Univerzita Karlova v Praze
Pedagogická fakulta

Katedra matematiky a didaktiky matematiky

Rozpoznávání jazyka na základě frekvenční analýzy

Autor: Jan Čadek

Vedoucí práce: RNDr. Antonín Jančařík, Ph.D.

Praha 2010

NÁZEV:

Rozpoznávání jazyka na základě frekvenční analýzy

ABSTRAKT:

Cílem této bakalářské práce je ověřit, zda lze jazyk textu (případně kryptogramu) rozpoznat pomocí základní kryptoanalytické pomůcky – frekvenční analýzy. První kapitola je věnována letmému pohledu do dějin kryptologie. Další část se zabývá frekvenční analýzou, její princip je zde vysvětlen na jednoduchém příkladu. Ve třetí části se již nachází popis tří vytvořených programů na rozpoznávání jazyka. Poslední kapitola popisuje výsledky, které by bez vzniku programů nemohly být zkoumány – jde především o porovnání statistické příbuznosti evropských jazyků s příbuzností reálnou. Zajímavým zjištěním je, že se skutečné vztahy evropských jazyků promítají do jejich frekvenčních analýz.

KLÍČOVÁ SLOVA:

frekvenční analýza, kryptologie, evropské jazyky, rozpoznávání, příbuznost jazyků

TITLE:

Language recognition based on the frequency analysis

SUMMARY:

The aim of this bachelor thesis is to use the basic cryptoanalytic aid – frequency analysis – for the recognition of the language of a text (or of a cryptogram). The first chapter is devoted to the history of cryptology. Chapter 2 focuses on frequency analysis, its principle is explained on a simple example. In the third part, there is the description of three programs for language recognition. Last chapter compares the results, which I gained through my programs, with the affinity of European languages as described in linguistics. One of the conclusions of my work is that the real affinity of languages is well approximated using the program analysis.

KEY WORDS:

frequency analysis, cryptology, European languages, recognition, relationships of languages

PROHLÁŠENÍ:

Prohlašuji, že jsem bakalářskou práci *Rozpoznávání jazyka na základě frekvenční analýzy* vypracoval pod vedením RNDr. Antonína Jančaříka, Ph.D. samostatně na základě vlastních zjištění a za použití pramenů uvedených v seznamu.

Praha, 23. června 2010

.....

Jan Čadek

PODĚKOVÁNÍ:

Tímto bych rád poděkoval vedoucímu své práce RNDr. Antonínu Jančaříkovi, Ph.D. za cenné připomínky a komentáře, za vypůjčení odborné literatury a korekturu textu.

Obsah

Úvod

O frekvenční analýze, její funkci a použití, jsem se poprvé dozvěděl v zimním semestru roku 2009 v semináři Teorie čísel a kryptografie, který vedl RNDr. Antonín Jančařík, Ph.D. Tento předmět plný zajímavostí z oblasti šifrování a dějin matematiky mě přivedl k tématu mé bakalářské práce.

Zadání mě zaujalo především novým využitím frekvenční analýzy (pro odhad jazyka textu), která se kdysi využívala hlavně pro dešifrování kryptogramů. Dále bylo téma uvedeno na internetových stránkách katedry matematiky a didaktiky matematiky jako vhodné pro studenty s dvojoborem Matematika – Informační a komunikační technologie, což jsem splňoval. Líbila se mi možnost využití znalostí informatiky v práci psané pod vedením KMDM.

Cílem této práce je ověřit, zda lze jazyk vstupního textu rozpoznávat pomocí frekvenční analýzy. Proto bylo třeba vyvinout program a přijít na vhodný porovnávací algoritmus, který by to dokázal. Získaná data z naprogramovaných aplikací pak byla využita ke zjištění, nakolik se reálný vývoj a příbuznost jazyků promítá do příbuznosti statistické; srovnání bylo provedeno pro různé porovnávací algoritmy a výsledky byly graficky zpracovány.

Otázkou tedy je, jestli jsou statistické vlastnosti jazyků založené na četnostech výskytu jednotlivých znaků dostatečné pro určení jazyka textu. Dále má práce přinést poznatek o vlivu jednoduché substituční šifry na samotnou frekvenční analýzu.

1 Historické pozadí vzniku frekvenční analýzy

Abych zde mohl popsat význam slovního spojení „frekvenční analýza“ a jeho vznik, musím se k tomuto termínu propracovat postupně. V následujících odstavcích objasňuji základní pojmy, které se v mé práci vyskytují, dále se zde věnuji alespoň základnímu nástinu dějin kryptologie, neboť je tento vědní obor úzce spjat s tématem mé práce.

Podle (Grošek, Porubský 1992, s. 10) je KRYPTOLOGIE věda, jež se zabývá šifrováním a dešifrováním. Název je odvozen z řeckého slova κρυπτόσ (čteno kryptós) – tj. zatajený, skrytý.

Kryptologie se dělí na dvě části:

- kryptografii – ta se věnuje návrhu šifrovacích metod
- kryptoanalýzu – ta se zaměřuje na metody luštění šifer.

Další užitečné termíny týkající se této problematiky shrnuje (Piper, Murphy 2006, s. 4-15) takto: *„Informaci, kterou je nutno nějakým způsobem zabezpečit, většinou označujeme jako **otevřený text**, proces zabezpečování zprávy pak jako **šifrování**. Zabezpečený otevřený text se stává **šifrovým textem** neboli **kryptogramem** a sada pravidel použitých pro šifrování otevřeného textu se nazývá **šifrovacím algoritmem**. Operace tohoto algoritmu se běžně odvíjejí od **šifrovacího klíče**, který společně s textem zprávy představuje vstupní informace pro algoritmus. Chce-li příjemce z kryptogramu obdržet původní zprávu, musí použít **dešifrovací algoritmus**, který ve spojení s **dešifrovacím klíčem** převede zašifrovaný text na původní otevřený text.“*

(Singh 2003) píše, že *„kryptografii můžeme rozdělit na dvě větve – **transpozici** a **substituci**“*. Transpoziční šifra je dle něj v podstatě přesmyčka, tedy písmena si zachovávají svou identitu, ale mění pozici. U substituční šifry se každý znak otevřeného textu nahradí znakem jiným, tedy písmena si zachovávají pozici, ale mění identitu.

Jedna z prvních zmínek o substituční šifře se objevuje v Kámasútře (Vátsjájana, 4. století n. l.), kde je doporučeno ženám ovládat tajná písma k ukrývání svých vztahů. V podstatě šlo o náhodné spárování písmen abecedy a následně jejich záměnu. Další popis využití substituce, tentokrát pro vojenské účely, je zdokumentován v díle Zápisky o válce galské od Julia Caesara. (Singh 2003, s. 23-24).

Caesarova šifra tedy spočívá v tom, že *„Každé písmeno zprávy bylo zaměněno za písmeno, které leželo o tři místa dále v abecedě. Z konce abecedy se přejde opět (cyklicky)*

na začátek abecedy.“ (Vondruška 2006, s.45). Tato kryptografická metoda spadá do kategorie jednoduchých substitučních šifer[1].

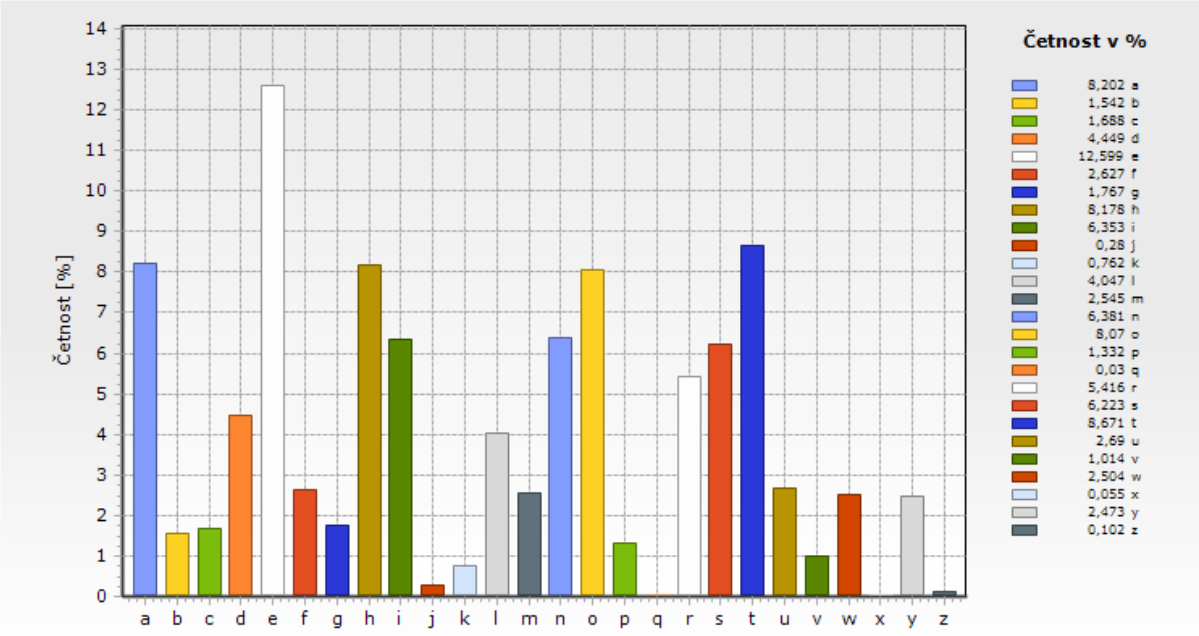
„Obdoba Cézarovskej myšlienky sa však už objavila na inom mieste predtým – bol to hebrejský ***atbaš****. V tejto šifrovacej metóde je prvé písmeno abecedy nahradené posledným, druhé predposledným atď. a naopak. Názov atbaš je odvodený od toho, že prvé písmeno hebrejskej abecedy* ***alef*** *je nahradené posledným písmenom* ***tav****, druhé* ***bet*** *je nahradené predposledným* ***sin****. Prejavy tohto šifrovania (aj keď doteraz nevyjasnené prečo) nájdem v Starom testamente.“* (Grošek, Porubský 1992, s. 15).

S rostoucím počtem používání šifer (převážně ve vojenství) se začala rozvíjet již zmíněná kryptoanalýza, tedy způsob, jak k kryptogramu získat původní text bez znalosti klíče. Za obrovský přínos se považuje práce Arabů, díky nimž vznikla právě frekvenční analýza. Aby k tomu mohlo dojít, bylo potřeba, aby se společnost dostala na dostatečně vysokou úroveň v oblasti matematiky, statistiky a lingvistiky. V teologických školách se velmi podrobně zkoumal Korán – na základě četností slov v textu určovali chronologii jeho částí. *„Je důležité, že teologové se přitom nezastavili na úrovni jednotlivých slov. Zkoumali i jednotlivá písmena a povšimli si jejich rozdílné relativní četnosti.“* (Singh 2003, s. 30). Z té doby (9. století n. l.) se dochoval popis této techniky od arabského učence Abú Jusúf Jaqúb ibn Isháq ibn as-Sabbáh ibn’omrán ibn Ismail al-Kindího v knize Rukopis o dešifrování kryptografických zpráv. (Singh 2003).

[1] *„V této šifře se každý znak otevřeného textu nahradí jedním znakem šifrované abecedy. Pro celý otevřený text se použije stejná šifrová abeceda.“* (Vondruška 2006, s. 31)

2 Frekvenční analýza

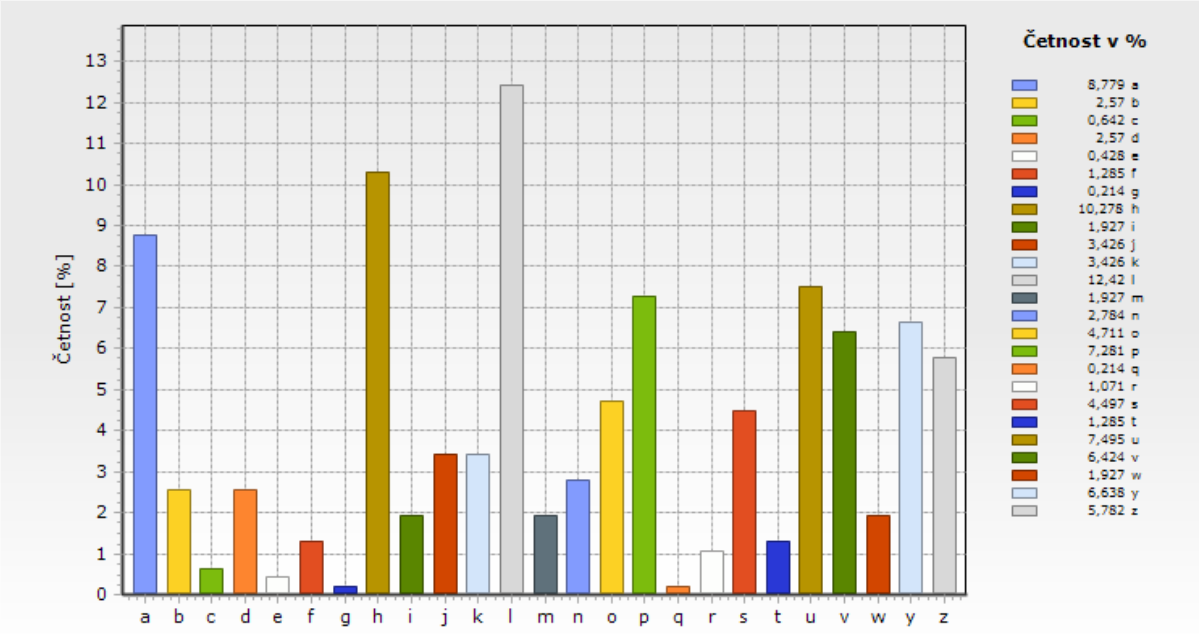
Frekvenční analýza (Vondruška 2006, s. 31/32) je studie o četnosti výskytu písmen nebo skupin písmen v šifrovaném textu. Tato metoda se používá při prolamování jednoduchých substitučních šifer. U každého jazyka světa je tato analýza specifická, vždy viditelně převládá jeden znak svou relativní četností nad ostatními, a právě díky tomu se dá v jednoduchém a dostatečně dlouhém kryptogramu (zašifrovaná zpráva) odhalit tento znak, který v případě substituční šifry odhalí i klíč (a ten pak určuje posun celé abecedy). Pro jednodušší pochopení vše ukážu na příkladu.



Obr. 1 – Analýza anglického jazyka

Na Obr. 1 je vidět, jak vypadá relativní četnost znaků v anglickém textu. Byla provedena na velmi dlouhém díle – mám na mysli Bibli (z www.bibledatabase.org), proto je velmi přesná. Nyní vezmu nějaký kryptogram, o kterém vím, že je v angličtině, např.:

```
olylbwvuslnyhukhyvzldpaohnyhclhukzahalsfhpyhukiyvbnoatlaolil
laslmyvthnshzzjhzlpudopjopadhzlujsvzlkpadhzhilhbapmbszjhyhih
lbzhukhaaohaaptlburuvduavuhabyhspzazvmjvbyzlhnylhawypglpuhzj
pluapmpjwvpuavmcpldaolyldlyladvyvbukishjrzwvazulhyvulleayltp
afvmaolihjrhukhsvunvululhyaolvaolyaolzjhslzdlyllejllkpunsfoh
ykhuknsvzzfdpaohssaolhwwlhyhujlvmibyupzolknvskaoldlpnoavmaol
puzljadhzclyfylthyrhislhukahrpunhssaopunzpuavjvuzpklyhapvupj
vbskohyksfishtlqbwpalymvyopzvwpupvuylzwljapunpa
```



Obr. 2 – Analýza kryptogramu

Na Obr. 2 je zobrazena analýza této šifry a je patrné, že písmeno L v kryptogramu bude pravděpodobně odpovídat písmenu E v dešifrovaném textu. Můžeme ho tedy nahradit a takto postupovat při lámání šifry s každým znakem, pokud bude výsledný text dávat smysl. V tomto případě jsem použil pouze jednoduchou substituční šifru (tedy abeceda kryptogramu je celá posunutá o konstantní hodnotu), takže původní text lze získat velmi snadno, protože odhalením jednoho znaku dostanu klíč, který odpovídá posunu celé abecedy. Písmena E a L jsou od sebe vzdálena v abecedě o 7 kroků, použil jsem tedy posun celé abecedy o 7. Snadným posunutím všech znaků o 7 kroků zpět získám původní zprávu (Poe 1991, s. 85):

```
hereuponlegrandarosewithagraveandstatelyairandbroughtmethebe
etlefromaglasscaseinwhichitwasencloseditwasabeautifulscaraba
eusandatthattimeunknowntonaturalistsofcourseagreatprizeinasc
ientificpointofviewthereweretworoundblackspotsnearoneextremi
tyofthebackandalongoneneartheotherthescaleswereexceedinglyha
rdandglossywithalltheappearanceofburnishedgoldtheweightofthe
insectwasveryremarkableandtakingallthingsintoconsiderationic
ouldhardlyblamejupiterforhisopinionrespectingit
```

Jak jsem již ale uvedl, tento postup lze využít pouze u jednoduchých šifer. Na základě této metody lze právě rozeznávat jazyk vstupního textu.

3 Program rozpoznávající jazyk textu

Nyní bych rád přešel k hlavní části mé bakalářské práce. Tato kapitola má za úkol přiblížit pozadí vývoje programu Frekvenční analýza 1.0, který byl kvůli mým slabším znalostem a schopnostem programovat velmi táhlý, z počátku obtížný, ale nakonec úspěšný.

3.1 Výběr vývojového prostředí

Na počátku veškerého snažení jsem byl postaven před důležité rozhodnutí – v čem program tvořit. Na střední škole jsem se s programováním vůbec nesetkal (pominu-li HTML – jazyk pro tvorbu webových stránek), základní informace jsem obdržel až zde, na Pedagogické fakultě na katedře informačních technologií a technické výchovy. Pracoval jsem v prostředí Turbo Pascal a později v Delphi 2005, které vychází právě z tohoto dnes již historického a dávno překonaného programovacího jazyka.

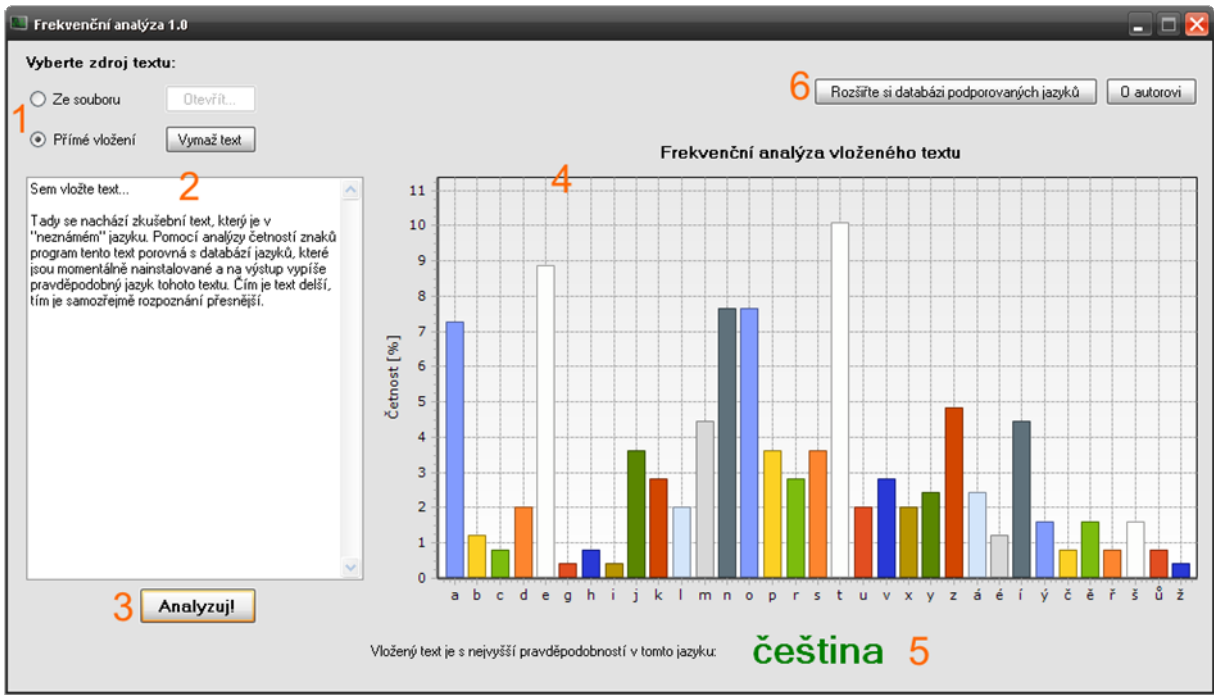
Logicky jsem se tedy chopil alespoň trochu známého vývojového prostředí Delphi. Při přemýšlení nad programováním aplikace jsem zjistil, že k mému specifickému úkolu budu potřebovat podporu kódování Unicode[2], abych dokázal generovat analýzy jednotlivých jazyků bez omezení jejich přirozených abeced; to ale bohužel verze Delphi 2005 neumí. Na oficiálním webu výrobce (http://www.embarcadero.com/) jsem se dozvěděl, že Unicode ovládá až prostředí Delphi 2009 a novější. Ke stažení sice byla k dispozici 30 denní zkušební verze, ale seznámit se s vylepšeným prostředím, zavzpomínat na znalosti Pascalu a vyvinout program během tak krátké doby bylo téměř nemyslitelné.

V říjnu 2009 jsem díky předmětu Vývoj a administrace databázových aplikací objevil krásu a jednoduchost Microsoft Visual Studio 2008 ve spojení s programovacím jazykem C#. Ačkoli byl tento jazyk i prostředí pro mne absolutně nové, po letmém prozkoumání jsem zjistil, že je to vhodný kandidát právě pro mou práci. Nejenže od základu stojí na Unicode, obsahuje také syntaktickou nápovědu IntelliSense. Další nespornou výhodou Visual Studia 2008 byl fakt, že jsem získal zcela legálně plnou verzi díky programu MSDN AA[3], který funguje na naší škole.

[2] Znaková sada Unicode – tabulka obsahující všechny abecedy světa (Eller 2002, s. 92)
[3] Licenční program firmy Microsoft, který umožňuje studentům získávat zdarma její vybraný software

3.2 Popis funkce programu

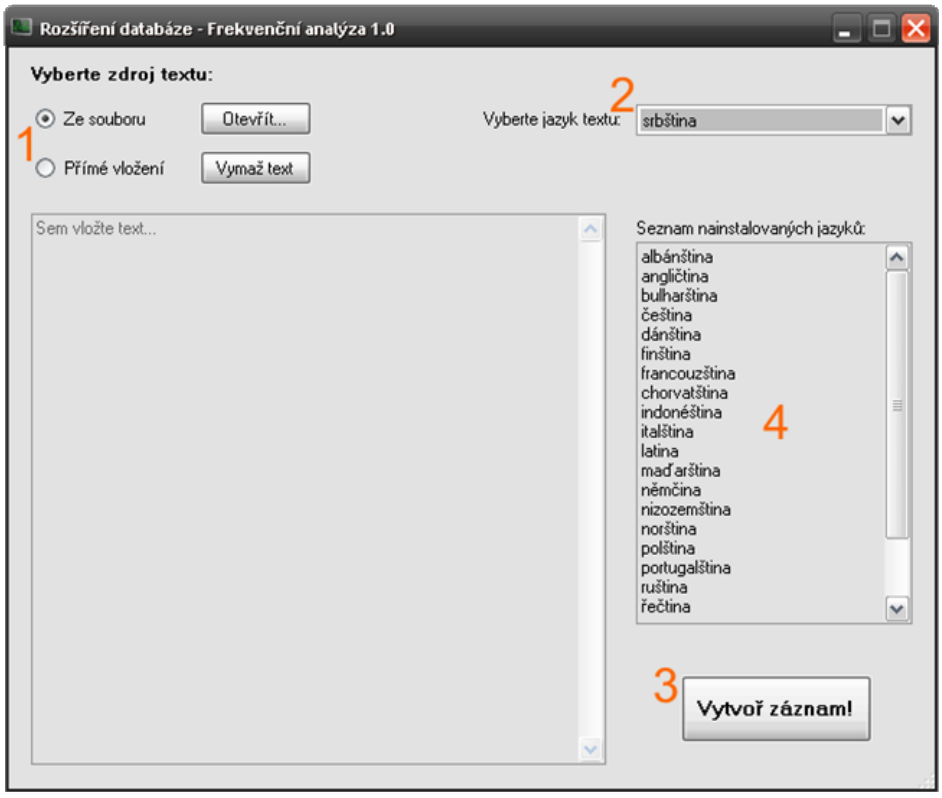
Již samotný název této práce napovídá, co je cílem programu Frekvenční analýza 1.0. Vloží se do něj text v neznámém jazyku, provede se jeho analýza, ta s porovná s analýzami známých jazyků, které jsou momentálně nainstalovány. Výstupem z programu je nejpravděpodobnější jazyk vloženého textu.



Obr. 3 – Hlavní formulář

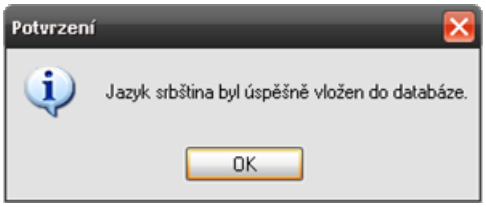
Uživatelské rozhraní programu Frekvenční analýza 1.0 (Obr. 3) bylo vytvořeno s důrazem na jednoduchost a přehlednost, ovládání je velmi intuitivní. Pomocí radiobuttonů[4] (1) uživatel vybere zdroj vstupního textu – buď načte řetězec z textového souboru (ten musí být v ovšem v Unicode) nebo ho vloží přímo do připraveného textboxu[5] (2). Pak stačí jen spustit porovnávání pomocí tlačítka (3). Číslo (4) označuje komponentu grafu, do kterého se zakreslí statistické údaje vstupního řetězce. Ve finále se na formuláři v místě (5) objeví nejpravděpodobnější jazyk textu ze vstupu. Tlačítko (6) zpřístupní vedlejší formulář, který umožňuje uživateli vkládat do programu nové jazyky.

[4] Přepínače – skupina ovládacích prvků, aktivní může být vždy jen jeden ze skupiny (Petzold 2003, s. 569)
[5] *„Základní prvek pro textový vstup. Omezení pro délku jeho obsahu je 64KB.“* (Pirkl 2005, s. 80)



Obr. 4 – Vedlejší formulář

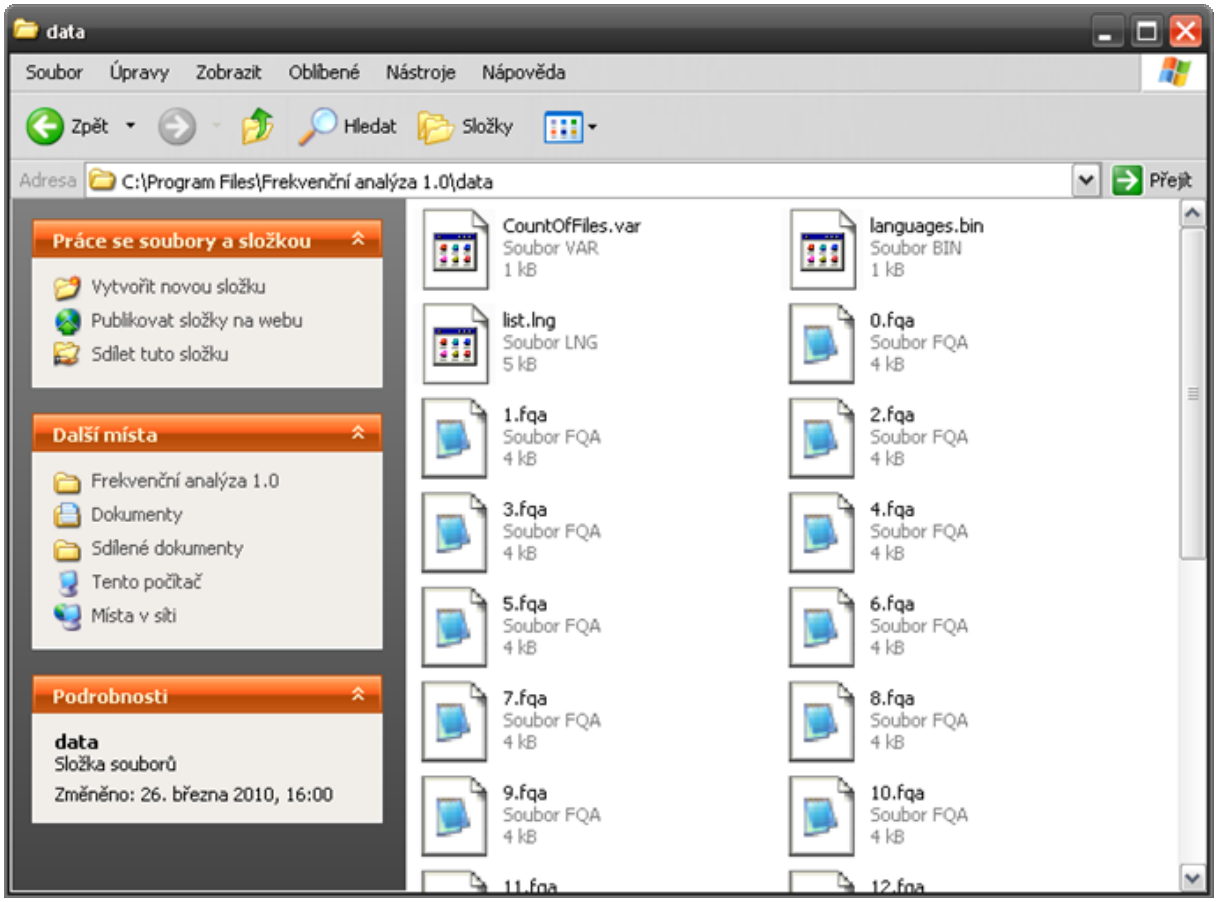
Na Obr. 4 je zobrazen vedlejší formulář sloužící k rozšíření databáze rozpoznatelných jazyků. Radiobuttony (1) opět slouží k určení vstupu, tentokrát ale pro vložení textu ve známém jazyku. Combobox[6] (2) obsahuje výběr jazyků, které lze ještě nainstalovat. Fixní výběr byl zvolen jako nejjednodušší ochrana vstupu před vložením jednoho jazyka vícekrát (což samozřejmě jde, ale pod jiným „názvem“). Pomocí tlačítka (3) se provede analýza vstupního řetězce, která se uloží do speciálního souboru. Po úspěšném vložení nového jazyka se objeví potvrzovací okno – Obr. 5, a tento jazyk se objeví v seznamu nainstalovaných jazyků (4).



Obr. 5 – Potvrzovací okno

Při pohledu na strukturu ukládání analýz a podpůrných souborů aplikace se dá celkem snadno odhadnout, jak funguje princip porovnávání jazyků, jde v podstatě o automatizovanou práci se soubory.

[6] *„Seznam hodnot, v jeden okamžik může být viditelná jedna hodnota.“* (Pirkl 2005, s. 61)



Obr. 6 – Složka ..\data

Na Obr. 6 je vyobrazena podložka programu \data\, která obsahuje 4 druhy souborů:

- *.fqa – textové soubory obsahující analýzy nainstalovaných jazyků
- languages.bin – soubor obsahující názvy jazyků, jejich indexy odpovídají číslům souborů s analýzami
- list.lng – soubor obsahující nabídku jazyků, která se načítá do comboboxu
- CountOfFiles.var – textový soubor obsahující jediné číslo, které odpovídá počtu souborů *.fqa

3.3 Princip porovnávání

Princip porovnávání frekvenční analýzy vstupu s databází jazyků je velmi jednoduchý. Pro představu ho uvádím pouze s anglickou abecedou (základních 26 znaků „a“ – „z“) a porovnávat budu se dvěma jazyky. Průchodem vstupního textu v neznámém jazyce se vygeneruje vstupní analýza, ta může vypadat například takto:

Písmeno	Relativní četnosti v [%]	Písmeno	Relativní četnosti v [%]
a	7,040229885	n	7,655993432
b	1,949917898	o	7,655993432
c	1,498357964	p	1,949917898
d	4,08456486	q	0,184729064
e	12,78735632	r	7,019704433
f	1,908866995	s	5,747126437
g	2,50410509	t	9,092775041
h	5,952380952	u	2,50410509
i	6,588669951	v	1,395730706
j	0,041050903	w	1,621510673
k	0,65681445	x	0,061576355
l	4,659277504	y	2,093596059
m	3,345648604	z	0

Nyní se tyto hodnoty porovnají kupř. s angličtinou a latinou, jejichž analýzy vypadají následovně:

Písmeno	Relativní četnosti v [%]		Písmeno	Relativní četnosti v [%]	
	Angličtina	**Latina**		**Angličtina**	**Latina**
a	8,202007797	8,050259404	n	6,38103263	6,342649743
b	1,542463331	1,578726884	o	8,069829447	5,336506726
c	1,687617222	3,512386757	p	1,332272336	2,27904341
d	4,448952408	3,462339254	q	0,030127482	1,410147654
e	12,59886676	12,53181118	r	5,41570984	5,795194185
f	2,627371503	0,961671838	s	6,22258439	7,571601736
g	1,767478961	1,031550141	t	8,671008212	8,867225647
h	8,178288383	0,99136576	u	2,690176942	8,189353823
i	6,35304117	11,70665471	v	1,013973094	1,482953528
j	0,280201526	0	w	2,503577838	0
k	0,762305004	0,000766744	x	0,054516397	0,452274687
l	4,046646907	2,686358988	y	2,47325907	0,084272189
m	2,544608599	5,61187255	z	0,10208275	0,063012455

V další tabulce je znázorněno, jak probíhá výpočet. Sloupec V obsahuje analýzu vstupního textu, sloupec A analýzu anglického jazyka a sloupec L analýzu latiny. V posledních dvou sloupcích jsou vypočítány rozdíly frekvencí odpovídajících si písmen, z nich je vytvořena absolutní hodnota.

Písmeno	Relativní četnosti v [%]			Abs. hodnoty rozdílů	
	V	A	L	| V – A |	| V – L |
a	7,040229885	8,202007797	8,050259404	1,161777912	1,010029519
b	1,949917898	1,542463331	1,578726884	0,407454567	0,371191015
c	1,498357964	1,687617222	3,512386757	0,189259258	2,014028793
d	4,08456486	4,448952408	3,462339254	0,364387548	0,622225607
e	12,78735632	12,59886676	12,53181118	0,188489562	0,255545139
f	1,908866995	2,627371503	0,961671838	0,718504508	0,947195158
g	2,50410509	1,767478961	1,031550141	0,736626129	1,472554949
h	5,952380952	8,178288383	0,99136576	2,22590743	4,961015192
i	6,588669951	6,35304117	11,70665471	0,235628781	5,117984764
j	0,041050903	0,280201526	0	0,239150623	0,041050903
k	0,65681445	0,762305004	0,000766744	0,105490554	0,656047705
l	4,659277504	4,046646907	2,686358988	0,612630597	1,972918516
m	3,345648604	2,544608599	5,61187255	0,801040005	2,266223946
n	7,655993432	6,38103263	6,342649743	1,274960802	1,313343688
o	7,655993432	8,069829447	5,336506726	0,413836015	2,319486706
p	1,949917898	1,332272336	2,27904341	0,617645562	0,329125511
q	0,184729064	0,030127482	1,410147654	0,154601582	1,22541859
r	7,019704433	5,41570984	5,795194185	1,603994593	1,224510248
s	5,747126437	6,22258439	7,571601736	0,475457953	1,824475299
t	9,092775041	8,671008212	8,867225647	0,421766829	0,225549394
u	2,50410509	2,690176942	8,189353823	0,186071852	5,685248732
v	1,395730706	1,013973094	1,482953528	0,381757612	0,087222822
w	1,621510673	2,503577838	0	0,882067165	1,621510673
x	0,061576355	0,054516397	0,452274687	0,007059958	0,390698332
y	2,093596059	2,47325907	0,084272189	0,379663011	2,00932387
z	0	0,10208275	0,063012455	0,10208275	0,063012455
			Součet:	14,88731316	40,02693753

Tabulka také v posledním řádku obsahuje dvě buňky nesoucí označení „Součet“, jde o součet hodnot rozdílů. Čím větší je tento součet u nějakého jazyka, tím menší je pravděpodobnost, že tento jazyk bude zvolen za vhodného kandidáta při rozpoznávání. Tímto způsobem se vytvoří součty absolutních hodnot rozdílů pro každý jazyk v databázi a logicky nejpravděpodobnější jazyk vstupu bude ten, který má tuto hodnotu nejnižší. V tomto konkrétním případě je tedy vstupní text spíše psán v angličtině (suma rozdílů je 14,89 %) než v latině (zde se rozdílnost analýz blíží k 40,03 %).

3.4 Rozbor zdrojového kódu

Na počátku zrodu této aplikace stál jednoduchý algoritmus, který počítal výskyty jednotlivých znaků v řetězci (Allen 2010a):

```
using System;
using System.IO;

class Program
{
      static void Main()
      {
            // Array to store frequencies.
            int[] c = new int[(int)char.MaxValue];
            // Read entire text file.
            string s = File.ReadAllText("text.txt");
            // Iterate over each character.
            foreach (char t in s)
            {
                  // Increment table.
                  c[(int)t]++;
            }
            // Write all letters found.
            for (int i = 0; i < (int)char.MaxValue; i++)
            {
                  if (c[i] > 0 &&
                  char.IsLetterOrDigit((char)i))
            {
            Console.WriteLine("Letter: {0}  Frequency: {1}", (char)i,
      c[i]);
      }
}
```

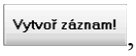
Je snadno pochopitelný a spolehlivý, proto jsem se jej rozhodl využít. Je sice paměťově náročnější kvůli zbytečné alokaci nevyužitého místa (většina prvků v poli bude nulová), ale v dnešní době výkonných počítačů to není až tak znatelné. Jsem si vědom toho, že bych si měl správně napsat nějakou svou třídu, která by se chovala přesně tak, jak bych potřeboval, ale zde jsem právě narazil na své chatrné znalosti a i přes snahu optimalizace algoritmu jsem se nakonec vrátil k tomuto původnímu. Kód jsem částečně modifikoval, aby na výstup dával místo absolutních četností znaků relativní, aby velká písmena převedl na malá

a zároveň ignoroval čísla ve vstupním řetězci. Vytvořil jsem z něj proceduru, kterou v případě potřeby volám:

```
1.    public static double[] AnalyzaTextu(string text)
2.          {
3.                double[] AnalyzaPole = new double[(int)char.MaxValue];
4.                int PocetPismen = 0;
5.                foreach (char t in text)
6.                {
7.                      AnalyzaPole[(int)t]++;
8.                      if (char.IsLetter(t)) { PocetPismen++; }
9.                }
10.               for (int i = 0; i < AnalyzaPole.Length; i++)
11.               {
12.                     AnalyzaPole[i]=(AnalyzaPole[i] / PocetPismen)*100;
13.               }
14.               return AnalyzaPole;
15.         }
```

Pole `AnalyzaPole` (3. řádek) je typu double – to znamená, že je s to pojmout reálná čísla odpovídající procentuálnímu zastoupení jednotlivých znaků. Právě u něj je nežádoucí jeho zbytečná velikost, která je `(int)char.MaxValue`, tedy maximální hodnota, kterou může objekt typu char (znak) nabývat – 65 535. Indexy položek v poli odpovídají ordinální hodnotě znaku na vstupu, položky samotné pak obsahují absolutní četnosti v textu, které jsou poté for cyklem (10. řádek) nahrazeny výskyty relativními.

Zdrojový kód obsahuje ještě důležitou třídu `Serializace` a v ní metodu `SerializujArrayList`, `Deserializace` (a metoda `DeserializujArrayList`) a také metodu `IndexMinimalniHodnoty` třídy `Minimum`. Tyto kódy je možné naleznout v Příloze I. (s. A-I), kde jsou i okomentovány.

Abych měl s čím porovnávat neznámý text, musím nejprve nainstalovat nějaké jazyky. Popíšu zde tedy nejprve generování souborů *.fqa v rámci vedlejšího formuláře. Nejdůležitější událost se vykoná při kliknutí na tlačítko 

, na pozadí to vypadá takto:

```
1.    private void btnVytvorZaznam_Click(object sender, EventArgs e)
2.          {
3.                string text = "";
4.                if (radioBtnZeSouboru.Checked)
5.                {
```

```
6.                      text = File.ReadAllText(soubor).ToLower();
7.                  }
8.                  else
9.                  {
10.                     text = textBoxVlozenyText.Text.ToLower();
11.                 }
12.                 if (comboBoxVyberJazyku.Text == "")
13.                     MessageBox.Show("Nevybrali jste jazyk textu!!!",
"Varování", MessageBoxButtons.OK, MessageBoxIcon.Warning);
14.                 else
15.                 {
16.                     StreamReader CountOfFiles = new
StreamReader(Application.StartupPath + @"\data\CountOfFiles.var");
17.                     int PocetSouboru =
Convert.ToInt16(CountOfFiles.ReadLine());
18.                     CountOfFiles.Close();
19.                     double[] NovyJazyk = Analyza.AnalyzaTextu(text);
20.                     string NazevSouboru = PocetSouboru + ".fqa";
21.                     StreamWriter VytvorJazyk = new
StreamWriter(Application.StartupPath + @"\data\" + NazevSouboru);
22.                     for (int i = 0; i < NovyJazyk.Length; i++)
23.                     {
24.                         if (char.IsLetter((char)i) &&
char.IsLower((char)i))
25.                         {
26.                             VytvorJazyk.WriteLine("{0}", NovyJazyk[i]);
27.                         }
28.                     }
29.                     StreamWriter ZmenPocetSouboru = new
StreamWriter(Application.StartupPath + @"\data\CountOfFiles.var");
30.                     ZmenPocetSouboru.WriteLine(PocetSouboru + 1);
31.                     ZmenPocetSouboru.Close();
32.                     VytvorJazyk.Close();
33.                     ArrayList jazyky =
Deserializace.DeserializujArrayList(Application.StartupPath +
@"\data\languages.bin");
34.                     jazyky.Add(comboBoxVyberJazyku.Text);
35.                     Serializace.SerializujArrayList(jazyky,
Application.StartupPath + @"\data\languages.bin");
```

```
36.                     ArrayList SeznamJazyku =
Deserializace.DeserializujArrayList(Application.StartupPath +
@"\data\languages.bin");
37.                     textBoxSeznamJazyku.Lines =
VypisJazyku(SeznamJazyku);
38.                     MessageBox.Show("Jazyk " +
comboBoxVyberJazyku.SelectedItem + " byl úspěšně vložen do databáze.",
"Potvrzení", MessageBoxButtons.OK, MessageBoxIcon.Information);
39.
comboBoxVyberJazyku.Items.RemoveAt(comboBoxVyberJazyku.SelectedIndex);
40.                 }
41.             }
```

Do řetězce `text` (3. řádek) se přiřadí buď text ze souboru či text z textBoxVlozenyText (4. – 11. řádek) a pomocí ToLower() se celý převede na malá písmena (velké a malé písmeno je pro nás jeden a ten samý znak, pro počítač je ovšem každý jiný, liší se ordinální hodnotou, proto je třeba je převést na malá či velká). V `comboBoxVyberJazyku`, do kterého se načítá při vytváření formuláře obsah souboru list.lng, musí uživatel vybrat jazyk, jinak vyskočí varovná hláška (13. řádek). Po splnění této podmínky dojde k následujícímu:

- Do proměnné `PocetSouboru` (17. řádek) se načte obsah souboru CountOfFiles.var a toto číslo bude tvořit název souboru *.fqa nového jazyka (20. řádek).
- Pole `NovyJazyk` je naplněno analýzou vstupního textu (19. řádek).
- Pomocí třídy StreamWriter a for cyklu (21. – 28. řádek) se z `NovyJazyk` do vzniklého souboru *.fqa zapíší jednotlivé řádky analýzy; nevkládají se ovšem relativní četnosti všech 65 535 znaků, vyberou se pouze ty znaky, které Visual Studio chápe jako malá písmena všech obsažených abeced – tím se značně sníží nejen objem dat uložených na disku, ale zároveň se zkrátí čas při porovnávání jazyků.
- Po ukončení zápisu se hodnota v souboru CountOfFiles.var zvýší o jedna (30. řádek).
- Poté se deserializací souboru languages.bin naplní pole `jazyky`, přidá se do něj položka odpovídající názvu jazyka, který vybral uživatel z `comboBoxVyberJazyku`.
- Tento nově nainstalovaný jazyk se také přidá do `textBoxSeznamJazyku` a nakonec se ještě odstraní ze seznamu v `comboBoxVyberJazyku`.

Tímto způsobem jsem nahrál do programu většinu evropských jazyků. Analýza byla téměř vždy vytvářena z velmi dlouhého textu (především z Bible), ke které jsem se dostal

v mnoha různých světových jazycích především díky aplikaci Bible Database (dostupná na http://bibledatabase.com/free_bible_software.htm, seznam použitých textů viz Příloha II., s. J-K). Tyto texty byly ovšem v kódování ANSI, proto jsem je musel všechny nejprve převést na UTF-8, abych zachoval citlivost na speciální znaky. Použití tak dlouhých textů sice není nezbytně nutné, ale čím je text delší, tím jsou výsledné relativní četnosti znaků přesnější.

Mám tedy vytvořené *.fqa soubory a můžu začít porovnávat neznámý text. Opět se hlavní událost skrývá pod tlačítkem, tentokrát s popiskem 

:

```
1.    private void btnAnalyzuj_Click(object sender, EventArgs e)
2.            {
3.                string text = "";
4.                if (radioBtnZeSouboru.Checked)
5.                {
6.                    text = File.ReadAllText(soubor).ToLower();
7.                }
8.                else
9.                {
10.                   text = textBoxVlozenyText.Text.ToLower();
11.               }
12.               grafAnalyzyVstupu.Series[0].Clear();
13.               double[] AnalyzaVstupu = Analyza.AnalyzaTextu(text);
14.               for (int i = 0; i < AnalyzaVstupu.Length; i++)
15.               {
16.                   if (AnalyzaVstupu[i] > 0 && char.IsLetter((char)i))
17.                   {
18.                       grafAnalyzyVstupu.Series[0].Add(AnalyzaVstupu[i],
Convert.ToString((char)i));
19.                   }
20.               }
21.               double[] AnalyzaUpravena = FiltrujPismena(AnalyzaVstupu);
22.               StreamReader CountOfFiles = new
StreamReader(Application.StartupPath + @"\data\CountOfFiles.var");
23.               int PocetSouboru =
Convert.ToInt32(CountOfFiles.ReadLine());
24.               CountOfFiles.Close();
25.               double[] rozdily = new double[PocetSouboru];
26.               for (int i = 0; i < PocetSouboru; i++)
27.               {
28.                   string soubor = Application.StartupPath + @"\data\" +
i + ".fqa";
```

```
29.                     StreamReader analyza = new StreamReader(soubor);
30.                     double[] pole = new double[AnalyzaUpravena.Length];
31.                     double[] vysledek = new
double[AnalyzaUpravena.Length];
32.                     double suma = 0;
33.                     for (int j = 0; j < AnalyzaUpravena.Length; j++)
34.                     {
35.                         pole[j] = Convert.ToDouble(analyza.ReadLine());
36.                         vysledek[j] = Math.Abs(AnalyzaUpravena[j] -
pole[j]);
37.                         suma = suma + vysledek[j];
38.                     }
39.                     rozdily[i] = suma;
40.                     analyza.Close();
41.                 }
42.                 ArrayList SeznamJazyku =
Deserializace.DeserializujArrayList(Application.StartupPath +
@"\data\languages.bin");
43.                 labelVyslednyJazyk.Visible = true;
44.                 labelVyslednyJazyk.Text =
Convert.ToString(SeznamJazyku[Minimum.IndexMinimalniHodnoty(rozdily)]);
45.             }
```

Dle radiobuttonů se do řetězce `text` přiřadí text ze vstupu převedený na malá písmena (4. – 11. řádek). Pole `AnalyzaVstupu`, které je typu double, je naplněno metodou AnalyzaTextu (13. řádek). Následuje for cyklus s podmínkou, která propustí pouze nenulové hodnoty a pouze písmena, a ten naplní graf `grafAnalyzyVstupu` hodnotami četností vstupního řetězce. Zde je pole stále hodně velké, indexy totiž představují ordinální hodnoty znaků; nicméně po načtení hodnot do grafu již ordinální hodnoty nebudou potřeba, a tak se v dalším kroku pole z 65 535 zmenší na pouhých 1 102.

Proč zrovna 1 102? Toto číslo není žádný odhad, ale je to počet různých znaků, které Visual Studio z celé množiny typu char vyhodnotilo jako malé písmeno (IsLetter(), ToLower()) některé z abeced obsažených v tabulce Unicode. Přesně tolik řádků mají také soubory *.fqa. Dalo by se namítnout, že je to stále velmi neúsporné, ukládat četnosti např. 26 základních znaků latinky do tisíce řádků. Tento fakt je ale vyvážen jednoduchým algoritmem porovnávání jazyků, který může probíhat automaticky bez složitých přepočtů ordinálních hodnot odpovídajících si znaků. Navíc, soubory *.fqa zabírají na disku každý pouze cca 4kB a výpočetní výkon současných počítačů si s tím bez problému poradí.

Metoda FiltrujPismena (21. řádek) tedy propustí jen malá písmena všech abeced, které jsou v množině char a naplní se tak pole AnalyzaUpravena. Do proměnné `PocetSouboru` se načte číslo ze souboru CountOfFiles.var, které udává horní hranici následujícího for cyklu (26. – 41. řádek), který má na starosti již samotné porovnávání. Tento cyklus se tedy provede tolikrát, kolik je nainstalovaných jazyků, s každým souborem *.fqa ze složky Data se provede toto:

Soubor je otevřen, začnou se z něj načítat jednotlivé řádky, které jsou ihned porovnávány s prvky pole `AnalyzaUpravena` (for cyklus na 33. – 38. řádku). Porovnávání pracuje na principu absolutní hodnoty rozdílu aktuálně srovnávaných řádků (resp. prvků v polích), která je pro každý jazyk nasčítána do proměnné `suma`, a toto číslo se pak zapíše do pomocného pole `rozdily`, které shromažďuje právě tyto jednotlivé součty rozdílů pro každý jazyk z databáze.

Po naplnění celého pole `rozdily` (tj. po komparaci pole `AnalyzaUpravena` se všemi dostupnými soubory *.fqa) se deserializací naplní pole `SeznamJazyku` (42. řádek) ze souboru languages.bin a na cílový `labelVyslednyJazyk` (44. řádek) se vypíše název jazyka z tohoto pole, jehož index je dán indexem minimální hodnoty v poli `rozdily` (Metoda `IndexMinimalniHodnoty` projde pole `rozdily`, najde v něm minimum a vrátí index tohoto prvku. Tento index pak odpovídá jazyku v poli `SeznamJazyku` s nejlepší shodou s vloženým textem .). Výsledek je vypsán, procedura končí.

3.5 Problémy při vývoji programu

Jak jsem již dříve naznačil, prvotní problém byl výběr vývojového prostředí. Ten byl však odstraněn a na řadu přišlo programování.

Samotný porovnávací algoritmus jsem měl v hlavě, překážkou však byla realizace získávání, uchovávání a přístupu k frekvenčním analýzám. Princip programu se mi podařilo ověřit v tabulkovém procesoru MS Excel (viz CD-ROM – Příloha IV. – komparace.xls, list Porovnávání), proto mě napadlo, že bych analýzy nainstalovaných jazyků hromadil v souboru *.xls, ke kterému bych přistupoval skrze moji aplikaci. MS Visual Studio 2008 sice neumí přímo tvořit a pracovat s dokumenty kancelářského balíku MS Office, nicméně umožňuje spuštění aplikace MS Office na pozadí a ovládání této aplikace skrze kód programu. To má tedy velkou nevýhodu v tom, že tento postup vyžaduje nainstalovaný balík MS Office na počítači, kde je spuštěn můj program (využívá totiž různé části tohoto software – jak samotné programy, tak i jejich knihovny). Navíc se mi tento přístup nepodařilo zprovoznit.

Po pročtení množství webů jsem zkusil k excelovskému souboru přistupovat jako k databázi, to bylo zase zbytečně složité. Další pátrání mě utvrdilo v tom, že ukládání do Excelu nebude dobrá volba, protože bych k tomu potřeboval nějakou komerční pomocnou knihovnu *.dll, které jsou velmi drahé. Freewarové ekvivalenty byly k mání pouze jen jako „writery“ (pouze zápis do souboru, nikoliv čtení z něj).

Po odborné konzultaci mi byla doporučena serializace, od které jsem také nakonec upustil a rozhodl se vyzkoušet prosté uložení analýz do textových souborů. Přístup k nim je jednoduchý, soubory lze plnit i číst pomocí cyklů. Kostru programu jsem si tedy poté vytvořil v konzolovém režimu.

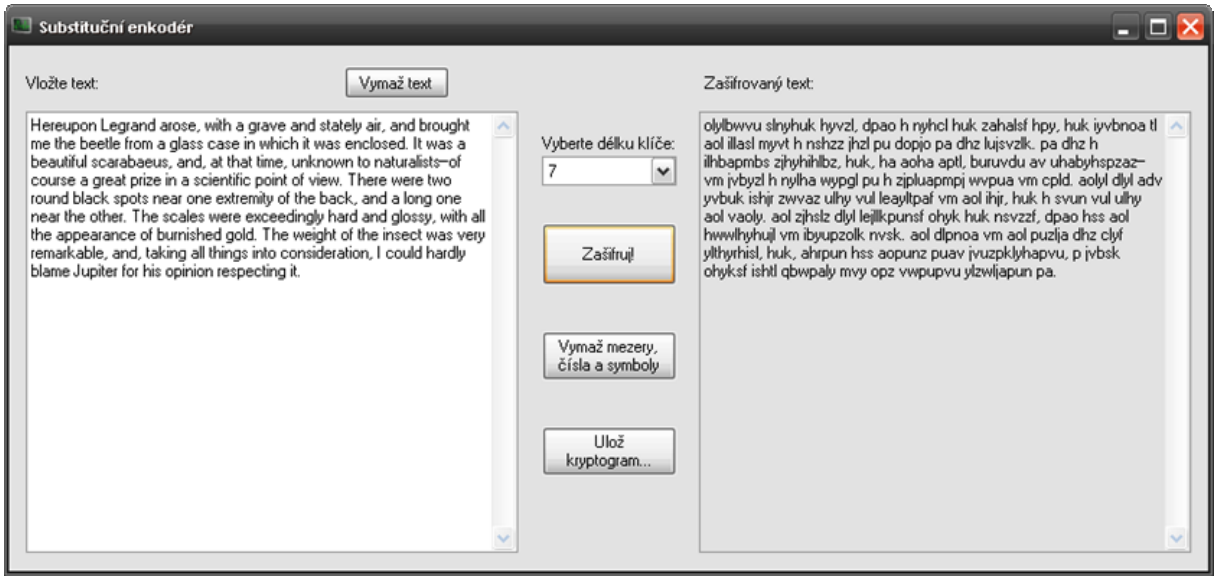
Když už byl program v podstatě vytvořen, bylo potřeba přejít ke GUI[7]. Ve Visual Studiu je práce na GUI velmi intuitivní, prostředí obsahuje všechny důležité komponenty, až na jednu. Primárně totiž nemá komponentu pro vykreslování a práci s grafy. Vyzkoušel jsem opět různá řešení pomocí externích doplňků, nakonec vyhrála komponenta TeeChart (http://www.steema.com/), která je de facto vylepšený sharewarový klon komponenty TChart z prostředí Delphi, kterou jsem uměl ovládat z předešlých studií na PedF. Jedinou její nevýhodou je, že verze zdarma plně funguje prvních 50 dní, poté má již omezené schopnosti a při spuštění programu hlásí vypršení platnosti. Grafy přesto ale vykresluje.

3.6 Frekvenční analýza 2.0

Požadavkem na druhou verzi programu bylo vytvoření aplikace, které bude rozpoznávat jazyk zadaného kryptogramu, který bude šifrován jednoduchou substituční šifrou. Tak vznikl program Frekvenční analýza 2.0, který je jen lehkou modifikací předchozí verze.

Hlavní rozdíl, který verze 2.0 obsahuje, je ten, že z analýz jazyků vypouští specifické znaky jednotlivých abeced a tím pádem se rapidně snížila rozpoznávací schopnost mého algoritmu. Vstupní abeceda je tedy omezena jen na základních 26 znaků latinky (a – z), proto byly také vypuštěny jazyky, které využívají cyrilice nebo alfabety. Pro tvoření kryptogramů byla naprogramována aplikace Substituční enkodér (Obr. 7), který vložený text zbaví diakritiky, mezer a speciálních znaků. Po zvolení délky posunu abecedy text zašifruje substituční metodou.

[7] Graphical User Interface – grafické uživatelské prostředí



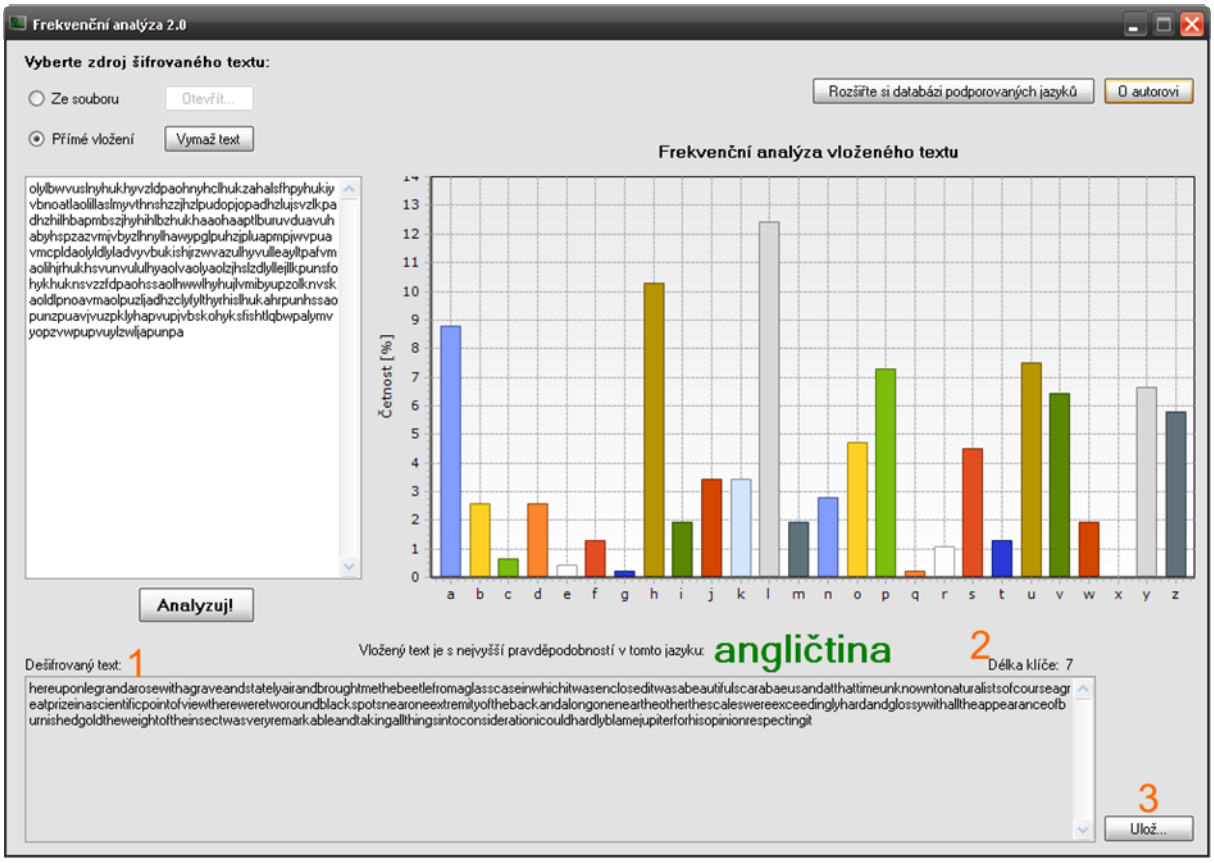
Obr. 7 – Prostředí aplikace Substituční enkodér

Frekvenční analýza 2.0 funguje tak, že zanalyzuje vložený kryptogram; nalezne index znaku, který je nejčastěji zastoupený. Pak pomocí for cyklu opět porovnává vstupní text s nainstalovanými jazyky. Rozdíl je ale v tom, že se nejprve u každého nainstalovaného jazyka také najde index nejčetnějšího znaku, pomocí rozdílu pozic těchto písmen se najde klíč a provede se dešifrování vstupního kryptogramu. Pak se provede analýza otevřeného textu a nakonec přichází komparace, která je principielně založena na první verzi programu.

Úpravy se dočkalo i GUI (Obr. 8), ve kterém přibyl panel zobrazující dešifrovaný text (1), štítek oznamující délku klíče (2) a také tlačítko Ulož (3), které umožňuje uložení získaného otevřeného textu do souboru.

Algoritmus porovnávání byl původně pro zlepšení výsledků upraven z jednoduchého součtu absolutních hodnot rozdílů četností znaků na součet absolutních hodnot rozdílů čtverců relativních četností. Tento postup však vykazoval paradoxně větší chybovost rozpoznávání kryptogramů než původní verze, proto od něj bylo upuštěno.

Zde byl vývoj druhé verze ukončen s tím, že je zde do budoucna možnost vylepšení rozpoznávacího algoritmu.



Obr. 8 – Upravené rozhraní verze 2.0

3.7 Rozpoznávání na základě 10 nejčetnějších znaků

Pro tento účel byl vytvořen pseudoprogram v Excelu (viz CD-ROM – Příloha IV. – komparace.xls, list 10 nejčetnějších). Nejprve bylo potřeba získat přehled deseti nejfrekventovanějších písmen a jejich procentuální zastoupení pro každý jazyk. Následovalo vymýšlení algoritmu. Bylo potřeba najít vzorec, který by dokázal spárovat odpovídající si znaky vstupního textu a porovnávané analýzy; aby dokázal místo vyhození chybové hlášky #N/A při nenalezení jistého znaku vložit do buňky nějaké číslo. Takový vzorec jsem našel, vypadá následovně:

```
=KDYŽ(JE.NEDEF(SVYHLEDAT($A19;$A$4:$B$13;2;NEPRAVDA));100;SV
YHLEDAT($A19;$A$4:$B$13;2;NEPRAVDA)).
```

Nápověda pro MS Excel o funkci SVYHLEDAT píše: *„Tato funkce vyhledá v prvním sloupci tabulky zadanou hodnotu a vrátí hodnotu odpovídající buňky ve stejném řádku jiného sloupce tabulky.“* Buňka A19 obsahuje první nejčetnější znak vstupního textu, který je vyhledáván v analýze prvního jazyka – v tabulce A4:B13 (sloupec A4:A13 obsahuje 10 znaků, sloupec B4:B13 příslušné četnosti). Třetí parametr funkce SVYHLEDAT je číslo sloupce, z něhož se

bude vracet odpovídající hodnota (v mém případě 2), posledním parametrem je typ – určuje, zda má funkce SVYHLEDAT najít přesnou nebo pouze přibližnou hodnotu (v mém případě je parametr zvolen jako NEPRAVDA, tedy vrací pouze přesnou shodu). Vzorec je ošetřen podmínkou KDYŽ – v případě, že znak není nalezen v analýze, je funkce SVYHLEDAT nedefinovaná (chybová hláška #N/A), a proto se buňka vyplní číslem 100 (zvýší se tedy výsledná suma absolutních hodnot rozdílů čtverců a tím pádem se sníží pravděpodobnost toho, že tento jazyk bude vhodným kandidátem při rozpoznávání vstupu), v případě nálezu znaku v analýze se do buňky vedle hledaného znaku přiřadí hodnota četností z analýzy. Ve vedlejším sloupci se již počítá rozdílnost ošetřená opět podmínkou:

```
=KDYŽ(D19=100;100;ABS($B19*$B19-D19*D19)),
```

která říká, že když je v buňce D19 (tedy buňka, ve které je vzorec uvedený výše) hodnota 100, pak v této buňce bude také hodnota 100, a pokud ne, vypočítá se absolutní hodnota rozdílu čtverců četností. Takto se vyhodnotí každý znak vstupu s analýzami všech 38 jazyků, u každého se provede součet oněch absolutních hodnot pomocí funkce SUMA. Stranou je buňka obsahující funkci MIN, která prohledá řádek součtů a najde minimální hodnotu. Toho je využito pro Podmíněné formátování, kdy se projde tento řádek součtů a buňka s hodnotou odpovídající buňce s funkcí MIN je barevně odlišena a tím je dám i výstup – určený jazyk.

4 Zpracování výsledků

Tato práce je primárně zaměřena na rozpoznávání jazyků textu či kryptogramu, vedlejším produktem jsou i zajímavé výsledky. Jelikož se na světě vyskytuje asi 7000 jazyků, byl jsem nucen se omezit na určitou oblast, abych byl vůbec schopen nějaké výsledky zpracovat. Proto jsem se zaměřil na státy Evropy, respektive na 38 úředních oficiálních jazyků. Původní záměr byl pokrýt celou Evropu, bohužel byl problém sehnat texty v elektronické podobě v černohorštině, kazaštině, lucemburštině, rétorománštině a moldavštině (i když ta je dle (Price et al. 2002, s. 291) ve skutečnosti rumunštinou, a proto jsem s ní tak zacházel), a tak jsou tyto jazyky vynechány.

Přesto se mi ale podařilo sesbírat 38 jazykových mutací nejznámější knihy světa – Bible. V některých jazycích jsem narazil jen na první knihu Starého zákona (Genesis), u jiných zase Žalmy či evangelium podle Lukáše. Celkový seznam použitých textů je v Příloze II. (s. J-K), texty samotné se nacházejí na CD-ROMu.

Uvědomuji si, že analýzy jazyků jsem měl nejlépe provádět na stejně dlouhých textech, nicméně relativní četnosti se ustálí na svých hodnotách po několika analyzovaných stránkách, s dalším počtem stran se již příliš nemění, pouze se zpřesňují. Samozřejmě jsem se snažil získat co nenovější překlady Bible, aby analýzy odpovídaly co nejpřesněji dnešním podobám evropských jazyků.

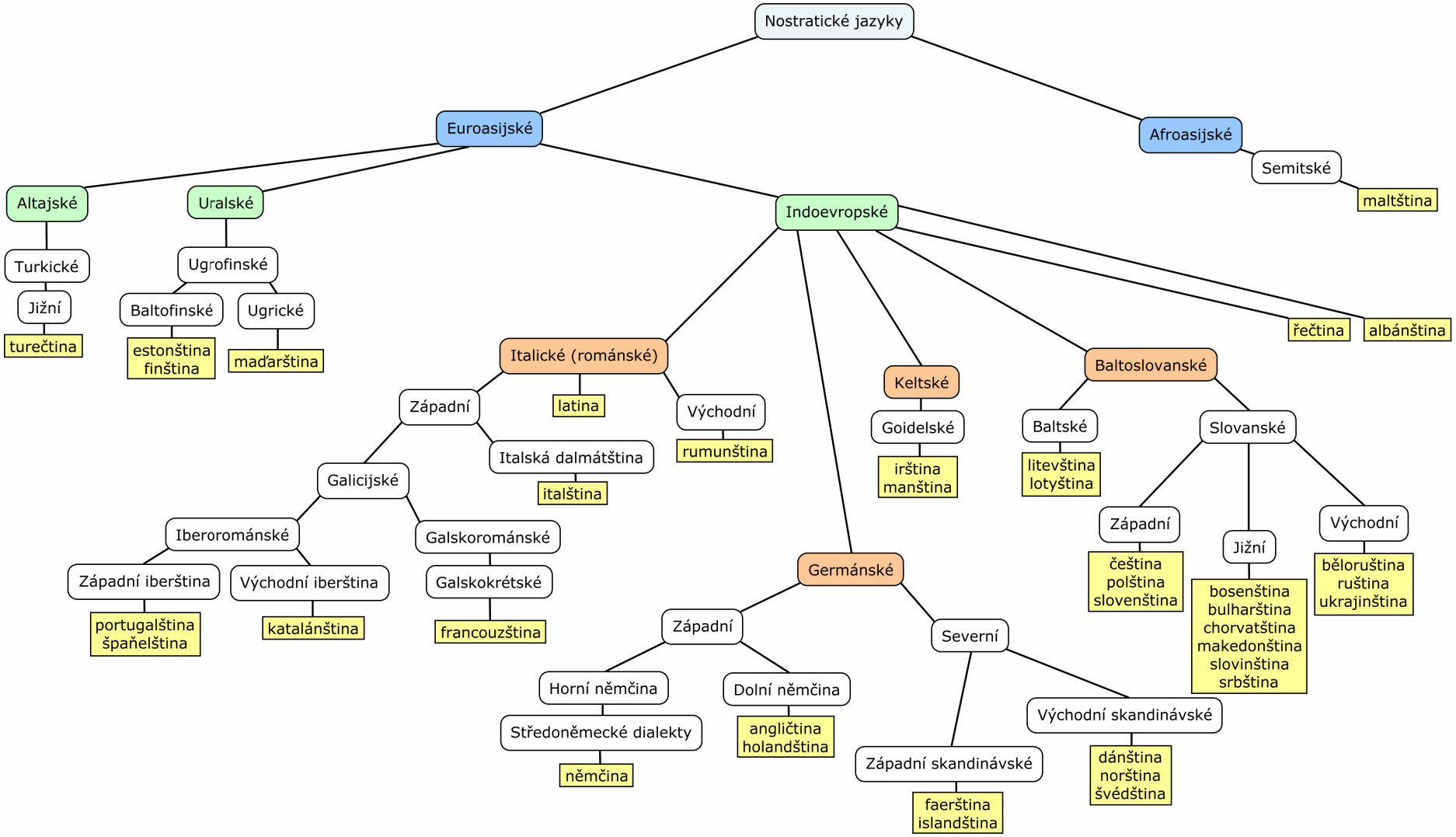
4.1 Evropské jazyky

Zde uvádím velmi stručné informace o jazycích, které jsem analyzoval. Odhalují reálné příbuznosti, které jsou pro přehlednost zpracovány v diagramu na Obr. 9 dle (Comrie, Matthews, Dolinsky 2007, s. 38-47). Vše doslovně citováno z (Price et al. 2002):

Albánština *(gjuha shqipe) je jediným existujícím reprezentantem jedné indoevropské jazykové větve, do níž pravděpodobně (což je dosud nejisté) patřila ilyrština a messapština. Mluví se jí převážně v západní části balkánského poloostrova, především v Albánské republice (...).*

Angličtina *patří k západní skupině germánských jazyků, nejblíže příbuzná je jí fríština. (...) V Evropě je rozšířena hlavně na Britských ostrovech (...). Na celém světě je podle odhadu prvním jazykem asi 377 milionů lidí.*

Běloruština *patří do východní větve slovanských jazyků. Je úředním jazykem Běloruské republiky (1991).*



Obr. 9 – Diagram reálné příbuznosti jazyků

Bosenština *je úřední název varianty srbochorvatštiny v Republice Bosna a Hercegovina. Bosenština je v podstatě mluva bosenských muslimů, kteří se snaží v mluveném jazyce odlišit jak od srbštiny, tak i chorvatštiny vyšší frekvencí slov neslovanského původu (tureckých, arabských a perských).*

Bulharština *spolu se slovinštinou, srbochorvatštinou (srbštinou a chorvatštinou) a makedonštinou patří do skupiny jižních slovanských jazyků. Bulharštinou se mluví hlavně v Bulharsku, ale jejími dialekty i jinde na Balkáně – v sousedním Srbsku, v Řecku, v rumunském Banátu a v bývalém Sovětském svazu.*

Čeština *spolu se slovenštinou, polštinou a lužickou srbštinou patří do západní podskupiny slovanských jazyků. Mluví se jí především v České republice, kde je úředním jazykem.*

Dánština *patří k severní skandinávské podskupině germánských jazyků. Kromě toho, že je jediným jazykem pevninského Dánska, figuruje také jako druhý úřední jazyk (společně s faerštinou) na Farských ostrovech a (s grónštinou) v Grónsku.*

Estonština *(eesti keel) patří s finštinou k baltofinské podskupině ugrofinských jazyků a je vzdáleněji příbuzná s maďarštinou. Mluví se jí hlavně v Estonské republice, kde je úředním jazykem.*

Faerština *patří do severogermánské podskupiny germánských jazyků a mluví se jí na Faerských ostrovech.*

Finština *spolu s estonštinou patří do baltofinské podskupiny ugrofinských jazyků a vzdáleněji je příbuzná s maďarštinou.*

Francouzština *patří do galorománské větve románských jazyků a mluví se jí (v Evropě) ve Francii, v jižní Belgii (Valonsku), v západním Švýcarsku, Lucembursku, Monaku, ve Val d'Aosta v Itálii, a na Normanských ostrovech.*

Chorvatština *(Srbochorvatština). Jazyk (jazyky) jižní podskupiny slovanských jazyků, který (které) patří spolu se slovinštinou k západní větvi této podskupiny. Název „srbochorvatština" je běžný v západní jazykovědě a široce se ho používalo v bývalé Jugoslávii. Vedle tohoto názvu se setkáváme v chorvatské jazykovědě s názvy jako „chorvatosrbština" (chorvato-srbština), „chorvatština nebo srbština".*

Irština *patří ke goidelské neboli gaelské větvi keltských jazyků. Je úředním jazykem Irské republiky.*

Islandština *patří k západní větvi skandinávské podskupiny germánských jazyků a je úředním jazykem na Islandu.*

***Italština**, člen románské jazykové skupiny, pocházející z latiny, je úředním jazykem Italské republiky, Vatikánu a republiky San Marino. Je rovněž jedním z úředních jazyků Švýcarska a do roku 1934 byla i úředním jazykem na Maltě.*

***Katalánština** patří do skupiny románských jazyků a mluví se jí v severovýchodní části Španělska (a na Baleárách), ve francouzském departmentu Východní Pyreneje, v Andoře a v přístavu Alghero na Sardinii.*

***Latina**, jazyk starého Říma a okolí, patřící do italické podskupiny indoevropských jazyků, se stala úředním jazykem a všeobecným dorozumívacím prostředkem a římské říši a jejích dědických státech. V psané formě se udržela jako učený jazyk v západní Evropě, a tuto funkci si v omezeném množství kontextů uchovala dodnes. Z jejích hovorových forem se vyvinuly románské jazyky.*

***Litevština** je jedním z živých jazyků (druhým je lotyština) východní skupiny baltských jazyků, jímž se mluví v samotné Litvě, ale také v rámci menšin v sousedních zemích a v emigrantských komunitách v západní Evropě (včetně Velké Británie), v Severní Americe a Austrálii.*

***Lotyština** (...) patří k východní skupině baltských jazyků. Mluví se jí především v samotném Lotyšsku, ale také v komunitách žijících jinde na východě a severu Evropy (včetně Británie), v Severní Americe a Austrálii.*

***Maďarština** patří do ugrofinské podskupiny uralských jazyků. K dalším členům této podskupiny patří vogulština (mansijština) a osťačtina (chanština). Většina maďarských mluvčích (zhruba 10 milionů) dnes žije v Maďarské republice.*

***Makedonština**. Jižní slovanský jazyk spolu se srbštinou a chorvatštinou (srbochorvatštinou), slovinštinou a bulharštinou. Úřední jazyk bývalé jugoslávské Svazové republiky Makedonie, která roku 1991 vyhlásila nezávislost.*

***Maltština**. Semitský jazyk, historický dialekt arabštiny (maghribská skupina), dostatečně odlišný od ostatních arabských dialektů (...), aby mohl být považován za samostatný jazyk.*

***Manština** (Manx, dříve psaná Manks), manská gaelština, patří do goidelské neboli gaelské podskupiny keltských jazyků (...). Manština se stala typickou variantou gaelštiny, jíž se mluví na Isle of Man uprostřed Irského moře (...).*

***Němčina** patří do západogermánské podskupiny germánských jazyků a je jí nejbližší nizozemština, angličtina, fríština a jidiš. V Evropě jí mluví přes 100 milionů lidí a je národním jazykem v Německu, Rakousku a Lichtenštejnsku, jedním z národních jazyků ve Švýcarsku (...) a Lucembursku (...).*

Nizozemština *patří k západní skupině germánských jazyků, je nejblíž příbuzná s dolní a horní němčinou, jíž se mluví na východ od nizozemského jazykového území. Vlámština se ve svém nejpřísnějším smyslu řadí ke skupině nizozemského nářečí (...).*

Norština *spolu s islandštinou a faerštinou patří k západní podskupině skandinávských jazyků, jež společně tvoří skupinu jazyků germánských. Norštinou mluví asi 4,2 milionu obyvatel Norska (...). Existují dvě psané formy jazyka, tzv. nynorsk a bokmål (...).*

Polština *patří k lechické větvi (její východní a nerozšířenější jazyk) západní podskupiny slovanských jazyků. Je úředním jazykem Polské republiky, kde jí mluví 35-40 milionů obyvatel.*

Portugalština *patří do iberorománské podskupiny románských jazyků. V Evropě se jí mluví v kontinentálním Portugalsku, na atlantických ostrovech Madeira a Azory a v několika jazykových enklávách ve Španělsku (..).*

Rumunština*. Románský jazyk, jímž se mluví hlavně v Rumunsku a Moldavské republice. Dialekty, resp. Jazyky (rumunština, istrorumunština, meglenorumunština) podstatně odlišné od vlastní rumunštiny, nacházíme na různých místech na Balkáně (...).*

Ruština *patří k východní podskupině slovanských jazyků a je blízce příbuzná s ukrajinštinou a běloruštinou. Nyní se jí mluví především na území Ruské federace a její znalost existuje (v různé míře) v ostatních nástupnických státech bývalého Sovětského svazu i v zemích bývalého sovětského bloku.*

Řečtina *představuje samostatnou větev indoevropské jazykové rodiny, i když je pravděpodobné, že kdysi tvořila poměrně uzavřenou podskupinu indoíránských jazyků a arménštiny uvnitř indoevropštiny.*

Slovenština*. Spolu s češtinou, polštinou a lužickou srbštinou patří k západní podskupině slovanských jazyků. Mluví se jí hlavně ve Slovenské republice, kde má status úředního jazyka.*

Slovinština *patří do jižní skupiny slovanských jazyků a spolu se srbochorvatštinou tvoří západní větev této podskupiny. Standardní spisovná slovinština je úředním jazykem Slovinské republiky, která je nezávislým státem od r. 1991.*

Srbština *viz chorvatština (srbochorvatština).*

Španělština*. Románský jazyk, blízce příbuzný portugalštině, galicijštině a katalánštině. Společně s těmito jazyky tvoří hispánskorománskou nebo také iberorománskou podskupinu románské jazykové rodiny.*

Švédština*. Severní germánský (skandinávský) jazyk, jímž se mluví ve Švédsku a v různých částech Finska. V obou těchto zemích je úředním jazykem.*

Turečtina *(Türkçe) je jeden z jazyků oghuzské neboli jihozápadní podskupiny turkických (tureckých) jazyků; mluví se jí v Turecku, na Kypru (120 tis.) a v různých částech Balkánu.*

Ukrajinština *patří do východní podskupiny slovanských jazyků a je jediným úředním jazykem v Ukrajinské republice.*

4.2 Statistická příbuznost v porovnání s reálnou příbuzností

Otázka, která se vyskytla před psaním této kapitoly, zněla: „Bude se reálný vývoj a příbuznost jazyků promítat i do příbuznosti statistické, která vychází z frekvenčních analýz těchto jazyků?“

Abych na tuto otázku mohl odpovědět, musel jsem zpracovat vstupní informace, které jsem měl v podobě analýz z Biblí (grafy analýz viz Příloha III., s. L-X). Opět jsem pro přehlednost a jednoduchost zvolil MS Excel. Využil jsem již zmiňovaného souboru komparace.xls (Příloha IV. na CD-ROM, list Porovnávání), který pracuje přesně jako program Frekvenční analýza 1.0. Do sloupce A listu Porovnávání jsem vždy vložil zkopírovanou analýzu porovnávaného jazyka (sám vůči sobě má tedy nulový rozdíl) a řádek 2, který obsahuje sumy absolutních hodnot rozdílů, jsem zkopíroval na list Statistická příbuznost. Prošel jsem tedy všech 38 jazyků totožným způsobem. Dále bylo potřeba tabulku trochu upravit – každá buňka, která obsahovala číslo 200 (tedy maximální rozdíl způsobený využitím jiné abecedy), byla vyplněna znakem „*“; nulové rozdíly (tedy porovnání dvou totožných jazyků) byly proškrtány. Nakonec jsem barevně odlišil první tři statisticky nejbližší jazyky (zelená – první nejbližší, žlutá – druhý nejbližší, červená – třetí nejbližší). Výsledná tabulka je na Obr. 10.

Přestože jsou vzájemné vazby zvýrazněny u všech jazyků, relevantní jsou jen ty, které mají rozdílnost menší než 50 %. Z toho je tedy vidět, že albánština, irština, maltština a turečtina nemají nejbližší příbuzné v rámci ostatních jazyků a to téměř přesně (až na irštinu, která je reálně příbuzná s manštinou) odpovídá realitě, neboť se právě tyto jazyky nacházejí v rozdílných větvích diagramu vývoje (Obr. 9). Řečtina v tabulce stojí úplně osamocená, protože jako jediná využívá alfabetu, což opět odpovídá realitě. Vypadá to tedy, že se pomalu začíná rýsovat odpověď na původní otázku.

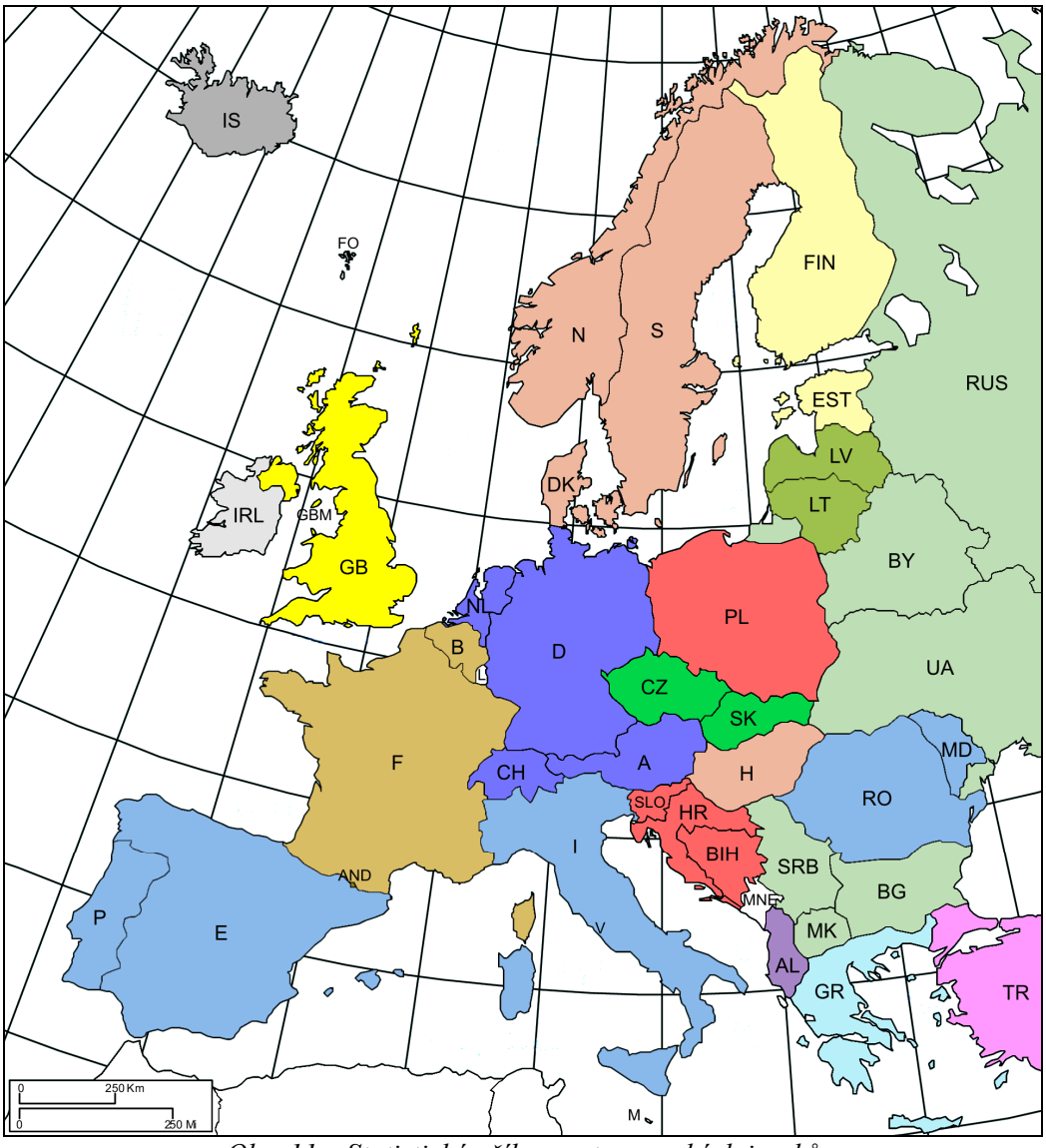
Angličtina se statisticky podobá manštině, španělštině a němčině, reálně z této trojice vychází pouze němčina v rámci západní germánské větve indoevropských jazyků. Běloruština má nejblíže ukrajinštině, to odpovídá skutečné příbuznosti.

	ALB	ANG	BĚL	BOS	BUL	ČEŠ	DÁN	EST	FAE	FIN	FRA	CHO	IRŠ	ISL	ITA	KAT	LAT	LIT	LOT	MAĎ	MAK	MAL	MAN	NĚM	NIZ	NOR	POL	POR	RUM	RUŠ	ŘEČ	SK	SLO	SRB	ŠPA	ŠVÉ	TUR	UKR
ALB		50,28	*	52,30	*	61,68	56,82	57,55	57,50	59,49	53,07	51,23	72,09	64,13	57,65	56,73	50,61	60,99	60,98	67,28	*	65,75	67,72	54,62	58,22	56,37	65,75	62,89	61,63	*	*	59,87	54,89	*	59,66	53,03	64,06	*
ANG	50,28		*	57,44	*	58,57	43,97	59,22	61,08	62,50	42,77	56,66	58,82	64,14	40,99	47,17	41,80	67,03	70,97	58,38	*	73,83	34,75	40,18	46,59	45,37	60,18	48,23	57,93	*	*	54,67	52,75	*	39,67	42,79	63,96	*
BĚL	*	*		*	66,29	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	69,63	*	*	*	*	*	*	*	*	61,31	*	*	*	71,99	*	*	*	49,69
BOS	52,30	57,44	*		*	43,43	58,76	42,90	59,30	55,78	60,60	6,58	69,12	65,66	41,30	52,40	53,86	38,94	41,45	63,27	*	56,05	66,95	65,20	58,82	62,37	49,09	43,71	56,10	*	*	35,49	18,95	*	47,33	54,71	57,83	*
BUL	*	*	66,29	*		*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	21,88	*	*	*	*	*	*	*	*	24,28	*	*	*	27,29	*	*	*	38,25
ČEŠ	61,68	58,57	*	43,43	*		60,86	64,23	64,74	65,58	63,79	41,95	79,48	67,01	60,87	62,67	70,95	60,27	61,58	57,21	*	78,28	71,34	72,28	65,52	61,92	59,52	61,88	69,91	*	*	21,72	40,02	*	51,74	59,19	71,28	*
DÁN	56,82	43,97	*	58,76	*	60,86		56,99	46,72	65,04	45,91	59,03	66,37	48,90	52,18	47,84	56,31	67,79	69,91	52,99	*	70,59	46,63	40,00	35,90	15,01	74,25	55,04	64,39	*	*	58,94	51,22	*	49,71	27,77	64,50	*
EST	57,55	59,22	*	42,90	*	64,23	56,99		56,77	35,51	47,70	42,90	67,54	64,07	48,17	40,21	51,15	46,43	45,40	53,33	*	51,38	66,51	61,64	64,17	53,36	69,98	51,22	56,17	*	*	64,35	44,28	*	50,47	49,45	55,13	*
FAE	57,50	61,08	*	59,30	*	64,74	46,72	56,77		52,39	62,90	56,72	52,91	18,09	58,45	58,44	62,53	55,65	57,71	55,99	*	62,17	61,35	59,78	63,21	48,91	75,72	68,03	64,16	*	*	58,58	60,42	*	62,81	43,07	60,27	*
FIN	59,49	62,50	*	55,78	*	65,58	65,04	35,51	52,39		58,65	52,53	76,31	60,53	58,39	55,94	56,60	47,84	55,20	59,38	*	61,62	69,97	67,45	66,47	60,10	74,19	69,92	60,70	*	*	63,83	58,59	*	61,32	49,41	59,36	*
FRA	53,07	42,77	*	60,60	*	63,79	45,91	47,70	62,90	58,65		58,40	66,88	65,20	32,08	22,50	27,72	59,01	58,43	58,69	*	71,43	50,96	43,93	53,11	42,99	72,03	34,98	40,53	*	*	61,93	54,43	*	32,11	50,75	63,84	*
CHO	51,23	56,66	*	6,58	*	41,95	59,03	42,90	56,72	52,53	58,40		69,17	63,93	40,58	50,85	51,90	37,03	39,86	62,87	*	55,37	65,57	65,37	58,55	62,34	48,61	47,06	53,25	*	*	34,47	16,54	*	46,95	54,57	56,56	*
IRŠ	72,09	58,82	*	69,12	*	79,48	66,37	67,54	52,91	76,31	66,88	69,17		54,25	59,70	57,72	64,49	72,72	73,29	76,44	*	75,45	57,19	56,62	72,52	74,34	76,73	63,46	68,95	*	*	74,86	75,56	*	56,79	60,31	71,44	*
ISL	64,13	64,14	*	65,66	*	67,01	48,90	64,07	18,09	60,53	65,20	63,93	54,25		63,75	59,06	69,88	64,92	68,28	57,32	*	68,53	61,87	59,62	65,55	55,45	78,89	67,97	70,38	*	*	61,16	65,62	*	61,63	45,38	63,35	*
ITA	57,65	40,99	*	41,30	*	60,87	52,18	48,17	58,45	58,39	32,08	40,58	59,70	63,75		33,10	33,73	55,23	60,87	60,94	*	60,98	49,57	50,46	56,54	53,30	59,29	36,55	32,14	*	*	54,54	37,23	*	30,59	47,94	54,61	*
KAT	56,73	47,17	*	52,40	*	62,67	47,84	40,21	58,44	55,94	22,50	50,85	57,72	59,06	33,10		36,24	54,19	56,40	58,17	*	61,71	48,49	48,09	59,82	47,87	71,95	29,21	45,26	*	*	59,71	50,44	*	26,36	46,50	57,87	*
LAT	50,61	41,80	*	53,86	*	70,95	56,31	51,15	62,53	56,60	27,72	51,90	64,49	69,88	33,73	36,24		51,79	53,26	65,94	*	63,31	57,91	51,74	61,21	54,03	64,41	39,85	34,47	*	*	63,81	55,31	*	41,52	55,51	67,58	*
LIT	60,99	67,03	*	38,94	*	60,27	67,79	46,43	55,65	47,84	59,01	37,03	72,72	64,92	55,23	54,19	51,79		37,00	70,27	*	61,12	72,62	75,92	72,61	67,83	60,16	58,00	57,66	*	*	54,23	47,32	*	57,42	63,72	67,36	*
LOT	60,98	70,97	*	41,45	*	61,58	69,91	45,40	57,71	55,20	58,43	39,86	73,29	68,28	60,87	56,40	53,26	37,00		68,40	*	62,47	78,28	70,88	72,23	70,05	64,73	62,96	58,95	*	*	56,28	49,86	*	63,66	62,14	63,80	*
MAĎ	67,28	58,38	*	63,27	*	57,21	52,99	53,33	55,99	59,38	58,69	62,87	76,44	57,32	60,94	58,17	65,94	70,27	68,40		*	69,72	65,24	68,59	61,65	50,71	68,50	68,63	68,48	*	*	57,99	53,80	*	58,18	46,19	62,91	*
MAK	*	*	69,63	*	21,88	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*		*	*	*	*	*	*	*	*	29,77	*	*	*	20,43	*	*	*	45,18
MAL	65,75	73,83	*	56,05	*	78,28	70,59	51,38	62,17	61,62	71,43	55,37	75,45	68,53	60,98	61,71	63,31	61,12	62,47	69,72	*		84,03	74,85	73,11	73,19	71,04	74,49	65,97	*	*	77,15	59,34	*	73,23	63,24	65,92	*
MAN	67,72	34,75	*	66,95	*	71,34	46,63	66,51	61,35	69,97	50,96	65,57	57,19	61,87	49,57	48,49	57,91	72,62	78,28	65,24	*	84,03		45,66	50,62	48,54	68,91	50,23	66,73	*	*	67,17	61,08	*	42,44	45,22	61,71	*
NĚM	54,62	40,18	*	65,20	*	72,28	40,00	61,64	59,78	67,45	43,93	65,37	56,62	59,62	50,46	48,09	51,74	75,92	70,88	68,59	*	74,85	45,66		30,42	38,34	73,88	53,41	60,95	*	*	69,87	65,17	*	51,05	47,09	57,40	*
NIZ	58,22	46,59	*	58,82	*	65,52	35,90	64,17	63,21	66,47	53,11	58,55	72,52	65,55	56,54	59,82	61,21	72,61	72,23	61,65	*	73,11	50,62	30,42		36,45	66,93	62,87	66,93	*	*	61,96	53,97	*	59,17	47,00	63,47	*
NOR	56,37	45,37	*	62,37	*	61,92	15,01	53,36	48,91	60,10	42,99	62,34	74,34	55,45	53,30	47,87	54,03	67,83	70,05	50,71	*	73,19	48,54	38,34	36,45		77,79	55,07	65,35	*	*	62,41	53,96	*	49,93	26,37	65,65	*
POL	65,75	60,18	*	49,09	*	59,52	74,25	69,98	75,72	74,19	72,03	48,61	76,73	78,89	59,29	71,95	64,41	60,16	64,73	68,50	*	71,04	68,91	73,88	66,93	77,79		68,73	67,87	*	*	55,78	50,63	*	63,74	71,17	66,18	*
POR	62,89	48,23	*	43,71	*	61,88	55,04	51,22	68,03	69,92	34,98	47,06	63,46	67,97	36,55	29,21	39,85	58,00	62,96	68,63	*	74,49	50,23	53,41	62,87	55,07	68,73		54,72	*	*	52,54	50,62	*	19,77	53,75	68,60	*
RUM	61,63	57,93	*	56,10	*	69,91	64,39	56,17	64,16	60,70	40,53	53,25	68,95	70,38	32,14	45,26	34,47	57,66	58,95	68,48	*	65,97	66,73	60,95	66,93	65,35	67,87	54,72		*	*	66,19	52,97	*	50,55	59,12	59,39	*
RUŠ	*	*	61,31	*	24,28	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	29,77	*	*	*	*	*	*	*	*		*	*	*	32,63	*	*	*	27,14
ŘEČ	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*		*	*	*	*	*	*	*
SK	59,87	54,67	*	35,49	*	21,72	58,94	64,35	58,58	63,83	61,93	34,47	74,86	61,16	54,54	59,71	63,81	54,23	56,28	57,99	*	77,15	67,17	69,87	61,96	62,41	55,78	52,54	66,19	*	*		32,84	*	48,00	57,24	68,94	*
SLO	54,89	52,75	*	18,95	*	40,02	51,22	44,28	60,42	58,59	54,43	16,54	75,56	65,62	37,23	50,44	55,31	47,32	49,86	53,80	*	59,34	61,08	65,17	53,97	53,96	50,63	50,62	52,97	*	*	32,84		*	44,54	53,58	55,39	*
SRB	*	*	71,99	*	27,29	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	20,43	*	*	*	*	*	*	*	*	32,63	*	*	*		*	*	*	46,34
ŠPA	59,66	39,67	*	47,33	*	51,74	49,71	50,47	62,81	61,32	32,11	46,95	56,79	61,63	30,59	26,36	41,52	57,42	63,66	58,18	*	73,23	42,44	51,05	59,17	49,93	63,74	19,77	50,55	*	*	48,00	44,54	*		47,00	58,86	*
ŠVÉ	53,03	42,79	*	54,71	*	59,19	27,77	49,45	43,07	49,41	50,75	54,57	60,31	45,38	47,94	46,50	55,51	63,72	62,14	46,19	*	63,24	45,22	47,09	47,00	26,37	71,17	53,75	59,12	*	*	57,24	53,58	*	47,00		53,84	*
TUR	64,06	63,96	*	57,83	*	71,28	64,50	55,13	60,27	59,36	63,84	56,56	71,44	63,35	54,61	57,87	67,58	67,36	63,80	62,91	*	65,92	61,71	57,40	63,47	65,65	66,18	68,60	59,39	*	*	68,94	55,39	*	58,86	53,84		*
UKR	*	*	49,69	*	38,25	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	45,18	*	*	*	*	*	*	*	*	27,14	*	*	*	46,34	*	*	*	

Obr. 10 – Výsledná tabulka statistických příbuzností

U bosenštiny je velmi zajímavý její hodně malý rozdíl oproti chorvatštině, pouze 6,58 %. To má své jasné opodstatnění – bosenština, chorvatština i srbština jsou v podstatě varianty srbochorvatštiny, kterou se mluvilo v bývalé Jugoslávii. Po rozpadu tohoto celku se státy snažily o své národní jazyky, které jsou ale téměř totožné (ovšem srbština se zapisuje hlavně cyrilicí, používá se však i latinka). Hlavním příbuzným bulharštiny je makedonský jazyk, čeština se zase blíží slovenštině, dánština norštině, estonština finštině, faerština islandštině apod.

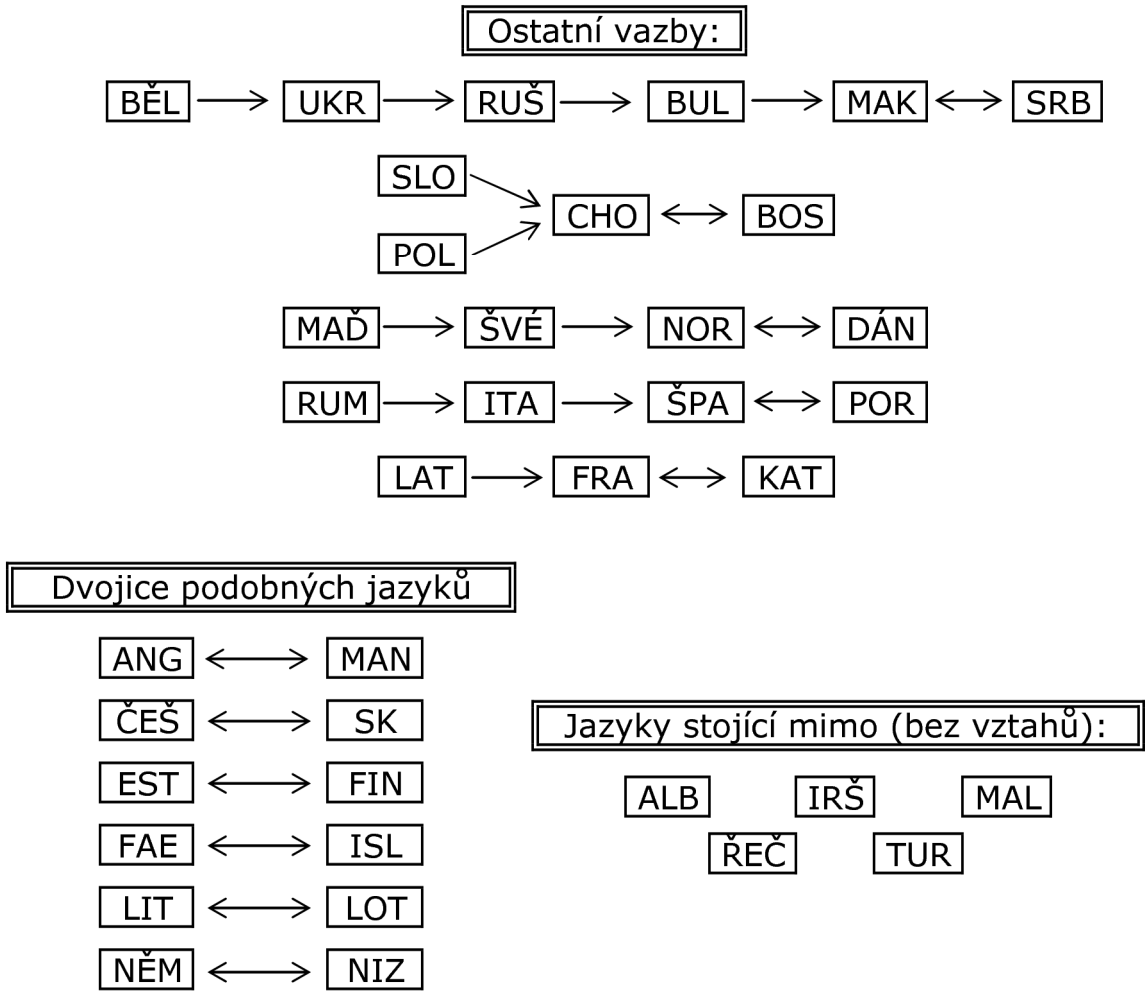
Takto mohu prezentovat celou tabulku, názornější je však zanesení výsledků do politické mapy Evropy a přiložení mapy skutečných vztahů mezi evropskými jazyky. To je graficky znázorněno na Obr. 11, nevybarvené státy (tj. bílé) jsou ty, jejichž hlavní úřední jazyky jsem nenašel v elektronické podobě. Pro komparaci je na následující straně uvedeno obdobné schéma (Breton 2007, s. 14), avšak obsahuje právě reálné příbuznosti (Obr. 12).



Obr. 11 – Statistická příbuznost evropských jazyků

Obr. 12 – Skutečné příbuznosti evropských jazyků (Breton 2007, s. 14)

Snažil jsem se v rámci možností dodržet obdobné barvy, aby vynikl zajímavý výsledek. Moje mapa vznikla tak, že jsem si z tabulky (Obr. 10) vypsal všechny jazyky a k nim první nejpříbuznější jazyk (nepřesahující rozdílnost 50 %, zeleně podbarvený). Tento způsob mě přivedl k diagramům, které jsou znázorněné na následujícím obrázku (Obr. 13).



Obr. 13 – Diagramy podobných jazyků

Z nich jsou zjištěné vazby dobře patrné. Vznikly tři skupiny – jazyky, jejichž nejbližší příbuzný přesahoval 50% hranici rozdílnosti (a ty jsou tedy brány jako samostatné); dvojice jazyků, jenž jsou si vzájemně „nejpříbuznější“; a pět grafů zahrnující ostatní vazby jazyků. Šipka „⟶“ označuje příbuzenský vztah ve smyslu nejpříbuznější pouze s nejbližším sousedem ve směru hrotu.

První graf (BĚL → … ↔ SRB) je tvořen slovanskými jazyky, které v psané formě využívají azbuku – už tím vzniká jistá vazba (protože analýza vychází z jednotlivých znaků), neboť jsou vždy porovnávány jazyky pouze se stejnou abecedou. Proto jsou zahrnuty v mapě pod jednu barvu, přestože jsem po podrobnějším přezkoumání zjistil, že např. běloruština má sice nejblíže k ukrajinštině (ovšem těsně, cca 49,69 %), ale rozdílnost oproti dalším jazykům v tomto diagramu přesahuje hranici dokonce 60 %. Po mírném přeuspořádání tabulky z Obr. 10 jsem zjistil, že „hlavním“ jazykem této skupiny je bulharština.

Dominantním jazykem druhého grafu (SLO → … ↔ BOS) je očividně chorvatština, která je prvním nejpodobnějším jazykem ostatních ve skupině. Odpovídá to i centrálnímu postavení tohoto jazyka v diagramu.

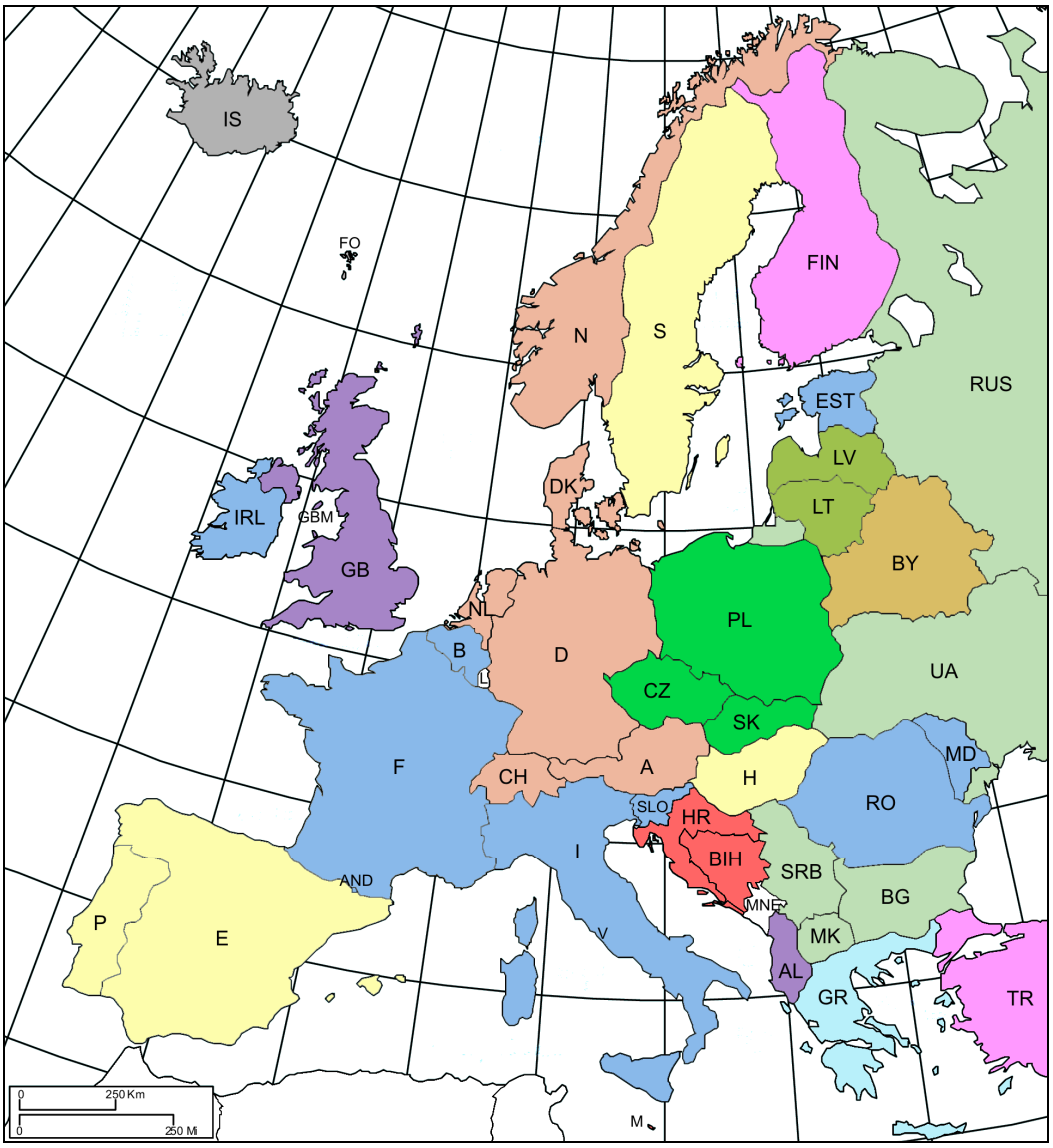
Třetí diagram (MAĎ → … ↔ DÁN) zobrazuje vztah východní skandinávské větve germánských jazyků, ke kterým se zajímavým způsobem připojuje maďarština. Domnívám se, že je již tak vzdálená původnímu ugrofinskému jazyku, že se neprojevila příbuznost např. s finštinou. Hlavní vztažný bod zde tvoří norština.

Poslední dva grafy by mohly být teoreticky propojeny (což by odpovídalo větvi románských jazyků), protože druhým nejpodobnějším jazykem latiny není katalánština, ale italština. Stejně tak je to i s rumunštinou, která má jako druhého nejbližšího souseda opět latinu. Existuje zde tedy jakýsi „most“, přesto jsem tyto skupiny neslučoval a vycházel jsem pouze z nejpříbuznějších vztahů 1 : 1. Hlavními jazyky jsou španělština a francouzština.

Každý z těchto pěti grafů obsahuje „jádro“ tvořené základním jazykem, ke kterému se postupně navazují formou stromu další jazyky a dvojice vzájemně si nejbližších jazyků. Čím více je jazyk v grafu vzdálenější od centra, tím méně je s ním příbuzný. Vycházím tedy ze souvislosti komponent grafů, nikoliv z tranzitivity relace „má nejblíže k“, která dle výsledků v mnoha případech tranzitivní není.

Tímto je v podstatě dána odpověď na otázku „Bude se reálný vývoj a příbuznost jazyků promítat i do příbuznosti statistické, která vychází z frekvenčních analýz těchto jazyků?“ Odpověď zní, že ano. V převážné většině se potvrdilo, že skutečné vztahy jednotlivých jazyků se vyskytují i u frekvenčních analýz těchto jazyků. To vysvětluje i občasnou chybovost rozpoznávání kratších textů pomocí mého programu.

Další porovnání bylo prováděno na základě deseti nejčetnějších znaků pomocí excelovského souboru komparace.xls (list 10 nejčetnějších). Výsledek je opět zaznamenán do mapy Evropy (Obr. 14), v potaz byly brány pouze takové příbuznosti, jejichž rozdíly spočívaly v max. 3 lišících se znacích. Porovnávání bylo provedeno obdobně jako v předchozím případě.



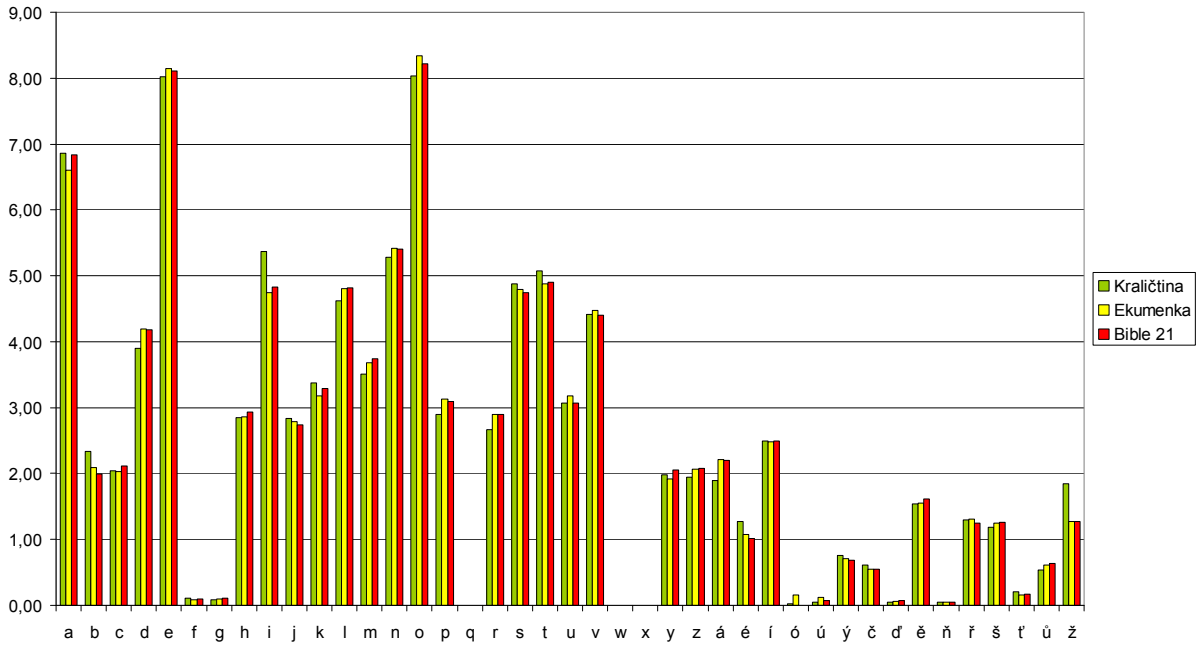
Obr. 14 – Statistická příbuznost dle deseti nejčetnějších písmen

Jak je vidět, výsledky se již trochu liší. To je způsobeno právě „ořezáním“ analýz o specifické znaky.

Poslední úkol byl zaměřen čistě na český jazyk. Spočíval v tom, že jsem měl porovnat frekvenční analýzy současné češtiny, češtiny Bible kralické (16. století) a češtiny Ekumenického překladu Bible (20. století). K analyzování současné češtiny jsem použil jak jinak než Bibli – nejmodernější překlad, který je dostupný – tedy Bibli 21 (http://www.bible21.cz/). Relativní četnosti výskytů jednotlivých písmen abecedy jsou uvedeny na Obr. 15, tyto hodnoty jsou pak zaneseny pro přehlednost do grafu (Obr. 16).

Písmeno	Relativní četnosti v [%]			Písmeno	Relativní četnosti v [%]		
	Kraličtina	**Ekumenka**	**Bible 21**		**Kraličtina**	**Ekumenka**	**Bible 21**
a	6,86	6,60	6,84	v	4,42	4,48	4,41
b	2,34	2,09	2,00	w	0,00	0,00	0,00
c	2,04	2,02	2,12	x	0,00	0,00	0,00
d	3,90	4,20	4,18	y	1,98	1,92	2,05
e	8,02	8,15	8,11	z	1,95	2,06	2,08
f	0,11	0,08	0,10	á	1,90	2,22	2,20
g	0,09	0,10	0,10	é	1,27	1,07	1,01
h	2,85	2,86	2,93	í	2,50	2,49	2,49
i	5,37	4,75	4,83	ó	0,02	0,16	0,00
j	2,84	2,79	2,74	ú	0,05	0,12	0,08
k	3,37	3,18	3,29	ý	0,76	0,71	0,68
l	4,63	4,81	4,82	č	0,61	0,55	0,54
m	3,51	3,68	3,74	ď	0,05	0,06	0,07
n	5,28	5,42	5,40	ě	1,55	1,56	1,62
o	8,04	8,33	8,22	ň	0,05	0,05	0,05
p	2,90	3,13	3,10	ř	1,30	1,31	1,25
q	0,00	0,00	0,00	š	1,18	1,24	1,26
r	2,67	2,90	2,89	ť	0,20	0,16	0,17
s	4,87	4,79	4,74	ů	0,53	0,61	0,63
t	5,07	4,88	4,91	ž	1,84	1,28	1,27
u	3,07	3,18	3,07				

Obr. 15 – Relativní frekvence výskytu znaků češtiny



Obr. 16 – Histogram frekvence výskytu znaků češtiny

Z Obr. 16 je zřejmé, že analýzy jednotlivých podob češtiny jsou téměř shodné, žádný extrémní rozdíl se zde nevyskytuje. Jedinou výjimkou jsou v případě kraličtiny písmena „i“ a „ž“, která se od dnešní češtiny odlišují lehce zvýšeným výskytem. To je pravděpodobně způsobeno častým výskytem slov jako např. kteréž, též, němž, protož; přechodníky apod. Vývoj jazyka za posledních pět století tedy nijak závratně neovlivnil frekvenční analýzu češtiny.

Závěr

Ve své bakalářské práci se mi podařilo ukázat, že lze rozpoznávat text na základě frekvenční analýzy. Přes počáteční překážky (nižší znalost objektově orientovaného programování, výběr vhodného vývojového prostředí či nedostatek literatury) se mi podařilo vytvořit tři aplikace – program Frekvenční analýza 1.0, který rozpoznává jazyk otevřeného textu; program Frekvenční analýza 2.0, který se zaměřuje na odhad jazyka kryptogramu; a pseudoprogram sestavený v MS Excel, který se snaží určit jazyk otevřeného textu na základě deseti nejčetnějších písmen. Celkově práce přináší nový způsob využití frekvenční analýzy, jenž jsem nikde jinde takto zpracovaný nenašel. Navíc Frekvenční analýza 2.0 svou funkcí ověřila, že jednoduchá substituční šifra založená na prostém posunu abecedy neovlivní frekvenční analýzu otevřeného textu, a proto lze určit jazyk kryptogramu.

Díky mnou navrženým programům jsem získal data, která mi umožnila provést menší průzkum provázanosti statistických vlastností jazyka s vlastnostmi reálnými. Zaměřil jsem se na 38 úředních evropských jazyků, neboť větší oblast by jistě byla mimo rozsah této práce. Po zpracování výsledků se mi dostalo odpovědi na otázku zmíněnou v podkapitole 4.2. Ve většině případů se potvrdilo, že se reálné příbuznosti jednotlivých jazyků promítají do frekvenčních analýz těchto jazyků. To vysvětluje i občasnou chybovost rozpoznávání kratších textů pomocí mého programu. Výstupy jsou zpracovány do politických map Evropy, aby byla lépe vidět i geografická provázanost jazyků, způsobená šířením a vzájemným ovlivňováním rozličných evropských kultur a stěhování národů. Dále se mi podařilo vytvořit histogramy analyzovaných jazyků, proto Příloha III. (s. L-X) může posloužit pro další zkoumání evropských jazyků jak z hlediska kryptologického, tak z hlediska lingvistického.

Přínosem práce je také její interdisciplinárnost. Shrnuje dějiny matematiky týkající se dané problematiky, zabývá se kryptoanalýzou a obsahuje část informatiky, která je zde s matematikou úzce spjata v podobě algoritmů programu. Objevuje se zde i statistika, která je využita pro komparaci příbuzností jazyků, dále jsou v práci prvky lingvistiky a geografie.

Případné pokračování této práce by si vzalo za cíl zdokonalit rozpoznávací algoritmy, které by mohly být založeny na frekvenčních analýzách bigramů, případně trigramů. Tím by se jistě docílilo větší efektivity a funkčnosti. Samozřejmě by tato modifikace přinesla další neméně zajímavé výsledky, které by teoreticky mohly poukázat na jinak nevídané podobnosti.

Literatura

ALLEN, Sam. *Dot Net Perls* [online]. c2010a, updated November 27, 2009 [cit. 2010-03-29]. C# Count Letter Frequencies. Dostupné z WWW: <http://dotnetperls.com/count-letter-frequencies>.

BRETON, Roland. *Atlas jazyků světa*. 1. vydání. Praha : Albatros, 2007. 79 s. ISBN 978-80-00-01883-6.

COMRIE, Bernard; MATTHEWS, Stephen; POLINSKY, Maria. *Atlas jazyků : Vznik a vývoj jazyků napříč celým světem*. 1. vydání. Praha : Metafora, 2007. 224 s. ISBN 978-80-7359-129-8.

ELLER, Frank. *C# - začínáme programovat : Podrobný průvodce začínajícího uživatele*. 1. vydání. Praha : Grada Publishing a.s., 2002. 240 s. ISBN 80-247-0324-6.

GROŠEK, Otokar ; PORUBSKÝ, Štefan. *Šifrovanie : algoritmy, metódy, prax*. Praha : Grada a.s., 1992. 272 s. ISBN 80-85424-62-2.

PETZOLD, Charles. *Programování Microsoft Windows v jazyce C#*. 1. svazek. Praha : SoftPress s.r.o., 2003. 600 s. ISBN 80-86497-54-2.

PIPER, Fred, MURPHY, Sean. *Kryptografie : Průvodce pro každého*. 1. vyd. Praha : Dokořán, 2006. 157 s. ISBN 80-7363-074-5.

PIRKL, Josef. *Řešené příklady v C# : aneb C# skutečně prakticky*. 1. vydání. České Budějovice : KOPP, 2005. 300 s. ISBN 80-7232-265-6.

POE, Edgar Allan. *The gold-bug and other tales*. New York : Dover Publications, Inc., 1991. 121 s. ISBN 0-486-26875-6.

PRICE, Glanville, et al. *Encyklopedie jazyků Evropy*. 1. vydání. Praha : Volvox Globator, 2002. 509 s. ISBN 80-7207-450-4.

SINGH, Simon. *Kniha kódů a šifer : Tajná komunikace od starého Egypta po kvantovou kryptografii*. 1. vydání. Praha : Dokořán, 2003. 382 s. ISBN 80-86569-18-7 (Dokořán), ISBN 80-7203-499-5 (Argo).

VONDRUŠKA, Pavel. *Kryptografie, šifrování a tajná písma*. 1. vydání. Praha : Albatros, 2006. 340 s. ISBN 80-00-01888-8.

Použité zdroje

ALLEN, Sam. *Dot Net Perls* [online]. c2010b, updated June 2, 2010 [cit. 2010-06-15]. C# ROT13. Dostupné z WWW: <http://dotnetperls.com/rot13>.

HARSH2327. *Daniweb : IT Discussion Community* [online]. Aug 14th, 2009 [cit. 2010-06-15]. Please help: Array return index number how to?. Dostupné z WWW: <http://www.daniweb.com/forums/thread211323-2.html#>.

JANOŠÍK, Dušan. *Programujte : zaměřeno na informační technologie* [online]. 03.07.2006 [cit. 2010-06-15]. Jak odstranit diakritiku z řetězce. Dostupné z WWW: <http://programujte.com/?akce=clanek&cl=2006062803-jak-odstranit-diakritiku-z-retezce>. ISSN 1801-1586.

SHUJA, Ali. *CodeGuru Forums* [online]. May 31st, 2006 [cit. 2010-06-15]. The TextBox control does not support shortcut keys. Dostupné z WWW: <http://www.codeguru.com/forum/showthread.php?t=388994>.

Příloha I. – zdrojový kód aplikace FA 1.0

Analyza.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;
using System.IO;
using System.Windows.Forms;

namespace Frekvencni_analyza_1._0
{
    class Analyza
    {
        public static double[] AnalyzaTextu(string text) //metoda vracejici
ze stringu pole s % zastoupenim jednotlivych znaku
        {
            double[] AnalyzaPole = new double[(int)char.MaxValue];
//vytvori pole pokryvajici velikostne cely typ char
            int PocetPismen = 0;
            foreach (char t in text) //prochazi textem a pro kazdou
ordinalni hodnotu znaku na vstupu plni pole
            {
                AnalyzaPole[(int)t]++;
                if (char.IsLetter(t)) { PocetPismen++; } //kdyz je znak na
vstupu pismeno, zvysi PocetPismen o jednicku
            }
            for (int i = 0; i < AnalyzaPole.Length; i++) //naplnene pole
jeste upravime tak, ze z absolutnich cetnosti udelame relativni v %
            {
                AnalyzaPole[i] = (AnalyzaPole[i] / PocetPismen) * 100;
            }
            return AnalyzaPole;
        }
    }
}
```

Deserializace.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;
using System.Collections;

namespace Frekvencni_analyza_1._0
{
    class Deserializace
    {
        public static ArrayList DeserializujArrayList(string filename)
//Deserializace souboru - na vstupu jeho nazev, jako vystup dava opet
ArrayList
        {
            ArrayList arr;
            FileStream fs = new FileStream(filename, FileMode.Open,
FileAccess.Read);
            BinaryFormatter sf = new BinaryFormatter();
```

```
            try
            {
                arr = (ArrayList)sf.Deserialize(fs);
            }
            finally
            {
                fs.Close();
            }
            return arr;
        }
    }
}
```

formAbout.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Frekvencni_analyza_1._0
{
    public partial class formAbout : Form
    {
        public formAbout()
        {
            InitializeComponent();
        }

        private void btnZavritFormular_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

formDatabaze.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Collections;

namespace Frekvencni_analyza_1._0
{
    public partial class formDatabaze : Form
    {
        private bool keyHandled;
        private string soubor = "";
        public formDatabaze()
        {
            InitializeComponent();
```

```
        }

        private void textBoxVlozenyText_KeyDown(object sender, KeyEventArgs
e) //metoda zajistujici funkci zkratky Ctrl + A v objektu
textBoxVlozenyText
        {
            if (e.Control && e.KeyCode == Keys.A)
            {
                textBoxVlozenyText.SelectAll();
                keyHandled = true;
            }
        }

        private void textBoxVlozenyText_KeyPress(object sender,
KeyPressEventArgs e)
        {
            if (keyHandled)
            {
                e.Handled = true;
                keyHandled = false;
            }
        }

        private void radioBtnZeSouboru_CheckedChanged(object sender,
EventArgs e)   //osetreni ovladacich prvku
        {
            textBoxVlozenyText.Enabled = false;
            btnOtevrit.Enabled = true;
            btnVytvorZaznam.Enabled = false;
        }

        private void radioBtnPrimeVlozeni_CheckedChanged(object sender,
EventArgs e) //osetreni ovladacich prvku
        {
            textBoxVlozenyText.Enabled = true;
            btnOtevrit.Enabled = false;
            btnVytvorZaznam.Enabled = true;
        }

        private void btnVymazText_Click(object sender, EventArgs e)
//vymaze text v textBoxVlozenyText
        {
            textBoxVlozenyText.Text = "";
        }

        private void btnOtevrit_Click(object sender, EventArgs e) //Po
kliknuti na Otevrit... vyhodi OpenFileDialog
        {
            OpenFileDialog OtevriSoubor = new OpenFileDialog();
            OtevriSoubor.Filter = "Textové soubory (*.txt)|*.txt";
            if (OtevriSoubor.ShowDialog() == DialogResult.OK)
            {
                soubor = OtevriSoubor.FileName;
                btnVytvorZaznam.Enabled = true;
            }
        }

        public string[] VypisJazyku(ArrayList array) //metoda, ktera seradi
seznam jazyku v ArrayListu podle abecedy
        {
            string[] buffer = new string[array.Count];
```

```
            for (int i = 0; i < array.Count; i++)
            {
                buffer[i] = Convert.ToString(array[i]);
            }
            Array.Sort(buffer);
            return buffer;
        }

        private void formDatabaze_Load(object sender, EventArgs e) //pro
nacitani formulare vyplni textbox se seznamem nainstalovanych jazyku
(ulozeno v languages.bin) a vyplni combobox nabidkou nainstalovatelnych
jazyku (list.lng)
        {
            ArrayList SeznamJazyku =
Deserializace.DeserializujArrayList(Application.StartupPath +
@"\data\languages.bin");
            textBoxSeznamJazyku.Lines = VypisJazyku(SeznamJazyku);
            ArrayList seznam =
Deserializace.DeserializujArrayList(Application.StartupPath +
@"\data\list.lng");
            comboBoxVyberJazyku.Items.AddRange(seznam.ToArray());
        }

        private void formDatabaze_Closing(object sender,
FormClosingEventArgs e) //pri zavirani formulare aktualizuje list.lng -
nove nainstalovane jazyky z nej odebere
        {
            ArrayList buffer = new ArrayList();
            buffer.AddRange(comboBoxVyberJazyku.Items);
            Serializace.SerializujArrayList(buffer, Application.StartupPath
+ @"\data\list.lng");
        }

        private void btnVytvorZaznam_Click(object sender, EventArgs e)
        {
            string text = "";
            if (radioBtnZeSouboru.Checked)      //podle radiobuttonu do
promenne text vlozi text na vstupu (ze souboru ci z textBoxu), velka
pismena jsou prevedena na mala
            {
                text = File.ReadAllText(soubor).ToLower();
            }
            else
            {
                text = textBoxVlozenyText.Text.ToLower();
            }
            if (comboBoxVyberJazyku.Text == "")
                MessageBox.Show("Nevybrali jste jazyk textu!!!",
"Varování", MessageBoxButtons.OK, MessageBoxIcon.Warning); //osetreni
comboboxu - zakaz nevybrani jazyku
            else
            {
                StreamReader CountOfFiles = new
StreamReader(Application.StartupPath + @"\data\CountOfFiles.var"); //nacte
z CountOfFiles.var cislo odpovidajici poctu nainstalovanych jazyku
                int PocetSouboru =
Convert.ToInt16(CountOfFiles.ReadLine());
                CountOfFiles.Close();
                double[] NovyJazyk = Analyza.AnalyzaTextu(text); //vytvori
pole s analyzou vstupniho textu
```

```
                string NazevSouboru = PocetSouboru + ".fqa"; //vytvoreni
automatickeho nazvu souboru s nove vytvorenou analyzou
                StreamWriter VytvorJazyk = new
StreamWriter(Application.StartupPath + @"\data\" + NazevSouboru);
                for (int i = 0; i < NovyJazyk.Length; i++)
                {
                    if (char.IsLetter((char)i) && char.IsLower((char)i))
//cyklus, ktery naplni soubor s novou analyzou - vklada pouze frekvence
malych pismen (velka budou mit frekvenci stejne 0, protoze jsme vstupni
text jiz na zacatku prevedli na mala)
                    {
                        VytvorJazyk.WriteLine("{0}", NovyJazyk[i]);
                    }
                }
                StreamWriter ZmenPocetSouboru = new
StreamWriter(Application.StartupPath + @"\data\CountOfFiles.var"); //vznikl
novy jazyk -> musime pocet souboru v CountOfFiles zvednout o jednicku
                ZmenPocetSouboru.WriteLine(PocetSouboru + 1);
                ZmenPocetSouboru.Close();
                VytvorJazyk.Close();
                ArrayList jazyky =
Deserializace.DeserializujArrayList(Application.StartupPath +
@"\data\languages.bin"); //vyvolame seznam nainstalovanych jazyku
                jazyky.Add(comboBoxVyberJazyku.Text);    //vlozi nazev
jazyku z comboboxu
// a zde do nej ulozime novy zaznam
                Serializace.SerializujArrayList(jazyky,
Application.StartupPath + @"\data\languages.bin");              //a opet
serializujeme
                ArrayList SeznamJazyku =
Deserializace.DeserializujArrayList(Application.StartupPath +
@"\data\languages.bin"); //opet deserializujeme a tim ziskame aktualitovany
seznam jazyku
                textBoxSeznamJazyku.Lines = VypisJazyku(SeznamJazyku);
                MessageBox.Show("Jazyk " + comboBoxVyberJazyku.SelectedItem
+ " byl úspěšně vložen do databáze.", "Potvrzení", MessageBoxButtons.OK,
MessageBoxIcon.Information); //potvrzeni uspesneho vlozeni

comboBoxVyberJazyku.Items.RemoveAt(comboBoxVyberJazyku.SelectedIndex);
//vymazani polozky z comboboxu - "ochrana" proti vlozeni dvou shodnych
jazyku
            }
        }
    }
}
```

formHlavni.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Collections;

namespace Frekvencni_analyza_1._0
{
    public partial class formHlavni : Form
```

```
    {
        private bool keyHandled;
        private string soubor = "";
        public formHlavni()
        {
            InitializeComponent();
        }

        private void btnAbout_Click(object sender, EventArgs e) //vyvola
FormAbout
        {
            formAbout formAbout = new formAbout();
            formAbout.ShowDialog();
        }

        private void textBoxVlozenyText_KeyDown(object sender, KeyEventArgs
e) //opet funkce pro ctrl+a
        {
            if (e.Control && e.KeyCode == Keys.A)
            {
                textBoxVlozenyText.SelectAll();
                keyHandled = true;
            }
        }

        private void textBoxVlozenyText_KeyPress(object sender,
KeyPressEventArgs e)
        {
            if (keyHandled)
            {
                e.Handled = true;
                keyHandled = false;
            }
        }

        private void radioBtnZeSouboru_CheckedChanged(object sender,
EventArgs e) //opet osetreni
        {
            textBoxVlozenyText.Enabled = false;
            btnOtevrit.Enabled = true;
            btnAnalyzuj.Enabled = false;
        }

        private void radioBtnPrimeVlozeni_CheckedChanged(object sender,
EventArgs e)
        {
            textBoxVlozenyText.Enabled = true;
            btnOtevrit.Enabled = false;
            btnAnalyzuj.Enabled = true;
        }

        private void btnOtevrit_Click(object sender, EventArgs e)
//opendialog
        {
            OpenFileDialog OtevriSoubor = new OpenFileDialog();
            OtevriSoubor.Filter = "Textové soubory (*.txt)|*.txt";
            if (OtevriSoubor.ShowDialog() == DialogResult.OK)
            {
                soubor = OtevriSoubor.FileName;
                btnAnalyzuj.Enabled = true;
            }
```

```
        }

        private void btnVymazText_Click(object sender, EventArgs e)
//tlacitko pro vymazani textu z textboxu
        {
            textBoxVlozenyText.Text = "";
        }

        private void btnRozsirDatabazi_Click(object sender, EventArgs e)
//otevre form pro vkladani jazyku
        {
            formDatabaze formDatabaze = new formDatabaze();
            formDatabaze.ShowDialog();
        }

        public double[] FiltrujPismena(double[] vstup) //analyzovane pole
je plne i znaku, ktere nejsou pismena, toto je odfiltruje pryc pomoci
temporaryfileu, ale prijdeme jiz o ordinalni hodnoty znaku, v souborech s
analyzami si radky odpovidaji co se tyce pismen
        {
            StreamWriter writer = new StreamWriter(Application.StartupPath
+ @"\data\temp.fa");
            for (int i = 0; i < vstup.Length; i++)
            {
                if (char.IsLetter((char)i) && char.IsLower((char)i))
                {
                    writer.WriteLine(vstup[i]);
                }
            }
            writer.Close();
            StreamReader reader = new StreamReader(Application.StartupPath
+ @"\data\temp.fa");
            double[] vysledne = new double[1102];
            for (int i = 0; i < vysledne.Length; i++)
            {
                vysledne[i] = Convert.ToDouble(reader.ReadLine());
            }
            reader.Close();
            File.Delete(Application.StartupPath + @"\data\temp.fa");
            return vysledne;
        }

        private void btnAnalyzuj_Click(object sender, EventArgs e)
//samotne rozpoznavani vlozeneho jazyka
        {
            string text = "";
            if (radioBtnZeSouboru.Checked)
            {
                text = File.ReadAllText(soubor).ToLower(); //nacte vlozeny
jazyk z urceneho zdroje
            }
            else
            {
                text = textBoxVlozenyText.Text.ToLower();
            }
            grafAnalyzyVstupu.Series[0].Clear(); //vycisti graf od
predchozich hodnot
            double[] AnalyzaVstupu = Analyza.AnalyzaTextu(text); //provede
analyzu
            for (int i = 0; i < AnalyzaVstupu.Length; i++)
            {
```

```
                if (AnalyzaVstupu[i] > 0 && char.IsLetter((char)i)) //pro
kazdy nenulovy vyskyt PISMENA vytvori sloupec grafu
                {
                    grafAnalyzyVstupu.Series[0].Add(AnalyzaVstupu[i],
Convert.ToString((char)i));
                }
            }
            double[] AnalyzaUpravena = FiltrujPismena(AnalyzaVstupu);
//pote jiz muzeme vytratit ordinalni hodnoty znaku, zacneme porovnavat
            StreamReader CountOfFiles = new
StreamReader(Application.StartupPath + @"\data\CountOfFiles.var"); //nacte
pocet souboru
            int PocetSouboru = Convert.ToInt32(CountOfFiles.ReadLine());
            CountOfFiles.Close();
            double[] rozdily = new double[PocetSouboru]; //pole do ktereho
budeme ukladat soucet rozdilu analyz textu vuci nainstalovanym jazykum
            for (int i = 0; i < PocetSouboru; i++) //pro vsechny soubory
obsahujici analyzy provede:
            {
                string soubor = Application.StartupPath + @"\data\" + i +
".fqa";
                StreamReader analyza = new StreamReader(soubor); //otevre
soubor
                double[] pole = new double[AnalyzaUpravena.Length]; //pole
naplnene ze souboru
                double[] vysledek = new double[AnalyzaUpravena.Length];
//pole rozdilu analyzy vlozeneho textu oproti databazi
                double suma = 0;
                for (int j = 0; j < AnalyzaUpravena.Length; j++)
                {
                    pole[j] = Convert.ToDouble(analyza.ReadLine()); //nacte
radek ze souboru
                    vysledek[j] = Math.Abs(AnalyzaUpravena[j] - pole[j]);
//poli s rozdily priradi rozdil techto radku
                    suma = suma + vysledek[j]; //pricte hodnotu radku
                }
                rozdily[i] = suma; //vlozi do pole s rozdily ten soucet
                analyza.Close();
            }
            ArrayList SeznamJazyku =
Deserializace.DeserializujArrayList(Application.StartupPath +
@"\data\languages.bin"); //vyvolame pole nainstalovanych jazyku
            labelVyslednyJazyk.Visible = true;
            labelVyslednyJazyk.Text =
Convert.ToString(SeznamJazyku[Minimum.IndexMinimalniHodnoty(rozdily)]);
//na label hodi urceny jazyk - ten je ziskan tim, ze z pole rozdily ziskame
pozici minimalni hodnoty v tomto poli a te odpovida i jazyk ulozeny pod
timto indexem v arraylistu seznam jazyku
        }
    }
}
```

Minimum.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Frekvencni_analyza_1._0
{
    class Minimum
```

```
    {
        public static int IndexMinimalniHodnoty(double[] hodnoty) //metoda,
ktera ze vstupniho pole vyda index prvku, ktery jest v poli minimem
        {
            double minimum = double.MaxValue;
            for (int i = 0; i < hodnoty.Length; i++) //najde minimum
            {
                if (hodnoty[i] < minimum)
                {
                    minimum = hodnoty[i];
                }
            }
            int MinimumIndex = 0;
            for (int i = 0; i < hodnoty.Length; i++)
            {
                if (minimum == hodnoty[i]) //projde pole a kdyz najde
minimum, vrati index jeho pozice
                {
                    MinimumIndex = i;
                }
            }
            return MinimumIndex;
        }
    }
}
```

Serializace.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;
using System.Collections;

namespace Frekvencni_analyza_1._0
{
    class Serializace
    {
        public static void SerializujArrayList(ArrayList array, string
filename) //metoda pro serializaci ArrayListu - vstupem jest ArrayList a
retezec pro nazev souboru, vystupem je soubor
        {
            FileStream fs = new FileStream(filename, FileMode.Create,
FileAccess.Write);
            BinaryFormatter sf = new BinaryFormatter();
            try
            {
                sf.Serialize(fs, array);
            }
            finally
            {
                fs.Close();
            }
        }
    }
}
```

Příloha II. – Seznam analyzovaných textů

albánština: http://bibledatabase.org/bibles_unicode.html

angličtina (World English Bible): http://bibledatabase.org/bibles.html

běloruština: http://knihi.com/biblija/01-byccio.html

bosenština: http://wwwyu.com/izrada/

bulharština: http://bibledatabase.org/exec/b_bulgarian.exe

čeština: http://online.bible21.cz/; http://www.cb.cz/karlovy.vary/download.htm

dánština:
http://beta.bibleserver.com/text?language=20&PHPSESSID=89bc8a317905189a2ac73fcf9af8e38a

estonština: http://www.jesus.org.uk/bible/Estonian+Genesis+and+NT

faerština: http://www.biblian.fo/default.asp?gerd=visbok&bokid=1&kapitulid=1

finština: http://bibledatabase.org/exec/b_finnish1938.exe

francouzština: http://bibledatabase.org/bibles_french.html

chorvatština: http://bibledatabase.org/bibles_other.html

irština: http://www.biblebc.com/Irish/IrishBible/Luke_in_Irish.htm

islandština: http://www.biblegateway.com/versions/?action=getVersionInfo&vid=18

italština: http://bibledatabase.org/exec/b_italian.exe

katalánština: http://www.ibecat.org/BIBLIA/

latina: http://bibledbdata.org/exec/modules/b_vulgate.exe

litevština: http://unbound.biola.edu/index.cfm?method=downloads.showDownloadMain

lotyština: http://unbound.biola.edu/index.cfm?method=downloads.showDownloadMain

maďarština: http://bibledatabase.org/bibles_unicode.html

makedonština: http://www.mkbible.net/biblija/index.php?Bitie%201

maltština: http://www.tarxienparish.org/Il-bibbja.htm

manština: http://unbound.biola.edu/index.cfm?method=downloads.showDownloadMain

němčina: http://unbound.biola.edu/index.cfm?method=downloads.showDownloadMain

nizozemština: http://bibledatabase.org/bibles_dutch.html

norština: http://www.bibleserver.com/text/Matou%C5%A1#/text/NOR/Matou%C5%A11

polština: http://sites.google.com/site/dfhmch/grabneronlinedownloadlinks

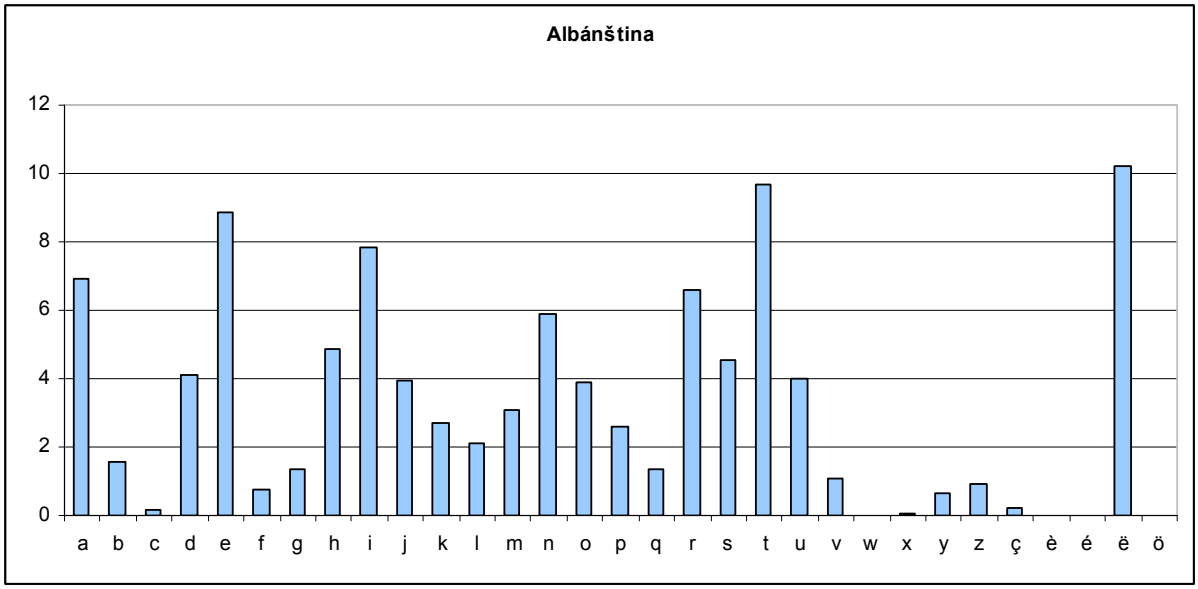
portugalština: http://bibledatabase.org/exec/b_portuguese.exe

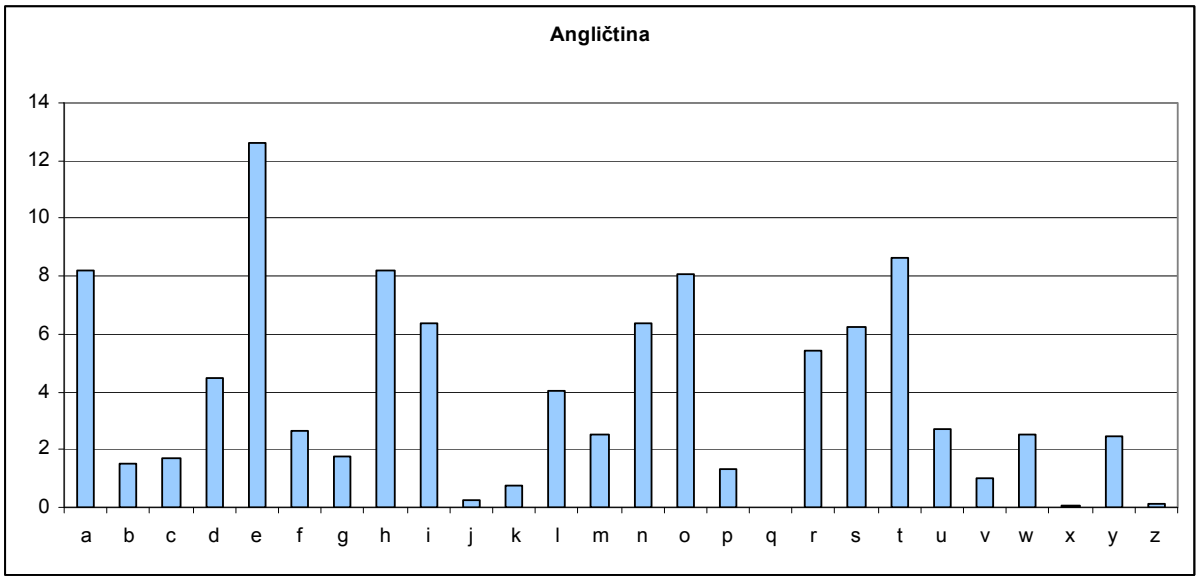
rumunština: http://unbound.biola.edu/index.cfm?method=downloads.showDownloadMain

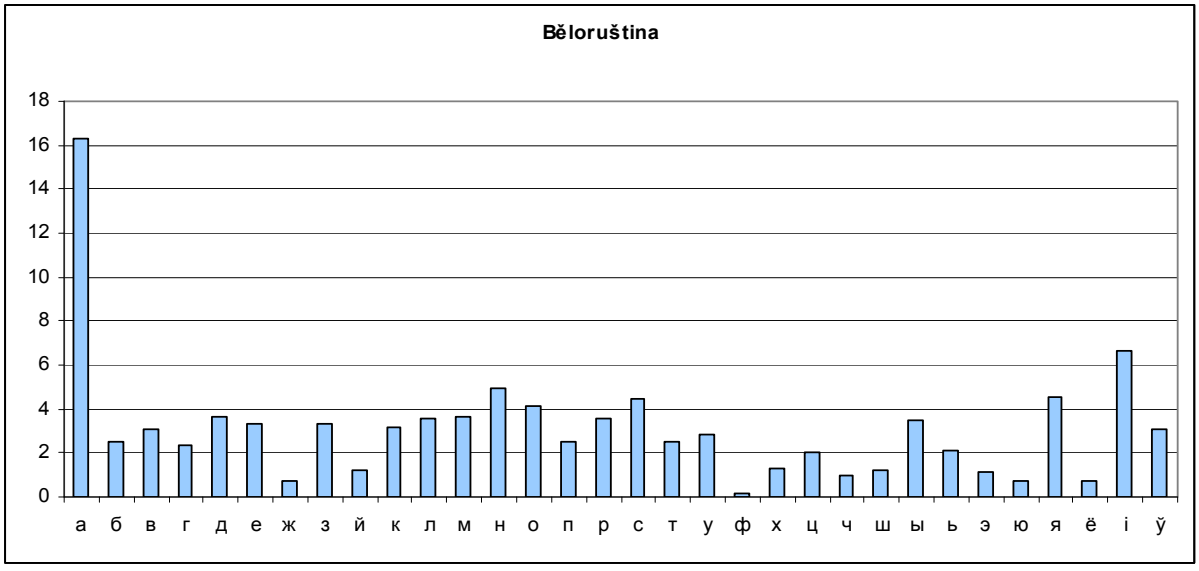
ruština: http://bibledatabase.com/bibles_unicode.html

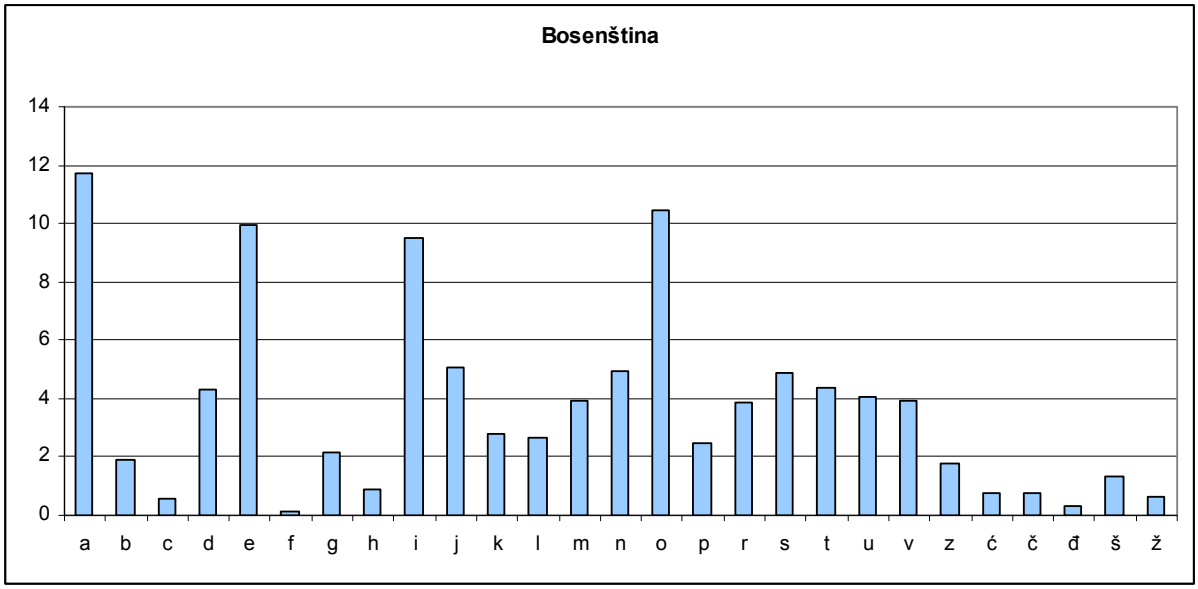
řečtina: http://bibledatabase.com/bibles_unicode.html
slovenština: http://www.biblebasicsonline.com/slovakian/download/zip_slov.exe
slovinština: http://www.biblija.net/biblija.cgi?m=1+Mz+1%2C1-50%2C26&id13=1&pos=0&set=2&l=sl
srbština: http://go-bible.googlegroups.com/web/SerbianDanicicKaradzicCyrillic.zip
španělština: http://bibledbdata.org/exec/modules/b_spanish_mod.exe
švédština: http://www.biblica.com/bibles/swedish/
turečtina: http://unbound.biola.edu/index.cfm?method=downloads.showDownloadMain
ukrajinština: http://bibledatabase.com/bibles_unicode.html

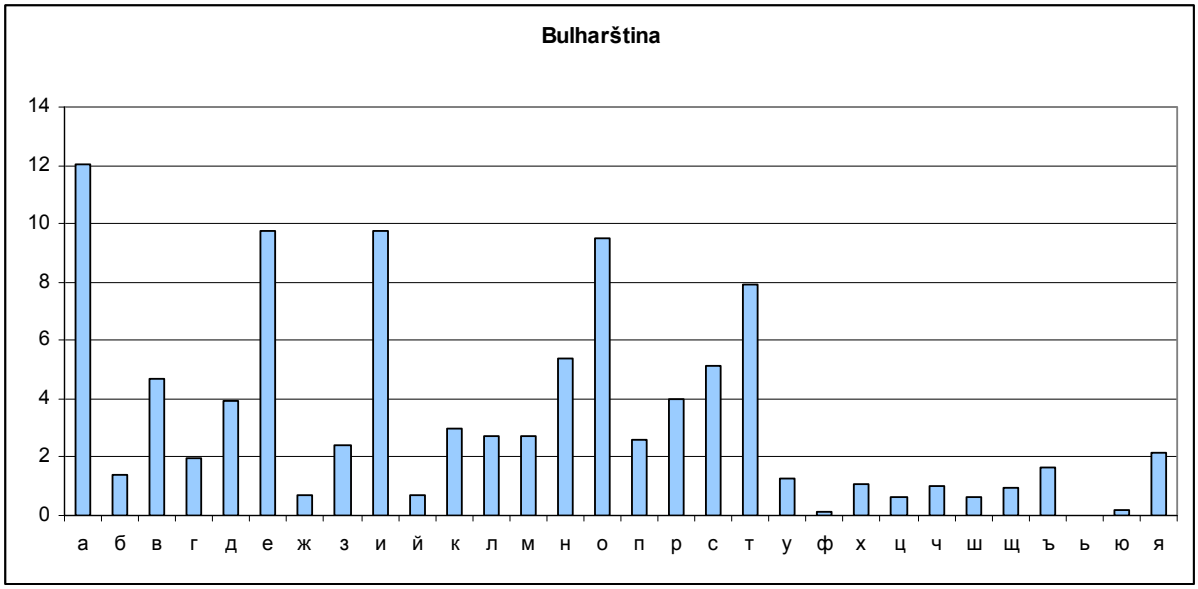
Příloha III. – Grafy jednotlivých frekvenčních analýz

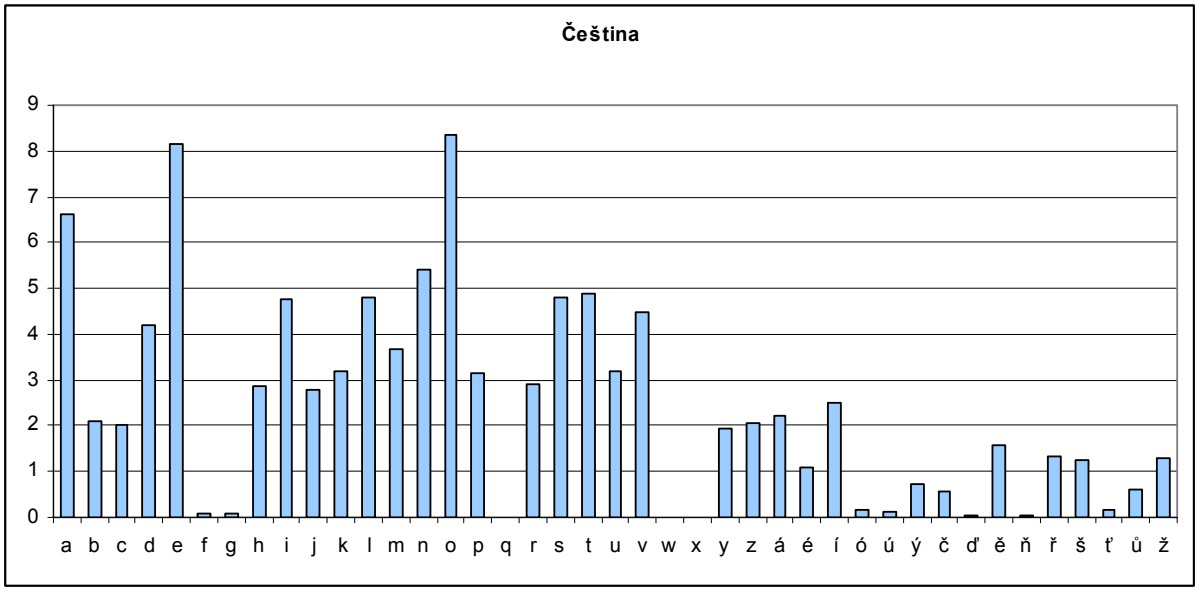


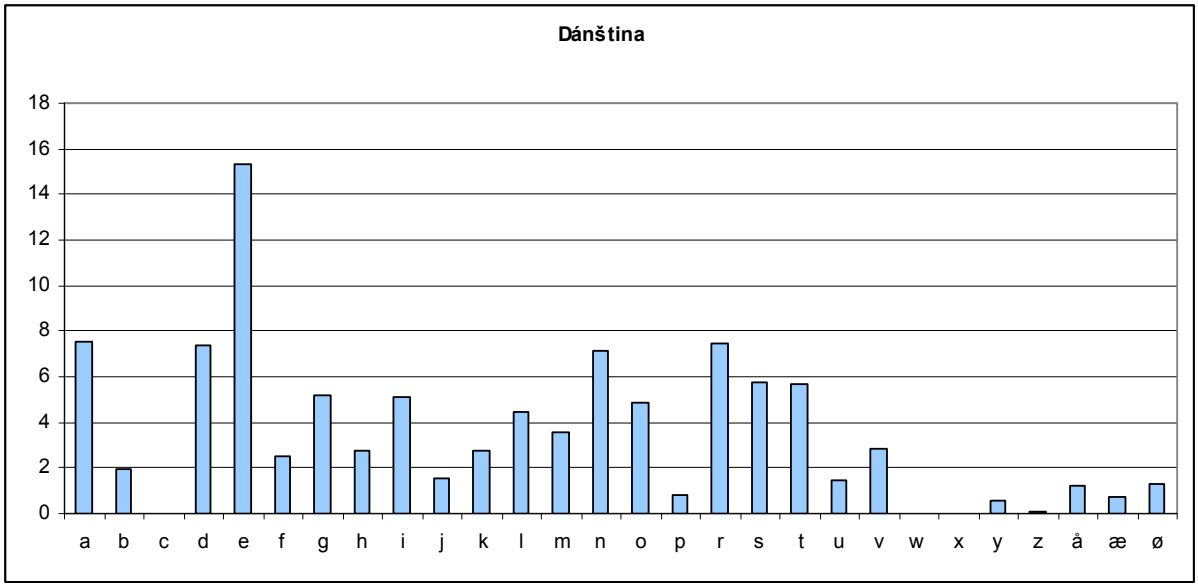


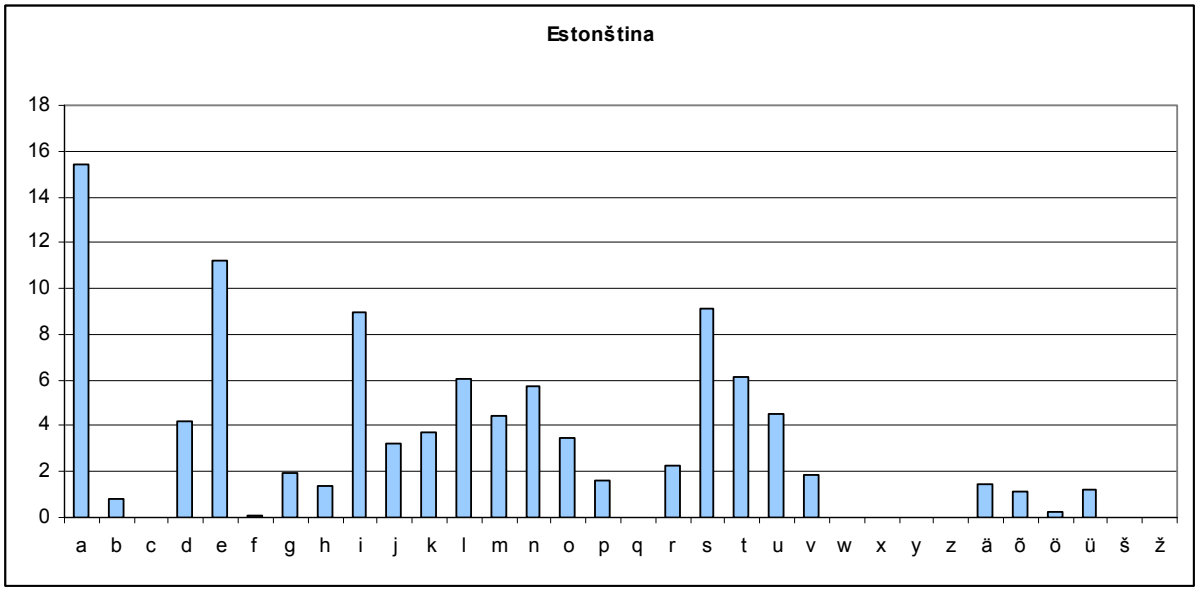


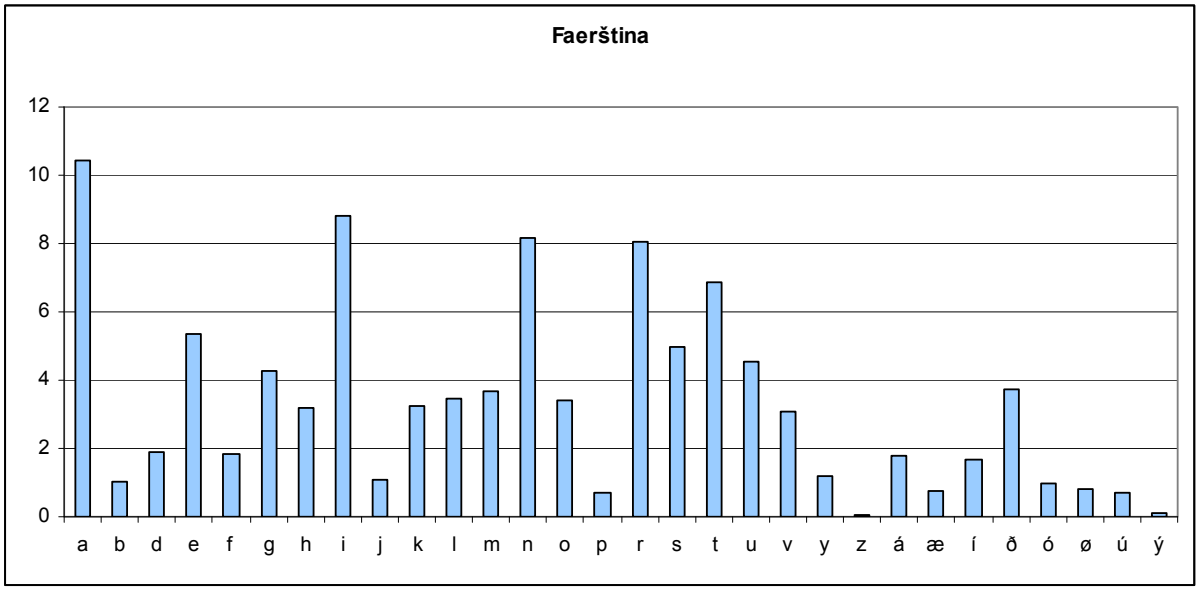

Bosenština
14
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p r s t u v z ć č đ š ž

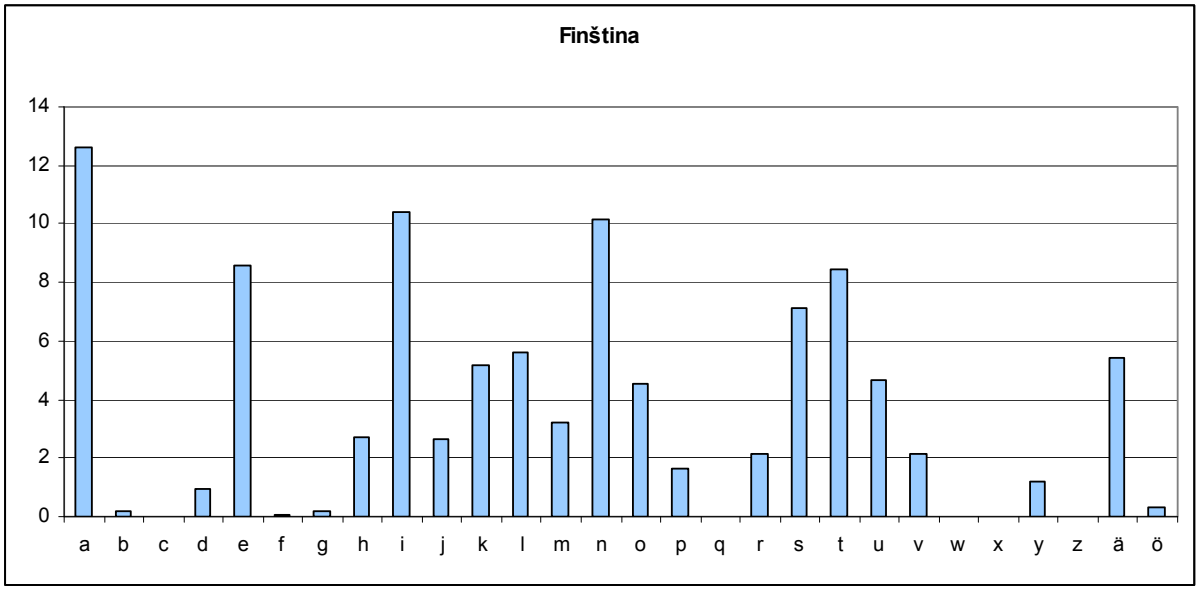

Bulharština
14
12
10
8
6
4
2
0
а б в г д е ж з и й к л м н о п р с т у ф х ц ч ш щ ъ ь ю я

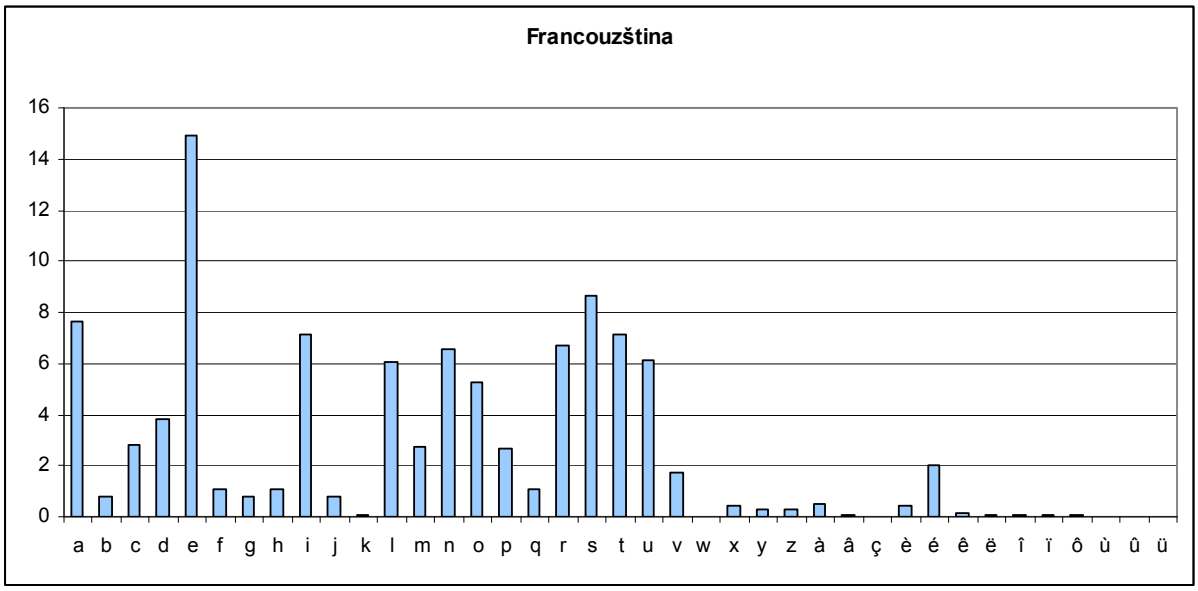

Čeština
9
8
7
6
5
4
3
2
1
0
a b c d e f g h i j k l m n o p q r s t u v w x y z á é í ó ú ý č ď ě ň ř š ť ů ž

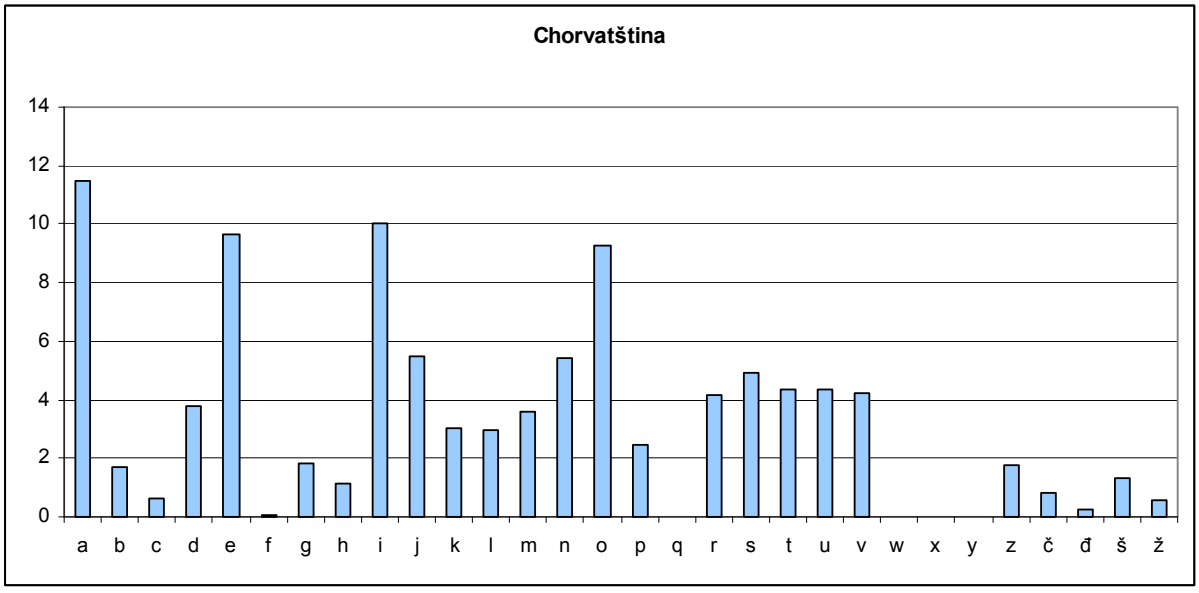

Dánština
18
16
14
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p r s t u v w x y z å æ ø

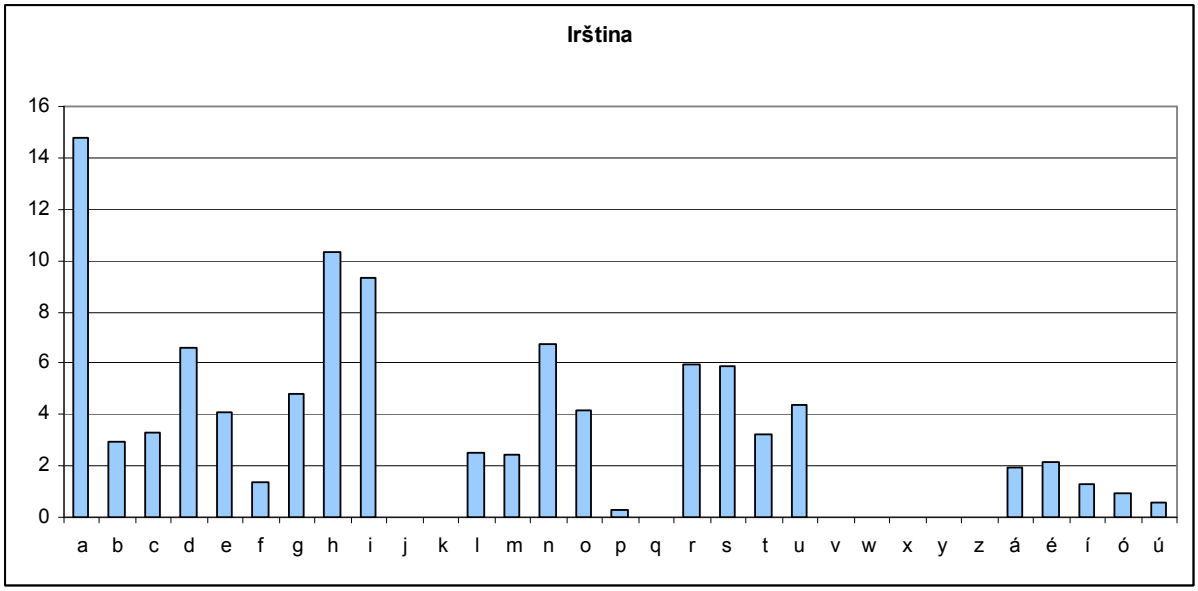

Estonština
18
16
14
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p q r s t u v w x y z ä õ ö ü š ž

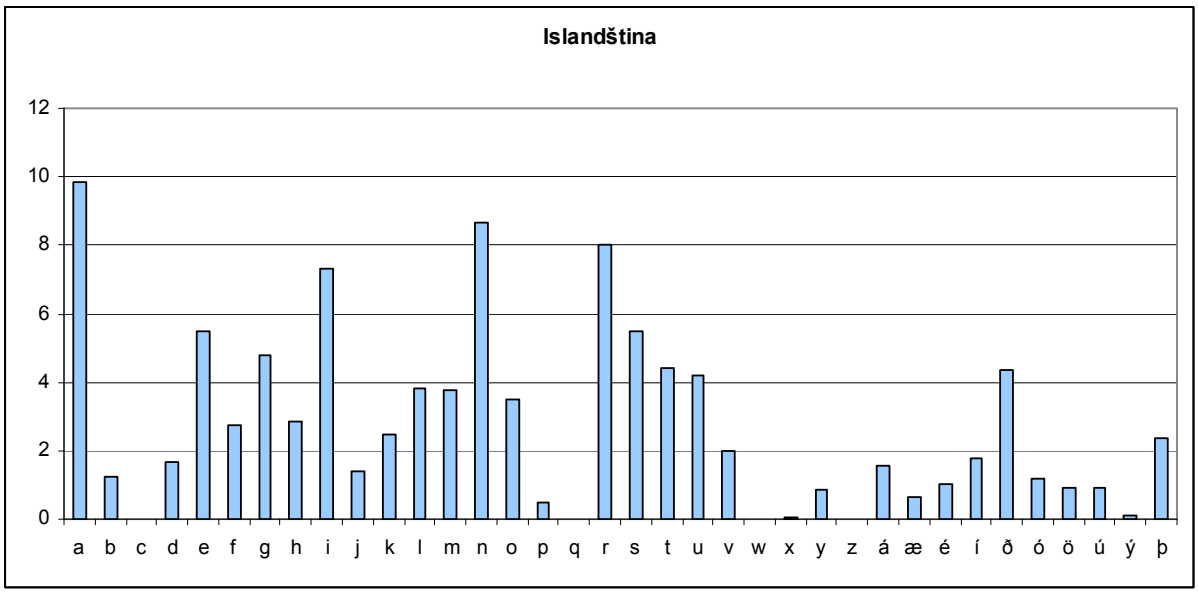

Faerština
12
10
8
6
4
2
0
a b d e f g h i j k l m n o p r s t u v y z á æ í ð ó ø ú ý

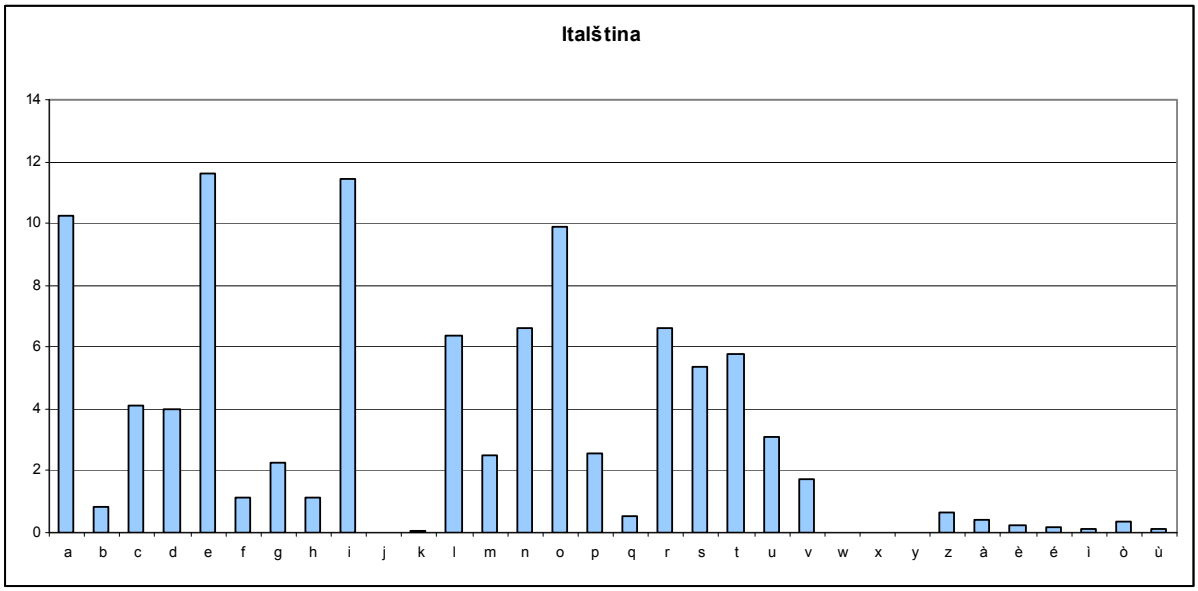

Finština
14
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p q r s t u v w x y z ä ö

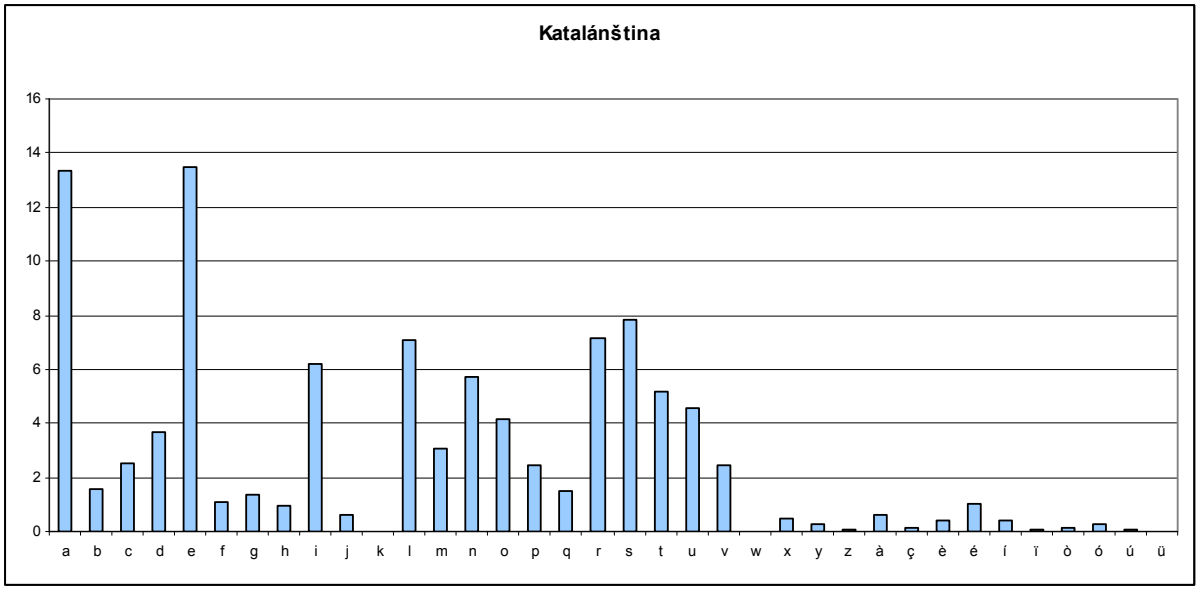

Francouzština
16
14
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p q r s t u v w x y z à â ç è é ê ë î ï ô ù û ü

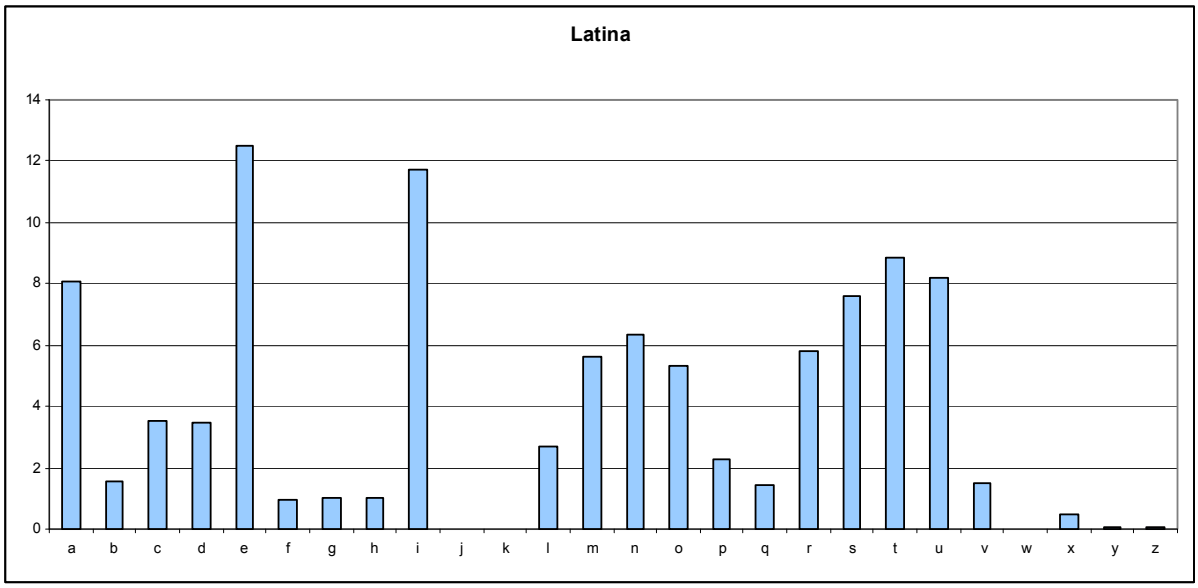

Chorvatština
14
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p q r s t u v w x y z č đ š ž

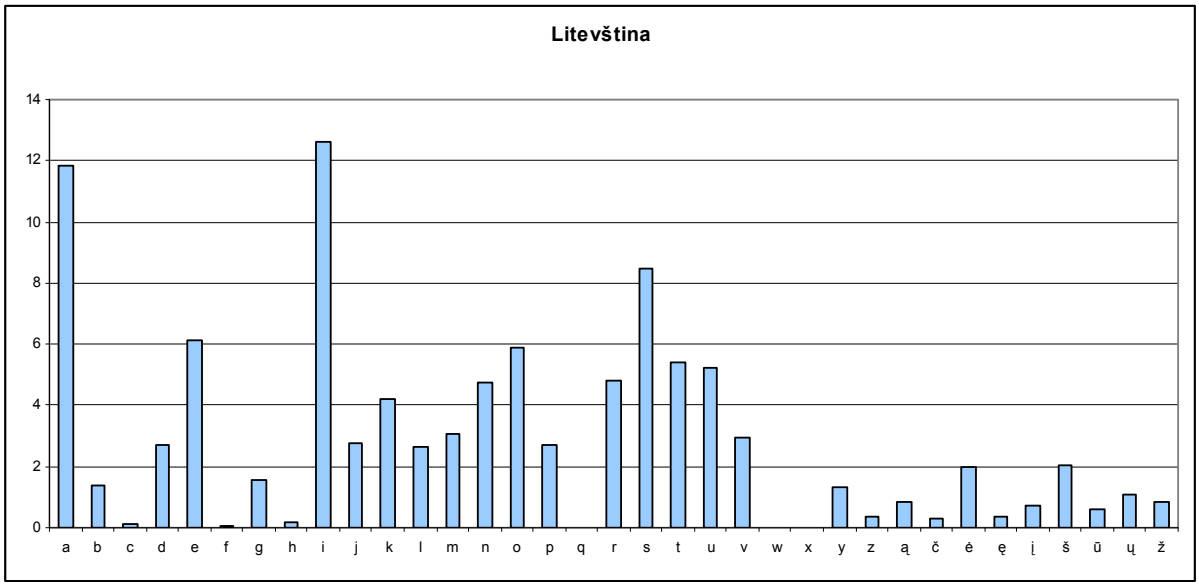

Irština
16
14
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p q r s t u v w x y z á é í ó ú

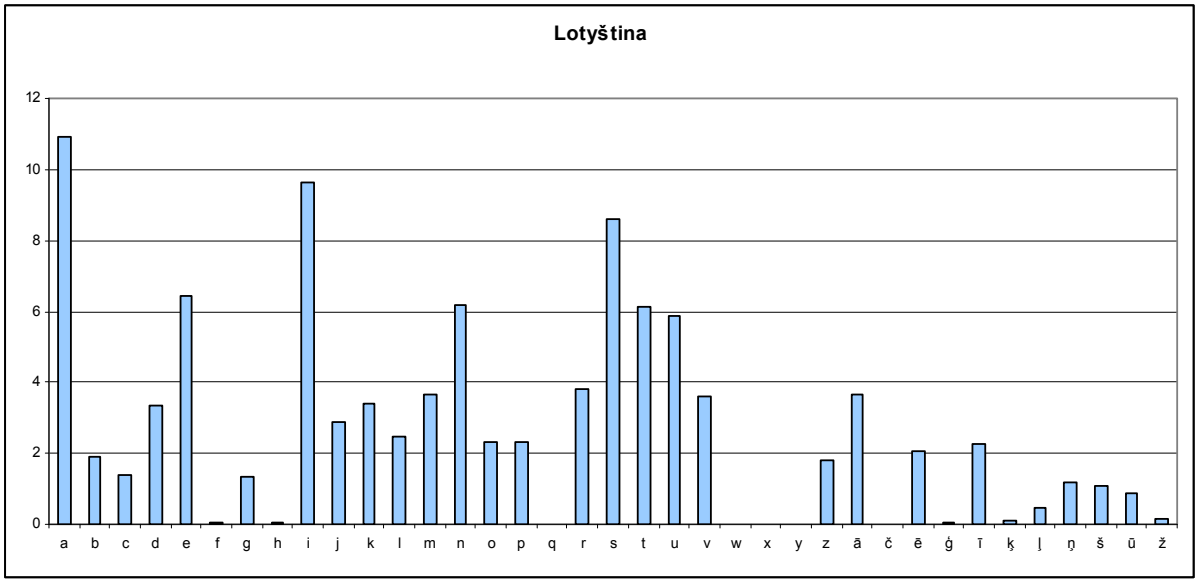

Islandština
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p q r s t u v w x y z á æ é í ð ó ö ú ý þ

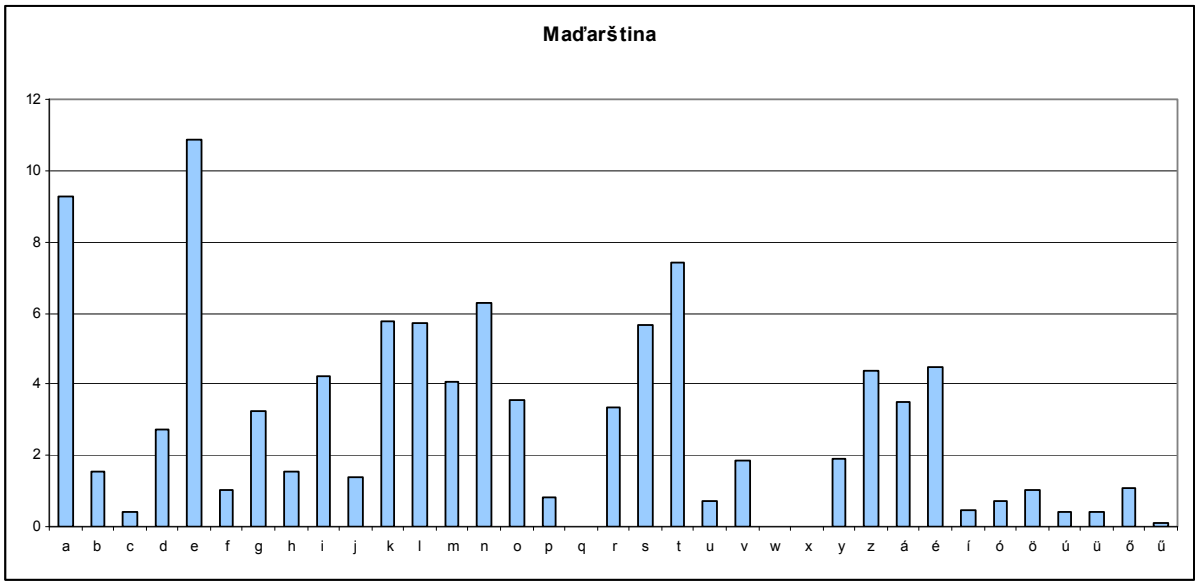

Italština
14
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p q r s t u v w x y z à è é ì ò ù

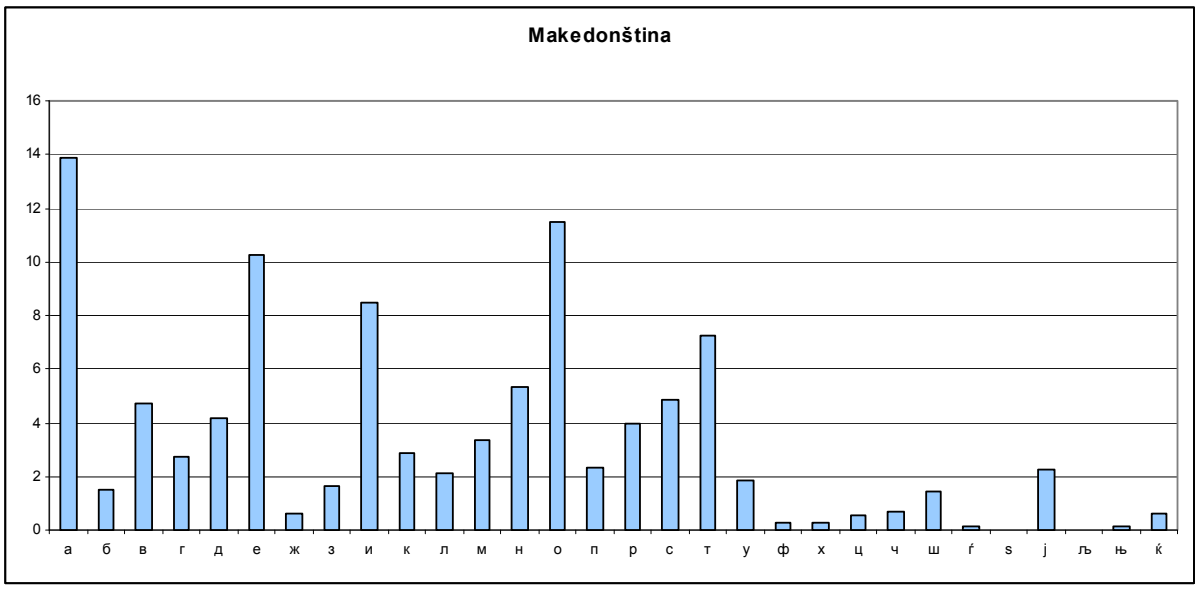

Katalánština
16
14
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p q r s t u v w x y z à ç è é í ï ò ó ú ü

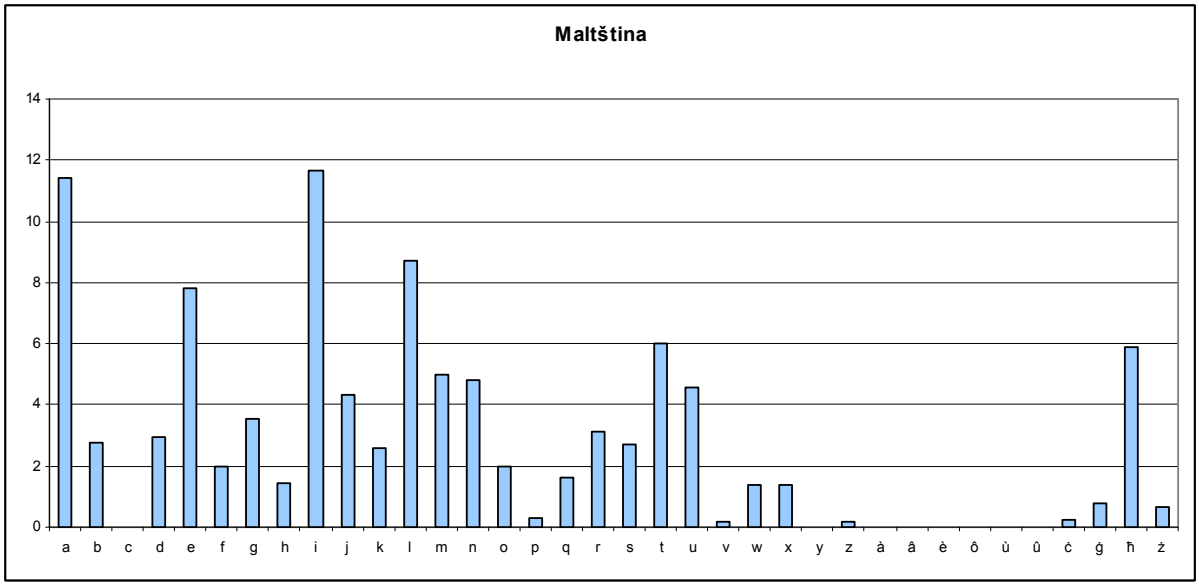

Latina
14
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p q r s t u v w x y z

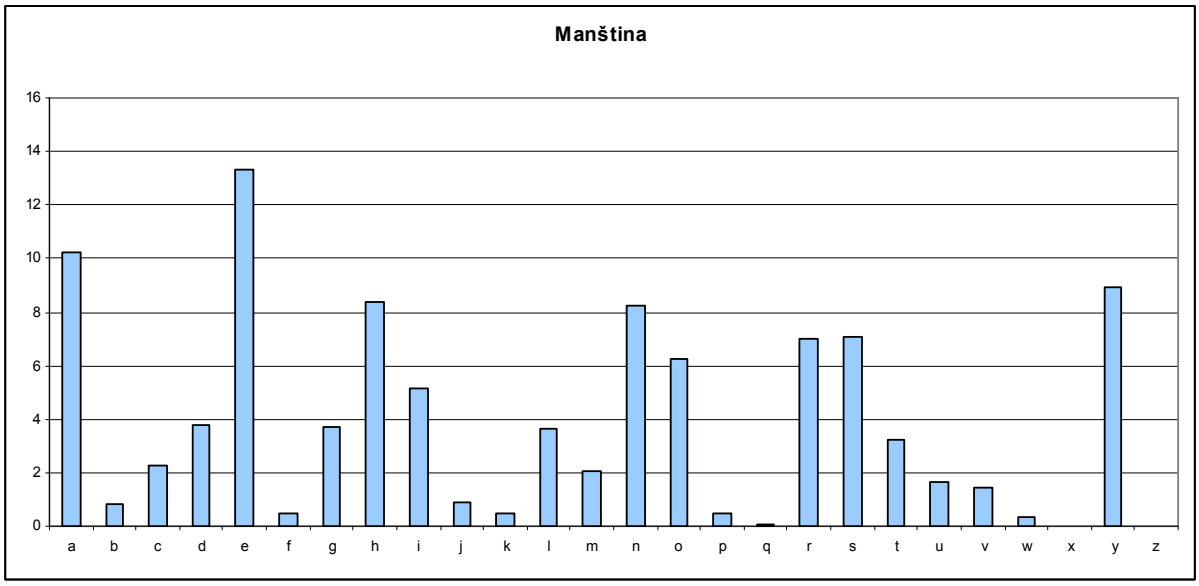

Litevština
14
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p q r s t u v w x y z ą č ė ę į š ū ų ž

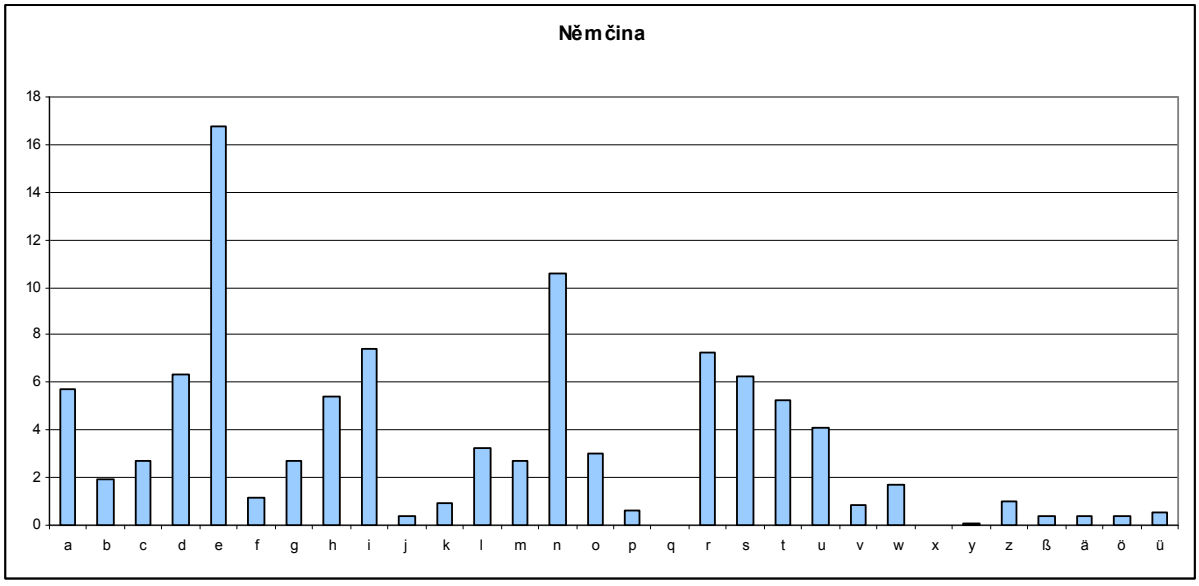

Lotyština
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p q r s t u v w x y z ā č ē ģ ī ķ ļ ņ š ū ž

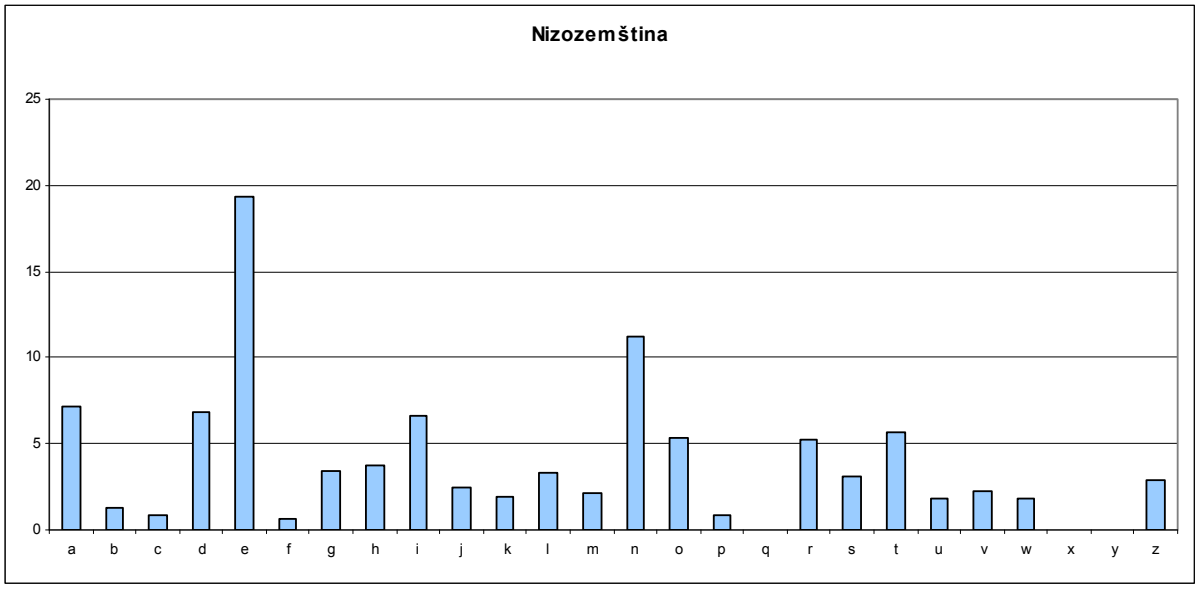

Maďarština
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p q r s t u v w x y z á é í ó ö ú ü ő ű

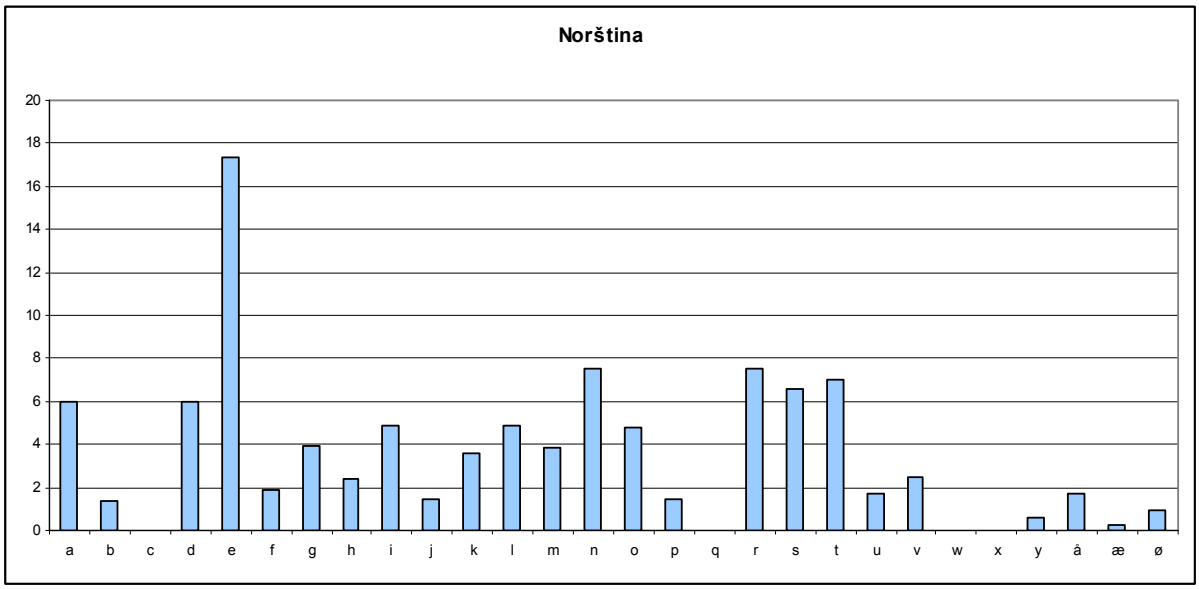

Makedonština
16
14
12
10
8
6
4
2
0
а б в г д е ж з и к л м н о п р с т у ф х ц ч ш ѓ ѕ ј љ њ ќ

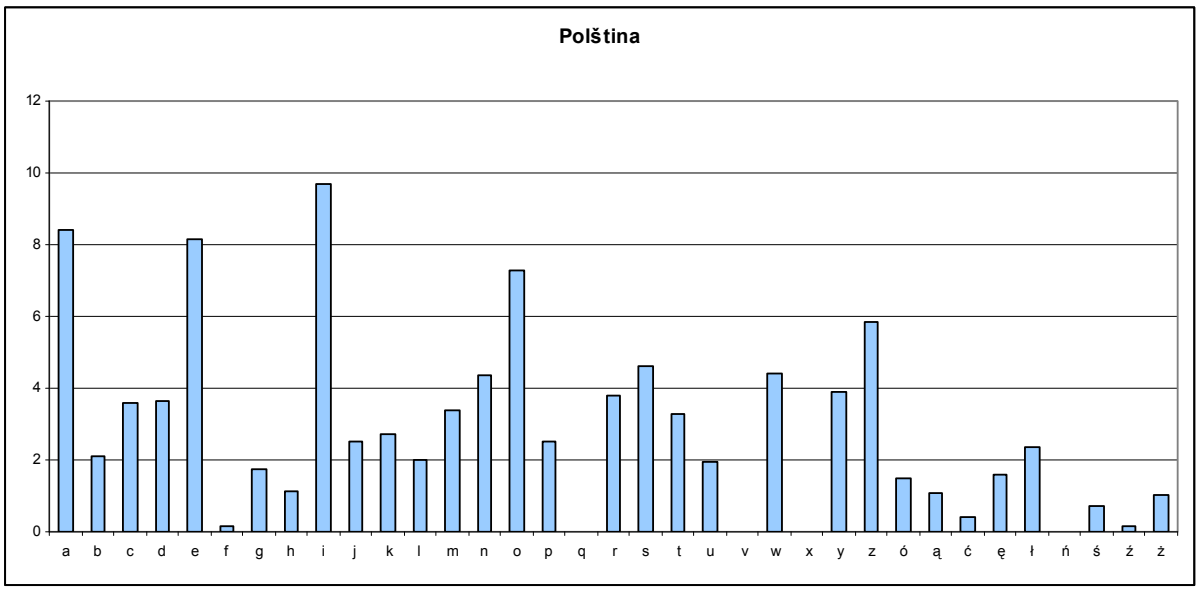

Maltština
14
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p q r s t u v w x y z à â è ô ù û ċ ġ ħ ż

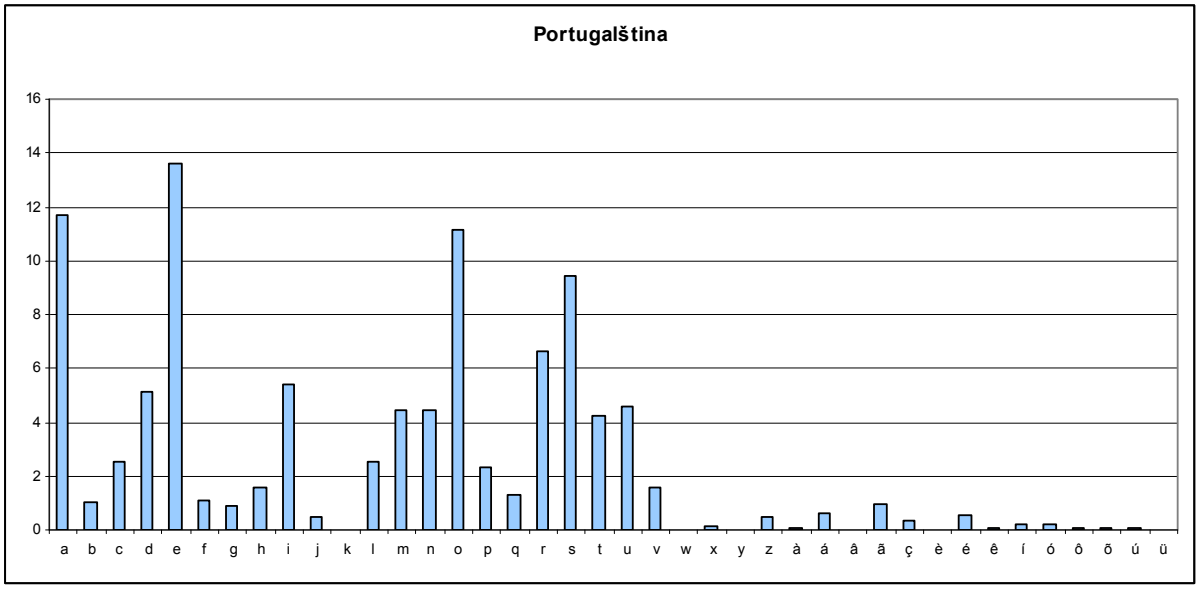

Manština
16
14
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p q r s t u v w x y z

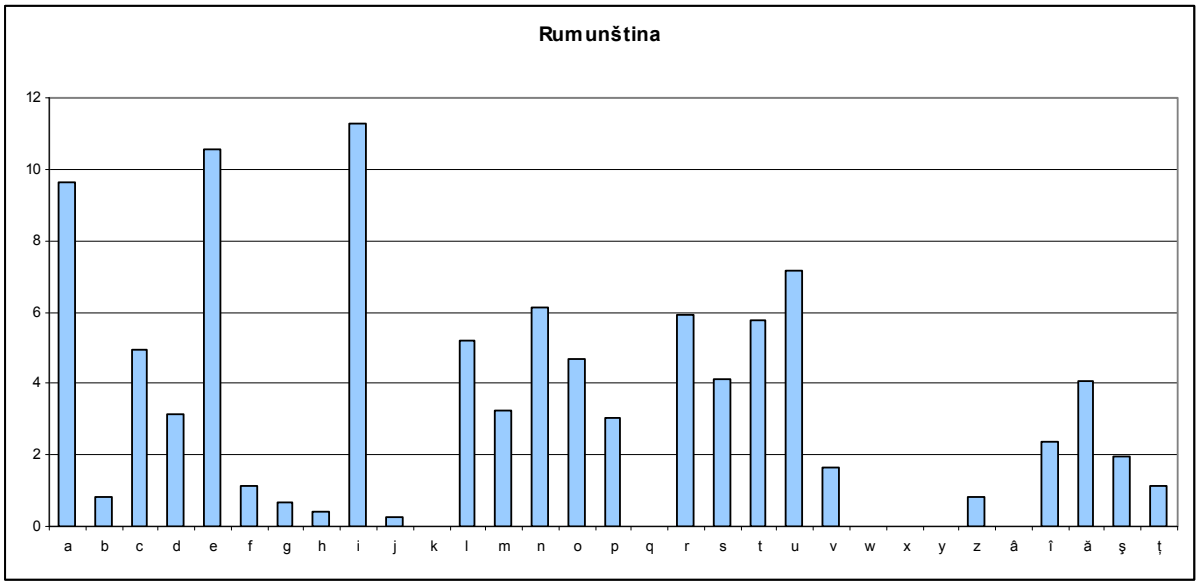

Němčina
18
16
14
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p q r s t u v w x y z ß ä ö ü

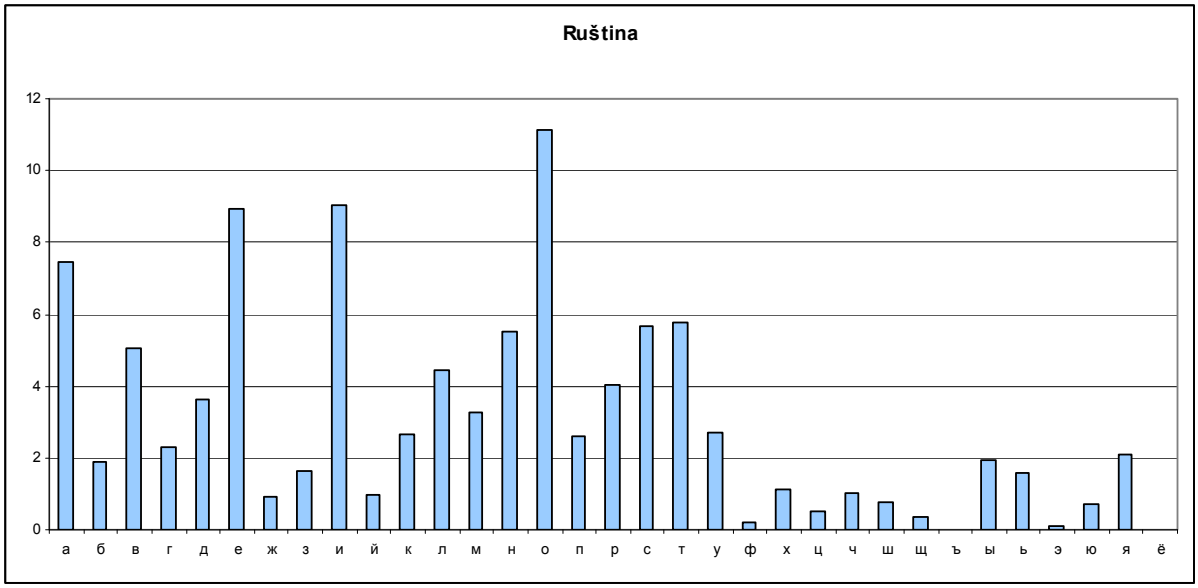

Nizozemština
25
20
15
10
5
0
a b c d e f g h i j k l m n o p q r s t u v w x y z

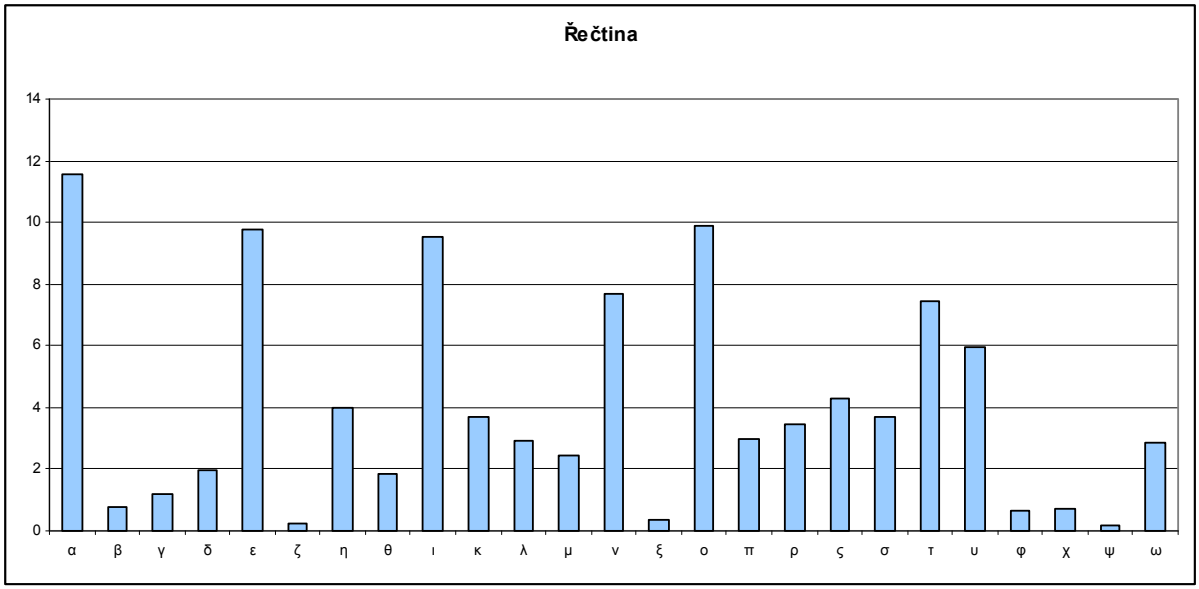

Norština
20
18
16
14
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p q r s t u v w x y å æ ø

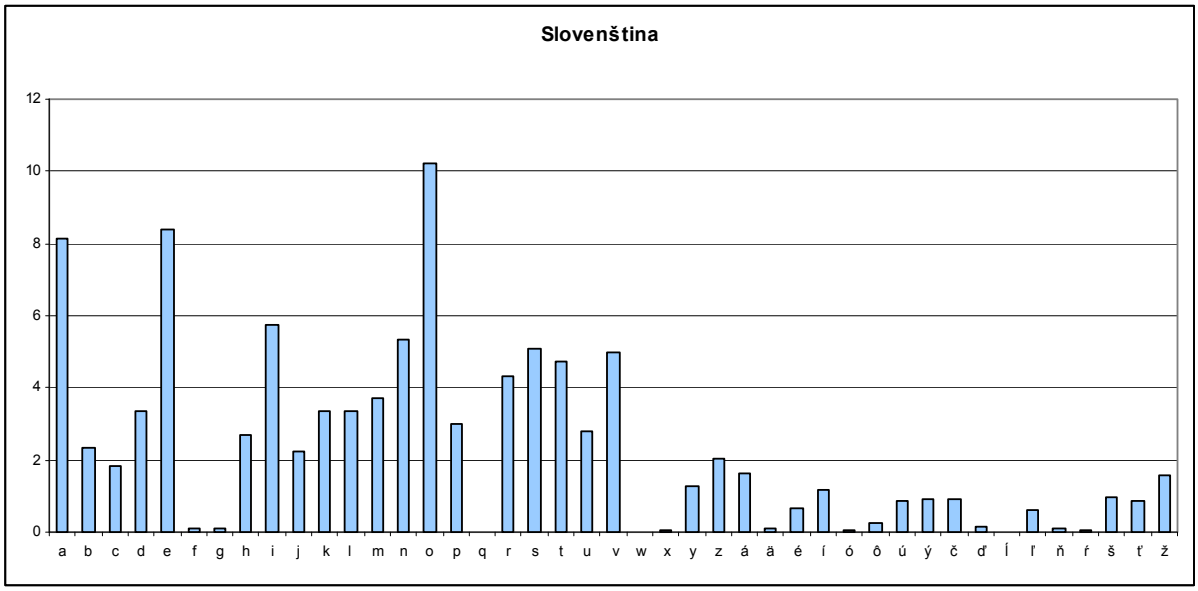

Polština
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p q r s t u v w x y z ó ą ć ę ł ń ś ź ż

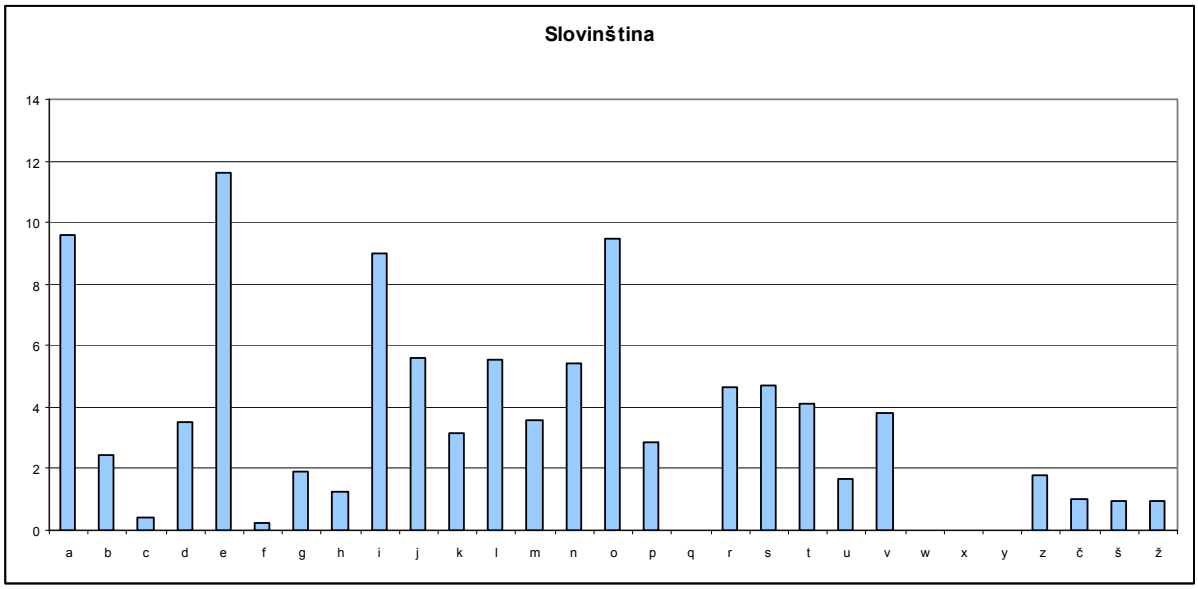

Portugalština
16
14
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p q r s t u v w x y z à á â ã ç è é ê í ó ô õ ú ü

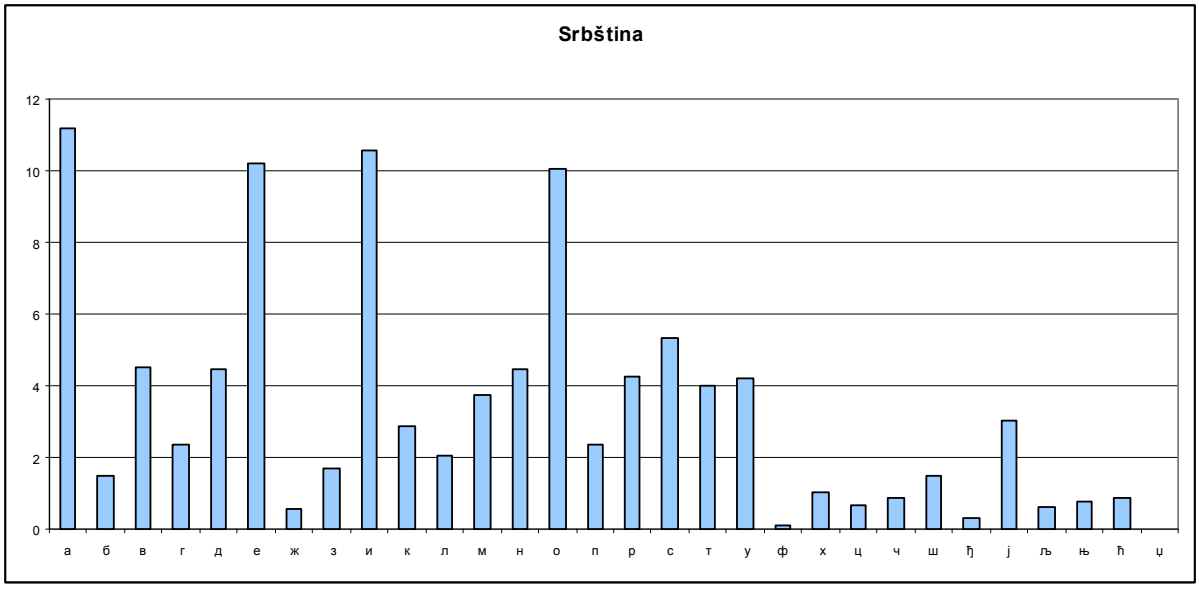

Rumunština
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p q r s t u v w x y z â î ă ş ţ

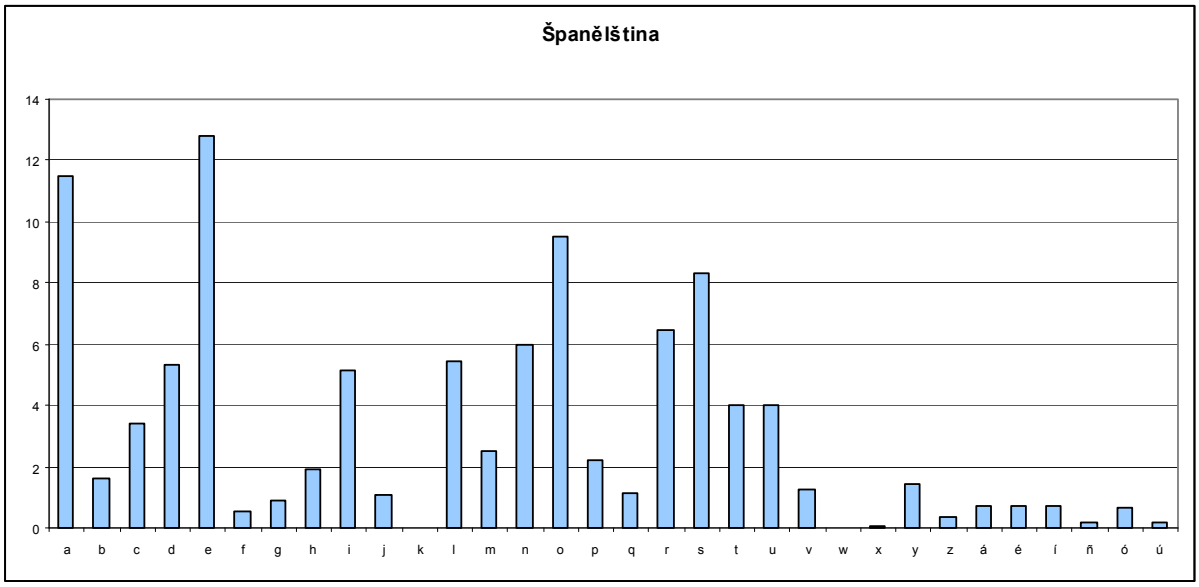

Ruština
12
10
8
6
4
2
0
а б в г д е ж з и й к л м н о п р с т у ф х ц ч ш щ ъ ы ь э ю я ё

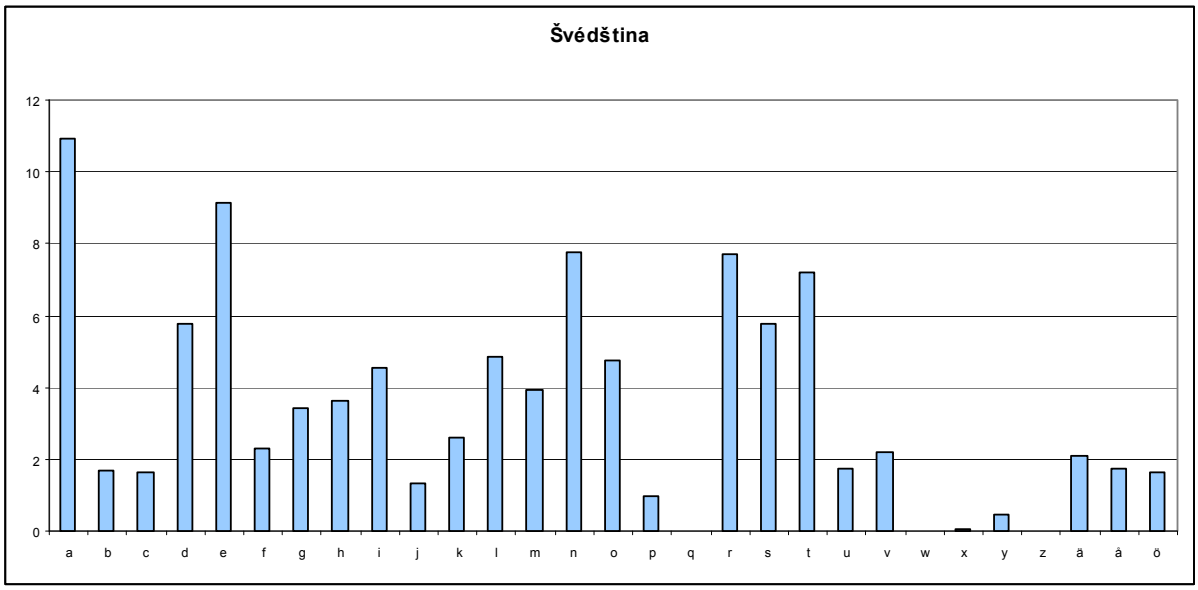

Řečtina
14
12
10
8
6
4
2
0
α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ ς σ τ υ φ χ ψ ω

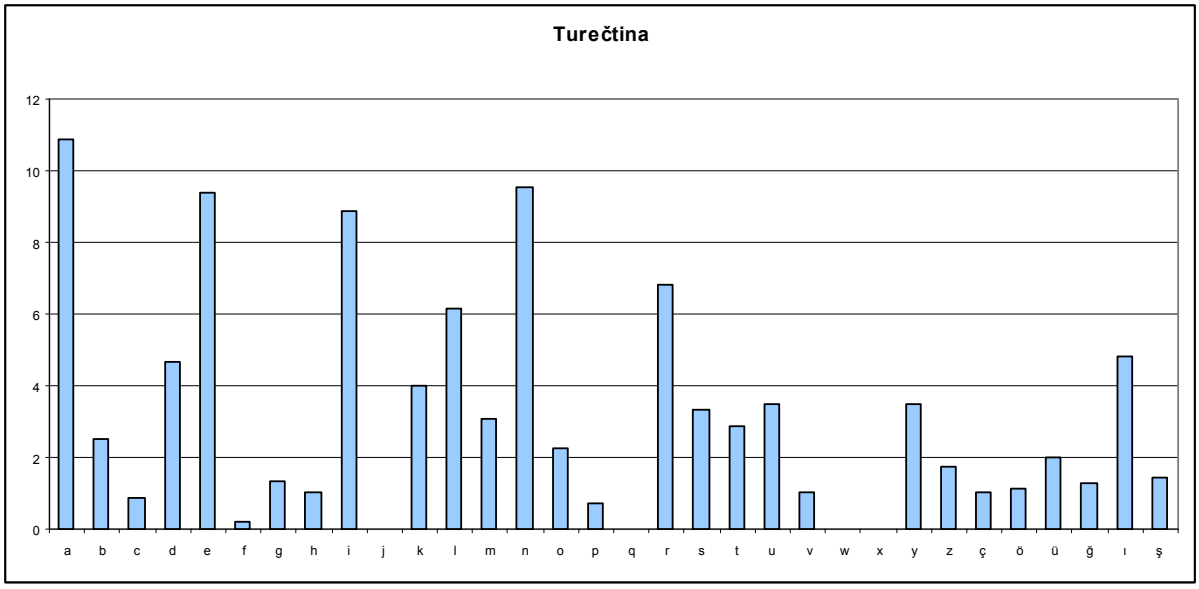

Slovenština
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p q r s t u v w x y z á ä é í ó ô ú ý č ď Í ľ ň ŕ š ť ž

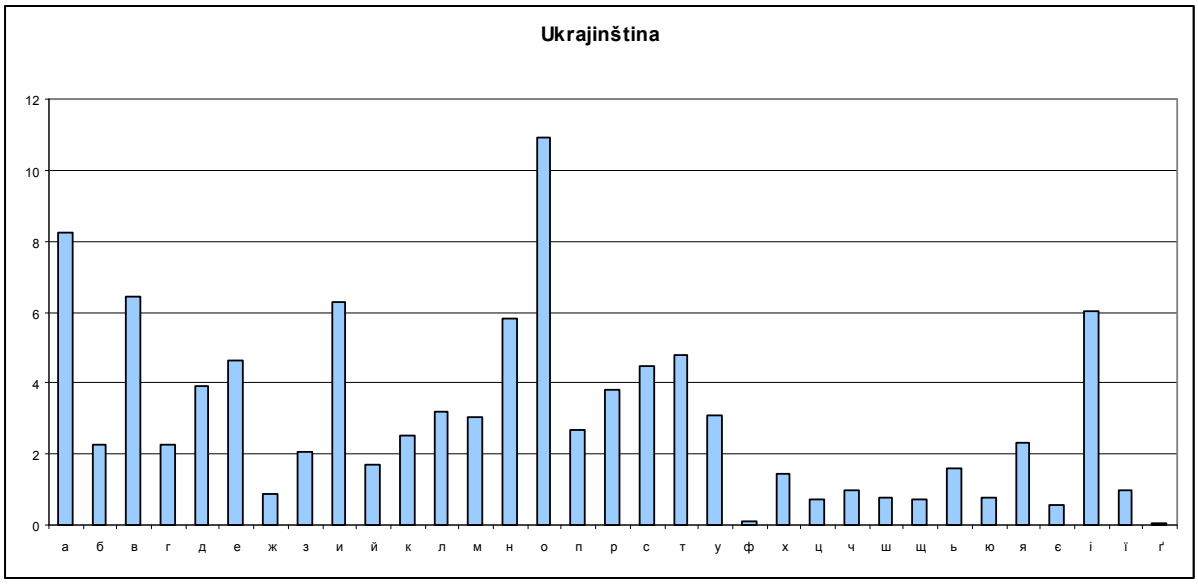

Slovinština
14
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p q r s t u v w x y z č š ž


Srbština
12
10
8
6
4
2
0
а б в г д е ж з и к л м н о п р с т у ф х ц ч ш ђ ј љ њ ћ џ


Španělština
14
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p q r s t u v w x y z á é í ñ ó ú


Švédština
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p q r s t u v w x y z ä å ö


Turečtina
12
10
8
6
4
2
0
a b c d e f g h i j k l m n o p q r s t u v w x y z ç ö ü ğ ı ş


Ukrajinština
12
10
8
6
4
2
0
а б в г д е ж з и й к л м н о п р с т у ф х ц ч ш щ ь ю я є і ї ґ