Filozofická fakulta Univerzity Palackého v Olomouci

Katedra obecné lingvistiky

Kvantitativně lingvistický software

magisterská diplomová práce

Autor: Bc. Vladimír Matlach

Vedoucí práce: Mgr. Radek Čech, Ph.D.

Olomouc

2014

Prohlášení

Prohlašuji, že jsem bakalářskou/magisterskou diplomovou práci „Název diplomové práce“ vypracoval/a samostatně a uvedl/a jsem veškerou použitou literaturu a veškeré použité zdroje.

V Olomouci dne 2. 5. 2014 Podpis

Abstrakt

Název práce: Kvantitativně lingvistický software

Autor práce: Bc. Vladimír Matlach

Vedoucí práce: Mgr. Radek Čech, Ph.D.

Počet stran a znaků: 79 pages, 130 000 characters

Počet příloh: 2

Abstrakt (minimálně 900 znaků):

Cílem této práce je vytvořit a dále představit software, který zprostředkuje kvantitativně lingvistickou analýzu textů, a to na základě zásady uživatelské přívětivosti a přístupu, který od uživatele tohoto softwaru nebude vyžadovat jakékoliv hlubší či systematičtější znalosti matematiky, statistiky, programování a kvantitativní lingvistiky, ať už v teoretické nebo jen praktické rovině. Smyslem této práce je tedy umožnit nejen kvantitativní lingvistice, ale i dalším vědním oborům a disciplínám (jakými jsou například historiografie, psychologie, biologie aj.), využívat kvantitativně-lingvistické analýzy textů pomocí specializovaného softwaru, který uživateli poskytne všechny potřebné metody a nástroje takové analýzy, počínaje samotným zpracováním textů a konče statistickým vyhodnocením a vizualizací výsledků. Text této práce se pak zabývá esenciální problematikou, která v tomto kontextu kvantitativní lingvistiku provází, tj. zejména problematikou tokenizace a lemmatizace. Dále se text věnuje několika plně ilustrovaným způsobům, jakými lze tento software používat, včetně několika ukázek jeho reálných aplikací. Závěrem se práce zaměřuje na možnosti, jakými lze představovaný software upravovat a implementovat do něj nové funkce.

Klíčová slova: Kvantitativní lingvistika, matematická lingvistika, software, statistika, indexy, analýza charakteristik textu.

Abstract

Title: Quantitative Linguistic Software

Author: Bc. Vladimír Matlach

Supervisor: Mgr. Radek Čech, Ph.D.

Number of pages and characters: 79 pages, 130 000 characters

Number of appendices: 2

Abstract (900 characters):

The aim of this thesis is to create and introduce new software, which will provide quantitative linguistic text analysis to users by respecting user-friendly control and over-all approach, which does not require any deeper or systematic knowledge of math, statistics, programming or quantitative linguistics, either in theoretical or practical form by its users. The purpose of this work is to enable quantitative linguistics and any other scientific discipline in the broadest sense (especially disciplines like historiography, psychology, biology etc.) to use quantitative linguistic text analysis with specialized software, which would provide all necessary methods and tools of this analysis to the user, starting with the text processing itself, and ending with statistical evaluation and visualization of the results. Text of this thesis is dedicated to present and introduce essential problems of quantitative linguistics in the context of natural text processing. Subsequently, the fully illustrated methods of the software usage are presented, including few previews of its real applications. The final section focuses on possibilities, how could be this presented software modified and how indices could be implemented.

Keywords: Quantitative linguistics, software, statistics, indices, text characteristics analysis.

Obsah

Úvod

Primárním cílem této práce je vytvořit software, který by zjednodušil přístup ke kvantitativně lingvistickým metodám a jejím aplikacím bez toho, aby bylo pro jeho uživatele nutné mít jakékoliv hlubší či systematičtější znalosti kvantitativní lingvistiky, statistiky či celého matematického aparátu stojícího v základech těchto metod. Cílem této práce je v tomto smyslu jednak umožnit co nejširšímu množství oborů využívat kvantitativní analýzu textů a dále podpořit samotnou kvantitativní lingvistiku nástrojem, který by umožnil vykonávat automatizované výpočty nad velkým množstvím vstupních dat. Zpracování pomocí takového softwaru by tedy umožnilo ušetřit čas, který by bylo jinak nutné věnovat rutinním a ne příliš triviálním výpočtům a jejich následným kontrolám. Úspora času a množství práce by tak umožnila testovat kvantitativně lingvistické hypotézy na nesrovnatelně větším množství dat, než by tomu bylo s alespoň minimálním komfortem při ručních výpočtech.

Sekundárním cílem je tento software (resp. jeho zdrojový kód) napsat a navrhnout tak, aby mohl kterýkoliv kvantitativní lingvista co nejjednodušeji přidat vlastní nový index (kvantitativní metodu) a bez jakýchkoliv dalších zásahů do zbytku softwaru jej okamžitě začít používat. Tento požadavek vznikl na základě empirické zkušenosti, kdy jsou rozličné indexy implementovány jako samostatné programy, skripty či makra, ve kterých se běžně opakují paradigmatické chyby softwarového návrhu. Tyto chyby následně znepříjemňují kvantitativním lingvistům další práci spojenou s úpravami těchto skriptů.

Text této práce je koncipován následovně: Kapitola 1 se zabývá problematikou propojení kvantitativní lingvistiky a počítačového zpracování jazykových dat. Kapitola 2 se věnuje popisu práce se softwarem QUITA, způsobům jeho nastavení a konkrétním příkladům jeho použití při běžných kvantitativně lingvistických úlohách. Kapitola 3 ilustruje použití softwaru na konkrétních kvantitativně lingvistických problémech. Kapitola 4 se věnuje způsobům přidávání vlastních indexů, jejich testování a celkově jejich samotné tvorbě způsobem „krok po kroku“ od úplných začátků, včetně mikro úvodu do programování a možností dalších úprav.

1 Kvantitativní lingvistika a počítačové zpracování

Počítačové zpracování indexů má mnohá úskalí. Ta se ve svém základu týkají především zcela obecné problematiky, jakou je samotné strojové zpracování přirozeného jazyka. Například dokázat rozlišit, co je ještě jedním slovem a co už dvěma či více slovy, není triviálním úkolem ani pro samotného člověka, natož, aby mohla být všechna taková pravidla pro tzv. *tokenizaci* verbalizována a zapsána do programu. Dále, pokud ponecháme tyto obecné problémy stranou, se dostáváme k možnostem, které pro své potřeby a cíle kvantitativní lingvistika má – v kontextu počítačů – k dispozici. Práce kvantitativních lingvistů je často omezena pouze na tabulkové procesory, různé statistické a matematické nástroje a různé skriptovací jazyky. Avšak spolupráce mezi všemi těmito programy a nástroji je zdlouhavá a především nejednotná. *Ad hoc* implementace indexů pomocí skriptovacích či jiných programovacích jazyků nejsou výjimkou. V následujících podkapitolách se blíže seznámíme s obecnou problematikou zpracování přirozeného textu pomocí počítačů a čistě ilustrativní metodou bude nabídnuto srovnání „skriptovací praxe“ a možností, které nabízí QUITA.

1.1 Problematika tokenizace, lemmatizace a POS Taggingu

Každý index kvantitativní lingvistiky přirozeně vyžaduje nějaký vstup, na kterém následně zakládá své výpočty. Typicky se jedná o informace jako *množství všech slov* daného textu (neboli tzv. počet *tokenů*, anglicky *tokens*, zkr. N) a počet jejich *unikátních výskytů* (tj. tzv. počet *typů,* anglicky *types*, zkr. V). Ovšem, i takové základní informace, jako jsou právě zmíněné *počty tokenů* a *počty typů*, je nejprve nutné pro daný text získat. Ačkoliv se mohou tyto dvě základní úlohy zdát marginální a jednoduché, je tomu právě naopak.

Identifikace tokenů (respektive jejich strojová identifikace) naráží na řadu problémů, které plynou především z různých očekávání toho, jaký výsledek by tento proces (tzv. *tokenizace*) měl vůbec mít. Příkladem může být jednoduchá věta: „Karel IV. byl král.“ V tomto konkrétním případě okamžitě docházíme k otázce, zda je „Karel IV.“ jeden, nebo více tokenů. V případě, že by se jednalo o jediný token, pak vyvstává otázka, jak by bylo možné takové tokeny obecně identifikovat. Musel by existovat jejich úplný výčet, který, vzhledem k povaze přirozeného jazyka, s největší pravděpodobností nikdy úplným být nemůže. Zcela obdobně se pak můžeme ptát na množství tokenů ve slově „zda-li“, anglickém „aren’t“ a „are not“ a především obecně vzato i na množství tokenů jakýchkoliv frazémů, jako např. „mírnyx týrnyx“.

Konkrétní způsob tokenizace tedy může být zcela otázkou cíle, osobních preferencí nebo jen čisté pragmatiky. Z pohledu implementace tokenizátorů je pak nasnadě, že běžný naivní přístup vymezení tokenů jako „něčeho, co je mezi mezerami“ selže v okamžiku nutnosti jemnějšího rozlišování (viz příklad „aren’t“ a „are not“ výše). Tato úskalí už v tak brzkých fázích zpracování přirozeného textu následně implikují i složitost identifikace dalších textových jednotek, jakými jsou mj. věty. Právě u vět okamžitě selhává prvotní naivně se nabízející definice jako „něčeho od tečky k tečce“. (Více informací k problematice tokenizace je dále možné nalézt např. v Indurkhya 2010.)

Ovšem tokenizace identifikuje pouze jednotlivé tokeny – druhou klíčovou informací, jak již bylo zmíněno výše, je počet (a často i plný seznam) typů (neboli *velikost slovníku* a *slovník*). Zjištění typů v daném textu je tedy hierarchicky závislé na výstupu procesu tokenizace a jakékoliv chyby v ní se pak v procesu zjišťování slovníku objevují a zesilují. Vytváření slovníku má navíc i vlastní úskalí – je jím problematika

ekvivalence dvou tokenů – tj. otázky, jakou metodou tokeny rozlišovat mezi sebou a jakou je k sobě naopak přiřazovat. Naivní a nejjednodušší přístup pouhého porovnání jednotlivých grafemických jednotek, ze kterých se tokeny skládají, totiž nemusí být správný hned z několika důvodů: (1) Ačkoliv jsou tokeny „aren't" a „are not" odlišné, tak by je bylo možné za určitých okolností považovat za tentýž typ. (2) Zcela opačný problém vůči předcházejícímu bodu – některé tokeny mohou být grafemicky stejné, avšak kontext, ze kterého jsou vytrženy, jim dává zcela odlišný význam (např. verbum „stát" a substantivum „stát"). Tento druhý bod pak vede k otázce, zda takové dva tokeny jsou jediným typem. Flexivní jazyky přinášejí další rozměr, a to, zda a jak k sobě přiřazovat různé slovní formy, které jsou však výsledkem ohýbání téhož slova.

Řešením pro flexivní (ale i jiné) jazyky je tzv. lemmatizace, která každé slovo (resp. token, který již navíc může být tokenizován chybně) převede na jeho základní tvar. Například tokeny „veverce", „veverko" a „veverkou" jsou převedeny na základní formu „veverka". Znovu zde však vznikají problémy s jemnějším rozlišováním slovních tvarů na základě kontextu a sémantiky. Např. to, zda je lemma pro token „století" *stoletý* nebo *století*, můžeme říct až po použití tohoto slova v konkrétní větě a kontextu: „Století starci." oproti „Dvacáté první století.". Aby tedy bylo vůbec možné provést správnou lemmatizaci, je nutné větě porozumět i po sémantické stránce, případně alespoň po stránce statistické.

Zcela obdobnými problémy jako lemmatizace pak trpí i určování slovních druhů pro jednotlivé tokeny. Tokenu „stát" je možné přiřadit slovní druh – v tomto případě buď *verbum*, nebo *substantivum* – až po analýze kontextu věty, ve které je tato slovní forma obsažena (např. „Musel stát na židli, aby tam dosáhl." a „Právní stát.").

Z výše uvedeného následně plyne, že strojové vytváření slovníku – tj. zcela základního a zásadního údaje používaného při výpočtech indexů kvantitativní lingvistiky – je zatíženo velkým množstvím problémů. Každý ze zmíněných partikulárních kroků zpracování textu se potýká s více či méně vlastními specifickými problémy, které pak ve svých důsledcích ovlivňují kroky dalšího zpracování. Z těchto důvodů je pak na místě zvážit, jakým způsobem text před výpočty zpracovat, aby bylo dosaženo kýžených výsledků a přesnosti. Rovněž z toho plyne nutnost pochopit, jakým způsobem jsou předávána vstupní a výstupní data nástrojů zpracovávající text a jak jsou tato data dále používána.

Tématice tokenizace, lemmatizace a POS taggingu se v kontextu programu QUITA budeme dále věnovat v kapitole 2. *Práce s programem QUITA* a podkapitolách 2.7.4 *Tokenizátor*, 2.7.5 *Lemmatizátory*, 2.7.6 *POS taggery* a 2.7.9 *Tokenizace, lemmatizace a POS tagging v QUITA*.

1.2 Skriptovací praxe kvantitativní lingvistiky

V praxi jsou různé indexy dle aktuální potřeby jednotlivě implementovány ve skriptovacích či jiných programovacích jazycích, které nemají příliš vhodnou podporu práce s jazykovými daty nebo je příliš složité takovou podporu správně uchopit a používat. Implementace tohoto stylu jsou problémové z mnoha důvodů, včetně několika klíčových: (1) implementace *ad hoc* často postrádají jakoukoliv míru abstrakce – tj. jednotlivé logické celky celého postupu nejsou sestaveny z oddělených částí plnících pouze jedinou konkrétní činnost (typicky „načítání ze souboru“, „tokenizace“, „samotný výpočet“ a „výstup“), ale jsou vměstnány „za sebe“ do jediného kusu kódu. (2) Jsou použity funkce a postupy, které s řešením čistě kvantitativně-lingvistického problému nesouvisí nebo by měly být již pro samotné výpočty hotové. Typicky se tak jedná např. o již zmíněné „načítání ze souboru“, provádění tokenizace, vypisování výsledků na obrazovku a mnoho dalších. Navíc je nutné implementovat i matematické funkce, které jsou pro kvantitativní lingvistiku typické (např. funkce sumy), které buď nejsou v daném jazyce k dispozici, nejsou dostatečně flexibilní, nebo je jen příliš složité tyto funkce používat a vyplatí se je implementovat *ad hoc* (v případě sumy např. *for* smyčkou). Veškeré tyto problémy pak znesnadňují čitelnost celého kódu, jeho rychlé pochopení a kontrolu použitého matematického aparátu. (3) Správa takového skriptu či programu je pak kvůli oběma předchozím bodům v zásadě obtížná a komplikovaná. Úpravy v kterékoliv části kódu (díky absenci abstrakce) ohrožují funkčnost celého programu a musí být předem velmi dobře promyšleny.

K ilustraci této problematiky se podíváme na index *Lambda* (Popescu 2011), který je definovaný následujícím vzorcem:

$$\Lambda = \frac{L(log_{10}\, N)}{N} \quad ,$$

kde N je počet tokenů, f_i frekvence s rankem i, V je počet typů a

$$L = \sum_{i=1}^{V-1} [(f_i - f_{i+1})^2 + 1]^{1/2} \quad .$$

Konkrétní realizace celého výpočtu indexu *Lambda* v jazyce R (http://www.r-project.org/; autor programu Radek Čech):

```
lamf<-function(x){
    #zpracovani textu
    prvni_text<-scan(file=x, what="char", sep="\n")
    prvni_text_mala_pismena<-tolower(prvni_text)

    prvni_text_mala_pismena<-gsub("'", "", prvni_text_mala_pismena)
    slova_text<-unlist(strsplit(prvni_text_mala_pismena, "\\W"))
    slova_text<-unlist(strsplit(slova_text, " "))
    frekvence_slov<- table(slova_text)
    klesajici_frekvence_slov<-sort(frekvence_slov, decreasing=T)
    word_frequencies <- klesajici_frekvence_slov

    #vypocty
    rf <- word_frequencies/sum(word_frequencies)
    V <- length(rf)
    a <- rep(0,V)
    N <- sum(word_frequencies)
    for (i in 2:(V-1))
    a[i] <- -(N-word_frequencies[1])*( (rf[i-1]-rf[i]) /(1-rf[1]))/
      sqrt( ((N-word_frequencies[1])^2)*( ((rf[i-1]-rf[i])/(1-rf[1]))^2)+1 )+
      (N-word_frequencies[1])*( (rf[i]-rf[i+1]) /(1-rf[1]))/
      sqrt( ((N-word_frequencies[1])^2)*( ((rf[i]-rf[i+1])/(1-rf[1]))^2)+1 )

    a[V] <- -(N-word_frequencies[1])*( (rf[V-1]-rf[V]) /(1-rf[1]))/
       sqrt( ((N-word_frequencies[1])^2)*( ((rf[V-1]-rf[V])/(1-rf[1]))^2)+1 )
    s1<-0
    for (i in 2:V) s1 <- s1 + (a[i]^2)*rf[i]*(1-rf[i]/(1-rf[1]))
    s2<-0
    for (i in 2:(V-1)) for (j in (i+1):V)
    s2 <- s2+rf[i]*rf[j]*a[i]*a[j]
    VarL <- ((N-word_frequencies[1])/(1-rf[1]))*s1 -
           2*((N-word_frequencies[1])/((1-rf[1])^2))*s2

    Var_Lambda <- VarL *((log10(N)^2))/(N^2)
    L<-0
    for (i in 1:(V-1)) L<-L+ sqrt((word_frequencies[i]-
word_frequencies[i+1])^2+1)
    Lambda<-L*log10(N)/N

    result<-c(N=N,V=V,lam=Lambda,lamvar=Var_Lambda)
    return(result)
}
```

Při pohledu na takový kód nemusí být okamžitě jasné, který výpočet a zápis je relevantní pouze pro výpočet samotného indexu *Lambda*. Nalezení např. chybného zápisu v jednom ze vzorců může být velmi komplikovaným a dlouhodobým úkolem. Framework zdrojového kódu QUITA se pak snaží na jednotlivé výše uvedené problémy poskytnout co nejpohodlnější (tj. nejčitelnější) řešení. Pro ilustraci se dále podívejme, jak vypadá implementace stejného indexu v QUITA. Pro úplné srovnání a názornost implementace znovu uvádím i oba vzorce.

Implementace indexu *Lambda* v QUITA:

$$L = \sum_{i=1}^{V-1} [(f_i - f_{i+1})^2 + 1]^{1/2} \quad , \quad \Lambda = \frac{L(log_{10}\, N)}{N} \quad .$$

```
Public Overrides Function Calculate(ByRef TextData As IQITAText) As IQITAResult
  Dim V As Long = TextData.V()
  Dim N As Long = TextData.N()
  Dim f As Integer() = TextData.GetFrequencyPositiveArray()
  Dim L As Double

  L = SumAllFrequencies(TextData, 1, V - 1, _
                        Function(fi As Long, i As Integer) _
                                  (Math.Sqrt((f(i) - f(i + 1)) ^ 2 + 1)))

  Dim Lambda As Double = (L * Log10(N)) / N

  Return New QITANumberResult(TextData, Me, Lambda)
End Function
```

Čitelnost celého kódu je zvýšena především tím, že se autor indexu v QUITA nemusí zaobírat partikulárními úkoly, které jsou nutné k získání všech potřebných dat a informací o textu – ty jsou mu poskytnuty společně s celým textem na vstupu výpočetní rutiny (*Function Calculate*). Dále není potřeba implementovat běžné matematické funkce, jakou je např. výše použitá funkce sumy a některé další. QUITA se tak snaží poskytnout všechny základní metody výpočtů a informací takovým způsobem, aby co nejvíce zjednodušila práci autorům samotných indexů. Autor se pak nemusí starat ani o způsob zobrazení výsledků – ten je udán již samotným typem výsledku a bude pro něj zvolen nejvhodnější způsob zobrazení. Jazyk, který byl pro implementaci programu QUITA cíleně zvolen, je jedním z velmi rychle zvládnutelných programovacích jazyků, a to díky jeho snadné a přehledné syntaxi a díky používání běžných klíčových slov. Způsobům, jak vytvářet vlastní indexy v QUITA, a základům použitého programovacího jazyka se dále věnuje kapitola č. 4 *Úpravy softwaru QUITA a práce se zdrojovými kódy*. V následující kapitole se budeme věnovat tomu, jak program QUITA používat po čistě uživatelské stránce.

2 Práce s programem QUITA

V této kapitole postupně rozebereme, co QUITA po uživatelské stránce nabízí a co konkrétně motivovalo implementaci těchto funkcí. Dále se zde budeme zabývat důležitými nastaveními – především nastavením tokenizátorů, lemmatizátorů, POS taggerů a různých nástrojů pro finální úpravy textu (tzv. *post-procesory)* a tím, jak tato různá nastavení ovlivňují chování aplikace a samotné výpočty. Na závěr kapitoly jsou ilustrovány základní postupy a práce s programem QUITA, a to pomocí několika úloh, které se v kvantitativní lingvistice běžně řeší

2.1 Cíle programu QUITA

Obecné cíle programu QUITA již byly vysvětleny a nastíněny v úvodu této práce. Zde si však rozebereme partikulární motivace, které vedly k celkovému návrhu programu a implementacím některých funkcí, které tak pomohou osvětlit i samotné ovládání celého programu.

Jedním ze základních cílů a motivací tvorby QUITA byla možnost pracovat s více texty zároveň a dále možnost porovnávat výsledky indexů pro jednotlivé texty mezi sebou. Tento požadavek vznikl na základě běžné kvantitativně lingvistické práce s více vzorky.[1] Dalším cílem bylo umožnění porovnání vzorků navzájem.[2] Tyto dvě motivace pak vedly k celému návrhu programu QUITA tak, aby uživateli umožňoval pracovat tímto způsobem bez jakýchkoliv problémů. QUITA tedy umožňuje vytvořit projekt, který bude obsahovat např. pouze kapitoly knihy *Osudy dobrého vojáka Švejka za světové války*. V rámci projektu je pak možné výsledky jednotlivých kapitol vzájemně porovnávat. Dále však může uživatel vytvořit další projekty, které budou obsahovat kapitoly např. jiných knih, nebo i stejné kapitoly zpracované jiným způsobem. Všechny tyto projekty a jejich partikulární výsledky QUITA umožňuje mezi sebou navzájem srovnávat. Kapitoly lze nahradit většími či menšími jednotkami, a tak koncept projektů použít např. pro porovnávání všech dostupných děl jednoho autora se všemi dostupnými díly dalšího autora.

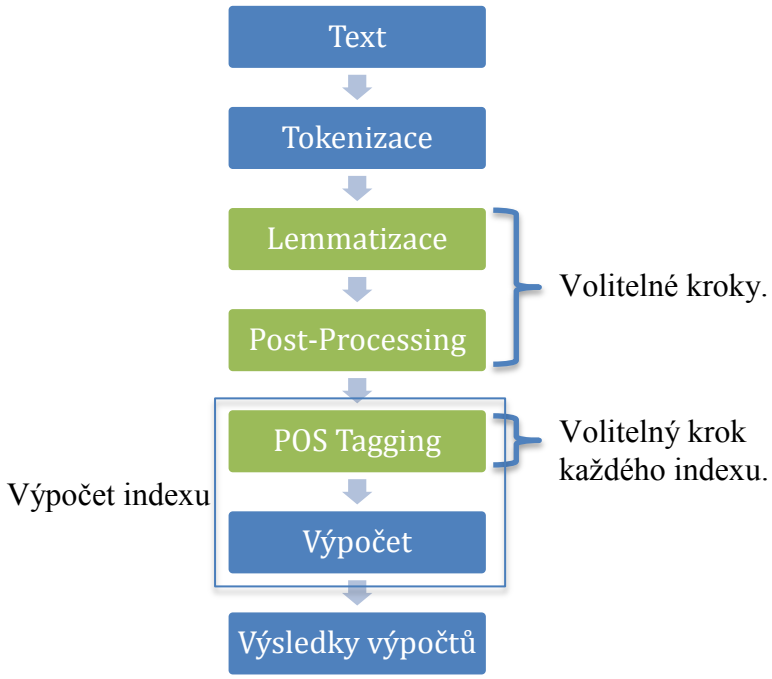
Současně bylo cílem vytvořit uživateli takový komfort, aby nutně nepotřeboval k základním úkolům další programy. QUITA proto umožňuje vytvářet grafy pro vykreslování prakticky libovolné kombinace závislostí. Tyto grafy lze následně exportovat do tisknutelné podoby. Dále QUITA obsahuje zabudovaný prohlížeč výsledků porovnání, a to jak vzorků daného projektu mezi sebou, tak i zobrazení výsledků porovnání mezi samotnými projekty. QUITA dále umožňuje veškeré výsledky exportovat do jakéhokoliv tabulkového procesoru (např. Microsoft Excel nebo OpenCalc) jediným kliknutím, a tak uživatele neomezovat pouze na práci v programu QUITA – ten může být použit pouze pro samotné výpočty.

[1] Pokud je např. potřeba zpracovat všechny kapitoly knihy *Osudy dobrého vojáka Švejka* (kterých je 27) s tím, že mají být všechny tyto kapitoly následně porovnány mezi sebou pomocí indexu *Tématické koncentrace*, znamenalo by to jednak vypočítat 27 hodnot indexu *Tématické koncentrace*, ale následně dalších 351 výpočtů rozptylů a dalších 351 *u*-testů. Pravděpodobnost, že při takových výpočtech prováděných ručně např. pomocí tabulkového procesoru, dojde vlivem nepozornosti, únavě atd. k chybě, je pak značně veliká.

[2] Tj. např. se vzorkem kapitol jiné knihy nebo porovnat všechna díla jednoho autora se všemi díly jiného autora. V takovém případě se již neporovnávají samotné kapitoly mezi sebou, ale porovnávají se všechny výsledky daného indexu pomocí statistických metod.

2.2 Schéma zpracování dat

Jak bylo vysvětleno v podkapitole 1.1 *Problematika tokenizace, lemmatizace a POS Taggingu*, je u každého programu pracujícím s texty nutné vědět, jak zachází se vstupními a výstupními daty z jednotlivých nástrojů zpracovávajících text. Zpracování dat v QUITA probíhá způsobem znázorněným jednoduchým diagramem níže (viz Obrázek 1 - Schéma zpracování dat v QUITA):



Obrázek 1 - Schéma zpracování dat v QUITA

(1) Text na vstupu je předán tokenizátoru, který text převede na řetěz tokenů. Tento řetěz tokenů pak slouží jako vstup volitelnému kroku (2) *lemmatizace*, ve kterém je každému tokenu přiřazeno lemma. (Lemmatizovaný) řetěz tokenů následně prochází dalším volitelným krokem (3) tzv. post-processingu, který může zpracovaný text upravit např. redukcí jeho délky na zadaný počet tokenů nebo z tokenů vytvořit tzv. *n-gramy*. Výsledek tohoto zpracování je následně předán jako vstup každému indexu. Každý index podle svých potřeb může využít (4) tzv. POS-tagging (tj. zjištění slovního druhu) daného tokenu. (5) Index pak produkuje finální výsledek (ten může být případně použit i jako vstup dalšímu indexu; typicky je takto využíván *h-bod*). (6) Výsledky indexu jsou zobrazeny uživateli.

Způsob předávání dat a způsob tokenizace, lemmatizace a POS taggingu je zcela závislý na způsobu implementace modulů zprostředkovávajících tuto činnost a také na implementaci indexů samotných. Primární způsob, kterým se v QUITA předávají výsledky od tokenizátoru do lemmatizátoru, jsou bezkontextové dotazy na lemma daného tokenu – tzn. že ve smyslu podkapitoly 1.1 *Problematika tokenizace, lemmatizace a POS Taggingu* není možné tímto způsobem udělat kvalitní lemmatizaci, protože analytické modely kvalitních lemmatizátorů přichází o mnoho cenných informací. Tato naivní metoda předávání dat byla zvolena kvůli její rychlosti a snadnosti implementace. Každá implementace modulů zprostředkovávajících tokenizaci, lemmatizaci i POS tagging však může být změněna a může využívat jakákoliv data, která ke svému ideálnímu zpracování vyžaduje. Srovnání rozdílů výsledků některých indexů založených právě na tomto naivním zpracování a výsledků založených na profesionálně

zpracovaných textech, naleznete v kapitole 3.2 *Test naivní tokenizace a lemmatizace*. Způsobům, jak upravovat software QUITA se věnuje kapitola 4 *Úpravy softwaru QUITA a práce se zdrojovými kódy* a blíže pak podkapitola 4.7 – *Vytváření vlastních tokenizátorů, lemmatizátorů a POS taggerů*.

2.3 Použití existujících tokenizátorů, lemmatizátorů a POS taggerů

Tokenizaci, lemmatizaci a POS tagging QUITA sama neimplementuje, respektive neimplementuje všechny tyto nástroje, funkce a metody, kromě těch zcela nejzákladnějších (viz např. tokenizátory *Default Generic Tokenizer* a *Line Tokenizer* dále v této kapitole). Všechny tyto nástroje a jejich používané metody jsou stálé předmětem vývoje, avšak jsou často v aktuálních verzích publikovány jako samostatně dostupné programy, které je možné používat. QUITA nabízí možnost s těmito nástroji spolupracovat, komunikovat s nimi a vzájemně je kombinovat. Způsob spolupráce QUITA s těmito nástroji je pak zcela věcí konkrétní implementace daného „přemostění“ mezi jednotlivými nástroji a QUITA (více viz již zmíněná kapitola 4.7 *Vytváření vlastních tokenizátorů, lemmatizátorů a POS taggerů*). Na ukázku takové spolupráce je v aktuálně předkládané verzi vytvořeno přemostění pro několik nástrojů, zejména pak s balíkem NLTK (Natural Language Toolkit; viz http://www.nltk.org/).

Typ nástroje	Jazyk	Jméno	Autor
Tokenizér, lemmatizér, POS tagger	**EN**	NLTK (Natural Language Toolkit)	http://www.nltk.org/
Lemmatizér	**AR, DE, DK, ES, FI, FR, IT, NL, NO, PT, RO, RU, SE**	NLTK (Natural Language Toolkit)	http://www.nltk.org/
Lemmatizér	**EN**	Alexander Pak's Morphology	Alexander Pak
Lemmatizér	**AR, EN, RU**	Služba Text-processing.com	Text-processing.com
Lemmatizér, POS tagger	**CZ**	MAJKA	Pavel Šmerk (2007)
Lemmatizér, POS tagger	**CZ**	„Corpus“	

2.4 Instalace

Software QUITA je možné nainstalovat spuštěním instalačního souboru *QUITA_INSTALL.msi*, který je možné nalézt na přiloženém CD (viz Obsah přiloženého CD), nebo je možné jej stáhnout z webové stránky projektu: http://oltk.upol.cz/software

QUITA je možné nainstalovat a používat na systémech:

- Windows XP,
- Windows Vista,
- Windows 7 a
- Windows 8.

Ke spuštění a používání QUITA je nutné mít nainstalovaný *.NET Framework* verze 3.5, na jehož případnou absenci je uživatel upozorněn ihned na začátku instalace s automatickou nabídkou jeho stažení a doinstalování.

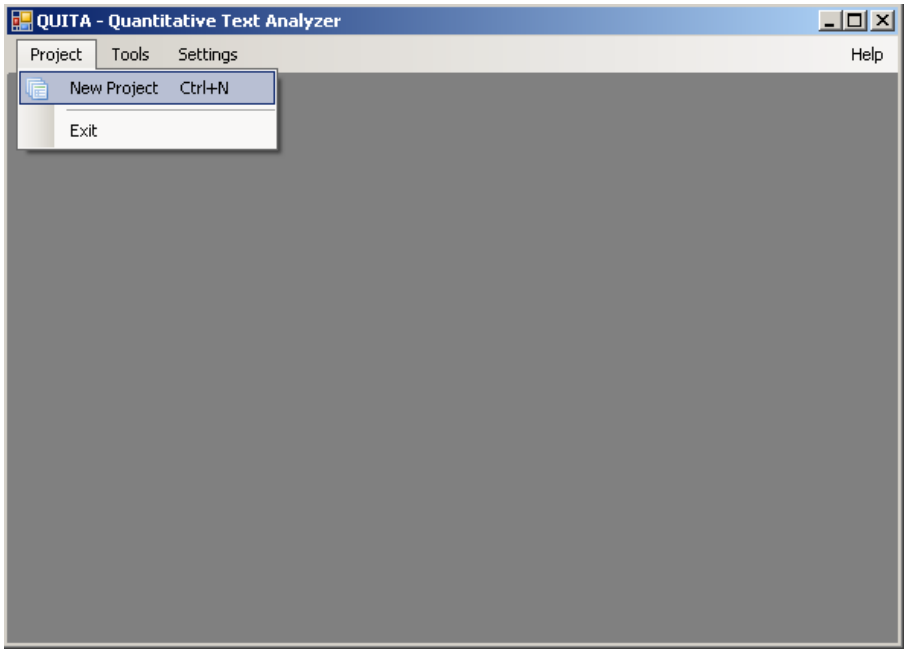
2.5 Spuštění aplikace

Spuštění aplikace je možné prostřednictvím automaticky vytvořeného zástupce „QUITA“ na ploše uživatele nebo spuštěním souboru *QUITA.exe*, nacházejícím se ve složce, kam byl program nainstalován. Výchozí cestou instalace je

C:\Program Files\QUITA Tools\QUITA .

2.6 Vytvoření nového projektu

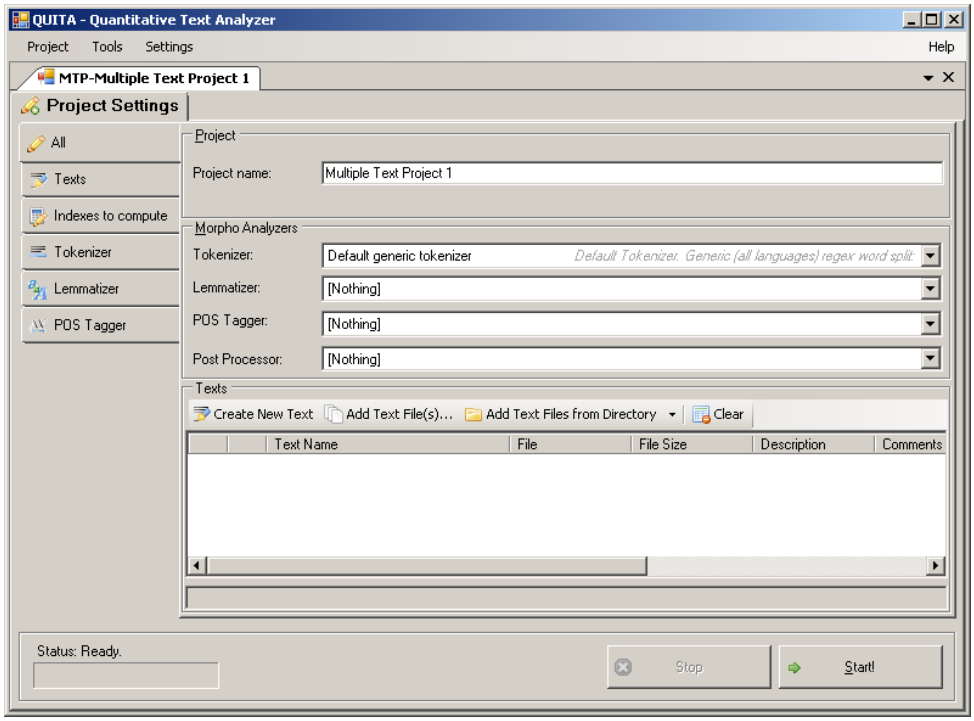
Po spuštění programu QUITA je možné okamžitě vytvořit nový projekt pomocí menu Project→New Project.



Obrázek 2 – Hlavní okno QUITA

2.7 Nastavení nového projektu

Nový projekt je nutné před samotnými výpočty nejprve nastavit tak, aby veškeré další zpracování vyhovovalo požadavkům jednotlivých indexů, a především odpovídalo požadavkům samotného uživatele. Nastavit je potřeba jméno projektu, způsob tokenizace, lemmatizace, případně použití POS taggeru a post-procesoru. Dále je samozřejmě nutné specifikovat či načíst texty, které mají být pomocí QUITA zpracovány. To vše je možné nastavit přímo na úvodní obrazovce nastavení projektu, tj. shrnující kartě „All“ (viz Obrázek 3 – Nový projekt), nebo přehledněji a s detailnějšími informacemi pomocí karet věnujícím se každému jednotlivému nastavení zvlášť (viz dále). Většinou však postačí vše nastavit právě v úvodní kartě „All“.



Obrázek 3 – Nový projekt

Po nastavení všech požadovaných vlastností zpracování a textů stačí spustit výpočty kliknutím na tlačítko Start, které se nachází ve spodní pravé části obrazovky. Dále si postupně rozebereme veškerá jednotlivá nastavení blíže.

2.7.1 Načítání textů

Texty, které mají být zpracovány, je do QUITA možné vložit čtyřmi základními způsoby:

1. Pomocí tlačítka v menu Create New Text, které nabídne jednoduchý a kompaktní textový editor, do kterého je možné vložit nebo napsat jakýkoliv text.
2. Pomocí „přetáhnutí“ (drag & drop) textových či jiných podporovaných souborů (viz dále) do okna seznamu textů.
3. Pomocí tlačítka v menu Add Text File(s), které nabídne dialog pro jednoduchý způsob označení jednoho či více souborů.
4. Tlačítkem Add Text Files From Directory, které načte všechny podporované soubory ze zadaného adresáře a jeho všech podadresářů podle předem zadaných podmínek a omezení. Blíže o tomto způsobu viz dále v této kapitole.

Podporované formáty

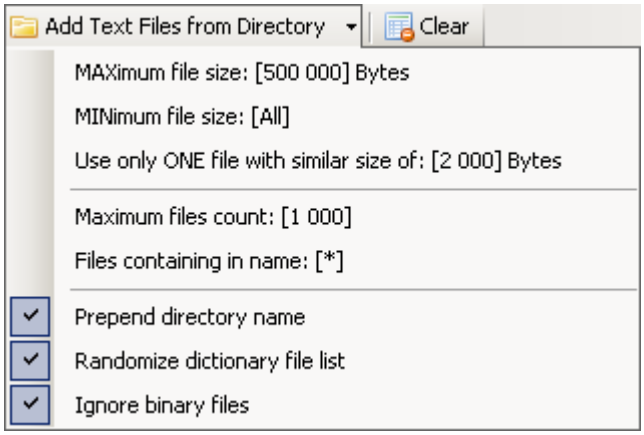
QUITA podporuje klasické textové soubory s příponou .TXT, tedy populárně řečeno tzv. „plain-text“ soubory. Dále podporuje HTML soubory s příponou .HTM a .HTML, ze kterých před zpracováním tokenizací odstraní všechny HTML značky atd. a získá z nich pouze obsažený čistý text. Dále jsou podporovány soubory s příponou .FNA obsahující nukleotidové sekvence. Podporované jsou všechny typy konců řádků, tj. CRLF, LFCR, CR nebo i jen LF. Podporována jsou všechna kódování dostupná v hostujícím systému, tj. kódování UTF-8, Unicode, Windows 1250 a desítky dalších kódování, podporujících různé jazyky a způsoby zápisu – více viz podkapitola 2.7.2 *Problematika kódování textu*. QUITA dále umožňuje načítat i čistě binární soubory, jako např. libovolné soubory .EXE, .DLL, .MP3, .AVI aj.. Tyto soubory pak převádí do čistě textové podoby a tím umožňuje, resp. se pokusí zprostředkovat možnost srovnávat přirozeně textová data s počítačovými konstrukty. Blíže viz *Překódování binárních souborů* dále v této kapitole.

Paměťová náročnost

Ačkoliv jsou jednotlivé textové soubory relativně malé, je stále možné jejich velkým množstvím – a především pak při samotných výpočtech a zpracování – vyčerpat natolik velké množství operační paměti systému, že dále nebude možné ve zpracování a výpočtech pokračovat. QUITA při zpracování textu a případně i po jeho zpracování uchovává v paměti výsledky všech zpracovaných indexů a případně i rozsáhlé tabulky obsahující seznamy všech tokenů, typů a tabulky jejich frekvencí, aby urychlila jejich případné (!) zobrazení uživatelem. Z tohoto důvodu je při velkém množství textů ke zpracování vhodné (1) zvolit k výpočtům pouze ty indexy, které jsou nezbytně nutné, (2) načíst a použít pouze texty, které jsou pro výsledky relevantní a (3) nastavit, jaké a zda vůbec mají být výše zmíněná rozsáhlá data v paměti uchovávány i po samotných výpočtech a zpracování textu (viz kapitola 2.7.7 *Nastavení Cache* dále v textu).

Hromadné načítání textů ze složek

QUITA nabízí možnost načíst všechny soubory ze zadané složky a všech jejích podsložek. K této možnosti dále nabízí i sadu filtrů, pomocí kterých lze toto hromadné načítání souborů omezit. Klepnutím na šipku tlačítka Add Text Files From Directory se rozbalí menu (viz Obrázek 4 – Načíst lze i všechny texty ze složek podle zadaných kritérií) s nastavením všech dostupných omezení a filtrů a dalších možností, které lze klepnutím upravovat.

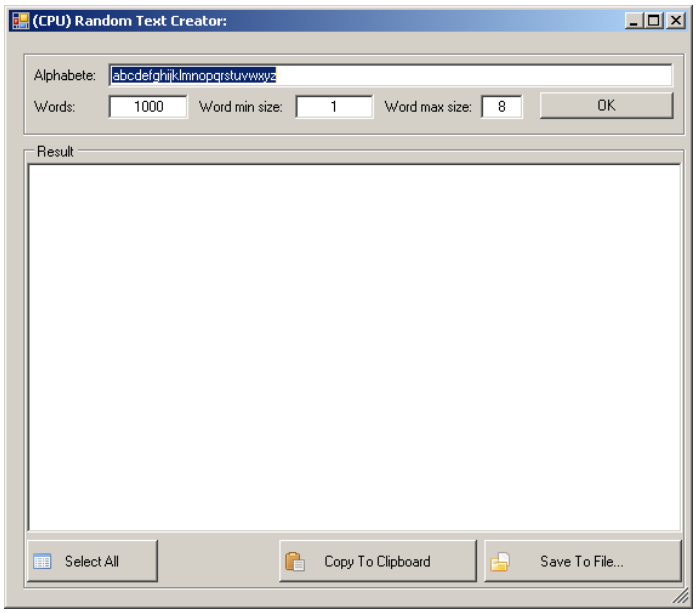


Obrázek 4 – Načíst lze i všechny texty ze složek podle zadaných kritérií

- Maximum file size … – specifikuje maximální velikost souboru v *bytech*, který ještě může být načten.
- Minimum file size … – specifikuje minimální velikost souboru v *bytech*, který ještě může být načten.
- Use only ONE file with similar size of … – specifikuje vynucený rozptyl velikosti v *bytech*. Při nastavení např. 2 000 *bytů* to znamená, že v případě, že by v určité složce bylo 1 000 souborů s náhodnými velikostmi, tak každá velikost ±2 000 bytů bude načtena pouze jednou. Tímto způsobem lze rychle a naivně zaručit, že vzorky textů načteným touto metodou budou rozdílně dlouhé.
- Maximum files count … – specifikuje maximální počet souborů, které mohou být načteny.
- Files containing in name … – specifikuje podřetězec, který musí být ve jméně souboru obsažen, aby byl načten.
- Prepend directory name – před jméno souboru přidá jméno adresáře, ve kterém se text nachází.
- Randomize dictionary file list – seznam souborů je (před testováním jednotlivých souborů na splnění výše zmíněných podmínek) nejprve promíchán. Tímto postupem je možné předejít načítání vždy stejných souborů na základě jejich abecedního výčtu
- Ignore binary files – nebude načítat a dekódovat binární soubory (viz dále).

Vytváření náhodných textů – Random Text Creator

QUITA obsahuje i několik menších nástrojů přístupných pomocí menu Tools v hlavním okně. Jedním z těchto nástrojů je i generátor (pseudo)náhodných textů, kterému stačí zadat počet slov, minimální a maximální velikost slova a abecedu. Náhodné texty jsou generovány pomocí funkce *Rand* obsažené v .NET Framework 3.5. Nejedná se tedy o ideálně náhodná data, viz jejich experimentální testování v kapitole 3. *Testování některých kvantitativně lingv. hypotéz*.



Obrázek 5 – Vytváření (pseudo)náhodného textu

Výsledný (pseudo)náhodný text je pak možné uložit do souboru klepnutím na tlačítko Save To File… a následně jej využít v QUITA.

Překódování binárních souborů

Počítače využívají binární soubory k uchovávání všech dat, avšak jen některá z nich lze smysluplně interpretovat jako přirozený text („textové soubory“). QUITA obsahuje nástroj, který libovolný binární soubor, ať už jde o soubor s obsahem hudby nebo o spustitelný soubor systému, dokáže překódovat na čitelný text (a případně zase zpět), a tím umožnit zkoumat i strojové informační struktury pomocí metod kvantitativní lingvistiky.

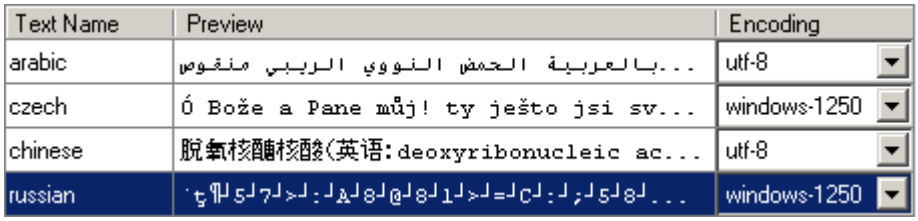
Podpora nukleotidových sekvencí

QUITA podporuje i práci s genovými sekvencemi – respektive s nukleotidovými řetězci, které jsou např. dostupné v genových bankách ve formátu FNA. Pro tento účel QUITA obsahuje i speciální tokenizátory *DNA Nucleotide Tokenizer* a *DNA Triplet Tokenizer*, které ze souborů FNA extrahují pouze nukleotidové řetězce a hlavičky a komentáře ignorují (viz *Tokenizátory* dále v této kapitole). Pomocí QUITA je tak možné experimentálně zkoumat DNA či RNA pomocí lingvisticko-kvantitativních metod.

2.7.2 Problematika kódování textu

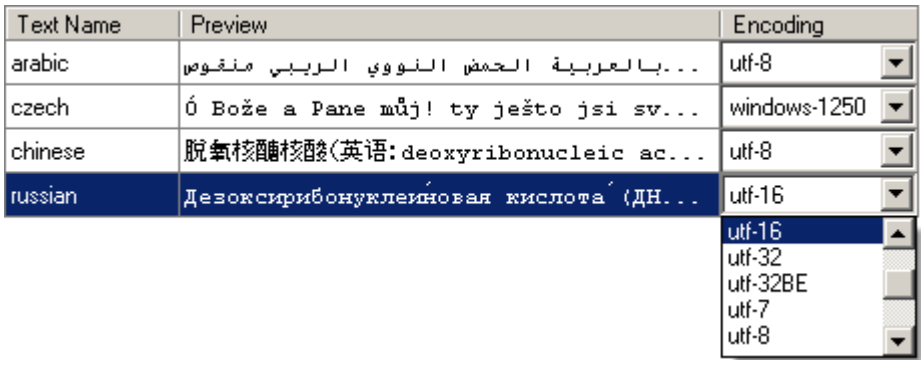
Každý soubor obsahující text v počítači nutně řeší problém, jakým způsobem uložené *byty* mají reprezentovat znaky. Jinak řečeno, jak všechny potřebné znaky daného jazyka přiřadit hodnotám 0 až 255 a jakým způsobem následně dát uživateli, který si chce text uložený v *bytech* přečíst, najevo, podle jaké tabulky může tyto *byty* dekódovat. Celý vývoj v této oblasti je poměrně složitý a zabral desítky let. Způsob ukládání *bytů* reprezentujících text je tedy pouze věcí dohody a norem, jejichž dodržování bylo, a stále je, věcí uživatele a partikulárních programů, které s texty pracují. Z tohoto důvodu je pak možné narazit na texty, které není možné zobrazit správně, čitelně, ve znacích, ve kterých je autor psal a pro jejichž správné zobrazení je nutné najít správnou *tabulku* (*encoding* či kódování), která text dokáže správně dekódovat. Celá problematika kódování je extrémně složitá: vše zde závisí na standardech, jejich různých verzích a samozřejmě i na tom, kdo dané standardy implementuje. Koherentní vysvětlení této problematiky by zabralo vlastní celou studii, proto se zde omezíme pouze na dopady, které na naši práci tato problematika má.

QUITA se snaží předcházet problémům s kódováním dvěma způsoby: (1) po načtení textu zobrazí okamžitě jeho náhled (*preview*), ve kterém je možné zkontrolovat, zda se text zobrazuje správně, a tedy, zda bylo pro jeho čtení použito správné kódování. Toto kódování je navíc možné pro každý text zvlášť pohodlně měnit pomocí rozbalovacího menu (umístěného po pravé straně) a tak i zároveň kontrolovat změny, které nově zvolené kódování vytváří. (2) QUITA se pokouší detekovat kódování pomocí několika heuristických metod, které často dokáží odhadnout použití některých běžných kódování. Tyto metody nejsou ideální (což plyne z celé výše popsané problematiky) a v některých případech selhávají, viz Obrázek 6 – *Chybné kódování textu* níže.



Obrázek 6 – Chybné kódování textu

Pomocí zmíněného rozbalovacího seznamu je však okamžitě možné (po kontrole náhledů) kódování změnit na libovolné jiné kódování, viz Obrázek 7 – Nalezení správného kódování textu níže:

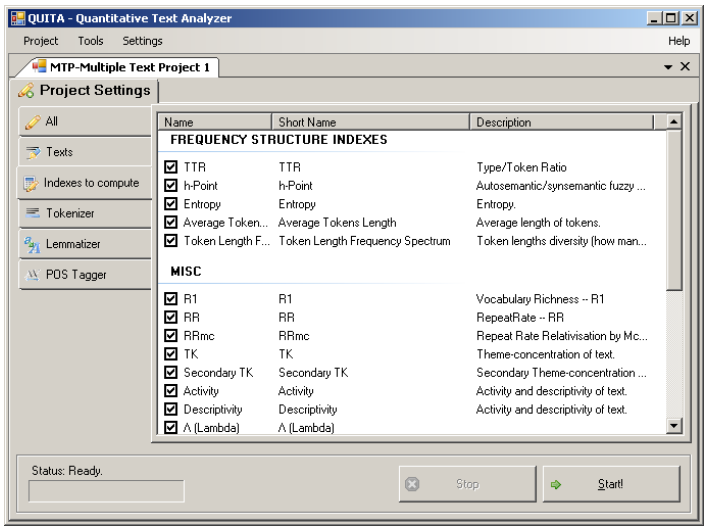


Obrázek 7 – Nalezení správného kódování textu

Nalezení správného kódování lze provést i „hrubou silou“, kdy je možné označit první kódování v seznamu a následně mačkáním *klávesy kurzorové šipky dolů* postupně procházet všechna dostupná kódování, dokud v náhledu není text zobrazen správně.

2.7.3 Indexy

Karta Indexy (*Indexes to compute*; viz Obrázek 8 – *Nastavení indexů*) nabízí přehled dostupných indexů, jejich popis a odkaz na literaturu, ve které je možné nalézt způsob výpočtu a další reference. Zaškrnuté indexy budou vypočítány, nezašrktnuté budou při zpracování ignorovány. Jejich nastavení lze upravit i po výpočtech a nechat tak dopočítat hodnoty pro další indexy. Pro menší počet textů je možné nechat všechny indexy zaškrnuté. Pro větší počty textů je vzhledem k paměťové náročnosti vhodné vybrat pouze ty indexy, které jsou nutné (více viz *Paměťová náročnost* výše v této kapitole).



Obrázek 8 – Nastavení indexů

2.7.4 Tokenizátory

Karta tokenizátor (*Tokenizer*) nabízí detailnější přehled všech dostupných tokenizátorů v QUITA. Interně jsou implementovány dvě základní metody tokenizace přirozeného textu (viz níže) a dvě speciální metody pro tokenizaci DNA:

- Default Generic Tokenizer nabízí běžnou naivní implementaci tokenizace pomocí regulárního výrazu: „\W+“. Za tokeny jsou považovány všechny alfanumerické řetězce. Např. větu „Karel IV. (1316 - 1378) byl 11. český král." tokenizuje na tokeny: „Karel“, „IV“, „1316“, „1378“, „byl“, „11“, „český“, „král“.
- Line Tokenizer považuje za token vše, co je na řádku. Tímto způsobem je možné do QUITA předat již, například ručně, tokenizovaný text v oblíbené formě tzv. vertikály.
- DNA Nucleotide Tokenizer považuje za token každý jednotlivý neukleotid.
- DNA Triplet Tokenizer považuje za token každé tři nukleotidy (tzv. triplety).

QUITA v předkládané verzi obsahuje na ukázku i několik tokenizátorů třetích stran. Zejména jde o ukázku spolupráce s nástrojovým balíkem NLTK (Natural Language Toolkit) dostupného na adrese http://www.nltk.org), která probíhá pomocí komunikace QUITA s Pythonovskými, Perlovskými a jinými druhy skriptů. Dále se jedná o tokenizátory webové služby Text-Processing.com. Do QUITA je dále možné přidávat podporu víceméně jakýchkoliv dalších tokenizátorů.

Poznámka k tokenizaci: QUITA má předem nastavenou ignoraci číselných tokenů, např. tokeny „1316“ a „11“ jsou ignorovány. Toto chování je možné vypnout pomocí menu Settings→Treat Numbers As Words. Obdobně jsou ignorovány i tokeny, které obsahující pouze interpunkci, mezery nebo jiné nealfanumerické znaky. Toto chování je také možné vypnout pomocí menu Settings→Treat non-alphanumeric characters as words. Ovšem to, zda se např. nealfanumerické znaky objeví jako tokeny, závisí na samotném výstupu lemmatizátoru.

2.7.5 Lemmatizátory

Karta Lemmatizátory (*Lemmatizer*) nabízí detailnější přehled všech dostupných lemmatizátorů. Nastavením lemmatizátoru na hodnotu „Nothing“ bude zpracovávaný text ponechán tak, jak byl zadán na vstup (tj., nebude lemmatizován žádným z dostupných nástrojů). Uživatel tedy může jako vstup použít již předem lemmatizované i tokenizované texty bez toho, aby je QUITA jakkoliv před samotnými výpočty zpracovávala. Na ukázku jsou do QUITA implementovány „přemostění“ několika lemmatizátorů z již zmíněného balíku NTLK. Pro češtinu je pak na ukázku vytvořena implementace zcela naivního přemostění s morfologickým analyzátorem MAJKA (Šmerk 2007; http://nlp.fi.muni.cz/ma/) a základní slovníková metoda založená na datech z ČNK (Křen et. al. 2014).

2.7.6 POS taggery

Karta POS taggery (*POS Tagger*) nabízí detailnější přehled všech dostupných POS taggerů. Některé z indexů nutně pro své výpočty vyžadují identifikaci slovních druhů – viz např. index *Tématické koncentrace* (Čech 2014, 17), kterou zprostředkovává právě POS tagger. V případě, že není POS tagger nastaven, mohou některé indexy namísto výsledku vypisovat chybovou hlášku, která je implicitně nastavena na hodno-

tu „[NO TAGGER]“. Stejně, jako u lemmatizátorů (viz výše) jsou pro ukázku implementovány přemostění s některými dalšími nástroji, především pak s POS taggery z balíku NLTK a pro češtinu opět s programem MAJKA a slovníkem ČNK.

2.7.7 Nastavení Cache

Použití tokenizátorů, lemmatizátorů a POS taggerů může být časově náročné. Pokud je po zpracování textů pomocí QUITA záměrem uživatele zobrazit např. tabulku frekvencí typů nebo pracovat se seznamem tokenů, jsou tato data načtena pouze z paměti a text tak není tokenizován/lemmatizován/taggován znovu. Pomocí uložení všech těchto výsledků v paměti cache je tedy umožněno se vyhnout časově náročným operacím. Nevýhodou tohoto přístupu je pak velké množství paměti, kterou všechna tato data vyžadují – při větším množství textů tak hrozí, že QUITA vyčerpá všechnu přidělenou paměť a nebude moci dále zpracovávat další data (viz výše Paměťová náročnost). Ukládání těchto rozsáhlých dat je možné vypnout právě v kartě Cache, a to buď úplně, nebo jen částečně pomocí následujících voleb:

- Disable Cache – vypne používání cache úplně. Tato možnost je nejvhodnější v případě práce s více texty, u kterých se primárně nepředpokládá vypisování nebo zobrazování seznamu tokenů, typů a zobrazování frekvenční tabulky, nebo je případné opakované zpracování tokenizací (a případně dále i lemmatizací, …) bez problémů možné.
- Enable Cache – zapne cache se zvolenými možnostmi:
    - Cache Tokens – „ukládat tabulku tokenů“ – tímto je možné se vyhnout opakované tokenizaci při prohlížení tabulky tokenů.
    - Cache Types – „ukládat tabulku typů“ – předchází opakované lemmatizaci při prohlížení tabulky typů.
    - Cache Frequency Table – „ukládat tabulku frekvencí typů“ – tímto je možné se vyhnout (1) náročnému počítání frekvencí, (2) v případě, že je vypnutá *cache typů* i případné lemmatizaci a (3) v případě vypnutí *cache tokenů* i tokenizaci.

Nastavení používání cache tedy závisí pouze na uživateli a na množství paměti, kterou může výpočtům obětovat. Obecně je však doporučeno cache používat jen pro malé množství textů.

2.7.8 Post Procesory

QUITA obsahuje základní post-procesory, které lze zatím v dosavadní verzi použít pouze exkluzivně (tzn. pouze jednu z nabízených možností najednou):

- N-Grams – vytváří z tokenů *n*-gramy.
- Reducing – redukuje počet tokenů textu na zadaných *n*. Tím je například umožněno zkoumat bez problémů i indexy, které jsou závislé na délce textu. (Tato redukce je nutně zařazena až do post-processingu kvůli faktu, že dokud není text tokenizován, tak jej není možné zkracovat na zadaný počet tokenů.)

2.7.9 Tokenizace, lemmatizace a POS tagging v QUITA

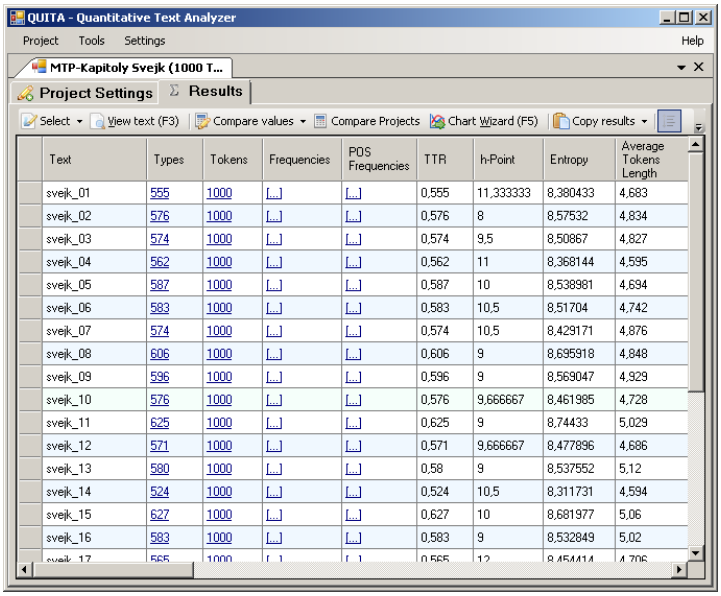
Používání tokenizátorů, lemmatizátorů a POS taggerů pomocí QUITA má v aktuálně předkládané verzi svá určitá úskalí a omezení. Tato omezení plynou nejen ze samotné implementace QUITA, ale i z implementací těchto jednotlivých nástrojů. (1) Tokenizátory nejsou dostupné pro všechny jazyky a ne všechny dostupné tokenizátory zatím mají v QUITA vlastní implementaci „přemostění“, které by s těmito nástroji dovolovalo pracovat. Tento problém je, jak již bylo řečeno výše, jednoduše překonatelný použitím již tokenizovaného textu ve formě vertikály a jeho načtením pomocí *Generic Line Tokenizer*, který za token identifikuje vše na daném řádku. Zcela stejný problém je pak s lemmatizátory – jejich absenci v QUITA nebo případně i jejich nedostatečnou přesnost, lze řešit jednoduše tím, že je použit již předem tokenizovaný a předem lemmatizovaný text ve vertikále – pouze je nutné vypnout lemmatizaci v QUITA. Lemmatizace a POS tagging mají dále v aktuální verzi QUITA pouze základní (naivní) implementaci. Ta byla navržena na základě dosud testovaných lemmatizátorů a POS taggerů, které na svůj vstup vyžadovaly pouze dotazované slovo bez libovolného kontextu. Tento naivní přístup však způsobuje nepřesnosti založené na nemožnosti dotazované slovo desambiguovat, a to ať už jakýmkoliv způsobem.

2.7.10 Spuštění výpočtů

Po nastavení je možné spustit výpočty okamžitě tlačítkem Start.

2.8 Výsledky výpočtů – Results

Výsledky jednotlivých výpočtů jsou zobrazeny v nové kartě Results – návrat zpět do nastavení projektu je možný jednoduchým klepnutím na kartu Project Settings. V kartě Results jsou zobrazeny výsledky výpočtů všech indexů a základní údaje o textu (jakými jsou např. počet tokenů, počet typů a tabulka jejich frekvencí). Některé výsledky, jak je možné vidět na Obrázek 9 – *Okno s výsledky výpočtů*, jsou podtržené a zabarvené modře. Takové výsledky, kromě zobrazené hodnoty v tabulce, obsahují další – rozsáhlé, komplexní či jen další (alternativní) výsledky, které je možné dvojklikem zobrazit do nové karty. Jedná se především o tabulky, seznamy a jiné výpisy, jakými jsou seznamy všech typů, tokenů, tabulky frekvencí, ale např. u indexu *Tématické koncentrace* to jsou výpisy tématických slov a jejich tématických vah (viz Čech 2014, 17). Zobrazené hodnoty „[…]“ pouze indikují, že pro daný index či informaci neexistuje „jednoduchá“ reprezentativní hodnota.



Text	Types	Tokens	Frequencies	POS Frequencies	TTR	h-Point	Entropy	Average Tokens Length
svejk_01	555	1000	[...]	[...]	0,555	11,333333	8,380433	4,683
svejk_02	576	1000	[...]	[...]	0,576	8	8,57532	4,834
svejk_03	574	1000	[...]	[...]	0,574	9,5	8,50867	4,827
svejk_04	562	1000	[...]	[...]	0,562	11	8,368144	4,595
svejk_05	587	1000	[...]	[...]	0,587	10	8,538981	4,694
svejk_06	583	1000	[...]	[...]	0,583	10,5	8,51704	4,742
svejk_07	574	1000	[...]	[...]	0,574	10,5	8,429171	4,876
svejk_08	606	1000	[...]	[...]	0,606	9	8,695918	4,848
svejk_09	596	1000	[...]	[...]	0,596	9	8,569047	4,929
svejk_10	576	1000	[...]	[...]	0,576	9,666667	8,461985	4,728
svejk_11	625	1000	[...]	[...]	0,625	9	8,74433	5,029
svejk_12	571	1000	[...]	[...]	0,571	9,666667	8,477896	4,686
svejk_13	580	1000	[...]	[...]	0,58	9	8,537552	5,12
svejk_14	524	1000	[...]	[...]	0,524	10,5	8,311731	4,594
svejk_15	627	1000	[...]	[...]	0,627	10	8,681977	5,06
svejk_16	583	1000	[...]	[...]	0,583	9	8,532849	5,02

Obrázek 9 – Okno s výsledky výpočtů

Kromě zobrazení výsledků nabízí karta Results i různé další důležité nástroje dostupné pomocí menu. Zejména se jedná o nástroje pro tvorbu grafu (viz podkapitola *Vytváření grafu pomocí Chart Wizard* dále v textu); nástroj pro porovnávání výsledků v rámci projektu (viz podkapitola *Porovnávání výsledků indexů v rámci* projektu) a nástroj pro porovnávání výsledků napříč všemi projekty (viz podkapitola *Porovnávání výsledků indexů mezi projekty)*. K dispozici jsou dále užitečné nástroje pro export tabulky výsledků do souboru CSV a možnost zkopírovat celou tabulku „tak jak je“ do schránky a jednoduše ji vložit do libovolného tabulkového procesoru (viz Menu→Copy results). Dále zde můžeme najít nástroj pro hromadné označování řádků (Menu→Select) a nástroj pro zobrazení samotného textu (Menu→View Text).

2.8.1 Okno s komplexními výsledky

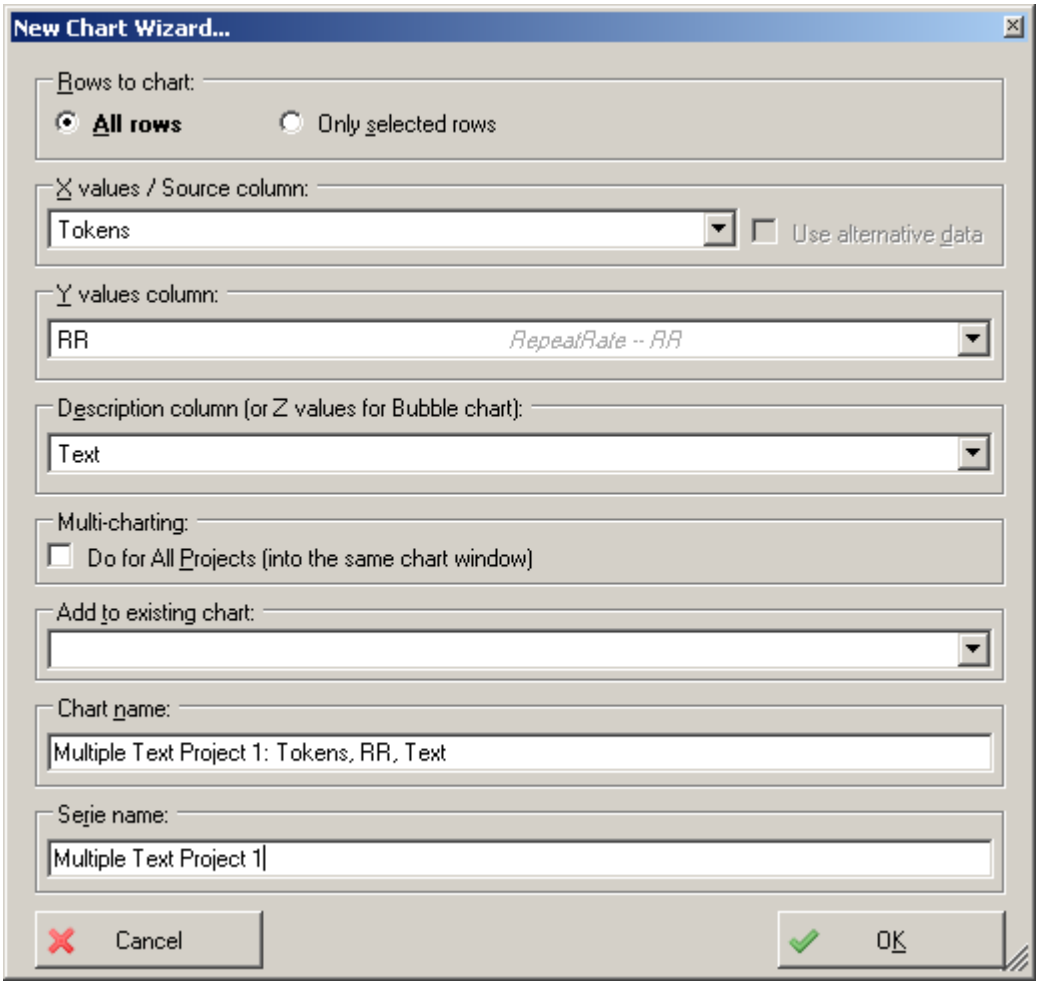
Se zobrazenými komplexními výsledky je možné pracovat běžným způsobem. Především je možné tabulku kopírovat do schránky nebo ji exportovat do souboru CSV. Návrat zpět do výpisu všech výsledků je možné provést buď zavřením nově otevřené karty, nebo pouhým překliknutím do karty Results.

2.8.2 Vytváření grafu pomocí Chart Wizard

QUITA umožňuje ze získaných výsledků v daném projektu vytvořit graf víceméně libovolných závislostí. Dialog pro tvorbu nového grafu je možné vyvolat klepnutím na tlačítko Menu→Chart Wizard v kartě Results. Postup nastavení parametrů nového grafu pak můžeme rozdělit do tří hlavních kroků:

1. Nastavení zdroje dat pro osu *X*, osu *Y* a zdroje dat pro popisku / případně osu *Z*.
2. Nastavení, zda přidat graf do již existujícího okna s grafem, nebo vytvořit nové okno.
3. Pojmenování samotného grafu a pojmenování řad(y).

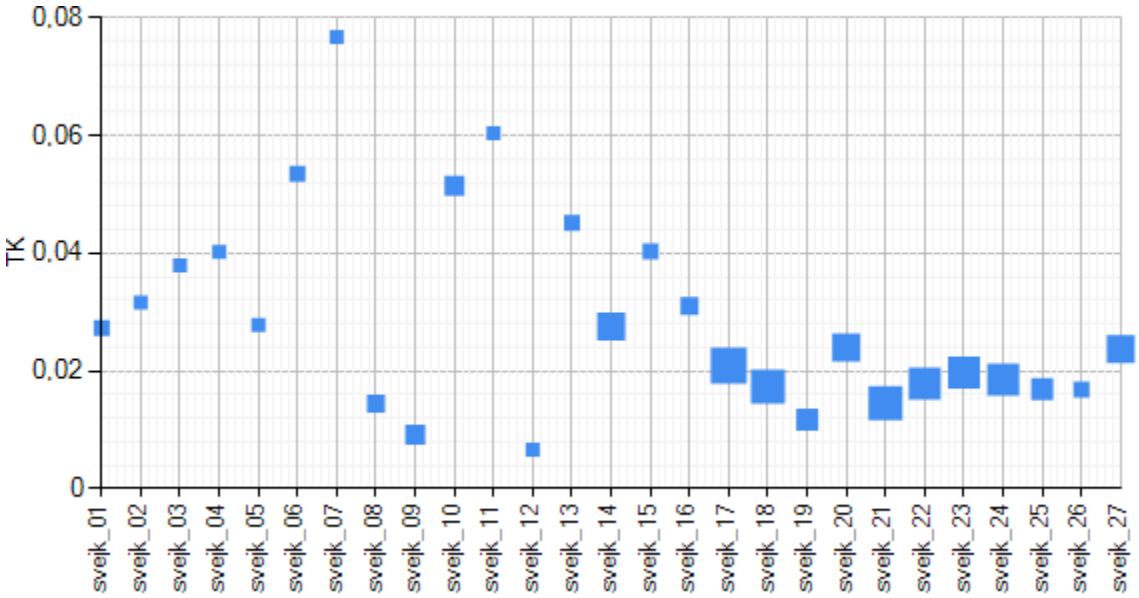
Zdrojem dat, ze kterých je graf vytvářen, jsou jednotlivé buňky tabulky výsledků – tj. všechny buňky daného sloupce (výsledky indexu) všech (zvolených) řádků (reprezentujících texty). Je tedy možné např. vykreslit závislost indexu *Repeat Rate* (RR; viz Čech 2014, 31) na množství tokenů (viz nastavení na Obrázek 10 – Vytváření grafu pomocí „New Chart Wizard“ níže), nebo nechat vykreslit postupně se měnící hodnoty indexu *Tématické koncentrace* pro každou kapitolu (text) zvlášť a sledovat tak, jak se *Tématická koncentrace* s vývojem díla mění.



Obrázek 10 – Vytváření grafu pomocí „New Chart Wizard“

Hodnotami pro osu *X* tedy mohou být i názvy samotných textů – pořadí hodnot na ose *X* pak bude odpovídat pořadí textů v tabulce. Osou *Y* může být jakýkoliv jiný sloupec

kromě sloupce *Text*. Hodnoty *popisky* jsou typicky přiřazovány právě ze sloupce *Text*, aby bod reprezentující výsledky určitého textu měl i jeho jméno. Avšak v případě, kdy je použit graf typu *Bubble*, je zdroj dat určený pro *popisku* použit jako zdroj dat pro osu *Z*, která je reflektována jako velikost jednotlivých vykreslovaných bodů – viz Obrázek 11 – Graf, ve kterém je osa *X* nastavena na *Text*, osa *Y* na hodnoty indexu *Tématické koncentrace* a osa *Z* na počet tokenů daného textu:

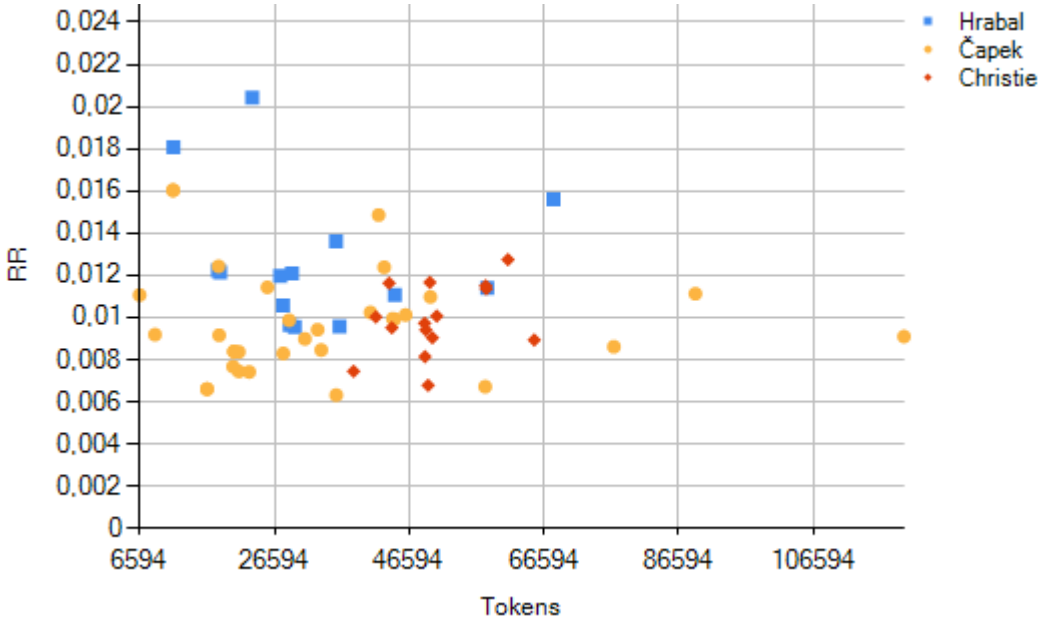


Obrázek 11 – Graf

Z takového grafu (Obrázek 11 – Graf) pak můžeme např. vyčíst, že kapitoly 17 až 27 jsou v počtu tokenů delší než kapitoly 1 až 16 a mají i stabilnější *tématickou koncentraci*.

Zdrojem dat pro vykreslování grafu mohou být i zmiňované komplexní (či alternativní) výsledky – tj. např. tabulky a pole, které mohou být výstupem některých indexů nebo jsou součástí základních informací o textu. V případě zvolení indexu/informace, která obsahuje komplexní výsledek, je nabídnuto pomocí zatrhávacího pole „Use alternative data“ použít za zdroj dat právě tato alternativní data. Všechny komplexní výsledky mají předem nastavené, které z jejich hodnot jsou určené pro osu *X*, *Y* a *Z*. Např. zvolením zdroje dat osy *X* za sloupec *TK* (index *Tématické koncentrace*), je možné vykreslovat hodnoty *TK* na osu *X*, a nebo, při zaškrtnutí „Use alternative data“, bude za zdroj dat pro osu *X* považován seznam *tématických slov* a za zdroj dat pro osu *Y* jejich *tématická váha*.

Při práci s více projekty je dále možné nechat do grafu zanést i data ze všech projektů zároveň pomocí zaškrnutí „Do for All Projects“. Data zvolených sloupců a řádků ze všech projektů budou do takového jediného grafu zanesena jako nové řady pojmenované podle jejich zdrojového projektu. Tímto způsobem je např. možné jednoduše srovnávat výsledné hodnoty děl různých autorů v jediném grafu, viz Obrázek 12 -- Více projektů v jednom grafu, který srovnává hodnoty indexu *Repeat Rate* v závislosti na délce textu tří různých autorů.



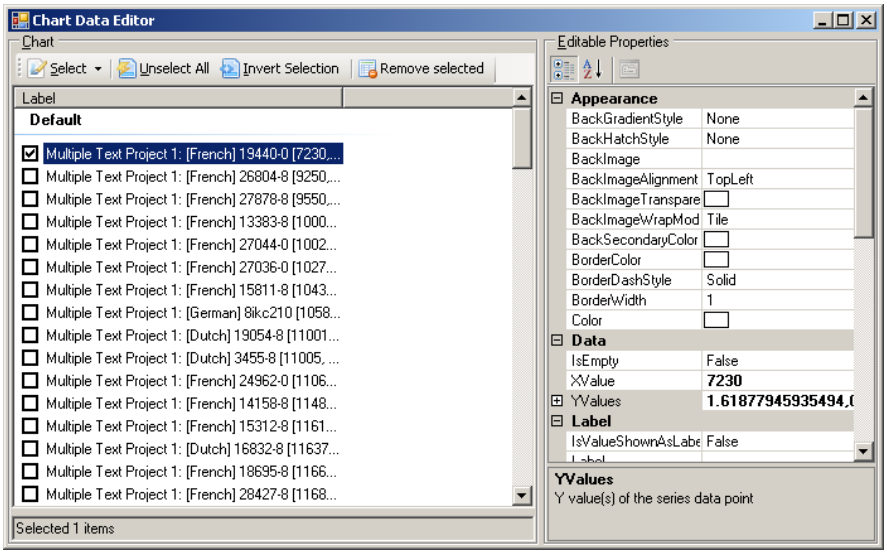
Obrázek 12 -- Více projektů v jednom grafu

Úpravy vlastností a vzhledu grafu

Typ grafu je možné nastavit pomocí menu grafu Chart→Chart Types, kde je na výběr mnoho různých typů zobrazení. Jednotlivé osy (a především pak jejich maximální a minimální hodnoty) lze editovat pomocí menu Chart→Edit X/Y Axis. Graf je také možné uložit do obrázku nebo jej přímo nakopírovat do schránky (Chart→Save to image file a Chart→Save to Clipboard). Pro kvalitnější výstup je pak možné využít tisk pomocí Chart→Print.

Editace bodů a řad

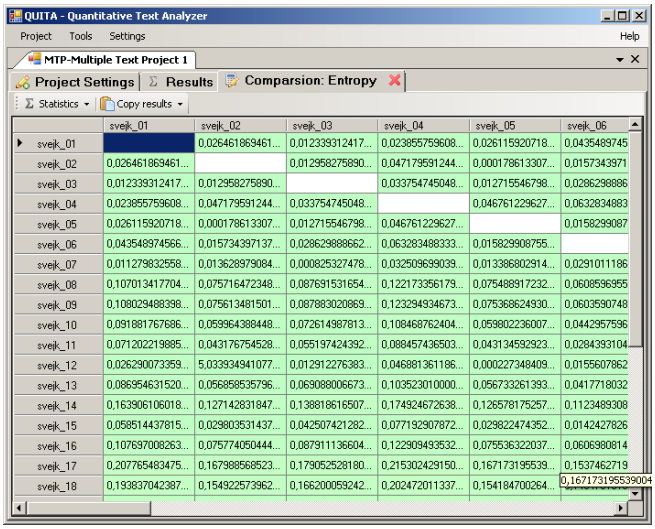
Jednotlivé zobrazené body a řady (případně i jejich skupiny) lze různě editovat pomocí *editoru bodů* a *editoru řad*. Tyto editory jsou přístupné přes menu grafu Chart→Edit points a Chart→Edit series nebo pouhým dvojklikem na daný bod / řadu v legendě. V editoru je možné upravovat vzhled jednotlivých bodů (jako barva, tvar), viditelnost, popisek ale i samotné hodnoty *X*, *Y* a *Z*.



Obrázek 13 – Úpravy jednotlivých bodů grafu

2.8.3 Porovnávání výsledků indexů v rámci projektu

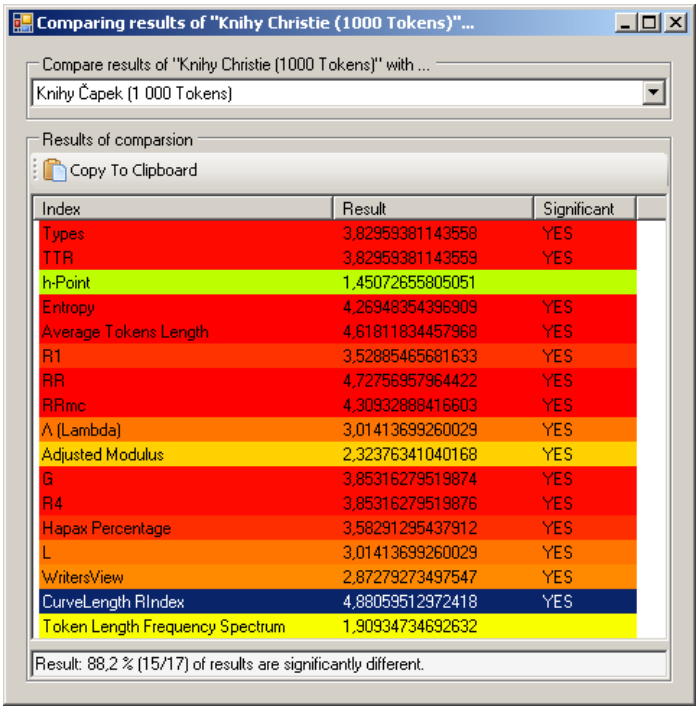
Některé indexy umožňují vypočítat rozptyl a následně své výsledky porovnávat pomocí metod specifikovaných přímo daným indexem. Porovnat všechny výsledky daného indexu v daném projektu je možné pomocí menu v kartě Results a dále pomocí rozbalovací nabídky Compare Values. Ta nabízí k porovnání všechny indexy, které mají porovnávání implementované.



Obrázek 14 – Porovnání výsledků

2.8.4 Porovnávání výsledků indexů mezi projekty

Porovnávat je možné také dva různé projekty mezi sebou pomocí tlačítka menu v kartě Results: Menu→Compare Projects. Mezi sebou jsou porovnávány všechny výsledky daného indexy pomocí *u*-testu. Tzn., že například všechny výsledky indexu *TK* projektu *X* jsou počítány jako jedna populace a všechny výsledky indexu *TK* projektu *Y* jako druhá populace. Na tyto dvě populace je pak aplikován zmíněný *u*-test.



Obrázek 15 – Porovnání výsledků mezi projekty

2.9 Poznámka k přesnosti

Reprezentace desetinných čísel ve výpočetní technice mají svá velká omezení, ze kterých následně mohou plynout i jisté nepřesnosti v samotných výpočtech. Zejména se jedná o výpočty s přesnými reálnými čísly, ve kterých může docházet k určitému zaokrouhlování. Ačkoliv je v celém programu QUITA použita nejpřesnější nativně dostupná varianta reprezentace reálných čísel, mohou být i výsledky QUITA, na základě této problematiky, částečně zkreslené.

Typicky je možné tuto problematiku ilustrovat na jednoduchém výpočtu `(1/3)*3`, který, při použití standardních reprezentací, vychází jako „1“, namísto 0,999… Pro ilustraci níže uvádím tabulku běžně používaných nástrojů pro výpočty a jejich výsledky:

Program/Jazyk	x=1/3	x=(1/3)*3
QUITA	0,333333333333333	1
Microsoft Excel 2010	0,333333333333333	1
OpenCalc 3.4.1	0,333333333333333	1
Perl 5.16	0,333333333333333	1
Perl 5.16 + BigFloat	0,999…	0,999…
Python 2.7.3	0,3333333333333333	1
Octave 3.6.4	0,333333333333333	1
R 3.1.0	0,33333333333333333148296	1

2.10 Základní příklady použití

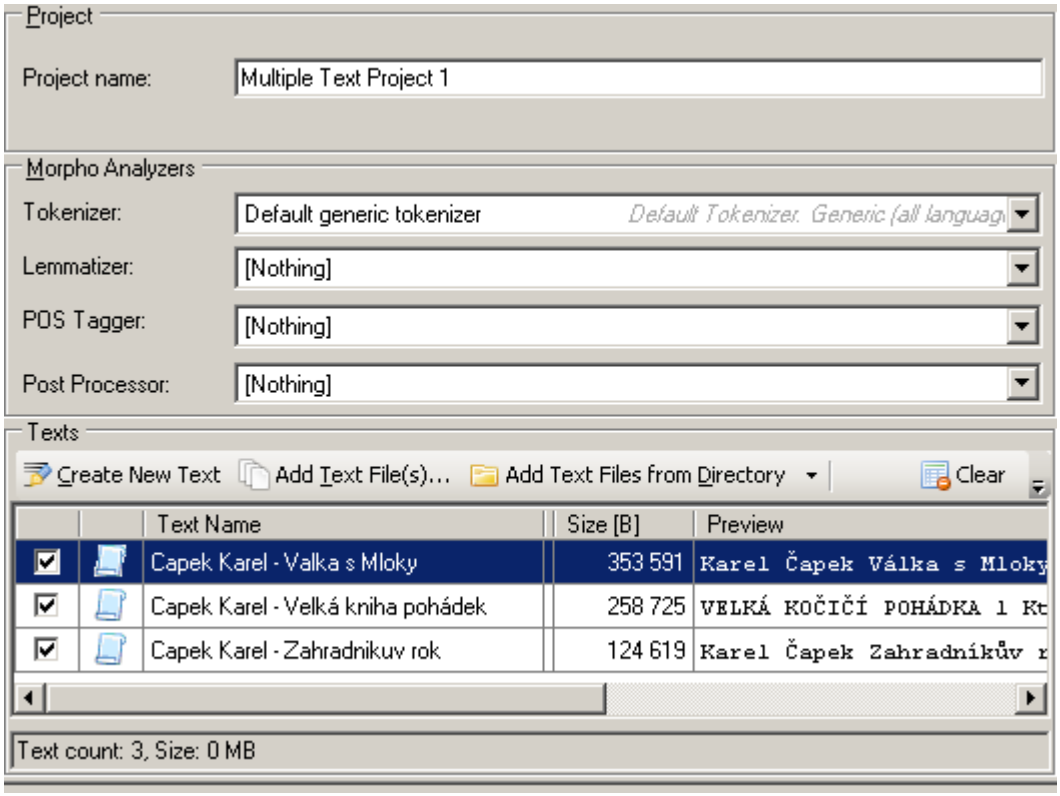
V této kapitole budeme ilustrovat použití QUITA způsobem „krok za krokem“ na několika základních úlohách, které mají za cíl ozřejmit veškeré důležité funkce tohoto programu v kontextu reálných požadavků kvantitativní lingvistiky.

2.10.1 Zjištění počtu tokenů, typů, jejich frekvencí a export do Excelu

Jedním ze zcela základních úkolů, se kterými se v kvantitativní lingvistice můžeme setkat, je zjištění počtu tokenů, počtu typů a jejich frekvencí pro několik zadaných textů uložených v TXT souborech a následně přenesení těchto dat do tabulkového procesoru (OpenCalc, Excel, …), který umožní s daty dále pracovat. V QUITA je tento úkol otázkou několika kliknutí myší:

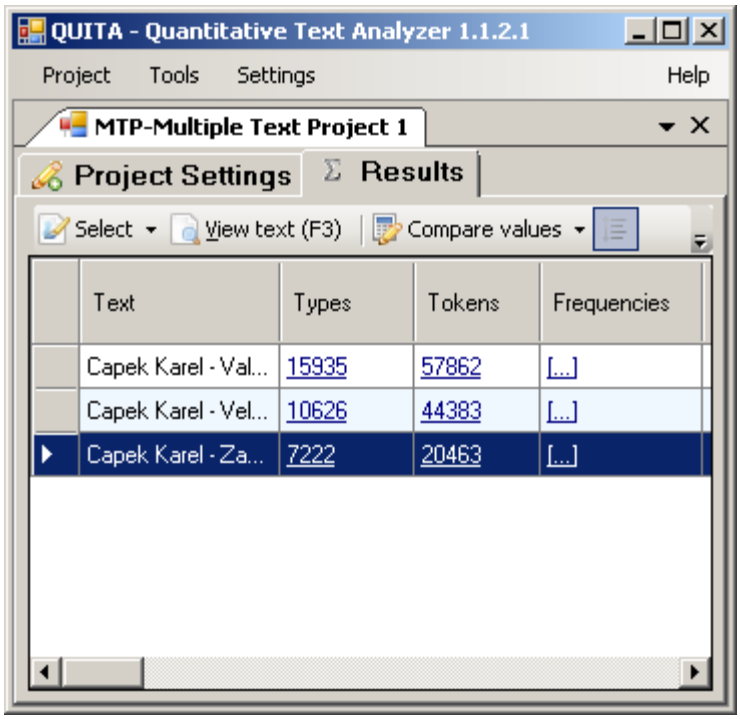
1. Nejprve je nutné spustit QUITA pomocí zástupce na ploše nebo kterýmkoliv jiným způsobem.
2. Pomocí menu nebo klávesovou zkratkou CTRL+N vytvoříme nový projekt.
3. Nastavení projektu je zcela na uvážení uživatele – pro tento první jednoduchý úkol není potřeba procházet celé nastavení a postačí tak k nastavení vše, co je zobrazeno na úvodní kartě „All“:
   3.1. Pojmenování projektu: Projekt není nutné pojmenovávat – stačí ponechat automaticky přidělené jméno. (Pojmenování projektů je vhodné až ve chvíli, kdy uživatel ví, že bude pracovat s více projekty zároveň a jejich jména budou sloužit k jejich jednoduchému rozlišení. Jméno projektu lze samozřejmě upravit i kdykoliv v průběhu práce s výsledky.)
   3.2. Tokenizace: Pro zběžnou tokenizaci češtiny postačí obecný tokenizátor obsažený v QUITA. Klepnutím do rozbalovacího seznamu tokenizátorů je možné zobrazit veškeré dostupné a funkční tokenizátory – zde zvolíme možnost *Default generic tokenizer*, který definuje token jako *cokoliv*, co je od jiného slova odděleno mezerou nebo jinými nealfanumerickými znaky.
   3.3. Lemmatizace: Zde závisí nastavení výhradně na očekávaném výsledku. Pokud má slovník (seznam typů) daného textu obsahovat veškeré slovní formy (tj. slova ve všech jeho tvarech) zvlášť, zůstane nastavení lemmatizátoru na možnosti „Nothing“, tj. „není nastaven žádný lemmatizátor“. Pokud je však potřeba ztotožnit všechny slovní druhy k jeho jediné formě, pak je nutné zvolit některý z dostupných lemmatizátorů klepnutím na rozbalovací nabídku, nebo případně použít již předem lemmatizovaný text.
   3.4. POS Tagger: Pro tento úkol nepotřebujeme zjišťovat slovní druhy, a tak může nastavení POS Taggeru zůstat na možnosti „Nothing.“
   3.5. Přidání textů ke zpracování: texty můžeme přidat tím, že je v prohlížeči „chytíme“ a přeneseme do tabulky Texts nebo pomocí dialogu Add text files.

   Výsledné nastavení projektu, samozřejmě s rozdílnými texty a případným nastavením lemmatizátoru, by tak mělo vypadat následovně (viz Obrázek 16 - Nastavení projektu):



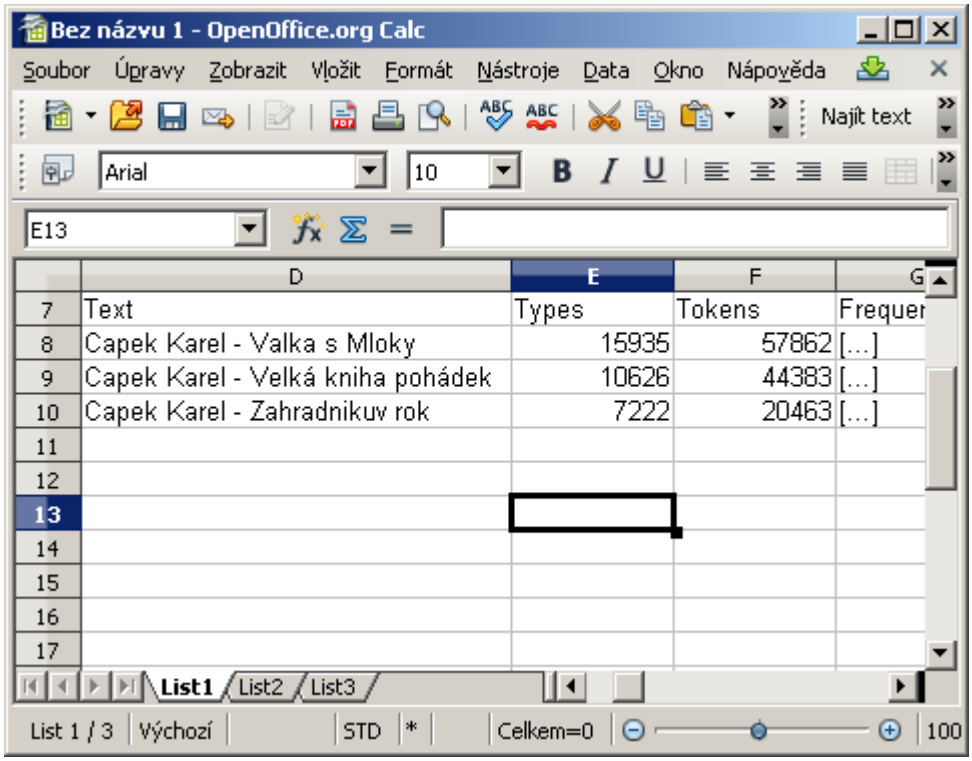
Obrázek 16 - Nastavení projektu

4. Nyní stačí spustit výpočty klepnutím na tlačítko Start, které je umístěno ve spodní části obrazovky.
5. Karta výsledků (Results), která by měla být následně zobrazena (viz Obrázek 17 – Karta výsledků) obsahuje všechny potřebné informace – počet tokenů, počet typů a frekvenční tabulku.



Obrázek 17 – Karta výsledků

5.1. Překopírování výsledků počtu tokenů a počtu typů do tabulkového procesoru: Celou tabulku výsledků jednoduše nakopírujeme do schránky pomocí menu Copy results→Copy Grid To Clipboard. Následně otevřeme tabulkový procesor a klávesovou zkratkou CTRL+V tabulku vložíme. Výsledek by měl být obdobný jako na Obrázek 18 - Exportovaná data v OpenCalc (viz níže):

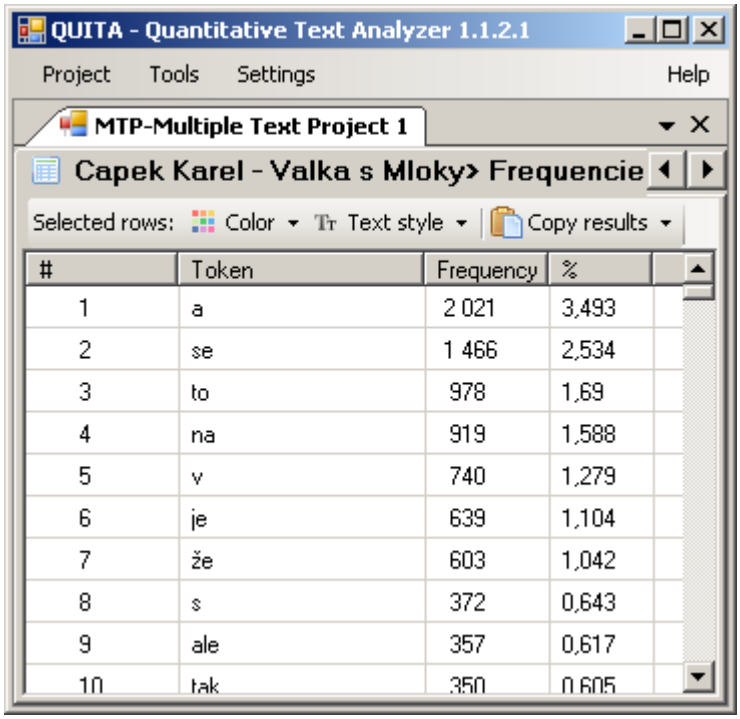


Obrázek 18 - Exportovaná data v OpenCalc

5.2. Zobrazení tabulky frekvencí typů pro jednotlivé texty a jejich nakopírování do tabulkového procesoru: Jedná se o obdobný postup jako kopírování celé tabulky výsledků výše, avšak s tím rozdílem, že v kartě výsledků Results nejprve zobrazíme tabulku frekvencí prvního textu tím, že dvakrát myší poklepeme na odkaz „[…]“ ve sloupci „Frequencies“. Tímto se otevře nová karta (záložka) hned vedle karty Results (viz Obrázek 19 níže). Zde následně stačí pomocí menu Copy results→Copy grid to Clipboard zkopírovat celou tabulku a pomocí klávesové zkratky CTRL+V ji vložit do tabulkového procesoru. Tímto postupem je pak možné kopírovat tabulky frekvencí všech textů.

6. Ukončit program.

Tento zcela základní příklad ilustroval nastavení projektu a kopírování výsledků výpočtů prakticky do libovolného jiného programu. QUITA však umožňuje s výsledky dále pracovat – viz následující úloha.



Obrázek 19 - Tabulka frekvencí

2.10.2 Sledování charakteristik kapitol

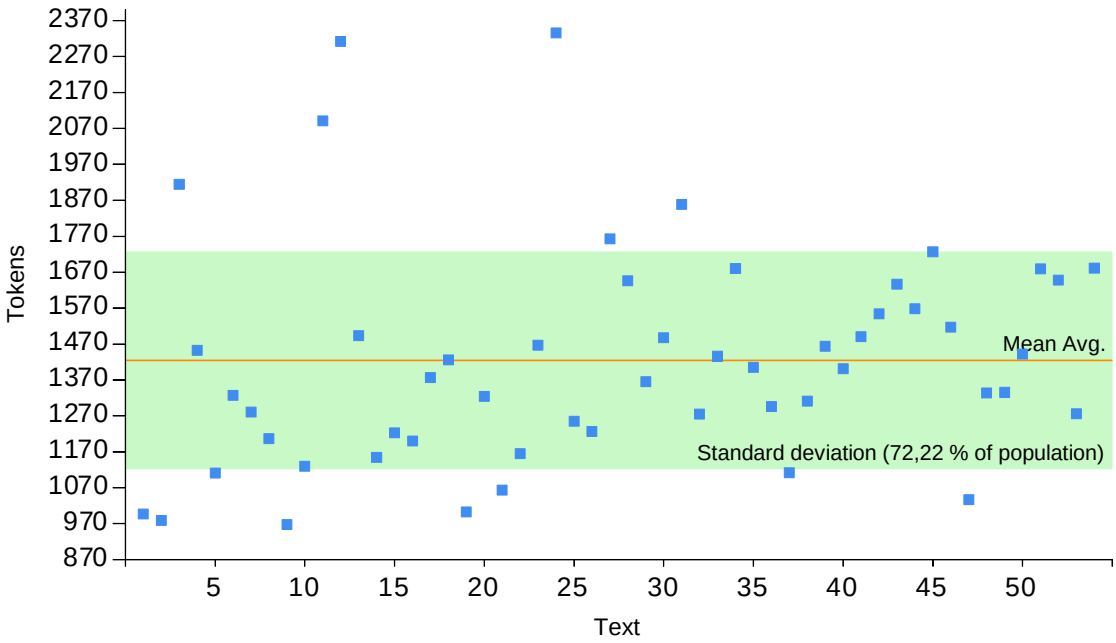
Následující úlohou se seznámíme s tím, jak se dá s výsledky v QUITA dále pracovat a jakým způsobem nám QUITA pomůže s vyhodnocováním některých statistických problémů. Ilustrující úloha je: (1) zobrazit průběh velikostí kapitol knihy *Krakatit* Karla Čapka a zjistit, jaké je množství kapitol, jejichž délka se pohybuje v rámci standardní odchylky. (2) Zjistit, jak se vyvíjí *Tématická koncentrace* po jednotlivých kapitolách a (3) graficky porovnat průběh vývoje *deskriptivity* kapitol s její nejmenší a nejvyšší hodnotou (viz Čech 2014, 55).

1. Spustíme program a vytvoříme nový projekt.
2. Nastavení projektu: v této chvíli je nutné si uvědomit, co vše má být zpracováno. První úloha vyžaduje pouze tokenizaci. Druhá úloha vyžaduje výpočet *Tématické koncentrace*. Ta nutně vyžaduje identifikaci *tématických slov* nad *h*-bodem – tedy rozpoznání slovních druhů a nejlépe i lemmatizovaný text (Čech 2014, 17). Třetí úloha rovněž vyžaduje rozpoznání slovních druhů (Čech 2014, 71). Nastavení nástrojů bude vypadat následovně: Tokenizátor nastavený na *Default generic tokenizer*, lemmatizátor *Majka+Corpus*, a POS tagger také *Majka+Corpus*.
3. Do textů ke zpracování přidáme kapitoly knihy *Krakatit* (které je možné nalézt na přiloženém CD, viz Obsah přiloženého CD). Po načtení textů zkontrolujeme náhled na jejich obsah ve sloupci *Preview*, který nám řekne, zda je text správně dekódován pomocí kódování UTF-8.
4. Spustíme výpočty klepnutím na tlačítko Start a podíváme se na výsledky, které by měly být shodné s tabulkou níže:

Text	Tokens	TK	Descriptivity
1__I	997	0,053457	0,148325
2__II	979	0,022222	0,211823
3__III	1915	0,020918	0,293249
4__IV	1453	0,029273	0,271003
5__V	1111	0,024427	0,282895
6__VI	1327	0,055314	0,283912
7__VII	1281	0,030075	0,317568
8__VIII	1207	0,014104	0,219858
9__IX	968	0,005471	0,484733
10__X	1130	0,064637	0,258865
11__XI	2091	0,016974	0,307054
12__XII	2312	0,030952	0,329372
13__XIII	1493	0,010585	0,357771
14__XIV	1155	0,021964	0,287879
15__XV	1223	0,026965	0,442379
16__XVI	1200	0,062937	0,297794
17__XVII	1376	0,083255	0,226537
18__XVIII	1426	0,056102	0,283388
19_XIX	1002	0,087476	0,353211
20__XX	1325	0,089275	0,382263
21__XXI	1064	0,127946	0,296
22__XXII	1165	0,02005	0,378906
23__XXIII	1467	0,122495	0,307229
24__XXIV	2336	0,066493	0,26789
25__XXV	1255	0,007416	0,31962
26__XXVI	1227	0,104971	0,28
27__XXVII	1763	0,060543	0,316038
28__XXVIII	1646	0,067186	0,462585
29__XXIX	1366	0,026985	0,365217
30__XXX	1488	0,045355	0,361039
31__XXXI	1859	0,021902	0,266525
32__XXXII	1275	0,0279	0,320513
33__XXXIII	1436	0,030862	0,219178
34__XXXIV	1680	0,038462	0,320988
35__XXXV	1405	0	0,235119
36__XXXVI	1296	0,026659	0,2375
37__XXXVII	1112	0,075948	0,214286
38__XXXVIII	1311	0,03813	0,322581
39__XXXIX	1464	0,021426	0,355372
40__XL	1401	0,029594	0,342029
41__XLI	1490	0,022546	0,26158
42__XLII	1554	0,021073	0,329815
43__XLIII	1636	0	0,255495
44__XLIV	1569	0	0,241627
45__XLV	1727	0,002642	0,199095
46__XLVI	1517	0,020673	0,349296
47__XLVII	1037	0,00375	0,334802
48_XLVIII	1334	0	0,380697
49__XLIX	1335	0,0281	0,290141
50_L	1442	0,034001	0,438438
51_LI	1679	0	0,204793
52__LII	1648	0,028938	0,289044
53__LIII	1276	0,010312	0,285714
54__LIV	1681	0,038515	0,242857

I. Řešení první úlohy

1. První částí úlohy je nejprve zobrazit velikosti jednotlivých kapitol v grafu. Toho docílíme kliknutím na tlačítko *Chart Wizard* v menu. V dialogu nastavení nového grafu specifikujeme zobrazení jmen kapitol na osu *X* (tj. nastavíme na hodnotu *Text*) a počet tokenů na osu *Y* (hodnota *Tokens*). Ostatní hodnoty můžeme ponechat stejné a klepnutím na tlačítko OK graf vykreslíme. (Okno grafu je možné za jeho titulek chytit myší a kamkoliv jej přesunout či zvětšit.)
2. Dále zjistíme, jaké množství kapitol má *normální* velikost vzhledem k ostatním kapitolám. Od pohledu si např. můžeme povšimnout, že velikost kapitol zřídkakdy přesáhne 2000 tokenů a stejně tak zřídkakdy klesne pod 1000 tokenů. Abychom mohli na výše položenou otázku odpovědět s určitou mírou exaktnosti, musíme aplikovat jednoduchý statistický test: Zjistíme standardní odchylku a množství kapitol spadajících do jejího intervalu. Zobrazení grafu neposkytuje pouze možnost vizualizace dat, ale obsahuje i některé základní statistické funkce, mezi kterými je výpočet průměru všech hodnot a výpočet standardní odchylky společně se zjištěním velikosti populace, která do této odchylky spadá. Pomocí menu Statistics nejprve odznačíme „Data is SAMPLE population“ (protože pracujeme s celou populací) a následně zobrazíme standardní odchylku pomocí Statistics->Show Mean Average + Standard deviation. Výsledek viz níže Obrázek 20 - Standardní odchylka délek kapitol:

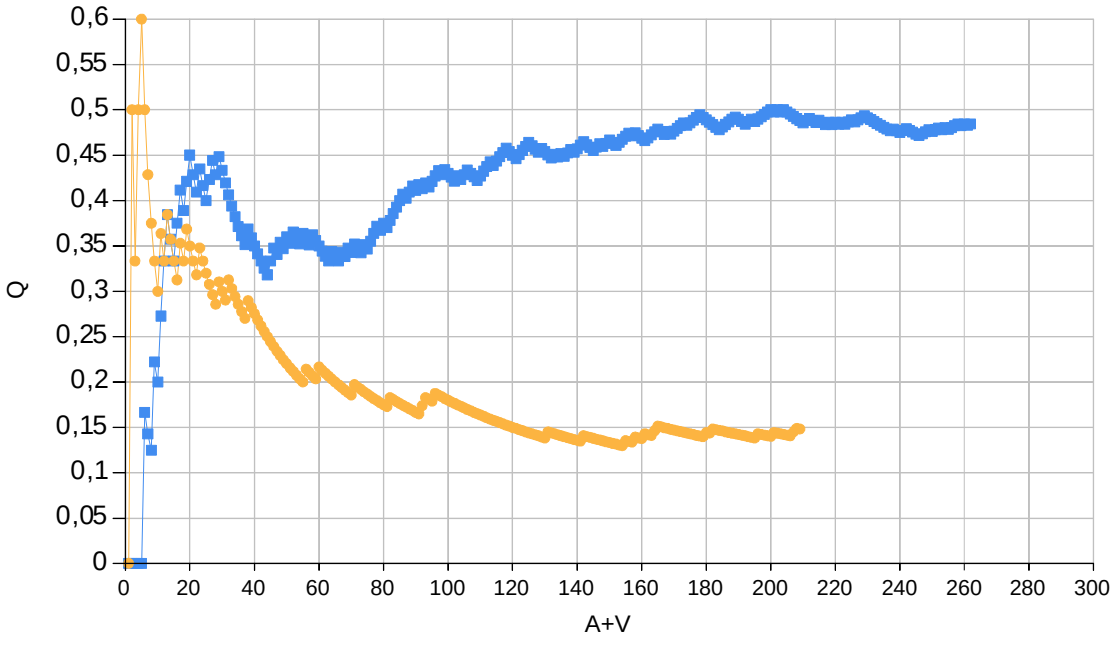


Obrázek 20 - Standardní odchylka délek kapitol

Můžeme tedy odpovědět, že 72,22 % kapitol knihy *Krakatit* leží svou velikostí v počtu tokenů ve standardní odchylce.

II. Druhý úkol je obdobný tomu prvnímu, respektive jde znovu o vytvoření grafu. Kliknutím na Chart Wizard nakonfigurujeme nový graf. Zdroj dat pro osu *X* je opět hodnota *Text* a zdroj dat pro osu *Y* budou nyní hodnoty *Tématické koncentrace* (*TK*).

III. Třetím úkolem je zobrazit průběh vývoje *deskriptivity* dvou kapitol (které mají nejnižší a nejvyšší hodnotu deskriptivity) do jednoho grafu tak, aby se tyto dva průběhy daly vizuálně srovnávat. Index *deskriptivity* (*Descriptivity*) obsahuje záznam (tabulku) o průběhu svého vývoje jako svůj alternativní výsledek, který lze zobrazit dvojklikem na číselný výsledek v kartě Results. Opětovně půjde o vytvoření grafu, ale tentokrát bude příprava složitější. Nejprve musíme v tabulce výsledků označit texty s nejmenší a nejvyšší *deskriptivitou*. Toho nejjednodušeji dosáhneme seřazením tabulky výsledků podle hodnot *deskriptivity* kliknutím na záhlaví sloupce *Descriptivity*. První a poslední řádek tabulky tak budou texty s extrémními hodnotami. Tyto řádky označíme držením klávesy CTRL a kliknutím myší. Dále otevřeme Chart Wizard, ve kterém specifikujeme zdroje dat: (1) data mají být čerpána pouze z označených řádků, tj. volba „Only selected rows“. (2) Jako zdroj dat pro „osu *X*“ zvolíme index *Descriptivity*, který ve svém komplexním (alternativním) výsledku obsahuje průběh vlastního vývoje – zaškrtneme tedy volbu „Use alternative data“. Nyní oba průběhy necháme vykreslit tlačítkem OK. Výsledkem je graf na Obrázek 21 - Průběh vývoje deskriptivity:



Obrázek 21 - Průběh vývoje deskriptivity

2.10.3 Porovnávání výsledků indexů

Výsledky jednotlivých indexů je často zajímavé porovnávat mezi sebou. Jejich porovnáním můžeme říci, zda jsou dané dva výsledky spíše stejné, nebo naopak signifikantně odlišné. QUITA takové porovnávání výsledků indexů umožňuje, včetně automatického vytvoření tabulky s výsledky porovnání daného indexu každého textu s každým. Zdlouhavé výpočty jednotlivých variancí a výpočty následných testů tak zcela odpadají. Následující úloha tedy bude ilustrovat, jak porovnávat výsledky indexů mezi sebou.

Příklad: Do všech kapitol knihy *Krakatit* Karla Čapka byla přimíchána jedna kapitola z knihy *Povídání o pejskovi a kočičce* autora Josefa Čapka (jednotlivé kapitoly, náhodně pojmenované, naleznete na přiloženém CD, viz Obsah přiloženého CD). Úkolem je se pokusit kapitolu z knihy Josefa Čapka identifikovat, a to pouze na základě kvantitativních odlišností indexů *entropie* (viz Čech 2014, 34), *R1* (viz Čech 2014, 36), *RR* (viz Čech 2014, 31) a *Giniho koeficientu* (viz Čech 2014, 41).

1. Spustíme QUITA, vytvoříme nový projekt. Tokenizaci ponecháme standardní (*Default generic tokenizer)*, lemmatizér nastavíme na *Majka+Corpus*, POS tagger není k žádnému zmíněnému indexu potřeba. V kartě *Indexes to compute* je možné omezit výpočty pouze na zadané indexy. Vstupní soubory načteme z přiloženého CD a ověříme správnost kódování. Spustíme výpočty klepnutím na tlačítko Start.
2. Nyní se pokusíme najít kapitolu knihy *Povídání o pejskovi a kočičce* tím, že přijmeme následující hypotézu: Kapitoly románu *Krakatit* autora Karla Čapka budou mít signifikantně odlišené charakteristiky od knihy pro děti autora Josefa Čapka. Tato hypotéza tedy implikuje následující: hledáme takový text, který má nejvíce signifikantně odlišných výsledků zmíněných indexů. Nyní tedy stačí jednotlivé výsledky zadaných indexů vzájemně porovnat a u každého takto porovnaného indexu vybrat nejsignifikantněji odlišný text.
3. Porovnání výsledků určitého indexu provedeme jednoduše: kliknutím na rolovací menu Compare values vybereme index, jehož výsledky chceme porovnávat.

I. Porovnání *entropie* (*Entropy*): Výsledků porovnání „každý s každým“ je už při padesáti textech dost velké množství na to, aby bylo hledání nejodlišnějšího textu velmi náročné. QUITA nám však pomůže tím, že pro každý sloupec – tj. výsledky porovnání jednoho textu se všemi ostatními – uvede průměr těchto rozdílů v řádku *Average*. Dále stačí, abychom v tomto řádku nalezli nejvyšší hodnotu (tj. nejvyšší průměrnou odlišnost od všech ostatních textů). (To provedeme např. tak, že tabulku nakopírujeme do tabulkového procesoru, řádek *Average* označíme a pomocí funkce *MAX* v ní nalezneme maximální hodnotu.)

Tabulka průměrných výsledků porovnání indexu *entropie* pro jednotlivé texty:

Text	Avg	Text	Avg	Text	Avg	Text	Avg
1	4,055	17	2,948	33	2,782	49	3,312
2	2,815	18	2,879	34	5,799	**50**	**12,486**
3	2,991	19	3,559	35	4,329	51	5,703
4	3,291	20	2,698	36	4,766	52	3,127
5	2,719	21	2,809	37	2,957	53	3,095
6	3,787	22	4,812	38	3,088	54	3,756
7	4,579	23	5,782	39	2,847	55	2,714
8	2,743	24	2,676	40	2,842		
9	2,985	25	2,742	41	3,360		
10	5,165	26	3,328	42	3,786		
11	2,869	27	3,874	43	2,960		
12	5,528	28	3,421	44	3,356		
13	4,482	29	4,697	45	3,174		
14	4,536	30	2,731	46	2,859		
15	4,724	31	6,583	47	2,918		
16	3,210	32	6,173	48	3,346		

Nejvyšší průměrnou odlišností je hodnota 12,489, což je skoro dvojnásobek druhé nejvyšší hodnoty 6,582. Nejodlišnější od zbytku kapitol je zde tedy jednoznačně text č. 50.

II. Porovnání *R1*. Zcela stejným způsobem provedeme porovnání indexu *R1* s následujícími výsledky:

Text	Avg	Text	Avg	Text	Avg	Text	Avg
1	1,333	17	1,414	33	2,474	49	2,054
2	1,385	18	1,407	34	1,673	**50**	**4,696**
3	1,684	19	2,718	35	1,679	51	1,869
4	3,917	20	1,440	36	1,615	52	1,327
5	1,342	21	1,593	37	2,378	53	1,622
6	3,016	22	2,243	38	2,160	54	1,701
7	1,543	23	1,593	39	1,405	55	1,369
8	1,579	24	1,468	40	1,413		
9	1,370	25	1,430	41	1,888		
10	1,561	26	1,276	42	3,027		
11	2,396	27	1,671	43	1,490		
12	3,288	28	1,812	44	1,447		
13	1,656	29	2,588	45	1,344		
14	1,286	30	1,462	46	1,738		
15	3,500	31	1,555	47	2,661		
16	1,508	32	3,275	48	2,951		

Text č. 50 má opětovně největší odlišnost od ostatních textů. Druhou největší hodnotu má text č. 4, tj. zcela odlišný text od textu č. 31, který měl druhou největší hodnotu u *entropie*.

III. Porovnání *RR*. Znovu provedeme celý test pro *RR*:

Text	Avg	Text	Avg	Text	Avg	Text	Avg
1	1,703	17	1,410	33	2,089	49	1,653
2	1,335	18	1,511	34	2,305	**50**	**4,600**
3	1,682	19	1,668	35	1,880	51	2,243
4	1,802	20	1,566	36	1,395	52	1,349
5	1,408	21	1,446	37	1,857	53	2,088
6	2,366	22	1,956	38	1,735	54	1,636
7	1,757	23	1,806	39	1,547	55	1,554
8	1,473	24	1,243	40	1,254		
9	1,297	25	1,492	41	1,493		
10	1,520	26	1,512	42	2,713		
11	1,749	27	1,298	43	1,658		
12	2,897	28	1,413	44	1,319		
13	1,608	29	1,983	45	1,368		
14	1,516	30	1,284	46	1,524		
15	3,465	31	2,114	47	2,515		
16	1,456	32	2,756	48	2,540		

Zde je opět textem s největší odlišností text č. 50. Text č. 15 s druhou největší hodnotou se objevuje poprvé.

IV. Poslední porovnání hodnot *Giniho koeficientu*:

Text	Avg	Text	Avg	Text	Avg	Text	Avg
1	1,462	17	1,409	33	1,960	49	1,327
2	1,407	18	1,869	34	1,272	**50**	**6,220**
3	1,725	19	2,396	35	2,011	51	1,911
4	4,528	20	1,429	36	2,303	52	1,433
5	1,370	21	1,688	37	1,526	53	1,712
6	2,451	22	2,358	38	2,497	54	1,537
7	1,514	23	1,686	39	1,401	55	1,404
8	1,573	24	1,505	40	1,618		
9	1,784	25	1,555	41	2,644		
10	2,326	26	1,363	42	2,652		
11	1,405	27	1,369	43	1,628		
12	2,759	28	1,747	44	1,488		
13	1,348	29	2,571	45	1,635		
14	1,281	30	1,379	46	2,379		
15	2,553	31	1,445	47	1,794		
16	1,966	32	3,429	48	2,448		

I v posledním testu vyšel text č. 50 jako ten nejodlišnější ze všech 50 textů. Text č. 4 se objevuje již podruhé (viz *R1* výše).

V. Závěr: Text č. 50 byl u všech zkoumaných indexů právě tím nejodlišnějším. Text č. 4 byl odlišný pouze u poloviny testovaných indexů a texty č. 15 a 31 byly odlišné pouze u jednoho ze čtyř zkoumaných indexů. Touto jednoduchou a ilustrativní metodou můžeme odhadnout, že onou vloženou kapitolou je text č. 50. V kartě Results označíme text č. 50 a klepnutím na tlačítko Menu→View Text zobrazíme obsah textu. Zjistíme, že se opravdu jedná o kapitolu z knihy Josefa Čapka.

2.10.4 Porovnání výsledků mezi projekty

Posledním příkladem bude ilustrace, jakým způsobem lze v QUITA porovnávat výsledky jednotlivých indexů mezi více projekty. Úlohou zde bude určit množství signifikantně odlišných charakteristik (indexů) tří děl: *Továrna na absolutno* a *Krakatit* autora Karla Čapka a díla *Povídání o pejskovi a kočičce* Josefa Čapka. Za hypotézu, kterou následně budeme chtít potvrdit či vyvrátit, přijmeme tvrzení, že díla *Krakatit* a *Továrna na absolutno* si budou na základě kvantitativně lingvistických charakteristik mnohem bližší, nežli kterékoliv z těchto dvou děl s dílem *Povídání o pejskovi a kočičce*. (Jednotlivé kapitoly všech zmíněných děl je opět možné najít na přiloženém CD).

1. Spustíme QUITA a vytvoříme nový projekt. Vzhledem k tomu, že nyní budeme pracovat s více projekty, je vhodné, abychom jméno projektu specifikovali jeho účelem – tj. první projekt bude sloužit ke kalkulacím indexů jednotlivých kapitol díla *Povídání o pejskovi a kočičce*. Do pole *Project name* tedy zadáme „Povídání o pejskovi a kočičce“. Tokenizér nastavíme na *Default generic tokenizer*, lemmatizér na *Majka+Corpus* a POS tagger rovněž na *Majka+Corpus*. Následně načteme všechny kapitoly této knihy jako vstupní texty. Tlačítkem Start necháme zpracovat.
2. Vytvoříme další projekt (opět pomocí menu Project→New project), který bude sloužit pro výpočty díla *Krakatit*. Jméno projektu nastavíme na „Krakatit“, tokenizátor, lemmatizér a POS tagger nastavíme stejně jako u předchozího projektu a následně načteme shodný počet kapitol knihy *Krakatit* jako u *Povídání o pejskovi a kočičce*, tj. devět kapitol. Následně necháme zpracovat tlačítkem Start.
3. Stejným způsobem vytvoříme projekt pro dílo *Továrna na absolutno*, včetně načtení pouze devíti kapitol a necháme zpracovat.
4. V tuto chvíli jsou ve třech kartách tři různé projekty, které obsahují vypočítané indexy ke všem obsaženým kapitolám. Nyní chceme potvrdit či vyvrátit hypotézu, která říká, že díla *Továrna na absolutno* a *Krakatit* si budou na základě statistického testu blíže, nežli k *Povídání o pejskovi a kočičce* (ať už z jakéhokoliv důvodu). Takové porovnání mezi projekty provedeme následovně: překlikneme se do karty s projektem *Povídání o pejskovi a kočičce* a klepneme na tlačítko Compare projects. Výsledkem porovnání projektu *Povídání o pejskovi a kočičce* (PPK) s projekty *Krakatit* a *Továrna na absolutno* vznikne následující tabulka:

	PPK, Krakatit		PPK, Továrna na absolutno	
Index	**Result**	**Significant**	**Result**	**Significant**
Types	5,318479	YES	7,768044	YES
Tokens	0,265994		0,009972	
TTR	9,103242	YES	7,406123	YES
h-Point	5,657558	YES	3,585454	YES
Entropy	8,443079	YES	11,097101	YES
Average Tokens Length	5,026768	YES	6,880609	YES
R1	7,458491	YES	8,061135	YES
RR	7,496099	YES	9,068117	YES
RRmc	6,304416	YES	7,323084	YES
TK	4,327516	YES	0,996792	
STC	4,934779	YES	0,306097	
Activity	1,960774	YES	4,059829	YES
Descriptivity	1,960774	YES	4,059829	YES
Λ (Lambda)	8,962413	YES	7,761355	YES
Adjusted Modulus	7,825307	YES	7,584533	YES
G	8,013538	YES	6,700607	YES
R4	8,013538	YES	6,700607	YES
Hapax Percentage	8,228711	YES	7,074024	YES
L	4,283438	YES	5,858203	YES
WritersView	0,999624		1,396998	
CurveLength RIndex	8,194472	YES	7,790862	YES
Verb Distances	0,788336		2,069498	YES
Token Length FS	1,562595		3,664099	YES
Significantly different	82,6 %		82,6 %	

Můžeme tedy říci, že projekt *Povídání o pejskovi a kočičce* je značně rozdílný vůči oběma dalším projektům. Dále musíme porovnat projekt *Krakatit* s projektem *Továrna na absolutno*. Zavřeme zobrazené okno porovnání projektu *Povídání o pejskovi a kočičce* a překlepneme se do karty projektu *Továrna na absolutno*. Otevřeme porovnání projektů pomocí Compare projects a ze seznamu vybereme porovnání s projektem *Krakatit*. Výsledkem porovnání projektů *Továrna na absolutno* a *Krakatit* je tabulka:

Index	Result	Significant
Types	0,115453	
Tokens	0,221792	
TTR	0,301061	
h-Point	0,849132	
Entropy	0,748041	
Average Tokens Length	2,56123	YES
R1	1,509127	
RR	2,21373	YES
RRmc	2,374761	YES
TK	2,47504	YES
STC	3,24209	YES
Activity	1,45753	
Descriptivity	1,45753	
Λ (Lambda)	0,595629	
Adjusted Modulus	0,5704	
G	0,038783	
R4	0,038783	
Hapax Percentage	0,545673	
L	0,255962	
WritersView	1,998852	YES
CurveLength RIndex	1,114085	
Verb Distances	0,702841	
Token Length FS	1,257942	
Significantly different	26,1 %	

5. Na základě výsledků z porovnání projektů můžeme potvrdit původní hypotézu, že *Továrna na absolutno* a *Krakatit* k sobě mají na základě svých charakteristik vzájemně mnohem blíže, nežli k dílu *Povídání o pejskovi a kočičce*.

Veškeré důležité funkce QUITA tímto byly demonstrovány. Je důležité znovu zdůraznit, že výše uvedené příklady mají pouze ilustrativní charakter toho, jak se samotným programem pracovat a jak si pomocí něj co nejvíce usnadnit práci. V následující kapitole se budeme věnovat některým obecnějším kvantitativně lingvistickým problémům, které se budeme snažit pomocí QUITA prozkoumat.

3 Testování některých kvantitativně lingv. hypotéz

V této kapitole se pomocí QUITA pokusíme odpovědět na některé otázky, které z implementace QUITA vyvstávají, a také se pokusíme prozkoumat lingvistickou hypotézu o zlatém řezu.

3.1 QUITA Random Text Creator vs. české texty

QUITA poskytuje možnost vytvářet náhodné texty a právě následující podkapitola má za cíl porovnat entropii takto vytvářených textů s různými českými nelemmatizovanými i lemmatizovanými texty.

Tabulka přirozených nelemmatizovaných českých textů, seřazeno podle počtu tokenů od nejmenšího po největší:

Text	N	Entropie	Text	N	Entropie	Text	N	Entropie
1	140	6,401495	**41**	28106	10,886808	**83**	67322	11,412691
2	462	7,889943	**43**	30505	11,740574	**81**	69677	10,49936
3	1020	8,75651	**42**	32743	10,643896	**82**	70119	10,946852
4	1543	8,632754	**46**	33324	11,072042	**84**	71616	10,985435
5	1673	9,139019	**44**	34345	10,526877	**85**	72187	11,048448
6	2792	9,342729	**45**	34978	10,718072	**86**	74825	10,899606
8	3274	10,366858	**49**	36103	11,233944	**87**	75748	10,874126
7	3570	9,021356	**47**	36124	10,674169	**89**	77744	10,828017
9	4514	9,580169	**48**	38635	10,630298	**88**	77837	10,876303
11	4720	9,800192	**52**	38893	11,017951	**90**	78813	11,334762
10	4829	9,566424	**51**	39253	10,903127	**93**	80429	11,174248
13	5375	10,369326	**50**	39999	10,900774	**96**	81195	11,767737
12	6050	9,778118	**56**	41535	10,915427	**91**	82740	11,002888
14	7044	10,585097	**53**	41672	11,062992	**94**	83514	11,005708
15	7128	10,506617	**55**	41715	11,099658	**92**	84883	10,840838
18	8208	10,305066	**58**	42448	11,17379	**95**	89426	11,284091
16	8442	10,401608	**54**	42887	11,086465	**97**	89939	11,223609
17	8569	10,4033	**57**	43176	10,824942	**99**	91168	11,503939
19	10874	10,152867	**59**	44685	10,645274	**98**	96242	10,957309
20	11386	10,483479	**60**	45584	11,052263	**101**	96575	11,263325
21	11505	10,511366	**61**	47726	10,612735	**102**	100071	11,119899
22	11974	10,626756	**62**	48299	10,676948	**100**	101080	11,01929
23	13024	10,423517	**63**	48369	10,815455	**103**	108051	10,975265
25	13567	10,387024	**64**	50287	10,501548	**105**	109983	11,116855
24	14563	10,119971	**65**	51067	10,808258	**104**	111087	11,17689
26	15958	10,373905	**66**	52056	10,993139	**106**	113999	11,538943
27	16595	11,266014	**67**	52490	10,897008	**115**	115907	11,720484
28	17500	10,814138	**72**	53447	11,790826	**107**	117215	11,244017
29	18777	10,633852	**68**	54048	10,527923	**108**	118299	11,127401
31	18936	10,926797	**69**	54646	10,791046	**109**	118978	11,312717
30	20098	10,00268	**70**	55787	11,316808	**111**	127267	11,243525
32	20800	10,442885	**71**	56644	11,042502	**112**	128025	11,391365
36	21061	11,080447	**74**	57390	11,311245	**110**	129572	10,481654
34	21441	10,459617	**73**	57760	11,211333	**116**	131235	11,257069
33	22161	10,326747	**76**	60093	11,427553	**117**	132711	10,845056
37	22975	11,113956	**75**	60349	10,775095	**113**	133508	11,161501
35	23189	10,199839	**77**	60445	11,285896	**114**	134290	10,691103
38	24350	10,668061	**80**	64124	10,722344	**118**	137104	11,615426
39	24649	10,576456	**79**	64660	10,973001	**120**	147776	11,072843
40	26260	10,946803	**78**	65786	10,680433	**119**	147791	11,303042

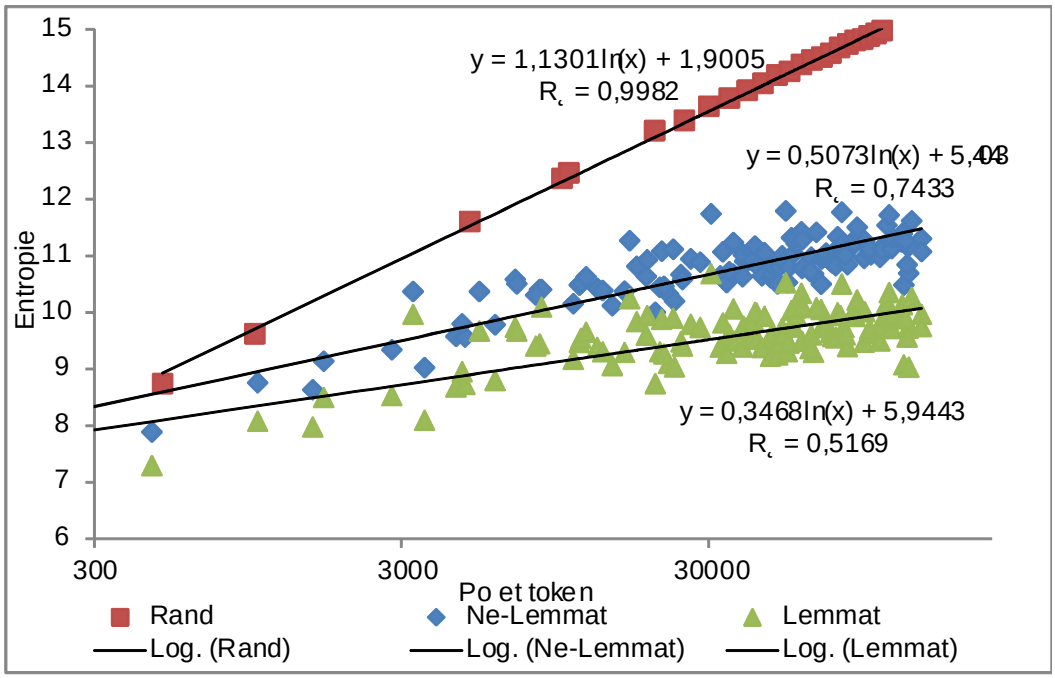
Tabulka lemmatizovaných českých textů, seřazeno podle počtu tokenů od nejmenšího po největší:

Text	N	Entropie	Text	N	Entropie	Text	N	Entropie
1	140	6,029488	**41**	28106	9,729379	**83**	67322	10,082137
2	462	7,288883	**43**	30505	10,677498	**81**	69677	10,047857
3	1020	8,075066	**42**	32743	9,384708	**82**	70119	9,738388
4	1543	7,973796	**46**	33324	9,811722	**84**	71616	9,729829
5	1673	8,493523	**44**	34345	9,27373	**85**	72187	9,753739
6	2792	8,526257	**45**	34978	9,607667	**86**	74825	9,641775
8	3274	9,962358	**49**	36103	10,059394	**87**	75748	9,571279
7	3570	8,090458	**47**	36124	9,492754	**89**	77744	9,550484
9	4514	8,680525	**48**	38635	9,38621	**88**	77837	9,555795
11	4720	8,951561	**52**	38893	9,752499	**90**	78813	9,987644
10	4829	8,723188	**51**	39253	9,725419	**93**	80429	9,936728
13	5375	9,669053	**50**	39999	9,697716	**96**	81195	10,49785
12	6050	8,790925	**56**	41535	9,551086	**91**	82740	9,731785
14	7044	9,73467	**53**	41672	9,872712	**94**	83514	9,601535
15	7128	9,665177	**55**	41715	9,838495	**92**	84883	9,389647
18	8208	9,400775	**58**	42448	9,996969	**95**	89426	10,052072
16	8442	9,443828	**54**	42887	9,986529	**97**	89939	9,922455
17	8569	10,089988	**57**	43176	9,520394	**99**	91168	10,217976
19	10874	9,163544	**59**	44685	9,373645	**98**	96242	9,470292
20	11386	9,462802	**60**	45584	9,930336	**101**	96575	10,005687
21	11505	9,536644	**61**	47726	9,220299	**102**	100071	9,703661
22	11974	9,646487	**62**	48299	9,538489	**100**	101080	9,628219
23	13024	9,37687	**63**	48369	9,641986	**103**	108051	9,490239
25	13567	9,299468	**64**	50287	9,243817	**105**	109983	9,765324
24	14563	9,059883	**65**	51067	9,450333	**104**	111087	9,897827
26	15958	9,291426	**66**	52056	9,71059	**106**	113999	10,173094
27	16595	10,241157	**67**	52490	9,61753	**115**	115907	10,347663
28	17500	9,833227	**72**	53447	10,52314	**107**	117215	9,899786
29	18777	9,593227	**68**	54048	9,30043	**108**	118299	9,710318
31	18936	9,939775	**69**	54646	9,455044	**109**	118978	9,963111
30	20098	8,733773	**70**	55787	10,029492	**111**	127267	9,886804
32	20800	9,299767	**71**	56644	9,743264	**112**	128025	10,132243
36	21061	9,870642	**74**	57390	10,071281	**110**	129572	9,067355
34	21441	9,263486	**73**	57760	9,936785	**116**	131235	9,880254
33	22161	9,092876	**76**	60093	10,33629	**117**	132711	9,563259
37	22975	9,893516	**75**	60349	9,488005	**113**	133508	9,718337
35	23189	9,036379	**77**	60445	10,106743	**114**	134290	9,031491
38	24350	9,476271	**80**	64124	9,356513	**118**	137104	10,260697
39	24649	9,403493	**79**	64660	9,602533	**120**	147776	9,747493
40	26260	9,789099	**78**	65786	9,291655	**119**	147791	9,962342

Tabulka entropie textů vytvořených pomocí QUITA s nastavením abecedy: „abcdefghijklmnopqrstuvwxyz“, velikost slov „1-8“:

QUITA Náhodné texty								
Text	**Tokens**	**Entropy**	**Text**	**Tokens**	**Entropy**	**Text**	**Tokens**	**Entropy**
1	500	8,743	**12**	50000	14,193	**23**	105000	14,928
2	1000	9,623	**13**	55000	14,256	**24**	110000	14,982
3	5000	11,604	**14**	60000	14,381	**25**	115000	15,045
4	10000	12,370	**15**	65000	14,463	**26**	120000	15,060
5	10500	12,465	**16**	70000	14,512	**27**	125000	15,095
6	20000	13,210	**17**	75000	14,579	**28**	130000	15,142
7	25000	13,394	**18**	80000	14,680	**29**	135000	15,199
8	30000	13,640	**19**	85000	14,743	**30**	140000	15,229
9	35000	13,788	**20**	90000	14,803	**31**	145000	15,276
10	40000	13,925	**21**	95000	14,829	**32**	150000	15,297
11	45000	14,047	**22**	100000	14,883			

Zobrazením závislosti *entropie* na *N* získáme následující graf, viz Obrázek 22 – Entropie textů níže:



Obrázek 22 – Entropie textů

Z grafu na Obrázek 22 – Entropie textů snadno nahlédneme, že entropie textů vytvářených pomocí QUITA má stále vyšší entropii nežli nelemmatizované texty.

3.2 Test naivní tokenizace a lemmatizace

Tokenizaci je možné provádět mnoha způsoby. Jedním z takových způsobů je i naivní metoda tokenizace pomocí regulárního výrazu \W+. Ten přijímá slova (respektive řetězy znaků) tvořené kterýmikoliv znaky kromě těch alfanumerických a znaku pro podtržítko – taková slova pak slouží jako oddělovače jednotlivých tokenů textu. Tato základní metoda tokenizace je implementována i v QUITA pod názvem *Default generic tokenizer*. Vzhledem k rozsáhlé problematice tokenizace, která byla popsána v úvodní kapitole této práce (viz 1.1 *Problematika tokenizace, lemmatizace a POS Taggingu*) vyvstává otázka, zda budou výsledky jednotlivých indexů signifikantně shodné u textů tokenizovaných výše popsanou naivní metodou s texty tokenizovanými specializovaným nástrojem. Tj. zda budou výsledky jednotlivých indexů stále přijatelné i při použití hrubé tokenizace založené na naivní metodě. QUITA dále disponuje i zcela základní, naivní a zcela ilustrační implementací lemmatizace češtiny, pojmenovanou jako *Majka+Corpus*. Ta je založena na dotazování se lemmatizátoru na lemma každého slova zvlášť bez jeho jakéhokoliv kontextu. Původní otázku položenou výše tak můžeme rozšířit i o lemmatizaci: Jaký bude rozdíl mezi výsledky jednotlivých indexů při tokenizaci a lemmatizaci pomocí výše zmíněných naivních metod v porovnání s výsledky týchž textů tokenizovaných a lemmatizovaných specializovaným nástrojem přímo určeným pro tuto činnost? Dále se proto budeme zabývat otázkou, jaké a jak velké rozdíly mezi oběma zpracováními jsou.

Rozdíly ve výsledcích indexů byly testovány na dvanácti různých textech (knihách a kapitolách různých knih a autorů). Ty byly zpracovány (1) pomocí naivních metod pomocí QUITA, tj. tokenizace pomocí *Default generic tokenizer* a lemmatizovány naivní lemmatizací *Majka+Corpus* (v tabulkách dále označeno jako metoda „QUITA“) a (2) zpracovány pomocí nástroje pro zpracování přirozeného jazyka TREEX (viz http://ufal.mff.cuni.cz/treex accessed 24. 4. 2014) dostupného rovněž jako webová služba (viz https://lindat.mff.cuni.cz/services/treex-web/#/ accessed 24. 4. 2014), a to konkrétně pomocí scénáře *Czech lemmatization* (v tabulkách dále jako metoda „TREEX“). Získané výsledky z obou zpracování jsou pro testované texty dále uvedeny v tabulkách paralelně vedle sebe pro lepší srovnání včetně vypočítaného *u*-testu (vzorec níže):

$$u = \frac{|\bar{X}_1 - \bar{X}_2|}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

Kde $\bar{X}_1$ a $\bar{X}_2$ jsou aritmetické průměry obou skupin výsledků; S_1 a S_2 standardní odchylkou dané skupiny; n_1 a n_2 počet výsledků dané skupiny. Testovány byly všechny indexy nabízené v QUITA, kromě těch, které vyžadují POS tagging a měření délky tokenů. Výsledky zpracování v tabulkách viz níže:

	Indexy					
	Tokens		Types		TTR	
Text	TREEX	QUITA	TREEX	QUITA	TREEX	QUITA
1	2331	2336	941	927	0,403689	0,396832
2	6593	6594	1678	1670	0,254512	0,253261
3	13953	13958	2488	2467	0,178313	0,176745
4	18335	18337	4393	4393	0,239596	0,23957
5	28118	28211	5181	5273	0,184259	0,186913
6	32964	33002	4494	4476	0,136331	0,135628
7	35774	35773	7357	7349	0,205652	0,205434
8	55042	56505	9691	9687	0,176066	0,171436
9	76868	76969	10165	10014	0,13224	0,130104
10	93268	93326	7276	7271	0,078012	0,07791
11	119956	120014	13234	13205	0,110324	0,110029
12	198670	198832	17124	17664	0,086193	0,088839
u-Test	0,007346		0,016226		0,030036	

	Indexy					
	h-Point		Entropy		R1	
Text	TREEX	QUITA	TREEX	QUITA	TREEX	QUITA
1	16,5	16	8,573307	8,567276	0,764103	0,767551
2	28	28	8,720112	8,735057	0,678902	0,68714
3	43	43	8,682218	8,694551	0,599405	0,602056
4	47,5	48	9,65619	9,658648	0,664474	0,664067
5	57	58	9,858057	9,882321	0,662369	0,662118
6	66,333333	68,5	9,2391	9,260268	0,582182	0,586211
7	62,5	63,5	10,25245	10,26392	0,671105	0,672186
8	80,666667	84,333333	10,19237	10,23558	0,62308	0,6243
9	88	89	9,957255	9,950687	0,577483	0,575407
10	110,4	114	9,339422	9,343381	0,510251	0,508679
11	111,25	114,5	9,956276	9,970432	0,539083	0,538722
12	150	149	10,06802	10,0749	0,516218	0,51325
u-Test	0,080912		0,049226		0,036154	

	Indexy					
	RR		RRmc		Lambda	
Text	**TREEX**	**QUITA**	**TREEX**	**QUITA**	**TREEX**	**QUITA**
1	0,009449	0,009467	0,933217	0,933359	1,496708	1,480065
2	0,011187	0,01106	0,916609	0,917282	1,156265	1,151577
3	0,014946	0,014022	0,895703	0,899699	1,069184	1,036046
4	0,00909	0,008901	0,918518	0,919528	1,226844	1,218706
5	0,006995	0,006784	0,929271	0,930446	0,984919	1,002878
6	0,010043	0,009418	0,913411	0,916654	0,831603	0,815558
7	0,006604	0,006314	0,929573	0,931401	1,110332	1,092541
8	0,007546	0,006884	0,922506	0,926442	1,042926	0,990112
9	0,008854	0,008606	0,914981	0,916389	0,86807	0,858004
10	0,009567	0,009056	0,912892	0,915576	0,648364	0,609057
11	0,009658	0,009092	0,909634	0,91259	0,800403	0,775628
12	0,008128	0,008086	0,916849	0,916975	0,671474	0,677663
u-Test	**0,435563**		**0,502868**		**0,173202**	

	Indexy					
	Adjusted Modulus		G		R4	
Text	TREEX	QUITA	TREEX	QUITA	TREEX	QUITA
1	17,063481	17,343444	0,534764	0,537187	0,465236	0,462813
2	16,037268	15,963799	0,666574	0,666844	0,333426	0,333156
3	15,412483	15,098932	0,73667	0,738361	0,26333	0,261639
4	22,186335	21,919762	0,690902	0,69184	0,309098	0,30816
5	20,897381	20,911633	0,726522	0,726313	0,273478	0,273687
6	15,983142	15,322756	0,778682	0,779996	0,221318	0,220004
7	26,345084	25,820093	0,71527	0,715836	0,28473	0,284164
8	26,184274	24,781078	0,755191	0,755241	0,244809	0,244759
9	25,084548	24,485303	0,790132	0,792021	0,209868	0,207979
10	16,12705	14,917423	0,839727	0,840541	0,160273	0,159459
11	25,581146	24,446493	0,815278	0,816506	0,184722	0,183494
12	23,914795	24,539165	0,843837	0,84465	0,156163	0,15535
u-Test	**0,251837**		**0,029255**		**0,029255**	

	Indexy					
	Hapax Percentage		L		WritersView	
Text	TREEX	QUITA	TREEX	QUITA	TREEX	QUITA
1	0,283569	0,273973	1036,016	1026,41	1,742097	1,730503
2	0,158046	0,157871	1996,095	1988,271	1,669792	1,669872
3	0,096538	0,096002	3599,402	3488,962	1,625422	1,628797
4	0,146878	0,147898	5276,263	5241,775	1,633434	1,636273
5	0,100541	0,103683	6224,78	6357,2	1,634655	1,633686
6	0,069348	0,069511	6067,443	5956,582	1,626575	1,631095
7	0,118354	0,118469	8723,056	8583,067	1,62365	1,629234
8	0,103049	0,096292	12108,93	11772,99	1,611733	1,619048
9	0,068533	0,06691	13657,45	13515,23	1,604169	1,60458
10	0,034256	0,034321	12167,98	11436,79	1,608274	1,613514
11	0,05482	0,055002	18903,86	18326,84	1,598531	1,601483
12	0,041511	0,044555	25179,03	25430,1	1,597979	1,598201
u-Test	0,034957		0,053988		0,111266	

	Index	
	CurveLength RIndex	
Text	TREEX	QUITA
1	0,897875	0,892639
2	0,832124	0,83117
3	0,684401	0,699813
4	0,827276	0,832771
5	0,827005	0,82418
6	0,734354	0,745027
7	0,839282	0,85183
8	0,796518	0,81881
9	0,740501	0,7371
10	0,592671	0,629955
11	0,696674	0,717085
12	0,676767	0,69136
u-Test	0,319376	

Průměrným výsledkem u-testu je hodnota 0,135091688. Nejvyšší hodnotou u-testu je 0,502868 (index *L*, viz Čech 2014, 39), nejnižší hodnotou u-testu je pak rozdílnost v počtu tokenů, a to 0,007346. Naivní implementace tokenizace i lemmatizace v porovnání se specializovaným tokenizátorem a lemmatizátorem tedy nezapříčiňuje signifikantně rozdílné výsledky jednotlivých testovaných indexů.

3.3 Zlatý řez

Článek *Quantitative Analysis of Italian texts* (Tuzzi, Popescu a Altmann 2010, 96) popisuje výpočet hodnoty α, která by dle tohoto článku měla s narůstajícím slovníkem konvergovat k tzv. zlatému řezu, tj. iracionální číselné hodnotě 1.618033... Zde se pomocí QUITA, která uvedený vzorec implementuje jako index pod jménem *WritersView*, pokusíme čistě experimentálně přidat další pozorování.

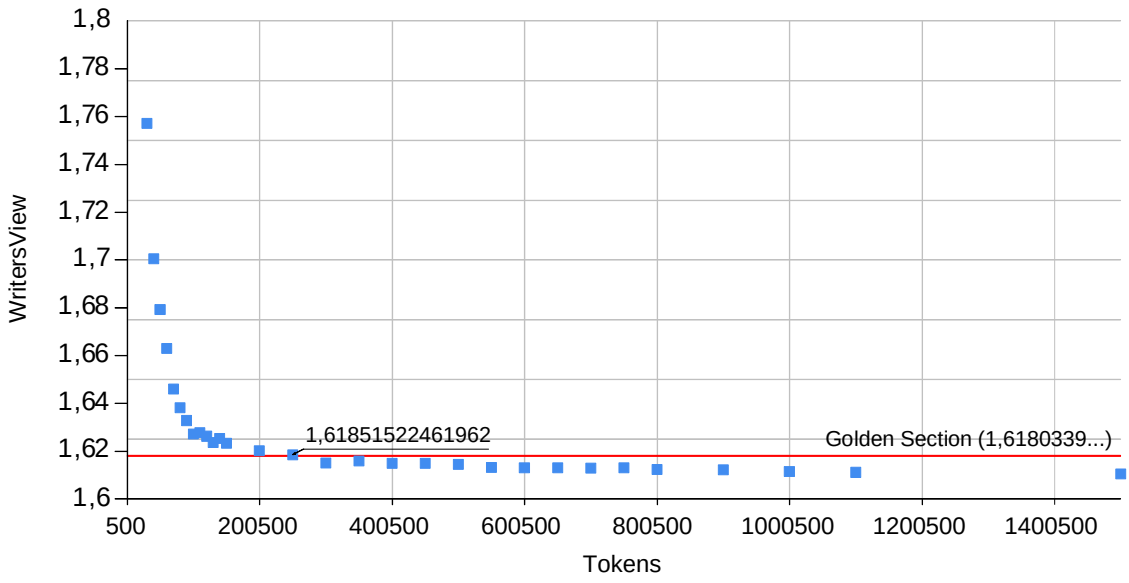
3.3.1 Test na náhodných datech

Hodnoty *WritersView* nejprve otestujeme na náhodných datech vytvořených pomocí QUITA (s nastavením 1-8 znaků, abeceda „abcdefghijklmnopqrstuvwxyz“; viz diskuze výše v kapitole QUITA Random Text Creator vs. české texty).

Tokens	Types	h-Point	WritersView
500	455	4	2,826494
1000	885	7	2,983278
5000	4185	20	2,446262
10000	7999	26	2,219775
10500	8427	26	2,220689
20000	15576	27	1,859588
25000	19087	27	1,784033
30000	22841	27	1,757156
35000	26390	27	1,724013
40000	30030	27	1,700611
45000	33532	27	1,693311
50000	37315	27	1,679254
55000	40617	27	1,668414
60000	44359	27	1,662919
65000	47896	27	1,655229
70000	51190	27	1,646071
75000	54678	27	1,641617
80000	58415	27	1,638143
85000	61881	28	1,63786
90000	65306	29	1,632821
95000	68511	30	1,631712
100000	71961	29	1,627127
105000	75295	31	1,62917
110000	78744	32	1,627658
115000	82419	33	1,627862
120000	85456	34	1,626311
125000	88682	34	1,623009
130000	92192	35	1,623586
135000	95719	38	1,625805
140000	99034	38	1,625369
145000	102504	37	1,622095
150000	105747	39	1,623296
200000	138386	48	1,620157
250000	171008	58	1,618515
300000	201874	68	1,614978

350000	233118	77	1,61593
400000	264358	86	1,614859
450000	295023	96	1,614849
500000	325793	105	1,614512
550000	356524	112	1,613097
600000	386263	123	1,612926
650000	417761	132	1,613046
700000	447049	141	1,612876
750000	478041	149	1,61302
800000	507502	158	1,612376
900000	567627	176	1,612171
1000000	626256	196	1,611542
1100000	685652	213	1,611174
1500000	919922	282	1,610432
5000000	2841699	702	1,600829
10000000	5377893	703	1,58563

Pro náhodná data se hodnota indexu *WritersView*, jak můžeme pozorovat v tabulce výše, s narůstajícím počtem tokenů až do množství 250000 přibližuje hodnotě zlatého řezu (tj. ≈1,6180339…) shora. S vyšším množstvím tokenů pak *WritersView* pod tuto hodnotu klesá. Průběh viz Obrázek 23 - WritersView a náhodná data:

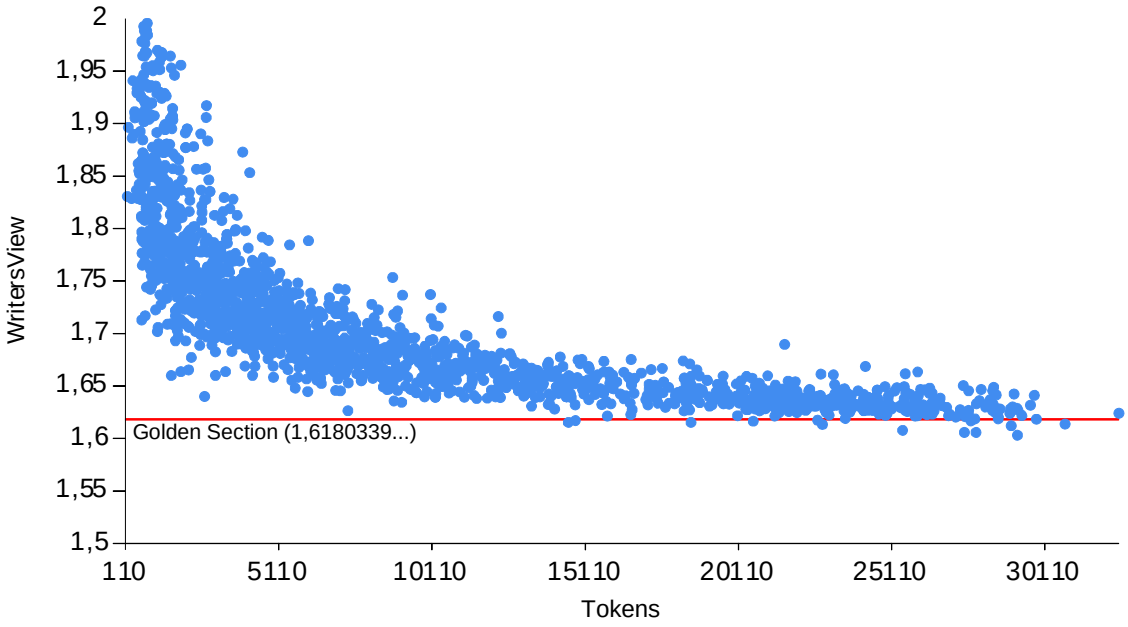


Obrázek 23 - WritersView a náhodná data

3.3.2 Test na českých textech

Dále provedeme test *WritersView* na 1901 přirozených českých nelemmatizovaných textech, tokenizovaných naivní metodou (diskutovanou výše). Veškeré výsledky viz *Příloha: Výsledky indexu WritersView na českých textech*.

Hodnoty výše uvedené tabulky v grafu (viz Obrázek 24 - WritersView a české texty):



Obrázek 24 - WritersView a české texty

Při porovnání výsledků *WritersView* získaných na českých nelemmatizovaných textech s výsledky získanými na náhodných datech můžeme pozorovat obdobný průběh.

4 Úpravy softwaru QUITA a práce se zdrojovými kódy

QUITA vznikla, jak již bylo nastíněno v úvodu, především za účelem zpřístupnit kvantitativní lingvistiku lidem, kteří nechtějí či nemají čas zdlouhavě počítat rozsáhlé matematické vzorce a následně kontrolovat výsledky všech těchto výpočtů a zjišťovat, zda jsou vůbec správné. Dále je QUITA také určena pro samotné kvantitativní lingvisty, pro které je rovněž jistým úskalím provádět všechny výpočty ručně, zapisovat složité vzorce do tabulkových procesorů nebo vytvářet stále tytéž procedury v programovacích jazycích určených pro statistické výpočty, ve kterých je dále libovolná změna v celém procesu – a tím je myšleno vše od změny tokenizace, lemmatizace, vytváření n-gramů, redukce textu a dalších *post-processingových* úloh – krajně obtížné nebo snad i skrze samotnou implementaci nereálné. QUITA tedy z tohoto důvodu není pouze programem, který je možné používat „tak, jak je“, ale je možné jej také upravovat a přidávat do něj další vzorce, výpočty a různé další postupy, které by bylo jinak složité realizovat vlastními silami od úplného začátku. QUITA byla navržena jako sada předem připravených nástrojů s cílem ulehčit práci komukoliv, kdo by chtěl s QUITA tímto způsobem pracovat a přidávat další obsah. Právě tato kapitola se věnuje vysvětlení, jak s *frameworkem* QUITA pracovat od naprosto úplných základů, tj. práce se zdrojovým kódem, úvodem do konstrukcí použitého programovacího jazyka, ilustrujícím příkladům takových nejdůležitějších konstrukcí, a následně, jak krok za krokem vytvořit vlastní index podle zadaného vzorce a mnohé další.

Právě z výše zmíněných (a jiných) důvodů byl vybrán programovací jazyk Visual Basic.NET (VB.NET), který má syntax, klíčová slova i způsob zápisu jednak velmi blízký angličtině (a tedy by měl být snadno pochopitelný kýmkoliv, kdo s žádným programovacím jazykem neměl zatím víceméně žádné zkušenosti), a dále je rovněž velmi blízký ostatním programovacím jazykům, ať už se jedná o populární C# nebo moderní skriptovací jazyky jako Perl, Python a Ruby. Zvládnutí jazyka VB.NET následně zaručuje relativně snadný přechod k jazykům kteréhokoliv směru.

QUITA je tedy tzv. „open source“ projekt, kdy je k samotnému programu přiložen i jeho zdrojový text (dále „zdrojový kód“), který je možný upravovat a upravený program nadále využívat a případně i šířit.

4.1 Úvod – editace zdrojového kódu a orientace ve Visual Studiu

Aby bylo možné zdrojový kód QUITA upravovat a testovat, je nutné využít specializované studio – tzv. IDE (Integrated Development Environment) neboli vývojové prostředí, které jednak slouží pro editaci zdrojového kódu, ale také pro tzv. kompilaci umožňující zdrojový kód „spustit“.

Jak již bylo řečeno výše, QUITA je napsána v programovacím jazyku Visual Basic.NET (VB.NET). IDE umožňující veškerou potřebnou práci s jeho zdrojovými kódy je ke stažení zcela zdarma na stránkách společnosti Microsoft ve verzi *Visual Basic.NET 2010 Express* (odkaz viz níže) nebo je případně možné využít i obsáhlejší verzi tohoto studia integrující mnoho dalších nástrojů a jazyků – *Visual Studio 2010*. *Visual Basic.NET Express* je možné stáhnout zde:

http://www.visualstudio.com/downloads/download-visual-studio-vs#DownloadFamilies_4

Po nainstalování *VB.NET Express* studia pak stačí vytvořit kopii zdrojového kódu QUITA přiloženého na CD (viz Obsah přiloženého CD) nebo stažitelného na stránkách projektu QUITA (http://oltk.upol.cz/software) na disk a pomocí *VB.NET Express IDE* tento projekt otevřít.

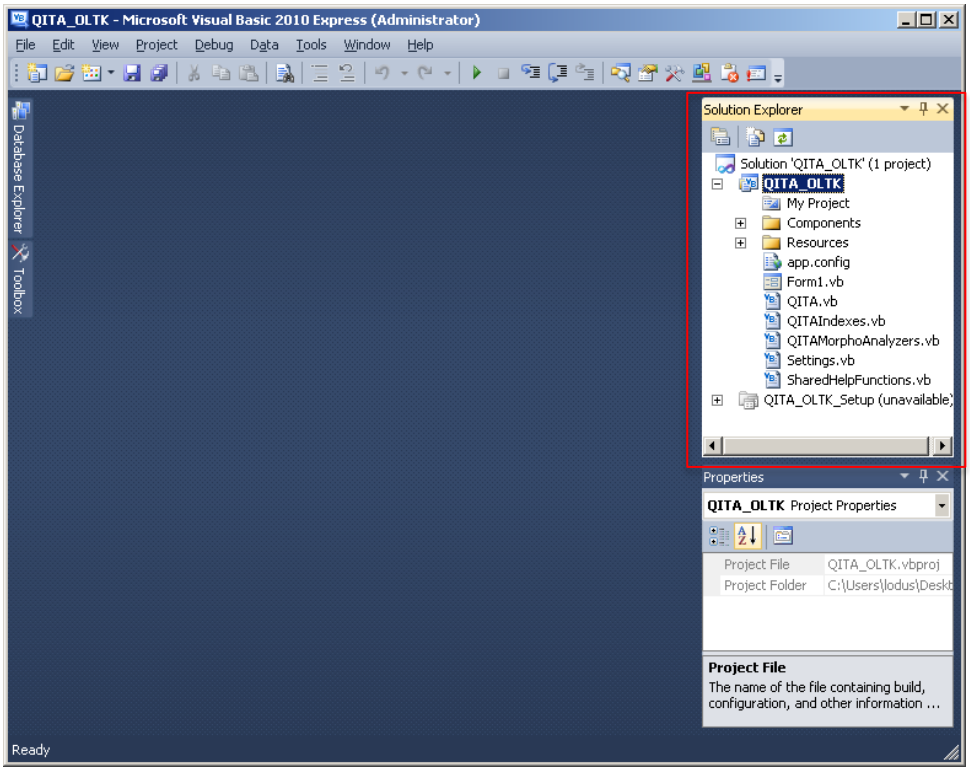
QUITA byla napsána ve starší verzi *Visual Studia* a novější IDE při otevření projektu nabídne aktualizaci zdrojového kódu (viz Obrázek 25 – Konverze projektu). Tlačítkem *Finish* aktualizaci přijmeme a dokončíme.

Obrázek 25 – Konverze projektu

Nyní je možné zdrojový kód QUITA editovat nebo jej okamžitě spustit klávesou F5.

4.1.1 Základní struktura zdrojového kódu QUITA

Zdrojový kód programu QUITA je pro lepší orientaci rozdělen do několika modulů, které jsou v IDE viditelné v okně *Solution Explorer* po pravé straně, viz Obrázek 26:



Obrázek 26 – VB.NET Express

Jedná se o následující čtyři důležité moduly:

- QITA,
- QITAIndexes,
- QITAMorphoAnalyzers,
- SharedHelpFunctions.

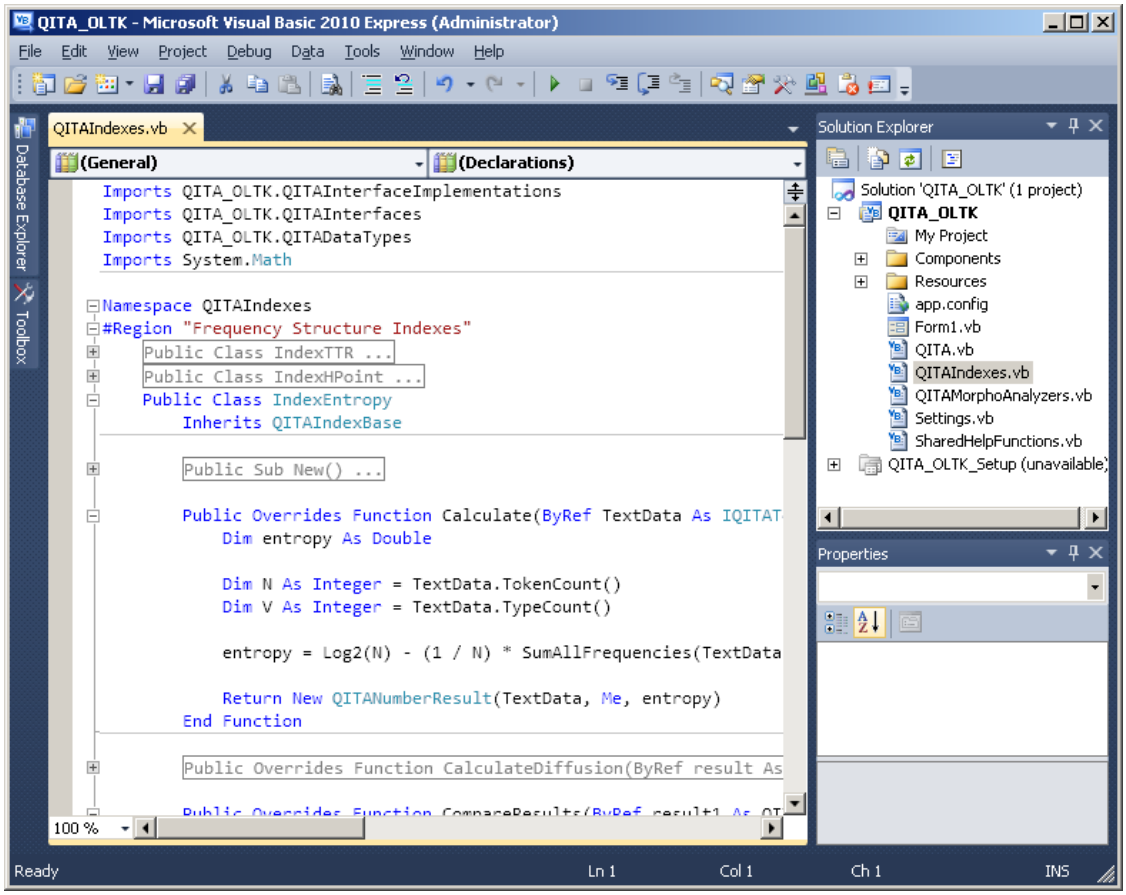
Modul `QITA` obsahuje implementace různých pomocných datových typů, jako jsou tabulky, metadatový typ pro uchovávání výsledků výpočtů aj.

Modul `QITAIndexes` obsahuje implementace jednotlivých indexů, tj. vzorce pro výpočet samotného indexu, výpočet rozptylu a implementace výpočtu pro porovnání.

Modul `QITAMorphoAnalyzers` obsahuje implementace tzv. *wrapperů*, tj. určitých přemostění zprostředkovávajících komunikaci programu QUITA s různými dalšími již hotovými programy, jako např. s Python či Perl skripty, spustitelnými soubory nebo i webovými službami.

Modul `SharedHelpFunctions` pak obsahuje pomocné funkce využívané programem.

Poklepáním myší na kterýkoliv z těchto modulů je možné zobrazit jeho obsah a dále jej upravovat, viz např. Obrázek 27:



Obrázek 27 – Editace souboru s indexy

4.2 Úvod do VB.NET konstrukcí

Aby bylo možné dále vysvětlit, jak se zdrojovým kódem pracovat, jak jej upravovat a testovat, je nutné vymezit některé pojmy a základní konstrukce jazyka VB.NET.

4.2.1 Třídy, instance, objekty

VB.NET je objektově orientovaný jazyk. Téměř vše je zde tzv. objektem, který má určité vlastnosti a metody, které s ním umožňují pracovat. Abstraktem takového objektu je jeho *třída* (*class*). Konkrétní „zhmotnění“ či realizace takové třídy se pak nazývá *instance*. QUITA definuje každý kvantitativně lingvistický index právě jako třídu s několika metodami a vlastnostmi, které s tímto indexem umožňují pracovat. Zcela intuitivně to znamená, že každý index má v QUITA metodu „vypočítej hodnotu pro zadaný text“ a dále také (vzhledem k tomu, že každý index počítá svůj rozptyl jiným způsobem) metodu „porovnej tyto dva výsledky“. Třída indexu, kromě zmíněných metod, obsahuje i *vlastnosti* (atributy), které uchovávají jméno indexu, zkratku, jméno autora a odkaz do literatury.

K vlastnostem a metodám třídy se přistupuje pomocí tzv. tečkové notace. Například, pokud bychom chtěli *točit volantem* u hypotetické třídy *auto*, pak bychom to provedli následovně:

```
Auto.Volant.Otoč(do prava)
```

Pokud bychom chtěli zjistit barvu sedadla, neboli jeho vlastnost, došli bychom k ní stejným způsobem:

```
Auto.Sedadlo.Barva
```

Není tedy žádným překvapením, že tvorba indexu v QUITA zahrnuje pouze nastavení všech potřebných vlastností (informací o indexu) a nadefinování metod, které pouze aplikující vzorce pro jednotlivé výpočty. Příkladem může být trochu zjednodušený zápis indexu *Type/Token Ratio*:

```
Class Index_TTR                          'implementace indexu TTR
    Rutina Inicializace
        Me.SetIndexName("TTR")
        Me.SetIndexDescription("Type/Token Ratio")
    End Rutiny

    Metoda Calculate(Text) As Výsledek
        Return TextData.TypeCount() / TextData.TokenCount()
    End Metoda
End Class
```

(Tento příklad je zjednodušený proto, že neobsahuje všechna klíčová slova vyžadovaná programovacím jazykem. Více viz dále.) Tímto je nadefinována třída jménem *Index_TTR*, kterému je nastaveno jméno a popiska. Metoda *Calculate* je volána ve chvíli, kdy QUITA potřebuje vypočítat hodnotu indexu TTR pro zadaný *Text*, který je předán jako *argument* (obecně *parametr*). *Text* je samozřejmě také objektem se svými vlastnostmi a metodami – jak je z tohoto příkladu patrné, dvěma metodami třídy *Text* jsou *počet typů* a *počet tokenů*. Tyto dvě hodnoty jsou v indexu TTR pak pouze vyděleny a vráceny (*return*) jako výsledek indexu „Type to token ratio“ (poměr typů a tokenů).

4.2.2 Stručné základy VB.NET

Proměnné

V případě, že chceme např. uložit mezivýsledek výpočtu nebo jen uložit některý z atributů do nějaké proměnné, je nutné takovou proměnnou nejprve zavést a nadefinovat – tzv. deklarovat. Pro naše účely jsou nejdůležitější následující typy proměnných:

- Integer – proměnná uchovávající menší celočíselné hodnoty,
- Long – proměnná uchovávající „dlouhé“ celočíselné hodnoty,
- Double – proměnná uchovávající reálné hodnoty (tj. jak celá čísla, tak i desetinná).
- String – textový řetězec,
- Boolean – uchovává pravdivostní hodnotu True/False.

Samotná deklarace proměnné se pak provádí následujícím způsobem:

```
Dim jménoProměnné As TypProměnné
```

Přiřazení hodnoty určité proměnné se pak provádí rovnítkem: Proměnná nalevo od rovnítka je cílovou proměnnou a výpočty či cokoliv dalšího napravo od rovnítka je zdrojem. Rovnítko bez klíčového slova „if“ („pokud“) případně bez přiřazení do Boolovské proměnné tedy plní funkci přiřazení a neznačí tak relaci ekvivalence. Viz příklad:

```
Dim x As Long = 1       'zavedeme proměnnou s přednast. hodnotou 1
Dim y As Double = 2     'totéž s hodnotou 2
Dim z As Double         'proměnná Z bude obsahovat výsledek výpočtu
Dim zprava As String    'proměnná zprava bude obsahovat zprávu pro uživatele

z = x + y               'do Z je zapsán výsledek „x + y“

if z = 3 then
       zprava = "Vysledek je 3!"
else
       zprava = "Vysledek není 3."
End if

MessageBox.Show(zprava) 'zobrazí zprávu uživateli
```

VB.NET obsahuje i komplexnější způsoby pro uchovávání dat. Např. jde o tzv. seznamy (list) určené pro práci s více hodnotami určitého typu. Deklarace seznamu se provádí následujícím způsobem:

```
Dim jménoSeznamu As New List(of TypProměnných)
```

Typem proměnných, které může *List* uchovávat, může být cokoliv od Long, Double, String až po jakékoliv další třídy (včetně seznamu samotného). V případě uchování číselných proměnných (Long, Double, …) *List* umožňuje základní operace jako suma, nalezení maximálního a nejmenšího prvku, výpočet průměru aj.:

```
Dim frekvence As New List(of Long) 'nový seznam čísel
frekvence.Add(100)                 'přidat 100, 79, 50
frekvence.Add(79)
frekvence.Add(50)
MessageBox.Show(frekvence.Average) 'zobrazí průměr frekvencí
```

Jak si můžete povšimnout v příkladech výše, je za klíčově slovo „As“ u deklarací seznamů uvedeno další klíčové slovo, a to „New“. Klíčové slovo „New“ vytvoří novou instanci (konkrétní realizaci) daného netriviálního typu, jakým je zde například onen *List*. Pokud bychom provedli pouze následující deklaraci:

```
Dim frekvence As List(Of Long)
```

byla by proměnná *frekvence* jen jakousi prázdnou obálkou, do které teprve má být vložena nějaká konkrétní instance (realizace) objektu typu „seznam na hodnoty typu Long“. Samotný VB.NET hodnotu takové *prázdné* proměnné označuje jako *Nothing* a při pokusu do takové proměnné přistupovat zobrazí chybu: „Odkaz na objekt není nastaven na instanci objektu.“ Klíčové slovo „New“ tedy zapříčiní vznik nového objektu uvedené třídy. Každá třída pak má předepsáno, co po jejím vytvoření udělat jako první. Tato rutina provádějící inicializaci objektu se nazývá *konstruktor*.

Cykly

Pomocí cyklů můžeme opakovat určitou činnost, procházet všechny prvky *Listu* atd. Nejdůležitějšími jsou pro nás cykly *For*, které explicitně uvádí počet vykonání těla cyklu – viz příklad níže, který zobrazí uživateli čísla 0 až 10:

```
Dim i As Integer
For i = 0 To 10
      MessageBox.Show(i)
Next
```

Obdobně můžeme procházet i *List* „frekvence“ z příkladu uvedeného výše:

```
For i = 0 to (frekvence.Count - 1)
      MessageBox.Show(frekvence.Item(i))
Next
```

Tato smyčka jednoduše zjistí, kolik je v seznamu *frekvence* položek a všechny je postupně zobrazí uživateli. (Pozor, v *List* jsou všechny položky indexované od 0, proto je od celkového počtu položek seznamu, tj. *frekvence.Count*, odečtena 1.)

Pro zjednodušení procházení všech prvků v seznamech (a obdobných konstrukcích) je možné dále používá cyklus *For Each*:

```
Dim prvek as Long
For Each prvek In frekvence
      MessageBox.Show(prvek)
Next
```

Tato smyčka dělá to samé, jako výše uvedený *For*. Jediným rozdílem zde je, že se elegantně vyhýbá nebezpečným výpočtům typu „odečítání 1 od vlastnosti *Count*“. Nevýhodou *For Each* cyklu pak je, že nelze kontrolovat pořadí, ve kterém jsou jednotlivé prvky procházeny.

Podmínky

Jak může být patrné z výše uvedených příkladů, konstrukce VB.NET jsou intuitivně pochopitelné i bez předchozích znalostí tohoto programovacího jazyka. Dalším důležitým konstruktem jsou podmínky, kterými můžeme větvit běh programu:

```
Dim pocetSloves As Integer = 0
Dim pocetNeSloves As Integer = 0
Dim slovo As String

For Each slovo In Text.GetAllTokens()
       If IsVerb(slovo) Then
              pocetSloves += 1
       Else
              pocetNeSloves += 1
       End if
Next
```

Tento jednoduchý příklad prochází všechny tokeny v zadaném textu a dále zjišťuje, zda je daný token sloveso pomocí funkce *IsVerb (...)*. Pokud je token sloveso, je inkrementován čítač počítající množství sloves a pokud token není sloveso, je inkrementován naopak čítač počítající množství ostatních slovních druhů.

V případě větvení s mnoha různými variantami, je vhodné použít konstrukci *Case*:

```
. . .
Dim slovniDruh = ZiskejSlovniDruh(slovo)
Select Case slovniDruh
  Case Sloveso
       pocetSloves+=1

  Case PridavneJmeno
       pocetPridavnychJmen +=1

  Case PodstatneJmeno
       pocetPodstatnychJmen += 1

  Case Else
       pocetOstatnichDruhu += 1
End Select
```

Funkce a rutiny

Některé části kódu se vyplatí zobecnit a vytvořit z nich vlastní obecnou funkci, kterou bude možné dále využívat bez jejího neustálého opakování. Funkce se deklaruje následujícím způsobem:

```
Private Function JménoFunkce() As Typ
       … tělo funkce …
       Return návratová hodnota
End Function
```

Tato jednoduchá deklarace říká, že jde o funkci bez jakýchkoliv parametrů a jejím výsledkem je *Typ*. Pokud bude mít určitá funkce parametry např. *slovo* a *index*, bude deklarace takové funkce vypadat následovně:

```
Private Function JménoFunkce(ByVal slovo as String, ByVal i as Integer) As _
                                                                    Boolean
       … tělo funkce …
       Return návratová hodnota
End Function
```

Výsledek funkce se vrací příkazem *Return*. (Podtržítko značí *pokračování na dalším řádku*.)

Konkrétním příkladem může být funkce *JeSlovesoModalni(sloveso)*, která bude pro zadané sloveso vracet hodnotu *True* v případě, že předané sloveso je modální a hodnotu *False*, pokud modální není:

```
Private Function JeSlovesoModalni(ByVal Sloveso As String) As Boolean
  Select Case Sloveso
       Case "být", "mít", "chtít", "muset", "moci", "smět", . . .
             Return True
       Case Else
              Return False
  End Select
End Function
```

Funkce, které nevrací a nepotřebují vracet žádnou hodnotu, se pak označují jako tzv. rutiny. Jejich deklarace vypadá následovně:

```
Private Sub JménoRutiny(… parametry…)
       … tělo rutiny …
End Sub
```

Příkladem takových rutin jsou již zmíněné konstruktory, které mají za cíl pouze inicializovat objekt.

Pro speciální účely pak existuje i zjednodušený zápis funkce, u které není nutné uvádět její jméno a je možné ji posílat například jiným funkcím jako parametr. Jde o tzv. *anonymní* či *lambda* funkce. Důvod jejich použití je velice praktický. Představme si implementaci funkce *Suma*, která odpovídá zjednodušené standardní matematické funkci suma: postupně prochází všechny číselné prvky zadaného seznamu čísel a na tyto jednotlivé prvky aplikuje sumační funkci. Jednotlivé výsledky sumační funkce pak sčítá a výslednou hodnotu součtu pak vrátí jako svůj výsledek. Implementace funkce *Suma* by pak mohla vypadat například následovně:

```
Public Function Suma(ByVal cisla As List(Of Long), _
                     ByVal sumacniFunkce As Func(Of Long, Double)) As Double
  Dim vysledek As Double = 0

  For Each x As Long In cisla
    vysledek += sumacniFunkce(x)
  Next

  Return vysledek
End Function
```

Takto implementovaná funkce *Suma* je velmi praktická, a to díky tomu, že na prvky můžeme aplikovat libovolnou sumační funkci, kterou dodáme jako parametr. Aby zde však bylo možné bez problémů sumační funkci aplikovat a nedocházelo k nedorozuměním při jejím používání (např. v tom, kolik má mít zrovna zde sumační funkce parametrů, pokud bychom měli funkcí *Suma* větší množství), je nutné mít vyžadovaný tvar tohoto parametru anonymní funkce explicitně deklarovaný. Deklarace parametru typu *funkce* vypadá následovně:

```
ByVal funkce As Func(Of datovýTypPrvníhoParametru, datovýTypDruhéhoParametru,
…, datovýTypVýsledku)
```

Použitá deklarace sumační funkce ve funkci *Suma*, tj.

```
ByVal sumacniFunkce As Func(Of Long, Double)
```

tedy říká, že na vstup této funkce příjde jediný parametr typu *Long* a jejím výsledek je hodnota typu *Double*. (V tomto konkrétním případě je výsledkem *Double*, protože sumační funkce může z *Long* vytvořit prakticky libovolné číslo – včetně těch reálných. Proměnné typu Long jsou určené pouze pro celá čísla.)

Definici konkrétní sumační funkce můžeme provést na jediném řádku přímo ve volání funkce *Suma*, čímž se co nejvíce přiblížit tomu, jak vypadá klasický matematický zápis této funkce. Zápis je následující:

```
Function(jménoParametru1, jménoParametru2, …) (… tělo funkce… )
```

Pokud bychom chtěli například vytvořit sumu „všech frekvencí umocněných na druhou“, tj. v matematickém zápise $\sum_{i=1}^{z} f(i)^2$, kde *f*(i) je frekvence na pozici *i* a *z* je počet frekvencí, pak by volání funkce *Suma* vypadalo následovně:

```
Dim f As List(Of Long) = . . .  načtení seznamu frekvencí . . .
Dim vysledekSumy As Double = Suma(f, Function(fi As Long) (fi^2))
```

(Nezapomínejme, že do sumační funkce v našem případě vstupuje jako argument právě samotný iterovaný prvek „*f(i)*“ a ne iterační proměnná – proto je parametr označen přímo jako „*fi*“ a ne pouze jako „*i*“.)

Třídy

Třídy, jak již bylo řečeno, jsou předpisem pro to, jak má vypadat určitý objekt *dané třídy* a slouží především pro čistší, přehlednější a intuitivnější návrh programového kódu, který ve výsledku šetří čas a mentální energii vývojáře. Třída by měla reflektovat pojetí určitého objektu a poskytnout k němu předem intuitivní rozhraní metod a vlastností. Jednotlivé indexy jsou v QUITA proto implementovány jako třídy, protože je intuitivně jasné, že index slouží k získání nějakého výsledku při aplikaci na nějaký text a také, že má nějaké své jméno a další vlastnosti. Od indexu v QUITA tedy intuitivně očekáváme základní rozhraní pro práci s takovým indexem, tj. funkci (metodu) „vypočítat pro daný text“ a rozhraní vlastností, kterými se můžeme dozvědět jeho jméno aj. Objektově orientovaný přístup k programování je tedy velice výhodné pochopit a plně využívat, neboť jeho koncept je nejblíže běžnému uvažování člověka.

Vytvoření vlastní třídy není nijak složité, s konkrétními příklady se seznámíme v příští kapitole přímo při implementaci vlastních indexů. Deklarace třídy se provádí následujícím zápisem:

```
Public Class
       . . . libovolné proměnné . . .

       Public Sub New()
              . . . kód inicializující nový objekt . . .
       End Sub

       . . . jakékoliv další funkce (metody) . . .
End Class
```

4.3 Vytváření vlastních indexů a jejich testování

Cílem QUITA bylo, mimo samotný program urychlující práci s metodami kvantitativní lingvistiky, vyprodukovat takový programový kód, do kterého bude možné zapisovat nové indexy bez toho, aby bylo nutné cokoliv dalšího v tomto programu upravovat či měnit. Tohoto cíle bylo rozhodně dosaženo a autor nových indexů nemusí do zbytku programu nijak zasahovat, pokud chce pouze přidat nový index – nemusí např. nijak „registrovat“ zobrazení výsledků nového indexu, nemusí upravovat styl zobrazování výsledků a ani nijak složitě řešit případné zobrazení dat v grafu atd. QUITA rovněž obsahuje mnoho vestavěných funkcí a různých datových struktur ulehčující běžné úkoly, které musí kvantitativní lingvisté běžně řešit např. v tabulkových procesorech. Vytvoření nového indexu v QUITA není složitým úkolem. Je pro to ale nutné porozumět všem možnostem, které QUITA v tomto ohledu nabízí a jak těchto možností plně a beze zbytku využít.

Aby bylo možné docílit výše zmíněné vlastnosti QUITA, je pouze nutné dodržet základní strukturu toho, jak má konkrétní implementace indexu vypadat.

4.3.1 Základní struktura implementace indexu

Každý index v QUITA je třídou. Implementaci indexu tedy začneme tím, že tuto třídu vytvoříme:

```
Public Class MůjNovýIndex
    Inherits QITAIndexBase
End Class
```

Aby QUITA věděla, že je *MůjNovýIndex indexem* a zahrnula jej do výpočtů, je nutné přidat řádek *Inherits QITAIndexBase*. Tímto řádkem získá *MůjNovýIndex* vlastnosti vyžadované QUITA pro každý index a vynutí implementaci všech vyžadovaných metod. V rutině *New()* stanovíme základní vlastnosti tohoto indexu, jakými jsou jeho jméno, autor, popiska a skupina, do které bude zařazen:

```
Public Class MujNovyIndex
Inherits QITAIndexBase

Public Sub New()
  Me.SetIndexName("Můj Index")
  Me.SetIndexAuthor("")
  Me.SetIndexDescription("Testovací index")
  Me.SetIndexGroup("Test")
End Sub

Public Overrides Function Calculate(ByRef TextData As IQITAText) As IQITAResult

End Function
End Class
```

Metoda (tj. *funkce určité třídy*) *Calculate* (která by měla být vygenerována automaticky po zápisu řádky *Inherits QITAIndexBase*), je právě metodou, která je volána, jakmile chce QUITA při výpočtech znát po indexu výsledky pro daný text, který je této metodě předán parametrem *TextData*. Výsledkem metody *Calculate* je speciální třída *IQITAResult*, která unifikuje práci s libovolným typem výsledků, ať už to jsou čísla, textové řetězce, tabulky, seznamy nebo jejich kombinace.

4.3.2 Důležité metody objektů IQITAText

Jak již bylo ilustrováno výše, QUITA předává metodě *Calculate* (tj. metodě, která bude obsahovat postup pro výpočet daného indexu) speciální třídu *IQITAText*. Tato třída vytváří jednotné rozhraní pro práci s textem a dále nabízí několik praktických metod usnadňující přístup k jeho základním metrikám. Nyní se podívejme, co nám tato třída nabízí:

Základní metody

Metoda	Vrací	Popis
`.TokenCount`	Integer	Počet tokenů v textu.
`.N`	Integer	Počet tokenů v textu.
`.TypeCount`	Integer	Počet typů v textu.
`.V`	Integer	Počet typů v textu.
`.Tokens`	String()	Pole všech tokenů.
`.Types`	String()	Pole všech typů.
`.GetWordFrequencyTable`	QITAFrequencyTable	Tabulka frekvencí, použití viz QITAFrequencyTable. Seřazeno od nejvyšší frekvence po nejnižší.
`.GetFrequencyToAveragedRankTable`	QITAPositiveArray	Tabulka frekvencí a jejich zprůměr. ranku. Použití viz QITAPositiveArray.
`.GetFrequencyPositiveArray`	Integer()	Pole všech frekvencí, seřazeno od nejvyššího po nejnižší.

Vytvoření klasického indexu *TTR* „Type to Token Ratio“ je tedy otázkou pár řádků a především pak jen několika minut:

```
. . .
Public Overrides Function Calculate(ByRef TextData As IQITAText) As IQITAResult
  Dim typeToTokenRatio As Double = TextData.V / TextData.N

  Return New QITANumberResult(TextData, typeToTokenRatio, Me)
End Function

. . .
```

Metoda *Calculate* indexu *TTR* vrací pouze číselný výsledek, tj. třída *QITANumberResult*. Více o návratových hodnotách viz podkapitola *Datový typ výsledku* dále v textu.

4.3.3 Pomocné datové typy IQITA…

Počítání indexů obnáší mnoho partikulárních kroků, které se často nejjednodušeji řeší pomocí tabulky. QUITA proto nabízí několik tříd pro práci s tabulkami a polemi („jednosloupcová“ tabulka). Dále si tyto třídy blíže předtavíme.

QITATable

QITATable je třídou implementující abstraktní práci s tabulkami a umožňuje nad nimi pohodlně pracovat. *QITATable* je nejprve nutné inicializovat stanovením jmen jednotlivých sloupců a případně i stanovit, které sloupce mohou být použity jako *x*-ové a *y*-ové hodnoty při vykreslování tabulky do grafu, viz příklad inicializace tabulky se třemi sloupci „Slovo“, „Délka“, „Počet konsonantů“ a nastavením, že za zdroje pro jednotlivé osy budou použity sloupce „Délka“ a „Počet konsonantů“:

```
Dim tabulka As New QITATable("Jméno tabulky")
tabulka.AddColumns("Slovo", "Délka", "Počet-Konsonantů")  'vytvoří tři sloupce
tabulka.SetChartableColumns("Délka", "Počet-Konsonantů")  'lze vykreslit
```

Následně je možné s *QITATable* pracovat jako s běžnou tabulkou několika základními metodami (pro kompletní přehled metod nabízených třídou *QITATable* viz *Programátorská příručka* na přiloženém CD).

`Tabulka.`**`AddRow`** `(… hodnoty … )`

Popis: Přidá do tabulky nový řádek se zadanými hodnotami. Pořadí parametrů odpovídá pořadí sloupců.
Návratová hodnota: index řádku.

`Tabulka.`**`AddRows`** `(počet řádků)`

Popis: Přidá do tabulky *n* nových prázdných řádků.
Návratová hodnota: index řádku.

`Tabulka.`**`SumColumn`** `(jméno sloupce)`

Popis: Sečte všechny hodnoty v zadaném sloupci.
Návratová hodnota: Výsledek součtu zadaného sloupce.

`Tabulka.`**`AverageColumn`** `(jméno sloupce)`

Popis: Vypočítá průměr hodnot v zadaném sloupci.
Návratová hodnota: Výsledný průměr zadaného sloupce.

```
Tabulka.Agregate (jméno zdrojového sloupce, _
                  jméno cílového sloupce, _
                 funkce f(Of obsahBuňky As Object, Object))
```

Popis: Metoda *agregate* postupně prochází *zdrojový sloupec*, na obsaženou hodnotu aplikuje funkci *f* a výsledek uloží do *cílového sloupce*.
Návratová hodnota: *True*, pokud byly sloupce nalezeny.

```
Tabulka.Agregate (sloupec, funkce f(Of indexŘádku As Integer, _
                                    obsahBuňky As Object, Object))
```

Popis: Metoda *agregate* postupně prochází *sloupec*, na obsaženou hodnotu *obsahBuňky* (která může být NULL) aplikuje funkci *f* (které je předán i index řádku). Výsledek funkce *f* je následně uložen jako nová hodnota zpracovávané buňky.
Návratová hodnota: *True*, pokud byl *sloupec* nalezen.

```
Tabulka.TableToString ()
```
Popis: Vytvoří tisknutelnou textovou reprezentaci tabulky.
Návratová hodnota: Textová reprezentace tabulky.

Příklad použití QITATable

```
Dim tabulka As New QITATable("Druhá mocnina")    'Vytvoří novou tabulku
tabulka.AddColumns("X", "Y")                     'Nastaví jména sloupců na X a Y
tabulka.SetChartableColumns("X", "Y")            'Oba sloupce lze zobrazit
tabulka.AddRows(5)                               'Přidá 5 prázdných řádků


'Pomocí funkce agregate do každé buňky sloupce X zapíše její index:
tabulka.Agregate("X", Function(index As Integer, content As Object) (index))

'Pomocí funkce agregate do každé buňky Y zapíše druhou mocninu čísla
'ze sloupce X:
tabulka.Agregate("X", "Y", Function(n As Object) (Math.Pow(n, 2)))

'Zobrazí tabulku:
MessageBox.Show(tabulka.TableToString())
```

Výstup:

```
X       Y
1       1
2       4
3       9
4       16
5       25
```

QITAFrequencyTable

Mnoho indexů využívá tabulku frekvencí jednotlivých typů. QUITA pro tento účel nabízí speciální tabulku *QITAFrequencyTable*, kterou lze pro daný text získat metodou třídy *IQITAText* *GetWordFrequencyTable*. Typy jsou v tabulce seřazeny od nejvyšší frekvence po nejnižší (na indexu 1 je tedy nejfrekventovanější type).

Základní metody

Metoda	Vrací	Popis
`.GetWordFrequency(`*typ*`)`	Integer	Získá frekvenci zadaného type.
`.GetWordAtIndex(`*index*`)`	String	Získá type na zadaném indexu (řádku tab.).

Příklad:

```
'Uložíme tabulku frekvencí do proměnné, ušetříme tak následující psaní:
Dim tabFrekvenci As QITAFrequencyTable = TextData.GetWordFrequencyTable

'Získáme frekvenci slova „být“:
Dim frekvSlovaByt As Integer = tabFrekvenci.GetWordFrequency("být")

'Získáme 10. nejfrekventovanější slovo:
Dim desateNejFrekvSlovo As String = tabFrekvenci.GetWordAtIndex(10)
```

QITAPositiveArray

QITAPositiveArray je speciální implementací pole (viz kapitola *Stručné základy VB.NET – Proměnné)*, která indexuje své hodnoty od 1 a ne od 0 a umožňuje tak pohodlnou implementaci vzorců (především pak těch, které obsahují sumu) bez toho, aby autor indexu musel nutně dávat pozor na různé druhy indexace.

Základní metody

Metoda	Vrací	Popis
`.At(index)`	Object	Získá uložený prvek na zadaném indexu.
`.Put(index, data)`	True	Nastaví prvek na zadaném indexu na *data*.
`.ToPositiveArray(of Typ)`	Pole	Konvertuje *QITAPositiveArray* na pole typu *typ*.

Příklad

```
'Získáme tabulku frekvencí s jejich průměrovaným rankem
Dim freqToAvgRank As QITAPositiveArray = TextData.GetFrequencyAveragedRankTable

'Zobrazí průměrovaný rank typů s frekvencí 10:
MessageBox.Show(freqToAvgRank.At(10))
```

4.3.4 Datový typ výsledku

Indexy mohou mít různé výstupy. Většina indexů má jako svůj výstup číslo, jiné indexy mají jako výstup číslo a tabulku a některé další indexy mají jako svůj výstup pouze tabulku nebo textový řetězec. Aby QUITA mohla správně zobrazit všechny tyto kombinace, nevrací metoda *Calculate* jednotlivých indexů striktně proměnné určitého typu (jako *String*, *Integer* atp.), ale speciální třídu zabalující výsledek do uniformní podoby objektů *IQITAResult*. Konkrétními výsledkovými třídami jsou *QITANumberResult*, *QITAStringResult* a *QITAComplexResult*. Tyto třídy kromě samotného výsledku výpočtu nesou i další důležité informace, jakými jsou reference na text (podle kterého byly vypočítány) a také referenci na samotný index, jehož jsou výsledky. Jak může být z názvů jednotlivých *výsledkových tříd* patrné, je každá z nich určena specifickému druhu výsledů, tj. číselným výsledkům, „textovým“ výsledkům a komplexním výsledkům.

QITANumberResult

QITANumberResult je třídou určenou pro výsledky typu *číslo*. Metoda *Calculate* tak vrací výsledek následujícím způsobem:

```
Return New QITANumberResult(TextData, výsledek, Me)
```

TextData je reference k textu, na který byl index aplikován, *výsledek* je libovolné číslo (*Integer*, *Long*, *Double*, …) a *Me* je reference na aktuální index.

QITAStringResult

Zcela obdobně je *QITAStringResult* třídou určenou pro výsledky typu *String*.

```
Return New QITANumberResult(TextData, Me, výsledek)
```

QITAComplexResult

QITAComplexResult je určena pro komplexní či složitý/kombinovaný výsledek. Např. index *Tématické koncentrace* (viz Čech 2014, 17) má za výsledek číslo (samotná hodnota *tématické koncentrace*), avšak je vhodné k tomuto výsledku přidat i tabulku obsahující jednotlivá *tématická slova* a jejich vypočítané *tématické váhy*, ze kterých je hodnota *TK* tvořena. Třída výsledků *QITAComplexResult* umožňuje vrátit oba takové výsledky.

```
Return New QITAComplexResult(TextData, Me, x, y)
```

X je „jednoduchý“ výsledek (*číslo*, *String*; např. ona hodnota TK). *Y* je komplexní hodnota (např. *QITATable*, *QITAPositiveArray*, *List*, *Array*; např. zmíněná tabulka *tématických slov* a *jejich vah*).

4.3.5 Základní matematické funkce a H-Bod

QUITA obsahuje již předem nachystané často používané matematické funkce. Kromě standardní sady funkcí přístupné přes *namespace Math*, se jedná především o funkci sumy a jednoduchý přístup k h-bodu.

h-Bod

QUITA umožňuje okamžitý a jednoduchý přístup k hodnotě h-bodu pomocí funkce *GetHPoint(text)*, viz příklad:

```
Public Function Calculate(ByVal TextData As IQITAText) As IQITAResult
       Dim h As Double = GetHPoint(TextData)
       . . .
End Function
```

Suma

QUITA obsahuje základní variantu počítaní sumy:

```
SumPositiveArray (pole As Integer(), start, konec, funkce) As Double
```

Kde *pole* je polem *integerů* indexovaných od 1. *Start* je dolní hranice, *konec* horní hranice. *Funkce* je pak funkcí:

```
Function(prvek, sumacniIndex) As Double
```

Kde *prvek* je hodnotou *pole* na sumačním indexu *sumacniIndex* (tedy není nutné se ve *funkci* na tuto hodnotu odkazovat stylem *pole(sumacniIndex)*, ale jen jako na *prvek*).

Příklad

Přepis části vzorce pro výpočet indexu *Lambda* (Popescu et al 2011, 1):

$$L = \sum_{i=1}^{V-1} \sqrt{(f_i - f_{i+1})^2 + 1}$$

kde V je počet typů, f_i frekvence typu na indexu i.

```
Dim L As Double
Dim V As Integer = TextData.V
Dim f As Integer() = TextData.GetFrequencyPositiveArray

L = SumPositiveArray(f, 1, V - 1, Function(fi As Long, i As Integer) _
                                      (Math.Sqrt((f(i) - f(i + 1)) ^ 2 + 1)))
```

QUITA má rovněž implementovánu funkci pro zjednušení počítání sumy nad frekvencemi:

```
SumAllFrequencies(IQITAText, start, konec, funkce) As Double
```

Tato varianta je stejná jako předchozí funkce *SumPositiveArray*, avšak jako první parametr vyžaduje text, ze kterého si funkce *SumAllFrequencies* sama načte pole frekvencí.

4.3.6 Přístup k lemmatizátoru a POS taggeru

Některé indexy potřebují spolupracovat s nastavenými POS taggery a případně i lemmatizátory, aby mohly své výpočty realizovat. Např. již několikrát zmiňovaný index *Tématické koncentrace* nutně vyžaduje komunikaci s POS taggerem, pomocí kterého zjišťuje, zda je aktuálně zkoumaný token vhodným slovním druhem (tj. podstatným jménem, slovesem nebo přídavným jménem). Každý index může s nastaveným POS taggerem i lemmatizátorem komunikovat skrze referenci uloženou v předaném argumentu *TextData*:

Atribut	Popis
`.AssignedPOSTagger`	Reference na třídu typu *IQITAMorphoAnalyzer*.
`.AssignedLemmatizer`	Reference na třídu typu *IQITAMorphoAnalyzer*.

Pozor: Atributy *.AssignedPOSTagger* i *.AssignedLemmatizer* jsou nastaveny na hodnotu *Nothing* v případě, že uživatel žádný POS tagger či lemmatizátor nenastaví. Proto je nutné před jejich použitím otestovat, zda jsou vůbec nastavené. Typicky je to možné provést následujícím způsobem:

```
. . .
If TextData.AssignedPOSTagger Is Nothing Then
       Return New QITAStringResult(TextData, Me, "[no tagger]")
End If
. . .
```

Případně pomocí funkce *IsAny(objekt)*:

```
. . .
If Not IsAny(TextData.AssignedPOSTagger) Then
       Return New QITAStringResult(TextData, Me, "[no tagger]")
End If
. . .
```

Třída *IQITAMorphoAnalyzer* zprostředkovává přístup k lemmatizaci a POS taggingu následujícím způsobem:

Metoda	Vrací
`.AssignedPOSTagger.GetWordType(slovo)`	*PartOfSpeechType*
`.AssignedLemmatizer.LemmatizeWord(slovo)`	*String*

Příklad:

```
Dim token As String = "šel"
Dim lemma As String = TextData.AssignedLemmatizer.LemmatizeWord(token)
Dim typ   As PartOfSpeechType = TextData.AssignedPOSTagger.GetWordType(lemma)

MessageBox.Show(Token + ", " + lemma + ", " + typ.ToString)
```

Zobrazí uživateli „`šel, jít, verb`“.

4.3.7 Příklad tvorby základního indexu

Postup pro vytvoření nového, zcela základního indexu má následující postup:

1. Ve studiu *VB.NET Express* otevřeme soubor obsahující jednotlivé indexy dostupné v QUITA: QITAIndexes.vb
2. Na konec souboru (popřípadě kamkoliv jinam) zapíšeme definici třídy. Zde budeme na ukázku implementovat kompletně již představený index *TTR*.

```
Public Class IndexNoveTTR
  Inherits QITAIndexBase

  Public Sub New()
  End Sub

  Public Overrides Function Calculate(ByRef TextData As IQITAText) As IQITAResult
  End Function
End Class
```

3. Vyplníme informace o indexu v konstruktoru a do metody *Calculate* zapíšeme výpočet:

```
Public Class IndexNoveTTR
  Inherits QITAIndexBase

  Public Sub New()
      Me.SetIndexName("NoveTTR")
      Me.SetIndexDescription("Nove Type/Token Ratio")
      Me.SetIndexGroup("Frequency Structure Indexes")
  End Sub

  Public Overrides Function Calculate(ByRef TextData As IQITAText) As IQITAResult
      Dim ttr as Double
      ttr = TextData.TypeCount() / TextData.TokenCount()

      Return New QITANumberResult(TextData, Me, ttr)
  End Function
End Class
```

4. Nyní stačí QUITA spustit klávesou F5 a nový index otestovat. Způsobům, jak ladit (tzv. *debugovat*) kód a prohlížet hodnoty jednotlivých proměnných přímo při výpočtech, se budeme věnovat v kapitole 4.6.1 *Testování ve Visual Studiu: Debuggování.*

4.4 Vytváření vlastních porovnávatelných indexů

QUITA umožňuje výsledky indexů porovnávat. Výsledek porovnání dvou výsledků téhož indexu zajišťuje metoda *CompareResults* implementovaná daným indexem. Tato metoda na vstup při požadavku na porovnání obdrží dva výsledky *result1* a *result2*. Deklarace metody *CompareResults* pak vypadá následovně:

```
Public Overrides Function CompareResults(ByRef result1 As IQITAResult, _
                                         ByRef result2 As IQITAResult) As _
                                        IQITAResult
```

Návratovou hodnotou porovnání je jakýkoliv výsledek typu *IQITAResult*.

Autor indexu však musí nutně explicitně v inicializaci indexu uvést, že se jedná o index, který může porovnávat své výsledky, a to nastavením atributu indexu *HasComparableResults* metodou `SetHasComparableResults`():

```
Me.SetHasComparableResults(True)
```

QUITA rovněž implementuje klasický *u*-test jako funkci *UasymptoticTest*, která na svůj vstup vyžaduje oba výsledky k porovnání a jejich vypočítané rozptyly, viz příklad níže.

Příklad porovnávací funkce

Typicky vypadá metoda *CompareResults* následovně:

```
Public Overrides Function CompareResults(ByRef result1 As IQITAResult, _
                                         ByRef result2 As IQITAResult) As _
                                        IQITAResult
  Dim u As Double

  u = UAsymptoticTest(result1, _
                      result2, _
                      Me.CalculateDiffusion(result1), _
                      Me.CalculateDiffusion(result2))

  Return New QITANumberResult(u)
End Function
```

Protože jednotlivé porovnávatelné indexy implementují vlastní metodu pro výpočet rozptylu (*CalculateDiffusion*) a následně často využívají *u*-testu pro samotné vyhodnocení. Samozřejmě může být metoda *CompareResults* implementována i jakkoliv jinak.

4.5 Kompletní příklad implementace indexu

Kompletní implementaci indexu včetně porovnávání si předvedeme na *Giniho koeficientu* (*G*; viz Čech 2014, 41). Výpočet indexu *G* vypadá následovně:

$$G = \frac{1}{V}(V + 1 - 2m_1') \quad ,$$

kde *V* je počet typů, *N* počet tokenů, *r* pořadí, *f(r)* frekvence slova v pořadí a

$$m_1' = \frac{\sum_{r=1}^{V} f(r)}{N} \quad .$$

Nejprve vytvoříme třídu indexu a k ní inicializační proceduru:

```
Public Class IndexGiniCoeficient
  Inherits QITAIndexBase

  'Inicializace informací o indexu
  Public Sub New()
    Me.SetIndexName("G")                          'zkratka
    Me.SetIndexDescription("Ginis Coeficient")    'popiska
    Me.SetHasComparableResults(True)              'výsledky lze porovnávat
  End Sub
```

Následně implementujeme funkce pro pomocné výpočty – zpřehlední to kód finálního výpočtu:

```
  'Pomocná funkce vypočítávající m‘1, tj. průměr frekvenční distribuce
  Private Function CalculateM1(ByRef TextData As IQITAText) As Double
    Dim sumaFrekvenci As Double

    sumaFrekvenci = SumAllFrequencies(TextData, _
                                      1,
                                      TextData.V, _
                                      Function(f, r) (r * f))
    Return (sumaFrekvenci / TextData.N)
  End Function
```

Implementujeme metodu *Calculate*:

```
  Public Overrides Function Calculate(ByRef TextData As IQITAText) As _
                                                                  IQITAResult
    Dim G As Double
    Dim m1 As Double

    m1 = Me.CalculateM1(TextData)
    G = (1 / TextData.V) * (TextData.V + 1 - 2 * m1)

    Return New QITANumberResult(TextData, Me, G)
  End Function
```

Implementujeme výpočet rozptylu podle vzorců:

$$Var(G) = \frac{4m_2}{V^2 N} \quad , \quad m_2 = \frac{1}{N}\sum_{i=1}^{V}(r_i - m_1')^2 f(r_i) \quad .$$

```
  Public Overrides Function CalculateDiffusion(ByRef result As IQITAResult) As _
                                                                        IQITAResult
    Dim varG, m1, m2 As Double
    Dim V As Integer = result.TextReference.V
    Dim N As Integer = result.TextReference.N

    m1 = Me.CalculateM1(result.TextReference)
    m2 = (1 / N) * SumAllFrequencies(result.TextReference, 1, V, _
                                     Function(fri, ri)(((ri - m1) ^ 2) * fri))
    varG = (4 * m2) / ((V ^ 2) * N)
    Return New QITANumberResult(varG)
  End Function
```

Samotné porovnání pak provedeme *u*-testem:

```
  Public Overrides Function CompareResults(ByRef result1 As IQITAResult, _
                                           ByRef result2 As IQITAResult) As _
                                                                        IQITAResult
    Dim u As Double
    u = UAsymptoticTest(result1, result2, _
                        Me.CalculateDiffusion(result1), _
                        Me.CalculateDiffusion(result2))

    Return New QITANumberResult(u)
  End Function
End Class
```

4.6 Použití a testování nového indexu

Jakmile je třída indexu napsaná a dědí z *QITAIndexBase*, stačí QUITA pouze spustit (klávesou F5). QUITA index automaticky zaeviduje a zařadí jej do výpočtů. Vzhledem k náročnosti přepsat vzorce do lineárního řetězce programového kódu je však doporučeno nejprve všechny výsledky nejprve ověřit. Také je možné sledovat průběh výpočtu indexu pomocí tzv. *debuggování*.

4.6.1 Testování ve Visual Studiu: Debuggování

Zpracování každého řádku kódu programu lze kdykoliv pozastavit a nahlédnout na všechny hodnoty proměnných (které je v případě potřeby možné i upravit) a případně pokračovat stejným způsobem dalšími řádky. Pozastavení běhu programu se provádí pomocí tzv. *breakpointu*, který se nastavuje na zadaný řádek klávesou F9. Zpracování řádku s breakpointem bude před jeho zpracováním pozastaveno. Pokračování na další řádek je možné provést klávesou F8. Hodnoty proměnných lze zobrazit jednak najetím kurzoru myši nad jméno proměnné nebo pomocí okna *Watch*, do kterého je možné proměnnou přidat pomocí kontextového menu (pravé kliknutí na jméno proměnné) a následně kliknutím na tlačítko *Add Watch*.

Dále je možné nechat vypisovat jakékoliv hodnoty do *Immediate* okna pomocí příkazu *Debug.Print(... řetězec ...)*, např. výpis proměnné *m1* do Immediate okna:

```
Debug.Print(m1)
```

4.7 Vytváření vlastních tokenizátorů, lemmatizátorů a POS taggerů

Jak již bylo řečeno v úvodních kapitolách – QUITA může implementovat vlastní tokenizátory/lemmatizátory/taggery a nebo s nimi může komunikovat pomocí určitých rozhraní. V této kapitole si ukážeme naprosté základy implementace třídy pro morfoanalýzu a základy pro tvorbu přemostění dovolujících QUITA komunikovat s těmito specializovanými nástroji.

Veškeré implementace tokenizátorů, lemmatizátorů a POS taggerů v QUITA musí nutně implementovat interface *IQITAMorphologyAnalyzer*, který zaručuje stejné rozhraní pro všechny morfoanalytické nástroje. Jak již bylo řečeno, implementace lemmatizace a POS taggingu není v aktuálně předkládané verzi QUITA ideální. Přesněji řečeno, aktuální verze implementace třídy *QITAText* není ideální, ale může být upravena a nebo nahrazena libovolnou jinou třídou implementující interface *IQITAText*. Pro bližší pochopení celého procesu zpracování textu pomocí třídy *QITAText* viz následující diagram (Obrázek 28):

Obrázek 28

Dotazování na POS Tagger pak provádí samotné implementace indexů *ad hoc*.

4.7.1 Vytvoření vlastního nástroje uvnitř QUITA

Všechny morfoanalytické nástroje či jejich přemostění jsou implementovány v souboru *QITAMorphoAnalyzers.vb*. Zde je také možné přidávat vlastní tokenizátory, lemmatizátory a taggery. Obdobně jako u indexů, musí každý z těchto nástrojů dědit speciální třídu *QITAMorphoAnalyzerBase* nebo její specifičtější implementace (viz dále) popřípadě přímo implementovat obecný interface *IQITAMorphologyAnalyze*. Pomocí implementace tohoto interface je jednak zajištěno jednotného rozhraní pro práci se všemi morfoanalytickými nástroji a za druhé, slouží jako klíčový identifikátor pro QUITA, kdy jsou třídy s tímto rozhraním automaticky zahrnuty do seznamu všech dostupných nástrojů. Nové třídy tedy není nutné nikam registrovat. Nyní se podíváme na vytváření jednotlivých nástrojů zvlášť.

Tokenizátor

Nový tokenizátor je nejsnadněji implementovatelný pomocí dědění ze třídy *QITATokenizerOnlyBase*, která již zahrnuje všechny potřebné procedury pro tvorbu vlastního tokenizátoru. Jediné, co je nutné v takovém případě doplnit, jsou (1) základní informace, které se (obdobně jako u indexů) vyplňují v konstruktoru třídy a (2) je potřeba implementovat samotnou tokenizační proceduru, a to do metody *TokenizeText*. Pokud budeme chtít například vytvořit nový tokenizátor, který bude text naivně tokenizovat na *větné celky*, a to na základě znaků *tečka*, *otazník* a *vykřičník*, pak to můžeme provést následujícím způsobem:

```
Public Class GenericNaiveSentenceTokenizer
  Inherits QITATokenizerOnlyBase

  Public Sub New()
    Me.SetAnalyzerName("Naivní tokenizátor větných celků")
    Me.SetAnalyzerLanguage("[ GENERIC ]")
    Me.SetAnalyzerDescription("Test naivního tokenizátoru")
  End Sub

  Public Overrides Function TokenizeText(ByVal sText As String) As String()
    Return Regex.Split(sText, "\.|\?|!")
  End Function
End Class
```

Metoda *TokenizeText* tedy na svůj vstup dostává v parametru *sText* text ke zpracování a zpět vrací pole tokenů.

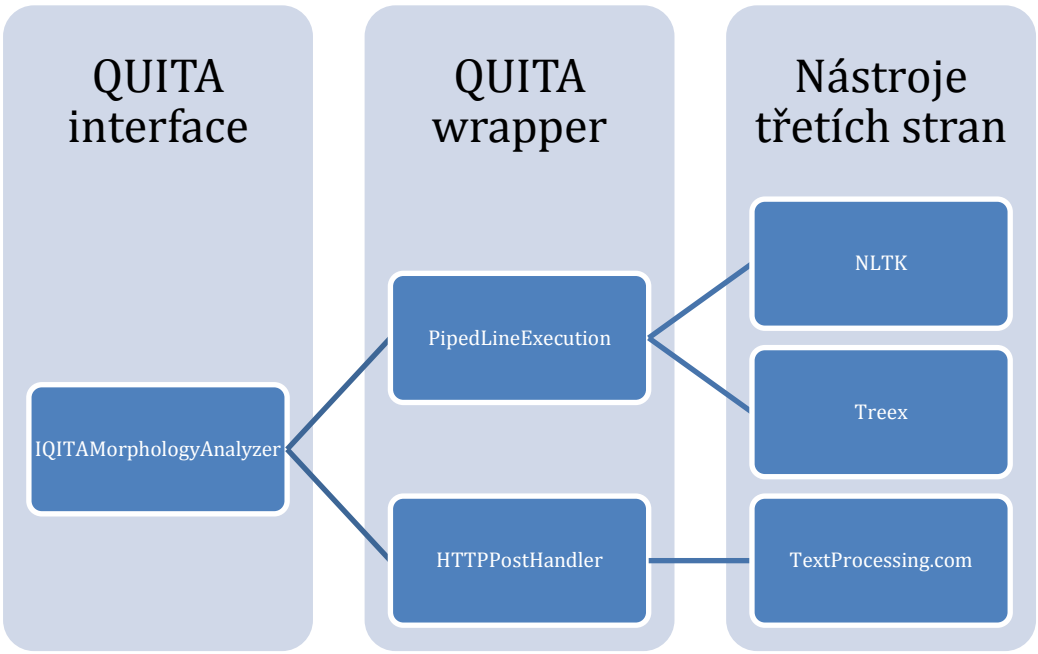
Lemmatizátor a POS tagger

Vytváření těchto nástrojů je již složitější a vyžaduje dědění třídy *QITAMorphoAnalyzerBase* nebo kompletní implementaci rozhraní *IQITAMorphologyAnalyze*. Tyto implementace nejsou triviální a svým charakterem již vyžadují pokročilejší dovednosti v programování. Zde se proto konkrétními implementacemi těchto nástrojů nebudeme zabývat, avšak veškeré potřebné informace o třídě *QITAMorphoAnalyzerBase* a rozhraní *IQITAMorphologyAnalyze* je možné nalézt v přiložené programátorské příručce. Dále je samozřejmě možné nahlédnout do již hotových lemmatizátorů a taggerů ve zdrojovém kódu QUITA.

4.7.2 Vytvoření vlastního přemostění

Vytvoření vlastního přemostění, obdobně jako tvorba lemmatizátorů a POS taggerů uvnitř QUITA, není triviální a vyžaduje navíc i znalosti a zkušenosti ohledně toho, jak mezi sebou programy mohou komunikovat a jakými způsoby toho lze docílit. Dále, hledání případných chyb v implementaci není triviální už zcela a často vyžaduje značnou obratnost v deuggování. Tato podkapitola se tak striktně omezí pouze na tak nutné minimum znalostí, aby bylo možné pochopit způsob, jakým jsou přemostění realizovány.

Schéma komunikace QUITA se *stand-alone* nástroji vypadá následovně (viz Obrázek 29 níže):



Obrázek 29

Veškeré tokenizátory, lemmatizátory a POS taggery jsou, jak již bylo řečeno, implementací rozhraní *IQITAMorpholoyAnalyzer*. Toto rozhraní standardizuje přístup k lokálně implementovaným nástrojům a dále i ke dvěma specializovaným třídám, které implementují dvě základní metody komunikace mezi programy, tj.:

- `PipedLineExecution` – zprostředkovává komunikaci pomocí trubky (pipe), skrze kterou je možné komunikovat s programy pracujícími v příkazové řádce (konzolové programy pracující se STDOUT a STDIN), nebo skripty různých jazyků jako Perl, Python aj.
- `HTTPPostHandler` – implementuje základní komunikaci s HTTP formuláři pomocí metody POST.

Pro vytvoření vlastního přemostění např. tokenizátoru s programem pracujícím se STDIN a STDOUT je třeba pouze vytvořit třídu implementující IQITAMorphoAnalyzer a dále využít možnosti třídy PipedLineExecution.

Hotové implementace takových přemostění si lze prohlédnout ve zdrojovém kódu programu QUITA. Veškeré informace k jednotlivým třídám je možné nalézt v přiložené programátorské příloze.

Závěr

Primárním cílem této práce bylo vytvořit software, který by sloužil kvantitativním lingvistům jako nástroj, pomocí kterého by bylo možné automatizovanou cestou zpracovávat kvantitativní analýzy velkého množství textů. Tento software pak měl být navržen takovým způsobem, který by dokázal kvantitativní analýzu textů zprostředkovat i dalším oborům (jakými jsou například historiografie, psychologie a biologie), tj. takovým způsobem, který by nutně po uživateli nevyžadoval hlubší znalosti matematiky, statistiky, programování nebo samotné kvantitativní lingvistiky. Sekundárním cílem pak bylo pokusit se celý software a jeho zdrojový kód navrhnout tak, aby byla umožněna jeho všestranná upravitelnost a relativně snadná implementace nových indexů.

Průběh této práce tedy sestával ze dvou částí: (1) Vytvoření softwaru na základě výše vytyčených cílů a (2) sepsání textové části za účelem seznámit uživatele s tímto softwarem a veškerými úskalími jeho používání.

Výsledkem této práce je pak software, který se výše uvedeným cílům pokouší co nejvíce dostát. Kvantitativním lingvistům umožňuje aplikovat automatizované a hlavně pak plně parametrizovatelné výpočty nad velkým množstvím textů, které navíc mohou být i v prakticky libovolných jazycích. Dalším oborům (tj. např. zmíněné historiografii, psychologii, biologii aj.) pak tento software poskytuje uživatelsky přívětivé prostředí pro jednoduché zpracování textů, od jejich načtení, až po statistické vyhodnocení a vizualizaci výsledků, to vše bez nutnosti provádět jakékoliv výpočty či ručně provádět jakékoliv zpracování dat. Návrh tohoto softwaru a jeho zdrojového kódu je pak navržen v souladu se sekundárním požadavkem, tj. požadavkem na snadnou modifikovatelnost a možnost přidávat další indexy. Zdrojový kód je dokumentován a v textové části této práce se jedna z kapitol výhradně věnuje pouze tomuto tématu. Text této práce je pak koncipován jako uživatelská příručka s konkrétními ilustrativními příklady reálného použití tohoto software a dále i ukázkami jeho aplikace.

Přínosů této práce je tedy několik. Kvantitativním lingvistům umožňuje zpracovávat velká množství textů, díky čemuž je následně možné rychleji a jednodušeji testovat kvantitativně lingvistické hypotézy. Ostatním oborům pak umožňuje pracovat s kvantitativní analýzou textů bez nutnosti hlouběji proniknout do celého kvantitativně lingvistického aparátu. A v neposlední řadě poskytuje opět kvantitativním lingvistům určitou náhradu či alternativu za skriptovací a jiné jazyky, umožněním snazší implementace indexů a jejich testování.

Samotný software a jeho implementace však mají řadu bodů, které je možné dále vylepšit nebo jinak upravit. Především se jedná o vylepšení ne zcela ideálně provedené spolupráce s různými nástroji zpracovávajícími přirozený jazyk a dále pak např. doplnění dalších možností tzv. *post-processingu* a přidání dalších vizualizačních nástrojů.

Literatura a zdroje

- Čech, Radek, Popescu Ioan-Iovitz a Gabriel Altmann. 2014. *Metody kvantitativní analýzy (nejen) básnických textů*. Univerzita Palackého v Olomouci: Olomouc. (in press)
- Popescu, Ioan-Iovitz et al. 2009. *Word Frequency Studies*. Mouton de Gruyter: Berlin, New York.
- Popescu, Ioan-Iovitz, Čech Radek a Gabriel Altmann. 2011. *The Lambda-structure of Texts*. RAM-Verlag: Lüdenscheid.
- Tuzzi, Arjuna, Popescu Ioan-Iovitz a Gabriel Altmann. 2010. *Quantitative Analysis of Italian texts*. RAM-Verlag: Lüdenscheid.
- Indurkhya, Nitin a Fred J. Damerau. 2010. *Handbook of natural language processing*. Chapman & Hall/CRC: Boca Raton, FL.
- Křen, M. et al. 2014. *Korpus SYN, verze 3 z 27. 1. 2014*. Ústav Českého národního korpusu FF UK: Praha. Dostupný z WWW: http://www.korpus.cz
- Šmerk, Pavel. 2007. Fast Morphological Analysis of Czech. In *Petr Sojka and Aleš Horák. Proceedings of Third Workshop on Recent Advances in Slavonic Natural Language Processing*, 13–16. RASLAN 2009. Brno: Masaryk University.

Obsah přiloženého CD

- BIN/

  Obsahuje soubor instalace softwaru QUITA.

- `DOC/`

  Obsahuje tuto práci ve formátu PDF a dokumentaci ke zdrojovému kódu.

- `SRC/`

  Kompletní zdrojové kódy softwaru QUITA, programy a knihovny třetích stran pro bezproblémové testování.

- `Readme.txt`

  Obsahuje instrukce k instalaci softwaru QUITA a veškeré požadavky.

Dále CD obsahuje:

- `DATA/`
  - `Kapitoly Krakatit/`

    Obsahuje jednotlivé kapitoly díla *Krakatit* Karla Čapka.

  - `Kapitoly Továrna na absolutno/`

    Obsahuje jednotlivé kapitoly díla *Továrna na absolutno* Karla Čapka.

  - `Kapitoly PPK/`

    Obsahuje jednotlivé kapitoly díla *Povídání o pejskovi a kočičce* Josefa Čapka.

  - `Kapitoly Krakatit PPK/`

    Obsahuje kapitoly díla *Krakatit* s přimíchanou kapitolou díla *Povídání o pejskovi a kočičce*.

Příloha: Výsledky indexu *WritersView* na českých textech

Text	N	V	α	Text	N	V	α
1	110	95	2,699681	**41**	491	356	1,933273
2	111	88	2,70147	**42**	495	352	2,049872
3	150	116	2,061223	**43**	505	341	2,012348
4	154	127	2,174927	**44**	511	310	1,828659
5	171	95	1,83055	**45**	512	355	1,929227
6	173	119	2,579671	**46**	515	335	2,202926
7	174	121	2,579225	**47**	528	374	1,861765
8	178	156	2,17187	**48**	553	386	1,8549
9	186	146	2,840966	**49**	558	375	1,842282
10	186	145	2,574863	**50**	558	392	1,931478
11	191	146	2,055568	**51**	569	403	1,852199
12	202	146	1,896245	**52**	571	375	2,127519
13	216	158	2,160323	**53**	595	388	1,892422
14	242	182	2,084453	**54**	596	402	1,835145
15	263	178	2,701062	**55**	607	367	2,019869
16	285	214	2,264674	**56**	608	419	1,925926
17	292	228	2,263473	**57**	608	419	1,925926
18	300	224	2,45915	**58**	610	431	1,827539
19	307	237	2,176039	**59**	610	437	1,863049
20	310	236	1,828705	**60**	618	408	1,92505
21	324	230	2,196383	**61**	618	370	2,118205
22	331	241	1,886243	**62**	619	399	1,941688
23	336	243	2,195286	**63**	621	407	1,908816
24	340	247	2,098639	**64**	629	440	1,811451
25	349	270	1,940845	**65**	630	401	1,864323
26	375	280	2,374441	**66**	637	394	1,906739
27	384	258	1,889536	**67**	637	444	2,083872
28	385	267	2,486006	**68**	640	487	1,810186
29	389	281	2,147575	**69**	640	383	2,036828
30	411	31	2,757351	**70**	641	407	1,712768
31	416	296	2,373434	**71**	642	420	1,978392
32	418	274	1,911201	**72**	642	427	1,863381
33	424	318	1,905326	**73**	643	405	1,789889
34	425	305	1,909268	**74**	643	425	1,830128
35	437	321	2,060262	**75**	643	434	1,796539
36	461	357	2,170162	**76**	649	424	1,776368
37	466	340	2,129232	**77**	650	444	2,100138
38	468	326	1,836526	**78**	652	441	2,017028
39	482	318	1,929379	**79**	657	325	1,865275
40	491	302	2,05478	**80**	661	463	1,764848

81	671	455	1,964695	**121**	785	492	1,828145
82	671	477	1,884297	**122**	786	530	1,864468
83	677	402	1,872355	**123**	786	404	1,820434
84	688	479	1,787525	**124**	787	460	1,818142
85	690	403	2,419002	**125**	792	510	1,847734
86	693	499	1,870939	**126**	793	479	1,953909
87	696	442	1,839771	**127**	793	528	1,847252
88	699	466	1,906723	**128**	794	515	1,937593
89	704	469	1,992672	**129**	801	462	1,744049
90	705	435	1,91143	**130**	802	496	1,988711
91	706	483	1,938522	**131**	808	545	1,920506
92	711	393	1,792021	**132**	808	504	1,967463
93	711	584	1,94662	**133**	817	560	1,863793
94	713	461	1,902326	**134**	818	526	1,781705
95	714	482	1,901675	**135**	823	501	1,818486
96	725	522	1,921089	**136**	824	574	1,995574
97	727	423	2,417979	**137**	828	538	1,98444
98	729	489	1,922023	**138**	834	561	1,852388
99	729	528	1,827291	**139**	839	539	1,823725
100	732	486	1,988214	**140**	841	563	2,046448
101	736	375	1,793916	**141**	850	568	1,858196
102	740	477	1,807802	**142**	851	468	1,856697
103	740	476	1,807834	**143**	870	542	2,074611
104	741	463	1,96446	**144**	873	636	1,776929
105	742	484	1,768722	**145**	873	636	1,776929
106	748	508	1,937786	**146**	877	548	1,934409
107	749	504	1,921583	**147**	882	565	1,792386
108	750	483	1,976139	**148**	887	500	1,903858
109	750	509	1,905507	**149**	895	537	2,097523
110	754	507	1,904546	**150**	897	587	1,798243
111	754	480	1,968286	**151**	899	552	1,909151
112	754	419	1,716737	**152**	899	613	2,023883
113	760	528	1,842615	**153**	903	645	1,826764
114	761	507	1,937814	**154**	904	605	1,824487
115	769	463	1,982595	**155**	907	605	1,764689
116	777	514	2,036293	**156**	909	581	1,810259
117	780	534	1,917733	**157**	910	623	2,023667
118	781	544	1,936846	**158**	915	612	1,866179
119	782	544	1,848221	**159**	924	615	1,935318
120	782	446	1,859785	**160**	926	562	1,951519

161	927	563	1,761004	**201**	1074	632	1,781574
162	928	533	1,741898	**202**	1079	594	1,868761
163	928	546	1,906499	**203**	1085	515	1,817938
164	930	603	2,073685	**204**	1093	568	1,722452
165	933	459	1,936194	**205**	1098	678	1,935567
166	933	628	1,809701	**206**	1101	721	1,778009
167	949	625	1,811373	**207**	1101	741	1,750322
168	949	575	1,746767	**208**	1109	740	1,834294
169	954	613	1,918777	**209**	1113	719	1,811575
170	956	431	1,813485	**210**	1115	656	1,802628
171	957	614	1,826422	**211**	1116	677	1,753588
172	960	594	1,907957	**212**	1117	714	1,834698
173	961	672	1,754584	**213**	1120	642	1,863165
174	962	587	1,829615	**214**	1136	653	2,070849
175	963	504	1,745615	**215**	1137	741	1,891428
176	972	583	1,919746	**216**	1138	705	1,928693
177	974	644	1,823715	**217**	1144	776	1,76689
178	980	613	1,832165	**218**	1147	671	1,819119
179	981	599	1,791703	**219**	1152	632	1,796358
180	981	592	1,757466	**220**	1152	670	1,969803
181	985	599	1,877532	**221**	1156	629	1,70169
182	989	669	1,835471	**222**	1160	752	1,789059
183	991	663	1,784528	**223**	1161	689	1,858218
184	1004	614	1,955307	**224**	1163	711	1,79513
185	1008	629	2,047346	**225**	1175	657	1,705733
186	1009	613	1,829019	**226**	1175	719	1,874946
187	1011	685	1,950303	**227**	1178	714	1,849832
188	1024	613	1,80955	**228**	1179	695	1,800019
189	1025	586	1,849103	**229**	1184	720	1,846323
190	1027	622	1,828825	**230**	1190	691	1,880725
191	1034	628	1,867913	**231**	1192	582	1,858458
192	1035	551	1,803516	**232**	1194	728	1,771364
193	1037	675	1,796268	**233**	1196	578	1,835567
194	1048	640	1,77392	**234**	1205	754	1,800203
195	1051	618	1,907348	**235**	1206	785	1,929876
196	1058	545	2,22418	**236**	1206	785	1,929876
197	1058	545	2,22418	**237**	1207	648	1,800712
198	1066	668	1,83549	**238**	1217	689	1,958611
199	1069	541	1,856072	**239**	1224	697	1,966847
200	1070	645	1,829647	**240**	1227	616	1,75522

241	1228	677	1,951307	**281**	1454	846	1,880531
242	1233	631	1,736951	**282**	1454	846	1,880531
243	1233	774	1,837723	**283**	1455	733	1,73362
244	1240	768	1,788876	**284**	1457	853	1,742585
245	1254	559	1,772223	**285**	1460	881	1,826108
246	1262	714	1,761729	**286**	1467	600	1,754069
247	1266	702	1,760105	**287**	1468	844	1,764027
248	1268	813	1,779399	**288**	1476	747	1,708737
249	1268	797	1,727388	**289**	1482	808	1,79321
250	1269	706	1,782579	**290**	1489	640	1,754655
251	1279	812	1,830762	**291**	1505	790	1,844041
252	1279	851	1,787793	**292**	1510	792	1,837652
253	1284	838	1,779061	**293**	1511	954	1,764511
254	1291	694	1,963429	**294**	1515	840	1,788037
255	1291	694	1,963429	**295**	1517	829	1,825672
256	1294	636	1,829948	**296**	1527	983	1,791528
257	1296	762	1,967606	**297**	1534	994	1,781705
258	1296	762	1,967606	**298**	1541	933	1,832237
259	1297	728	1,924202	**299**	1542	910	1,800036
260	1311	790	1,847596	**300**	1543	866	1,839828
261	1329	756	1,798279	**301**	1545	878	1,775952
262	1332	790	1,925683	**302**	1553	797	1,790119
263	1334	736	1,845102	**303**	1553	797	1,790119
264	1339	814	1,793083	**304**	1554	871	1,880147
265	1341	796	1,847498	**305**	1561	908	1,752679
266	1359	702	1,872346	**306**	1573	677	1,800375
267	1359	855	1,847699	**307**	1575	1036	1,964229
268	1364	735	1,89665	**308**	1577	798	1,709125
269	1369	825	1,856006	**309**	1577	836	1,813034
270	1371	848	1,752283	**310**	1578	888	1,904526
271	1374	614	1,767232	**311**	1589	903	1,814291
272	1387	685	1,928656	**312**	1593	973	1,779638
273	1390	843	1,796876	**313**	1601	947	1,952977
274	1408	754	1,8599	**314**	1605	822	1,659898
275	1410	646	1,899052	**315**	1606	918	1,791402
276	1411	788	1,894238	**316**	1606	879	1,74223
277	1414	885	1,839561	**317**	1622	953	1,820558
278	1416	832	1,873186	**318**	1626	1004	1,74078
279	1443	826	1,92609	**319**	1631	828	1,895078
280	1451	884	1,766441	**320**	1641	876	1,804356

321	1644	946	1,806772	**361**	1817	1036	1,71175
322	1644	799	1,907168	**362**	1817	752	1,724407
323	1644	799	1,907168	**363**	1819	746	1,763922
324	1652	915	1,914397	**364**	1819	926	1,84184
325	1654	977	1,903422	**365**	1821	1032	1,760491
326	1658	1073	1,801337	**366**	1826	1103	1,772265
327	1658	978	1,857093	**367**	1831	826	1,697238
328	1663	748	1,871314	**368**	1836	1016	1,839283
329	1665	870	1,746344	**369**	1838	997	1,774175
330	1672	1057	1,769488	**370**	1842	988	1,809341
331	1672	936	1,758469	**371**	1845	969	1,86564
332	1673	948	1,783016	**372**	1856	1114	1,766093
333	1692	1041	1,7779	**373**	1859	992	1,749663
334	1693	1093	1,801136	**374**	1869	981	1,855338
335	1704	870	1,818086	**375**	1880	1153	1,818414
336	1704	870	1,818086	**376**	1880	1153	1,818414
337	1710	825	1,831803	**377**	1887	1050	1,83755
338	1712	947	1,822304	**378**	1889	949	1,773596
339	1712	999	1,946164	**379**	1896	811	1,725607
340	1715	802	1,832271	**380**	1896	1071	1,766717
341	1715	988	1,814978	**381**	1905	1195	1,747737
342	1721	962	1,807652	**382**	1910	1019	1,713482
343	1721	994	1,76737	**383**	1915	1184	1,785062
344	1729	1042	1,868561	**384**	1915	896	1,955491
345	1739	862	1,721999	**385**	1920	1030	1,836478
346	1740	799	1,707149	**386**	1923	1050	1,663549
347	1744	987	1,717181	**387**	1926	1101	1,814156
348	1744	1018	1,693059	**388**	1936	1032	1,729388
349	1745	800	1,707128	**389**	1948	1049	1,738981
350	1746	842	1,721654	**390**	1955	1139	1,760165
351	1757	908	1,757439	**391**	1957	1031	1,846347
352	1759	809	1,788072	**392**	1966	1110	1,81076
353	1773	1031	1,801253	**393**	1966	1171	1,755879
354	1777	1045	1,745616	**394**	1967	990	1,775764
355	1789	896	1,734845	**395**	1984	1024	1,75308
356	1789	1038	1,78185	**396**	2008	1212	1,745988
357	1798	992	1,838456	**397**	2011	1156	1,792388
358	1801	1073	1,782118	**398**	2023	1044	1,748566
359	1803	783	1,702798	**399**	2025	1100	1,710159
360	1814	973	1,722549	**400**	2026	980	1,74339

401	2034	1176	1,72989	**441**	2297	1133	1,777452
402	2052	1017	1,723175	**442**	2303	1133	1,789454
403	2052	1176	1,708943	**443**	2321	1398	1,786957
404	2058	1186	1,746175	**444**	2326	1306	1,754109
405	2059	1211	1,756832	**445**	2329	1368	1,777793
406	2062	1257	1,774683	**446**	2330	1190	1,80138
407	2064	907	1,783506	**447**	2335	1401	1,877941
408	2069	880	1,891041	**448**	2336	1153	1,789226
409	2070	1192	1,785325	**449**	2337	1250	1,742518
410	2071	1137	1,776287	**450**	2338	1359	1,778232
411	2071	794	1,692866	**451**	2351	1331	1,719552
412	2075	1032	1,877155	**452**	2362	1326	1,748857
413	2075	1032	1,877155	**453**	2365	1376	1,76481
414	2091	1206	1,746003	**454**	2369	1156	1,801735
415	2106	1099	1,751128	**455**	2372	1256	1,779065
416	2125	1176	1,894735	**456**	2385	1190	1,745951
417	2125	1176	1,894735	**457**	2400	1189	1,791744
418	2136	1095	1,723721	**458**	2405	1217	1,701786
419	2152	936	1,728931	**459**	2423	1229	1,727785
420	2154	1119	1,796079	**460**	2427	1307	1,856472
421	2166	1149	1,706605	**461**	2436	1251	1,72019
422	2172	1177	1,665304	**462**	2436	1251	1,72019
423	2179	1359	1,705731	**463**	2445	1463	1,74086
424	2190	1081	1,83398	**464**	2457	989	1,727375
425	2191	1250	1,816042	**465**	2472	1242	1,744652
426	2204	1392	1,732052	**466**	2502	1191	1,731502
427	2207	1132	1,798535	**467**	2519	1340	1,74572
428	2213	1204	1,770617	**468**	2521	1340	1,757919
429	2215	1132	1,826749	**469**	2538	1600	1,750548
430	2228	1126	1,728238	**470**	2541	1252	1,777796
431	2230	1156	1,742057	**471**	2541	1453	1,688642
432	2230	1152	1,74751	**472**	2548	1474	1,707709
433	2238	1175	1,760976	**473**	2566	1407	1,698833
434	2241	1168	1,754962	**474**	2570	996	1,708599
435	2245	1315	1,72163	**475**	2571	1241	1,890331
436	2250	1295	1,729046	**476**	2573	1281	1,722953
437	2256	1245	1,776182	**477**	2582	1346	1,836639
438	2263	1136	1,676989	**478**	2590	1532	1,716786
439	2283	1321	1,787507	**479**	2595	1258	1,821524
440	2284	1173	1,779408	**480**	2599	1321	1,807928

481	2600	1411	1,795871	**521**	2858	1438	1,7203
482	2601	1338	1,777624	**522**	2870	1326	1,835322
483	2614	1567	1,815374	**523**	2870	1326	1,835322
484	2617	1473	1,753607	**524**	2881	1139	1,714502
485	2630	1415	1,779275	**525**	2882	1601	1,759053
486	2644	1547	1,856399	**526**	2886	1480	1,714178
487	2669	1459	1,733141	**527**	2886	1495	1,747551
488	2671	1528	1,7262	**528**	2889	1313	1,722429
489	2685	1396	1,699746	**529**	2889	1048	1,72157
490	2691	1340	1,639815	**530**	2889	1048	1,72157
491	2694	1229	1,71743	**531**	2891	1521	1,709003
492	2695	1471	1,755777	**532**	2899	1405	1,747472
493	2698	1614	1,827507	**533**	2899	1440	1,711879
494	2707	1392	1,786954	**534**	2922	1617	1,698627
495	2725	1380	1,857591	**535**	2929	1528	1,776405
496	2726	1398	1,752068	**536**	2934	1719	1,688805
497	2726	1398	1,752068	**537**	2938	1529	1,786105
498	2730	1590	1,728618	**538**	2946	1504	1,718581
499	2731	1632	1,827407	**539**	2951	1204	1,767851
500	2733	1233	1,731824	**540**	2970	1587	1,736348
501	2733	1576	1,737251	**541**	2972	1542	1,735493
502	2740	1221	1,905835	**542**	2976	1325	1,757298
503	2751	1424	1,737244	**543**	2987	1489	1,786651
504	2758	1388	1,917203	**544**	2988	1470	1,726806
505	2771	1574	1,777819	**545**	2992	1472	1,747636
506	2771	1574	1,777819	**546**	3004	423	3,031337
507	2773	1154	1,732756	**547**	3021	1665	1,812565
508	2776	1394	1,742791	**548**	3024	1425	1,774353
509	2776	1688	1,790691	**549**	3026	1397	1,727364
510	2777	1020	1,696647	**550**	3026	1506	1,740776
511	2788	1410	1,763334	**551**	3036	1576	1,705228
512	2792	1354	1,719505	**552**	3037	1519	1,750737
513	2795	1329	1,883539	**553**	3047	1611	1,682581
514	2803	1492	1,763214	**554**	3048	1562	1,726511
515	2805	1563	1,755164	**555**	3048	1417	1,783227
516	2820	1577	1,74886	**556**	3050	1335	1,660131
517	2834	1243	1,846379	**557**	3053	1495	1,757572
518	2840	1352	1,720122	**558**	3056	1463	1,753642
519	2846	1322	1,751653	**559**	3057	1280	1,700975
520	2848	1367	1,719106	**560**	3058	1553	1,718279

561	3060	1354	1,720828	**601**	3359	1416	1,721867
562	3062	1602	1,696013	**602**	3366	1904	1,735343
563	3063	1664	1,722408	**603**	3374	1760	1,728491
564	3064	1484	1,754908	**604**	3375	1624	1,747862
565	3082	1084	1,748267	**605**	3377	1657	1,663618
566	3090	1354	1,745703	**606**	3388	1844	1,715758
567	3109	1522	1,738225	**607**	3389	1713	1,739849
568	3116	1449	1,74463	**608**	3395	1770	1,73941
569	3129	1569	1,763741	**609**	3415	1436	1,789966
570	3165	1319	1,770987	**610**	3415	1640	1,709098
571	3172	1396	1,716051	**611**	3415	1436	1,789966
572	3173	1655	1,708439	**612**	3419	1517	1,706391
573	3187	1590	1,707533	**613**	3420	1547	1,794192
574	3187	1590	1,707533	**614**	3420	1547	1,794192
575	3188	1466	1,757176	**615**	3430	1378	1,78597
576	3191	1339	1,746631	**616**	3451	1625	1,768064
577	3192	1588	1,733596	**617**	3459	1772	1,693895
578	3194	1628	1,71733	**618**	3463	1900	1,710775
579	3206	1433	1,77253	**619**	3467	1729	1,758623
580	3216	1568	1,711288	**620**	3468	1660	1,762447
581	3221	1486	1,757001	**621**	3488	1688	1,714043
582	3222	1701	1,762309	**622**	3509	1594	1,758656
583	3236	1646	1,77677	**623**	3515	1919	1,741046
584	3242	1360	1,782862	**624**	3521	1751	1,818755
585	3244	1694	1,745486	**625**	3529	1776	1,77398
586	3252	1512	1,713203	**626**	3542	1739	1,697283
587	3254	1732	1,701145	**627**	3561	1702	1,708974
588	3254	1618	1,783274	**628**	3570	1805	1,721039
589	3256	1645	1,807526	**629**	3570	1242	1,736031
590	3274	1716	1,730026	**630**	3571	1648	1,77051
591	3275	1653	1,731794	**631**	3583	1426	1,68271
592	3277	1772	1,731556	**632**	3584	1649	1,741442
593	3291	1863	1,719493	**633**	3622	1573	1,749493
594	3294	1824	1,709006	**634**	3625	1561	1,828104
595	3297	1676	1,816062	**635**	3645	1605	1,729979
596	3309	1602	1,746227	**636**	3652	1826	1,768936
597	3322	1630	1,695849	**637**	3657	1924	1,708112
598	3322	1630	1,695849	**638**	3659	1611	1,772651
599	3326	1408	1,829436	**639**	3659	1611	1,772651
600	3334	1756	1,729809	**640**	3681	1829	1,797477

641	3689	1733	1,714116	**681**	4044	2024	1,691746
642	3695	1803	1,776167	**682**	4054	1871	1,715951
643	3702	1768	1,799103	**683**	4057	1793	1,735897
644	3720	1703	1,699357	**684**	4059	1823	1,705989
645	3736	1980	1,72375	**685**	4062	1826	1,705973
646	3742	1887	1,748639	**686**	4079	1872	1,732841
647	3745	1858	1,6992	**687**	4080	1833	1,719083
648	3745	1871	1,729405	**688**	4081	2044	1,691664
649	3755	1952	1,81242	**689**	4082	1639	1,735837
650	3759	1909	1,689941	**690**	4083	1924	1,719236
651	3767	1465	1,703136	**691**	4091	1812	1,74375
652	3785	1787	1,721136	**692**	4115	1835	1,689711
653	3789	1801	1,723361	**693**	4116	1810	1,741334
654	3815	1860	1,698351	**694**	4118	1583	1,723524
655	3815	2176	1,718865	**695**	4118	2097	1,781301
656	3816	1601	1,720498	**696**	4120	1725	1,724078
657	3816	1852	1,722513	**697**	4135	1812	1,694015
658	3820	1973	1,744263	**698**	4138	1923	1,730446
659	3827	2079	1,710022	**699**	4138	1743	1,755706
660	3828	1865	1,768856	**700**	4152	1781	1,749826
661	3836	1681	1,747383	**701**	4157	1969	1,717614
662	3842	1471	1,690229	**702**	4163	1591	1,853192
663	3843	1803	1,75098	**703**	4175	2000	1,732435
664	3863	1942	1,739009	**704**	4176	1736	1,75772
665	3876	2088	1,707877	**705**	4182	2112	1,71043
666	3909	2005	1,747103	**706**	4218	1787	1,703521
667	3910	1953	1,706434	**707**	4240	2077	1,683076
668	3920	1889	1,752684	**708**	4247	1897	1,739047
669	3933	1999	1,708063	**709**	4250	2028	1,730019
670	3936	1699	1,872967	**710**	4251	1980	1,743494
671	3973	1554	1,752199	**711**	4252	1859	1,711138
672	3976	2140	1,700794	**712**	4252	2074	1,76966
673	3989	1884	1,75824	**713**	4257	1486	1,747208
674	4013	1877	1,710437	**714**	4258	2047	1,66013
675	4014	1675	1,668823	**715**	4270	2053	1,694091
676	4026	1792	1,797704	**716**	4279	1791	1,767453
677	4033	1559	1,726704	**717**	4288	2089	1,681988
678	4033	2119	1,740503	**718**	4300	2190	1,713497
679	4041	1985	1,686215	**719**	4310	1909	1,706518
680	4041	1985	1,686215	**720**	4319	1957	1,678158

721	4339	2147	1,668701	**761**	4694	2119	1,71515
722	4349	2197	1,695486	**762**	4707	1662	1,745761
723	4352	2309	1,717799	**763**	4708	2329	1,708946
724	4357	1825	1,761538	**764**	4710	1664	1,745745
725	4363	2200	1,747555	**765**	4719	2213	1,692437
726	4364	1913	1,698244	**766**	4720	1984	1,758404
727	4368	2081	1,725237	**767**	4744	1915	1,695148
728	4368	2008	1,703904	**768**	4759	2369	1,675702
729	4373	1785	1,7253	**769**	4777	1983	1,788658
730	4376	1635	1,767253	**770**	4794	2435	1,72373
731	4382	2046	1,689279	**771**	4798	1880	1,756255
732	4382	1954	1,749998	**772**	4798	1897	1,701361
733	4402	2009	1,752088	**773**	4801	2370	1,68005
734	4403	2177	1,739936	**774**	4801	2370	1,68005
735	4469	1886	1,75452	**775**	4822	2325	1,682797
736	4474	2169	1,718445	**776**	4825	2264	1,700959
737	4497	2131	1,743131	**777**	4833	2189	1,768382
738	4505	1756	1,731013	**778**	4834	1946	1,713026
739	4511	1945	1,718257	**779**	4836	1910	1,692014
740	4514	1942	1,719289	**780**	4854	2346	1,68271
741	4514	1942	1,719289	**781**	4865	2495	1,683555
742	4516	1877	1,696514	**782**	4872	2305	1,732063
743	4529	2725	1,689928	**783**	4873	2424	1,717444
744	4549	1875	1,707116	**784**	4875	2221	1,724173
745	4557	1791	1,739294	**785**	4884	2133	1,698955
746	4572	2186	1,791718	**786**	4894	1820	1,695072
747	4584	2209	1,688334	**787**	4912	2126	1,680966
748	4592	2387	1,730013	**788**	4935	2230	1,668698
749	4593	1806	1,711087	**789**	4936	2420	1,658174
750	4605	1772	1,772206	**790**	4954	1963	1,692533
751	4611	2126	1,737917	**791**	4954	1962	1,692538
752	4633	2298	1,719489	**792**	4957	2366	1,750216
753	4638	1766	1,763111	**793**	4962	2385	1,734002
754	4643	2114	1,753322	**794**	4965	2386	1,69958
755	4644	2267	1,704355	**795**	4973	2027	1,739864
756	4648	2431	1,73072	**796**	4996	2509	1,73129
757	4661	2089	1,706793	**797**	5006	2035	1,717535
758	4668	2350	1,715772	**798**	5060	2481	1,708908
759	4682	2035	1,746002	**799**	5068	2007	1,705544
760	4688	2130	1,684072	**800**	5073	2189	1,733638

801	5089	2522	1,71328	**841**	5473	2286	1,701071
802	5091	2182	1,730395	**842**	5478	2432	1,676818
803	5102	2103	1,753098	**843**	5485	2455	1,714371
804	5103	2902	1,727055	**844**	5496	2098	1,723442
805	5111	2175	1,722267	**845**	5507	2417	1,684571
806	5125	2187	1,740651	**846**	5507	2519	1,703848
807	5148	2839	1,694282	**847**	5513	2333	1,712082
808	5151	2468	1,757399	**848**	5518	2391	1,70944
809	5153	2162	1,716058	**849**	5521	2499	1,696745
810	5154	2301	1,68423	**850**	5526	2875	1,705131
811	5163	2271	1,686628	**851**	5532	2496	1,655314
812	5164	2020	1,725091	**852**	5559	2217	1,678435
813	5164	2007	1,680742	**853**	5581	2534	1,711473
814	5196	2215	1,705187	**854**	5586	2485	1,678059
815	5212	2017	1,707364	**855**	5598	2649	1,739031
816	5217	2275	1,722775	**856**	5631	2485	1,737154
817	5229	2416	1,729847	**857**	5643	2424	1,718457
818	5236	2337	1,704381	**858**	5650	2525	1,676037
819	5247	2112	1,715512	**859**	5655	2618	1,647849
820	5251	2589	1,694561	**860**	5658	2124	1,719192
821	5261	2807	1,730304	**861**	5668	2431	1,685631
822	5284	1998	1,670223	**862**	5675	2572	1,7175
823	5314	2219	1,689342	**863**	5677	2745	1,701415
824	5319	2730	1,714954	**864**	5678	2600	1,677696
825	5339	2312	1,730999	**865**	5729	2740	1,70611
826	5369	2729	1,70591	**866**	5730	2400	1,721948
827	5375	2731	1,721711	**867**	5733	2637	1,690255
828	5377	2378	1,735771	**868**	5737	2377	1,658092
829	5377	2299	1,747176	**869**	5739	2732	1,748259
830	5379	2695	1,740765	**870**	5743	2240	1,71713
831	5400	2314	1,708409	**871**	5748	2366	1,684955
832	5416	3017	1,665465	**872**	5751	2371	1,703627
833	5417	2181	1,691986	**873**	5761	2846	1,694997
834	5421	2483	1,693928	**874**	5771	2656	1,704921
835	5431	2504	1,69461	**875**	5772	2548	1,667486
836	5434	2515	1,69382	**876**	5776	2527	1,73801
837	5451	2079	1,727115	**877**	5783	2685	1,698061
838	5468	2150	1,784412	**878**	5791	2915	1,705411
839	5469	2510	1,691429	**879**	5808	2667	1,719904
840	5471	2483	1,734322	**880**	5814	2720	1,681465

881	5815	2371	1,699182	**921**	6207	2349	1,731895
882	5815	2371	1,699182	**922**	6232	2625	1,686159
883	5823	2652	1,72686	**923**	6232	2668	1,66102
884	5839	2935	1,710113	**924**	6234	3031	1,674693
885	5847	2247	1,703475	**925**	6245	2747	1,684035
886	5854	2449	1,685958	**926**	6248	2906	1,695669
887	5870	2462	1,717908	**927**	6249	2504	1,681399
888	5878	2306	1,676952	**928**	6296	2527	1,672899
889	5881	2134	1,671976	**929**	6308	2691	1,678235
890	5882	2573	1,691798	**930**	6311	2494	1,698774
891	5884	2839	1,693287	**931**	6313	3011	1,684297
892	5897	2546	1,694091	**932**	6335	2300	1,70839
893	5903	2951	1,699263	**933**	6351	3145	1,655429
894	5903	2951	1,699263	**934**	6351	2515	1,715844
895	5920	2518	1,684609	**935**	6355	2750	1,691123
896	5922	2593	1,691729	**936**	6383	2589	1,708871
897	5938	2886	1,700568	**937**	6417	3207	1,688205
898	5945	2595	1,671365	**938**	6422	2961	1,702562
899	5956	2615	1,693962	**939**	6441	2768	1,720869
900	5971	2532	1,717644	**940**	6441	2623	1,696182
901	5982	2629	1,714172	**941**	6443	2661	1,667587
902	6006	2526	1,717756	**942**	6443	2661	1,667587
903	6010	2999	1,706511	**943**	6443	2630	1,696156
904	6019	2710	1,656736	**944**	6451	2854	1,679024
905	6028	2462	1,714302	**945**	6455	2901	1,694642
906	6029	2987	1,700649	**946**	6460	2218	1,719662
907	6057	2307	1,718059	**947**	6465	2739	1,657587
908	6057	2668	1,670026	**948**	6468	2872	1,65721
909	6063	3518	1,644515	**949**	6487	2642	1,703343
910	6078	2407	1,788433	**950**	6494	2695	1,700931
911	6087	2987	1,668199	**951**	6497	2237	1,719554
912	6120	2772	1,719999	**952**	6501	2846	1,652285
913	6134	2714	1,683986	**953**	6503	2387	1,690124
914	6142	3163	1,672648	**954**	6509	2537	1,670145
915	6142	3163	1,672648	**955**	6519	2593	1,672413
916	6150	3086	1,688296	**956**	6528	3131	1,691251
917	6154	2512	1,738487	**957**	6529	2781	1,694931
918	6156	2795	1,689226	**958**	6538	2413	1,685823
919	6157	2639	1,664259	**959**	6547	2591	1,716185
920	6166	2543	1,723027	**960**	6552	2818	1,670049

961	6552	2696	1,672254	**1001**	7047	2557	1,683361
962	6556	2782	1,69779	**1002**	7047	2705	1,692748
963	6561	2357	1,674592	**1003**	7048	2709	1,692734
964	6567	2843	1,699336	**1004**	7055	2633	1,742552
965	6576	2595	1,691185	**1005**	7087	2761	1,723832
966	6582	3253	1,684135	**1006**	7089	3204	1,671726
967	6601	2643	1,697644	**1007**	7089	2663	1,706911
968	6602	3026	1,666164	**1008**	7110	2967	1,719266
969	6615	2822	1,665187	**1009**	7113	2992	1,684865
970	6620	2953	1,674253	**1010**	7122	3263	1,645352
971	6665	2654	1,690103	**1011**	7129	3157	1,677642
972	6677	2920	1,672341	**1012**	7147	3302	1,683086
973	6696	2818	1,703294	**1013**	7159	2831	1,682476
974	6724	2180	1,677186	**1014**	7174	3181	1,664714
975	6738	2980	1,668715	**1015**	7178	2869	1,661709
976	6742	2993	1,68883	**1016**	7190	2769	1,675669
977	6744	2973	1,727949	**1017**	7203	3202	1,679946
978	6757	2920	1,675482	**1018**	7209	3265	1,685402
979	6758	2763	1,687419	**1019**	7214	3327	1,72354
980	6768	2462	1,734123	**1020**	7218	3257	1,707231
981	6774	3342	1,683818	**1021**	7257	3202	1,716493
982	6782	2900	1,723591	**1022**	7257	3202	1,716493
983	6785	3193	1,672039	**1023**	7258	2831	1,731
984	6796	3054	1,68921	**1024**	7259	3170	1,660963
985	6799	2648	1,677749	**1025**	7259	2936	1,69564
986	6800	2949	1,701159	**1026**	7280	2833	1,741758
987	6811	2838	1,707928	**1027**	7283	3105	1,69131
988	6906	3226	1,695985	**1028**	7317	3313	1,680738
989	6914	2877	1,664107	**1029**	7329	3314	1,664394
990	6923	2983	1,677354	**1030**	7362	2412	1,713098
991	6934	2924	1,663446	**1031**	7370	2824	1,626322
992	6947	2256	1,690497	**1032**	7372	2726	1,667914
993	6950	2490	1,703352	**1033**	7377	3291	1,653104
994	6969	3166	1,645751	**1034**	7403	3128	1,69062
995	6970	3214	1,718763	**1035**	7407	3241	1,685006
996	7003	2854	1,725945	**1036**	7409	3255	1,690426
997	7026	3336	1,674896	**1037**	7426	2875	1,680442
998	7034	2057	1,719124	**1038**	7440	3348	1,679274
999	7040	2849	1,690462	**1039**	7440	3348	1,679274
1000	7040	2928	1,656525	**1040**	7454	2785	1,671199

1041	7472	3097	1,671663	**1081**	8118	3377	1,664203
1042	7488	2734	1,690786	**1082**	8130	3340	1,675767
1043	7493	3288	1,703434	**1083**	8135	3188	1,705859
1044	7516	2661	1,715895	**1084**	8146	3281	1,727668
1045	7556	2525	1,713293	**1085**	8147	3415	1,67275
1046	7573	2640	1,67754	**1086**	8166	3811	1,664822
1047	7580	3300	1,695979	**1087**	8184	3307	1,657378
1048	7628	2980	1,709645	**1088**	8200	3110	1,680859
1049	7638	3400	1,689773	**1089**	8208	3197	1,688117
1050	7656	3213	1,693418	**1090**	8212	3295	1,691564
1051	7663	2888	1,661572	**1091**	8226	2829	1,715231
1052	7665	2659	1,672328	**1092**	8267	2746	1,665449
1053	7670	3648	1,670272	**1093**	8273	2971	1,674097
1054	7692	2844	1,671609	**1094**	8275	3870	1,702751
1055	7717	3350	1,669638	**1095**	8297	3343	1,667901
1056	7734	3219	1,685969	**1096**	8313	3426	1,707726
1057	7743	4019	1,67025	**1097**	8362	3119	1,722408
1058	7758	2802	1,701079	**1098**	8373	3775	1,675518
1059	7802	3400	1,654743	**1099**	8387	3193	1,666043
1060	7804	2769	1,690659	**1100**	8442	3655	1,668405
1061	7817	3677	1,655385	**1101**	8448	3080	1,658657
1062	7827	30	3,004255	**1102**	8464	3883	1,658269
1063	7829	3570	1,686892	**1103**	8468	3261	1,685125
1064	7852	3554	1,659935	**1104**	8484	3019	1,70082
1065	7862	3141	1,660048	**1105**	8512	3192	1,678965
1066	7878	3478	1,67719	**1106**	8529	3431	1,648913
1067	7908	2961	1,700666	**1107**	8549	3324	1,665672
1068	7959	3504	1,695367	**1108**	8549	3211	1,678215
1069	7959	3506	1,695363	**1109**	8551	3930	1,655901
1070	7962	3328	1,673838	**1110**	8555	3948	1,655626
1071	7978	3723	1,686191	**1111**	8583	3633	1,686406
1072	7979	3294	1,702464	**1112**	8640	3653	1,646466
1073	8003	3152	1,709356	**1113**	8650	3427	1,701533
1074	8011	2785	1,669282	**1114**	8659	3452	1,682953
1075	8050	3465	1,67331	**1115**	8668	3390	1,655166
1076	8068	3370	1,703588	**1116**	8683	3372	1,679165
1077	8095	3177	1,689056	**1117**	8690	3373	1,679163
1078	8100	3762	1,664514	**1118**	8698	3714	1,675302
1079	8115	3193	1,689677	**1119**	8702	2965	1,683684
1080	8115	3193	1,689677	**1120**	8708	3654	1,646837

1121	8711	3294	1,663763	**1161**	9327	3923	1,678368
1122	8751	3445	1,649446	**1162**	9336	3066	1,6786
1123	8763	3771	1,660344	**1163**	9359	3533	1,682231
1124	8775	3550	1,661912	**1164**	9360	2896	1,678972
1125	8775	3266	1,683505	**1165**	9374	4037	1,674367
1126	8775	3948	1,689252	**1166**	9417	3136	1,646488
1127	8778	3196	1,679952	**1167**	9440	3513	1,6954
1128	8778	3196	1,679952	**1168**	9457	2965	1,680538
1129	8801	3588	1,698262	**1169**	9457	2965	1,680538
1130	8825	2880	1,753314	**1170**	9469	3859	1,676031
1131	8845	3650	1,653488	**1171**	9544	3745	1,685784
1132	8857	3054	1,718213	**1172**	9549	4242	1,695979
1133	8868	3810	1,635687	**1173**	9556	3019	1,685037
1134	8906	3512	1,672704	**1174**	9597	3713	1,64354
1135	8934	3358	1,715341	**1175**	9600	3716	1,643533
1136	8947	3929	1,666876	**1176**	9609	3990	1,652887
1137	8954	3018	1,686588	**1177**	9702	3478	1,672044
1138	8959	3225	1,696394	**1178**	9724	3684	1,680298
1139	8970	3718	1,668479	**1179**	9749	3634	1,675853
1140	8979	3343	1,680824	**1180**	9775	3772	1,657806
1141	8983	3662	1,645747	**1181**	9805	3508	1,669695
1142	8987	3175	1,721572	**1182**	9810	3709	1,668429
1143	8991	3750	1,685978	**1183**	9820	4353	1,65448
1144	9010	2826	1,688645	**1184**	9838	4126	1,65232
1145	9013	4004	1,658913	**1185**	9867	3569	1,639666
1146	9020	3482	1,664064	**1186**	9875	3998	1,687605
1147	9075	3405	1,705513	**1187**	9876	4445	1,663608
1148	9091	3525	1,666905	**1188**	9895	4672	1,673776
1149	9120	4550	1,634428	**1189**	9910	4231	1,677368
1150	9145	3672	1,687848	**1190**	9910	4231	1,677368
1151	9152	3616	1,73644	**1191**	9918	4290	1,681991
1152	9199	3672	1,664971	**1192**	9920	3803	1,678979
1153	9201	3446	1,700383	**1193**	9945	3451	1,644666
1154	9217	3427	1,687301	**1194**	10003	4489	1,656961
1155	9227	3857	1,69197	**1195**	10022	3837	1,66592
1156	9233	3988	1,669715	**1196**	10049	4063	1,693073
1157	9249	3477	1,649864	**1197**	10065	2631	1,737241
1158	9270	3323	1,669062	**1198**	10066	3800	1,6714
1159	9278	3702	1,682879	**1199**	10069	3428	1,658493
1160	9324	3258	1,658057	**1200**	10081	4610	1,71407

1201	10093	3627	1,662109	**1241**	10655	3896	1,657458
1202	10108	3819	1,671357	**1242**	10666	3793	1,643191
1203	10144	3589	1,694129	**1243**	10741	3752	1,664086
1204	10168	3797	1,662842	**1244**	10770	4465	1,645331
1205	10191	3750	1,707876	**1245**	10793	3842	1,657151
1206	10209	3785	1,662222	**1246**	10795	4883	1,686358
1207	10218	4049	1,653538	**1247**	10844	4136	1,68913
1208	10219	3782	1,639204	**1248**	10876	5088	1,66312
1209	10219	3782	1,639204	**1249**	10888	4488	1,671644
1210	10220	3526	1,683825	**1250**	10888	4486	1,671359
1211	10231	4017	1,642515	**1251**	10897	3789	1,660818
1212	10232	4385	1,659902	**1252**	10914	4674	1,655646
1213	10237	4016	1,678177	**1253**	10950	3643	1,668646
1214	10251	3707	1,659702	**1254**	10976	4143	1,66987
1215	10271	4293	1,665134	**1255**	10981	3488	1,685913
1216	10289	4140	1,651985	**1256**	11020	4359	1,644103
1217	10319	3745	1,707062	**1257**	11038	3935	1,664124
1218	10333	4376	1,65457	**1258**	11093	4265	1,668958
1219	10354	4132	1,677535	**1259**	11106	4090	1,678849
1220	10381	3811	1,653029	**1260**	11135	4419	1,667011
1221	10406	4157	1,662634	**1261**	11136	3788	1,681201
1222	10411	3740	1,687833	**1262**	11154	4322	1,668786
1223	10414	3509	1,724267	**1263**	11158	4567	1,675155
1224	10417	4166	1,662619	**1264**	11171	2494	1,639285
1225	10440	4214	1,649159	**1265**	11208	4353	1,698238
1226	10464	4212	1,660911	**1266**	11258	4162	1,697502
1227	10491	4670	1,687268	**1267**	11296	5378	1,648233
1228	10503	3925	1,68105	**1268**	11324	4663	1,666152
1229	10525	3348	1,692666	**1269**	11347	4698	1,664132
1230	10527	4288	1,669492	**1270**	11364	4447	1,672243
1231	10532	3285	1,660503	**1271**	11386	4468	1,650257
1232	10567	4575	1,665461	**1272**	11396	3073	1,686723
1233	10569	4008	1,651119	**1273**	11398	3898	1,665897
1234	10580	3990	1,6516	**1274**	11418	4315	1,653422
1235	10598	4073	1,664268	**1275**	11420	4575	1,65594
1236	10600	4277	1,679895	**1276**	11428	4403	1,658348
1237	10606	3669	1,657536	**1277**	11460	3622	1,671955
1238	10630	4160	1,655149	**1278**	11484	5078	1,640619
1239	10636	3967	1,671408	**1279**	11499	4276	1,689016
1240	10639	4122	1,650894	**1280**	11538	4476	1,679598

1281	11566	2708	1,638056	**1321**	12755	5132	1,659212
1282	11576	3942	1,668675	**1322**	12777	4509	1,65675
1283	11588	3929	1,680024	**1323**	12790	4634	1,671148
1284	11592	5169	1,678122	**1324**	12806	4771	1,651291
1285	11604	4786	1,668083	**1325**	12835	4768	1,672533
1286	11692	4616	1,659646	**1326**	12839	4624	1,657889
1287	11716	4131	1,674292	**1327**	12902	4533	1,660494
1288	11778	4645	1,666883	**1328**	12930	5178	1,645198
1289	11798	4845	1,655251	**1329**	13001	4249	1,663063
1290	11839	5294	1,679142	**1330**	13025	4270	1,647456
1291	11848	2910	1,666283	**1331**	13063	5566	1,66293
1292	11884	5169	1,675172	**1332**	13073	4253	1,664749
1293	11917	4048	1,638291	**1333**	13081	5027	1,670882
1294	11974	4688	1,671456	**1334**	13104	4605	1,640107
1295	12050	4516	1,645825	**1335**	13116	4520	1,651522
1296	12060	3216	1,678346	**1336**	13149	4791	1,656831
1297	12148	4258	1,661727	**1337**	13154	4652	1,649063
1298	12154	4158	1,665423	**1338**	13166	5063	1,642228
1299	12176	4652	1,639605	**1339**	13168	4452	1,652273
1300	12189	4989	1,66701	**1340**	13171	4638	1,652006
1301	12228	4907	1,674609	**1341**	13207	5587	1,643663
1302	12246	5012	1,647712	**1342**	13216	4152	1,65648
1303	12272	4423	1,644483	**1343**	13216	4152	1,65648
1304	12273	3874	1,716218	**1344**	13218	4034	1,667214
1305	12274	5040	1,654515	**1345**	13223	4943	1,659523
1306	12339	4765	1,639021	**1346**	13247	5011	1,648836
1307	12340	4834	1,655009	**1347**	13284	4135	1,650808
1308	12341	3635	1,658102	**1348**	13285	4661	1,655793
1309	12354	4341	1,681207	**1349**	13289	4349	1,650364
1310	12368	4046	1,660997	**1350**	13345	4694	1,647044
1311	12376	4435	1,700326	**1351**	13354	5274	1,630833
1312	12471	4318	1,66558	**1352**	13354	4384	1,666877
1313	12486	4189	1,660771	**1353**	13389	4484	1,660081
1314	12509	4170	1,656313	**1354**	13424	5004	1,656656
1315	12509	4170	1,656313	**1355**	13445	4660	1,652247
1316	12538	4930	1,64848	**1356**	13456	4627	1,65607
1317	12622	4225	1,680173	**1357**	13481	3521	1,639072
1318	12645	5354	1,654318	**1358**	13521	4611	1,664912
1319	12652	4677	1,680762	**1359**	13567	4392	1,650941
1320	12699	4643	1,677831	**1360**	13567	4392	1,650941

1361	13568	4364	1,664584	**1401**	14563	4219	1,615123
1362	13576	4860	1,662465	**1402**	14564	5458	1,653152
1363	13614	4426	1,647666	**1403**	14578	6103	1,659615
1364	13630	4865	1,639575	**1404**	14578	6084	1,659633
1365	13652	4496	1,656971	**1405**	14616	5173	1,641549
1366	13657	5647	1,6538	**1406**	14620	6062	1,642489
1367	13664	4236	1,650292	**1407**	14653	5339	1,638795
1368	13689	4977	1,65333	**1408**	14768	4528	1,652192
1369	13709	4539	1,650167	**1409**	14780	5096	1,63184
1370	13739	4190	1,652271	**1410**	14794	4914	1,616995
1371	13743	5144	1,659624	**1411**	14851	5888	1,657732
1372	13876	5596	1,670402	**1412**	14868	4655	1,674526
1373	13889	5437	1,668644	**1413**	14945	4595	1,671374
1374	13898	4680	1,670482	**1414**	14990	5719	1,654494
1375	13905	4734	1,632256	**1415**	15002	4071	1,675302
1376	13906	5752	1,663262	**1416**	15021	4784	1,646468
1377	13915	5224	1,672565	**1417**	15074	4773	1,642317
1378	13922	5477	1,659437	**1418**	15074	4773	1,642317
1379	13924	5463	1,642374	**1419**	15121	5223	1,649515
1380	13938	5079	1,633308	**1420**	15177	4745	1,648224
1381	13958	3936	1,659966	**1421**	15177	4745	1,648224
1382	13990	4816	1,640252	**1422**	15209	5822	1,656129
1383	14010	4921	1,659071	**1423**	15215	5011	1,668456
1384	14045	5596	1,659294	**1424**	15229	4823	1,636805
1385	14113	6266	1,627912	**1425**	15232	4644	1,652997
1386	14166	4763	1,649009	**1426**	15232	3995	1,66668
1387	14166	4763	1,649009	**1427**	15238	3997	1,666675
1388	14182	3923	1,664241	**1428**	15315	4833	1,6492
1389	14182	3923	1,664241	**1429**	15363	6252	1,651034
1390	14228	4327	1,65702	**1430**	15385	4640	1,659311
1391	14232	4859	1,643153	**1431**	15418	4561	1,658001
1392	14237	4500	1,64282	**1432**	15423	4743	1,656255
1393	14295	4843	1,677647	**1433**	15435	5261	1,642028
1394	14377	3856	1,668471	**1434**	15516	5187	1,660599
1395	14413	4500	1,656661	**1435**	15607	4996	1,64707
1396	14413	5117	1,650776	**1436**	15652	5671	1,632259
1397	14500	5008	1,639406	**1437**	15705	5555	1,642896
1398	14500	5008	1,639406	**1438**	15709	5792	1,643582
1399	14514	4924	1,647683	**1439**	15719	5128	1,673264
1400	14545	5160	1,649131	**1440**	15722	4663	1,663488

1441	15744	33	2,941677	**1481**	17448	6726	1,644045
1442	15795	5676	1,635432	**1482**	17460	5597	1,645394
1443	15834	5043	1,621317	**1483**	17460	5597	1,645394
1444	15867	5029	1,662606	**1484**	17466	6029	1,646669
1445	15888	4821	1,663195	**1485**	17482	5395	1,633559
1446	15900	5523	1,648767	**1486**	17621	4191	1,666792
1447	15913	4944	1,652618	**1487**	17639	6042	1,643494
1448	15917	4930	1,658178	**1488**	17667	5940	1,653477
1449	15949	5095	1,632932	**1489**	17667	5940	1,653477
1450	15968	4875	1,653699	**1490**	17723	5413	1,62953
1451	16004	5588	1,660418	**1491**	17831	5955	1,64864
1452	16028	5069	1,647032	**1492**	17851	5552	1,633495
1453	16138	4766	1,653928	**1493**	18029	6704	1,632831
1454	16252	4629	1,642233	**1494**	18051	5889	1,640227
1455	16372	5837	1,663038	**1495**	18058	5473	1,642455
1456	16383	3767	1,637664	**1496**	18145	5755	1,643687
1457	16423	5725	1,640552	**1497**	18189	5874	1,638779
1458	16423	5725	1,640552	**1498**	18270	6147	1,645404
1459	16525	7001	1,658015	**1499**	18307	3902	1,673766
1460	16575	6029	1,633615	**1500**	18321	6790	1,637946
1461	16595	7297	1,622672	**1501**	18337	6759	1,626077
1462	16604	5219	1,65429	**1502**	18345	5898	1,65705
1463	16605	4529	1,675094	**1503**	18346	5860	1,657418
1464	16624	6848	1,650792	**1504**	18389	5969	1,646853
1465	16642	6689	1,627248	**1505**	18395	5811	1,652128
1466	16652	5546	1,65506	**1506**	18410	7158	1,633571
1467	16707	6528	1,643025	**1507**	18451	6266	1,639425
1468	16720	5294	1,657298	**1508**	18461	6581	1,629337
1469	16943	5092	1,662225	**1509**	18464	7343	1,637225
1470	17088	6622	1,636924	**1510**	18467	4850	1,641363
1471	17130	5773	1,640826	**1511**	18510	5946	1,627167
1472	17156	6090	1,632078	**1512**	18526	5591	1,670835
1473	17193	5842	1,646415	**1513**	18528	5959	1,653023
1474	17199	5947	1,651527	**1514**	18551	5581	1,659022
1475	17258	6216	1,665501	**1515**	18562	5444	1,615078
1476	17326	6037	1,648707	**1516**	18588	6906	1,650072
1477	17367	5907	1,649511	**1517**	18616	5521	1,642792
1478	17377	6641	1,639176	**1518**	18665	6205	1,643712
1479	17396	6016	1,650649	**1519**	18698	6458	1,637056
1480	17442	6644	1,639174	**1520**	18751	5725	1,665236

1521	18819	6413	1,647367	**1561**	20352	6233	1,62757
1522	18837	5548	1,659511	**1562**	20375	5866	1,659011
1523	18843	6721	1,633947	**1563**	20385	6450	1,634352
1524	18868	6497	1,64305	**1564**	20405	7595	1,627262
1525	18921	5919	1,646241	**1565**	20439	5424	1,63302
1526	18936	6727	1,644483	**1566**	20535	5611	1,636163
1527	19083	5552	1,655319	**1567**	20555	4494	1,657258
1528	19094	4903	1,655092	**1568**	20573	5454	1,634667
1529	19146	5946	1,639631	**1569**	20594	6785	1,616507
1530	19161	6328	1,630724	**1570**	20609	5831	1,641455
1531	19246	6881	1,631808	**1571**	20629	5532	1,634022
1532	19370	5960	1,637231	**1572**	20642	5700	1,6359
1533	19383	6488	1,644143	**1573**	20650	5925	1,649709
1534	19426	6153	1,651446	**1574**	20658	6176	1,627937
1535	19510	6669	1,645018	**1575**	20663	5603	1,63637
1536	19560	6869	1,641321	**1576**	20721	7161	1,641957
1537	19587	6341	1,651913	**1577**	20776	6120	1,639229
1538	19605	5947	1,63343	**1578**	20777	7219	1,634517
1539	19613	3700	1,659457	**1579**	20790	5335	1,652683
1540	19615	5265	1,638598	**1580**	20796	5382	1,649508
1541	19618	5719	1,632954	**1581**	20800	5795	1,626257
1542	19630	5667	1,635045	**1582**	20817	6660	1,637249
1543	19660	5786	1,646566	**1583**	20818	5767	1,646283
1544	19684	5587	1,638059	**1584**	20834	5779	1,646267
1545	19740	5609	1,636077	**1585**	20837	5392	1,644353
1546	19756	5419	1,642376	**1586**	20839	5514	1,641537
1547	19794	5509	1,634912	**1587**	20852	6134	1,641287
1548	19807	5761	1,632533	**1588**	20872	5970	1,636512
1549	20036	5522	1,640109	**1589**	20885	6055	1,631828
1550	20040	7347	1,651739	**1590**	20895	5226	1,647926
1551	20082	6720	1,621557	**1591**	20905	5154	1,645824
1552	20120	7074	1,637454	**1592**	20921	5792	1,6437
1553	20135	6712	1,63357	**1593**	20928	5765	1,632163
1554	20180	5976	1,644013	**1594**	20964	5822	1,628543
1555	20191	5447	1,654782	**1595**	20964	5525	1,637819
1556	20263	6864	1,645836	**1596**	20989	6371	1,628137
1557	20304	6527	1,638116	**1597**	20996	6023	1,633273
1558	20307	5149	1,64562	**1598**	21003	6085	1,639152
1559	20317	5244	1,637688	**1599**	21041	5816	1,640518
1560	20321	4656	1,645875	**1600**	21059	5998	1,635604

1601	21059	5930	1,638886	**1641**	21634	5955	1,638578
1602	21061	7409	1,659744	**1642**	21639	5778	1,634852
1603	21067	5559	1,637171	**1643**	21648	6096	1,640116
1604	21090	5994	1,640413	**1644**	21672	5813	1,63375
1605	21120	5291	1,645202	**1645**	21686	5747	1,630632
1606	21143	5640	1,637816	**1646**	21700	7333	1,634896
1607	21172	6069	1,636633	**1647**	21713	5650	1,639298
1608	21173	5546	1,638902	**1648**	21718	5882	1,642813
1609	21193	6103	1,638555	**1649**	21730	7271	1,654453
1610	21213	5894	1,644705	**1650**	21749	5719	1,640948
1611	21230	5986	1,634626	**1651**	21756	6039	1,625563
1612	21256	5807	1,630152	**1652**	21763	6199	1,627612
1613	21291	7645	1,621263	**1653**	21786	5715	1,637852
1614	21308	6895	1,633209	**1654**	21803	6920	1,628794
1615	21310	4719	1,650963	**1655**	21803	5982	1,639819
1616	21313	5566	1,637683	**1656**	21804	5033	1,650916
1617	21327	5893	1,643125	**1657**	21806	6270	1,637924
1618	21332	5522	1,63654	**1658**	21815	5803	1,638953
1619	21346	5498	1,64021	**1659**	21828	6673	1,63355
1620	21394	5782	1,638893	**1660**	21903	6123	1,635023
1621	21408	5339	1,640485	**1661**	21908	6049	1,629984
1622	21409	5723	1,632598	**1662**	21937	5787	1,64059
1623	21441	5331	1,62451	**1663**	21938	7288	1,638769
1624	21460	6037	1,633929	**1664**	21938	5951	1,630012
1625	21462	5644	1,63863	**1665**	21964	6296	1,654033
1626	21468	6904	1,634232	**1666**	21970	6160	1,634858
1627	21478	6943	1,644203	**1667**	22002	4958	1,6269
1628	21512	5812	1,637436	**1668**	22005	6887	1,635766
1629	21513	6870	1,646291	**1669**	22015	6042	1,627807
1630	21527	6236	1,629358	**1670**	22015	7223	1,624513
1631	21535	7009	1,646759	**1671**	22018	4917	1,644971
1632	21537	7073	1,631459	**1672**	22020	5684	1,637969
1633	21548	5902	1,63601	**1673**	22074	5645	1,632193
1634	21585	5780	1,643879	**1674**	22105	6189	1,632104
1635	21588	5980	1,641117	**1675**	22132	6055	1,638352
1636	21592	7072	1,633515	**1676**	22161	5907	1,63765
1637	21593	5193	1,637466	**1677**	22173	6674	1,638645
1638	21602	5916	1,635169	**1678**	22186	5794	1,638079
1639	21613	5176	1,62919	**1679**	22210	5542	1,631888
1640	21614	7032	1,689599	**1680**	22297	7066	1,626831

1681	22306	6187	1,638054	**1721**	23141	6292	1,630717
1682	22333	6497	1,628056	**1722**	23151	5758	1,638364
1683	22398	6838	1,645273	**1723**	23157	6258	1,635737
1684	22406	6124	1,631524	**1724**	23207	5108	1,660623
1685	22432	7239	1,646476	**1725**	23227	6999	1,637138
1686	22443	5779	1,64341	**1726**	23246	5454	1,651247
1687	22477	6439	1,630961	**1727**	23254	6240	1,6427
1688	22497	5995	1,640659	**1728**	23275	9043	1,632321
1689	22533	6463	1,633082	**1729**	23275	9043	1,632321
1690	22552	6308	1,633077	**1730**	23292	7806	1,642947
1691	22576	5956	1,633357	**1731**	23310	6984	1,631683
1692	22588	7046	1,630598	**1732**	23364	5969	1,632218
1693	22621	6017	1,636409	**1733**	23400	5983	1,637367
1694	22627	6071	1,630717	**1734**	23434	6095	1,641332
1695	22652	5864	1,637571	**1735**	23445	6710	1,646083
1696	22663	6286	1,638984	**1736**	23517	6227	1,636516
1697	22689	6575	1,617378	**1737**	23548	6952	1,62604
1698	22700	5706	1,640273	**1738**	23551	6087	1,636293
1699	22760	5901	1,632204	**1739**	23567	7658	1,647088
1700	22762	6360	1,635592	**1740**	23574	6266	1,63043
1701	22780	5566	1,641149	**1741**	23584	6514	1,633369
1702	22794	6917	1,637681	**1742**	23593	3466	1,618775
1703	22797	4953	1,661146	**1743**	23597	6373	1,64561
1704	22807	6275	1,636112	**1744**	23619	6705	1,634594
1705	22840	5829	1,632785	**1745**	23620	5878	1,6329
1706	22850	8994	1,613177	**1746**	23620	6486	1,639919
1707	22879	7557	1,635528	**1747**	23635	6633	1,643104
1708	22883	6457	1,631917	**1748**	23636	6787	1,628192
1709	22920	6448	1,627127	**1749**	23638	7495	1,633192
1710	22947	8184	1,627321	**1750**	23638	7495	1,633192
1711	22954	6397	1,627789	**1751**	23721	6918	1,633809
1712	22974	8121	1,629166	**1752**	23722	6844	1,637129
1713	22975	8006	1,624479	**1753**	23726	6625	1,630565
1714	23000	7072	1,636178	**1754**	23729	6187	1,641734
1715	23032	6651	1,628047	**1755**	23731	6437	1,638888
1716	23063	6062	1,636132	**1756**	23742	6897	1,624801
1717	23068	7520	1,62211	**1757**	23760	5899	1,634627
1718	23102	6123	1,637512	**1758**	23770	7439	1,627327
1719	23107	7379	1,63287	**1759**	23780	6093	1,63995
1720	23132	6588	1,636816	**1760**	23827	7305	1,633942

1761	23877	6346	1,632909	**1801**	25143	7626	1,629636
1762	23926	7721	1,63112	**1802**	25189	7632	1,644925
1763	24013	6114	1,645367	**1803**	25278	7294	1,648467
1764	24089	6112	1,629718	**1804**	25362	6891	1,630616
1765	24098	8697	1,633115	**1805**	25399	7151	1,629122
1766	24190	6414	1,637568	**1806**	25467	5375	1,607611
1767	24192	7514	1,626491	**1807**	25482	8012	1,632575
1768	24192	7514	1,626491	**1808**	25489	7729	1,630448
1769	24197	6419	1,641841	**1809**	25492	8187	1,634088
1770	24233	6268	1,638441	**1810**	25513	8981	1,649186
1771	24249	6423	1,668646	**1811**	25518	5741	1,638119
1772	24285	5970	1,636125	**1812**	25551	6847	1,626769
1773	24293	8957	1,638495	**1813**	25553	6539	1,6616
1774	24324	5978	1,632209	**1814**	25554	7213	1,627489
1775	24337	7878	1,626737	**1815**	25574	8225	1,643981
1776	24349	6520	1,638892	**1816**	25620	7593	1,625483
1777	24350	6830	1,638796	**1817**	25713	8472	1,642107
1778	24486	7785	1,633792	**1818**	25751	6897	1,627816
1779	24503	7579	1,626808	**1819**	25782	6947	1,630265
1780	24503	7579	1,626808	**1820**	25830	7438	1,626844
1781	24563	6820	1,639154	**1821**	25843	6587	1,634181
1782	24575	6144	1,63259	**1822**	25869	6464	1,635293
1783	24603	6831	1,63074	**1823**	25873	7231	1,621362
1784	24611	7722	1,638941	**1824**	25879	6236	1,633407
1785	24637	5831	1,645975	**1825**	25900	9049	1,628427
1786	24637	5831	1,645975	**1826**	25952	7985	1,621439
1787	24649	6609	1,638163	**1827**	25955	5410	1,663423
1788	24657	6253	1,630646	**1828**	25974	7796	1,640573
1789	24690	6908	1,623175	**1829**	25986	7645	1,625645
1790	24720	5973	1,631932	**1830**	26024	7872	1,629254
1791	24813	7137	1,631068	**1831**	26045	6834	1,646671
1792	24817	7886	1,62556	**1832**	26046	6929	1,631198
1793	24888	6665	1,649387	**1833**	26057	7624	1,630202
1794	24905	6120	1,621952	**1834**	26108	6530	1,628727
1795	24952	6871	1,623634	**1835**	26127	7773	1,624979
1796	24970	7370	1,629993	**1836**	26132	7275	1,631871
1797	24988	7534	1,636506	**1837**	26166	7574	1,627636
1798	25001	6592	1,640621	**1838**	26166	7574	1,627636
1799	25047	7570	1,631578	**1839**	26187	6832	1,646351
1800	25094	7450	1,627907	**1840**	26188	7837	1,629951

1841	26189	8433	1,630318	**1881**	28521	7220	1,628492
1842	26260	8625	1,622543	**1882**	28522	7440	1,64129
1843	26342	6254	1,642802	**1883**	28578	10023	1,618546
1844	26349	8728	1,647774	**1884**	28592	8680	1,620403
1845	26351	8096	1,627433	**1885**	28615	8423	1,627546
1846	26409	8753	1,62676	**1886**	28638	8219	1,627997
1847	26412	7396	1,633789	**1887**	28962	6843	1,623881
1848	26427	8722	1,642895	**1888**	28962	6843	1,623881
1849	26428	7095	1,623113	**1889**	28991	6553	1,628373
1850	26474	7134	1,635562	**1890**	29007	9080	1,612074
1851	26484	7740	1,625963	**1891**	29050	8830	1,630154
1852	26500	6082	1,635107	**1892**	29098	7339	1,642381
1853	26690	8668	1,638401	**1893**	29188	7323	1,628715
1854	26731	8700	1,631347	**1894**	29207	8192	1,603084
1855	26892	8613	1,629554	**1895**	29231	7096	1,627103
1856	26962	8976	1,621459	**1896**	29356	7068	1,621717
1857	27001	7722	1,623117	**1897**	29640	6482	1,63176
1858	27196	7337	1,620368	**1898**	29766	6572	1,64103
1859	27344	8903	1,627346	**1899**	29825	7397	1,618243
1860	27442	5645	1,650157	**1900**	30764	9052	1,613705
1861	27482	7943	1,605727	**1901**	32523	9147	1,623908
1862	27511	8107	1,620506				
1863	27612	7696	1,624001				
1864	27615	7078	1,645358				
1865	27617	7266	1,626747				
1866	27690	6208	1,616686				
1867	27819	8193	1,618404				
1868	27859	8017	1,6057				
1869	27870	5293	1,63279				
1870	27969	7163	1,635888				
1871	28029	7437	1,646669				
1872	28167	8503	1,629698				
1873	28251	7355	1,633085				
1874	28320	7725	1,636873				
1875	28345	8585	1,631108				
1876	28391	7845	1,626803				
1877	28406	8324	1,645413				
1878	28412	6627	1,632497				
1879	28430	8602	1,648705				
1880	28500	7610	1,624317				