dlhých slov

Naše súbory stále obsahujú dáta, ktoré zatiaľ vyhovujú podmienkam, ale sú pre nás nežiadané. Napríklad slová, ktoré majú niekedy aj viac ako 200 znakov. Zvyčajne sú to celé vety alebo súvetia bez medzier. Preto sme si usporiadali všetky n-gramy vzostupne podľa ich dĺžky a po ich prezretí sme určili maximálnu dĺžku slova 30 znakov, ktorú bude môcť slovo nadobúdať. Slová nad 30 znakov odstránime.

Tabuľka 15 – Výsledky čistenia, odstraňovanie príliš dlhých slov

N-gramy	Pôvodné n-gramy		Nové n-gramy		Rozdiel	
	Veľkosť (MB)	Početnosť	Veľkosť (MB)	Početnosť	Veľkosť (MB)	Početnosť
1	60.32	4 716 624	60.28	4 715 688	0.04	936
2	1 044.66	42 024 550	1 044.656	42 024 522	0.004	28
3	984.52	34 513 840	984.516	34 513 831	0.004	9
4	404.86	12 075 320	404.856	12 075 312	0.004	8

Príklady odstránených 1-gramov:

Tabuľka 16 – Príklady odstránených 1-gramov, odstraňovanie príliš dlhých slov

sivernýmfanúšikomzvolenskéhohokeja	12
tichémaléstvoreniektoréradofantazíruje	7
rádpovzbudzuješvojtímpriamoztribúny	7
najnevykryštalizovávateľnejšieho	7
najnerozkrasokorčuľovateľnejšieho	7
supercalifragilisticexpialidocious	6
projektprezentovalizástupcoviaslovenskejlekárskejkomory	6
najneobhospodarovávateľnejšieho	6
hematologickotransfuziologickom	6
štyridsaťdvatisícosemstodvadsať	5
oznamovaciapovinnosťpredržiťapovolenianapredajspotrebiteľského	5
poznášhistóriuisúčasnosťhkmzvolen	4
hematologickotransfuziologického	4
zisteniestavuspracovaniaziadostireq	3
teoretickoumeleckointerpretačné	3
bpputclesfkglbqqzondrxscbfimrkpvqs	3
xoxoxoxoxoxoxoxoxoxoxoxoxoxoxoxo	2
vkrasňanochavračínávštevnicizaplatiamenej	2
tyranosaurusbrachiosaurusdiplodocusallosaurusstegosaurus	2
transfervýchovnéodstupnéregistračné	2

2.3 Zhodnotenie vyčistenia dát

Na prvý pohľad sa nám môže zdať, že sme dáta previedli až príliš prísnymi čistiacími cyklami, no opak je pravdou. Z vyčistených súborov budeme v ďalších kapitolách vytvárať konečnú údajovú štruktúru, kde uvidíme, že v niektorých prípadoch budeme mať až neúnosne veľa údajov, z ktorých budeme musieť vyberať len tie najrelevantnejšie.

Vyčistenie stále nie je úplne dokonalé, no ďalšie cykly by si už vyžadovali ručné prezeranie jednotlivých dát, čo pri našom počte n-gramov nie je možné. Súborny stále obsahujú nekorektné, chybné či úplne nesprávne slová, no v našom prípade to pre nás nemusí byť úplne nevýhoda, pretože ľudia často píšú texty s chybami, nesprávnymi či nespisovnými slovami, navyše aj s gramatickými chybami. A mi budeme chcieť rekonštruovať diakritiku aj v týchto slovách do miery akej to bude možné.

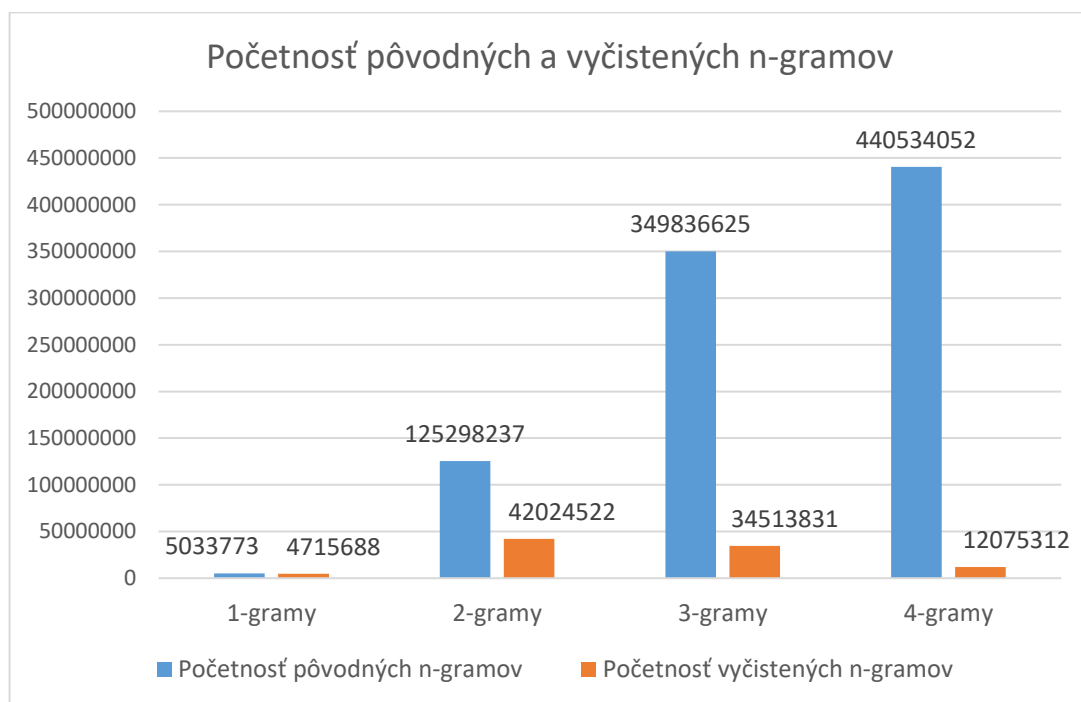
Pozrime sa na posledné tabuľky v tejto kapitole, ktoré nám ukážu ako sme prečistili n-gramy vzhľadom na úplne pôvodné súborny.

Tabuľka 17 – Vyčistenie n-gramov vzhľadom na pôvodné súborny

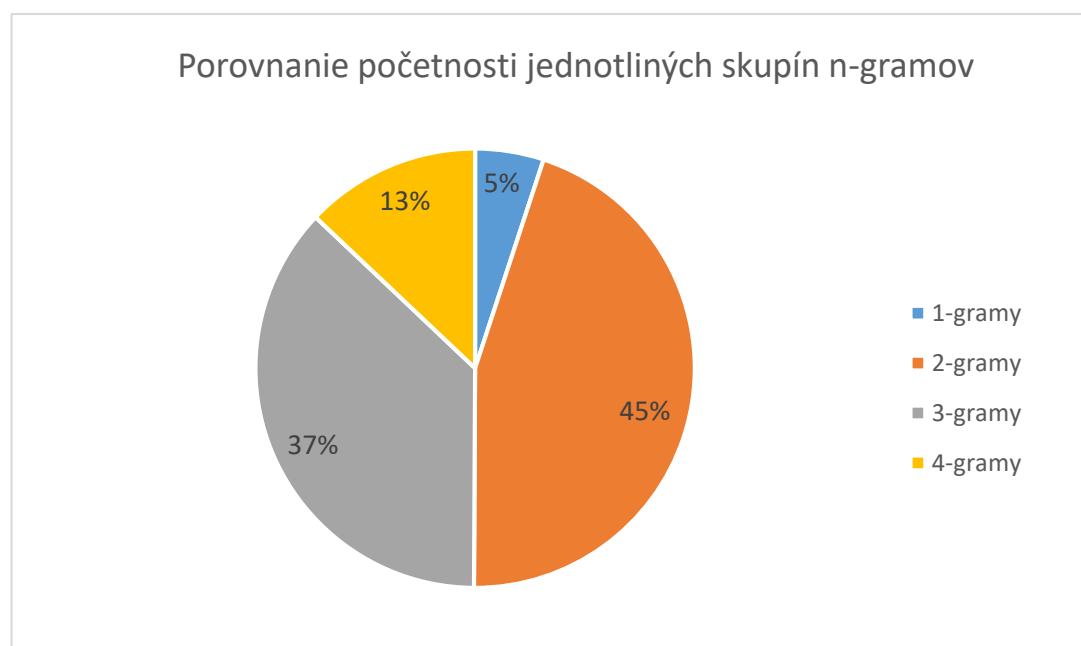
N-gramy	Pôvodné n-gramy		Nové n-gramy		Rozdiel	
	Veľkosť (MB)	Početnosť	Veľkosť (MB)	Početnosť	Veľkosť (MB)	Početnosť
1	64.04	5 033 773	60.28	4 715 688	3.76	318 085
2	3 062.20	125 298 237	1 044.656	42 024 522	2 017.544	83 273 715
3	10 267.70	349 836 625	984.516	34 513 831	9 283.184	315 322 794
4	15 372.78	440 534 052	404.856	12 075 312	14 967.924	428 458 740

Tabuľka 18 – Percentuálne využitie pôvodných súborov

N-gramy	Početnosť pôvodných n-gramov	Početnosť vyčistených n-gramov	Percentuálne využitie pôvodných súborov
1	5 033 773	4 715 688	93.68 %
2	125 298 237	42 024 522	33.54 %
3	349 836 625	34 513 831	9.87 %
4	440 534 052	12 075 312	2.74 %



Graf 1 – Porovnanie početností pôvodných a vyčistených n-gramov



Graf 2 – Percentuálne porovnanie početností vyčistených n-gramov

3 Vlastný algoritmus a jeho implementácia

Po predstavení a vyčistení súborov, ktoré budeme používať, pre dopĺňanie diakritiky, uvedieme hlavnú myšlienku nášho algoritmu a jeho implementáciu.

3.1 Návrh algoritmu

Našu konkrétnu implementáciu predstavíme až v podkapitole 3.3 Spôsoby implementácie. Zdôrazňujeme, že v tejto podkapitole uvedieme iba všeobecnú myšlienku algoritmu. Naša implementácia sa bude v niektorých detailoch líšiť.

Náš algoritmus je založený na porovnávaní okolia slova, ktorému chceme doplniť diakritiku, s n-gramami prislúchajúcimi tomuto slovu. Za okolie slova považujeme 3 slová pred a 3 slová za týmto slovom v kontexte, ak tieto slová samozrejme existujú.

Pozrime sa na zjednodušený UML diagram aktivít (Diagram 1), zobrazujúci základný cyklus algoritmu. Potom algoritmus podrobnejšie popíšeme.

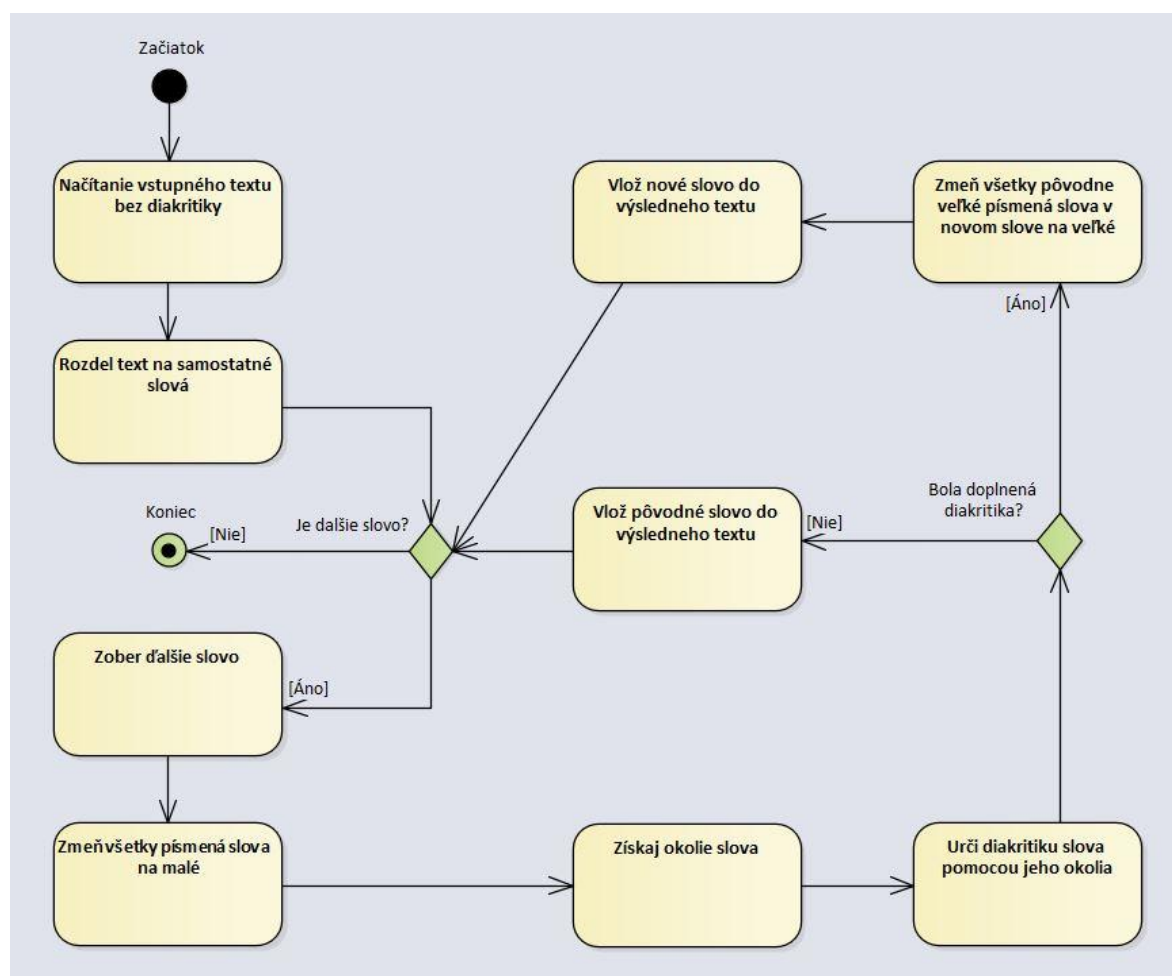


Diagram 1 – Diagram aktivít – Základný cyklus algoritmu

Po načítaní vstupného textu bez diakritiky, čo je text ktorému chceme rekonštruovať diakritiku, rozdelíme text na samostatné slová. Pri tomto kroku treba myslieť na to, že v našom výslednom texte chceme zachovať všetky interpunkčné, biele a iné znaky, ktoré používateľ zadá. Teraz budeme postupne spracovávať všetky slová. Ak máme ešte nejaké slovo, zoberieme ho a znormalizujeme ho (t. j. prevedieme ho do normovanej formy, ktorú budeme vedieť spracovávať), teda všetky veľké písmená prevedieme na malé. Robíme to z toho dôvodu, že všetky slová v n-gramoch sú iba s malými písmenami, čo zjednodušuje prácu a výrazne šetrí pamäť (ak by sme mali pracovať s dátami, kde sa navyše rozlišuje medzi malými a veľkými písmenami, spracovávanie a prehľadávanie takýchto dát by bolo omnoho zložitejšie). Následne získame okolie slova. Pri porovnávaní budeme používať maximálne 4-gramy, čo sú štvorice slov. Ak by sa prvé slovo v 4-grame rovnalo nášmu dopĺňanému slovu, na porovnanie potrebujeme 3 nasledujúce slová za našim slovom. Ak by sa posledné slovo v 4-grame rovnalo nášmu slovu, na porovnanie potrebujeme 3 slová pred našim slovom. Preto za okolie slova považujeme 3 slová pred a 3 slová za týmto slovom v kontexte, samozrejme ak tieto slová existujú. Potom určíme diakritiku slova (samotným určením diakritiky sa budeme zaoberať v nasledujúcej samostatnej časti). Ak sme úspešne doplnili diakritiku, zmeníme v novom slove malé písmená na veľké, tak ako boli v pôvodnom slove bez diakritiky a zapíšeme ho do výsledného textu. Ak sme však úspešný neboli, do výsledného textu doplníme pôvodné slovo, ktorému nepotrebujeme meniť malé písmená na veľké. Pri vkladaní slov do výsledného textu si opäť potrebujeme dať pozor na správne vkladanie interpunkčných, bielych a iných znakov, aby sme používateľovi zachovali pôvodný formát textu. Následne, ak máme ďalšie slovo, začíname jeho spracovaním. Ak nie, náš algoritmus končí.

Určenie diakritiky slova pomocou jeho okolia

Teraz popíšeme samotný algoritmus pre určovanie diakritiky. Pozrime sa, opäť, najskôr na príslušný diagram (Diagram 2).

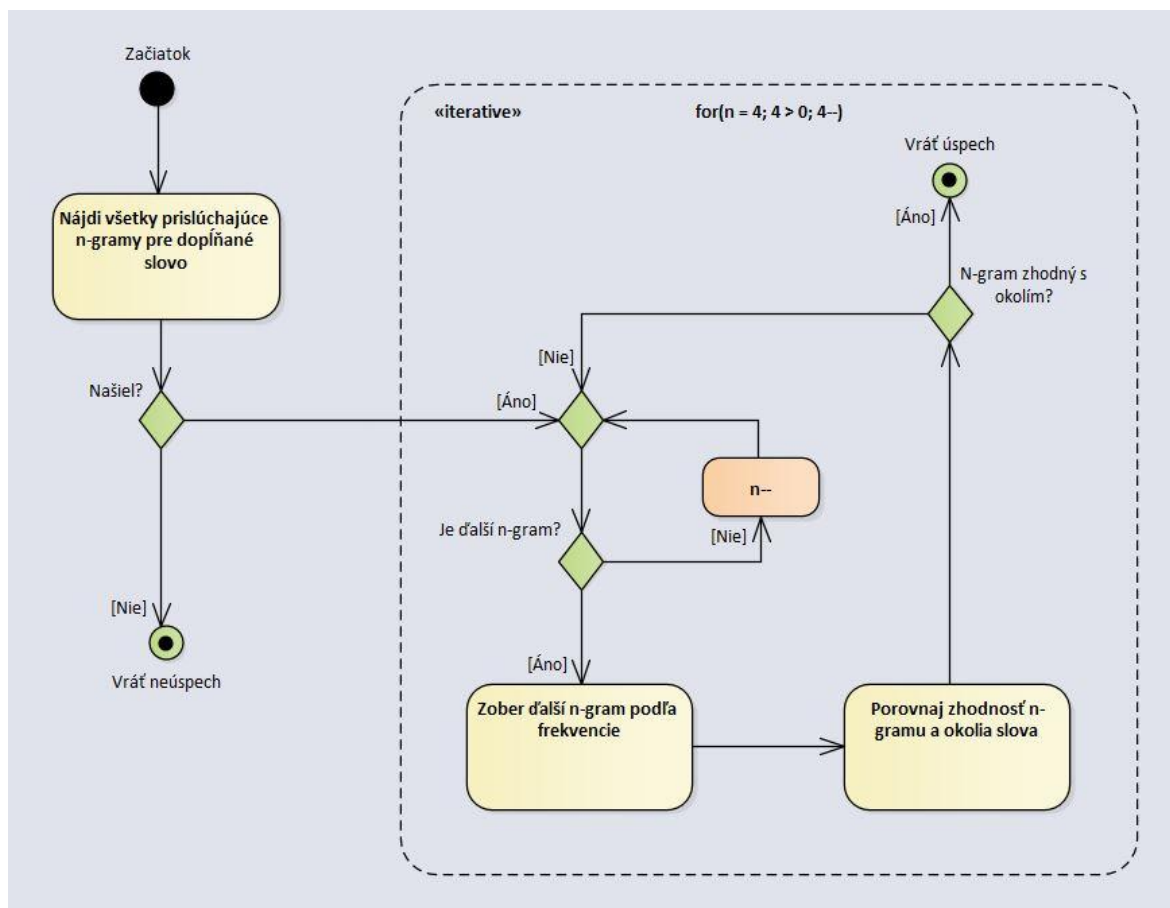


Diagram 2 – Diagram aktivít – Určenie diakritiky slova pomocou jeho okolia

V tomto algoritme predpokladáme, že máme presne určené slovo pre doplnenie diakritiky a jeho okolie. Ako prvé, nájdeme všetky prislúchajúce n-gramy pre dopĺňané slovo. Prislúchajúcimi n-gramami máme na mysli každý n-gram, v ktorom aspoň jedno z n slov, zodpovedá nášmu dopĺňanému slovu (samozrejme, ak pred ich porovnaním odstránime diakritiku slovu z n-gramu). Ak nenájdeme žiaden takýto n-gram, končíme neúspechom, to znamená, že do daného slova sme nedoplnili žiadnu diakritiku. Ak nájdeme aspoň jeden takýto n-gram, môžeme pristúpiť ku ich postupnému spracovávaní. V uvedenom diagrame vidíme *for*-cyklus *for* ($n = 4; n > 0; n--$), ktorý hovorí o tom, že budeme postupne spracovávať skupiny prislúchajúcich n-gramov a to postupne od 4-gramov až po 1-gramy. Ak už v aktuálnej skupine nemáme žiadny n-gram, dekrementujeme n . Tým chceme povedať, že prejdeme zo skupiny n-gramov na skupinu $(n - 1)$ -gramov (napríklad zo skupiny 4-gramov na skupinu 3-gramov). Takto môžeme postupovať maximálne po $n = 1$. Ak máme v aktuálnej skupine ešte nejaký n-gram, zoberieme ten, ktorý má najvyššiu absolútnu frekvenciu. V n-grame nájdeme, kde sa vyskytuje naše dopĺňané slovo (musíme myslieť na to, že hľadané slovo sa v n-grame môže vyskytovať viac-krát), a okolité slová n-gramu

porovnáme s okolím dopĺňaného slova (samozrejme, pred zisťovaním ich zhodnosti, odstránime diakritiku z n-gramu). Ak sme našli zhodu, to znamená, že okolie slova súhlasí s n-gramom, môžeme vrátiť úspech (doplníme do slova diakritiku podľa nájdeného slova z n-gramu). Ak nie, pozrieme sa, či máme ešte nejaký n-gram v aktuálnej skupine n-gramov. Môžeme si všimnúť skutočnosť, že ak vstúpime do *for-cyklu*, tak naše jediné možné ukončenie je vrátením úspechu. Je to preto, lebo predpokladáme, že v skupine n-gramov, ktorá prislúcha danému slovu, je nevyhnutne aspoň jeden 1-gram. Táto podmienka zabezpečí, že ak cyklus prejde na porovnávanie 1-gramov, tak hneď pri prvom 1-grame, cyklus vráti úspech, pretože 1-gram, nemá žiadne okolie, a teda zhoda je istá.

Prečo sme zvolili takýto postup pri prehľadávaní príslušných n-gramov? Veríme, že takýmto prístupom doplníme diakritiku najlepším možným spôsobom, v rámci našich možností. Ak nájdeme zhodu okolia slova s n-gramom s vyšším n , považujeme ju za relevantnejšiu ako zhodu s nižším n , aj keby tento n-gram mal väčšiu absolútnu frekvenciu. To je dôvod, prečo prehľadáваме najprv všetky 4-gramy podľa frekvencie a až potom všetky 3-gramy podľa frekvencie (atď.).

3.2 Použité softvérové technológie a údajová štruktúra

Ešte pre samotnou implementáciou si v tejto podkapitole uvedieme, aké softvérové technológie a údajové štruktúry budeme používať.

Programovací jazyk C#

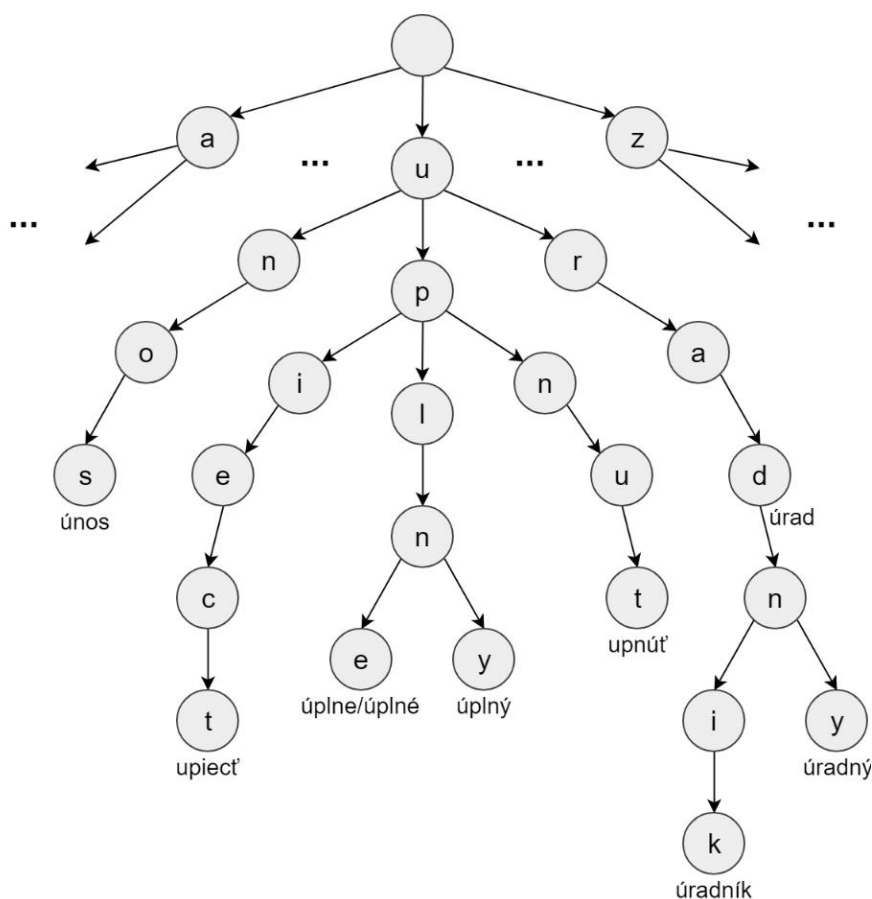
Celú prácu programujeme v objektovom jazyku C#, kvôli jeho pohodlnému písaniu, čitateľnosti a momentálnej popularite medzi programátormi. Tiež je vždy veľmi dobré, naučiť sa niečo nové. Tento jazyk ponúka veľmi jednoduché rozšírenie pomocou frameworkov, ktoré poskytuje priamo spoločnosť Microsoft, ako napríklad ASP.NET Core. Je vhodný na tvorbu webových aplikácií a my ho budeme v našej práci využívať tiež. Vývojové prostredie Microsoft Visual Studio, v ktorom vyvíjame program, taktiež ponúka veľmi jednoduché rozšírenie o knižnice od Microsoft-u, .NET-u a tretích strán (takzvané NuGet Packages), priamo z prostredia programu.

Údajová štruktúra Trie

Aby sme v našej aplikácii mohli rýchlo prehľadávať všetky údaje prislúchajúce ku každému slovu, potrebujeme mať veľmi rýchly prístup všade do pamäte podľa kľúča, ktorým

je práve dané slovo. Na tento prípad je úplne ideálna prehľadavacia údajová štruktúra Trie (Retrieval, Písmenový strom alebo Prefixový strom).

Ide o prehľadavací k-cestný strom, kde sú kľúčmi zvyčajne reťazce (tak, ako aj v našom prípade). Jednotlivé vrcholy sú reprezentované časťami kľúča (v našom prípade to sú teda samostatné znaky - písmená). Kmeňom je prázdny reťazec a listy stromu predstavujú nami uložené dáta. Pre lepšiu názornosť sa pozrime na nasledujúci obrázok.



Obrázok 1 – Údajová štruktúra Trie

Trie má výbornú zložitosť operácie nájdi prvok $O(n)$, kde n sa rovná počtu znakov v reťazci. Túto výhodu údajovej štruktúry Trie využijeme naplno pri hľadaní n -gramov k príslušnému slovu.

V programe využijeme voľne stiahnutelnú hotovú generickú štruktúru `PBCD.DataStructures.Trie` [15] (od autora: Pouya Bisadi) stiahnutú do projektu ako NuGet balíček.

3.3 Spôsob implementácie

V tejto podkapitole konkrétne popíšeme dve rôzne implementácie navrhnutého algoritmu. Uvidíme, že je potrebné robiť kompromisy medzi rýchlosťou a presnosťou (úspešnosťou) algoritmu, preto sme sa rozhodli implementovať dva rôzne spôsoby, kde jeden bude zameraný na rýchlosť (3.3.1 Údajová štruktúra v operačnej pamäti (in-memory)) a druhý bude zameraný na úspešnosť (3.3.2 Údajová štruktúra na pevnom disku). Vždy bude rozhodujúce, ako a kde uložíme dáta (n-gramy), ktoré budeme chcieť prehľadávať.

3.3.1 Údajová štruktúra v operačnej pamäti (in-memory)

Implementácia údajovou štruktúrou v operačnej pamäti je veľmi jednoduchá. Ideme podľa návrhu algoritmu ako je uvedené v kapitole 3.1, pričom keď prideme ku aktivite *Nájdí všetky prislúchajúce n-gramy pre dopĺňané slovo* v diagrame Diagram 2, prichádza na rad spomínaná prehľadávacia štruktúra Trie z predchádzajúcej podkapitoly 3.2. Zvolili sme ten najjednoduchší prístup – každé slovo bude mať uložené príslušné n-gramy vo vrchole Trie-stromu. Takto zabezpečíme skoro okamžitý prístup ku všetkým potrebným n-gramom. Navyše, formát uloženia dát vo vrcholoch môžeme prispôbiť tak, aby sme vôbec nemuseli vykonávať *for-cyklus*, ani vyhľadávanie ďalšieho n-gramu z aktivity *Zober ďalší n-gram podľa frekvencie* z diagramu Diagram 2. N-gramy uložíme do zoznamu zoradené podľa frekvencie zostupne tak, že ako prvé budú v zozname všetky príslušné 4-gramy, potom 3-gramy, 2-gramy a nakoniec 1-gramy. Ak budeme mať v pamäti takto uložené n-gramy, nemusíme vôbec držať informácie o ich frekvenciách (čo nám ušetrí pamäť). Diagram algoritmu sa zmení nasledovne:

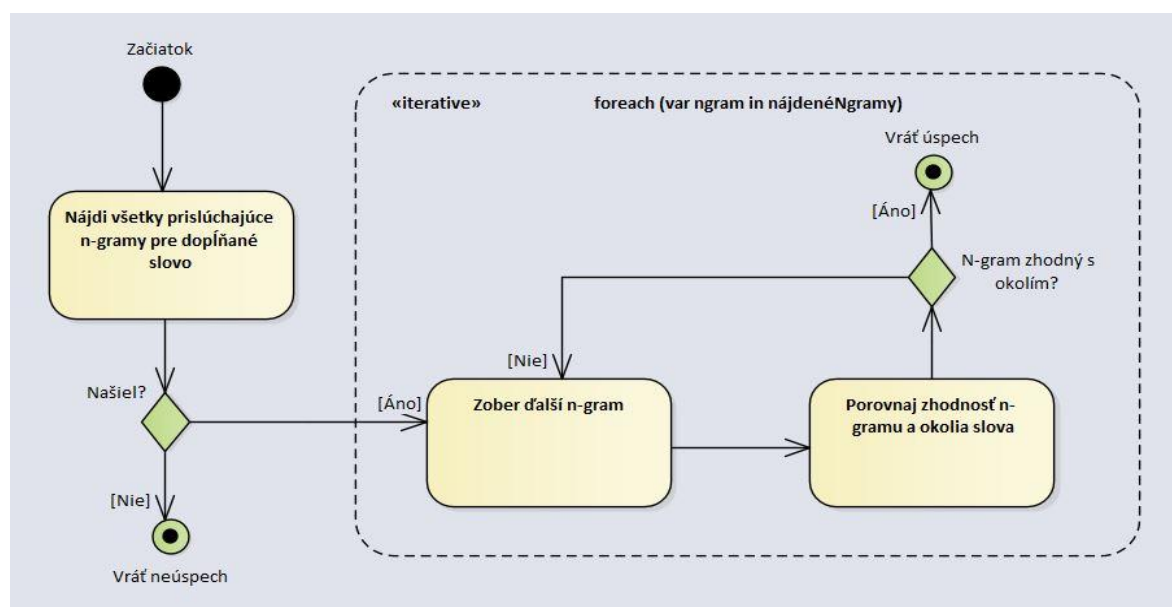


Diagram 3 – Diagram aktivít – Implementácia určenia diakritiky

Optimalizácia štruktúry Trie

Veľkosť a zároveň aj rýchlosť štruktúry Trie, môžeme bez akejkoľvek straty úspešnosti dopĺňania diakritiky optimalizovať nasledovne:

1. V štruktúre napríklad vôbec nemusia byť slová, ktoré neobsahujú žiadnu diakritiku. Ak sa tieto slová v Trie-i nenájdu, diakritika sa nedoplní (rovnako by sa nedoplnila, ak by sa aj slovo v Trie-i našlo). Príklady takýchto slov sú: „roh“, „lampa“, „obloha“, atď.
2. Ak má slovo iba jednu diakritickú formu (existuje práve jedno zmysluplné doplnenie diakritiky), nemusíme sa pri ňom rozhodovať pomocou prehľadávania a porovnávania jeho okolia a príslušných n-gramov. Stačí ak bude v pamäti uložený iba jeden jeho 1-gram, ktorý sa pri nájdení hneď doplní. Príklady takýchto slov sú: „akcionári“, „vzájomnom“, „ovzdušie“, „veľvyslanec“, atď.
3. Ak algoritmus príde k 1-gramom, tak sa vždy doplní prvý z nich, pretože 1-gramy nemajú žiadne okolie a teda nemôže prebiehať jeho porovnanie. Preto môžeme pre každé slovo uložiť iba jeden jeho 1-gram a to ten, ktorý má najväčšiu absolútnu frekvenciu.

Obmedzenie štruktúry Trie

Program vyvíjame a testujeme na počítači s veľkosťou RAM pamäte 8 GB (ostatné parametre sú uvedené v prílohe). Preto by sme radi udržali požiadavky programu na pamäť v rozmedzí 4 až 5 GB. Keďže program potrebuje množstvo pamäte na réžiu algoritmu a Trie štruktúry, všetky dáta sa do pamäte „nezmestia“. Z našich skúšobných pokusov s menším počtom použitých n-gramov (vykonávaných na skupine testovacích textoch) sme zistili, že skupiny n-gramov sa vždy využívajú na určenie diakritiky v pomerne presných pomeroch.

Tabuľka 19 – Percentuálne využitie n-gramov pre doplnenie diakritiky

N-gramy	4-gramy	3-gramy	2-gramy	1-gramy
Percentuálne využitie pre doplnenie diakritiky	10 %	24 %	37 %	29 %

V uvedenej tabuľke je možné si všimnúť, že zo všetkých slov z testovacích textov, ktorým bola doplnená diakritika pomocou n-gramov, sa 10 % doplnilo 4-gramami, 24 % 3-gramami, 37 % 2-gramami a 29 % 1-gramami. Tieto poznatky využijeme pre rozumné obmedzenie výslednej Trie štruktúry.

Keďže pre každé slovo máme práve jeden 1-gram, nemôžeme skupinu 1-gramov viac obmedziť. Uvedieme tabuľku využitia skupín n-gramov väčších ako 1 pre doplnenie diakritiky.

Tabuľka 20 – Percentuálne využitie n-gramov väčších ako 1 pre doplnenie diakritiky

N-gramy	4-gramy	3-gramy	2-gramy
Percentuálne využitie pre doplnenie diakritiky	14 %	34 %	52 %

Tabuľka hovorí o tom, že zo všetkých slov z testovacích textov, ktorým bola doplnená diakritika pomocou n-gramov väčších ako 1, sa 14 % doplnilo 4-gramami, 34 % 3-gramami a 52 % 2-gramami. Pre jednoduchšie výpočty zaokrúhlime tieto rozdelenia na **15 %, 35 % a 50 %**.

Ďalej sme pokusmi zistili, že ak chceme zachovať požadovanú veľkosť v pamäti RAM (4-5 GB), musíme obmedziť maximálny počet n-gramov pre každé slovo, a to na počet 700 n-gramov (konkrétne výsledky aj s veľkosťou v RAM predstavíme v kapitole 4.2 Porovnanie). Ak sme pri pokusoch zvolili maximálny počet n-gramov väčší ako 700, program nadobúdala v RAM pamäti hodnoty väčšie ako 5 GB. Týchto 700 povolených n-gramov rozdelíme pre každú skupinu n-gramov podľa jej percentuálneho využitia. Do 700 n-gramov neuvažujeme jeden 1-gram.

Tabuľka 21 – Maximálne početnosti skupín n-gramov pre Trie štruktúru v RAM

N-gramy	4-gramy	3-gramy	2-gramy	1-gramy
Pomer rozdelenia max. počtu n-gramov	15 %	35 %	50 %	-
Maximálna početnosť	105	245	350	1

Ak by slovo nemalo toľko n-gramov v skupine, necháme v štruktúre všetky.

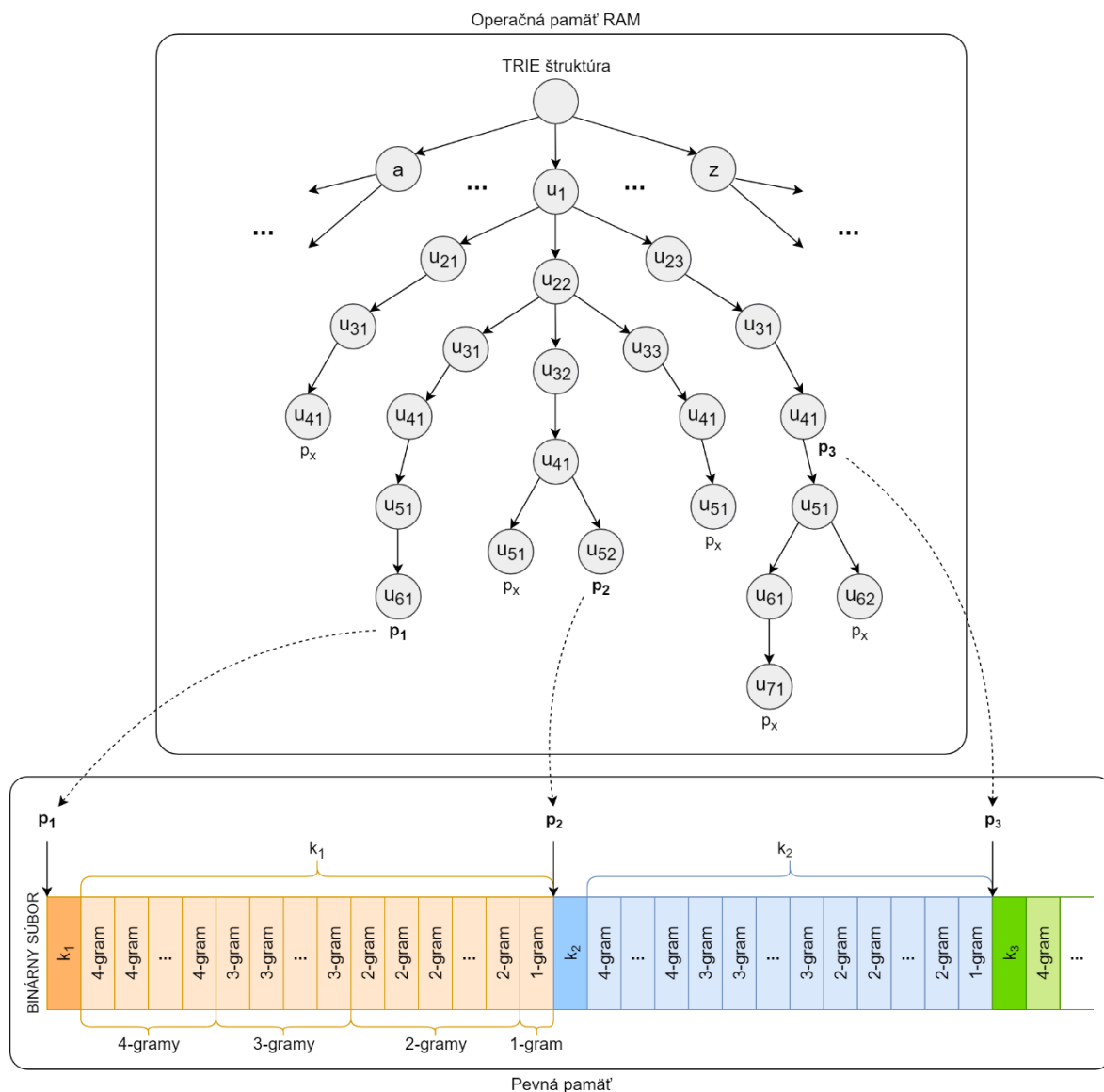
Maximálna zistená veľkosť v RAM po vykonaní rekonštrukcie diakritiky na testovacích textoch bola 4,3 GB. Podrobné výsledky testovania budú uvedené v kapitole 4 Testovanie a porovnanie algoritmov.

3.3.2 Údajová štruktúra na pevnom disku

V predchádzajúcej podkapitole sme boli výrazne obmedzený veľkosťou RAM pamäte, a preto by sme radi vyskúšali implementovať navrhnutý algoritmus s využitím pevnej pamäte (hard-disku) na ukladanie prehľadávaných dát. Pri Trie štruktúre v RAM pamäti sme museli určiť hornú hranicu počtu n-gramov (700), no teraz by sme chceli túto hranicu zvýšiť (nie zrušiť – v tejto kapitole uvedieme dôvody).

Otázkou je, ako budeme ukladať údaje (n-gramy) na pevný disk, tak aby sme k nim mali čo najrýchlejší prístup. Z hardvérového hľadiska uvedieme, že máme k dispozícii SSD SAMSUNG MZNLN256MHQ-000H1 (úplná špecifikácia je uvedená v prílohe). Potrebovali by sme minimalizovať dĺžku času prístupu na disk, resp. čítanie z disku. V ideálnom prípade, by sme chceli mať všetky potrebné dáta (n-gramy) pre každé slovo uložené na jednom pamäťovom mieste na disku, aby nám stačilo jeden raz pristúpiť na daný pamäťový blok a načítať všetky potrebné údaje naraz. Podobne ako pri Trie štruktúre v operačnej pamäti sme mali všetky požadované n-gramy uložené v jej vrcholoch (viac v kapitole 3.3.1 Údajová štruktúra v operačnej pamäti (in-memory)). Preto by sme sa tejto štruktúre chceli čo najviac priblížiť.

Budeme využívať štruktúru Trie, tak ako v predchádzajúcej podkapitole, s tým rozdielom, že vo vrcholoch nebudeme mať uložené samotné dáta, ale iba pozíciu (index) do nami vytvoreného binárneho súboru, odkiaľ budeme potrebné dáta čítať. Pozrime sa na nasledujúci obrázok.



Obrázok 2 – Uloženie n-gramov do binárneho súboru

Ako sme už spomenuli, vo vrcholoch Trie štruktúry, ktorá bude v RAM pamäti, budeme mať uloženú práve jednu pozíciu p_x do binárneho súboru. Takto budeme môcť priamo pristúpiť na presnú pozíciu v súbore, odkiaľ ako prvé prečítame počet príslušných n-gramov k_i , ktoré následne prečítame. Počet n-gramov k_i je 4-bajtové číslo (*int*) a jednotlivé n-gramy sú reťazce znakov (*string*). Všimnime si, že v binárnom súbore sú n-gramy usporiadané tak, ako v Trie štruktúre v operačnej pamäti (n-gramy sú usporiadané zostupne podľa frekvencie a zoskupené od 4-gramov až po 1-gram). Z toho vyplýva, že môžeme použiť zjednodušený

algoritmus

z predchádzajúcej

kapitoly

(

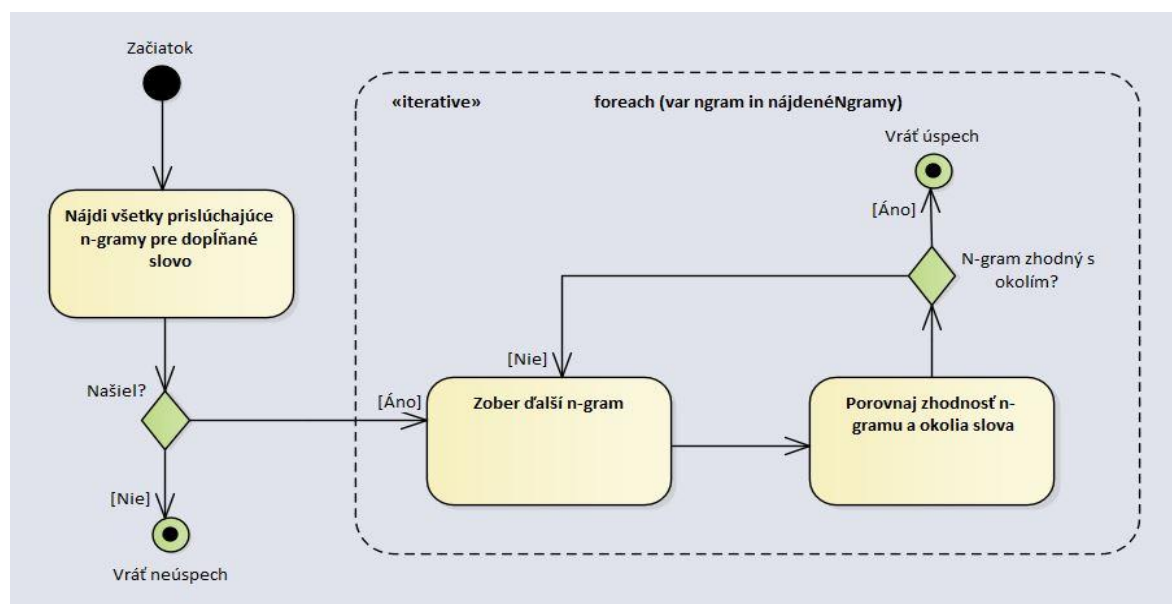


Diagram 3 – Diagram aktivít – Implementácia určenia diakritiky). Binárny súbor tiež optimalizujeme presne podľa postupu *Optimalizácia štruktúry Trie* z kapitoly 3.3.1, s tým rozdielom, že Trie nahradíme binárnym súborom.

Obmedzenie binárneho súboru

Keďže na disku máme dostatočný priestor, sme schopný uložiť celý súbor bez väčších problémov. Pri testovacích pokusoch sme však zistili, že rekonštrukcia diakritiky je príliš pomalá. Dôvodom boli často vyskytujúce sa slová, ktoré mali niekedy až niekoľko miliónov n-gramov, ktoré sa museli porovnávať. Po dôkladnom štúdiu slov a ich n-gramov sme zistili, že počet n-gramov pre jedno slovo (ak zanedbáme extrémne príliš frekventovaných slov a nefrekventovaných slov) sa najčastejšie pohybuje v rozmedzí 30 000-40 000. Preto sme stanovili hornú hranicu, 40 000, pre maximálny počet n-gramov pre jedno slovo (podobne ako horná hranica 700 pre Trie v RAM). Samozrejme, rozdeľujeme túto hranicu v pomere využitia skupín n-gramov podľa tabuľky – Tabuľka 20 – Percentuálne využitie n-gramov väčších ako 1 pre doplnenie diakritiky.

Tabuľka 22 – Maximálne početnosti skupín n-gramov v binárnom súbore

N-gramy	4-gramy	3-gramy	2-gramy	1-gramy
Pomer rozdelenia max. počtu n-gramov	15 %	35 %	50 %	-
Maximálna početnosť	6 000	14 000	20 000	1

Týmto obmedzením sme zrýchlili rekonštrukciu jedného slova z 0,27s na 0,009s, pričom sme úspešnosť správneho doplnenia diakritiky príliš neznížili (všetky výsledky finálneho testovania uvedieme v kapitole 4 Testovanie a porovnanie algoritmov).

Pomocná Cache pamäť

Podme sa však pozrieť, ako by sme mohli ešte zrýchliť náš algoritmus bez toho, aby sme neznížili úspešnosť rekonštrukcie diakritiky.

Vychádzajme z predpokladu, že v rámci jedného textu, sa často vyskytujú rovnaké slová v rovnakom alebo podobnom kontexte. Preto by bolo vhodné aby sme vyhladané n-gramy ukladali do nejakého buffra resp. cache pamäte, aby sme k nim mali rýchlejší prístup keď opäť narazíme na tieto slová. Táto cache pamäť musí byť obmedzená, pretože ak by sme chceli rekonštruovať príliš veľký text, operačná pamäť by sa nám iba plnila (vôbec by sa nevyprázdňovala) a mohlo by prísť ku pádu aplikácie kvôli nedostatku pamäte. Aby sme dosiahli rýchlu cache pamäť, využijeme na jej implementáciu Trie štruktúru, spolu s obyčajným zoznamom (List), ktorý bude slúžiť ako pomocný objekt Trie štruktúry na jej réžiu. Cache obmedzíme na posledných 1000 vyhladaných slov s ich použitými n-gramami. V spomínanom zozname budeme držať tieto slová. Algoritmus bude teraz vyzeráť nasledovne:

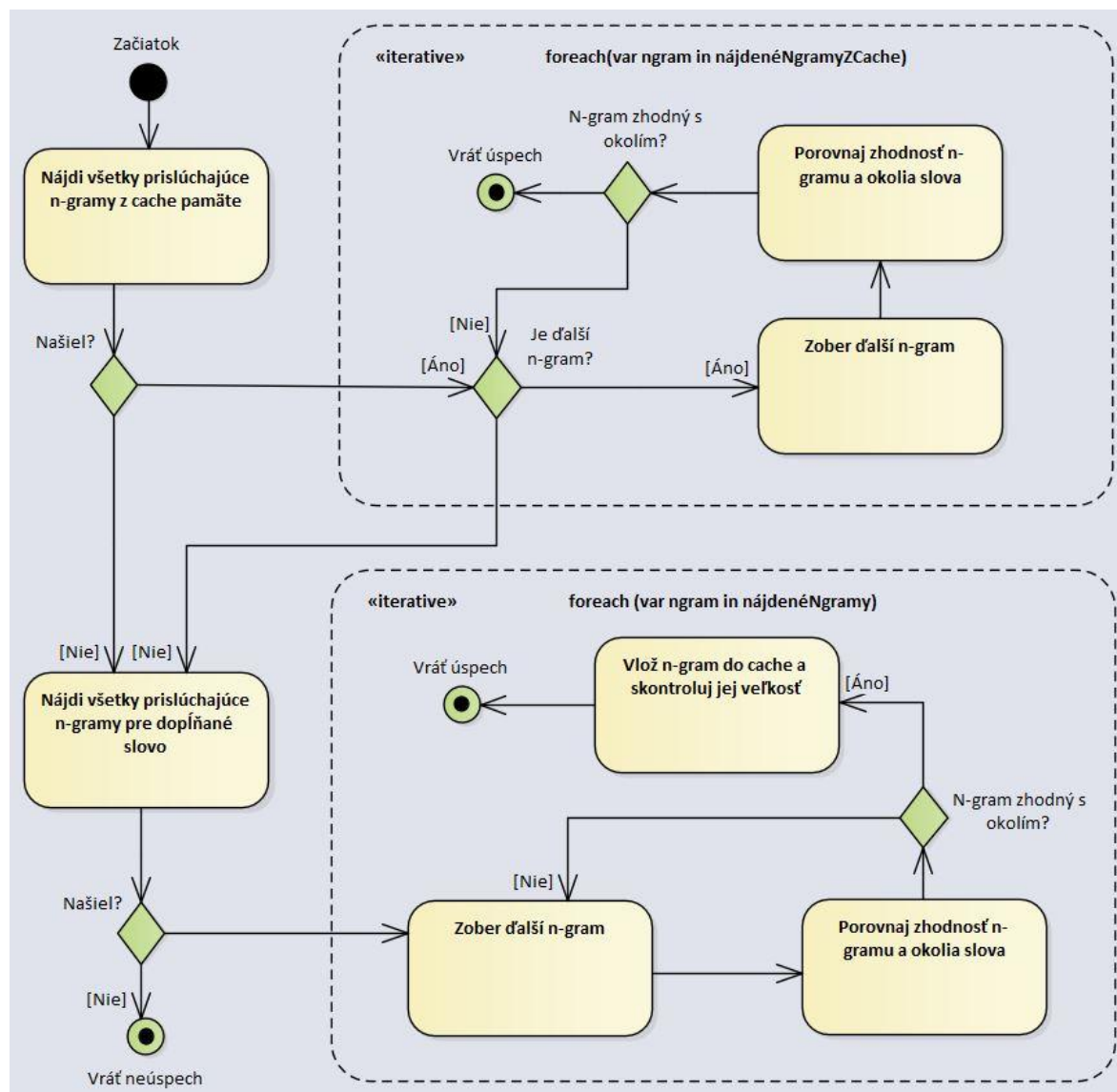


Diagram 4 – Diagram aktivít – Implementácia určenia diakritiky s Cache pamäťou

Na začiatku sa pozrieme do cache, či neobsahuje nejaké n-gramy pre naše aktuálne rekonštruované slovo. Ak áno, skúsime porovnávať zhodnosť n-gramov a okolia slova. Ak nájdeme n-gram zhodný s okolím, vrátime úspech, inak pokračujeme doteraz spomínaným postupom. Po nájdení zhodného n-gramu nám však pribudne ešte jeden krok, a to vloženie n-gramu do cache pamäte s kontrolou, či veľkosť cache nepresiahla hornú hranicu cache (ak áno, musíme odstrániť slovo s n-gramami, ktoré prišlo ako prvé – použijeme princíp úplného FIFO frontu).

Priemerný čas rekonštrukcie jedného slova sa oproti poslednej úprave znížil o polovicu. Z 0,009s (9ms) na 0,0046 (4,6ms). Úspešnosť algoritmu ovplyvnená nebola. Využitie cache pamäte (koľko slov sa za jej pomoci zrekonštruovalo) uvedieme v kapitole 4.

3.3.3 Pomocné procesy algoritmu

Pozrieme sa aj na niektoré malé pomocné algoritmy, ktoré zásadne ovplyvňujú chod a správanie celého programu.

Rozdeľovanie vstupného textu (metóda Split)

Jednou z podstatných zásad nášho algoritmu bolo, aby sme pri dopĺňaní diakritiky zachovali formátovanie textu a všetky interpunkčné znamienka (ako sme spomenuli v kapitole 3.1 Návrh algoritmu). Pri rozdeľovaní vstupného textu (string-u) na jednotlivé slová máme možnosť použiť vstavanú funkciu `public String[] Split(params char[] separator)` z jazyka C#. Táto funkcia by nám však rozdelila text podľa zadaného znaku/znakov napr. (' ', '\n', '\t', '\r') a to by znamenalo, že rozdelené slová by pri sebe obsahovali rôzne (nami nepožadované) znaky ako čiarky, bodky, úvodzovky a iné. Pozrime si nasledujúce ukážkové súvetie:

„Ale predsa kedy-tedy pridu,“ poznamenal, „nemyslis?!“ „Urcite nie...“ odpovedal.

Jeho rozdelenie pomocou vstavanej funkcie Split:

„Ale	predsa	kedy-tedy	pridu,“	poznamenal,	„nemyslis?!“	„Urcite	nie...“	odpovedal.
------	--------	-----------	---------	-------------	--------------	---------	---------	------------

Vidíme, že text sa nám rozdelil na 9 slov, z ktorých väčšina obsahuje nami neakceptované znaky, ktoré sa ani nenachádzajú v našich n-gramoch (viac v kapitole 2 Údaje a ich čistenie – 2.2.2 Odstránenie číslíc a neznámych znakov). Takéto rozdelenie prináša komplikáciu, pretože každé slovo musíme ešte zvlášť skontrolovať a očistiť, aby sme mohli začať porovnávanie s n-gramami. Možnosť najprv odstrániť všetky neakceptované znaky a následne rozdeliť text nie je možná, ak chceme zachovať vo výslednom zrekonštruovanom texte všetky tieto znaky. Ideálne rozdelenie by bolo, aby sme mali všetky slová osamostatnené a pritom aby sme nestratili žiadne informácie o interpunkčných znamienkach.

Preto sme implementovali vlastnú funkciu Split, ktorá rozdelí text nasledovne:

„	Ale		predsa		kedy	-	tedy		pridu	,	“	poznamenal
---	-----	--	--------	--	------	---	------	--	-------	---	---	------------

,	„	Nemyslis	?!“	,	„Urcite		nie	...“	odpovedal	.
---	---	----------	-----	---	---------	--	-----	------	-----------	---

Vo vytvorenom poli máme každý druhý prvok zodpovedajúci požadovanému slovu a každý prvý určujúci interpunkciu medzi slovami. Teraz nám stačí zobrať z poľa iba prvky obsahujúce slová. Ostatné prvky poľa budú obsahovať interpunkciu a aj biele znaky ako

medzery, tabulátory, nové riadky a iné. Takto zabezpečíme zhodnosť formátovania výsledného textu s originálnym textom.

Ignorovanie URL adries

Naše testovacie texty náhodne stiahnuté z internetu taktiež obsahujú URL adresy. Ak by text obsahoval adresu napr.: <https://svet.sme.sk/c/22100125/katedralu-notre-dame-v-parizi-zachvatil-poziar.html>, algoritmus by ju zrekonštruoval do podoby: <https://svet.sme.sk/c/22100125/katedrálu-notre-dame-v-paríži-zachvátil-požiar.html>, čo je samozrejme nežiadúce. Preto sme implementovali ignorovania URL adries. Je obsiahnutý vo vlastnej metóde Split, ktorú sme spomenuli vyššie. Ak metóda Split nájde URL, nechá ju v jednom prvku poľa (ako jedno slovo), čo zabezpečí následné nedoplnenie diakritiky. URL je rozpoznaná na základe jedného zo začiatkových kľúčových slov: „www“, „http“, „https“, „ftp“ alebo „ssh“. Ak algoritmus príde ku takémuto slovu, pridá ku nemu všetky nasledujúce znaky až po prvú medzeru, tabulátor alebo nový riadok.

3.4 Popísanie výsledného softvéru

Pre pochopenie softvérového návrhu uvedieme diagram tried výsledného softvéru. Obmedzíme sa však len na jadro problému, a tým je hlavná funkcionálna rekonštrukcia diakritiky. Riešenie má niekoľko podporných projektov pre čistenie dát, vytváranie súborov, alternatívne prístupy ku riešeniu problematiky či vizualizáciu webovou aplikáciou, ktoré pre jednoduchosť nebudeme uvádzať.

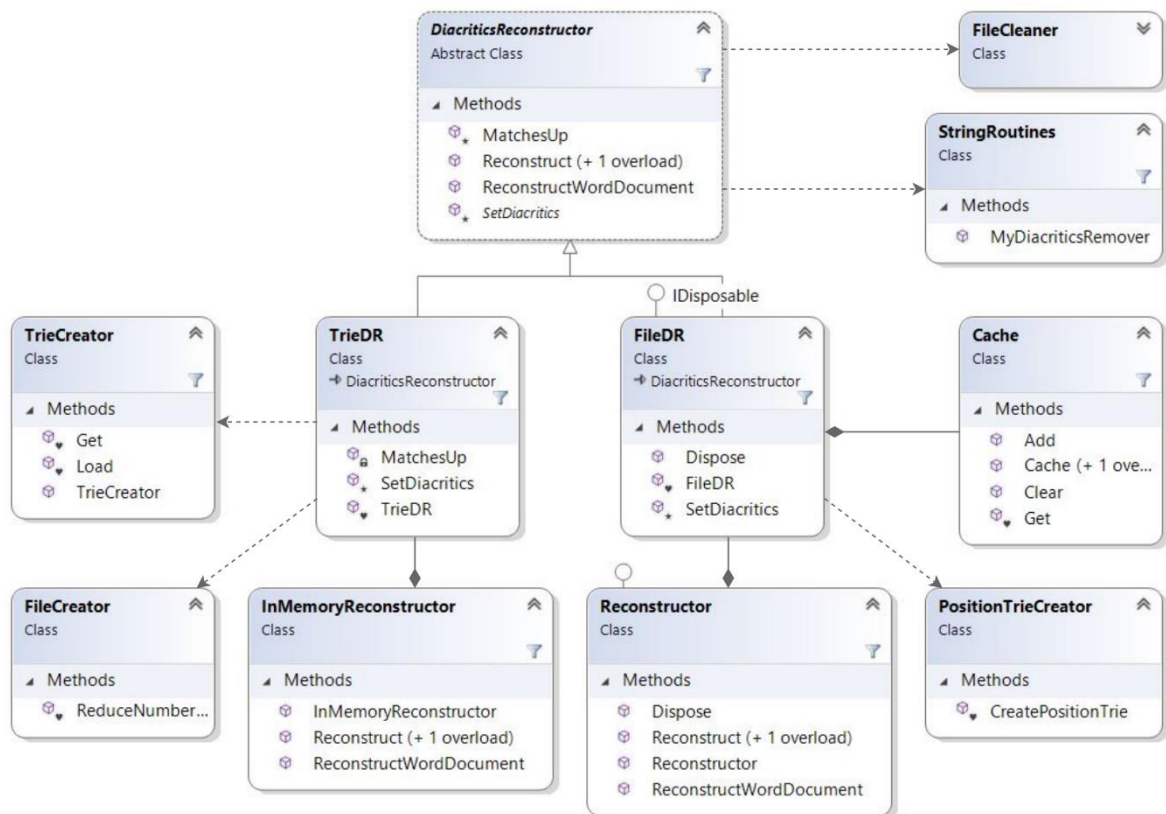


Diagram 5 – Diagram tried – Hlavná časť programu

V krátkosti popíšeme každú triedu:

- **TrieDR** – Trie Diacritics Reconstructor, potomok triedy DiacriticsReconstructor, je konkrétnou implementáciou algoritmu z kapitoly 3.3.1 (Údajová štruktúra v operačnej pamäti (in-memory)).
- **FileDR** – File Diacritics Reconstructor, potomok triedy DiacriticsReconstructor, ktorá je implementáciou algoritmu z kapitoly 3.3.2 (Údajová štruktúra na pevnom disku).

- **DiacriticsReconstructor** – abstraktná trieda obsahujúca spoločnú logiku pre obe implementácie navrhnutého algoritmu, obsahuje abstraktnú metódu `SetDiacritics`, ktorá je kľúčovou metódou pre dopĺňanie diakritiky a každý potomok tejto triedy si musí túto metódu a jej logiku implementovať sám.
- **FileCleaner** – trieda má na starosti čistenie súborov, no v hlavnej logike pre rekonštrukciu diakritiky má za úlohu iba určovať, či reťazce obsahujú validné alebo nevalidné znaky. Preto celú triedu neuvádzame.
- **StringRoutines** – obsahuje jednu statickú metódu `MyDiacriticsRemover`, ktorá odstraňuje diakritiku zo vstupného reťazca.
- **TrieCreator** – vytvára objekt Trie pre triedu `TrieDR`.
- **FileCreator** – má na starosti vytváranie binárneho súboru, no v hlavnej logike poskytuje iba jednu metódu `ReduceNumberOfNgrams`, ktorú využíva `TrieDR`.
- **Cache** – predstavuje cache pamäť opísanú v podkapitole 3.3.2 (Údajová štruktúra na pevnom disku), ktorá drží posledných m doplnených slov s ich použitými n-gramami.
- **PositionTrieCreator** – vytvára štruktúru Trie, ktorú využíva `FileDR`, je to Trie obsahujúci slová s ich pozíciami do binárneho súboru.
- **InMemoryReconstructor** – jedna z dvoch verejných tried, zaobalujúca triedu `TrieDR`, má v čo najväčšej miere oddeliť vnútornú implementáciu od okolia.
- **Reconstructor** – jedna z dvoch verejných tried, zaobalujúca triedu `FileDR`, má v čo najväčšej miere oddeliť vnútornú implementáciu od okolia.

4 Testovanie a porovnanie algoritmov

Po vytvorení funkčného softvéru by sme chceli náš program otestovať na vybranej množine testovacích textov a zistiť jeho presnosť a rýchlosť. Tiež chceme na rovnakej množine textov otestovať jestvujúce programy, ktoré sme spomenuli v 1. kapitole, a zistené výsledky porovnať.

Množinu testovacích textov sme rozdelili do troch kategórií podľa literárnych štýlov: odborné, publicistické a umelecké texty. Tieto literárne štýly sa najčastejšie vyskytujú v písanej forme. Snažili sme sa vyberať texty tak, aby jednotlivé kategórie v celkovej množine boli v približne rovnakom zastúpení t. j. po jednej tretine. Po náhodnom vybratí dvadsiatich piatich textov podľa kategórií sme získali množinu obsahujúcu viac ako 53 000 slov, pričom každá kategória má približne 17 000 – 18 000 slov. Zoznam týchto textov je možné vidieť v prílohe tejto práce.

4.1 Automatizácia testovania

Pre automatické testovanie sme vytvorili konzolovú aplikáciu, ktorej poskytneme textový súbor obsahujúci originálny text s diakritikou. Ako prvé vytvorí nový textový súbor s textom bez diakritiky, potom dá zrekonštruovať tento text nami zvolenému algoritmu a následne porovná originálny text s doplneným textom. Program vytvorí súbor so štatistikou o úspešnosti, rýchlosti, využití n-gramov a ďalších informáciách a tiež súbor s vypísanými chybami (slovami, ktoré mali zle doplnenú diakritiku aj s ich okolím) a originálnymi slovami.

Testovanie jestvujúcich programov sme vykonávali poloautomatizovane. Dali sme zrekonštruovať texty bez diakritiky každému rekonštruktoru a výsledné texty sme ukladali do textových súborov. Modifikovaním jestvujúceho programu sme vytvorili tester, ktorého vstupom sú originálny a zrekonštruovaný text a výstupom sú štatistický súbor a súbor s chybami (podobne ako pri testovaní našich algoritmov). Tento program viac popíšeme v kapitole 5.3 Testovač diakritiky.

4.2 Porovnanie

Pre každý algoritmus uvidíme samostatnú tabuľku s výsledkami. Podrobnejšiu tabuľku s výsledkami každého jedného programu pre každý testovací text uvádzame v prílohe.

Pre naše algoritmy budeme uvádzať informácie ako rýchlosť či veľkosť v pamäti. Tieto údaje však nebudeme môcť uviesť pre jestvujúce programy, pretože tieto programy sú dostupné ako internetové služby, resp. stránky, čo spôsobuje, nemožnosť odmerania presnej rýchlosti.

Náš algoritmus s údajovou štruktúrou v operačnej pamäti (in-memory) (kapitola 3.3.1)

Tabuľka 23 – Štatistika testovania – Algoritmus s údajovou štruktúrou v RAM

Texty	Čas (ms)	Počet slov	Počet chýb	Úspešnosť	Využitie n-gramov				Všetky n-gramy	Vyriešenie n-gramami
					4-gramy	3-gramy	2-gramy	1-gramy		
Odborné	3993.91	18437	25	99.86 %	373	2014	6475	5197	14059	76.25 %
Publicistické	4362.82	17672	223	98.74 %	576	2884	7591	3917	14968	84.70 %
Umelecké	4300.60	17187	513	97.02 %	198	1687	8027	5315	15227	88.60 %
SPOLU	12963.19	53296	1026	98.07 %	1147	6585	22093	14429	44254	83.03 %

- Maximálna veľkosť v RAM: 4 599 615 776 bajtov resp. **4,28 GB**
- Priemerná rýchlosť rekonštrukcie jedného slova: 0,00024 s resp. **0,24 ms**

Celkovú úspešnosť sme dosiahli 98,07 % a pomocou n-gramov sme doplnili diakritiku pre 83,03 % slov (ostatným slovám nebola doplnená diakritika).

Náš algoritmus s údajovou štruktúrou na pevnom disku (kapitola 3.3.2)

Tabuľka 24 – Štatistika testovania – Algoritmus s údajovou štruktúrou na pevnom disku

Texty	Čas (ms)	Počet slov	Počet chýb	Úspešnosť (%)	Využitie n-gramov				Všetky n-gramy	Vyriešenie n-gramami	Z cache pamäte	Z cache (%)
					4-gramy	3-gramy	2-gramy	1-gramy				
Odbor.	61708.8	18437	26	99.86	835	3054	5338	4832	14059	76.25 %	6402	45.54
Public.	79790.0	17672	200	98.87	1476	4538	5915	3039	14968	84.70 %	6518	43.55
Umelec.	76311.2	17187	508	97.04	559	3061	6091	5516	15227	88.60 %	6801	44.66
SPOLU	222198.3	53296	975	98.17	2870	10653	17344	13387	44254	83.03 %	19721	44.56

- Maximálna veľkosť v RAM: 1 249 472 808 bajtov resp. **1.16 GB**
- Priemerná rýchlosť rekonštrukcie jedného slova: 0,00417 s resp. **4,17 ms**

Do údajovej štruktúry na disku sa „zmestí“ viac n-gramov, preto algoritmus dosahuje lepšiu úspešnosť, no zase je pomalší. Všimnime si, že zo všetkých použitých n-gramov bolo až **44,56 %** z cache pamäte. Vďaka tomu má algoritmus rýchlosť 4,17ms (ak by sme cache pamäť nemali, rýchlosť rekonštrukcie jedného slova by bola minimálne dvojnásobná).

DIAKRITIK JÚĽŠ – nástroj na rekonštrukciu diakritiky [3]

Jazykovedný ústav Ľudovíta Štúra ponúka možnosť zvolenia metódy, ktorou sa bude diakritika dopĺňať. Ide o metódy: first (výber prvej možnosti, ktorú nájde), random (výber náhodnej diakritiky pre každé slovo), naive (výber diakritiky, ktorá sa v jazyku vyskytuje najčastejšie), n-gram (výber diakritiky na základe okolia slova v dĺžke n slov, kde n môže byť 2, 3, 4, 5 alebo 6), surreal a maximalist. Predvolenou možnosťou sú 4-gramy. JÚĽŠ tiež uvádza, že pri 5-gramoch a 6-gramoch je zlepšenie minimálne. Preto sme sa rozhodli testovať tento rekonštruktor s prednastavenou metódou.

Tabuľka 25 – Štatistika testovania – DIAKRITIK JÚĽŠ

Texty	Počet slov	Počet chýb	Chybovosť	Úspešnosť
Odborné	18437	571	3.10 %	96.90 %
Publicistické	17672	318	1.80 %	98.20 %
Umelecké	17187	739	4.30 %	95.70 %
SPOLU	53296	1628	3.05 %	96.95 %

Ako môžeme vidieť, na našej množine testovacích textov úspešnosť vyšla 96,95 %, pričom na stránke je uvedené, že chybovosť je 0,21 % resp. úspešnosť je 99,79 %. Táto nezhoda môže byť spôsobená malou testovacou množinou.

Rekonštrukcia je rýchla, no obmedzená. Zistili sme, že rekonštruovaný text môže mať maximálne 10 000 znakov. Všetky ďalšie znaky sú odstránené. To znamená, že pri testovaní dlhších textov, sme ich museli rozdeľovať a opäť spájať. Texty sme rozdeľovali podľa odsekov, takže nenastala chyba rozdelenia v strede vety alebo v strede slova. Toto obmedzenie je pochopiteľné, aby nenastalo zahlienie servera príliš veľkými textami.

Našli sme aj niekoľko nedostatkov rekonštruktora:

- Ignorovanie viacnásobných medzier – Ak sa medzi slovami vyskytujú viac ako jedna medzera resp. biely znak, rekonštruktor tieto medzery ignoruje. Toto spôsobilo problémy pri automatizovanom testovaní správnosti doplnenia diakritiky. Museli sme náš testovací softvér upraviť tak, aby túto chybu akceptoval.
- Doplňanie diakritiky do slova „a“ – V slovenčine sa najčastejšie vyskytuje slovo „a“ bez diakritiky. Napriek tomu rekonštruktor veľmi často doplnil diakritiku „ä“, „á“ alebo „ä“.
- URL adresy – V URL adresách sa zvyčajne diakritika nevyskytuje. No napriek tomu ju DIKARITIK dopĺňa. Napríklad: <https://referáty.centrum.sk/prírodné-vedy/biológia-á-geológia/56757/?print=1>,
https://zlatyfond.sme.sk/dielo/83/Chalúpka_Mor-ho/1#ixzz5ZDaOUGbT,
<https://www.retzer-land.at/informace-v-češtine>,
<https://ekonomika.sme.sk/c/20963930/ako-sa-esetacom-pokúsili-ukradnúť-hotel-carlton-rozhovor-kadela-relevans.html#ixzz5ZDWRoaAe>

Doplňovač diakritiky ŠTATISTICKÝ DIAKRITIKOVAČ [4]

Tabuľka 26 – Štatistika testovania – ŠTATISTICKÝ DIAKRITIKOVAČ

Texty	Počet slov	Počet chýb	Chybovosť	Úspešnosť
Odborné	18475	780	4.22 %	95.78 %
Publicistické	17680	497	2.81 %	97.19 %
Umelecké	17191	1138	6.62 %	93.38 %
SPOLU	53346	2415	4.53 %	95.47 %

Úspešnosť nám vyšla 95,74 % pričom úspešnosť, ktorá je uvedená v [4], je 98 %. Na stránke je uvedené, že diakritikovač dopĺňa diakritiku len do správne napísaných slov bez preklepov a podľa [5] program tiež neobsahuje vlastné mená. Keďže naše testovacie texty nekontrolujeme, nezabezpečujeme túto podmienku. Preto nám vyšla úspešnosť nižšia o viac ako 2 %.

Počas rekonštrukcie sa vyskytli rôzne problémy:

- Ignorovanie viacnásobných medzier – Podobne ako pri programe DIAKRITIK JÚLŠ, viacnásobné medzery sú ignorované a nahradené jedinou medzerou resp. bielym znakom.
- Veľké písmená – Program tiež v niektorých prípadoch zmení veľké písmená na malé. Podľa [5], by mal program vždy navrátiť veľké písmená do pôvodného stavu. No v niektorých prípadoch, keď sa slovo viaže s číslom alebo iným nepísmenovým znakom, sa tak nedeje. Napr.: 3t (3T), smer-sd (Smer-SD), mp3 (MP3), a iné.
- Problém s úvodzovkami – Pri niektorých typoch úvodzoviek, program akoby niekedy ignoroval celé slová, ktoré sú v okolí úvodzoviek a iných špeciálnych interpunkčných znakov (" „ “ ” ' ` ‘ ’ ... :). Tento problém sme vyriešili normalizáciou pôvodného textu, resp. odstránením všetkých týchto znakov alebo ich nahradením za akceptované znaky. Môže však nastať problém stratenia potrebnej informácie pre rekonštrukciu diakritiky slov.
- Číslice – Ak text obsahuje niektoré čísla, nahradí tieto čísla rôznymi znakmi. Uvedieme niektoré doplnené čísla: па1перт8 (18), я3зы3къ (33), в8та1йнэ (81), о3те1цъ (31) a ďalšie.
- Slovo „ty“ – Slová „ty“, „TY“, „Ty“ a „tY“ spôsobujú pád aplikácie a vyhodenie výnimky „string index out of range“. Stačí, aby sme dali doplniť diakritiku pre jediné slovo „ty“ a aplikácia padne. Snímky obrazoviek zo stránky uvádzame nižšie. Tento problém riešime odstránením všetkých slov „ty“, „TY“, „Ty“ a „tY“ z textov (potom už program nepadá). Po zrekonštruovaní opäť tieto slová vrátime do textu.

<type 'exceptions.IndexError'> at /diakritikovac
string index out of range

Python /var/www/html/diakritika.py in reconstruct_case, line 113
Web POST http://text.fiit.stuba.sk:8081/diakritikovac

Traceback (innermost first)

/var/www/html/diakritika.py in reconstruct_case

```
106.
107. def reconstruct_case(tok_orig, tok):
108.     slovo = ''
109.     tok = tok.decode('UTF-8')
110.     if(tok.isalpha() == False):      #ak je tokenom ciarka, bodka...
111.         return tok
112.     for i, c in enumerate(tok):
113.         if(tok_orig[i].isupper()):
114.             slovo = slovo + (c.upper())
115.             elif(tok_orig[i].islower()):
116.                 slovo = slovo + (c.lower())
117.     return slovo
118.
119. def reconstruct(text):
```

▼ Local vars

Variable	Value
c	u'y'
i	2
slovo	u't\u043f'
tok	u't\u043fy\u0441\u0442\u044a'
tok_orig	u'ty'

/var/www/html/diakritika.py in reconstruct

```
126.         r_case.append( (ws, tok_orig, reconstruct_case(tok_orig, tok)) )
```

► Local vars

/var/www/html/diakritika.py in POST

```
154.         r = reconstruct(text)
```

► Local vars

/usr/lib/python2.7/site-packages/web/application.py in handle_class

```
395.         return tocall(*args)
```

► Local vars

/usr/lib/python2.7/site-packages/web/application.py in _delegate

```
419.         return handle_class(cls)
```

► Local vars

Response so far

HEADERS

Variable	Value
Content-Type	'text/xml'

BODY

Request information

INPUT

Variable	Value
text_input	'ty'

COOKIES

Variable	Value
_ga	'GA1.2.759960265.1553763191'
_gid	'GA1.2.1427107562.1555996309'

META

Variable	Value
app_stack	[<web.application.application instance at 0x3cd0e60>]
data	'text_input=ty'
fullpath	u'/diakritikovac'
headers	[('Content-Type', 'text/xml')]
home	u'http://text.fiit.stuba.sk:8081'
homedomain	u'http://text.fiit.stuba.sk:8081'
homepath	u''
host	u'text.fiit.stuba.sk:8081'
ip	u'95.105.177.115'
method	u'POST'
output	u''
path	u'/diakritikovac'
protocol	u'http'
query	u''
realhome	u'http://text.fiit.stuba.sk:8081'
status	'200 OK'

ENVIRONMENT

Variable	Value
ACTUAL_SERVER_PROTOCOL	'HTTP/1.1'
CONTENT_LENGTH	'13'
CONTENT_TYPE	'application/x-www-form-urlencoded'
HTTP_ACCEPT	'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3'
HTTP_ACCEPT_ENCODING	'gzip, deflate'
HTTP_ACCEPT_LANGUAGE	'en,sk;q=0.9,es-ES;q=0.8,es;q=0.7,cs;q=0.6'
HTTP_CACHE_CONTROL	'max-age=0'
HTTP_CONNECTION	'keep-alive'
HTTP_COOKIE	'_ga=GA1.2.759960265.1553763191; _gid=GA1.2.1427107562.1555996309'
HTTP_HOST	'text.fiit.stuba.sk:8081'
HTTP_ORIGIN	'http://text.fiit.stuba.sk:8081'
HTTP_REFERER	'http://text.fiit.stuba.sk:8081/'

Obrázok 3 – Snímky obrazoviek – Pád aplikácie ŠTATISTICKÝ DIAKRITIKOVAČ

Ako sme uviedli, niektoré chyby vznikajúce pri rekonštrukcii diakritiky v texte, sme sa snažili opraviť normalizáciou pôvodného textu, aby sme vôbec mohli text nechať zrekonštruovať a aby program nespadol. No témou tejto bakalárskej práce nie je vytváranie textu pre diakritikovač, aby dosiahol čo najväčšiu úspešnosť. Preto sme normalizovali text tak, aby sa dal vôbec zrekonštruovať a niektoré chyby sme ani neriešili.

Všimnime si, že v tabuľke je celkový počet slov väčší ako pri ostatných rekonštruktoroch. Je to spôsobené normalizáciou textu, kedy sme v ňom nahradzovali slová medzerami alebo inými vhodnými znakmi.

Doplňač diakritiky DIAKRITIKA BRM [6]

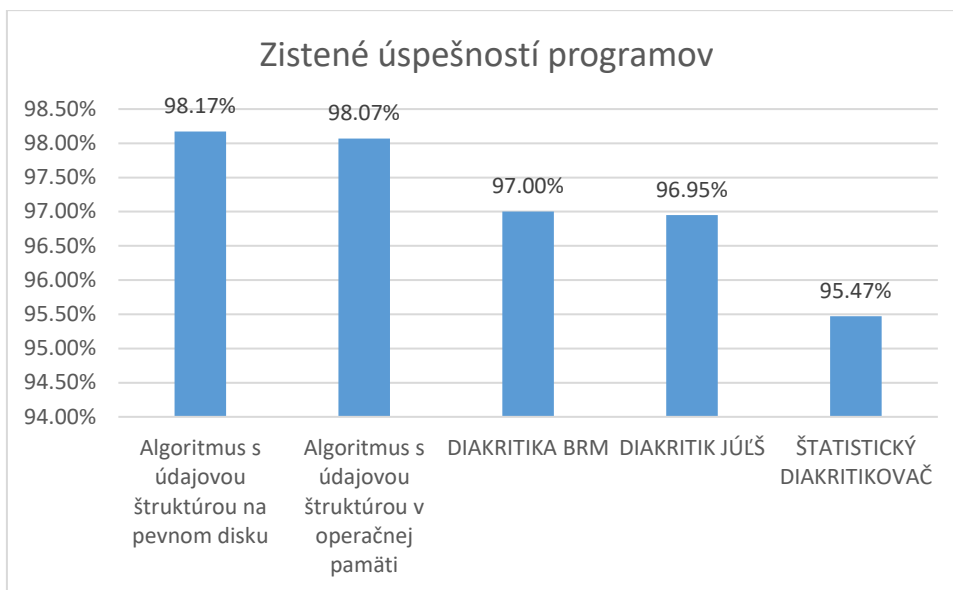
Tabuľka 27 – Štatistika testovania – DIAKRITIKA BRM

Texty	Počet slov	Počet chýb	Chybovosť	Úspešnosť
Odborné	18437	476	2.58 %	97.42 %
Publicistické	17672	375	2.12 %	97.88 %
Umelecké	17187	750	4.36 %	95.64 %
SPOLU	53296	1601	3.00 %	97.00 %

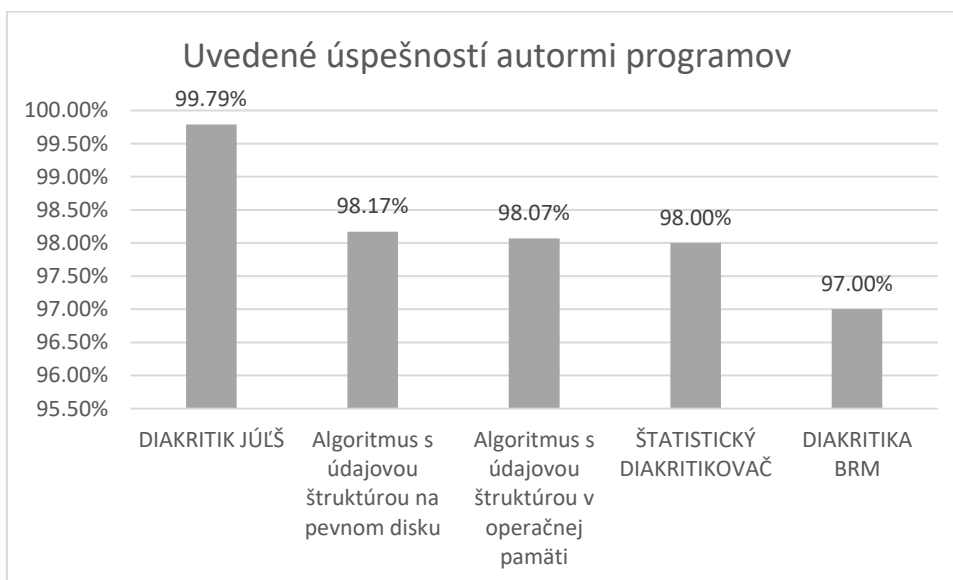
Aj keď tento program funguje iba na princípe doplnenia diakritiky s najväčšou pravdepodobnosťou (resp. najväčším výskytom), dosiahol veľmi dobrú úspešnosť. Nemá žiadny z problémov, ktoré sme uviedli vyššie. Chyby, ktoré vytvára, sú čisto spôsobené nesprávnym doplnením diakritiky.

4.3 Zhrnutie

Po porovnaní rôznych programov a ich zistených úspešnostiach sa pozrieme na zhrňujúce grafy.



Graf 3 – Porovnanie programov – Zistené úspešnostíí programov



Graf 4 – Porovnanie programov – Uvedené úspešnostíí autormi programov

Vzniknuté rozdiely môžu byť spôsobené rôznymi chybami, s ktorými sa nepočítalo pri vytváraní a testovaní programov a tiež malou množinou našich testovacích textov. Pre dôslednejšie otestovanie týchto rekonštruktorov diakritiky, by sme potrebovali vytvoriť veľkú množinu testovacích textov, čo by bolo časovo náročné. Tiež by problém nastával, ak by sme chceli mať texty v požadovanom pomere: 1/3 odborné, 1/3 publicistické a 1/3 umelecké texty.

Testovacie texty sme nekontrolovali. To znamená, že môžu obsahovať gramatické alebo štylistické chyby, preklepy, nespisovné, nárečové alebo úplne nesprávne slová. Tento aspekt tiež môže spôsobiť rozdiel v zistených úspešnostiach programov.

5 Výsledný softvér

V tejto stručnej kapitole predstavíme výsledný softvér, ako hotový produkt. Náš výsledný algoritmus sme zaobalili do troch hlavných produktov. Prvý z nich je knižnica DLL, druhý je webová stránka s API službou a tretí je testovač diakritiky. Všetky spomenuté programy a súbory sú priložené v prílohe.

5.1 Knižnica - DLL

Z nášho programu sme extrahovali len tie najpotrebnejšie časti programu pre samotné dopĺňanie diakritiky a vložili sme ich do nového DLL projektu. Chceli sme zachovať jednoduchosť a čo najmenšiu veľkosť DLL. Diagram tried zodpovedá diagramu:

Diagram 5 – Diagram tried – Hlavná časť programu.

Pre používanie je potrebné len pripojiť knižnicu do projektu. Programátor môže využívať dve verejné triedy: `InMemoryReconstructor` a `Reconstructor`. Obe poskytujú nasledovnú množinu metód:

- `public string Reconstruct(string text)` – vráti zrekonštruovaný text vložený parametrom
- `public void Reconstruct(string sourcePath, string destinationPath)` – zrekonštruuje text v zdrojovom textovom súbore (.txt) a vloží ho do cieľového textového súboru (.txt)
- `public void ReconstructWordDocument(string sourcePath, string destinationPath)` – zrekonštruuje text v zdrojovom súbore programu Microsoft Word (.docx) a vloží ho do cieľového súboru (.docx)

`InMemoryReconstructor` predstavuje rýchly algoritmus pracujúci so štruktúrou v RAM a `Reconstructor` predstavuje algoritmus využívajúci štruktúru na pevnom disku. Konštruktory oboch tried vyžadujú cesty ku nami vytvoreným súborom:

- *CompoundBinFile* – Binárny súbor s jednotlivými n-gramami pre každé slovo (Obrázok 2 – Uloženie n-gramov do binárneho súboru)
- *PositionTrie* – Súboru pozícií do binárneho súboru, resp. Trie štruktúra uložená v súbore (Obrázok 2 – Uloženie n-gramov do binárneho súboru)

5.2 Webová aplikácia

Tiež sme chceli našu aplikáciu spraviť dostupnú pre verejnosť a nasadiť ju na používanie ako webovú aplikáciu s API službou – <https://diakritika.fri.uniza.sk>. Využili sme framework ASP.NET Core na vytvorenie základnej MVC Web aplikácie.

Stránka obsahuje dve podstránky:

- Domovská stránka zobrazuje dve textové polia. Do prvého je možné zadať text a stlačiť tlačidlo „Doplň diakritiku“ a v druhom textovom poli sa zobrazí text s diakritikou.
- Podstránka O stránke ponúka informácie ako vznikla táto webová služba a informácie o nej, uvádza jej autorov a kontakty, odkazy na súvisiace inštitúcie, GitHub repozitár a oboznámenie so službou API.

Služba REST API počúva na žiadosť http metódy POST na <https://diakritika.fri.uniza.sk/api/Reconstructor>, prijíma JSON s textom na rekonštrukciu diakritiky a vracia JSON so zrekonštruovanou diakritikou.

Stránka využíva spomínanú DLL knižnicu a z nej triedu Reconstructor. Teda rekonštrukcia je pomalšia, ale zato dosahuje lepšiu úspešnosť. Tiež vyžaduje menšie nároky na RAM a nebude príliš zaťažovať server. Museli sme obmedziť aj dĺžku vstupného textu na 10 000 znakov, aby neprišlo k zahlteniu servera či prípadnému zneužívaniu.

Obrázky z webovej stránky:

Užívateľia si v novej cyklistickej sezóne budú môcť vybrať z troch základných programov.

Služba zdieľaných bicyklov, bikesharing, v Bratislave vstupuje do novej sezóny. Jeho prevádzkovatelia tvrdia, že odstránili problémy z minuloročnej testovacej prevádzky a sľubujú väčší počet bicyklov i spustenie mobilnej aplikácie. Zároveň predstavili nový cenník.

Ponuka platná od začiatku apríla zahŕňa programy pre dlhodobé aj krátkodobé využitie služby, a rovnako myslí aj na študentov, mladých ľudí i učiteľov.

Užívateľia si v novej cyklistickej sezóne budú môcť vybrať z troch základných programov, a to podľa miery využívania bikesharingu. "Nový cenník odráža to, čo sa predpokladalo už od začiatku, a síce, že po úvodnej testovacej fáze, počas ktorej sa dalo bicyklovať len za jedno euro, sa poplatky za využívanie služby dostanú na úroveň zodpovedajúcej hodnote tejto služby," vysvetľujú prevádzkovatelia.

Štandardný ročný lístok bude stať 29,40 eur, mesačný 9 eur a denný lístok 6 eur. Práve cena požičania bicykla za jeden deň vyvolala najväčšiu vlnu kritiky na facebookovskej stránke Slovnafť BAjk. Viacerým diskutujúcim sa cena zdala príliš vysoká a niektorí zas dávali príklad nižšie ceny z európskych metropol.

"Ak sa bavíme o alternatívnej doprave a chceme, aby ľudia namiesto auta/MHD využívali bicykel, tak sa bavíme o dlhodobom používaní. A teda pre človeka, ktorý je odhodlaný využívať alternatívnu formu dopravy, nie je denný program relevantný," reagoval Slovnafť BAjk na jeden z kritických príspevkov. Ľuďi teda chcú k alternatívnej doprave motivovať dlhodobým ročným programom.

Registrovalo sa 41 tisíc užívateľov

Užívateľia si v novej cyklistickej sezóne budú môcť vybrať z troch základných programov.

Služba zdieľaných bicyklov, bikesharing, v Bratislave vstupuje do novej sezóny. Jeho prevádzkovatelia tvrdia, že odstránili problémy z minuloročnej testovacej prevádzky a sľubujú väčší počet bicyklov i spustenie mobilnej aplikácie. Zároveň predstavili nový cenník.

Ponuka platná od začiatku apríla zahŕňa programy pre dlhodobé aj krátkodobé využitie služby, a rovnako myslí aj na študentov, mladých ľudí i učiteľov.

Užívateľia si v novej cyklistickej sezóne budú môcť vybrať z troch základných programov, a to podľa miery využívania bikesharingu. "Nový cenník odráža to, čo sa predpokladalo už od začiatku, a síce, že po úvodnej testovacej fáze, počas ktorej sa dalo bicyklovať len za jedno euro, sa poplatky za využívanie služby dostanú na úroveň zodpovedajúcej hodnote tejto služby," vysvetľujú prevádzkovatelia.

Štandardný ročný lístok bude stať 29,40 eur, mesačný 9 eur a denný lístok 6 eur.

Práve cena požičania bicykla za jeden deň vyvolala najväčšiu vlnu kritiky na facebookovskej stránke Slovnafť BAjk. Viacerým diskutujúcim sa cena zdala príliš vysoká a niektorí zas dávali príklad nižšie ceny z európskych metropol.

"Ak sa bavíme o alternatívnej doprave a chceme, aby ľudia namiesto auta/MHD využívali bicykel, tak sa bavíme o dlhodobom používaní. A teda pre človeka, ktorý je odhodlaný využívať alternatívnu formu dopravy, nie je denný program relevantný," reagoval Slovnafť BAjk na jeden z kritických príspevkov. Ľuďi teda chcú k alternatívnej doprave motivovať dlhodobým ročným programom.

Registrovalo sa 41 tisíc užívateľov

Dolpň diakritiku

© 2019 - DiacriticsWeb

O stránke

Služba **DiacriticsWeb** pre dopĺňanie diakritiky vznikla ako bakalárska práca študenta **Fakulty riadenia a Informatiky na Žilinskej univerzite v Žiline**.

Algoritmus rekonštrukcie diakritiky využíva jazykové korpusy **prim-8.0** voľne dostupné na stránke **Slovenského národného korpusu Jazykovedného ústavu Ľudovíta Štúra Slovenskej akadémie vied**. Ich využitie na vedecko-výskumné účely je bezplatné.

Zistená úspešnosť algoritmu pre správne dopĺňanie diakritiky je **98,17%**.

API

Request: POST: <https://diacritics.fri.uniza.sk/api/Reconstructor>

```
{
  "text": "Automatická rekonštrukcia diakritiky pre slovenčinu..."
}
```

Response:

```
{
  "text": "Automatická rekonštrukcia diakritiky pre slovenčinu..."
}
```

Odkazy

Bakalárska práca: Automatická rekonštrukcia diakritiky pre slovenčinu

GitHub: [Diacritics](#)

Vytvorili

Emanuel Zaymus

Vedúci práce: Ing. Štefan Toth, PhD.

Kontakt

Email: zaymus1@stud.uniza.sk



15.4. 2019, Žilina

© 2019 - DiacriticsWeb

Obrázok 4 – Snímky obrazovky – Vytvorená webová aplikácia

5.3 Testovač diakritiky

Pre jednoduchosť testovania doplnenej diakritiky sme vytvorili konzolovú aplikáciu, ktorá otestuje správnosť doplnenia diakritiky na základe porovnávania zrekonštruovaného textu s originálnym. Testovač vytvára súbor so štatistikou, koľko slov mal pôvodný text, koľko zrekonštruovaný text a koľko chýb sa zistilo. Jednotlivé chybné slová aj s ich okolím sú vypísané v ďalšom textovom súbore.

Konzolová aplikácia môže prijať parametre v dvoch formátoch:

1. Cesta ku originálnemu textovému súboru, cesta ku zrekonštruovanému textovému súboru, 0 – nechcem spísať štatistiku / 1 – chcem spísať štatistiku
2. Samotný originálny text, samotný zrekonštruovaný text, cesta ku súboru pre zapísanie chýb s štatistiky, 0 – nechcem spísať štatistiku / 1 – chcem spísať štatistiku

Pri zadaní nesprávnych vstupov, sa do konzoly vypíše návod, ako správne zadať vstupné parametre.

Záver

V našej práci sme sa pozreli na problematiku textov bez diakritiky, uviedli sme si dôvody, prečo takéto texty vôbec vznikajú a čo je problémom pri dopĺňaní diakritiky. Pozreli sme sa, aké programy už existujú a navrhli sme vlastný algoritmus rekonštrukcie diakritiky. Predstavili sme údaje, ktoré sme využili, ich pôvod, problémy, ktoré so sebou prinášajú a ich následné čistenie. Naš algoritmus sme implementovali dvoma spôsobmi. Zhrnieme, že náš algoritmus je založený na rýchlom prístupe ku množine prislúchajúcich n-gramov pre každé slovo a následnom porovnávaní okolia slova s jeho n-gramami. Pre rýchli prístup k n-gramom využívame pokročilú údajovú štruktúru Trie.

Vytvorili sme testovaciu množinu textov, ktorá obsahuje viac ako 53 000 slov. Otestovali sme naše algoritmy a jestvujúce programy, zozbierali sme výsledky a porovnali. Prvá implementácia nášho algoritmu v RAM dosahuje presnosť rekonštrukcie **98,07 %** s rýchlosťou **0,24 ms** na jedno slovo. Pritom potrebuje 4,28 GB RAM pamäte. Naša druhá implementácia na pevnom disku ponúka rekonštrukciu s presnosťou **98,17 %** a rýchlosťou **4,17 ms** na jedno slovo. Potrebná veľkosť pamäte RAM je 1,16 GB. Aj na prvý pohľad jednoduchý algoritmus a jeho ešte jednoduchšia implementácia dokážu získať veľmi dobrú úspešnosť pri samotnom dopĺňaní diakritiky. A to sme nemuseli využiť ani žiadne sofistikované matematické alebo štatistické metódy, či umelú inteligenciu.

Algoritmus sme zaobalili do DLL knižnice, ktorú sme využili vo webovej aplikácii, ktorá tvorí príjemné používateľské rozhranie pre sprístupnenie nášho programu verejnosti. Stránka ponúka aj REST API službu pre jednoduchú rekonštrukciu diakritiky pomocou HTTP POST požiadavky. Ako vedľajší produkt vznikla aj konzolová aplikácia pre automatizované testovanie textov s doplnenou diakritikou.

V budúcnosti by sa dal program určiť vylepšiť a zrýchliť. Taktiež sme uvažovali nad alternatívnymi prístupmi ku riešeniu problému, ktoré by sme mohli v budúcnosti vyskúšať.

6 Zoznam použitej literatúry

- [1] „Wikipédia - ASCII,“ Wikipédia, 19 2 2019. [Online]. Available: <https://sk.wikipedia.org/wiki/ASCII>. [Cit. 2 5 2019].
- [2] „Wikipédia SMS,“ Wikipédia, 24 4 2019. [Online]. Available: <https://en.wikipedia.org/wiki/SMS>. [Cit. 2 5 2019].
- [3] SNK JÚĽŠ SAV, „DIAKRITIK JÚĽŠ – nástroj na rekonštrukciu diakritiky,“ 18 8 2014. [Online]. Available: <https://korpus.sk/diakritik.html>.
- [4] J. GEDERA, „ŠTATISTICKÝ DIAKRITIKOVAČ,“ 6 2015. [Online]. Available: http://text.fiit.stuba.sk/statisticky_diakritikovac.php#focus. [Cit. 2 5 2019].
- [5] J. GEDERA, „Diakritikovač slovenského textu. Vedúci práce: Dr. Marián Šimko. Bakalárska práca. Fakulta informatiky a informačných technológií STU v Bratislave. Bratislava, 2015, 35s,“ 10 5 2015. [Online]. Available: <https://opac.crzp.sk/?fn=detailBiblioForm&sid=203CA706F79D2AD722A9ABC65647&seo=CRZP-detail-kniha>. [Cit. 2 5 2019].
- [6] R. HRAŠKA, „Dopĺňáč diakritiky,“ [Online]. Available: <https://diakritika.brm.sk/>.
- [7] V. ŽÁK, „Automatické dopĺňovanie diakritiky pro slovenštinu,“ 26 6 2007. [Online]. Available: <https://dspace.cuni.cz/handle/20.500.11956/10957>. [Cit. 2 5 2019].
- [8] M. ŠUPPA, „Diacritics Restoration for Slovak Texts,“ 5 5 2018. [Online]. Available: <http://cogsci.fmph.uniba.sk/~farkas/theses/marek.suppa.dip18.pdf>. [Cit. 2 5 2019].
- [9] J. ŽELEZNÍK, „Dopĺňovanie diakritiky pro mobilní zařízení,“ 22 6 2010. [Online]. Available: <https://dspace.cuni.cz/handle/20.500.11956/21593>. [Cit. 2 5 2019].
- [10] J. VRÁNA, „Obnovení diakritiky v českém textu,“ 8 12 2002. [Online]. Available: <https://jakub.vrana.cz/texty/diplomka.pdf>. [Cit. 2 5 2019].
- [11] D. HLADEK, J. JUHÁR a J. STAŠ, „Unsupervised Spelling Correction,“ Department of Electronics and Multimedia Communications, Faculty of Electrical Engineering and Informatics, Technical University of Kosice,, 11 2013. [Online]. Available:

- https://www.researchgate.net/publication/274705792_Unsupervised_Spelling_Correction_for_the_Slovak_Text. [Cit. 2 5 2019].
- [12] SNK JÚLŠ SAV, „Korpus,“ SNK JÚLŠ SAV, 31 1 2018. [Online]. Available: <https://korpus.sk/files/prim-8.0/>. [Cit. 2 5 2019].
- [13] M. FEDERICO, N. BERTOLTI a M. CETTOLO, „IRSTLM Toolkit,“ 2008. [Online]. Available: <https://hlt-mt.fbk.eu/technologies/irstlm>. [Cit. 2 5 2019].
- [14] SNK JÚLŠ SAV, „Štruktúra korpusu prim-8.0,“ 31 1 2018. [Online]. Available: [https://korpus.sk/prim\(2d\)8\(2e\)0.html](https://korpus.sk/prim(2d)8(2e)0.html). [Cit. 2 5 2019].
- [15] P. BISADI, „PBCD.DataStructures.Trie,“ 1 6 2018. [Online]. Available: <https://www.nuget.org/packages/PBCD.DataStructures.Trie/>. [Cit. 2 5 2019].
- [16] L. VESELÝ, „Korektor diakritiky,“ 2007. [Online]. Available: <https://core.ac.uk/download/pdf/44388922.pdf>. [Cit. 2 5 2019].

Zoznam príloh

Príloha A Parametre počítača, s ktorým sa vyvíjalo a testovalo

Príloha B Zoznam testovacích textov

Príloha C Podrobné výsledky testovania

Príloha D Obsah DVD

Prílohy

Príloha A: Parametre počítača, s ktorým sa vyvíjalo a testovalo**HP ZBook 15 G3 Mobile Workstation**

Procesor: Intel Core i7-6700HQ s Intel HD graphics 530 (2.60 GHz, 2133 MHz, 6 MB L3 cache, 4 cores, 45W), taktovaný až na 3.50 GHz s Intel Turbo Boost Technology

Čipset: Mobile Intel CM236

Grafika: Intel HD graphics 530, NVIDIA Quadro M600M

Disk: SAMSUNG MZNLN256HMHQ-000H1 - 256GB SSD Hard Drive (M2 Sata - 3 TLC)

OS: Windows 10 Pro 64-bit

Príloha B: Zoznam testovacích textov**Odborné texty (odborný/náučný štýl) (18 437 slov)**

1. Astronómia (1 475 slov)
Zdroj: <https://sk.wikipedia.org/wiki/Astron%C3%B3mia>
2. Robby Robson (4 673 slov)
Zdroj: https://sk.wikipedia.org/wiki/Bobby_Robson
3. Erózia (1 352 slov)
Zdroj: <https://sk.wikipedia.org/wiki/Er%C3%B3zia>
4. Etika a génové inžinierstvo (1 705 slov)
Zdroj: <https://referaty.centrum.sk/odborne-humanitne/psychologia/56805/etika-a-genove-inzinierstvo>
5. Mimopľúcne TBC (2 097 slov)
Zdroj: <https://referaty.centrum.sk/ostatne/nezaradene/56757/mimoplucne-tbc>
6. Staroveký Rím (5 878 slov)
Zdroj: https://sk.wikipedia.org/wiki/Starovek%C3%BD_R%C3%ADm
7. Včela medonosná (1 257 slov)
Zdroj: https://sk.wikipedia.org/wiki/V%C4%8Dela_medonosn%C3%A1

Publicistické texty (publicistický štýl) (17 672 slov)

8. Ako sa esetákom pokúsili ukradnúť hotel Carlton (rozhovor) (706 slov)
Zdroj: <https://ekonomika.sme.sk/c/20963930/ako-sa-esetacom-pokusili-ukradnut-hotel-carlton-rozhovor-kadela-relevans.html>
9. Bratislavský bikesharing predstavil nový cenník. Ľudí pobúrila cena za jeden deň (467 slov)
Zdroj: <https://finweb.hnonline.sk/ekonomika/1915689-bratislavsky-bikesharing-predstavil-novy-cennik-ludi-poburila-cena-za-jeden-den>
10. Domáca škola ju naučila s radosťou slúžiť druhým (1 007 slov)
Zdroj: <https://www.slovoplus.sk/domaca-skola-ju-naucila-s-radostou-sluzit-druhym/>
11. Huawei P30 Pro oficiálne predstavený: Ponúka krásny dizajn, 4 Leica fotoaparáty a 10x zoom (803 slov)

- Zdroj: <https://androidportal.zoznam.sk/2019/03/huawei-p30-pro-predstaveny-dizajn-leica-fotoaparaty-zoom/>
12. Macron ako novodobý Ľudovít XVI.? Je to varovanie, že sa nepoučil z histórie, hovoria Francúzi (651 slov)
Zdroj: <https://hnonline.sk/svet/1856429-macron-ako-novodoby-ludovit-xvi-je-to-varovanie-ze-sa-nepoucil-z-historie-hovoria-francuzi>
13. Najlepší politický film súčasnosti (1 638 slov)
Zdroj: <https://www.postoj.sk/40389/najlepsi-politicky-film-sucasnosti>
14. NATO má 70 rokov: Aliancia až príliš úspešná pre svoje vlastné dobroU (1 409 slov)
Zdroj: <https://www.postoj.sk/42081/nato-ma-70-rokov-aliancia-az-prilis-uspesna-pre-svoje-vlastne-dobro>
15. Páči sa mi robiť humor bez nadávok (3 480 slov)
Zdroj: <https://www.postoj.sk/39921/paci-sa-mi-robit-humor-bez-nadavok>
16. Retz - tekvice, vinice a mlyn (1 266 slov)
Zdroj: <https://blog.sme.sk/>
17. Rokovania s USA budú pokračovať, potvrdil Pellegrini (592 slov)
Zdroj: <https://www.teraz.sk/slovensko/rokovania-o-obrannej-spolupraci-s-usa-b/386983-clanok.html>
18. Smrť domorodých Američanov po príchode Európanov ochladila planétu (772 slov)
Zdroj: <https://science.hnonline.sk/klima-a-fyzika/1915464-smrt-domorodych-americanov-po-prichode-europanov-ochladila-planetu>
19. Smutná premiérka na odchode (1 385 slov)
Zdroj: <https://www.postoj.sk/42020/smutna-premierka-na-odchode>
20. Ukrajinské voľby: Porošenko reprezentuje starý systém (1 414 slov)
Zdroj: <https://www.postoj.sk/42068/ukrajinske-volby-porosenko-stratil-vylucnu-podporu-ameriky>
21. Vedecká fantastika je predobrazom toho, čo je možné (2 172 slov)
Zdroj: <https://www.postoj.sk/39669/vedecka-fantastika-je-predobrazom-toho-co-je-mozne>

Umelecké texty (umelecký štýl) (17 187 slov)

22. PETER HOTRA * Banality (ukážka z knihy) (1 019 slov)

Zdroj: <https://www.popular.sk/peter-hotra-banality-ukazka-z-knihy>

23. Jozef Gregor Tajovský: Maco Mlieč (2 995 slov)

Zdroj: https://zlatyfond.sme.sk/dielo/127/Tajovsky_Maco-Mliec/1

24. Samo Chalupka: Mor ho! (text básne) (1 298 slov)

Zdroj: https://zlatyfond.sme.sk/dielo/83/Chalupka_Mor-ho/1

25. Timrava: Ďapákovci (11 895 slov)

Zdroj:

https://wiki.szsbb.eu/lib/exe/fetch.php?media=dudasova:timrava_tapakovci.pdf

Spolu 53 296 slov

FRI

Algoritmus s údajovou štruktúrou v operačnej pamäti														
Štýl textu	Text	Názov	Veľkosť v RAM (bajty)	Čas (ms)	Počet slov	Počet chýb	Chybovosť	Úspešnosť	Vyríšenie					Vyríšenie n-gramami
									4-gramy	3-gramy	2-gramy	1-gramy	SUM	
odborný	1	Astronómia	4599417440	381.94	1475	25	1.69%	98.31%	26	133	505	492	1156	78.37%
odborný	2	Bobby Robson	4599519768	1001.221	4673	48	1.03%	98.97%	156	593	1542	991	3282	70.23%
odborný	3	Erfózia	4599583376	339.7776	1352	24	1.78%	98.22%	26	111	487	512	1136	84.02%
odborný	4	Etika a génové inžinieri	4599583264	452.8889	1705	16	0.94%	99.06%	34	276	731	427	1468	86.10%
odborný	5	Minimoplcene TBC	4599583416	446.2822	2097	40	1.91%	98.09%	23	177	630	674	1504	71.72%
odborný	6	Staroveký Rím	4599583264	1371.849	5878	119	2.02%	97.98%	80	582	2126	1695	4483	76.27%
odborný	7	Vieľa medonosná	4599613368	305.8577	1257	18	1.43%	98.57%	28	142	454	406	1030	81.94%
publicistický	8	Ako sa eseťákom pokia	4599615432	169.1678	706	13	1.84%	98.16%	16	108	245	193	562	79.60%
publicistický	9	Bratislavský bikesharin	4599615632	117.5485	467	2	0.43%	99.57%	14	85	180	99	378	80.94%
publicistický	10	Domáca škola ju naučí	4599615712	273.4971	1007	12	1.19%	98.81%	27	152	497	227	903	89.67%
publicistický	11	Huawei P30 Pro oficiál	4599615664	177.5274	803	4	0.50%	99.50%	16	84	256	207	563	70.11%
publicistický	12	Macron ako novodobý	4599615552	135.9524	561	4	0.71%	99.29%	21	92	224	124	461	82.17%
publicistický	13	Najlepší politický film s	4599615528	410.0962	1638	32	1.95%	98.05%	31	217	668	409	1325	80.89%
publicistický	14	NATO má 70 rokov	4599615472	336.822	1409	21	1.49%	98.51%	45	213	548	333	1139	80.84%
publicistický	15	Päť sa mi robí humor	4599615408	832.7536	3480	42	1.21%	98.79%	161	692	1679	654	3186	91.55%
publicistický	16	Retz	4599615424	293.5966	1266	22	1.74%	98.26%	41	200	501	301	1043	82.39%
publicistický	17	Rokovania s USA budú	4599615288	160.0801	592	3	0.51%	99.49%	26	88	267	126	507	85.64%
publicistický	18	Smt domorodých Ame	4599615648	198.5086	772	7	0.91%	99.09%	19	94	329	168	610	79.02%
publicistický	19	Smutná premiérka na c	4599615776	346.035	1385	13	0.94%	99.06%	48	250	586	306	1190	85.92%
publicistický	20	Ukrajinské voľby	4599615416	358.6764	1414	13	0.92%	99.08%	41	219	641	292	1193	84.37%
publicistický	21	Vedecká fantastika je t	4599615464	552.556	2172	35	1.61%	98.39%	70	390	970	478	1908	87.85%
umelecký	22	Banalita	4599615560	252.2544	1019	13	1.28%	98.72%	34	159	484	228	905	88.81%
umelecký	23	Maco Milieč	4599615216	718.7558	2995	67	2.24%	97.76%	38	368	1439	842	2687	89.72%
umelecký	24	Mor ho	4599615216	343.63	1298	73	5.62%	94.38%	23	62	507	515	1107	85.29%
umelecký	25	Ľapákovci	4599615272	2985.964	11875	360	3.03%	96.97%	103	1098	5597	3730	10528	88.66%
		Súčet		12963.19	53296	1026	1.93%	98.07%	1147	6585	22093	14429	44254	83.03%
		Priemer	4599598583											
		Maximum	4599615776											
		Rýchlosť rekonštrukcie 1 slova		0.24323										

Algoritmus s údajovou štruktúrou na pevnom disku																					
Štýl textu	Text	Názov	Veľkosť v RAM (bajty)	Čas (ms)	Počet slov	Počet chýb	Chybovosť	Úspešnosť	4-gramy	3-gramy	2-gramy	1-gramy	SUM	Vyhľadanie n-gramami	Z cache pamäte	Z cache					
odborný	1 Astronómia		1249206184	5727.94	1475	26	1.76%	98.24%	56	213	465	422	1156	78.37%	487	42.13%					
odborný	2 Bobby Robson		1249360408	16531.56	4673	48	1.03%	98.97%	348	888	1019	1027	3282	70.23%	1587	48.35%					
odborný	3 Erozia		1249443632	5267.423	1352	23	1.70%	98.30%	59	191	417	469	1136	84.02%	494	43.49%					
odborný	4 Etika a génové inžinieri		1249444100	8731.37	1705	12	0.70%	99.30%	102	454	628	284	1468	86.10%	611	41.62%					
odborný	5 Mimopľúcne TBC		1249444432	5310.039	2097	39	1.86%	98.14%	41	250	558	655	1504	71.72%	655	43.55%					
odborný	6 Staroveký Rim		1249441032	20140.47	5878	100	1.70%	98.30%	170	846	1854	1613	4483	76.27%	2082	46.44%					
odborný	7 Včela medonosná		1249472808	4388.189	1257	19	1.51%	98.49%	59	212	397	362	1030	81.94%	486	47.18%					
publicistický	8 Ako sa eseíkom pokú		1249208936	2906.84	706	12	1.70%	98.30%	46	170	199	147	562	79.60%	268	47.69%					
publicistický	9 Bratislavský bikesharin		1249307640	2158.476	467	0	0.00%	100.00%	39	142	153	44	378	80.94%	153	40.48%					
publicistický	10 Domáca škola ju naučil		1249323544	6166.968	1007	9	0.89%	99.11%	65	304	363	171	903	89.67%	373	41.31%					
publicistický	11 Huawei P30 Pro oficiál		1249348400	3378.594	803	3	0.37%	99.63%	46	136	263	118	563	70.11%	230	40.85%					
publicistický	12 Macron ako novodobý		1249347224	2241.729	561	5	0.89%	99.11%	57	117	227	60	461	82.17%	215	46.64%					
publicistický	13 Najlepší politický film s		1249347800	6938.425	1638	32	1.95%	98.05%	83	379	528	335	1325	80.89%	590	44.53%					
publicistický	14 NATO má 70 rokov		1249380192	6941.109	1409	16	1.14%	98.86%	122	350	497	170	1139	80.84%	435	38.19%					
publicistický	15 Páči sa mi robí humor		1249381600	14368	3480	33	0.95%	99.05%	364	1039	1130	653	3186	91.55%	1497	46.99%					
publicistický	16 Retz		1249411384	5956.398	1266	24	1.90%	98.10%	101	279	395	268	1043	82.39%	476	45.64%					
publicistický	17 Rokovania s USA budú		1249411288	3290.515	592	2	0.34%	99.66%	69	147	209	82	507	85.64%	222	43.79%					
publicistický	18 Smrť domorodých Ame		1249413240	3415.491	772	6	0.78%	99.22%	49	180	282	99	610	79.02%	255	41.80%					
publicistický	19 Smutná premiéra na c		1249413976	6330.762	1385	13	0.94%	99.06%	125	378	479	208	1190	85.92%	504	42.35%					
publicistický	20 Ukrajinské voľby		1249414304	6393.45	1414	13	0.92%	99.08%	115	350	463	265	1193	84.37%	508	42.58%					
publicistický	21 Vedecká fantastika je f		1249413696	9303.286	2172	32	1.47%	98.53%	195	567	727	419	1908	87.85%	792	41.51%					
umelecký	22 Banality		1249413576	5404.539	1019	13	1.28%	98.72%	88	289	397	131	905	88.81%	360	39.78%					
umelecký	23 Maco Mlieč		1249411320	12065.78	2995	69	2.30%	97.70%	121	621	1008	937	2687	89.72%	1257	46.78%					
umelecký	24 Mor ho		1249406288	6635.313	1298	71	5.47%	94.53%	56	157	519	375	1107	85.29%	416	37.58%					
umelecký	25 Ľapákovci		1249404952	52205.61	11875	355	2.99%	97.01%	294	1994	4167	4073	10528	88.66%	4768	45.29%					
	Súčet		222198.27	53296	975	1.83%	98.17%	2870	10653	17344	13387	44254	83.03%	19721	44.56%						
	Priemer		1249382754																		
	Maximum		1249472808	1.16 GB																	
	Rýchlosť rekonštrukcie 1 slova		4.169136																		

DIAKRITIK								
Štýl textu	Text	Názov			Počet slov	Počet chýb	Chybovosť	Úspešnosť
odborný	1	Astronómia			1475	58	3.93%	96.07%
odborný	2	Bobby Robson			4673	101	2.16%	97.84%
odborný	3	Erozia			1352	48	3.55%	96.45%
odborný	4	Etika a génové inžinierstvo			1705	37	2.17%	97.83%
odborný	5	Mimopliúčne TBC			2097	124	5.91%	94.09%
odborný	6	Staroveký Rím			5878	162	2.76%	97.24%
odborný	7	Včela medonosná			1257	41	3.26%	96.74%
publicistický	8	Ako sa esetákom pokúsili ukradnúť hotel Carlton			706	34	4.82%	95.18%
publicistický	9	Bratislavský bikesharing predstavil nový cenník			467	4	0.86%	99.14%
publicistický	10	Domáca škola ju naučila s radosťou slúžiť druhým			1007	18	1.79%	98.21%
publicistický	11	Huawei P30 Pro oficiálne predstavený			803	15	1.87%	98.13%
publicistický	12	Macron ako novodobý Ľudovít XVI			561	10	1.78%	98.22%
publicistický	13	Najlepší politický film súčasnosti			1638	33	2.01%	97.99%
publicistický	14	NATO má 70 rokov			1409	24	1.70%	98.30%
publicistický	15	Páči sa mi robíť humor bez nadávok			3480	51	1.47%	98.53%
publicistický	16	Retz			1266	32	2.53%	97.47%
publicistický	17	Rokovania s USA budú pokračovať			592	13	2.20%	97.80%
publicistický	18	Smrť domorodých Američanov po príchode Európe			772	9	1.17%	98.83%
publicistický	19	Smutná premiérka na odchode			1385	21	1.52%	98.48%
publicistický	20	Ukrajinské voľby			1414	24	1.70%	98.30%
publicistický	21	Vedecká fantastika je predobrazom toho čo je mo			2172	30	1.38%	98.62%
umelecký	22	Banalita			1019	20	1.96%	98.04%
umelecký	23	Maco Mlieč			2995	110	3.67%	96.33%
umelecký	24	Mor ho			1298	102	7.86%	92.14%
umelecký	25	Ťapákovci			11875	507	4.27%	95.73%
	Súčet				53296	1628	3.05%	96.95%

ŠTATISTICKÝ DIAKRITIKOVAČ								
Štýl textu	Text	Názov			Počet slov	Počet chýb	Chybovosť	Úspešnosť
odborný	1	Astronómia			1476	40	2.71%	97.29%
	2	Bobby Robson			4695	135	2.88%	97.12%
	3	Erózia			1352	94	6.95%	93.05%
odborný	4	Etika a génové inžinierstvo			1708	53	3.10%	96.90%
odborný	5	Mimopľúčne TBC			2100	163	7.76%	92.24%
odborný	6	Staroveký Rím			5887	248	4.21%	95.79%
odborný	7	Včela medonosná			1257	47	3.74%	96.26%
publicistický	8	Ako sa esetákom pokúsili ukradnúť hotel Carlton			706	35	4.96%	95.04%
publicistický	9	Bratislavský bikesharing predstavil nový cenník			468	14	2.99%	97.01%
publicistický	10	Domáca škola ju naučila s radosťou slúžiť druhým			997	32	3.21%	96.79%
publicistický	11	Huawei P30 Pro oficiálne predstavený			803	48	5.98%	94.02%
publicistický	12	Macron ako novodobý Ľudovít XVI			565	10	1.77%	98.23%
publicistický	13	Najlepší politický film súčasnosti			1640	34	2.07%	97.93%
publicistický	14	NATO má 70 rokov			1409	36	2.56%	97.44%
publicistický	15	Páči sa mi robiť humor bez nadávok			3487	83	2.38%	97.62%
publicistický	16	Retz			1267	45	3.55%	96.45%
publicistický	17	Rokovania s USA budú pokračovať			592	11	1.86%	98.14%
publicistický	18	Smrť domorodých Američanov po príchode Európe			772	20	2.59%	97.41%
publicistický	19	Smutná premiérka na odchode			1386	23	1.66%	98.34%
publicistický	20	Ukrajinské voľby			1414	48	3.39%	96.61%
publicistický	21	Vedecká fantastika je predobrazom toho čo je mo			2174	58	2.67%	97.33%
umelecký	22	Banalitý			1019	28	2.75%	97.25%
umelecký	23	Maco Mlieč			2999	133	4.43%	95.57%
umelecký	24	Mor ho			1298	110	8.47%	91.53%
umelecký	25	Ďapákovci			11875	867	7.30%	92.70%
		Súčet			53346	2415	4.53%	95.47%

DIAKRITIKA BRM								
Štýl textu	Text	Názov			Počet slov	Počet chýb	Chybovosť	Úspešnosť
odborný	1	Astronómia			1475	30	2.03%	97.97%
	2	Bobby Robson			4673	88	1.88%	98.12%
	3	Erózia			1352	30	2.22%	97.78%
	4	Etika a génové inžinierstvo			1705	51	2.99%	97.01%
	5	Mimoplúcne TBC			2097	107	5.10%	94.90%
	6	Staroveký Rím			5878	134	2.28%	97.72%
	7	Včela medonosná			1257	36	2.86%	97.14%
publicistický	8	Ako sa esetákom pokúsili ukradnúť hotel Carlton			706	35	4.96%	95.04%
publicistický	9	Bratislavský bikesharing predstavil nový cenník			467	3	0.64%	99.36%
publicistický	10	Domáca škola ju naučila s radostou slúžiť druhým			1007	28	2.78%	97.22%
publicistický	11	Huawei P30 Pro oficiálne predstavený			803	9	1.12%	98.88%
publicistický	12	Macron ako novodobý Ľudovít XVI			561	7	1.25%	98.75%
publicistický	13	Najlepší politický film súčasnosti			1638	28	1.71%	98.29%
publicistický	14	NATO má 70 rokov			1409	32	2.27%	97.73%
publicistický	15	Páči sa mi robiť humor bez nadávok			3480	65	1.87%	98.13%
publicistický	16	Retz			1266	41	3.24%	96.76%
publicistický	17	Rokovania s USA budú pokračovať			592	10	1.69%	98.31%
publicistický	18	Smrť domorodých Američanov po príchode Európe			772	11	1.42%	98.58%
publicistický	19	Smutná premiérka na odchode			1385	22	1.59%	98.41%
publicistický	20	Ukrajinské voľby			1414	33	2.33%	97.67%
publicistický	21	Vedecká fantastika je predobrazom toho čo je mo			2172	51	2.35%	97.65%
umelecký	22	Banalitty			1019	16	1.57%	98.43%
umelecký	23	Maco Mlieč			2995	127	4.24%	95.76%
umelecký	24	Mor ho			1298	113	8.71%	91.29%
umelecký	25	Ťapákovci			11875	494	4.16%	95.84%
	Súčet				53296	1601	3.00%	97.00%

Príloha D: Obsah DVD

Priložené DVD obsahuje:

- Práca v elektronickej podobe (formát PDF)
- VisualStudio Projekt – Diacritics (zdrojové súbory)
- Testovacie texty – jednotlivé rekonštrukcie, štatistiky, súbory s chybami (formát .txt)
- Súbory nevyhnutné ku spusteniu aplikácie
 - compoundBinFile.dat
 - positionTrie.txt
- Ukážka n-gramov (formát .txt)