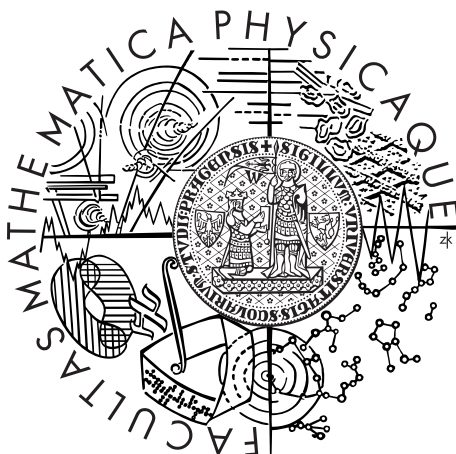


Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Bc. Ondřej Odcházal

Automatické doporučování ilustračních snímků

Ústav formální a aplikované lingvistiky

Vedoucí diplomové práce: RNDr. Pavel Pecina, Ph.D.

Studijní program: Informatika

Studijní obor: Matematická lingvistika

Praha 2014

děkuji máje, že se nebojí létat

Prohlašuji, že jsem tuto diplomovou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V dne

Podpis autora

Název práce: Automatické doporučování ilustračních snímků

Autor: Bc. Ondřej Odcházal

Katedra: Ústav formální a aplikované lingvistiky

Vedoucí diplomové práce: RNDr. Pavel Pecina, Ph.D., Ústav formální a aplikované lingvistiky

Abstrakt:

Klíčová slova: vyhledávání obrazových informací

Title: Automatic suggestion of illustrative images

Author: Bc. Ondřej Odcházal

Department: Institute of Formal and Applied Linguistics

Supervisor: RNDr. Pavel Pecina, Ph.D., Institute of Formal and Applied Linguistics

Abstract:

Keywords: information retrieval, image retrieval

Obsah

1 Úvod	3
Úvod	3
1.1 Práce s daty	3
1.2 Extrakce klíčových slov	3
1.3 Překlad do češtiny	3
1.4 Detekce jazyka	3
1.5 Webová aplikace	4
1.6 Testování	4
2 Zadání	5
3 Teorie: Jak najít vhodné obrázky	6
3.1 Popis datové sady	6
3.2 Teoretické cíle	6
3.3 Rešerše, vhodné algoritmy	6
3.3.1 TF-IDF	7
3.3.2 Extrakce bez korpusu	7
3.4 Řešení: jaké algoritmy zvoleny, získání tréninkových dat	8
3.5 Evaluace výsledků	8
4 Tvorba multijazyčného vyhledávání	9
5 Backend	10
5.1 Databáze	10
5.2 Programovací jazyk	10
5.2.1 NodeJS	11
5.2.2 Go	11
5.2.3 Ruby a Ruby on Rails	12
6 Stemmer	13
6.1 Český stemmer	13
7 Praktická část: Implementace moderní webové aplikace	14
7.1 Frontend: AJAXová aplikace na zobrazování obrázků	14
7.2 Anotační rozhraní	14
7.3 Návod k použití	14
7.4 Překlad	14
8 Možnosti tvorby moderního webového frontendu	15
8.1 Možnosti programování frontendu	15
8.2 JavaScriptové frameworky	15
8.2.1 jQuery	15
8.2.2 Google Closure	16
8.3 Stylování uživatelského rozhraní a CSS	16

9	Instalace a zprovoznění	17
9.1	Instalace	17
9.2	Práce s metadaty k obrázkům	17
9.3	Překlad metadat	17
9.3.1	Export slov a frází	18
9.4	Jazykový korpus	19
9.5	Příprava dat pro detekci jazyků	19
9.6	Import dat do databáze	20
9.7	Import metadat obrázků	20
10	Anotační rozhraní	21
10.1	Instalace rozhraní	21
10.2	Přidání uživatelů	21
10.3	Import anotačních dat	21
10.4	Export anotačních dat	22
10.5	Import obrázků a textů	23
10.6	Anotační proces	23
11	Evaluace výsledků	24
11.1	Metodika	24
11.2	Výsledky	24
11.3	Možná zlepšení	24
12	Závěr	25
	Závěr	25
	Seznam použité literatury	26
	Seznam tabulek	27
	Seznam použitých zkratk	28
	Přílohy	29

1. Úvod

Cílem diplomové práce je implementovat kompletní webovou aplikaci pro doporučování a vyhledávání ilustračních obrázků v textu. Vytvořit takovou aplikaci přináší mnoho rozličných úkolů a problémů. Tato kapitola se bude snažit tyto problémy načrtnout. Další kapitola se bude jednotlivými problémy zabývat podrobně.

1.1 Práce s daty

Zadaná data obsahují 20 milionů anotací obrázků. Základním úkolem je být schopen takové množství dat vůbec nahrát do databáze a být schopný obsloužit mnoho požadavků za minutu. Bude zmíněn současný stav vývoje databázového software pro práci s velkými daty zejména s ohledem na snadnost hledání a škálovatelnost.

1.2 Extrakce klíčových slov

Extrakce klíčových slov je důležitý podobor NLP. V práci budou rozebrány algoritmy pro extrakci klíčových slov. Bude kladen zejména důraz na rychlost a nenáročnost na zdroje. Z uživatelských testování společnosti Google vychází, že rychlost načtení stránky je jedním z klíčových vlastností pro spokojenost uživatele. Klíčová slova budou mít v aplikaci dvě využití. Pokud uživatel zadá pouze text článku, extrahovaná klíčová slova se použijí na vyhledávání relevantních obrázků. Prvních několik klíčových slov bude navíc použita jako nápověda uživateli, ten pak může tato klíčová slova využít k exaktnímu omezení množiny klíčových slov.

1.3 Překlad do češtiny

Popisky klíčových slov jsou v angličtině. Tato práce řeší překlad množiny klíčových slov do češtiny. Kromě překladu je pro hledání také nutno implementovat algoritmus na stemming. Celá aplikace je navržena tak, aby případný další jazyk mohl být přidán co nejjednodušeji.

1.4 Detekce jazyka

Jednou z drobností, kterou ocení uživatel aplikace je detekce jazyka. Uživatel bude mít možnost zadat jazyk vstupního článku exaktně, ale aplikace bude také jazyk vstupního textu sama detekovat. Budou prozkoumány možnosti detekce jazyka. Opět se nejedná o nějakou klíčovou funkci aplikace. Výstup detekce bude moci být uživatelem měněn (podobně jako funguje Google Translate¹), důraz bude tedy kladen na rychlost a jednoduchost.

¹<https://translate.google.com/>

1.5 Webová aplikace

Všechny předchozí komponenty se spojí v jedné webové aplikaci. Webový vývoj zažívá bouřlivý rozvoj. Na backendu jsou nové zejména způsoby práce s velkým množstvím dat v distribuovaném prostředí. Ve frontendové části probíhá rozvoj pomocí implementace nových technologií, známých pod hlavičkou HTML5, do moderních prohlížečů. Práce bude rozebírat všechny možnosti tvorby moderních webových aplikací.

1.6 Testování

Aplikace bude otestována na několika úrovních. Extrakce klíčových slov bude otestována pomocí korpusu článků a klíčových slov. Bude vytvořena komplexní webová aplikace pro testování doporučených obrázků. Tato aplikace bude vydělena ze samotné webové aplikace a bude používána i nezávisle.

2. Zadání

Většina zpravodajských serverů často opatřuje publikované články tzv. ilustračními snímky, jejichž úkolem je vizuálně dokreslovat obsah článku a upoutat na něj čtenářovu pozornost. Ilustrační snímky většinou pocházejí z rozsáhlých fotografických databází, jsou vybírány autory článku a s obsahem článku souvisejí jen relativně volně. Výběr ilustračních snímků probíhá nejčastěji na základě porovnávání klíčových slov specifikovaných autorem textu a popisků, kterými jsou obrázky v databázi opatřeny (typicky svými autory).

Proces výběru ilustračních snímků (dotazování ve fotografické databázi) je obtížný jednak pro samotný vyhledávací systém (hledání relevantních fotografií na základě uživatelských dotazů), jednak pro autory, kteří musí dotazy vytvářet. Konstrukce dotazů spočívá v několika krocích: uživatel nejdříve musí identifikovat ústřední téma (či témata) článku, které chce ilustrovat vhodnou fotografií, a ta potom popsat vhodnými klíčovými slovy, zvolit a zkombinovat je tak, aby vedla k nalezení vhodného obrázku. Tento proces by mohl být zjednodušen tím, že konstrukce dotazů pro vyhledávání bude prováděna automaticky pouze na základě textu článku.

Cílem diplomové práce je navržení a implementace komfortní webové aplikace pro automatické navrhování ilustračních snímků na základě textu článku, bez nutnosti explicitně konstruovat vyhledávací dotazy. Součástí práce bude i uživatelská evaluace celého systému. Pro experimenty bude použita kolekce ilustračních snímků od společnosti Profimedia.

3. Teorie: Jak najít vhodné obrázky

3.1 Popis datové sady

Datová sada poskytnutá firmou Profimedia obsahuje 20 014 394 oannotovaných obrázků ve formě CSV souboru "profi-text-cleaned.csv". CSV obsahuje sloupce "locator", "title", "description", "keywords". Sloupec locator obsahuje ID obrázku v databázi Profimedia. Sloupec description je prázdný. Sloupce title a keywords obsahují řetězce anglických slov popi sujících obrázků a oddělených mezerou.

Příklad jednoho řádku souboru profi-text-cleaned.csv:

```
1 "0000000980","hradec kings holy ghost cathedral","",  
  "outdoors nobody urban scenes architecture houses  
  towers czech czech republic europe buildings build  
  history historical churches church fronts holy  
  ghost cathedral spirit ceska republika cathedrals  
  sv hradec kralove"~M
```

Na příkladu je vidět, že data obsahují fráze jako "holy ghost cathedral", tyto fráze však nejsou strojově čitelně vyznačené. Dalším problémem je špatný překlad dat do angličtiny. Fráze "hradec kings" vznikla evidentně doslovným překladem názvu "hradec králové".

Důležitým aspektem dat je jejich nepodobnost běžnému novinovému textu. Anotované texty neobsahují většinu nejfrekventovanějších anglických slov.

3.2 Teoretické cíle

V teoretické části je hlavním cílem práce nalézt nejvhodnější metodu extrakce klíčových slov textu. Tato klíčová slova pak budou použita při vyhledávání ilustračních obrázků v databázi Profimedia. Celá práce, pokud nebude uvedeno jinak, označuje za slova stemy vstupních slov. Jako stemmer se využívá ??? stemmer.

Vstupní text tedy nejprve rozdělíme na slova. Čísla a interpunkce nás v této úloze nezajímají, jelikož se v datech nenachází. Ze slov pak získáme stemy. Vstupem algoritmu pro nalezení klíčových slov tedy bude množina vhodných stemů. Ke každému stemu si ještě uložíme jednu jeho nestemovou variantu, kterou pak můžeme zobrazit uživateli.

Nyní můžeme použít některý z algoritmů na extrakci klíčových slov uvedených v další kapitole.

3.3 Rešerše, vhodné algoritmy

Algoritmy na extrakci klíčových slov lze rozdělit do dvou kategorií. V jedné máme k dispozici korpus podobných dokumentů, druhá kategorie tento korpus ke své práci nepotřebuje.

3.3.1 TF-IDF

Technika TF-IDF je známý algoritmus na měření významnosti slov v textu. Využívá korpusu dokumentů D a dvou složek.

$$TFIDF(t, d, n, N) = TF(t, d) \times IDF(n, N) \quad (3.1)$$

Složka TF znamená *TERM FREQUENCY* a pokud t je slovo a $d \in D$ je dokument, je TF

$$TF(t, d) = \begin{cases} 1 & \text{pokud } t \in d \\ 0 & \text{jinak} \end{cases} \quad (3.2)$$

$$TF(t, d) = \sum_{slovo \in d} \begin{cases} 1 & \text{pokud } slovo = t \\ 0 & \text{jinak} \end{cases} \quad (3.3)$$

Jedná se tedy o frekvenci slova (stemu) v dokumentu.

Složka IDF , tedy *INVERSE DOCUMENT FREQUENCY* vyjadřuje, jak moc daný termín popisuje dokument. Pokud je N počet všech dokumentů v D , tedy $N = |D|$ a n je počet dokumentů, ve kterých se vyskytuje slovo t , je IDF tohoto slova

$$IDF(n, N) = \log \left(\frac{N}{n} \right) \quad (3.4)$$

$$IDF(n, N) = \log \left(\frac{N - n}{n} \right) \quad (3.5)$$

Čím je tedy slovo v korpusu častější, tím více se s logaritmem snižuje jeho informační hodnota. Slova, která jsou velmi běžná většinou klíčovými slovy nejsou.

Výsledný vzorec pak jde shrnout jako:

$$TFIDF(t, d, n, N) = \left(\sum_{slovo \in d} \begin{cases} 1 & \text{pokud } slovo = t \\ 0 & \text{jinak} \end{cases} \right) \times \log \left(\frac{N - n}{n} \right) \quad (3.6)$$

Problémem tohoto algoritmu pro je odlišný charakter korpusu a vstupních dat. Vstupní data jsou typicky novinový článek. Pokud bychom jako korpus použily anotované obrázky, získáme špatné výsledky. Běžná anglická slova, jako "the", nebo "a" se v takovém korpusu vyskytují velmi zřídka, jejich IDF tedy bude vysoká. Naopak TF v běžném novinovém textu je vysoké. Takovýto korpus nám pak označuje jako klíčová slova běžná anglická slova.

3.3.2 Extrakce bez korpusu

Dalším druhem algoritmů ke své práci korpus nepotřebují a pracují pouze se vstupním textem.

3.4 Řešení: jaké algoritmy zvoleny, získání tréninkových dat

Jako nejvhodnější řešení byl nakonec zvolen TF-IDF algoritmus. Jako kandidáti jsou odfiltrována slova, která se nenacházejí v datech Profimedia. Jako korpus k měření IDF byla použita data článků z Wikipedie.

Pro rychlé testovací účely bylo oannotováno pár článků z anglických wikinews. V každém článku jsem označil pět klíčových slov. Porovnávání algoritmů na extrakci klíčových slov pak vzalo pět nejpravděpodobnějších klíčových slov podle algoritmu a porovnávalo v kolika procentech se algoritmus trefil s anotací.

Zbývá: Zkusit otestovat další metody. Překlad do češtiny.

3.5 Evaluace výsledků

Výsledkem práce by mělo být rozhraní pro anotaci obrázků využívající algoritmus na hledání klíčových slov v textu.

Tento algoritmus se dá uživatelsky testovat několika způsoby. Uživatel vidí text a několik (cca 5) vrácených obrázků algoritmem. Uživatel vybere množinu relevantních obrázků. Další možností je mezi 5 vrácených obrázků vložit jeden náhodný. Úkolem anotátora je pak vybrat ten náhodně vybraný. Přesnost algoritmu pak jde měřit pomocí toho, kolikrát se anotátor trefí do špatného obrázku (potřeba zdrojový článek)

4. Tvorba multijazyčného vyhledávání

Metadata k obrázkům jsou v anglickém jazyce. Tato práce se snaží

5. Backend

Backend je jednou z klíčových částí aplikace. Kromě obsluhy uživatele statickými soubory (HTML, CSS, Javascript) je jeho hlavní úlohou poskytnout API pro vyhledávání.

5.1 Databáze

Úkolem databáze je uložit data a umožnit jejich prohledávání. Uživatelé této aplikace nemají možnost databázi modifikovat. Zápis do databáze provede administrátor pouze jednou před startem aplikace. Důležitým požadavkem je důraz na rychlost a snadnou škálovatelnost. V posledních několika letech vzniklo mnoho nových databází v kategorii vágně označené jako NoSQL. Tato kategorie databází se těžko popisuje, na každou popsanou vlastnost existuje NoSQL databáze, která tuto podmínku nesplňuje. Obecně ale lze říct že NoSQL databáze nepracují s prvky v tabulkovém uspořádání. Jejich výhodou oproti standardním relačním databázím může být vyšší výkon a snadná škálovatelnost.

Nevýhodou je většinou obtížnější práce s daty. Většina NoSQL databází například mapodporuje databázové transakce a vůbec celý ACID. Pro práci s databázovými daty se často používá model Map Reduce. Algoritmus Map Reduce vyvinula a publikovala společnost Google, která na něj má i patent. Google však oznámil, že algoritmus MapReduce postupně přestává používat¹.

První verze aplikace byla postavena na databázi CouchBase. Výhodou CouchBase je dobrý výkon, velmi dobrá dokumentace a také podpora knihoven pro mnoho jazyků přímo od společnosti Couchbase. Brzy se však ukázalo, že implementace textového vyhledávání v Couchbase by byla velmi náročná.

Vhodnější pro daný účel se ukázala knihovna Elasticsearch. Nejedná se v pravém smyslu o databázi. Jejím hlavním cílem je poskytnout snadné vyhledávání nad daty. Je postavená nad knihovnou Apache Lucene a poskytuje snadnou škálovatelnost. Komunikace mezi knihovnou a klientem probíhá pomocí RESTového HTTP API. Hlavním podporovaným formátem dat je JSON. Elasticsearch poskytuje programátorovi velké možnosti v nastavení prohledávání. V textovém vyhledávání může uživatel databáze použít všechny tokenizery a stemmery z knihovny Lucene. Důležitou vlastností je, že u hledaných slov může uživatel databáze určit váhu jednotlivých slov. V průběhu implementace aplikace navíc vyšla verze knihovny 1.0.

5.2 Programovací jazyk

Se vzrůstající popularitou webů vzniká stále větší množství webových frameworků a dokonce programovacích jazyků zaměřených primárně na programování pro web.

¹<http://www.datacenterknowledge.com/archives/2014/06/25/google-dumps-mapreduce-favor-new->

5.2.1 NodeJS

Jedním z nových trendů je tvorba webového backendu v Javascriptu pomocí knihovny NodeJS. Frontendoví vývojáři jsou prakticky nuceni Javascript používat. Pokud se v Javascriptu tvoří i backendová část aplikace, může snadněji dojít ke sdílení kódu i pracovních pozic. Trend psaní všech aplikací v Javascriptu glosoval již před sedmi lety Jeff Atwood ve svém pravidle:

- 1 Any application that can be written in JavaScript, will eventually be written in JavaScript.
- 2 Vše co může být napsáno v Javascriptu, bude v Javascriptu napsáno.

Nevýhody použití Javascriptu jako backendového programovacího jazyka jsou poměrně zřejmé. Javascript byl navržen pro programování webového frontendu, jeho standardní knihovna je v porovnání s ostatními jazyky velmi chudá, podpora objektového programování je celkem nepřímocará. Obsáhlý kód v Javascriptu může být poměrně nepřehledný a jazyk svádí k vytvoření takzvaného „callback hell“, který může mít strukturu jako na obrázku (source <http://strongloop.com/strongblog/node-js-callback-hell-promises-generators/>). Pokud chce navíc programátor sdílet kód z backendu i na frontendu, musí být kód kompatibilní s podporovanými prohlížeči. Funkce pro jednodušší práci s polem podporuje webový prohlížeč Internet Explorer až od verze 9 [<http://kangax.github.io/compat-table/es5/>]. V součtu nám převažily nevýhody NodeJS frameworku na výhodami a myšlenku vývoje backendu v Javascriptu jsme opustily.

```
1 doAsync1(function () {  
2     doAsync2(function () {  
3         doAsync3(function () {  
4             doAsync4(function () {  
5             })  
6         })  
7     })  
8 })
```

5.2.2 Go

Go (známý také jako golang) je programovací jazyk od společnosti Google. První stabilní verze byla zveřejněna v Květnu 2012. Go je kompilovaný, staticky typovaný jazyk s garbage collectorem. Syntaxe je inspirována jazykem C a přizpůsobena pro rychlou kompilaci. Velkou výhodou je snadná práce s vlákny pomocí „go rutin“. Programátoři v Javě, nebo C++ může překvapit poněkud netypická podpora práce s objekty.

První verze aplikace byla napsána právě v jazyce Go. V praxi se ukázaly všechny výše uvedené výhody. Hlavní nevýhodou se však ukázal nedostatek kvalitních knihoven. Přestože je jazyk velmi mladý, stal se velmi populárním, o čemž svědčí například počet repozitářů na GitHubu, nebo otázek na StackOverflow. Bude to ovšem trvat ještě nějaký čas, než se knihovny pro go a jejich vývoj stabilizují. Jeden z nyní nejpopulárnějších webových frameworků pro go — Martini — v době začátku práce na aplikaci ani neexistoval. Právě nedostatek knihoven pro go vedl k volbě jiného jazyka. Rozhodujícím pro opuštění backendu v jazyce go

byla neexistující knihovna pro práci s Elasticsearch. API je sice postaveno na protokolu HTTP, takže šlo s Elasticsearch komunikovat bez použití specializované knihovny, v praxi se to ovšem ukázalo být problematické. Vývoj aplikace si žádal testování různých nastavení a rychlé prototypování v kódu.

Z interních zdrojů máme informaci o tom, že se Elasticsearch chystá vytvořit knihovnu i pro jazyk Go. Pro naše účely se ukázalo výhodnější přepsat aplikaci do dynamického jazyka s lepší podporou pro Elasticsearch.

5.2.3 Ruby a Ruby on Rails

Ruby je dynamicky typovaný jazyk, silně inspirovaný Perlem s důslednou podporou objektového programování. Popularita Ruby vzrostla zejména kvůli webovému frameworku Ruby on Rails, který je v Ruby napsaný. RoR zpopularizovaly koncept Model-View-Controller při tvorbě webových aplikací. Backend aplikace byl nakonec kompletně přepsán jako aplikace v Ruby on Rails. Elasticsearch poskytuje pro práci s Ruby vlastní knihovnu. Další výhodou se ukázala podpora knihovny Rake, což je jakási obdoba Makefile skriptů v Ruby. Pomocí Rakefilu jdou napsat přehledné úlohy pro manipulaci s daty.

Standartní implementace jazyka Ruby byla v minulosti kritizována pro svou pomalost. V průběhu práce na této aplikaci vyšla verze Ruby 2.0, která Ruby dosti zrychluje. Tato rychlost byla pro aplikaci shledána dostatečnou.

6. Stemmer

Důležitou vlastností systémů na vyhledávání v textu je, aby dokázali najít i v jiných tvarech slova. Pokud uživatel hledá slovo „praha“, většinou očekává, že se mu zobrazí i výsledky obsahující slovo „praze“. Je tedy potřeba mít nějaký algoritmus, který k sobě slova jako „praha“ a „praze“.

První možností je použít lematizér. Úkolem lematizéru je ke každému slovu přiřadit jeho základní tvar. U podstatných jmen je to většinou první pád jednotného čísla („praha“), u sloves infinitiv.

Alternativou lematizátoru může být stemmer. Ten většinou používá jednoduché heuristiky k odstranění koncovek slov. Stemmer nemusí vrátit regulérní slovo jazyka, například pro slova „praha“ a „praze“ může stemmer vrátit slovo „prah“. Výhodou stemmeru oproti lematizéru je, že většinou používá pouze jednoduché heuristiky. Je tedy většinou rychlejší a méně náročný na zdroje než lematizér. Ve vyhledávacích aplikacích je stemmer dostačující.

Pro angličtinu je nejznámějším stemmerem pro angličtinu je Porterův stemmer popsáný[2] Martinem Porterem již v roce 1980. Kromě oficiální implementace existují porty do různých jazyků včetně Ruby. Tato aplikace využívá implementaci z Ruby Gemu „stemmify“¹ v licenci MIT.

6.1 Český stemmer

Pro jazyky jako je čeština, která má bohatší morfologii než angličtina, je tvorba stemmeru náročnější. Testovali jsme několik implementací českých stemmerů. Jako nejkvalitnější byla nakonec vybrána implementace českého stemmeru v knihovně Lucene. Tuto implementaci využívá i Elasticsearch.

V rámci této práce byl portován soubor `CzechStemmer.java` ze zdrojového kódu knihovny Lucene do jazyka Ruby. Výsledkem je Ruby Gem pod licencí MIT, který lze použít nezávisle na zbytku aplikace. Instaluje se příkazem

```
1 gem install czech-stemmer
```

Knihovna obsahuje pouze jednu třídu `CzechStemmer` s funkcí `stem`, která přijímá i vrací řetězec:

```
1 require 'czech-stemmer'
2
3 CzechStemmer.stem("praha") # => "prah"
4 CzechStemmer.stem("praze") # => "prah"
5 CzechStemmer.stem("předseda") # => "předsd"
6 CzechStemmer.stem("mladými") # => "mlad"
```

¹<https://rubygems.org/gems/stemmify>

7. Praktická část: Implementace moderní webové aplikace

7.1 Frontend: AJAXová aplikace na zobrazování obrázků

Jaké jsou dnešní možnosti vývoje frontendu. Single page aplikace. Možnosti moderních prohlížečů. JavaScriptové knihovny. Proč to nedělám v jQuery, ale používám Google Closure. Google Closure Library, Templates, Compiler.

Návrh rozhraní bez jediného tlačítka. Responzivní webdesign.

7.2 Anotační rozhraní

Jak lze v Ruby on Rails vyrobit jednoduše anotační rozhraní s uživateli a s ukládáním do databáze.

7.3 Návod k použití

Popis prvků. Screenshoty aplikace.

7.4 Preklad

Dva postupy jak použít anglická data v jiných jazycích. Buď je možné přeložit vždy zadany český dotaz do cílového jazyka. Nebo je možné přeložit všechna data u fotek. Pak je nutné použít pro každý jazyk nějaký lemmatizer, nebo stemmer. Pro cestinu jsme nakonec zvolili druhou variantu.

Jak přeložit 20 milionů popisů? Jednou možností je překlad slovo od slova. Použít pouze slovník. Překlad pomocí google je drahý. Stalo to zhruba 1300Kč. Překlad celých frází by byl lepší (automaticky objeví fráze), ale pomocí google velmi drahý. Rozhodl jsem tedy překladový nástroj Moses s modelem přiloženým ve verzi 2.1 (<http://www.statmt.org/moses/RELEASE-2.1/models/en-cs/model/>). Po několikahodinovém načítání se model načítal (i když mám SSD disk). Překlady v proloženém modelu jsou velmi pomale (jeden segment trvá přibližně 3s). Překlad není ideální a mám spoustu —UNK slov.

Google:

Překlad jiného obrázku:

Moses:

Google Translate:

8. Možnosti tvorby moderního webového frontendu

Možnosti tvorby webových aplikací se posledních několik let rapidně zvětšují. Prohlížeče implementují stále nové technologie, které rozšiřují možnosti.

8.1 Možnosti programování frontendu

Programátor webového frontendu si dnes může vybírat z několika paradigmat tvorby webové stránky. Standardem je dnes programovací jazyk Javascript. Spíše historicky bylo výhodné místo Javascriptu používat pro frontendový vývoj různé pluginy. Nejznámější je Adobe Flash, nebo Microsoft Silverlight. Programovat pro tyto pluginy mělo velké výhody. Například snadné přehrávání videa, zobrazení stránky v celoobrazovkovém režimu, nebo přístup k uživatelské webkamerě. V době, kdy byla Javascriptová API velmi chudá a rozdílně implementovaná mezi prohlížeči, nabízel Flash zejména tvůrcům webových her konzistenci mezi všemi platformami.

Velkou nevýhodou těchto pluginů byla jejich proprietálnost. Ostatní firmy se bály pustit cizí plugin do svých výrobků. Zlomový moment pro ústup Flashe ze slávy bylo uvedení telefonu iPhone, který podporu pro Flash nenabízel. Software-oví giganti Microsoft, Apple a Google začali místo proprietárních řešení tlačit otevřenou specifikaci, která se později nazvala HTML5. HTML5 je zastřešující termín pro spoustu technologií, které snaží webová API specifikovat a vytvářet nové.

8.2 JavaScriptové frameworky

Druhým hnacím motorem moderního Javascriptu jsou frameworky. Ještě před několika lety byla práce na interaktivních webech velmi náročná. Prohlížeče se zásadně lišily v implementaci práce s eventy a DOMem. Nové frameworky do jisté míry odstínily programátora od odlišných implementací Javascriptu v prohlížečích.

8.2.1 jQuery

Nejpopulárnějším frameworkem současnosti je jQuery. Ten nabízí jednoduché rozhraní a pro velkou většinu menších webových aplikací je zcela dostačující. Čím je ale aplikace větší, tím začíná být její vývoj s pomocí jQuery náročnější. Zkusme ukázat, jak by v jQuery vypadalo přidání CSS třídy `red` oknu s id `okno`:

```
1 $( "#okno" ).addClass( "red" );
```

Kód má několik problémů. Pokud neexistuje žádné okno s id `okno`, jQuery nevrátí žádnou chybu. Stačí malý překlep a chyba v kódu se hledá dost obtížně. jQuery nenabízí žádnou funkci typu `vratObjektPodleId`. Pokud by tyto funkce nabízel, velikost jeho kódu by se zvětšila. Pokud chce programátor použít knihovnu jQuery, musí si ji uživatel stránky celou stáhnout. Verze XX má XX bajtů.

8.2.2 Google Closure

Jiný přístup k vývoji frontendových aplikací přináší Google. Pro svou první webovou aplikaci Gmail vyvinul sadu nástrojů, kterou později vydal jako open source pod názvem Google Closure. Kromě Gmailu ji Google využívá v Google Vyhledávání, Google Mapách, nebo Google Dokumentech. Skládá se ze tří částí - Closure Compiler, Closure Library a Closure Templates. Closure Library je obšírlá knihovna funkcí pro práci s DOMem, Eventy, matematickými výpočty a spoustou dalších věcí, které webový programátor může využít.

Closure Compiler je inteligentní minifikátor Javascriptového kódu. Odstraňuje funkce, které nejsou volány, přejmenovává všechny názvy funkcí a proměnných na co nejkratší řetězce a v ADVANCED módu se snaží i o pokročilejší optimalizace kódu (například kód funkcí, které jsou volány pouze jednou, je vlože inline). Nejlepších výsledků dosahuje s použitím speciálních anotací, které například vynucují typ proměnné a jsou schopny udělat z Javascriptu typovaný jazyk. Closure Compiler tyto anotace vyhodnocuje a při chybném přiřazení hodnoty vrátí chybu. To umožňuje tvořit více bezpečný javascriptový kód.

Třetí částí Google Closure jsou Closure Templates, šablonovací systém pro Javascript a Javu. Pomocí Closure Templates se snadno vytváří zanořené HTML šablony. Všechny uživatelské vstupy jsou escapované což zabraňuje sniffing útokům.

Části Templates a Compiler jdou použít odděleně v jakémkoliv Javascriptovém projektu. Používat Closure Library bez Compileru nedává příliš smysl, uživatel by při návštěvě webu musel stahovat ohromné množství zbytečných dat.

Tato práce na frontendu používá všechny části knihovny Google Closure. Díky tomu si uživatel při první návštěvě webu musí stáhnout pouze jediný soubor, který má pouze XX Kb.

8.3 Stylování uživatelského rozhraní a CSS

Specifikace HTML5 rozšiřuje i možnosti vizualizace pomocí CSS stylů. Nejviditelnějšími novinkami je podpora kulatých rohů, stínování, nebo barevných přechodů. Webový programátor nyní může ke stránce načíst i vlastní font. Všechny tyto možnosti velmi rozšířily možnosti webovým grafikům.

9. Instalace a zprovoznění

Celé anotační rozhraní je webová aplikace napsaná v jazyce Ruby a frameworku Ruby on Rails. Je k dispozici pod svobodnou licencí MIT. K jejímu spuštění potřebujete ruby verze alespoň 2.0 (nižší verze nejsou otestované), javu a ke stažení zdrojového kódu git. Program jde spustit na Linuxu a Macu.

9.1 Instalace

Zdrojový kód je volně dostupný na webu GitHubu¹. Stáhnout tedy lze příkazem

```
1 git clone https://github.com/hypertornado/diplomka
```

Tento příkaz vytvoří adresář diplomka. Závislosti aplikace nainstalujete pomocí bundleru:

```
1 bundle install
```

Instalace může vyžadovat přístup administrátora. Dále je potřeba stáhnout knihovnu elasticsearch² do adresáře bin/elasticsearch. Stačí verze 1.0 a vyšší. Ve verzi 1.2.1 jsme objevili menší chybu³, která je způsobena chybou v Javě a jde obejít nastavením delšího hostname počítače.

Nyní je možné celý projekt spustit. Nejprve se spustí elasticsearch databáze pomocí příkazu `rake es:start`, poté je možné spustit samotnou aplikaci příkazem `rails server`. Po spuštění severu je uživatelské rozhraní dostupné ve webovém prohlížeči na adrese `http://localhost:3000`. Po načtení stránky se zobrazí uživatelské rozhraní, ale veškeré AJAXové dotazy skončí chybou. V databázi nejsou importována data.

9.2 Práce s metadaty k obrázkům

Metadata k obrázkům a obrázky samotné jsou poskytovány firmou Profimedia a nejsou volně dostupné. Ke zprovoznění aplikace je nutné vložit CSV soubor `keyword-cleaned-phrase-export.csv` do adresáře data.

9.3 Překlad metadat

Soubor obsahuje metadata k obrázkům v angličtině. Jedním z úkolů této práce je poskytnout doporučení obrázků i v jiných jazycích, primárně v českém jazyce. Bylo tedy nutné metadata přeložit. Pokoušeli jsme se použít volný nástroj na překlad Moses. Ve verzi 2.1⁴ nabízí volně dostupné modely pro překlad z češtiny do angličtiny. I na SSD disku trvá několik hodin, než se překladový model načte do paměti. Překlad jednoho segmentu s tímto modelem byl poměrně pomalý (překlad metadat k jednomu obrázku trval zhruba 3 sekundy) a také dosti nepřesný. Například řádek

¹<https://github.com/hypertornado/diplomka>

²<http://www.elasticsearch.org/downloads/1-0-3/>

³<https://github.com/elasticsearch/elasticsearch/issues/6611>

⁴<http://www.statmt.org/moses/RELEASE-2.1/models/en-cs/model/>

```
1 "0000000003","little baby smiling","", "child children
  baby babies infants kids childhood single faces
  body naked naked facial expressions smile smiling
  viewing watching laying fun amusing amusement
  amused amuse dallying frolics playing wanton
  open" ^ M
```

byl do češtiny přeložen takto:

```
1 "0000000003","little|UNK|UNK|UNK dítě
  smiling","", "child|UNK|UNK|UNK děti , dětské děti
  kojence děti dětství jednotného čelí orgán nahé
  naked|UNK|UNK|UNK pořídili vyjádření usmívat usmívá
  odůvodněním , která zábavné sledovat zábavné
  zábavných i pobavena tím amuse|UNK|UNK|UNK
  dallying|UNK|UNK|UNK frolics|UNK|UNK|UNK hrát
  wanton|UNK|UNK|UNK open" ^ M|UNK|UNK|UNK
```

Je vidět poměrně velké množství nepřeložených slov (koncovky |UNK) a překlad je relativně nepřesný. Je pravděpodobné, že by s lepšími daty šel natrénovat lepší překladový a jazykový model. Strojový překlad není hlavním tématem této práce, takže bylo jednodušší komerční automatický překlad Google Translate, který přeloží ukázková metadata takto:

```
1 "0000000003", "malé dítě s úsměvem", "", "dítě děti
  dítě děti kojenci děti dětství jednotlivé plochy
  těla nahá naked výrazy obličeje, úsměvu, usmívavý
  sledování sledování kterým zábava zábavné zábavní
  pobavený pobavit laškoval frolics hrát wanton
  otevřený" ^ M
```

I z této ukázky je zřejmé, že Google poskytuje kvalitnější překlad, než anglicko-český překladový model v releasu Moses. Google poskytuje překlad zdarma přes webové rozhraní. Pokud se ale do překladového formuláře nahraje nespecifikované větší množství dat, přestane překlad fungovat. Google pro překlad poskytuje placené API. Platí se XX amerických dolarů za přeložené slovo. Korpus Profimedia obsahuje zhruba (wc vystup 20119222 347129204 4811998848), takže by celý překlad stál XX dolarů. Jelikož se slova v textu opakují, lze použít překlad slovo po slovu. Ještě lepší by bylo překládat přímo klíčové fráze. Bohužel korpus Profimedia jednotlivé fráze neodděluje. Je ale možné použít učicí algoritmus a klíčové fráze detekovat.

9.3.1 Export slov a frází

Nejprve příkazem `rake data:export_profimedia_words_for_translation` vyexportujeme do souboru `data/word_list.txt` seznam všech slov použitých v metadatech k obrázkům. Tento příkaz běží několik hodin i na moderním počítači s SSD diskem. Z Profimedia dat získáme seznam 352862 slov. Tento soubor je nutné přeložit z angličtiny do dalších podporovaných jazyků, v našem případě češtiny.

9.4 Jazykový korpus

Tato práce potřebuje jazykové korpusy pro podporované jazyky ze dvou důvodů. První je potřeba pro určení relativní jazykové frekvence slov v algoritmu TF-IDF. Zadruhé je potřeba jazykový korpus pro rozpoznávání jazyků. Přirozeným jazykovým korpusem by mohla být samotná Profimedia data, ale pro oba účely jsou tato data nepoužitelná. Klíčová slova a názvy obrázků jsou odlišnými druhy textů, než je průměrný článek. Je tedy potřeba jazyková data získat jinde.

Wikipedia se jako korpus velmi hodí. Textový obsah je pod licencí Creative Commons a jeho struktura je velmi podobná běžnému publicistickému článku. Data se dají získat stažením přímo ze serverů wikipedie (pro češtinu zde), nebo pomocí sdílených torrentů. Práce wiki dumpy všech podporovaných jazyků v adresáři data pod názvem typu `wiki_dump_jazyk.xml`. Pro práci s daty z wikipedie je potřeba nejprve převést XML data do textového formátu. K tomu slouží python skript `lib/WikiExtractor.py` od Wikipedie. Pro převod anglických dat lze použít příkaz:

```
1 rake wiki:extract_words_from_wiki en
```

Stejný příkaz je potřeba spustit i pro ostatní podporované jazyky. Příkaz vytvoří v `data/wiki_en` adresářovou strukturu. Není potřeba převést všechna data, pouze tolik, abychom dostali reprezentativní korpus. Pro angličtinu stačí převést zhruba 10000 článků, pro podobné množství českých dat je potřeba převést zhruba 20000 článků (údaj o exportovaných článcích je průběžně vypisován na konzoli).

Nyní je možné vytvořit seznam slov v korpusu s frekvencemi. Ve skutečnosti nás zajímají pouze stemy slov, ne jednotlivé tvary. Příkaz

```
1 rake wiki:frequency_list_from_wiki
```

vytvoří seznam slov a jejich frekvencí s cestou `data/wiki_freq_list_count.en`. Ve skutečnosti nás nezajímají všechna slova z korpusu wikipedie, ale pouze ta, která se vyskytují mezi klíčovými slovy v Profimedia korpusu. Příkazem

```
1 rake data:create_tf_df_list
```

se z korpusu profimedia dat exportují stemy všech slov v profimedia korpusu spolu s hodnotami udávajícími frekvenci termínu v celém korpusu (TF) a frekvenci toho, v kolika popiskách obrázků se slovo objevilo (DF). Pro angličtinu jsou tato data uložena do souboru `data/tf_df_list_en.txt`. Nyní můžeme spárovat informaci o stemech z wikipedie a nahrát je do databáze. Párování se provede příkazem:

```
1 rake data:pair_profimedia_and_wiki_data
```

který vytvoří soubor `data/paired_wiki_and_profimedia.txt` se statistikami všech stemů z Profimedia korpusu.

9.5 Příprava dat pro detekci jazyků

Automatická detekce jazyka probíhá ve frontendové části. Ke svému chodu ale potřebuje data z korpusu, konkrétně seznam nejčastějších trigramů pro každý podporovaný jazyk. Příkaz

```
1 rake trigrams:extract_most_frequent_trigrams
```

vytvoří pro každý jazyk seznam padesáti nejčastějších trigramů. Seznam pro angličtinu je uložen v souboru `data/most_frequent_trigrams_en.txt`. Příkaz

```
1 rake trigrams:trigrams_to_javascript_classes
```

pracuje se seznamy nejčastějších trigramů ze kterých vytvoří javascriptovou třídu `oo.diplomka.Languages.Data` v notaci Google Closure. Ta je uložena v souboru `public/js/js/diplomka/languages/data.js`.

9.6 Import dat do databáze

Veškerá data jsou importována do databáze elasticsearch. Aplikace očekává Elasticsearch připojený na portu 9200.

9.7 Import metadat obrázků

10. Anotační rozhraní

V rámci této práce bylo implementováno anotační rozhraní pro vyhodnocování algoritmů, které přiřazují vhodné obrázky k textům. Anotační rozhraní je velmi univerzální. Anotátor má ve webové aplikaci v levém sloupci novinový text a v pravém sloupci galerii obrázků. Jeho úkolem je označit obrázky, které se k danému textu hodí a obrázky které se k textu nehodí. Má také možnost nechat obrázek neoznačený, pokud by se nemohl rozhodnout ani pro jednu variantu.

10.1 Instalace rozhraní

Celé rozhraní je aplikace napsaná v jazyce Ruby a frameworku Ruby on Rails. Aplikace je volně šiřitelná pod licencí MIT. Pro zprovoznění anotační aplikace je potřeba UNIXový systém (Linux, Mac). Zdrojový kód aplikace je uložen na severu GitHub¹ a nejlépe jde stáhnout pomocí git. Pro běh serveru je potřeba verze ruby 2.0 a vyšší. Celá aplikace se zprovozní následujícím pořadím BASH příkazů:

```
1 git clone https://github.com/hypertornado/cemi_anotace
2 cd cemi_anotace
3 bundle install #nainstaluje vsechny ruby zavislosti
4 rake db:migrate #vytvori sqlite databazi s tabulkami
5 rails server #spusti anotacni server na portu :3000
```

10.2 Přidání uživatelů

Po spuštění serveru je možné přidat anotátory v administračním rozhraní. Přístup je zaheslován HTTP autentifikací. Defaultní uživatelské jméno je cfo a heslo cfo85. Administrátorské přístupové údaje lze změnit v souboru `ROOT_APLIKACE/app/controllers`. Uživatelé mají pouze dvě datové položky, uživatelské jméno (Name) a heslo (Password). Uživatele jde přidávat, mazat a upravovat. Nepředpokládá se, že by anotovaná data byla vysoce citlivá, heslo je proto v databázi uloženo v plaintextu.

10.3 Import anotačních dat

Data pro anotaci lze nahrát pomocí příkazu

```
1 rake data:import
```

Příkaz očeká existenci souboru `ROOT_APLIKACE/public/annotation_inputs.csv`. Ten musí mít speciální formát, kdy je každý řádek rozdělen mezerami na šest sloupců s následujícími položkami:

INDEX

unikátní číslo jedné anotace

¹https://github.com/hypertornado/cemi_anotace

LABEL

interní popis pokusu

PRIORITY

priorita, celé číslo ≥ 0 . Určuje prioritu s jakou se má anotace přiřadit. Čím vyšší číslo, tím vyšší priorita.

PREFER_USER

uživatelské jméno preferovaného anotátora. Pokud není žádný anotátor preferován, použije se pomlčka

TEXT_FILE

cesta k textovému souboru s referenčním článkem

IMAGE_FILES

seznam cest k obrázkům. Cesty nemohou obsahovat mezery a jsou oddělené středníkem.

Ukázka importovaných dat:

```
1 1 basics 0 - ./text/aha/aha-00263.txt.gz
   img/1.jpg;img/2.jpg;img/3.jpg
2 2 basics 1 - ./text/aha/aha-00006.txt.gz
   img/1.jpg;img/2.jpg
3 3 basics 0 - ./text/aha/aha-00009.txt.gz  img/2.jpg
```

10.4 Export anotačních dat

Hotové anotace lze exportovat příkazem

```
1 rake data:export
```

Tento příkaz vypíše na konzolu řádky, které mají tabulátorem oddělené položky:

INDEX

ID anotace. Stejně jako u importovaných dat.

USER

Jméno anotátora, který anotaci vytvořil.

TIME

Čas uložení hotové anotace ve formátu UNIX timestamp.

SKIPPED

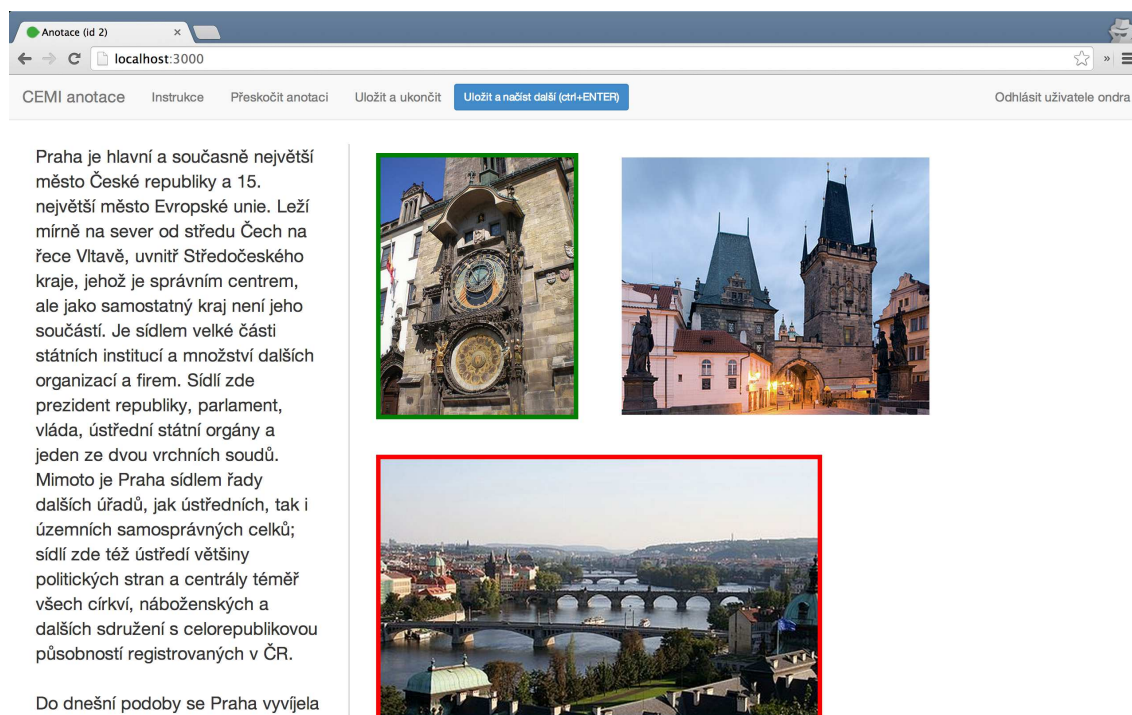
Pokud uživatel anotaci přeskočil, je hodnota True, jinak False.

APPROPRIATE

Seznam obrázků které anotátor označil jako vhodné k textu ve formátu relativních cest oddělených středníkem.

NOT_APPROPRIATE

Seznam obrázků které anotátor označil jako nevhodné k textu ve formátu relativních cest oddělených středníkem.



Obrázek 10.1: Anotační rozhraní. Vhodné obrázky jsou označené zeleným rámečkem, nevhodné červeným.

10.5 Import obrázků a textů

Anotační texty a obrázky musí být nahrány do adresáře `ROOT_APLIKACE/public` tak, aby jejich cesty odpovídali cestám v souboru `ROOT_APLIKACE/public/annotation_inputs.csv`. Pokud tedy importovaný soubor obsahuje cestu k obrázku `img/1.jpg`, musí být nahrán odpovídající soubor do `ROOT_APLIKACE/public/img/1.jpg`.

Obrázky musí být ve formátu, který podporují webové prohlížeče, tedy hlavně JPEG a PNG. Texty musí být uloženy v textových souborech s kódováním utf-8 a komprimovaný pomocí gzip².

10.6 Anotační proces

Úkolem anotátora je přiřadit vhodné a nevhodné obrázky. Po přihlášení do anotačního rozhraní vidí v levé části text a v pravé obrázky. Levým tlačítkem myši může označit obrázky, které odpovídají textu, pravým tlačítkem myši označí obrázky, které textu neodpovídají. Pokud si anotátor není jistý, nechá obrázek neoznačený. Uživatel může použít klávesovou zkratku `ctrl+ENTER` k uložení a načtení další anotace. Může také anotaci přeskočit (pak se označí jako přeskočená a nepřihradí se jinému anotátorovi), nebo uložit a ukončit.

²<http://www.gzip.org/>

11. Evaluace výsledků

11.1 Metodika

Jak budu měřit. Porovnání naivního a nejlepšího algoritmu. Nejlepší algoritmus vybrán pomocí oannotovaných klíčových slov ve Wikinews článcích. Pak anotátoři se budou snažit najít nejméně vhodný obrázek z nabízených hodnot.

Evaluace jenom pro angličtinu, nebo i pro češtinu?

11.2 Výsledky

Grafy, tabulky.

11.3 Možná zlepšení

Jak bychom mohli mít lepší data a co omezuje použitý algoritmus.

12. Závěr

Seznam použité literatury

- [1] LAMPORT, Leslie. *LaTeX: A Document Preparation System*. 2. vydání. Massachusetts: Addison Wesley, 1994. ISBN 0-201-52983-1.
- [2] PORTER, Martin F. *An algorithm for suffix stripping*. Program: electronic library and information systems, 1980, 14.3: 130-137

Seznam tabulek

Seznam použitých zkratek

Přílohy