ANALÝZA VLASTNOSTÍ SLOV

BAKALÁRSKA PRÁCA

MARIKA MITRENGOVÁ

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY,FYZIKY A INFORMATIKY
KATEDRA INFORMATIKY

9.2.1 Informatika

Vedúci : RNDr. Richard Ostertág Bratislava 2009

Abstrakt

Táto práca sa zaoberá analýzou vlastností slov. Cieľom tejto bakalárskej práce je vypracovanie slovníka slovenských slov s diakritikou aj bez diakritiky a anglických slov, analýza slovníkov - rozoberanie vlastností z pohľadu anagramov, diakritiky, T9, stanovenie podobnosti slov, fonetickej podobnosti slov a alternaternatívneho spôsobu písania slov.

Kľúčové slová: klávesnica, ShapeWriter, Fittsov zákon, slovník, anagram, palindrom, T9, podobnosť

Abstract

This thesis deals with analysis of properties of words. The goal of this bachelor thesis is working out of slovak dictionary with diacritic and without diacritic and english dictionary. Analysis of these dictionaries - examination of anagrams, diacritic, T9, determination of similarity words, phonetic similarity and alternatives approach of writing words.

Keywords: keyboard, ShapeWriter, Fitts law, dictionary, anagram, palindrom, T9, similarity

Čestné prehlásenie

Vyhlasujem, že som bakalársku prácu vypracovala samostatne s použitím uvedených zdrojov.

podpis

Poďakovanie

Ďakujem vedúcemu bakalárskej práce RNDr. Richardovi Ostertágovi za odbornú pomoc a pripomienky pri jej vypracovávaní.

Obsah

Úvod

Počítač sa stal neoddeliteľnou súčasťou života väčšiny ľudí. Umožňuje používateľom pomerne rýchlo, efektívne a bezchybne vykonať niektoré operácie, ktoré by bez počítača boli veľmi zdĺhavé a chybovosť by bola celkom vysoká. Väčšina inštitúcií, organizácií a bežných používateľov doma využíva špeciálne aplikácie v počítači na uľahčenie práce. Preto sme si ako hlavný cieľ našej bakalárskej práce stanovili vytvorenie aplikácie, ktorá umožňuje analyzovať vlastnosti slovenských slov a anglických slov podľa vlastností, ktoré sú zadané (aplikáciu je možné rozšíriť aj pre iné jazyky). Medzi vlastnosti, ktoré sa môžu skúmať patria anagramy, palindromy, kolízie slov pri písaní s využitím T9, podobnosť slov, fonetická podobnosť slov, podobnosť slov po odstránení diakritiky, kolízie slov pri písaní slov s využitím ShapeWritera s klávesnicou QWERTY. Nájsť najefektívnejšie rozloženie písmen na deväť kláves, aby bolo čo najmenej slov, ktoré sa budú rovnako písať metódou T9. Posledným cieľom je nájsť efektívnejšie rozloženie kláves na virtuálnej klávesnici.

Je zaujímavé rozoberať niektoré texty, vlastnosti slov, porovnávať slovenské texty s diakritikou s textami bez diakritiky alebo aj s anglickým textami. Zaujímavé môže byť porovnanie počtu kolidujúcich slov v T9 v týchto textoch alebo hľadanie najväčšej množiny anagramov. Hľadanie, ktorý jazyk obsahuje najviac palindromov, ktoré anglické slová sa podobne vyslovujú a pod.
Program dokáže zistiť, ktoré slová sa najčastejšie používajú v danom texte. Vďaka tomu bude možné nájsť kľúčové slová pre konkrétny text. To môžu vyžiť napr. historici, keď budú analyzovať historické diela (bude možné zisťovať, ktoré témy boli zaujímavé v ktorom období), učitelia žurnalistiky, keď budú kontrolovať, či žiaci v ich dielach nepoužívajú veľmi často niektoré slová, ale aj jazykovedci, študenti a mnohí iní, ktorí rozoberajú články, rôzne texty, literárne diela a pod. Aplikáciu budú môcť využívať aj ľudia, ktorí si budú cibriť pamäť slovnými hrami (anagramy), aj tvorcovia slovných hier tým, že si môžu vyhľadať k danému slovu množinu jeho anagramov .

Kapitola 1

Základné pojmy

1.1 T9

T9 je štandard pre písanie textu na deviatich klávesoch vytvorený pre mobilné telefóny. T9 uľahčuje a urýchľuje písanie textových správ. Umožňuje vloženie slov pri jednom stláčaní tlačidla pre každé písmeno slova. V kontraste s týmto spôsobom písania je staršia technika, v ktorej je niekoľko písmen priradených k jednému tlačidlu, vybranie jedného písmena často vyžaduje viacnásobné stlačenie tlačidla.
Kombinuje skupinu písmen na každom tlačidle, ktoré bolo stlačené, so slovníkom slov. Vyhľadávanie slova v slovníku korešponduje s postupnosťou stlačených tlačidiel a zoraďuje ich podľa frekvencie použitia. Tým, že získava slová a frázy, ktoré používateľ bežne používa, urýchľuje to proces, pretože najprv ponúkne často používané slová, a potom nechá užívateľa vybrať si iné voľby jedným alebo viacerým stlačením preddefinovaného tlačidla „Ďalej“.

1.2 Anagram

Je výsledok preusporiadania písmen slova alebo frázy, pričom vznikne nové slovo alebo fráza, ktorá obsahuje všetky pôvodné písmená presne raz, napr. *orchestra = carthorse*, *Eleven plus two = Twelve plus one.*
Každé slovo alebo fráza, ktorá presne reprodukuje všetky písmená, ale v inom poradí je anagram.

Definícia 1.2.1 *Nech Π_n je množina všetkých permutácií nad $\{1, 2, \ldots, n\}$. Definujme reláciu $\equiv$ medzi slovami $w_1 = a_1 a_2 \ldots a_n$ a $w_2 = b_1 b_2 \ldots b_m$ nasledovne: $w_1 \equiv w_2 \Leftrightarrow m = n \wedge \exists \pi \in \Pi_n : \forall i = 1, 2, \ldots, n : a_i = b_{\pi(i)}$.*
Potom $anagram(w) = \{u \mid u \equiv w\}$ je množina všetkých anagramov slova w.

1.3 Palindrom

Je slovo, fráza, číslo alebo iná postupnosť znakov, ktorá má tú vlastnosť, že sa dá čítať v ľubovoľnom smere (z prava do ľava alebo z ľava do prava) a má vždy rovnaký význam. Pri posudzovaní, či je to ten istý význam sa obyčajne neberú do úvahy medzery mezi slovami a diakritika. Slovo palindrom pochádza z gréckych slov „palin"-naspäť a „dromos"-smer.

Definícia 1.3.1 *Nech slovo* $w = a_1 a_2 \dots a_n$, *potom* $w^R = a_n a_{n-1} \dots a_1$. *Slovo w je palindrom práve vtedy, keď* $w = w^R$.

1.4 Diakritika

Predstavuje ju znamienko v okolí (nad, pod alebo vedľa) písmena, ktoré pozmeňuje význam písmena (najčastejšie označuje jeho odlišnú výslovnosť).

1.5 Podobnosť

Podobnosť slov znamená, ako veľa sa dané slová na seba podobajú. Existuje mnoho kritérií podobnosti, napr. sa môže zisťovať, ktoré slová sú podobné k nejakému slovu pri zámene jedného písmena z daného slova.

1.5.1 Fonetická podobnosť anglických slov

Je špeciálny prípad podobnosti slov, kde sa nehľadá, či sa slová „skoro" rovnako píšu, ale ako sa podobá ich výslovnosť. V tejto práci budeme skúmať podobnosť slov podľa dvoch známych algoritmov: Soundex a Metaphone.

1.6 Bigram

Inak nazvaný aj digraf, je skupina dvoch nasledujúcich písmen, používa sa v štatistických analýzach textu.

Definícia 1.6.1 *Majme slovo* $w = a_1 \dots a_n$ *také, že* $\forall\, i :\ a_i \in \Sigma$.
Potom $bigram(w) = \{xy \mid x, y \in \Sigma \wedge \exists\, i \in \{1, 2, \dots, n-1\}$ *také, že* $x = a_i \wedge y = a_{i+1}\}$.

1.7 Klávesnica

Je vstupné zariadenie pozostávajúce z kláves (tlačidiel) stláčaných prstami, slúži na zadávanie dát alebo na ovládanie zariadenia.

1.8 Virtuálna klávesnica

Je softvér, ktorý umožňuje používateľovi vkladať znaky. Je zobrazená na výstupnom zariadení (displej) a užívateľ si buď prstom alebo hrotom (perom) vyberá znaky.

1.9 ShapeWriter

Inak nazvaná aj Shorthand-Aided Rapid Keyboarding (SHARK), je metóda vkladania textu cez virtuálnu (grafickú) klávesnicu pre tablety a mobilné telefóny vyvinutá Shumin Zhai-om a Per Ola Kristensson-om. Text sa na rozdiel od klasického spôsobu stláčania kláves vkladá ťahaním pera/hrotu po grafickej klávesnici, tento pohyb pera spája všetky písmená daného slova.

Kapitola 2

Podobnosť

2.1 Podobnosť slov

V súčasnosti zohráva analýza podobnosti slov dôležitú úlohu, najmä pri porovnávaní podobnosti DNA / RNA štruktúry, vo vyhľadávačoch (Google,Yahoo, apod.), opravovaní chýb počas písania textu a pri ochrane proti plagiátorstvu.

2.1.1 Levenshtein distance

V tejto podkapitole vychádzam z [11]. Levenshtein distance (inak nazvaná aj ako Edit-Distance) je jedna z možných metrík pre meranie stupňa odlišnosti medzi dvoma slovami. Majme slovo u a potrebujeme ho prepísať na slovo w. Na transformáciu u na w môžeme použiť jednu z nasledujúcich operácií:

1. Vymazanie písmena zo slova. Písmená napravo od vymazaného písmena sa posunú doľava, takže nevznikne žiadne voľné miesto.

2. Vloženie písmena na nejakú pozíciu v slove. Písmená napravo od tejto pozície budú posunuté doprava.

3. Nahradenie jedného písmena iným.

Levenshtein distance sa definuje ako minimálny počet operácií potrebných na transformáciu jednoho slova na druhé. Napríklad Levenshtein distance medzi slovami *gumbi* a *gambol* je 3:
gumbi $\xrightarrow{a \rightsquigarrow u}$ gambi $\xrightarrow{i \rightsquigarrow o}$ gambo $\xrightarrow{\textit{vloženie } l}$ gambol.

```
int LevenshteinDistance(char s[1..m], char t[1..n])
 // d je tabuľka s  m+1 riadkami a n+1 stĺpcami
 declare int d[0..m, 0..n]

 for i from 0 to m
  d[i, 0] := i
 for j from 0 to n
  d[0, j] := j

 for j from 1 to n
  for i from 1 to m {
   if s[i] = t[j] then
     cost := 0
   else
     cost := 1
   d[i, j] := minimum(
                d[i-1, j] + 1,      // vloženie
                d[i, j-1] + 1,      // vymazanie
                d[i-1, j-1] + cost   // substitúcia
              )
 }

 return d[m, n]
```

Obr. 2.1: Pseudokód algoritmu Levenshtein [13]

Popis Levenshtein distance algoritmu:

Tento algoritmus ráta najmenší počet upravovacích operácií potrebných pre zmenu jedného slova na druhé. Najbežnejší spôsob počítania je pomocou dynamického programovania. Inicializuje sa matica typu $m + 1 \times n + 1$, kde n a m sú dĺžky daných slov. Bunka $(m \times n)$ predstavuje Levenshtein distance medzi m znakmi jedného slova a n znakmi druhého slova. Každý skok horizontálne alebo vertikálne znamená jednu z operácií: vloženie, substitúcia alebo vymazanie. Cena je nastavená na 1 pre každú z operácií. Každá bunka minimalizuje cenu lokálne.

2.1.2 Dice koeficient

Definícia 2.1.1 *Majme množinu X a Y, Dice koeficient pre tieto množiny je definovaný nasledovne [12]:*

$$D_{X,Y} = \frac{2|X \cap Y|}{|X| + |Y|}$$

Keď chceme pomocou tohto koeficientu merať podobnosť slov u a w, budeme ho počítať s použitím bigramov. Dice koeficient je definovaný ako pomer dvojnásobku počtu spoločných bigramov k celkovému počtu bigramov v oboch slovách.

Definícia 2.1.2 *Majme slová u a w, Dice koeficient pre slová u a w je definovaný nasledovne:*

$$D_{u,w} = D_{bigram(u),bigram(w)} = \frac{2n_t}{n_u + n_w}$$

kde n_t je počet spoločných bigramov slov u a w, n_u je počet bigramov v slove u a n_w je počet bigramov v slove w.

Hodnoty Dice koeficientu sa pohybujú od 0 (slová sa vôbec nepodobajú) po 1(slová sú rovnaké).

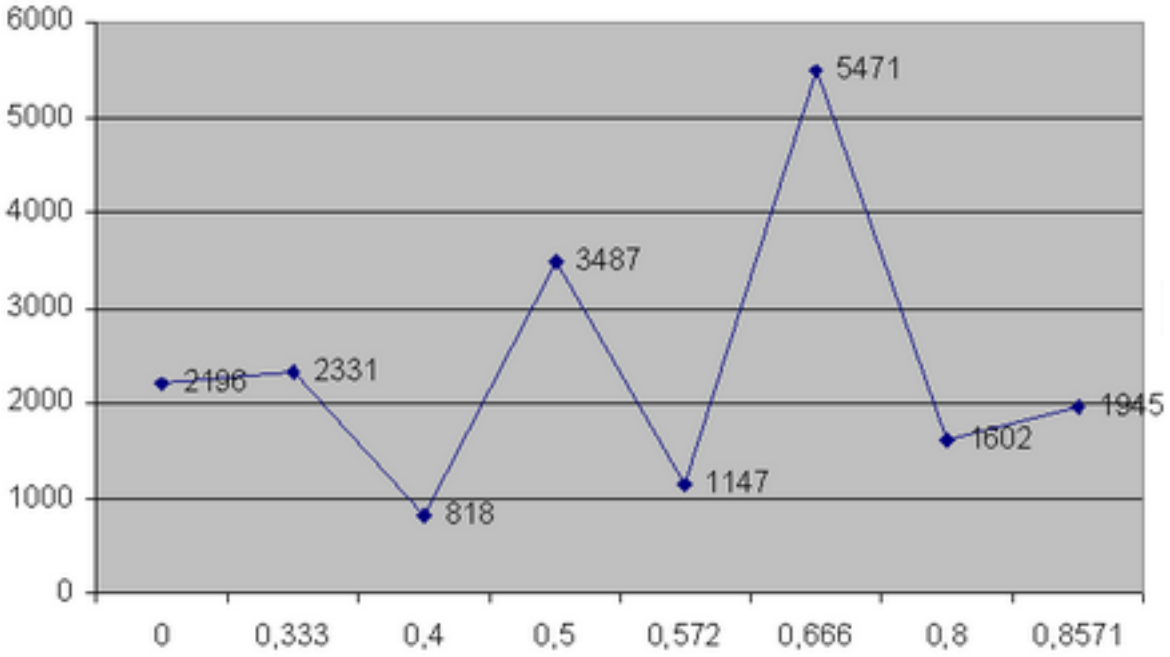
Napr. slová colour a couleur: colour má bigrami co, ol, lo, ou, ur a couleur co, ou, ul, le, eu, ur. Majú tri spoločné bigrami co, ou, ur, takže Dice koeficient je

$$s = \frac{6}{11}$$

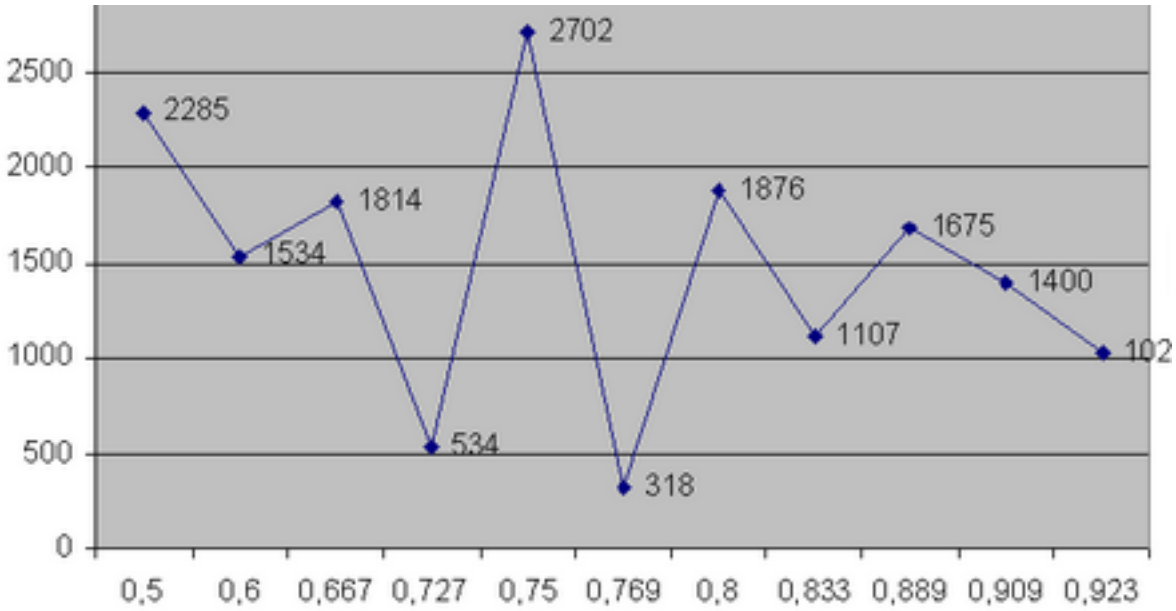
Nedostatky Dice koeficientu:

Ak budeme pomocou Dice koeficientu hľadať podobné slová, nepresnosti vzniknú pri takých slovách, ktoré nie sú rovnaké ako celé slovo, ale majú rovnaké bigramy. Vtedy je pre takéto slová Dice koeficient rovný 1 (pričom slová nie sú rovnaké), alebo je blízko 1. Napr. Dice koeficient pre slovo oboo a oobo je 1 a pre slová kebab a babke je 0,75.

Pri dĺžke slov menšej ako 5 a hodnote Levenstein vzdialenosti pre tieto slová je priemerná veľkosť Dice koeficientu 0,53. Pri dĺžke menšej ako 8 je 0,74 a pri dĺžke väčšej a rovnej ako 8 je 0,876.



Obr. 2.2: Obrázok hodnôt Dice koeficientu pre slová s dĺžkou menšou ako 5 a s hodnotou Levenstein vzdialenosti 1



Obr. 2.3: Obrázok hodnôt Dice koeficientu pre slová s dĺžkou menšou ako 8 a s hodnotou Levenstein vzdialenosti 1

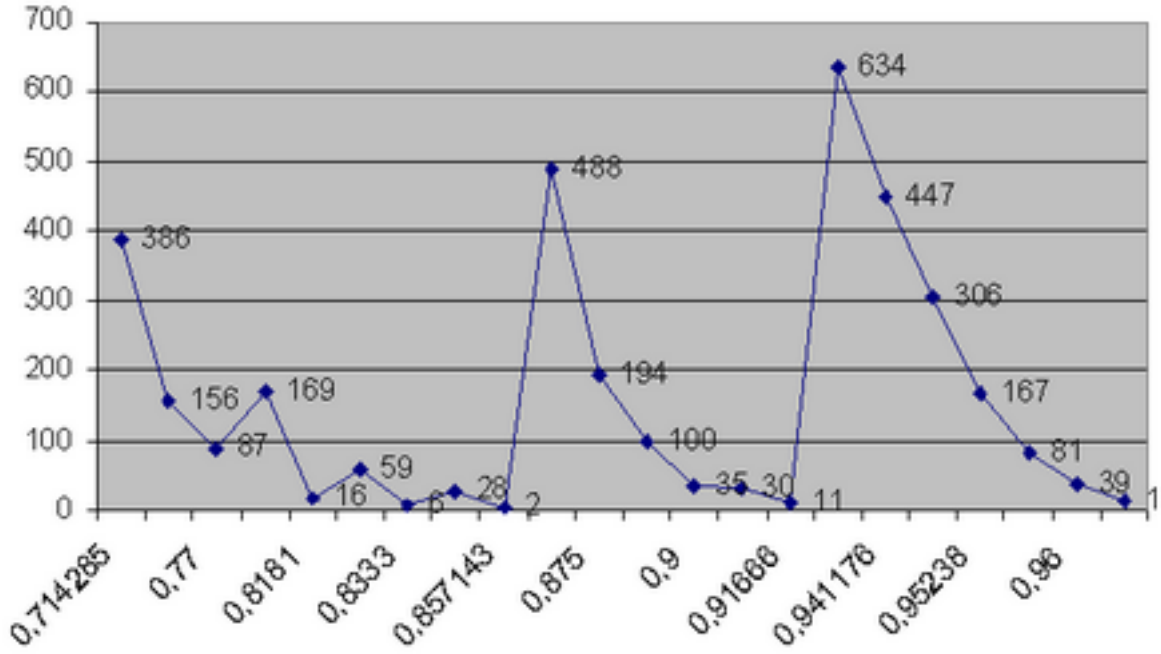
2.2 Fonetická podobnosť slov

2.2.1 Fonetické algoritmy

V tejto kapitole vychádzam z [9], [10] a [7]. Fonetický algoritmus je algoritmus pre triedenie slov do skupín podľa ich výslovnosti. Existuje mnoho algoritmov s rôznymi pravidlami a výnimkami, pretože anglické hláskovanie/pravopis a výslovnosť je komplikovaná historickými zmenami vo výslovnosti a slovami prevzatými z rôznych jazykov, avšak aplikovanie pravidiel týchto algoritmov na iné jazyky nemusí dávať dobré výsledky. Najznámejšie fonetické algoritmy sú:

Soundex - jeden z prvých algoritmov pre hľadanie slov s rovnakou výslovnosťou. Bol vyvinutý pre kódovanie priezvisk. Soundex kódy sú štvorpísmenkové stringy zložené z jedného písmena a troch čísel. Soundex je bežne používaná technika, ktorá bola modifikovaná aj pre iné jazyky ako anglický.

Phonex - algoritmus je kombináciou dvoch metód, Soundex a Metaphone. Táto metóda dávala celkovo dobré výsledky, keď bola aplikovaná na mená v anglickom jazyku.



Obr. 2.4: Obrázok hodnôt Dice koeficientu pre slová s dĺžkou väčšou a rovnou ako 8 a s hodnotou Levenstein vzdialenosti 1

Metaphone - publikovaný Lawrencom Phillipsom v 1990 a **Double Metaphone**(upravený Metaphone), sú vhodné pre používanie pre anglické slová, nie iba mená. Tento algoritmus sa často využíva v programoch na kontrolu pravopisu.

NYSIIS - New York State Identification and Intelligence System algoritmus je abecedný algoritmus, ktorý je ľahko implementovateľný a ktorý využíva kanonický indexový kód podobný Soundexu. Avšak NYSIIS sa odlišuje od Soundexu zachovaním informácie o pozícii samohlások kódovaných slov pretransformovaním všetkých samohlások na písmeno A. NYSIIS metóda vracia čistý abecedný kód.

ISG algoritmus - Index of Similarity Group je hybridná technika kombinujúca abecedný a fonetický prístup. Porovnanie podobnosti je založené na Guthovej metóde. Metóda bola implementovaná Bouchardom a Pouyezom.

LIG algoritmus - (napr. LIG1, LIG2 a LIG3) je hybridný algoritmus, ktorý kombinuje foneticky a hláskovo založený prístup používajúc podobnosť meranú ako pravdepodobnosť, ktorú opísal Snae. Algoritmus je kombináciou troch porovnávacích metód: Levenshtein, ISG a Guth. LIG algoritmus má najviac presných odpovedí nad všetkými ostatnými spomenutými metódami.

My sme sa zamerali v tejto práci na hľadanie foneticky podobných slov algoritmom Soundex a Metaphone.

2.2.2 Soundex

Soundex je hašovací systém pre anglické slová. Z anglického slova sa vygeneruje kód, ktorý zhruba opisuje, ako sa dané slovo vyslovuje. Podobne znejúce slová budú mať rovnaké kódy. Soundex bol používaný v United States Census Bureau pre nájdenie podobných mien v „census records“. Bol vytvorený Robertom C. Russellom z Pittsburghu v Pennsylvánii.
Vygenerovaný kód sa skladá z písmena a troch čísel. Prvé písmeno slova je dané ako prvé písmeno kľúča. Ostávajúca časť sú čísla od 1 po 6, ktoré indikujú rôzne kategórie zvukov. Ak je slovo príliš krátke pre generovanie troch čísel, sú pridané nuly na koniec, podľa počtu koľko je treba. Ak generovaný kód je dlhší, zvyšné čísla sa vymažú.

Pravidlá:

- písmená B, F, P, V $\overset{nahradenie}{\longrightarrow}$ 1;

- C, G, J, K, Q, S, X, Z $\overset{nahradenie}{\longrightarrow}$ 2;
- D, T $\overset{nahradenie}{\longrightarrow}$ 3;
- L $\overset{nahradenie}{\longrightarrow}$ 4;
- M, N $\overset{nahradenie}{\longrightarrow}$ 5;
- R $\overset{nahradenie}{\longrightarrow}$ 6;
- A, E, H, I, O, U, W, Y, hlásky (a H, W, Y) $\overset{vynechanie}{\longrightarrow}$ ϵ;

Výnimky:

- ak sú písmená s rovnakým soundex číslom vedľa seba, tak to druhé v poradí sa vymaže. Napr. Pfizer $\longrightarrow$ Pizer, Sack $\longrightarrow$ Sac;
- ak sú dve písmená s rovnakým soundex číslom oddelené s 'H' alebo 'W', použije sa iba prvé písmeno. Napr. Ashcroft $\longrightarrow$ Ashroft.

Ako príklad uvedieme Washington = W252, Wu = W000, DeSmet = D253.

Iba 36.37% z výsledkov algoritmu Soundex bolo správnych, zatiaľ čo viac ako 60% správnych mien nebolo nikdy vrátených[17].

2.2.3 Metaphone

Metaphone algoritmus bol publikovaný v roku 1990, autor je Lawrence Philips. Je to algoritmus pre rozdelenie slov do skupín podľa anglickej výslovnosti. Algoritmus produkuje kľúče rôznej dĺžky ako výstup oproti Soundexu, ktorého kľúče majú fixnú dĺžku. Slová, ktoré majú podobnú výslovnosť, majú tie isté alebo podobné kľúče.

Pravidlá originálneho systému [6], [14]:

Tento algoritmus redukuje slová na 16 spoluhlások (*B X S K J T F H L M N P R 0 W Y*, kde 0 reprezentuje zvuk *„th“*). Dĺžka generovaných kódov je od 0 po 4.

Transformácie:

F, J, L, M, N, R $\longrightarrow$ sa zachovávajú;

$$\text{A} \longrightarrow \begin{cases} A & \text{ak} \quad \text{je na začiatku slova;} \\ \varepsilon & \text{inak;} \end{cases}$$

$$\text{E} \longrightarrow \begin{cases} E & \text{ak} \quad \text{je na začiatku slova;} \\ \varepsilon & \text{inak;} \end{cases}$$

$$\text{I} \longrightarrow \begin{cases} I & \text{ak} \quad \text{je na začiatku slova;} \\ \varepsilon & \text{inak;} \end{cases}$$

$$\text{O} \longrightarrow \begin{cases} O & \text{ak} \quad \text{je na začiatku slova;} \\ \varepsilon & \text{inak;} \end{cases}$$

$$\text{U} \longrightarrow \begin{cases} U & \text{ak} \quad \text{je na začiatku slova;} \\ \varepsilon & \text{inak;} \end{cases}$$

$$\text{B} \longrightarrow \begin{cases} B & \text{ak} \quad \text{na konci slova po písmene „m“ ako napr. v „dumb“;} \\ \varepsilon & \text{inak;} \end{cases}$$

$$\text{C} \longrightarrow \begin{cases} X & \text{ak} \quad \text{-cia- } \vee \text{ -ch-;} \\ S & \text{ak} \quad \text{-ci- } \vee \text{ -ce- } \vee \text{ -cy-;} \\ K & \text{inak;} \end{cases}$$

$$\text{D} \longrightarrow \begin{cases} J & \text{ak} \quad \text{v -dge- } \vee \text{ -dgy- } \vee \text{ -dgi-;} \\ T & \text{inak;} \end{cases}$$

$$\text{G} \longrightarrow \begin{cases} \varepsilon & \text{ak} \quad \text{v -gh- a nie na konci } \vee \text{ pred samohláskou v (-gn- } \vee \text{ v -gned-);} \\ J & \text{ak} \quad \text{(pred i } \vee \text{ e } \vee \text{ y) } \wedge \text{ nie je dvojité gg;} \\ K & \text{inak;} \end{cases}$$

$$\text{H} \longrightarrow \begin{cases} \varepsilon & \text{ak} \quad \text{je po samohláske a nenasleduje žiadna samohláska;} \\ H & \text{inak;} \end{cases}$$

$$\text{K} \longrightarrow \begin{cases} \varepsilon & \text{ak} \quad \text{po „c“;} \\ K & \text{inak;} \end{cases}$$

$$\text{P} \longrightarrow \begin{cases} F & \text{ak} \quad \text{pred „h“;} \\ P & \text{inak;} \end{cases}$$

$$\text{S} \longrightarrow \begin{cases} X & \text{ak} \quad \text{pred „h“ } \vee \text{ v -sio- } \vee \text{ -sia-;} \\ S & \text{inak;} \end{cases}$$

$$\text{Y} \longrightarrow \begin{cases} \varepsilon & \text{ak} \quad \text{nenasleduje samohláska;} \\ Y & \text{ak} \quad \text{nasleduje samohláska;} \end{cases}$$

$$\text{W} \longrightarrow \begin{cases} \varepsilon & \text{ak} & \text{nenasleduje samohláska;} \\ W & \text{ak} & \text{nasleduje samohláska;} \end{cases}$$

$$\text{T} \longrightarrow \begin{cases} X & \text{ak} & \text{-tia- } \vee \text{ -tio-;} \\ 0 & \text{ak} & \text{pred „h“ (th);} \\ \varepsilon & \text{ak} & \text{v -tch-;} \\ T & \text{inak;} & \end{cases}$$

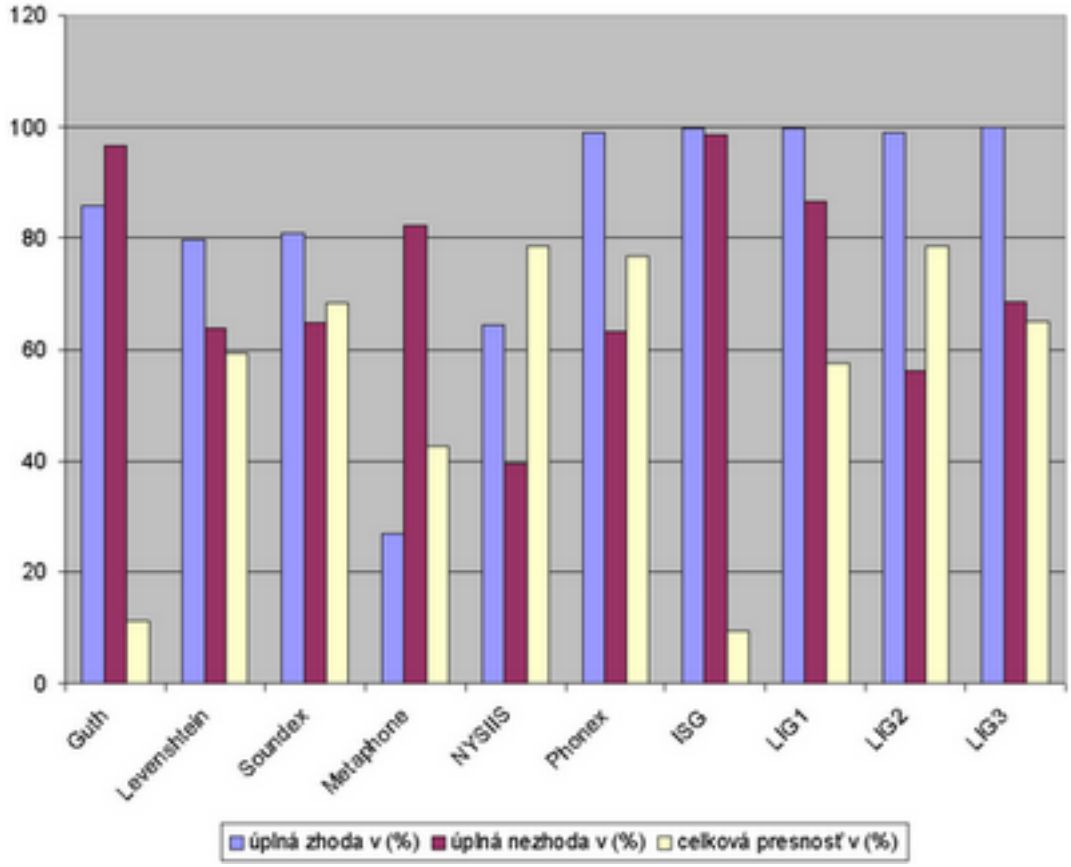
$$\text{V} \longrightarrow F;$$

$$\text{X} \longrightarrow KS;$$

$$\text{Z} \longrightarrow S.$$

Výnimky:

- počiatočné kn-, gn-, pn, ae- , wr- $\longrightarrow$ vyhoď prvé písmeno;
- počiatočné x- $\longrightarrow$ zmeň na „s“;
- počiatočné wh- $\longrightarrow$ zmeň na „w“.



Obr. 2.5: Porovnanie 10-tich metód hľadania fonetickej podobnosti slov [7]

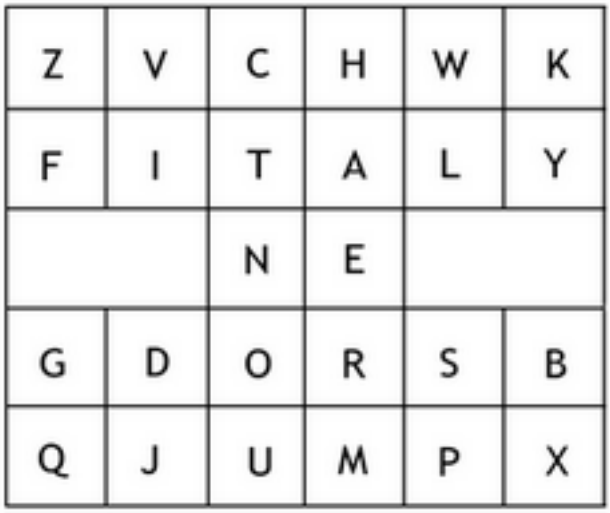
Kapitola 3

Alternatívna metóda písania textu

V tejto kapitole vychádzame z článkov [2] a [4]. V dnešnej dobe písanie textu predstavuje jednu z najčastejších požiadaviek užívateľa na počítač. Zahŕňa to písanie mailov, vypĺňanie formulárov, písanie poznámok, programov a mnoho iných požiadaviek. Klávesnica typu QWERTY sa stala štandardným nástrojom na uskutočnenie tejto požiadavky pre desktopové počítače. Avšak s prenikavým prechodom techniky k malým rozmerom sú trendom vo vývoji výpočtovej techniky mobilné počítačové zariadenia. Tieto zariadenia zahŕňajú PDA, počítače s tabletom alebo komunikačné zariadenia ako „pejdžre“ a mobilné telefóny. S rozšírením internetu pre tieto zariadenia je vkladanie textu (napr. chat, e-mail, alebo vkladanie URL adresy) jednou z prekážok v ich pokroku. To si žiadalo alternatívne riešenia písania textu.
Existuje mnoho typov metód vkladania textu pre mobilné zariadenia od zredukovania veľkosti fyzickej klávesnice, rozpoznávania ručne napísaného textu (handwriting recognition), rozpoznávania hlasu (voice recognition) až po virtuálnu klávesnicu. Každá z týchto metód má svoje kritické nedostatky v použití.

Fyzická klávesnica -prekážkou v jej používaní je jej veľkosť. Sú dva prístupy ako zredukovať jej veľkosť. Buď zmenšiť veľkosť každej klávesy, potom ale písanie na takejto klávesnici je pomalé a náročné kvôli malej veľkosti kláves. Alebo zdieľať jednu klávesu pre niekoľko písmen ako v mobilných telefónoch.

Ručné písanie - je veľmi ťažké písať čitateľne a pritom rýchlo. Autori článku [4] testovali rýchlosť písania textu týmto spôsobom na skúsených používateľoch Graffiti na PalmPilot PDA a výsledok bol 20 slov za



Obr. 3.1: Obrázok rozloženia FITALY

minútu, čo je oveľa pomalšie ako píšu skúsení pisári na fyzickej klávesnici.

Rozpoznanie hlasu - napriek pokroku tejto techniky, efektívna rýchlosť vkladania textu s plynulým rozpoznávaním slov z hlasu bola stále nižšia, ako pri ostatných klávesniciach.

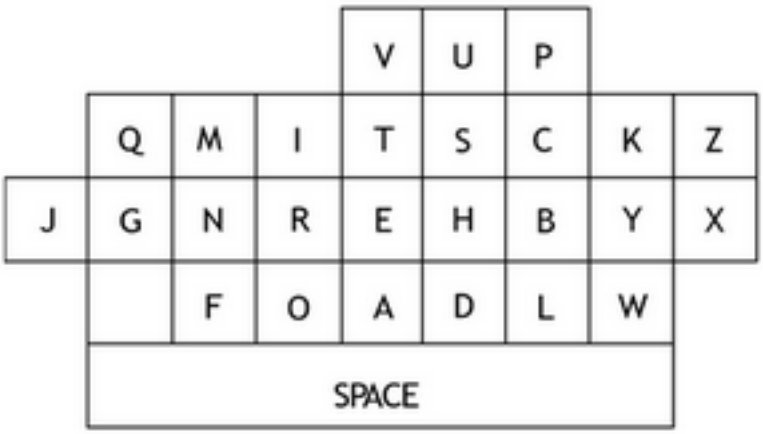
Virtuálna klávesnica - klávesy sú zobrazené na obrazovke, text sa vkladá perom alebo prstom. Najpoužívanejšie klávesnice dnes prevzali tradičnú QWERTY formu. Lenže QWERTY usporiadanie je neefektívne pre písanie na virtuálnej klávesnici, pretože bolo vytvorené pre písanie dvomi rukami, zatiaľ čo pre písanie na virtuálnej klávesnici sa používa jedna ruka.

Najznámejšie typy usporiadania klávesníc:

1. **pre virtuálne klávesnice**

   FITALY klávesnica - je komerčný produkt firmy Textware Solutions. Princíp dizajnu tejto klávesnice spočíva v stredovom umiestnení najčastejšie používaných kláves, dvoch medzerníkov s veľkosťou dvakrát väčšou ako ostatné klávesy a frekvencii výskytu digrafov.

   Chubon klávesnica - je rozloženie navrhnuté pre ľudí s ohraničenými možnosťami pohybu. Najviac používané písmená sú umiestnené



Obr. 3.2: Obrázok rozloženia Chubon

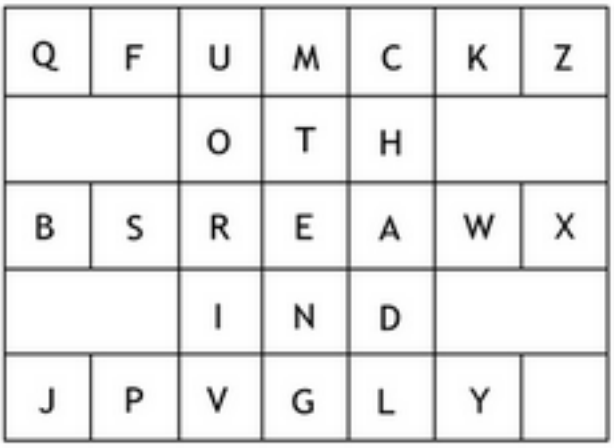
okolo medzerníka v strede klávesnice. Zvyšné písmená sú umiestnené okolo obvodu [16].

OPTI klávesnica - bola navrhnutá MacKenzieom a Zhangom. Najprv umiestnili 10 najfrekventovanejších písmen do centra klávesnice, potom k nim priradili 10 najfrekventovanejších digrafov a nakoniec umiestnili zvyšné klávesy. Neskôr vytvorili klávesnicu veľkosti 5×6 s názvom **OPTI 2**. V oboch druhoch OPTI klávesnice sú rozdistribuované štyri medzerníky. Používateľ si môže vybrať hociktorý.

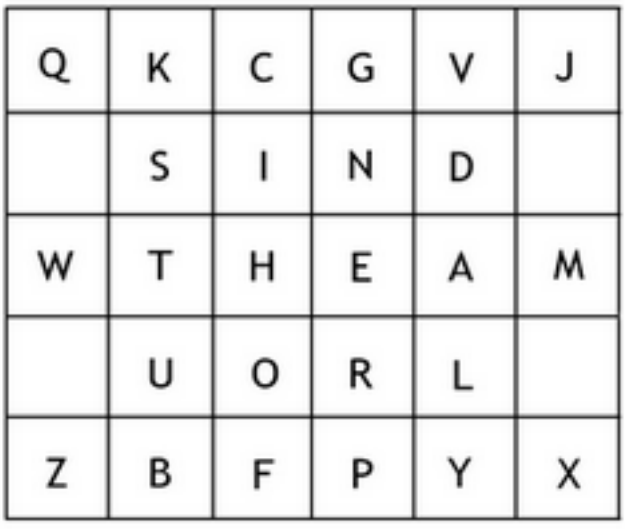
Lewis–Kennedy–LaLomia klávesnica - namiesto rôznych heuristík pre vytvorenie usporiadania Lewis, Kennedy a LaLomia použili viac systematickú metódu pri ich návrhu. Najprv vytvorili symetrickú maticu relatívnej frekvencie neusporiadaných digrafov anglického jazyka, a potom túto maticu analyzovali s „Pathfinder network-definition program“ pre vytvorenie minimálne spojenej siete, ktorá tvorila základ ich dizajnu. Táto metóda neobsahovala všetky spojenia digrafov, takže efektívnosť písania na takejto klávesnici nie je dobrá.

2. **pre fyzické klávesnice**

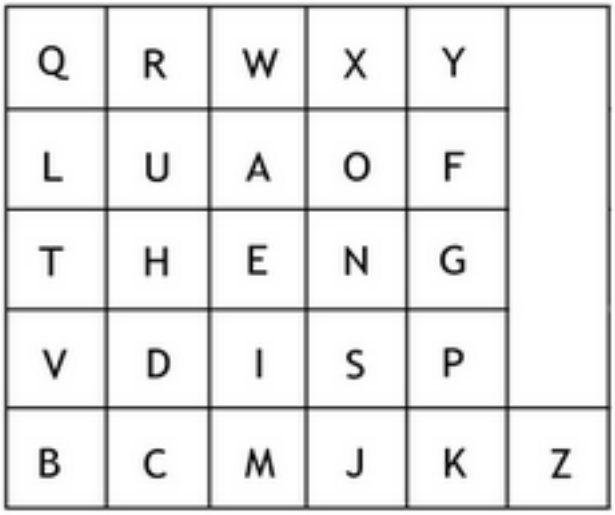
Dvorak klávesnica - toto rozmiestnenie redukuje množstvo pohybu potrebného pre písanie bežných anglických slov. Existujú tri Dvorak rozmiestnenia: jedno pre používateľov píšucich dvomi rukami, druhé pre používateľov píšucich iba pravou rukou a tretie pre



Obr. 3.3: Obrázok rozloženia OPTI



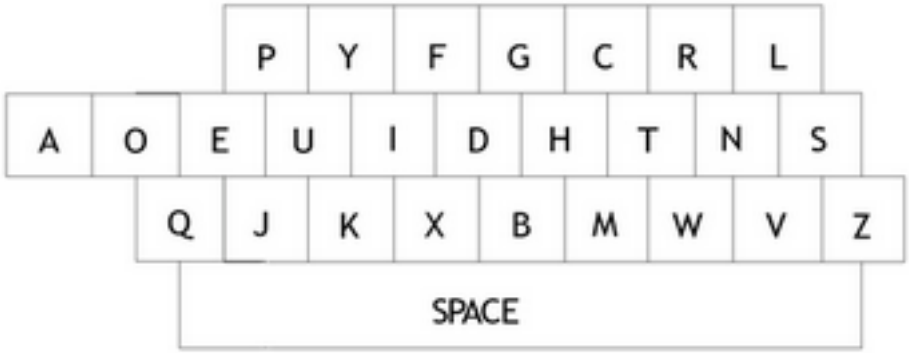
Obr. 3.4: Obrázok rozloženia OPTI 2



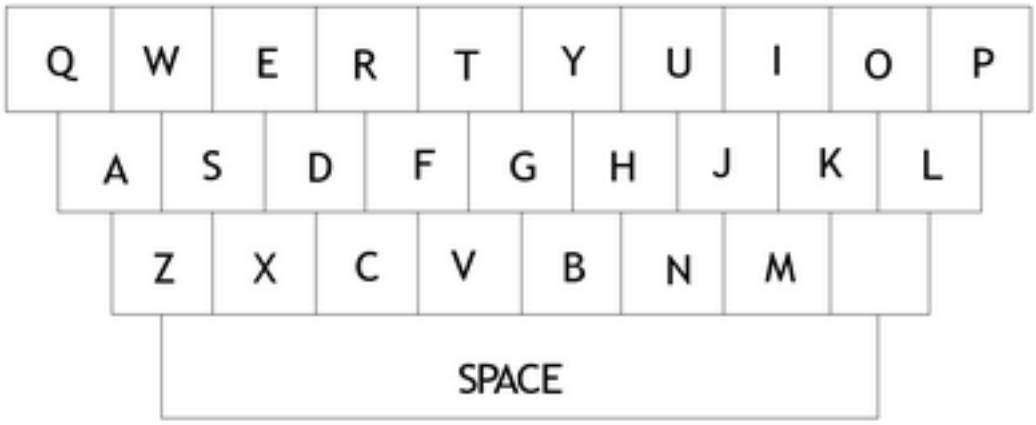
Obr. 3.5: Obrázok rozloženia Lewis–Kennedy–LaLomia

používateľov píšucich ľavou rukou. Dvorak rozmiestnenie pre používateľov píšucich dvomi rukami má najpoužívanejšie spoluhlásky umiestnené na pravej strane spodného riadka klávesnice a samohlásky na ľavej strane. Zastáncovia tohto rozloženia tvrdia, že toto rozloženie zvyšuje rýchlosť písania a je oveľa pohodlnejšie ako štandardné qwerty rozloženie [16].

QWERTY klávesnica bola vyvinutá Christopherom Sholesom, Carlosom Gliddenom, a Samuelom W. Souleom v roku 1868 kvôli mi-



Obr. 3.6: Obrázok rozloženia Dvorak pre písanie dvoma rukami



Obr. 3.7: Obrázok rozloženia QWERTY

nimalizovaniu mechanického rušenia. Rozloženie kláves bolo usporiadané tak, aby veľa susedných párov písmen (digrafov) bolo rozložených na opačných stranách klávesnice. Tento dizajn je vhodný pre písanie dvomi rukami, pretože umožňuje striedanie ľavej a pravej ruky [15].

V tejto práci sme sa zamerali na metódu vkladania textu cez virtuálnu klávesnicu ShapeWriter, pretože je to zatiaľ najrýchlejší spôsob vkladania textu do mobilných zariadení a optimalizáciu rozloženia klávesnice pre slovenský jazyk. A na optimalizáciu fyzickej klávesnice pre mobilné telefóny, kde jedna klávesa obsahuje niekoľko písmen, aby bolo čo najmenej kolízií slov pri písaní na takejto klávesnici formou T9.

3.0.4 ShapeWriter

Shape writing, inak známy ako shorthand aided rapid keyboarding (SHARK), je metóda písania textu vyvinutá pre umožnenie rýchlejšieho písania textu užívateľom najmä na mobilných telefónoch oproti predchádzajúcim spôsobom (T9). Okrem mobilných telefónov môže byť táto metóda použitá aj pri písaní textu do desktopového počítača za použitia externého tabletu pripojeného k počítaču pre tých, ktorí potrebujú efektívnu alternatívu písania dvomi rukami na fyzickej klávesnici. Narozdiel od ručného písania, ShapeWriting je definovaný a používaný s grafickou klávesnicou.

Základný princíp:

ShapeWriting system (ShapeWriter) zobrazuje používateľovi grafickú klávesnicu. Namiesto stláčania klávesnice, ktorá zodpovedá danému písmenu v slove, užívateľ kĺže/pohybuje perom na grafickej klávesnici nad všetkými písmenami slova postupne, podľa ich poradia v danom slove.

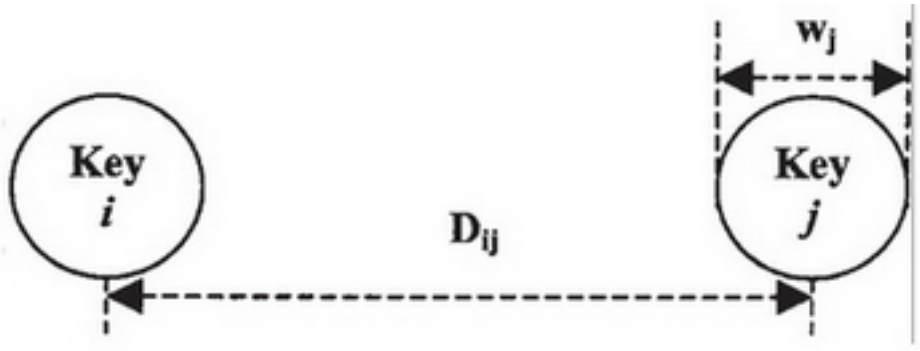
Ideálna cesta je nazvaná **sokgraph** - shorthand on keyboard as a graph. Je to súvislá cesta formovaná sériovým pospájaním stredových bodov kláves na grafickej klávesnici pre písmená slova. Pre shape writing je grafická klávesnica matica znakov uložená v tradičnej QWERTY alebo v optimalizovanej forme.
Pre napísanie slova na ShapeWritery používateľ ťahá perom po klávesnici, daný ťah aproximuje sokgraph slova. Keď sa zastaví ťah perom, ShapeWriting system porovná ťah perom na klávesnici so všetkými generovanými ťahmi v lexikóne a vráti najbližšiu zhodu. ShapeWriting toleruje niektoré chyby ako napr. chvenie ruky [3].

3.0.5 Optimalizovanie klávesnice

Bežne používané grafické klávesnice majú QWERTY rozloženie. Tento dizajn je vhodný pre písanie dvomi rukami, pretože umožňuje striedanie ľavej a pravej ruky. Keby sa použila klávesnica s hrotom, znamenalo by to, že hrot by sa pohyboval dozadu a dopredu častejšie a na väčšie vzdialenosti, ako je potrebné.
Ďalším riešením by mohlo byť abecedné usporiadanie písmen na klávesnici s hrotom. Noví používatelia by ľahko našli klávesy, avšak efektívnosť tohto rozloženia tiež nie je dobrá.

Pre minimalizovanie pohybu po klávesnici sa musia brať do úvahy dva faktory. Jeden faktor je prechodová frekvencia z jedného písmena na druhé - frekvencia jednotlivých digrafov. Druhým je relatívna vzdialenosť medzi klávesami. Cieľom je usporiadať klávesy tak, aby dĺžka krivky, ktorá prislúcha k danému slovu bola čo najkratšia. To znamená, že najfrekventovanejšie klávesy majú byť umiestnené v strede klávesnice a klávesy najfrekventovanejších digrafov budú bližšie k sebe. Nakoniec sa pripoja písmená najmenej frekventovaných digrafov.

Obr. 3.8: Obrázok rozloženia QWERTY



Obr. 3.9: Aplikovanie Fittsovho zákona na tlačidlá klávesnice

3.0.6 Fitts-Digraph model virtuálnej klávesnice

Fittsov zákon je model ľudského pohybu v interakcii človek-počítač, ktorý predpovedá, že čas potrebný pre rýchly pohyb do cieľovej oblasti je funkcia vzdialenosti do cieľa a veľkosti cieľa. Fittsov zákon má rôzne formulácie. Jednou z formulácií je Shannonova formulácia pre pohyb pozdĺž jednej dimenzie [18]:

$$MT = a + b \log_2 \left(\frac{D}{W} + 1 \right)$$

kde MT je priemerný čas potrebný na vykonanie pohybu, a reprezentuje začiatočný/koncový čas zariadenia, b je prirodzená rýchlosť zariadenia. Tieto konštanty môžu byť určené experimentálne. D je vzdialenosť zo začiatočného bodu do stredu cieľa, W je šírka cieľa meraná pozdĺž osi pohybu. W môžeme vnímať aj ako povolenú toleranciu chyby vo finálnej pozícii, pretože konečný bod pohybu musí byť vo vzdialenosti $\pm W/2$ od stredu cieľa [18].

MacKenzie a jeho kolegovia ako prví použili kvantitatívny prístup k modelovaniu virtuálnej klávesnice. Ich model predpokladá výkon používateľov ako súčet Fittsovho zákona času pohybu (Fitts law movement times) MT medzi všetkými digrafmi, vážené sú frekvenciou výskytu digrafov. Použitie Fittsovho zákona umožňuje odhadnúť výkon tým, že dáva porovnanie rýchlosti písania.
V [5] boli pre porovnanie výsledkov zvolené $a = 0, b = 1/4.9$. Ďalej uvažovali Fittsov index výkonu (Fitts index of performance) $IP = 4.9$ bitov za sekundu(bps).
Podľa Fittovho zákona a distribúcie diagrafov definovali priemerný čas po-

hybu po klávesnici ako Fitts-Digraph energiu:

$$e = \sum_{i=1}^{27}\sum_{j=1}^{27} MTP_{ij} = \sum_{i=1}^{27}\sum_{j=1}^{27} \frac{P_{ij}}{IP}\left[\log_2\left(\frac{D_{ij}}{W_j}+1\right)\right]$$

kde $\log_2\left(\frac{D_{ij}}{W_j}+1\right)$ je Fittsov zákon výpočtu času pre pohyb hrotu z klávesy i do klávesy j pre danú vzdialenosť D_{ij} a veľkosť klávesy W_j. IP je Fittsov index výkonu. P_{ij} je frekvencia diagrafu ij v anglických textoch. Jej hodnota bola vypočítaná z textov, ktoré nazbierali na internete z chatovacích miestností.

3.0.7 Fitts-Digraph energia a algoritmus Metropolis

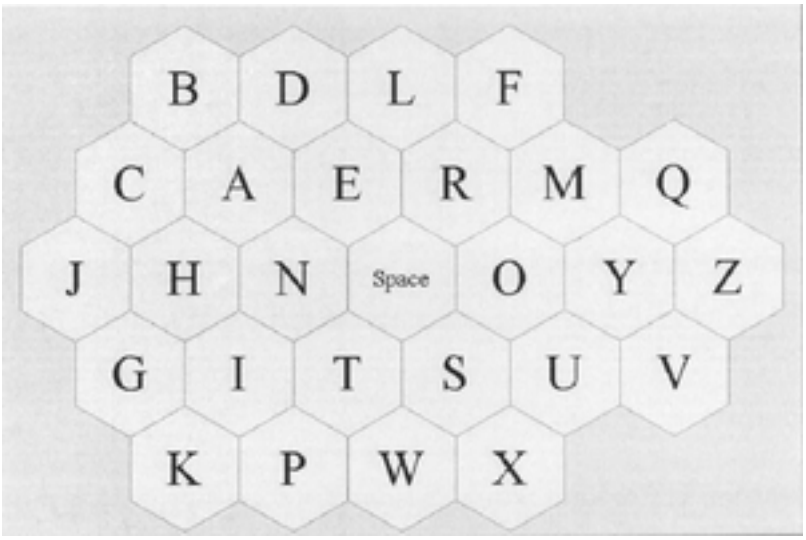
Metropolis algoritmus sa používa najčastejšie v štatistickej fyzike. Ak definujeme predchádzajúcu rovnosť ako Fitts-Digraph energiu, problém dizajnu efektívnej klávesnice sa stane ekvivalentný s hľadaním štruktúry molekuly (klávesnica) v stabilnom nízkoenergetickom stave v interakcii so všetkými atómami (klávesy).
Aplikovaním tohto prístupu autori článku implementovali softvér, ktorý náhodne „kráčal“ po klávesnici. V každom kroku zobral algoritmus klávesu a hýbal ňou v náhodnom smere o náhodnú vzdialenosť na dosiahnutie novej konfigurácie. Úroveň Fitts-Digraph energie v novej konfigurácii bola vypočítaná na základe predchádzajúcej rovnice. Nová konfigurácia bola použitá ako štartovacia pozícia pre ďalšiu iteráciu závislej na nasledujúcej Metropolis funkcii:

$$P_{O\to N} = \begin{cases} e^{-\frac{\triangle E}{kT}} & \text{ak} \quad \triangle E > 0 \\ 1 & \text{ak} \quad \triangle E \leq 0 \end{cases}$$

$P_{O\to N}$ je pravdepodobnosť zmeny z konfigurácie O(old) na konfiguráciu N(new), $\triangle\, E$ bola zmena energie, k koeficient, T „teplota“, ktorá môže byť interaktívne nastavená. Použitie tejto rovnosti robí metropolis metódu lepšiu, pretože hľadanie sa nie vždy hýbe smerom k lokálnemu minimu. Cieľom tohto algoritmu bolo usporiadanie atómov tak, aby celková energia medzi atómami bola v minime.

ATOMIK - Alphabetically Tuned and Optimized Mobile Interface Keyboard[1]

Je virtuálna klávesnica optimalizovaná tromi črtami:



Obr. 3.10: Obrázok rozloženia ATOMIK

- Na tvorbu tohto rozloženia bol využitý algoritmus Metropolis s Fitts-Digraph energiou, preto má vyššiu efektívnosť pohybu ako ostatné druhy rozloženia klávesníc.
- Schéma bola abecedne upravená. Je tu tendencia, že písmená od A po Z sú usporiadané z horného ľavého rohu do dolného pravého rohu klávesnice, ale nie striktne v abecednom poradí. Toto uľahčuje hľadanie písmen nováčikom.
- Spojenie niektorých písmen najčastejšie používaných slov. Veľa bežných slov alebo častí slov ako napr. „ing“ sú úplne spojené.

Kapitola 4

Návrh implementácie programu

V tejto kapitole sa budeme podrobnejšie venovať zložitosti algoritmov, ktoré používame v programe a čistení dát (slov) získaných z textov. Pri tvorbe programu sme použili objektovo orientované programovanie a jazyk Java.

4.1 Vytvorenie a čistenie slovníka

Pri tvorbe slovníka sme sa snažili zahrnúť široký okruh slov, vložili sme biblický text, literárne diela rôznych autorov a s rôznou tématikou, odborné texty, ale aj texty s tématikou športu, módy a zdravia. Texty na vytvorenie slovníka sme sťahovali zo stránok:
http://diplomovka.sme.sk,
http://www.gutenberg.org/wiki/Main_Page,
http://www.poznanie.sk/knihy/,
http://zlatyfond.sme.sk,
http://www.vydavatelstvo-f.sk,
http://www.newscientist.com,
http://www.o-bible.org/download/bbe.txt.
V textoch, ktoré sme stiahli sú chyby, ktoré sa dajú opraviť, ak máme (okrem algoritmu na túto kontrolu) aj nejaký existujúci slovník slov, ktorý obsahuje korektne napísané slová, a s ktorým môžeme porovnávať slová. Avšak, takýto slovník nemáme, takže slová získavame iba postupným parsovaním slov z daných textov. Pritom sa odstraňujú iba úvodzovky, čísla, čiarky a pod., takže výsledkom by mali byť iba slová. Ak sú dve slová oddelené pomlčkou, spoja sa a vznikne z nich jedno slovo (tento postup som zvolila preto, lebo predpokladám, že sa vyskytuje viac slov rozdelených na konci riadkov ako samostatných slov píšucich sa s pomlčkou, napr. sem-tam).

4.2 Zložitosť algoritmov

Väčšina algoritmov, ktoré sú použité, má časovú zložitosť $O(n)$, kde n je dĺžka slova. Je to najmä preto, lebo vytvárajú nejaký kód z daného slova, takže algoritmus prejde všetkými písmenami slova a na základe toho vygeneruje kód. Metóda writeT9, ktorá k danému slovu vypíše jeho číselný kód, ktorý má pri písaní technológiou T9 na mobilných telefónoch, má časovú zložitosť $O(n)$ v najlepšom aj najhoršom prípade. Metódu metaphone som získala prepísaním pôvodného algoritmu napísaného v jazyku basic do jazyku Java [14], [6]. Jej časová zložitosť je v najlepšom prípade $O(4)$ a v najhoršom je $O(n)$. Metóda soundex má najlepšiu časovú zložitosť $O(4)$ a najhoršiu $O(n)$. Metóda ShapeWriter, ktorá pre dané slovo vypíše všetky písmená v presnom poradí ako za sebou idú pri postupnom ťahaní pera po virtuálnej klávesnici, má zložitosť $O(n)$.
Slovník je uložený v hašovacej tabuľke, takže vyhľadanie jedného konkrétneho slova je v čase $O(1)$.
Vytvorenie slovníka, ktorý obsahuje k slov má časovú zložitosť $O(k) * O(5n)$, čiže $O(kn)$. Vyhľadávanie všetkých palindromov, anagramov, kolidujúcich slov v T9 a ShapeWritery, ale aj vyhľadávanie slov, ktoré sú anagramy k jednému konkrétnemu slovu, vyhľadávanie kolidujúcich slov v T9 a ShapeWritery ku konkrétnemu slovu, má časovú zložitosť $O(k)$. Vyhľadávanie podobných slov algoritmom Levenshtein a Dice trvá $O(k^2) * c$, kde c je čas potrebný na vypočítanie vzdialenosti dvoch slov jedným z algoritmov Dice alebo Levenshtein. Ak sa podobnosť slov počíta algoritmom Levenshtein, potom je c rovné $O(n^2)$, ak Dice, tak $O(n)$.

Kapitola 5

Zistenia

5.1 Návrh virtuálnej klávesnice pre ShapeWriter

Pri návrhu virtuálnej klávesnice pre slovenský jazyk sme postupovali podľa publikácie [5] a [8]. Najprv sme spočítali výskyt všetkých písmen v danom slovníku, a potom výskyt všetkých bigramov. Na základe týchto zistení sme zostavili počiatočnú klávesnicu veľkosti 7×7. Voľné miesta v klávesnici sme nahradila jednotkami. Klávesy na virtuálnej klávesnici sú v tvare 6-uholníkov, takže každá klávesa má šiestich susedov a celá klávesnica uložená vo forme Stringu je jeden z parametrov metódy energy, ktorá počíta energiu klávesnice na základe Fittovho zákona. Konštantu k a teplotu T zanedbávame, takže pravdepodobnosť zmeny z konfigurácie O na konfiguráciu N je:

$$P_{O \to N} = \begin{cases} e^{\triangle E} & \text{ak} \quad \triangle E > 0 \\ 1 & \text{ak} \quad \triangle E \leq 0 \end{cases}$$

Jej ďalším parametrom je šírka klávesnice. Pri tomto návrhu sme nezohľadnili 3. pravidlo, ktoré bolo použité na vytvorenie klávesnice ATOMIK, a to je, že klávesy nie sú usporiadané abecedne od ľavého horného rohu po pravý dolný a časti slov, ktoré sa najčastejšie vyskytujú, nie sú pospájané.
Algoritmus metropolis sme spočiatku nechali opakovať 100-krát, potom 5000-krát, ale stále nedával dobré výsledky (buď boli horšie ako energia rozloženia QWERTY alebo len o máličko lepšie). Potom sme zmenili cyklus for na while a nechali opakovať, kým energia klávesnice nebola pod stanovenou hodnotou. Rozloženie klávesnice typu QWERTY má energiu **0.6834391372653876**. Zatiaľ najlepšie výsledky sú pre rozloženie 11fxwq111mtsuc11dikl111ze vo11y narp11gjhb11111111 s energiou **0.5568333635330525**.

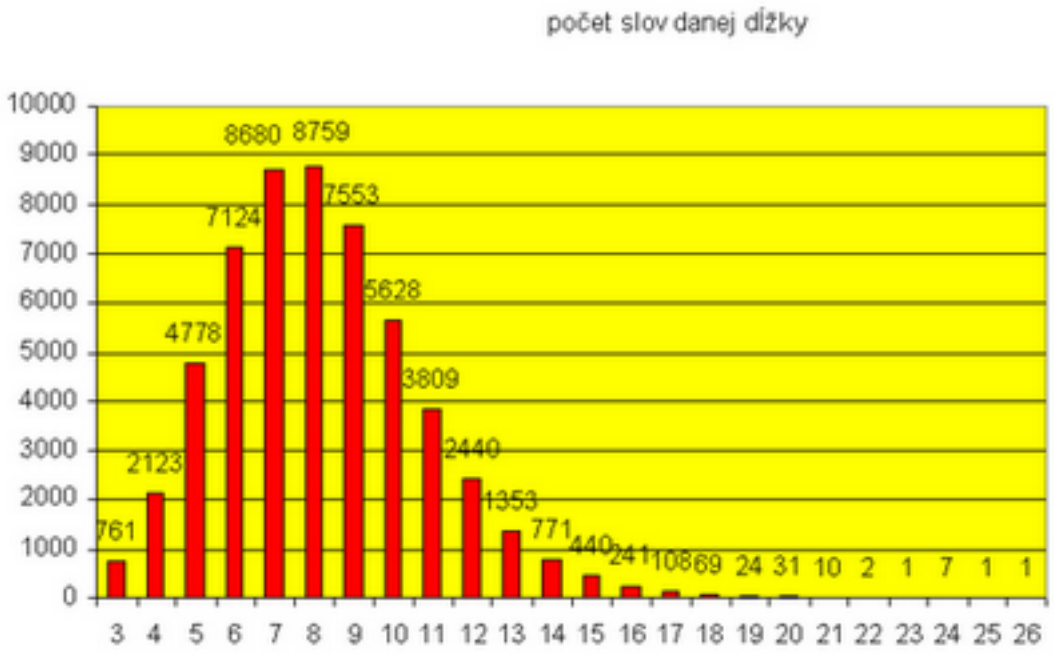
5.2 Návrh rozloženia písmen v T9 pre slovenský jazyk

Jedným z cieľov tejto práce bolo nájsť najefektívnejšie rozloženie písmen k číslam v T9. To ale nie je možné, pretože by sme museli porovnať efektívnosť 26! rozložení, čo je časovo aj priestorovo veľmi náročné. Preto sme pri tomto hľadaní použili heuristickú metódu, v ktorej sme sa inšpirovali predchádzajúcimi myšlienkami z tvorby efektívnej virtuálnej klávesnice:
Najprv sme našli 8 najčastejších začiatočných písmen (p - 8134, s - 5264, n - 5091, z - 4502, v - 4414, o - 3347, k - 2777, d - 2683), rozvrhli sme ich k daným číslam náhodne, potom sme našli 8 najčastejšie používaných písmen (ak sa nachádzalo nejaké písmeno z najčastejších ôsmich začiatočných písmen medzi ôsmimi najčastejšie používanými, vynechali sme ho a nahradili písmenom, ktoré išlo za ním v poradí) a rozdistribuovali k prvým ôsmim najčastejšie sa vyskytujúcim začiatočným písmenám (a - 44877, e - 38846, i - 35066, r - 23269, t - 22890, l - 20470, u -18 667, c - 16 881). Zvyšné písmená sme náhodne umiestňovali. Takto náhodne zostavenú klávesnicu sme nechali prejsť algoritmom metropolis, pokým nenájde lepšie rozloženie ako bolo to predošlé: algoritmus si náhodne vybral dve písmená a vymenil ich pozície, ak nové rozloženie písmen malo lepšiu mieru efektívnosti, zmeny v usporiadaní písmen sa uložili a algoritmus skončil. Inak sa vygenerovali dve náhodné čísla z intervalu (0,1). Ak bolo prvé vygenerované číslo menšie ako druhé, zmeny v usporiadaní sa uložili, inak sa pokračovalo od začiatku náhodným výberom dvoch písmen a ich výmene na pôvodnom rozložení.
Ako mieru efektívnosti rozloženia písmen sme si zvolili nasledovný vzťah:

Definícia 5.2.1 *Nech* Π *je množina všetkých permutácií čísel 2,3,4,5,6,7,8,9 takých, že 2,3,4,5,6,8 sa vyskytujú 3-krát a 7,9 4-krát. Nech* $\pi \in \Pi$ *je ľubovoľná permutácia tejto množiny, nech*
H_i*, kde* $i = 1, 2, \ldots, n$ *je trieda ekvivalencie množín kolízií, ktorých veľkosť je* i *a nech* $k = \sum_{i=1}^{n} |H_i|$ *je počet všetkých kolízií, potom miera efektívnosti rozloženia písmen na permutáciu* π *je*

$$v_\pi = \sum_{i=0}^{n} \frac{i * |H_i|}{k}.$$

V ideálnom prípade by bola miera efektívnosti rovná 1, ale to je nemožné, pretože je viac slov ako počet rôznych permutácií čísel 2-9 dĺžky daných slov. Efektívnosť rozloženia klasickej T9 pre slovenský jazyk je
1.076064610866373. Zatiaľ najlepší výsledok v efektívnosti som našla pri



Obr. 5.1: Obrázok počtu dĺžok slov

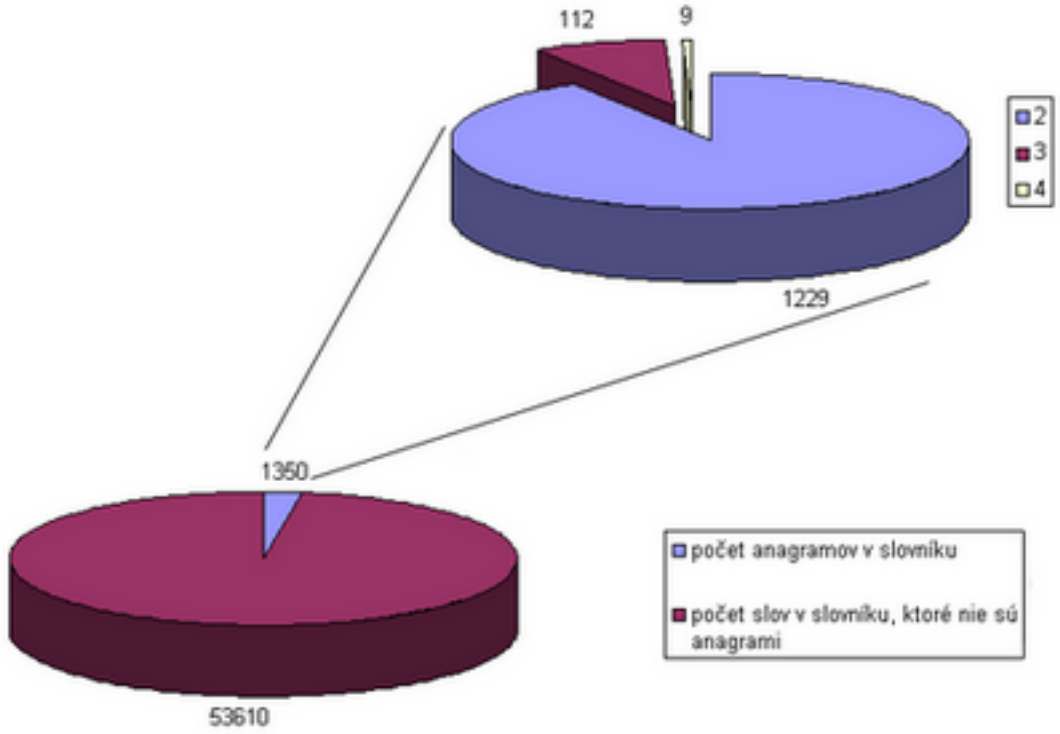
permutácii 638722478793763428549265 59, to znamená, že písmená a, á sa nahradia 6; b 3; c, č 8; d, ď 7; atď.. Miera efektívnosti takéhoto rozloženia je 1.0686993213681528.

5.3 Slovník slovenských slov s diakritikou

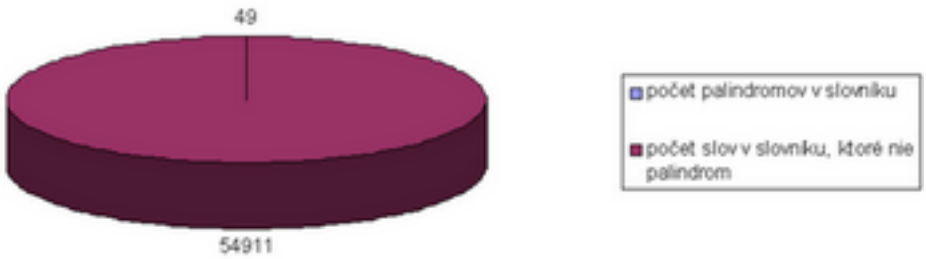
Slovník obsahuje 54 960 slov. Priemerná dĺžka slov je 8. Najdlhšie slovo má dĺžku 26 - *hypertriacylglycerolémiách*.

5.3.1 Anagramy

Najväčšiu množinu tvorí skupina anagramov s veľkosťou 2, je ich 1229, po nej nasleduje skupina veľkosti tri, tých je 112 *{ktosi, kosti, kosit}*. Posledná, najmenej zastúpená je množina s veľkosťou podmnožín veľkosti 4, tých je 9 *{kroja, rojka, jarok, okraj}*.



Obr. 5.2: Obrázok pomeru počtu všetkých anagramov k počtu všetkých slov slovníka slovenských slov s diakritikou, ktoré nie sú anagramy a počtu veľkostí množín anagramov



Obr. 5.3: Obrázok pomeru počtu všetkých palindromov k počtu všetkých slov slovníka slovenských slov s diakritikou, ktoré nie sú palindromy

5.3.2 Palindromy

Počet slov, ktoré sa rovnako čítajú z predu aj zozadu je 49. Napr. *mottom, kolok, lapal, mám, júj, krk, zaraz, ťahať, volov, dvd, oko, level.*

5.3.3 T9

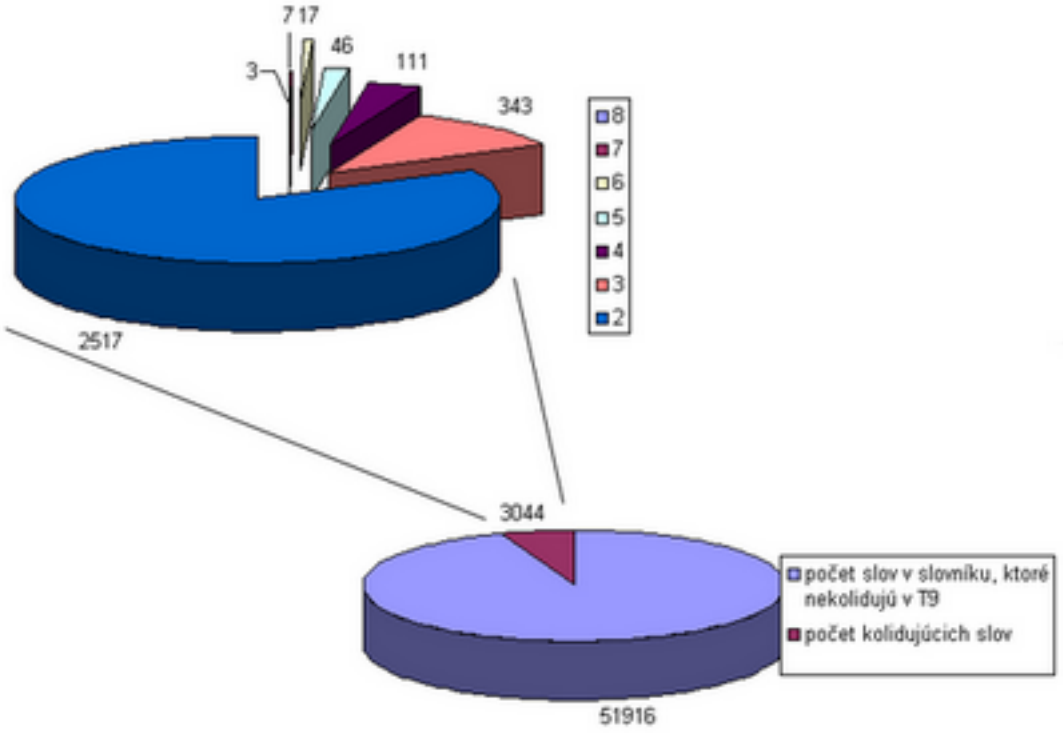
Najväčšia veľkosť podmnožiny kolidujúcich slov je 8, sú 3, napr. *{pasú, rásť, rasu, rast, päsť, pásť, páru, pašu}*. Množín s mohutnosťou 7 je 7, 6 je 17 *{vôľou, tokov, voľnú, volov, vojov, vojnu}*, 5 je 46 *{vine, vhod, uhne, time, víne}*, 4 je 114, 3 je 343 a nakoniec podmnožín s najmenšou mohutnosťou 2 je najviac - 2517. Podmnožiny s počtom prvkov 1 vynechávam.

5.3.4 ShapeWriter

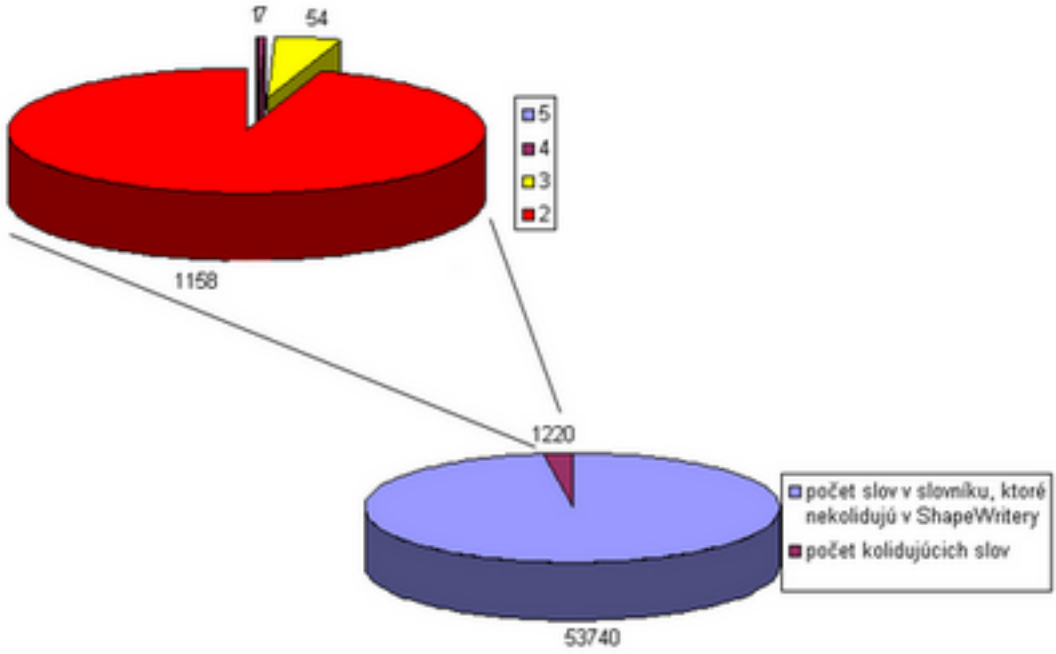
Kolízií pri písaní slov pomocou ShapeWritera je oveľa menej. Najväčšia podmnožina má mohutnosť 5 *{mašľa, masla, mála, malá, mala}* je jedna. Ďalej sú podmnožiny veľkosti 4 *{nákladu, náladu, nadhľadu, náhľadu}*, tých je 7, potom 3, je ich 54 a nakoniec, opäť podmnožín s mohutnosťou 2 je najviac - 1158 *{času, času}*.

5.3.5 Diakritika

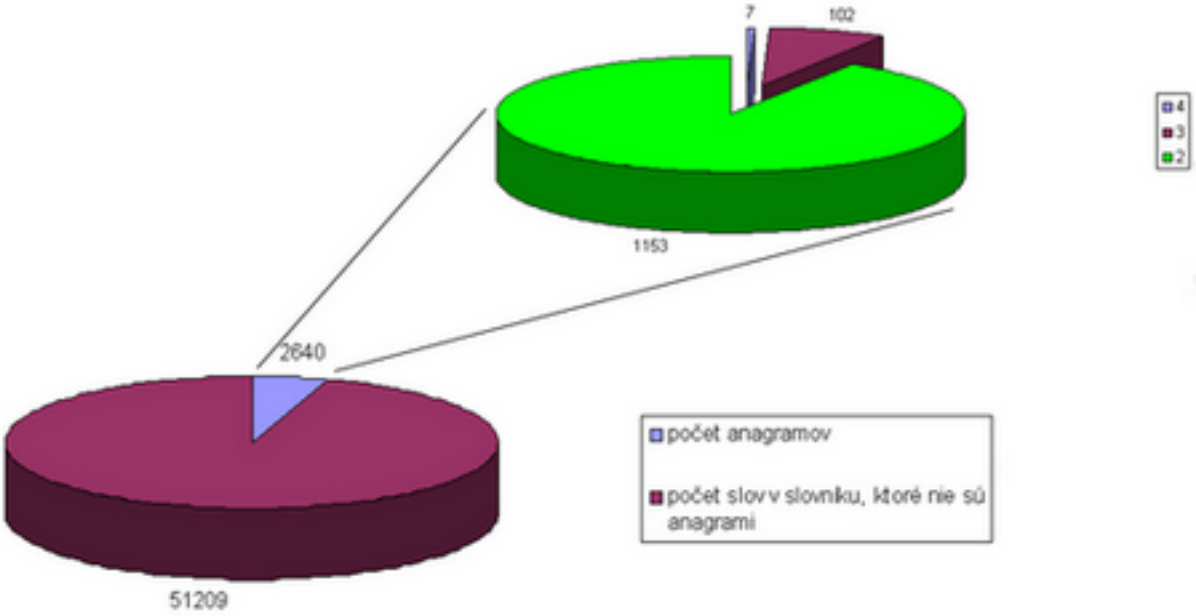
Po odstránení diakritiky z tohto slovníka sa znížil počet slov v ňom o 665. Napríklad slovo *čele* sa zmenilo na *cele* a už má iný význam alebo slovo *rozkáž* na *rozkaz*, *nápadne* na *napadne* , *samozrejmé* so slovom *samozrejme*.



Obr. 5.4: Obrázok pomeru počtu všetkých kolízií v T9 k počtu nekolidujúcich slov slovníka slovenských slov s diakritikou a pomer veľkostí množín kolízií v T9



Obr. 5.5: Obrázok pomeru počtu všetkých kolízií v ShapeWritery k počtu nekolidujúcich slov slovníka slovenských slov s diakritikou a pomer veľkostí množín kolízií v ShapeWritery



Obr. 5.6: Obrázok pomeru počtu všetkých anagramov k počtu všetkých slov slovníka slovenských slov bez diakritiky, ktoré nie sú anagramy a počtu veľkostí množín anagramov

5.4 Slovník slovenských slov bez diakritiky

Slovník obsahuje 53 849 slov.

5.4.1 Anagramy

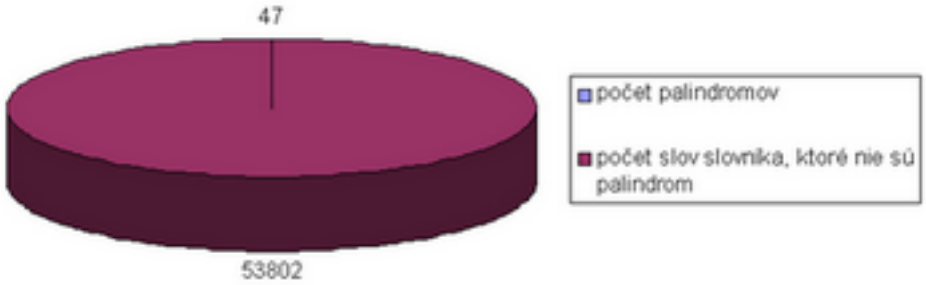
Počet podmnožín anagramov veľkosti 4 je 7 *{vtisla, stavil, vstali, vlasti}*, ďalej je množina s veľkosťou 102, ktorej podmnožiny majú veľkosť 3, posledná a najväčšia je množina anagramov veľkosti 2 s mohutnosťou 1153.

5.4.2 Palindromy

Počet palindromov je 47. Napr. *zaraz, tahat, odo, oko, ono, mam, mecem.*

5.4.3 T9

Najväčšia veľkosť podmnožiny kolidujúcich slov je 6. Takýchto podmnožín je 11, napr. *{kasnu, jasot, jasnu, jarou, jarnu, larmu}* Podmnožín veľkosti 5 je 23 *{nasu, masu, mast, maru, mapu}*, 4 je 79 *{vaha, ucia, ucha, taha}*,



Obr. 5.7: Obrázok pomeru počtu všetkých palindromov k počtu všetkých slov slovníka slovenských slov bez diakritiky, ktoré nie sú palindromy

veľkosti 3 je 274. Najviac podmnožín je opäť veľkosti 2 - 1836.
Miera efektívnosti pre slovenský jazyk bez diakritiky je 1.054312285854136.

5.4.4 ShapeWriter

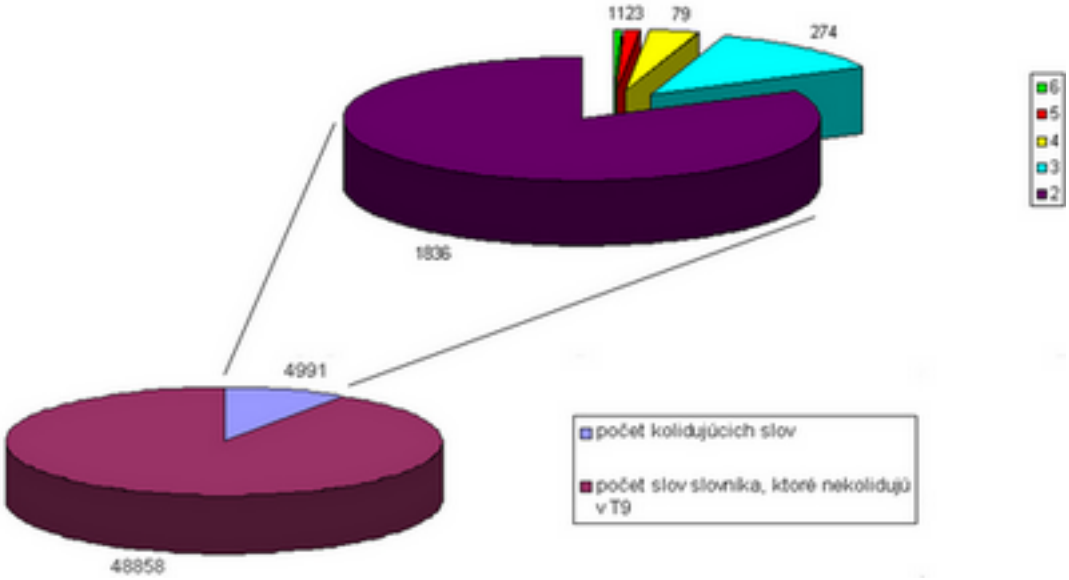
Najväčšia veľkosť podmnožín kolidujúcich slov metódou ShapeWriter je 4 s počtom podmnožín 2 *{nahladu, nadhladu, naladu, nakladu}*, *{per, puer, poter, pier}*. Po nej sú podmnožiny s veľkosťou 3, tých je 13, napr. *{nahlady, nalady, naklady}*, *{typ, top, tip}*, *{odstupte, odstupite, odstupe}*. Podmnožín kolidujúcich slov veľkosti 2 je 147, čo je veľmi výrazný rozdiel v porovnaní s počtom kolidujúcich slov veľkosti 2 v slovníku slovenských slov s diakritikou, kde je 1158 takýchto podmnožín. Medzi podmnožiny mohutnosti 2 v slovníku bez diakritiky patria: *{prestat, potrestat}*, *{pravy, potravy}*, *{nitre, nie}*, *{nasla, nahla}*.

5.5 Slovník anglických slov

Slovník obsahuje 32 622 slov.

5.5.1 Anagramy

Najväčšiu množinu tvoria anagramy o veľkosti 2, je ich 1538. Ďalej je množina anagramov veľkosti 3, je ich 291, patrí sem napr. podmnožina *{adders, dreads, sadder}* aj *{later, alter, alert}*. Potom je množina anagramov veľkosti 4, je ich 51, ako príklad uvediem *{diet, tied, tide, edit}*, *{onset, notes, stone, tones}*, potom podmnožín veľkosti 5 je 12. Nakoniec je množina anagramov veľkosti 6, obsahuje 3 podmnožiny, napr. *{etam, tema, team, tame, meat, mate }*.



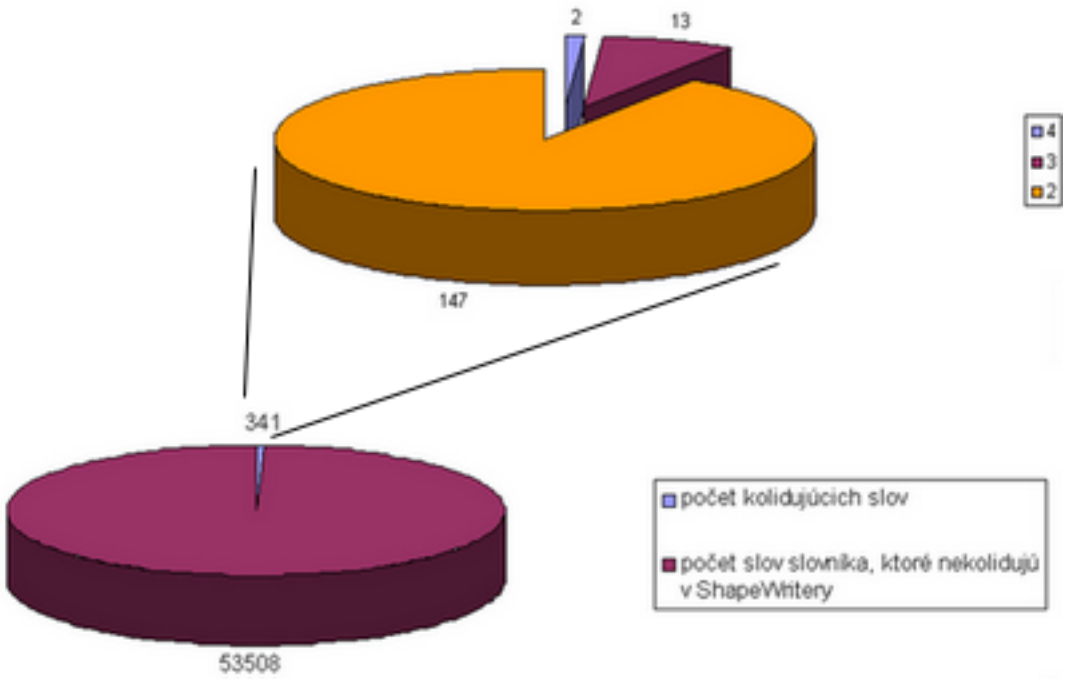
Obr. 5.8: Obrázok pomeru počtu všetkých kolízií v T9 k počtu nekolidujúcich slov slovníka slovenských slov bez diakritiky a pomer veľkostí množín kolízií v T9

5.5.2 Palindromy

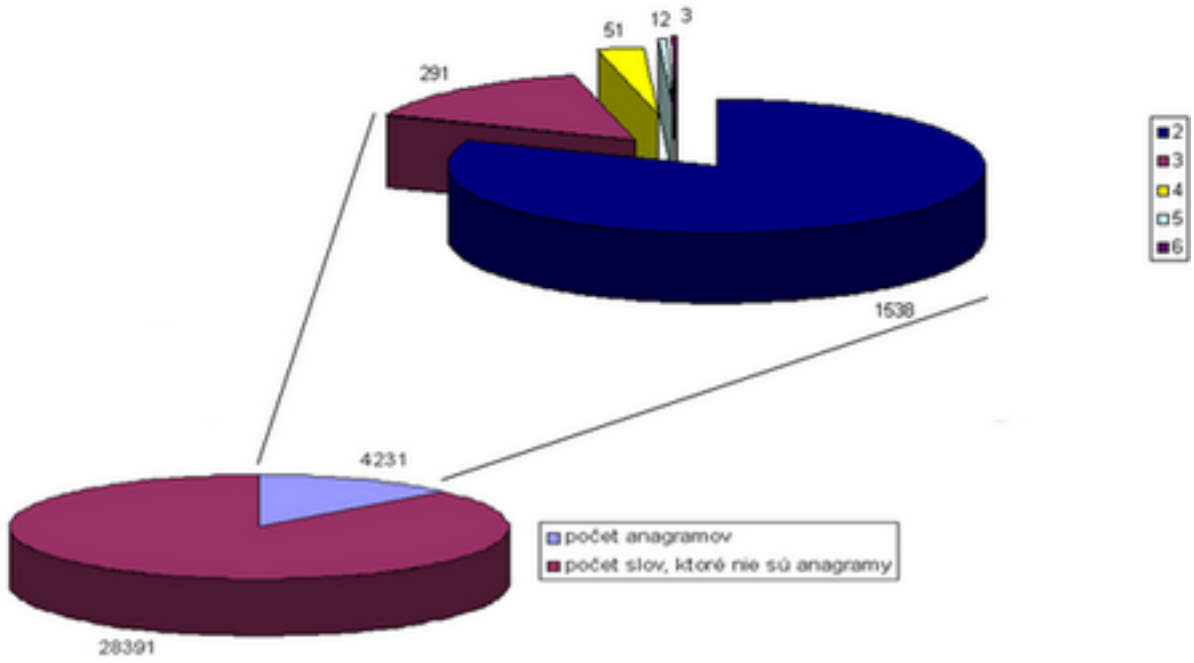
Počet palindromov v tomto slovníku je 67 napr. *madam, terret, pullup, hannah, huh, abba, arara.*

5.5.3 T9

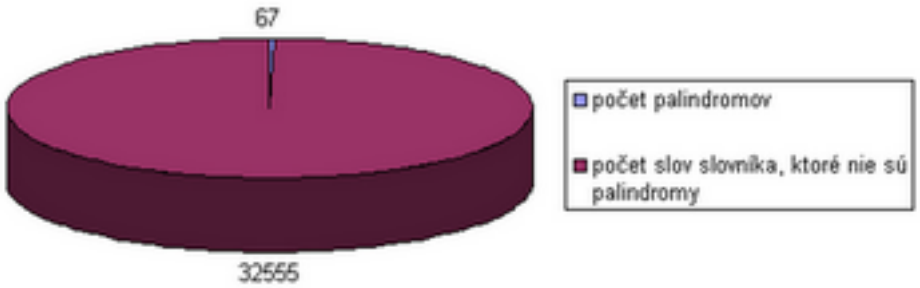
Najmohutnejšiu množinu, čo do počtu podmnožín, tvoria opäť podmnožiny veľkosti dva, je ich 1518. Potom je množina, ktorá obsahuje 323 podmnožín veľkosti tri napr. *{side, shed, ride}*. Počet podmnožín veľkosti 4 je 97 napr. *{sour, soup, pour, pots}*, veľkosti 5 je 48 *{sheer, sheep, rides, rider, sides}*. Medzi najmešie množiny patrí množina obsahujúca 14 podmnožín veľkosti 6 napr. *{paid, page, said, sage, raid, rage}*, množina obsahujúca 3 podmnožiny veľkosti 7 *{acres, bases, bards, cases, cares, cards, capes}* a množina, ktorú tvoria dve podmnožiny veľkosti 8 *{goods, gomer, inner, hoofs, hoods, homes, homer, immer}*. Miera efektívnosti T9 pre anglický jazyk je 1.0920228969303385.



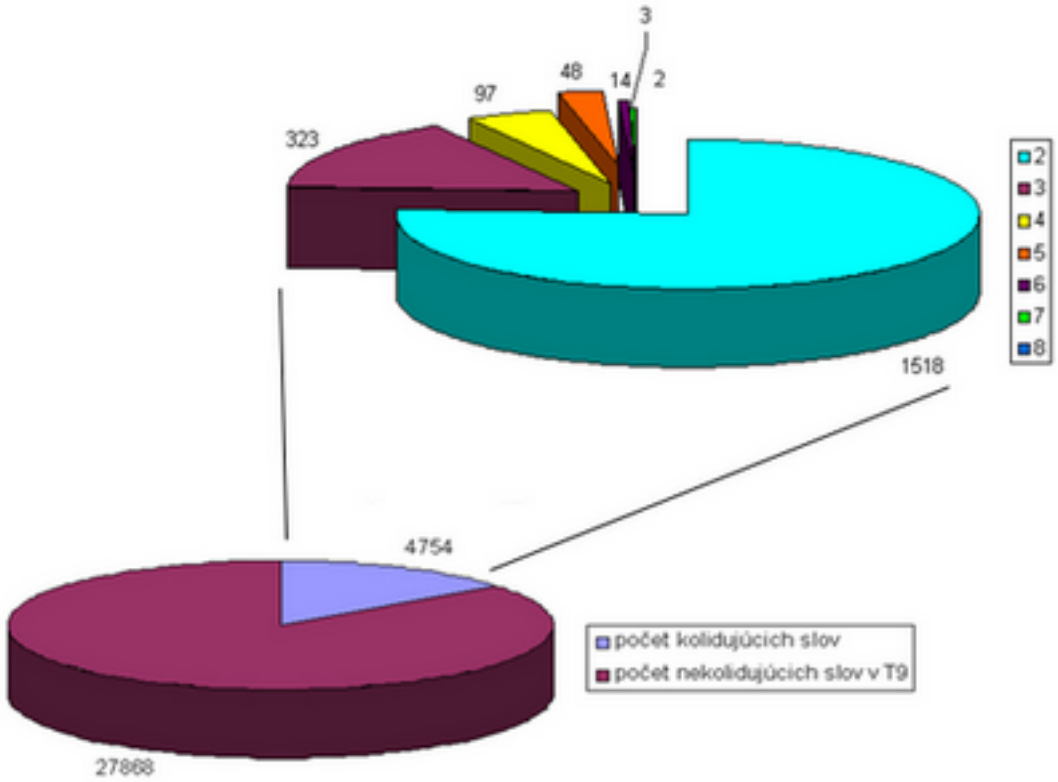
Obr. 5.9: Obrázok pomeru počtu všetkých kolízií v ShapeWritery k počtu nekolidujúcich slov slovníka slovenských slov bez diakritiky a pomer veľkostí množín kolízií v ShapeWritery



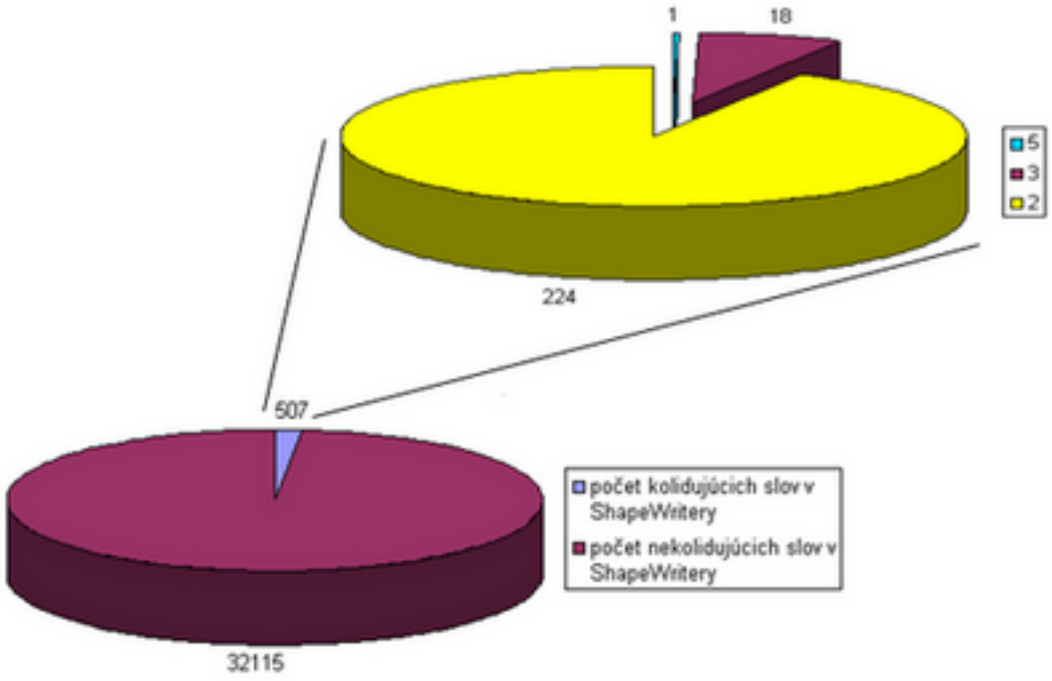
Obr. 5.10: Obrázok pomeru počtu všetkých anagramov k počtu všetkých slov slovníka anglických slov, ktoré nie sú anagramy a počtu veľkostí množín anagramov



Obr. 5.11: Obrázok pomeru počtu všetkých palindromov k počtu všetkých slov slovníka anglických slov, ktoré nie sú palindromy



Obr. 5.12: Obrázok pomeru počtu všetkých kolízií v T9 k počtu nekolidujúcich slov slovníka anglických slov a pomer veľkostí množín kolízií v T9



Obr. 5.13: Obrázok pomeru počtu všetkých kolízií v ShapeWritery k počtu nekolidujúcich slov slovníka anglických slov a pomer veľkostí množín kolízií v ShapeWritery

5.5.4 ShapeWriter

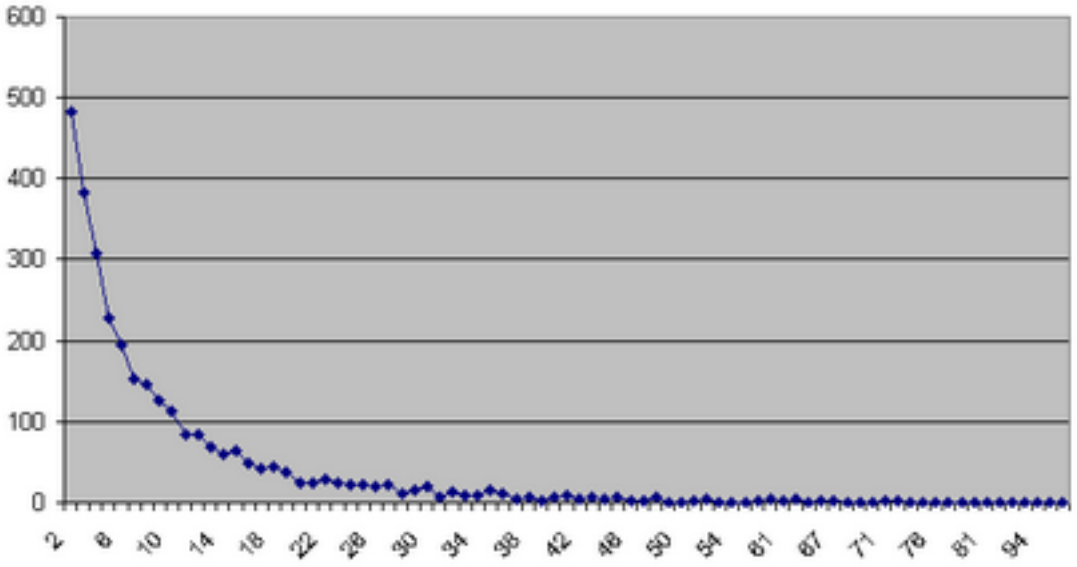
Je jedna podmnožina kolidujúcich slov veľkosti 5 : *{spred, sped, spied, spired, spued}*, potom sú podmnožiny veľkosti 3, tých je 18, napr. *{sue, sute, sure}*. Podmnožín veľkosti 2 je 224, napr. *{propose, purpose}, {trend, tend}*.

5.5.5 Soundex

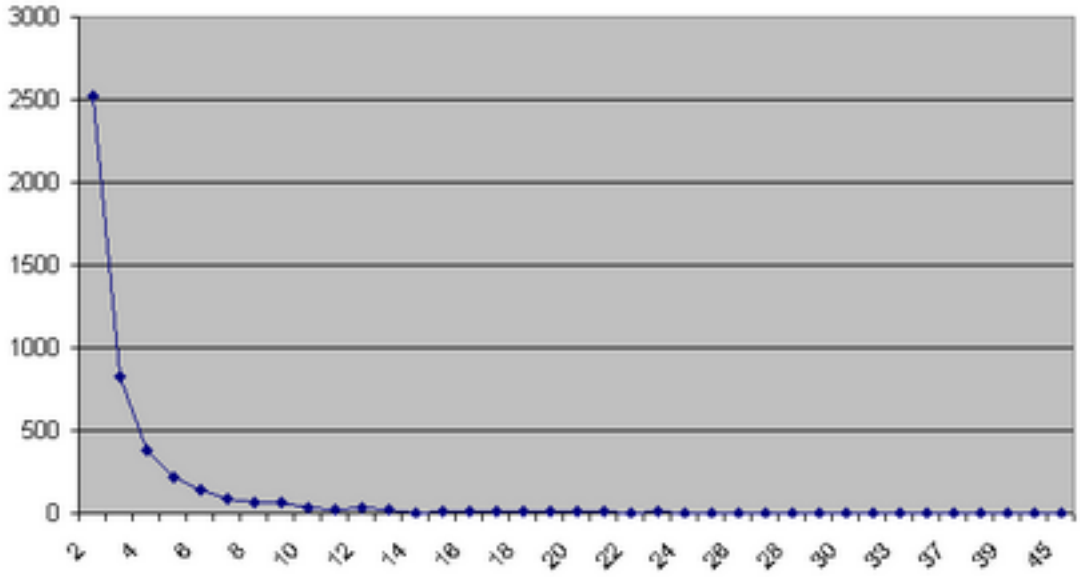
Zaujímavé výsledky sú pri fonetickej podobnosti algoritmom Soundex. Najväčšia veľkosť podmnožín je 114, takto veľká podmnožina je len jedna. Najmenšia veľkosť je 2. Takýchto podmnožín je 483.

5.5.6 Metaphone

Pri hľadaní foneticky podobných slov algoritmom Metaphone je najväčšia veľkosť podmnožín 45, takáto podmnožina je jedna. Najmenšia množina je množina s mohutnosťou 2516 s veľkosťou podmnožín 2.



Obr. 5.14: Obrázok veľkosti množín podobných slov slovníka anglických slov algoritmom Soundex



Obr. 5.15: Obrázok veľkosti množín podobných slov slovníka anglických slov algoritmom Metaphone

Záver

Cieľom tejto bakalárskej práce bolo vytvoriť aplikáciu, ktorá by vytvorila z vybraného textu slovník a analyzovala vlastnosti slov tohto slovníka. V programe si môžme zvoliť, či chceme načítať text, ktorý chceme analyzovať, alebo už hotový slovník. Nad vytvoreným slovníkom môžeme hľadať anagrami, palindromi, kolízie T9, kolízie ShapeWritera, foneticky podobné anglické slová algoritmom Metaphone a Soundex, podobné slová algoritmom Levenshtein a Diece, vyhľadávať slová, ktoré majú určité tagy (môžeme vyhľadávať slová ktoré majú nejaké tagy a zároveň nemajú niektoré tagy).

Druhým cieľom bolo zameranie sa na efektívnosť priradenia písmen k číslam pri technike písania textu do mobilných telefónov T9, lepšie rozloženie kláves na virtuálnej klávesnici a pokúsiť sa navrhnúť lepšie priradenie a rozloženie pre slovenský jazyk.
Na základe výsledkov, klasické priradenie písmen k číslam v T9 nie je zlé. Je len malý rozdiel medzi doteraz nájdeným lepším priradením. Takže nie je dobré, vyrábať telefóny s lepším rozložením, pretože užívateľom by sa horšie hľadali písmená na tlačidlách oproti klasickej T9, kde sú rozložené abecedne. Zistenia hľadania efektívnejšej virtuálnej klávesnice majú lepšie výsledky, takže by bolo dobré pokračovať v simuláciách a hľadať lepšie rozloženia.

Návrhy na budúce zlepšenie:
Do hľadania efektívneho priradenia písmen pre T9 by sme mohli zahrnúť aj frekvenciu výskytu daných slov, takže ak by bola množina s veľkou mohutnosťou a obsahovala by podmnožiny takisto s veľkou mohutnosťou, ale slová v daných podmnožinách by sa nepoužívali často, tak by nám to nevadilo, pretože pravdepodobnosť, že budeme takéto slová písať je veľmi malá. Ďalej by sa do priradenia mohlo zahrnúť aj abecedné usporiadanie, teda, aby písmená priradené k číslam 2 až 9 mali tendenciu ísť od a po z, takže by sa novým používateľom lepšie hľadali.
Na vytvorenie efektívneho rozloženia kláves by sme moholi použiť postup, ako pri vytváraní rozloženia ATOMIC, čiže okrem čo najmenšej energie (v

energii je zahrnutá pravdepodobnosť výskytu jednotlivých bigramov) by sa zahrnula aj abecedná postupnosť, pre lepšie hľadanie. Treťou podmienkou je, že časti slov, alebo celé slová, ktoré sa často používajú by boli spojené.

Literatúra

[1] **Smith, B. & Zhai S.**, *Optimised Virtual Keyboards with and without Alphabetical Ordering – A Novice User Study*, URL https://www.almaden.ibm.com/u/zhai/papers/Softkeyboard/Alpha10.pdf

[2] **Zhai S. & Kristensson, P.**, *Shorthand Writing on Stylus Keyboard*, URL http://www.almaden.ibm.com/u/zhai/papers/SharkFinal.pdf

[3] **Zhai S. & Kristensson, P.**, *Introduction to Shape Writing*, URL http://www.almaden.ibm.com/u/zhai/papers/IBM_Research_Report_ShapeWritingrj10393.pdf

[4] **Zhai S. & Hunter, M. & Barton, S.**, *The Metropolis Keyboard – An Exploration of Quantitative Techniques for Virtual Keyboard Design*, URL http://www.almaden.ibm.com/u/zhai/papers/Softkeyboard/UISTCamera.pdf

[5] **Zhai S. & Hunter, M. & Barton, S.** *Performace Optimization of Virual Keyboards* URL http://www.almaden.ibm.com/cs/people/zhai/papers/ZhaiHunterSmithHCIGalley.pdf

[6] URL http://aspell.net/metaphone/metaphone.basic

[7] **Snae, Ch.**, *A Comparison and Analysis of Name Matching Algorithms*, URL http://www.waset.org/pwaset/v19/v19-47.pdf

[8] **Antonio Francisco Pereira de Araujo**,*Monte Carlo methods and the Metropolis algorithm*, URL http://www.unb.br/ib/cel/chico/artigos/thesis/node6.html

[9] **Carstensen, A.**, *AN INTRODUCTION TO DOUBLE METAPHONE AND THE PRINCIPLES BEHIND SOUNDEX*, URL http://www.datamanagementgroup.com/Resources/Articles/Article_Introduction ToDoubleMetaphone.asp

[10] **Smet, A.**, *What is a Soundex code?*, URL http://www.highprogrammer.com/alan/numbers/soundex.html

[11] *The Levenshtein-Algorithm* , URL http://www.levenshtein.net

[12] *Dice's coefficient*, URL http://en.wikipedia.org/wiki/Dice_coefficient

[13] http://en.wikipedia.org/wiki/Levenshtein_distance

[14] **Lawrence Philips**, *Hanging on the Metaphone*, Computer Language Vol. 7 No. 12, [December 1990]

[15] **QWERTY**, http://en.wikipedia.org/wiki/Qwerty

[16] **Ivers, K.** *A teacher's guide to ussing technology in the classroom*

[17] **Lait, A.J. & Randell, B.**, *An Assessment of Name Matching Algorithms*, 1996

[18] *Fitts's law*, URL http://en.wikipedia.org/wiki/Fitts%27_law