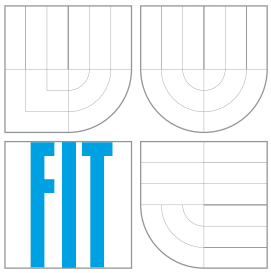
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SYSTÉM PRO VYHLEDÁVÁNÍ A VÝBĚRY RELEVANTNÍCH ČLÁNKŮ Z WIKIPEDIE PODLE TÉMATU

WIKIPEDIA PAGE CLASSIFICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ONDŘEJ SUCHÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. RNDr. PAVEL SMRŽ, Ph.D.

BRNO 2015

Abstrakt

Cílem této práce je navrhnout a implementovat systém, který umožní výběr tematicky zaměřených článků z Wikipedie za účelem úspory místa při jejím offline uložení. Řešení tohoto problému je dosaženo s využitím metod spadajících do oblasti vyhledávání informací a jejich konkrétní implementací v rámci nástroje Elasticsearch. Systém se na základě zadaných klíčových slov snaží určit, o jakou tematickou oblast se uživatel zajímá a články z této oblasti zařadit do výsledného výběru. K tomu využívá především mechanismy pro určení podobných dokumentů a zahrnutí všech článků z kategorií, které se ve výběru často opakují. Velikosti souborů generovaných výsledným systémem na základě dotazů nad Simple English Wikipedia se obvykle pohybují pod 30 MB.

Abstract

The goal of this paper is to design and implement a system for selection of Wikipedia articles relevant to a given topic in order to reduce the amount of memory taken by its offline version. The solution of this problem was achieved with use of methods from information retrieval and theirs implementation using Elasticsearch search engine. The system tries to determine the area of user's interest by given keywords and make a selection of articles from that area. This is achieved by measuring of similarity of articles and adding all articles from frequent categories in the selection. The sizes of the output files for queries over Simple English Wikipedia are usually below 30 MB.

Klíčová slova

vyhledávání informací, Wikipedie, Elasticsarch, podobnost dokumentů, vyhledávací systém

Keywords

information retrieval, Wikipedia, Elasticsarch, document similarity, search engine

Citace

Ondřej Suchý: Systém pro vyhledávání a výběry relevantních článků z Wikipedie podle tématu, bakalářská práce, Brno, FIT VUT v Brně, 2015

Systém pro vyhledávání a výběry relevantních článků z Wikipedie podle tématu

Prohlášení

Prohlašuji, že jsem svoji bakalářskou práci na téma Systém pro vyhledávání a výběry relevantních článků z Wikipedie podle tématu vypracoval samostatně pod vedením pana Doc. RNDr. Pavla Smrže, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Ondřej Suchý
18. května 2015

Obsah

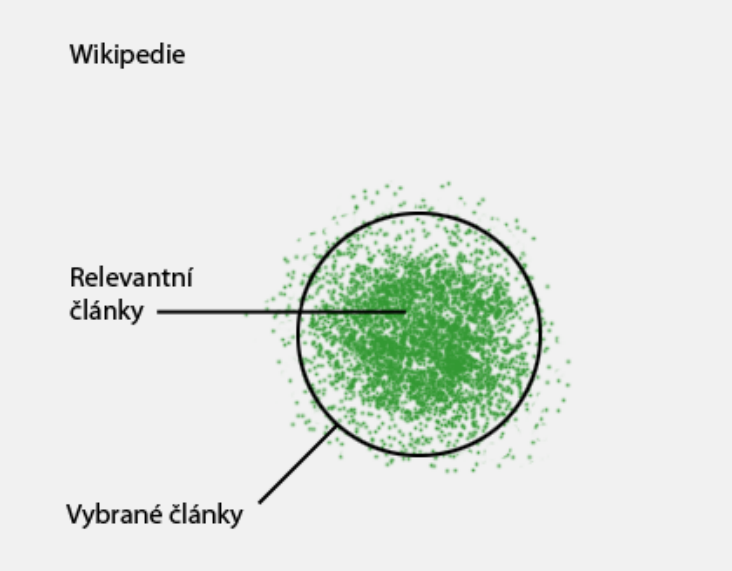
Kapitola 1

Úvod

Wikipedie je v současné době jedním z nejpopulárnějších zdrojů informací a díky mobilním technologiím je možné k ní přistupovat prakticky odkudkoliv. Nemáme-li k dispozici internetové připojení, můžeme se k požadovaným datům dostat prostřednictvím offline čteček, pro které si lze stáhnout Wikipedii ve vhodném formátu.

Ne vždy je však výhodné mít u sebe celý její obsah. Pokud například cestujeme do zahraničí a Wikipedii si bereme s sebou jen kvůli informacím o zemi, do které jedeme, část paměti bude zabrána zbytečnými články. Smyslem této práce je tedy vytvořit systém, který umožní uživateli provést výběr tematicky zaměřených článků.

Pro lepší pochopení tohoto cíle si představme celou Wikipedii jako plochu tak, jak je znázorněna na obrázku 1.1. Zelené tečky představují články týkající se zadaného tématu, v šedé oblasti se pak nacházejí všechny zbylé, nevýznamné. Úkolem systému je co nejpřesněji identifikovat relevantní oblast a vrátit její obsah uživateli. Snažíme se tedy získat maximum zeleně označených článků a při tom zahrnout jen minimum neoznačených. K tomu využijeme především metody a postupy spadající do oboru *vyhledávání informací*.



Obrázek 1.1: Princip předpokládané činnosti systému.

Jedním z dílčích cílů práce bude vybrat metody pro výběr tematicky zaměřených článků. Tyto metody budou na základě testovacích dotazů analyzovány a vhodně zakomponovány do výsledného řešení.

Dále budou vytvořeny nástroje pro zpracování článků Wikipedie, jejichž výstupem bude vhodná datová struktura pro optimalizované vyhledávání mezi nimi. Pro práci s ní bude vytvořen nástroj realizující samotný výběr. Ten bude následně propojen s nástroji na vygenerování souboru pro vybranou offline čtečku.

Kapitola 2 shrnuje teoretická východiska této práce a popisuje problematiku *vyhledávání informací*, zároveň také rozebírá strukturu a formát relevantních dat Wikipedie. V kapitole 3 navrhneme architekturu celého systému a probereme implementaci jeho jednotlivých částí. Zhodnocení systému, jeho porovnání s podobně zaměřenými řešeními a návrh možných vylepšení budou provedeny v kapitole 4.

Kapitola 2

Rozbor relevantní problematiky

2.1 Vyhledávání informací

V této sekci si probereme metody, postupy a datové struktury, které budou při práci využity. Uváděné informace, byly čerpány především ze zdrojů [8] a [12].

2.1.1 Vymezení pojmu

Význam pojmu *vyhledávání informací* (*information retrieval*) může být velmi široký. V této práci na něj budeme nahlížet tak, jak jej definuje C. D. Manning v [8]:

Definice. *Vyhledávání informací je získávání položek (obvykle dokumentů) nestrukturované povahy (obvykle text), které uspokojují potřebu získání informace z rozsáhlých kolekcí dokumentů (obvykle uložených na počítači).*

Pojmem „nestrukturovaná data“ zde rozumíme data bez jasné sémantické struktury, kterou by bylo možné snadno analyzovat počítačem. Jedná se o opak strukturovaných dat, jejichž příkladem může být relační databáze. V praxi nejsou téměř žádná data skutečně nestrukturovaná. Vedle vlastní struktury přirozeného jazyka většina textů obsahuje nadpisy, odstavce a další elementy, které jsou v dokumentech explicitně označeny.

Obor *vyhledávání informací* také zahrnuje podporu procházení a filtrování kolekcí, případně další zpracování množiny vrácených dokumentů.

Srovnání s vyhledáváním dat

Vyhledávání informací postupně vytlačuje tradiční databázově orientovaný přístup *vyhledávání dat*. Mezi základní rozdíly patří vybírání dokumentů odpovídajících zadanému dotazu. Na výstupu *vyhledávání dat* očekáváme položky přesně odpovídající zadání, zatímco v případě *vyhledávání informací* se spokojíme i s částečnou shodou a získáme tak relevantní položky.

Další důležitý rozdíl je v jazyce používaném pro dotazy. U *vyhledávání dat* budeme obvykle používat umělý jazyk s přesně vymezenou syntaxí a slovní zásobou (např. SQL). Pro *vyhledávání informací* naopak preferujeme dotazování pomocí přirozeného jazyka. Tyto dotazy tak budou vždy neúplné.

2.1.2 Požadavky na vyhledávání

Jedním ze základních požadavků kladených na *vyhledávání informací* je bezesporu rychlost celého procesu.

Základní přístup při hledání dokumentů obsahujících určitá slova je sekvenčně projít jejich text a označit ty, které obsahují daná slova. To může být velmi efektivní pro kolekce čítající jednotky milionů slov.

Vzhledem k rostoucímu objemu dat však tento přístup postupně přestává být dostačující, často navíc máme i podstatně náročnější požadavky na vyhledávání. Může nás například zajímat i vzájemná pozice slov v dokumentu nebo chceme získané dokumenty seřadit podle toho, jak dobře vyhovují zadaným slovům.

2.1.3 Index

Kolekci můžeme prohledávat efektivněji, pokud investujeme čas do jejího předzpracování a vytvoříme strukturu, která umožní optimalizované vyhledávání. Jedna z možností je pro každý jednotlivý dokument uložit, jaké *termy* obsahuje. Takto vzniklá struktura se nazývá *index*.

Termy můžeme v této práci chápat jako jednotlivá slova, obecně to ale mohou být i složitější výrazy. Jedná se o úseky, na které je text dokumentu rozdělen.

Uvažujme jako příklad kolekci, jejíž dokumenty jsou Shakespearova literární díla. Část *indexu* této kolekce můžeme vidět na obrázku 2.1.

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
Antony	1	1	0	0	0	1	
Brutus	1	1	0	1	0	0	
Caesar	1	1	0	1	1	1	
Calpurnia	0	1	0	0	0	0	
Cleopatra	1	0	0	0	0	0	
mercy	1	0	1	1	1	1	
worser	1	0	1	1	1	0	
...							

Obrázek 2.1: Podoba indexu (převzato z [8]).

Index je realizován jako *incidenční matice* (*incidence matrix*), jejíž řádky představují termy a sloupce dokumenty. Patří-li term do dokumentu, je tato skutečnost reprezentována hodnotou 1 na příslušné pozici v matici.

Pro každý z termů tedy dostaneme bitový vektor, podle nějž jsme schopni určit všechny dokumenty obsahující tento term. Při vyhledávání pak na tyto vektory aplikujeme logické operace.

Pokud hledáme například dokumenty, které obsahují slova `Brutus` a `Caesar`, ale neobsahují slovo `Calpurina`, provedeme operaci

$$110100 \wedge 110111 \wedge \neg 101111 = 100100$$

Vyhledávání na základě logických výrazů se označuje jako *booleovský vyhledávací model* (*Boolean retrieval model*). V rámci tohoto modelu není určována úroveň relevance

vrácených dokumentů [6], dotazy jsou ve formě booleovského výrazu používající operátory AND ($\wedge$), OR ($\vee$) a NOT ($\neg$).

Minimalizace paměťových nároků

Pro kolekci s 1 000 000 dokumentů a celkovým počtem kolem půl milionu unikátních termů, bude mít vzniklá tabulka přibližně $5 \cdot 10^{11}$ buněk a v paměti zabere neúnosně velký prostor. Celou strukturu je tedy nutné ještě optimalizovat.

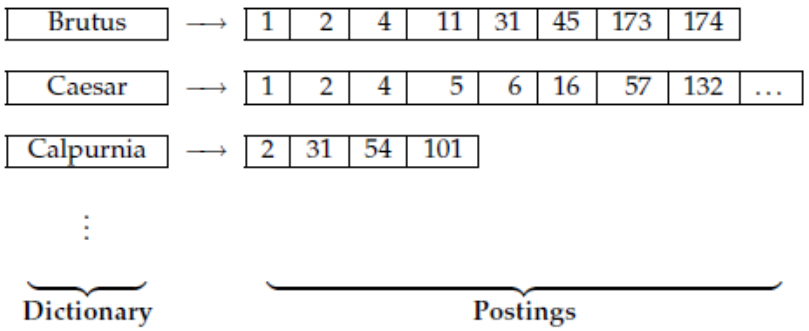
V praxi se ukazuje, že výsledná matice je extrémně řídká. Pokud bychom uvažovali průměrnou délku dokumentu kolem 1000 slov, nemůžeme nikdy v celé matici získat více než 10^9 hodnot 1. Z toho plyne, že přes 99% všech pozic zůstává prázdných. Paměť tak můžeme uspořit uložením pouze nenulových hodnot.

2.1.4 Invertovaný index

V *invertovaném indexu* (*inverted index* nebo *inverted file*) máme pro jednotlivé termy místo bitového vektoru seznam konkrétních dokumentů, které tento term obsahují. Pro seznam termů se používá označení *slovní zásoba* (*dictionary*), pro jednotlivé dokumenty *umístění* (*postings*).

Každý dokument je v *invertovaném indexu* reprezentován svým jednoznačným identifikátorem neboli *docID*. Seřazení dokumentů podle *docID* umožňuje optimalizovat vyhledávací proces.

Ve struktuře se navíc mohou vyskytnout i další informace, jako například váha termu v každém dokumentu, jeho relativní pozice v dokumentu [16] apod. Toho využijeme především při *ohodnocovaném vyhledávání*.



Obrázek 2.2: Podoba invertovaného indexu (převzato z [8]).

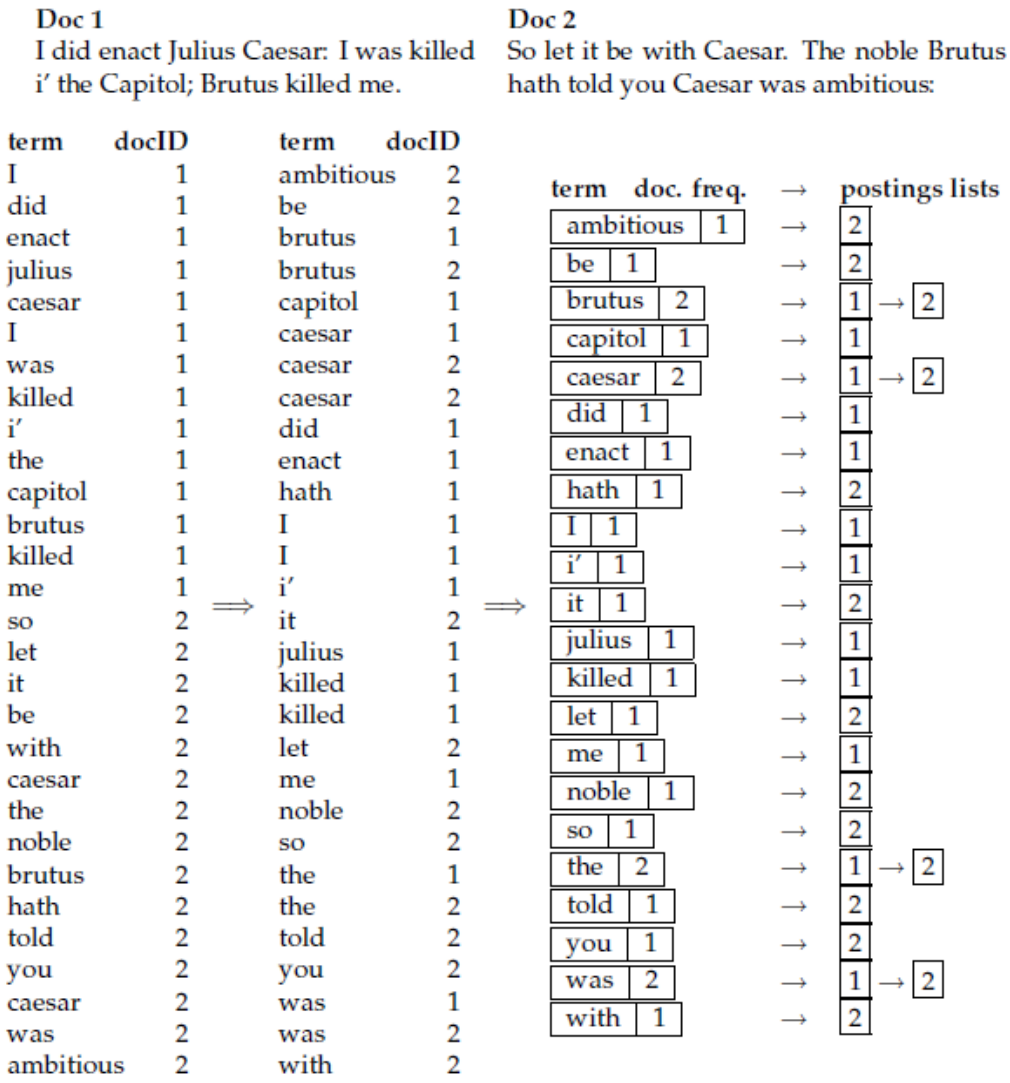
Pokud budeme chtít v *invertovaném indexu* vyhledat dokumenty na základě booleovského dotazu `Brutus` AND `Caesar`, budeme postupovat následovně:

1. Vyhledáme `Brutus` a získáme seznam jeho dokumentů
2. Vyhledáme `Caesar` a získáme seznam jeho dokumentů
3. Uděláme průnik obou získaných seznamů

Tvorba invertovaného indexu

Dokument je nejdříve nutné rozdělit na jednotky označované jako *tokeny*. Token můžeme v našem případě definovat jako posloupnost znaků v konkrétním dokumentu, které jsou seskupeny do sémantické jednotky pro zpracování.

Normalizací těchto tokenů pak získáme seznam termů. Položky tohoto seznamu jsou dvojice, ve kterých vždy figuruje term a dokument, v němž se vyskytl. Seznam je nejprve abecedně seřazen, termy v něm jsou následně seskupeny, jak ukazuje obrázek 2.3.



Obrázek 2.3: Postup při tvorbě invertovaného indexu. Termy v rámci dokumentu jsou seřazeny podle abecedy. Stejné termy jsou seskupeny nejdříve podle nesené posloupnosti znaků, poté podle dokumentu (převzato z [8]).

Normalizace tokenů

Procesem normalizace se snažíme dosáhnout toho, aby token odpovídal jinému, a to i přes drobné rozdíly ve vlastních posloupnostech znaků. Pokud například vyhledáváme výraz `NATO`, je žádoucí, aby zadání odpovídal i token `N.A.T.O.`

V praxi se obvykle používá například odstranění diakritiky, odstranění velkých písmen apod. Použití těchto metod je ale nutné rozumně korigovat, abychom neztratili původní význam tokenu.

S procesem normalizace souvisí také tzv. *stemming*, jehož úkolem je například převést slovo v množném čísle do jednotného apod.

2.1.5 Ohodnocované vyhledávání

Při vyhledávání je praktické, pokud se nemusíme starat o zvolení vhodných operátorů a získané výsledky jsou seřazeny od nejrelevantnějších po méně relevantní. Tento styl dotazování, který je velmi populární na webu, nahlíží na dotazy jednoduše jako na množinu slov. Vhodný ohodnocovací mechanismus pak vypočítá skóre jako součet vah jednotlivých termů z dotazu pro daný dokument.

Četnost termu a inverzní dokumentová četnost

Každý term má v dokumentu svoji váhu, určenou jako *četnost termu* (*term frequency*) se zápisem $\mathrm{tf}_{t,d}$. Tato hodnota odpovídá počtu výskytů termu t v dokumentu d. Vzájemná pozice termů není brána v úvahu.

Dále je nutné snížit vliv často se vyskytujících termů na určení relevantnosti dokumentu. K tomu využijeme *dokumentovou četnost* (*document frequency*) df_t, vyjadřující počet dokumentů v kolekci obsahující term t.

Slovo	cf	df
try	10422	8760
insurance	10440	3997

Tabulka 2.1: Počet výskytů v kolekci (cf) a *dokumentová četnost* (df) pro slova `try` a `insurance` (převzato z [8]).

Důvod, proč preferujeme *dokumentovou četnost* před celkovým počtem výskytů termu v kolekci, ukazuje tabulka 2.1, z níž plyne, že počet výskytů v kolekci a dokumentová četnost jsou hodnoty s odlišným chováním. Zatímco výskytů v kolekci je pro uvedené termy téměř stejně, jejich *dokumentová četnost* se výrazně liší. Intuitivně bychom chtěli, aby malý počet dokumentů obsahujících `insurance` dostal vyšší hodnocení pro dotaz obsahující `insurance`, než dostanou dokumenty obsahující `try` pro dotaz na `try`.

Na základě dokumentové četnosti df a celkového počtu dokumentů N zavedeme *inverzní dokumentovou četnost* (*inverse document frequency*). Hodnota idf bude tedy vysoká pro málo se vyskytující termy a naopak nízká pro časté termy.

$$\mathrm{idf}_t = \log \frac{N}{\mathrm{df}_t}$$

Hodnocení na základě tf-idf

Abychom získali požadovanou váhu termu v dokumentu, zkombinujeme četnost termu a inverzní dokumentovou četnost následujícím způsobem:

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \cdot \text{idf}_t$$

Termu t tedy tf-idf přiřazuje váhu, která je

1. nejvyšší, pokud se t vyskytne mnohokrát v rámci malého počtu dokumentů,
2. nižší, pokud se t vyskytne méně často nebo jsou jeho výskyty rozloženy do více dokumentů,
3. nejnižší, pokud se t vyskytne ve všech dokumentech.

Dokument tak můžeme chápat jako vektor s jednou složkou pro každý term a s váhou získanou jako tf-idf pro každou složku. Skóre dokumentu d se pak může spočítat například jako součet všech vah pro jednotlivé termy z dotazu q.

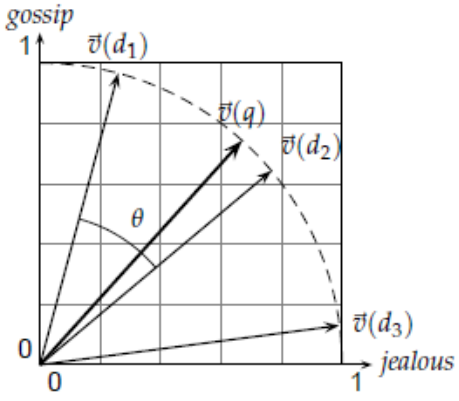
$$\text{Score}(q, d) = \sum_{t \in q} \text{tf-idf}_{t,d}$$

Podobnost dokumentů

Kdybychom chtěli měřit podobnost dokumentů na základě rozdílu jejich vektorů, narazíme na problém s jejich délkami. Přestože rozložení termů ve zkoumaných dokumentech je podobné, jejich rozdíl může být značný jen proto, že jeden dokument je delší než druhý.

Abychom tomu předešli, standardně se podobnost dvou dokumentů d_1 a d_2 vyčísluje jako úhel mezi jejich vektory neboli *kosinová podobnost* jejich vektorových reprezentací [6] $\vec{V}(d_1)$ a $\vec{V}(d_2)$ podle vzorce

$$\text{sim}(d_1, d_2) = \cos\theta = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{|\vec{V}(d_1)| \cdot |\vec{V}(d_2)|}$$



Obrázek 2.4: Kosinová podobnost vektorů jako úhel θ (převzato z [8]).

V praxi se jmenovatel tohoto zlomku neuvažuje a jako výsledek vybíráme dokumenty, které mají nejvyšší skalární součin se zadaným dokumentem. Tento prvek se ve vyhledávačích vyskytuje pod označením *more like this*.

2.1.6 Textová klasifikace a její význam

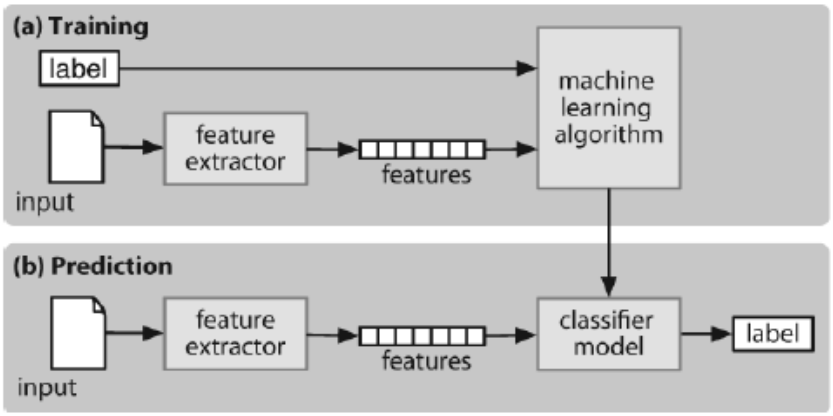
Textová klasifikace nebo také *textová kategorizace* je v současné době disciplína stojící na rozhraní oborů strojového učení a *vyhledávání informací*. Následující informace byly čerpány ze zdrojů [3], [10] a [15].

Textovou klasifikaci můžeme použít například pro

- rozhodnutí, zda email je nebo není spam,
- zařazení novinového článku do jednoho z nabízených témat jako „sport“, „politika“ nebo „technologie“,
- rozlišení, zda slovo `měsíc` je použito ve smyslu měsíc v roce nebo jako označení vesmírného tělesa.

Pokud máme soubor dokumentů $\mathbb{D}$ a pevnou množinu tříd $\mathbb{C} = \{c_1, c_2, \ldots, c_j\}$, klasifikace se snaží každé dvojici $\langle d_i, c_j \rangle$ přiřadit booleovskou hodnotu. Tato hodnota bude `True` (T) pokud dokument d_i patří do třídy c_j, jinak `False` (F). Jinými slovy se snažíme najít funkci $\Phi : \mathbb{D} \times \mathbb{C} \rightarrow \{T, F\}$.

Hledanou funkci Φ se klasifikátor naučí na základě trénovací množiny dokumentů. Tato metoda učení se nazývá *učení s učitelem* (*supervised learning*). Většina učících algoritmů očekává reprezentaci atribut-hodnota, dokumenty se proto převedou do vektorové reprezentace.



Obrázek 2.5: Postup při stojovém učení s učitelem. Nejprve se klasifikátoru dá množina oklasifikovaných dokumentů a pomocí vhodného algoritmu se podle nich získá potřebné mapování pro klasifikaci. Poté se na vstup mohou dávat dokumenty, které se klasifikátor pokusí zařadit do jedné ze tříd (převzato z [3])

Klasifikace a Wikipedie

Na první pohled by se mohla textová klasifikace jevit jako smysluplná metoda pro analýzu článků na Wikipedii. Články budeme dávat klasifikátoru a on nám je zařadí do příslušných oborů, kterých se týkají. Je však důležité zamyslet se nad základními fakty.

V roce 2004 byl na Wikipedii zaveden systém kategorií [7], který umožňuje její manuální klasifikaci. Každý článek tedy spadá do jedné nebo více kategorií, kam jej zařadil jeho autor. Používat textový klasifikátor, abychom si udělali představu o obsahu daného článku,

tak nemá velký význam. Jeho výstup by pravděpodobně řekl de facto to samé, co nám říkají kategorie, ruční klasifikace by navíc pravděpodobně zůstala přesnější a spolehlivější. Struktura kategorií bude podrobněji probrána v části 2.2.2.

Další problém, na který bychom narazili je absence trénovací množiny. Jak bylo zmiňováno výše, než klasifikátor uvedeme do provozu, musíme jej nejprve naučit, jak klasifikovat, na vhodné množině dat. Jakou množinu ale zvolit? Použití samotné Wikipedie a jejích kategorií nedává smysl vzhledem k tomu, že na vstupu klasifikace se budou objevovat stejné dokumenty, jako by byly v trénovací množině. I pokud bychom našli vhodnou trénovací množinu nebo zvolili učení klasifikátoru jinou metodou než na základě trénovací množiny, nejspíš bychom opět nezískali nic navíc oproti existujícím kategoriím.

Wikipedie tak sice může představovat užitečnou trénovací množinu pro klasifikátor, určující například témata odborných článků, není ale přínosné aplikovat klasifikaci na ni samotnou za stejným účelem. Vše potřebné se můžeme dozvědět z existujících kategorií.

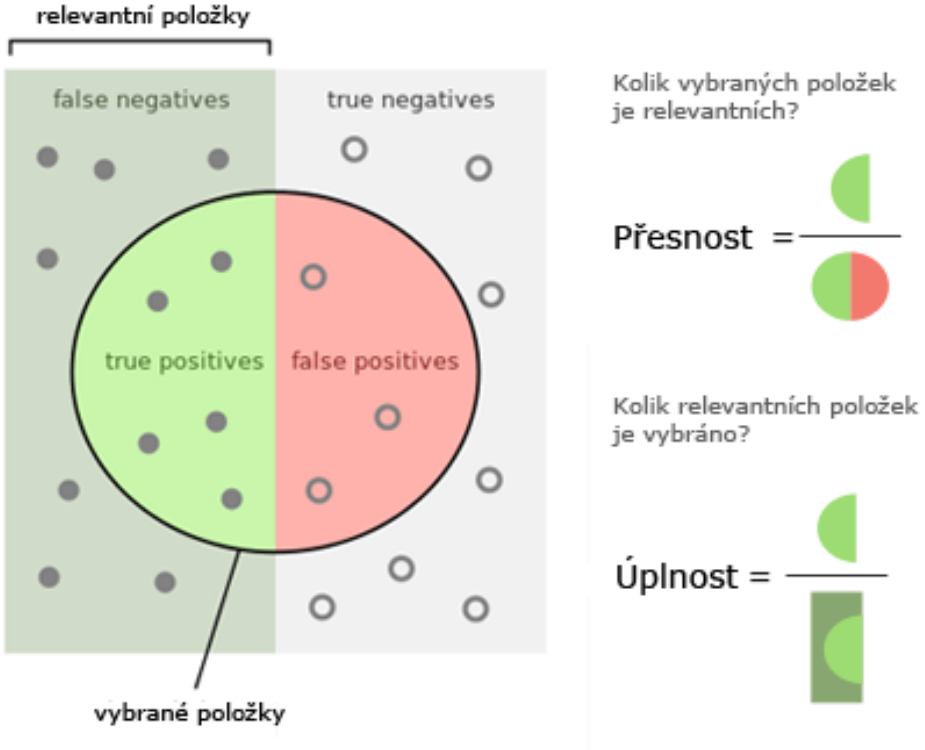
2.1.7 Vyhodnocení vyhledávácího systému

Při vyhodnocování systému pro vyhledávání informací se snažíme určit, do jaké míry je systém schopen uspokojit uživatelovu potřebu po informaci [17]. K tomu jsou využívány testovací kolekce dat, nad kterými jsou vykonávány experimenty za účelem porovnání různých přístupů. Kolekce obsahují množinu témat (*topics*), seznamy všech dokumentů, které mají být vráceny pro jednotlivá témata a samotnou kolekci dokumentů [4].

Důležité metriky pro hodnocení systému jsou *přesnost* (*precision*) a *úplnost* (*recall*) charakterizující jeho efektivitu. Jejich význam je následující:

- *přesnost* udává, jaká část ze získaných dokumentů je skutečně relevantní
- *úplnost* určuje, kolik ze všech relevantních dokumentů v kolekci bylo vráceno

Od efektivního systému čekáme, že vrátí relevantní dokumenty a vynechá nerelevantní [13]. Určení *přesnosti* a *úplnosti* je ukázáno na obrázku 2.6.



Obrázek 2.6: Demonstrace pojmů *přesnost* a *úplnost* (převzato z http://en.wikipedia.org/wiki/Precision_and_recall).

Množina dokumentů, které jsme chtěli získat a skutečně jsme je získali, se označuje jako `true positives` (`tp`). Nerelevantní dokumenty vrácené navíc se značí `false positives` (`fp`). Dále nám ještě zbývají množiny nevybraných relevantních (`false negatives`, `fn`) a nerelevantních dokumentů (`true negatives`, `tn`).

V literatuře se toto rozdělení kolekce na množiny obvykle zapisuje pomocí tabulky (viz 2.7). Sloupce **+R** a **-R** značí po řadě relevantní a nerelevantní dokumenty, řádky **+P** a **-P** pak vybrané a nevybrané dokumenty.

	+R	−R	
+P	tp	fp	pp
−P	fn	tn	pn
	rp	rn	1

	+R	−R	
+P	A	B	A+B
−P	C	D	C+D
	A+C	B+D	N

Obrázek 2.7: Tabulky pro zápis hodnot `true`/`false positives`/`negatives` s možnými notacemi (převzato z [11]).

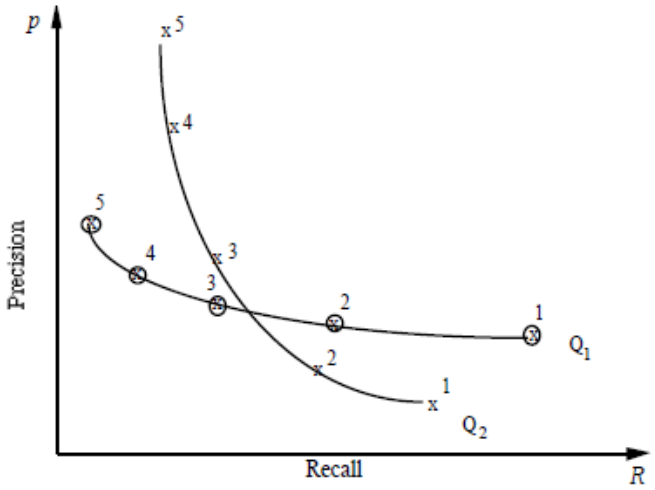
Pro značení buněk lze použít písmena abecedy nebo zkratky pro pojmy `true` a `false`, `real` a `predicted positives` a `negatives`. Velká písmena se často používají, pokud uvádíme skutečné počty, malá v situacích, kdy hodnoty udáváme jako poměry [11].

Na základě uvedených tabulek můžeme definovat vztahy pro *přesnost* a *úplnost*:

$$\text{přesnost} = \frac{tp}{tp + fp} = \frac{tp}{pp} = \frac{A}{A + B}$$

$$\text{úplnost} = \frac{tp}{tp + fn} = \frac{tp}{rp} = \frac{A}{A + C}$$

Tyto dvě metriky jdou proti sobě (viz obrázek 2.8) a v reálných situacích je obvykle jedna hodnota důležitější než druhá. V praxi tedy hledáme kompromis, abychom dosáhli dostatečné *úplnosti* bez příliš vysoké ztráty *přesnosti* [13].



Obrázek 2.8: Křivka *přesnosti* a *úplnosti* pro dva různé dotazy Q_1 a Q_2 (převzato z [12]).

Správnost systému

Jestliže budeme na systém pohlížet jako na klasifikátor, který zařazuje dokumenty do jedné ze dvou tříd (relevantní a nerelevantní), nabízí se nám jako alternativa pro jeho hodnocení *správnost* (*accuracy*). Ta se určí jako počet správně klasifikovaných dokumentů.

$$\text{správnost} = \frac{tp + tn}{tp + fp + fn + tn}$$

Pro *vyhledávání informací* však *správnost* není vhodná metrika. Běžně i více než 99.9 % dokumentů patří mezi nerelevantní a pro její maximalizaci tak stačí označit všechny dokumenty jako nerelevantní pro všechny možné dotazy.

Tento přístup je ale nepřípustný. Uživatel vždy očekává od systému jako výsledek alespoň nějaké dokumenty a pravděpodobně bude ochoten tolerovat i jistý počet nerelevantních dokumentů, pokud získá alespoň nějaké přínosné informace.

2.2 Datové a programové zdroje

V této části kapitoly je popsána serializace, získání dat Wikipedie a také smysl a struktura jejích kategorií. Informace o serializaci byly čerpány ze zdrojů [1] a [9], o Wikipedii pak z [2], [7], [14] a [18].

2.2.1 Serializace datových objektů

Pojmem *serializace* rozumíme převedení dat do posloupnosti bytů, které mohou být snadno přenášeny, uloženy a později rekonstruovány. Rekonstrukcí této posloupnosti jsme schopni vytvořit sémanticky totožnou kopii původního objektu.

Serializace je často používána pro rozšíření pole působnosti objektu za hranice jeho momentálního umístění a aktuálního času. Jinými slovy ji můžeme využít například pro vytvoření zálohy databáze v podobě dumpu a chránit tak systém před neočekávanými negativními událostmi.

Mezi nejpopulárnější modely momentálně patří XML a JSON. XML platí víceméně jako standard pro výměnu libovolných dat, JSON se pak používá především pro serializaci objektů v programovém prostředí.

DTD

XML dokumentům můžeme pomocí DTD (Document Type Definition) přiřadit seznam pravidel a omezení. DTD nám tedy umožňuje definovat, jaké elementy jsou v našem dokumentu přípustné, jaké mohou mít atributy a jaký může být jejich obsah.

Obecně se element se jménem `el-name` může definovat například zápisem

```
<!ELEMENT el-name (child1, (child2|child3)?, child4*) >
```

Uvedený zápis říká, že `el-name` bude vždy obsahovat potomka `child1`, volitelně jednoho z dvojice `child2`, `child3` a libovolný počet potomků `child4`.

Pokud může element obsahovat text, bude jako přípustný potomek uvedeno `#PCDATA` („parsed character data“), prázdné elementy pak mají na místě potomků hodnotu `EMPTY`.

Budeme-li chtít elementu `el-name` definovat atribut `attr-name`, použijeme zápis

```
<!ATTLIST el-name attr-name attr-type attr-value >
```

Z možných hodnot `attr-type` si uvedeme pouze `CDATA` značící textová data. Hodnoty `attr-value` jsou uvedeny v tabulce 2.2 včetně jejich významu.

Hodnota	Význam
hodnota	Implicitní hodnota atributu
#REQUIRED	Atribut je povinný
#IMPLIED	Atribut je volitelný
#FIXED *hodnota*	Hodnota atributu je pevně daná

Tabulka 2.2: Možné hodnoty pole `attr-value` a jejich význam (převzato z http://www.w3schools.com/dtd/dtd_attributes.asp).

JSON

JSON („JavaScript Object Notation“) je odlehčený jazykově nezávislý formát navržený pro serializaci Javascriptových objektů. Základní konstrukci představuje dvojice klíč-hodnota ve formátu `,,klíč‘‘: hodnota`.

Kromě základních typů (řetězec, číslo atd.) můžeme definovat i složitější typy, a to buď *objekt* nebo *pole* s následujícími vlastnostmi:

- *Objekt* je neuspořádaná množina dvojic jméno-hodnota oddělených čárkami a uzavřených ve složených závorkách, hodnoty mohou být obecně různého datového typu.
- *Pole* je uspořádaná kolekce hodnot stejného typu uzavřená v hranatých závorkách.

Objekty a pole lze libovolně zanořovat.

```
,,director‘‘: {
    ,,last_name‘‘: ,,Fincher‘‘,
    ,,first_name‘‘: ,,David‘‘,
    ,,birth_date‘‘: 1962
}

,,actors‘‘: [,,Eisenberg‘‘, ,,Mara‘‘, ,,Garfield‘‘, ,,Timberlake‘‘]
```

Obrázek 2.9: Příklad objektu `director` a pole `actors` ve formátu JSON.

Každý JSON dokument může být převeden do XML, opačně to však říci nelze. JSON na rozdíl od XML nepodporuje jmenné prostory ani odkazovaní na jiné objekty. Neexistuje ani prostředek pro určení typových omezení pro JSON, jako je například DTD.

2.2.2 Wikipedie

Podle oficiálních statistik anglická Wikipedie v současné době (duben 2015) obsahuje více než 4 800 000 článků, díky čemuž je pravděpodobně největší volně dostupný zdroj informací a vědomostí. Počet článků se navíc stále zvyšuje.

Získávání dat

Nadace Wikimedia uvolňuje pravidelně kompletní zálohu Wikipedie v podobě veřejně dostupných dumpů. Jména všech souborů v dumpu začínají označením jazykové verze Wikipedie a datem jejich vydání. Kompletní texty všech článků jsou uloženy ve formě zkomprimovaného XML. Jedná se o soubor s koncovkou `pages-articles.xml.bz2`. Pro anglickou Wikipedii může tedy celý název souboru s obsahem článků být například `enwiki-20150304-pages-articles.xml.bz2`. Tato část dumpu obsahuje všechna data, která jsou pro nás důležitá, proto si nyní rozebereme jejich strukturu.

Kořenovým elementem je element `<mediawiki>` s atributy definujícími jmenné prostory XML. Celý soubor se dá dále rozdělit na dvě části. První z nich je vymezena elementem `<siteinfo>` a obsahuje informace o verzi Wikipedie, adresu její hlavní stránky apod. Dále jsou zde uvedeny jmenné prostory Wikipedie v elementech `<namespace>` a jejich identifikátory `key`, které rozlišují význam jednotlivých stránek.

Druhá část souboru, která následuje za `<siteinfo>`, je tvořena libovolným počtem elementů `<page>`. Pro každou stránku Wikipedie je zde jeden tento element. Stránka se

může týkat například kategorie, uživatele, šablony, ale především se může jednat o konkrétní článek. Pro lepší představu o struktuře `<page>` si uvedeme její DTD.

```
<!ELEMENT page (title,ns,id,restrictions?,revision,redirect?)>
  <!ELEMENT title (#PCDATA)>         <!-- Nadpis s případnou předponou -->
  <!ELEMENT ns (#PCDATA)>            <!-- označení jmenného prostoru -->
  <!ELEMENT id (#PCDATA)>            <!-- číselný identifikátor -->
  <!ELEMENT restrictions (#PCDATA)>  <!-- volitelné restrictions -->
  <!ELEMENT redirect EMPTY>          <!-- přesměrování na jiný článek -->
  <!ATTLIST redirect title CDATA  #REQUIRED >
<!ELEMENT revision
  (id?,parentid?,timestamp,contributor,
   minor?,comment?,text,sha1,model,format)
>
  <!ELEMENT timestamp (#PCDATA)>     <!-- podle ISO8601 -->
  <!ELEMENT parentid (#PCDATA)>
  <!ELEMENT sha1 (#PCDATA)>
  <!ELEMENT model (#PCDATA)>
  <!ELEMENT format (#PCDATA)>
  <!ELEMENT minor EMPTY>             <!-- minor příznak -->
  <!ELEMENT comment (#PCDATA)>
  <!ELEMENT text (#PCDATA)>          <!-- Wikisyntax -->
  <!ATTLIST text xml:space CDATA  #FIXED ,,preserve‘‘>
<!ELEMENT contributor ((username,id) | ip)>
  <!ELEMENT username (#PCDATA)>
  <!ELEMENT ip (#PCDATA)>
```

Každá stránka má svůj nadpis v elementu `<title>`, který může mít jednu z předpon uvedených v `<namespaces>` v části `<siteinfo>`. Ta reprezentuje příslušnost stránky do jednoho ze jmenných prostorů Wikipedie. Stránky nesoucí obsah článku patří do hlavního jmenného prostoru, jehož předpona je prázdná.

Například stránka pro kategorii `Art` tak bude obsahovat `<title>Category:Art</title>`, zatímco stránka s článkem o umění bude mít nadpis jen `<title>Art</title>`. V elementu `<ns>` je pak uveden identifikátor jmenného prostoru, do nějž předpona spadá.

Tělo článku je uvedeno v elementu `<text>`, který je potomkem `<revision>`. Speciálně označené jsou v tomto textu prvky jako infoboxy, citace, odkazy a další. Pro nás budou důležité především odkazy na jiné stránky `<page>`, které jsou vždy uzavřeny ve zdvojených hranatých závorkách. Díky nim jsou všechny záznamy na Wikipedii provázané a z libovolného záznamu je tak možné dostat se na libovolný jiný.

Odkaz vždy vede na stránku, jejíž nadpis je uveden v závorkách. Za nadpisem může být ještě uveden text, který se zobrazí při prohlížení stránky v prohlížeči, oddělený znakem '`|`'. Například `[[flight|fly]]` značí odkaz na článek `Flight`, který se v textu zobrazí jako slovo `fly`.

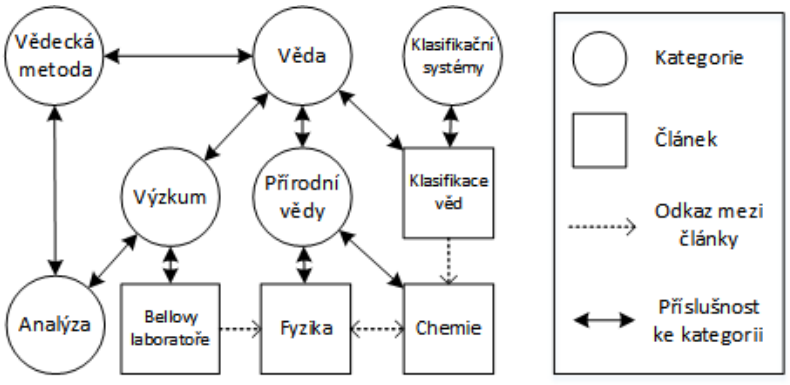
Zbývající elementy v `<page>` pro nás nejsou podstatné a nebudeme se jimi podrobněji zabývat. Příklad jednoho článku Wikipdie v jeho XML reprezentaci je ukázán na obrázku 2.10.

```
<page>
  <title>Nadpis článku</title>
  <ns>0</ns>
  <id>6</id>
  <revision>
    <id>4896310</id>
    <parentid>4831012</parentid>
    <timestamp>2014-09-25T12:18:19Z</timestamp>
    <contributor>
      <username>RandomUser</username>
      <id>12345</id>
    </contributor>
    <minor />
    <comment>Text komentáře</comment>
    <text xml:space=,,preserve‘‘>Tělo článku s [[odkazy]] na jiné články.</text>
    <sha1>kwsvsphp51dyy3ipoiy4golr6epsq3j</sha1>
    <model>wikitext</model>
    <format>text/x-wiki</format>
  </revision>
  <restrictions>edit=sysop:move=sysop</restrictions>
</page>
```

Obrázek 2.10: Možná podoba jednoho článku wikipedie v jeho XML reprezentaci

Struktura kategorií

Smyslem kategorií je sdružovat články Wikipedie s podobným obsahem. Kategorie mohou být vytvářeny i měněny uživateli podobně jako články. Příslušnost článku do kategorie je v dumpu zachycena v poli `<text>` stejným způsobem jako odkaz na jiný článek, tedy název kategorie ve zdvojených hranatých závorkách. Pokud máme například článek `Drawing`, který patří do kategorie `Art`, bude v závěru jeho textu uvedeno `[[Category:Art]]`.



Obrázek 2.11: Struktura kategorií Wikipedie a odkazy mezi články.

Každá kategorie může mít libovolný počet podkategorií a článků, podobně pak každý článek může patřit do libovolného počtu kategorií. Díky podkategoriím tvoří celý systém strukturu podobnou stromu, kde jednotlivé kategorie představují uzly a články listy. Každá

položka této struktury ale může mít více předků a nejsou vyloučeny ani cykly a odpojené kategorie, přestože jsou vzácné.

Proč vybírat články?

Jak již bylo řečeno, anglická Wikipedie obsahuje téměř 5 milionů článků, což znamená, že její offline podoba bude zabírat nemalý prostor v paměti. Bližší představu o paměťových nárocích na její uložení si můžeme udělat na základě tabulky 2.3 s velikostmi vybraných jazykových verzí Wikipedie. Pro srovnání jsou zde uvedeny velikosti dumpu s texty článků a převedených souborů pro offline čtečky Aard Dictionary a Kiwix.

Verze Wikipedie	dump		Aard Dictionary (jen text)	Kiwix (včetně obrázků)
	zkomprimovaný	rozbalený		
anglická	11 GB	48 GB	11.5 GB	40 GB
česká	483 MB	1.9 GB	872 MB	3.1 GB
simplewiki	103 MB	470 MB	152 MB	1.2 GB

Tabulka 2.3: Srovnání velikostí dumpů Wikipedie, archívů bez obrázků a archívů s obrázky. Vzhledem k neustálému vývoji Wikipedie jsou uvedené velikosti pouze orientační, platné v době psaní této práce.

Přestože pro běžné počítače není příliš velký problém poradit si s takto objemnými soubory, práce s nimi na mobilních zařízeních může být přinejmenším nepohodlná, ne-li neřešitelná.

Kapitola 3

Návrh a implementace

Jak již bylo naznačeno v úvodu této práce, úkolem celého systému, je přijmout od uživatele požadavek specifikující určitý tematický okruh, o který se zajímá, na jeho základě vybrat z Wikipedie články související s tímto tématem, vygenerovat z nich soubor ve formátu pro některou z dostupných offline čteček Wikipedie a ten pak vrátit uživateli. K tomu je zapotřebí mít vytvořený index Wikipedie, aby bylo možné články efektivně vyhledávat, a také vhodný nástroj pro generování výsledného souboru.

3.1 Použité nástroje

3.1.1 Výběr offline čtečky

Existuje poměrně velké množství různých čteček, které umožňují offline prohlížení Wikipedie. Jako zástupce si uvedeme Kiwix[1] a Aard Dictionary[2]. Následující informace byly čerpány z jejich oficiálních stránek.

Kiwix

Kiwix je multiplatformní offline čtečka primárně určena pro prohlížení Wikipedie, obecně je ale vhodná pro prohlížení libovolného HTML obsahu. Obsah určený ke čtení je uložen ve formátu ZIM, který obsahuje zkomprimovaná data a metadata. Tento formát podporuje také přidání obrázků z dané stránky. V případě Wikipedie však obrázky ve výsledném souboru znamenají výrazný nárůst jeho velikosti, jak ukazuje tabulka 2.3. Z pohledu této práce je hlavní nevýhodou Kiwixu, že není možné generovat formát `.zim` přímo z XML dumpů, což komplikuje celý proces.

Aard Dictionary

Aard Dictionary umožňuje rychlé vyhledávání slov i ve velkých kolekcích dat, což z něj dělá kvalitní offline čtečku Wikipedie. Vyhledávat je možné i ve více slovnících najednou bez nutnosti přepínání mezi nimi. Stejně jako Kiwix je i Aard Dictionary multiplatformní a je vhodný pro mobilní zařízení. K ukládání dat používá vlastní formát `.aar`, pro jehož vytvoření je volně k dispozici kolekce nástrojů Aard Tools. Díky nim je možné snadno

[1] http://www.kiwix.org
[2] http://aarddict.org

vygenerovat slovník přímo z dumpu Wikipedie. Tento formát na rozdíl od `.zim` nepodporuje obrázky.

K dispozici je momentálně (duben 2015) nově také beta verze čtečky s názvem Aard 2, která využívá formát `.slob`. Slovníky ve formátu `.aar` je možné převést na `.slob` pomocí volně dostupného nástroje `aar2slob`.

Jako cílová čtečka pro tento projekt byl zvolen Aard Dictionary, a to především kvůli zmiňované dostupnosti konverzních nástrojů a jednoduché práci s nimi.

3.1.2 Elasticsearch

Jedním z nástrojů pro tvorbu indexu, prohledávání a analýzu dat je například Elasticsearch. Jedná se open-source fulltextový vyhledávač založený na knihovně Apache Lucene[3]. Celý systém funguje jako server napsaný v jazyce Java. Komunikace s ním je možná přes RESTful API, díky čemuž s ním lze pracovat pomocí webového klienta libovolného programovacího jazyka nebo přímo přes příkazovou řádku.

Pro rychlé fulltextové vyhledávání používá Elasticsearch strukturu *invertovaný index*, který byl podrobně probrán v části 2.1.4, včetně kroků nutných k jeho vytvoření. Systém umožňuje ukládat komplexní entity v podobě strukturovaných JSON dokumentů. Všechna jejich pole jsou indexována a lze v nich vyhledávat, všechny indexy pak mohou být prohledávány najednou. Systém je také schopen automaticky rozpoznat strukturu dat zadaného dokumentu.

3.2 Architektura systému

Primárním cílem systému jsou čtečky nainstalované na mobilních zařízeních, není však možné, aby na nich běžel celý systém. Především narazíme na problém, že potřebné nástroje nejsou dostupné pro mobilní operační systémy a ani jejich výpočetní výkon nejspíš nebude dostačující. Dále bude nutné manipulovat s velkými objemy dat, jak plyne z údajů uvedených v části 2.2.2, což by mohlo znamenat komplikace.

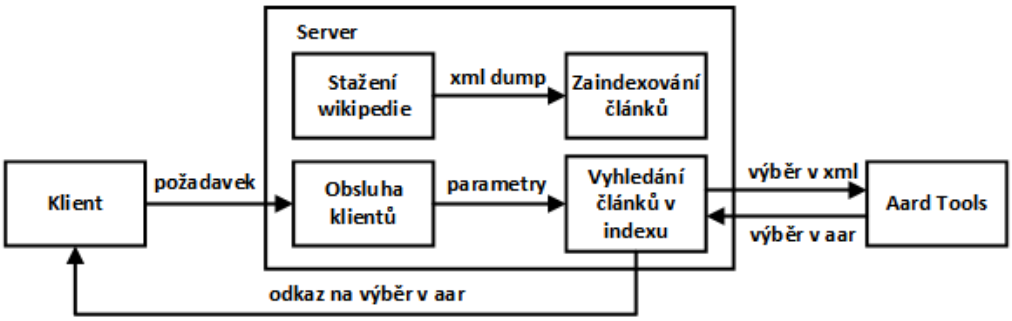
Nabízí se tedy možnost, že výběr bude uživatel provádět na svém počítači a do mobilního zařízení si uloží výsledek. To ovšem zahrnuje instalaci a spuštění potřebných externích nástrojů na tomto počítači, což může především méně zkušeným uživatelům působit obtíže. Nástroje Aard Tools nutné k vygenerování výsledného souboru jsou navíc dostupné pouze pro Unixové systémy. Dále by také celý proces mohl být příliš pomalý v případě nedostatečného výpočetního výkonu daného počítače.

Z výše uvedených důvodů bude tedy vhodné, když celý systém poběží jako server na dostatečně výkonném počítači, na nějž uživatel jen zašle požadavek a převezme si výsledek.

3.2.1 Komunikace s klientem a vyřízení požadavku

Část s názvem „Obsluha klientů“ na obrázku 3.1 bude fungovat jako jednoduchá webová služba. Konkrétně se bude jednat o PHP skript s názvem `wiki_service.php`, který získá od uživatele parametry jeho požadavku prostřednictvím metody `GET`. Tyto parametry budou dále předány části realizující samotný výběr článků. Vyřízení jednoho požadavku může být časově náročné, a proto klient dostane zpět jen informaci o tom, zda byly parametry přijaty a proces byl úspěšně zahájen.

[3] https://lucene.apache.org/core/



Obrázek 3.1: Schéma systému pro výběr článků z Wikipedie.

Po dokončení výběru a vygenerování XML s vybranými články je tento soubor předán ke zpracování nástrojům Aard Tools. Ty z něj vygenerují požadovaný `.aar` soubor, který je následně uložen na vyhrazené místo na serveru. Po dokončení tohoto procesu je uživateli zaslán e-mail s URL, na které si může stáhnout výsledek svého požadavku. Adresa uživatele je součástí předaných parametrů.

3.2.2 Oddělení částí systému

Jednotlivé části systému jsou navrženy tak, aby bylo možné systém používat i jako jednoduchý nástroj v příkazové řádce bez webového rozhraní. To znamená, že část „Obsluha klientů“ může být zcela vypuštěna a bude se pracovat pouze se skripty určenými k analýze a výběru článků. Ty jsou psané v jazyce python3.

Jádro celého systému tvoří část „Vyhledání článků v indexu“ reprezentovaná souborem `xml_creator.py`, který se stará o vygenerování příslušných dotazů na Elasticsearch, zpracování získaných článků, iniciování převodu XML výstupu do formátu pro čtečku a uvědomění klienta o vyřízení jeho požadavku. Seznam parametrů a přepínačů tohoto skriptu je uveden v jeho nápovědě.

Dalším důležitým souborem je `dump_procesor.py`. Ten má na starosti stažení nejnovějšího dumpu Wikipedie, jeho analýzu a vytvoření indexu v Elasticsearch. Tento skript je nutné spustit v rámci inicializace celého systému, jinak by `xml_creator.py` neměl zdroj článků, ze kterých by vybíral.

`Dump_procesor.py` je dále možné použít i pro aktualizaci indexu. Pokud již existuje index Wikipedie v Elasticsearch, skript zkontroluje stáří dumpu, ze kterého byl vytvořen, a pokud najde novější verzi, stáhne jej, vytvoří nový index a starý následně smaže.

3.2.3 Uživatelské rozhraní

Jako uživatelské rozhraní systému pro případ jeho použití jako webové služby byl implementován jednoduchý formulář v jazyce HTML5, který je uložen v souboru `request_form.html`. Rozhraní je ale možné napsat na libovolné platformě a v libovolném programovacím jazyce s využitím jeho webového klienta pro předání parametrů serveru.

Prostřednictvím tohoto formuláře je možné, kromě zadání klíčových slov a e-mailové adresy, ovlivnit také operátor mezi zadanými slovy, oblast vyhledávání (nadpisy, texty nebo kategorie) a přidání odkazovaných článků respektive nejčastějších kategorií.

Aby se zamezilo nepovoleným kombinacím, pomocí JavaScriptu je implementováno i blokování elementů, které jsou v konfliktu s aktuálním zadáním. Pokud například vybereme

jako oblast vyhledávání kategorie, automaticky nám odpadá možnost zahrnout do výsledku odkazované články.

3.3 Práce v Elasticsearch

V této sekci si uvedeme základní prvky Elasticsearch, které jsou v práci využity. Uváděné informace byly čerpány z dokumentace na https://www.elastic.co/ a ze zdroje [5].

3.3.1 Analýza vstupních dat

Analyzátor Elasticsearch v sobě kombinuje tři základní funkce:

Filtrování znaků – vstupní řetězec je „očištěn“ před samotným rozdělením na tokeny (např. odstranění HTML značek apod.)

Tokenizátor – rozdělí řetězec na jednotlivé termy, například podle bílých znaků nebo interpunkce

Tokenový filtr – každý term projde filtrem, který jej může změnit (např. zrušit velká písmena), odstranit zbytečné termy nebo přidat synonyma

Implicitní vestavěný analyzátor používaný v Elasticsearch se nazývá `Standard analyzer`. Termy rozděluje podle hranic slov (mezery, uvozovky apod.), odstraňuje většinu interpunkce a všechny termy převede do malých písmen.

3.3.2 Query DSL a použité konstrukce

Dotazy na Elasticsearch se formulují pomocí jazyka Query DSL založeném na JSON. Základními konstrukcemi jsou *query* a *filtr*.

Query je vhodné pro fulltextové vyhledávání a jeho výsledkem jsou relevantní dokumenty. Zároveň také počítá skóre jednotlivých dokumentů a tím určuje, jak moc vyhovují zadanému dotazu. *Flitr* naopak pouze odpovídá na otázku, zda dokument vyhovuje dotazu nebo ne.

Vyhledávání a filtry

Základní dotaz pro běžné fulltextové vyhledávání je `match` query. Pokud budeme chtít prohledávat tímto způsobem více polí najednou, můžeme použít alternativu `multi_match` query. Příklady obou možností jsou uvedeny níže.

```
{ ,,match‘‘: { ,,title‘‘: ,,About Search‘‘ }}

{
    ,,multi_match‘‘: {
        ,,query‘‘:    ,,full text search‘‘,
        ,,fields‘‘:   [ ,,title‘‘, ,,body‘‘ ]
    }
}
```

Pokud chceme omezit kolekci dokumentů jen na ty, které mají požadovanou hodnotu v zadaném poli, máme k dispozici `term` filtr. Více přípustných hodnot můžeme specifikovat pomocí filtru `terms`.

```
{ ,,term‘‘: { ,,age‘‘: 26 }}

{ ,,terms‘‘: { ,,tag‘‘: [ ,,search‘‘, ,,full_text‘‘, ,,nosql‘‘ ] }}
```

Přizpůsobení mapování

Typy jednotlivých polí a aplikované analyzátory jsou při analýze dokumentu určovány automaticky. Pokud potřebujeme s některým polem pracovat jiným způsobem, je nutné upravit jeho mapování. To lze provést například takto:

```
{
  ,,mappings‘‘: {
    ,,tweet‘‘ : {
      ,,properties‘‘ : {
        ,,tweet‘‘ : {
          ,,type‘‘ :    ,,string‘‘,
          ,,analyzer‘‘: ,,english‘‘
        }}}}}
```

Pomocí mapování také můžeme pole nastavit jako `not_analyzed`. Tím zamezíme jeho rozložení na termy, což nám umožní v něm vyhledávat přesnou shodu.

Hledání podobných dokumentů

Pro vyhledání všech dokumentů v indexu, které mají podobný obsah jako zadaný dokument, respektive množina dokumentů, máme v Elasticsearch k dispozici `more_like_this` query. Z možných parametrů `more_like_this` si uvedeme ty, které pro nás budou důležité:

`ids` – seznam identifikátorů dokumentů, které použijeme jako vstup

`max_query_terms` – maximální počet termů, které budou vybrány jako reprezentativní množina

`min_term_freq` – minimální počet výskytů termu ve vstupním dokumentu, aby nebyl ignorován

`min_doc_freq` – minimální počet vstupních dokumentů, ve kterých se musí term vyskytnout, aby nebyl ignorován

`stop_words` – seznam slov, která budou ignorována

`minimum_should_match` – minimální počet termů z reprezentativní množiny, které se musí vyskytnout ve výsledném dokumentu

3.4 Zaindexování Wikipedie

Při vytváření indexu Wikipedie budeme vycházet z jejího dumpu, konkrétně ze souboru `pages-articles.xml`, jehož struktura byla rozebrána v části 2.2.2. V tomto souboru se však kromě samotných článků vyskytují také stránky v jiných jmenných prostorech Wikipedie (kategorie, šablony…) a také informace v `<siteinfo>`. Tato data není nutné v indexu uchovávat, některá však mohou hrát roli pro čtečku výsledného souboru.

Na samém začátku analýzy dumpu se tedy separuje zmiňovaná část `<siteinfo>` a uloží se do zvláštního *temp souboru* pro danou verzi Wikipedie s názvem například `simplewiki_temp.xml` pro Simple English Wikipedia. Tento soubor se bude později používat jako šablona pro generování výběru článků.

Dále je nutné odfiltrovat nepotřebné stránky týkající se Wikipedie samotné (`Wikipedia:`) a jednotlivých kategorií (`Category:`) tak, že je jednoduše přeskočíme. Jiný přístup však musíme zvolit v případě šablon (`Template:`). Jejich úkolem je pomáhat s navigací mezi články, udávat formátování apod. Pokud bychom je ve výsledném souboru vypustili, čtečka nebude schopná zobrazit některé prvky stránky jako například infoboxy. Proto vždy, když narazíme na `<page>` nesoucí šablonu, přidáme ji do *temp souboru*.

3.4.1 Zpracování článku

Index obsahující texty článků a informace o nich budeme označovat jako *primární index*. Struktura článku v něm bude v podstatě shodná s jeho strukturou v dumpu. Jeho převod do JSON reprezentace je popsán v části 3.4.3. Nevýhodou je, že články v XML nemají explicitně uvedeny, kam se odkazují a do kterých kategorií patří. Vše je obsaženo v elementu `<text>`, což není optimální pro pozdější vyhledávání podobně zaměřených článků. Ještě před jejich nahráním si tedy tyto informace vyextrahujeme a uložíme do zvláštních polí.

Abychom našli všechny odkazy a kategorie v textu článku, je nutné hledat text ohraničený zdvojenými hranatými závorkami (viz 2.2.2). K tomu je použit regulární výraz `{\[\[(.*?)(\]\]|\|)}`. Odkazy na jiné články jsou uloženy do pole `links`. Zpracování odkazů s předponou `Category:` značící příslušnost do kategorie je popsáno v části 3.4.2.

3.4.2 Uložení kategorií

Jak bylo zmíněno v části 2.2.2, Wikipedie nabízí množství kategorií pro sdružování článků s podobným obsahem, čehož je vhodné využít. Pokud při analýze těla článku narazíme na odkaz týkající se kategorie, uložíme si jej do pole `categories`. Narozdíl od pole `links` ale budeme chtít články podle tohoto pole později filtrovat, proto bude obsahovat přesné hodnoty a nebude Elasticsearchem analyzováno. Vždy při vytváření indexu explicitně definujeme mapování pro toto pole:

```
,,mappings‘‘: {
  ,,article‘‘: {
    ,,dynamic‘‘: ,,true‘‘,
    ,,properties‘‘: {
      ,,page‘‘: {
        ,,properties‘‘: {
          ,,categories‘‘: {
            ,,type‘‘: ,,string‘‘,
            ,,index‘‘ : ,,not_analyzed‘‘ }}}}}}
```

Kromě filtrování ale budeme chtít mezi kategoriemi také vyhledávat, proto vytvoříme *sekundární index* jen pro ně. Jednotlivé kategorie bychom mohli odchytávat již na úrovni `<page>` stejně jako šablony, pokud je ale získáme až z článků, máme jistotu, že v indexu budou ty a jen ty kategorie, které se vyskytly u některého z článků. Nebude tak vadit, pokud by náhodou existovala prázdná kategorie nebo některá z kategorií neměla svoji stránku.

Při zjištění kategorie v těle článku tedy kromě jejího přidání do pole `categories` proběhne ještě kontrola, jestli je nalezená kategorie již zaindexovaná, a pokud ne, přidá se její název do indexu.

3.4.3 Převod článku do JSON

Dokumenty nahrávané do Elasticsearch jsou popsány v jazyce JSON, články v dumpu jsou ale v XML a vyžadují proto převod. V části 2.2.1 jsme uvedli, že nemusí být vždy možné převést XML dokument na ekvivalentní JSON dokument. Při práci s dumpem Wikipedie nám však tato omezení nepůsobí potíže, a proto můžeme články převést bez ztráty důležitých informací.

Při převádění XML stromu se řídíme konvencí popsanou na http://www.xml.com/pub/a/2006/05/31/converting-between-xml-and-json.html. Pravidla, která obvykle použijeme při konverzi článku, jsou

```
XML                          JSON
1.  <e/>                     ,,e‘‘:null
2.  <e>text</e>              ,,e‘‘:,,text‘‘
3.  <e name=,,value‘‘ />     ,,e‘‘:{,,@name‘‘:,,value‘‘}

    <e name=,,value‘‘>
4.    text                   ,,e‘‘:{,,@name‘‘:,,value‘‘, ,,#text‘‘:,,text‘‘}
    </e>

    <e>
5.    <a>text</a>            ,,e‘‘:{,,a‘‘:,,text‘‘, ,,b‘‘:,,text‘‘}
      <b>text</b>
    </e>
```

Článek uvedený v XML na obrázku 2.10 tak bude v Elasticsearch reprezentován následujícím JSON dokumentem:

```
{
  ,,page‘‘: {
    ,,title‘‘: ,,Nadpis článku‘‘,
    ,,ns‘‘: ,,0‘‘,
    ,,id‘‘: ,,6‘‘,
    ,,revision‘‘: {
      ,,id‘‘: ,,4896310‘‘,
      ,,parentid‘‘: ,,4831012‘‘,
      ,,timestamp‘‘: ,,2014-09-25T12:18:19Z‘‘,
      ,,contributor‘‘: {
        ,,username‘‘: ,,RandomUser‘‘,
        ,,id‘‘: ,,12345‘‘},
      ,,minor‘‘: null,
```

```
      ,,comment‘‘: ,,Text komentáře‘‘,
      ,,text‘‘: {
        ,,@xml:space‘‘: ,,preserve‘‘,
        ,,#text‘‘: ,,Tělo článku s [[odkazy]] na jiné články.‘‘},
      ,,sha1‘‘: ,,kwsvsphp51dyy3ipoiy4golr6epsq3j‘‘,
      ,,model‘‘: ,,wikitext‘‘,
      ,,format‘‘: ,,text/x-wiki‘‘},
    ,,restrictions‘‘: ,,edit=sysop:move=sysop‘‘,
    ,,links‘‘: [,,odkazy‘‘],
    ,,categories‘‘: []}
}
```

3.5 Výběr článků a jejich zpracování

Po sérii dotazů na Elastichsearch získáme množinu článků, ze které je nutné vytvořit výstup celého systému. Při vytváření indexu z dumpu jsme reprezentaci jednotlivých článků převáděli z XML do JSON, nyní je třeba udělat zpětný převod a vytvořit z výběru podmnožinu původního dumpu.

Abychom neztratili informace o dumpu, které by mohli být později důležité, výstupní soubor inicializujeme jako kopii *temp souboru* se všemi šablonami `Template` a elementem `<siteinfo>`. Jednotlivé články budeme jeden po druhém převádět zpět do jejich původní XML reprezentace a přidávat na konec tohoto souboru.

Po zapsání všech získaných článků doplníme do výstupu značku `</mediawiki>` uzavírající kořenový element celého XML dokumentu a předáme jej k dalšímu zpracování nástrojům Aard Tools, které z něj vytvoří slovník ve formátu `.aar` vhodný pro cílovou čtečku.

Celý vyhledávací proces může proběhnout jedním ze dvou způsobů:

1. klíčová slova se vyhledají mezi kategoriemi Wikipedie a z nalezených kategorií se přidají všechny články,
2. klíčová slova se vyhledají přímo v tělech nebo nadpisech článků a na základě získané množiny se následně vyhledají podobně zaměřené články.

3.5.1 Vyhledávání podle kategorií

Pokud uživatel zvolil vyhledání zadaných slov v kategoriích, je sestaven jednoduchý dotaz na *sekundární index* obsahující jen názvy kategorií. Z *primárního indexu* se pak vyfiltrují všechny články, které spadají do vybraných kategorií:

```
{
  ,,query‘‘: {
    ,,filtered‘‘: {
      ,,filter‘‘: {
        ,,terms‘‘: {
          ,,categories‘‘: [
            kategorie1,
            kategorie2,
            ...
          ]
        }}}}}
```

Tento přístup je výrazně jednodušší, protože nezahrnuje žádnou pokročilejší logiku pro určování článků s podobným obsahem a pracuje pouze kategoriemi definovanými samotnou Wikipedií. I přes to ale může být praktický, pokud uživatel přesné ví, jakou oblast Wikipedie vymezenou právě kategoriemi chce získat. Jako jednoduchý příklad si můžeme uvést uživatele, který by chtěl získat všechny články týkající se fotbalu. Stačí zadat jako klíčové slovo `footbal` a vyhledat odpovídající kategorie.

Získaný výsledek bude mít velmi vysokou *přesnost*, protože se dá předpokládat, že naprostá většina článků v kategorii obsahující v názvu slovo `footbal` bude relevantní, pokud nebudeme striktně odmítat například články o americkém fotbalu. V tomto konkrétním případě bude uspokojivá také *úplnost*, protože velká část článků týkajících se fotbalu patří alespoň do jedné z kategorií, které toto slovo ve svém názvu obsahují.

3.5.2 Vyhledávání v textech článků

Podstatně komplikovanější je vyhledávání klíčových slov přímo v jednotlivých článcích. V tomto případě již nestačí pouze vrátit ty, které obsahují zadanou množinu klíčových slov. Takto získané články nám však mohou sloužit jako *referenční množina*, na jejímž základě se pokusíme vyhledat články s podobným obsahem.

K tomu můžeme využít kromě `more_like_this` query z Elasticsearch také kategorie Wikipedie a odkazy mezi jednotlivými články. Úkolem této metody výběru tedy bude

1. podle klíčových slov vybrat referenční množinu článků,
2. na základě této množiny provést `more_like_this` query,
3. přidat všechny články z nejčastěji zastoupených kategorií ve výběru,
4. přidat všechny odkazované články.

Z uvedeného výčtu jsou první dva body povinné a provedou se vždy, všechny ostatní jsou volitelné a záleží na uživateli, zde je nechá provést nebo ne.

Získání referenční množiny

Při získávání článků na základě klíčových slov můžeme vyhledávat buď pouze v jejich nadpisech, pokud máme konkrétní představu o tom, co chceme získat, nebo i v jejich tělech pro obecnější vyhledávání. Vždy se ale musíme rozhodnout, zda chceme články obsahující všechna zadaná slova (operátor AND) nebo nám stačí, aby obsahovaly libovolné ze slov (operátor OR).

Pokud použijeme operátor AND, získáme malou množinu článků, která ale pravděpodobně bude velmi relevantní, čímž podpoříme *přesnost* výsledku, v případě OR naopak dostaneme velké množství článků, včetně těch, které jsou relevantní, přestože obsahují jen některá ze zadaných slov a zvýšíme tak *úplnost*.

Požadavek na *referenční množinu* je, aby obsahovala nejreprezentativnější články pro požadované téma. Bude nás tedy zajímat spíše *přesnost*, a proto jako výchozí operátor zvolíme AND. Uživatel však může mít i jinou představu o zadávaných klíčových slovech. Například při hledání článků týkajících se budhismu a hinduismu bude pravděpodobně preferovat dotaz `budhism OR hinduism`. Z tohoto důvodu je umožněna i volba operátoru OR.

Pro specifikování náročnějšího požadavku, který si nevystačí jen s AND a OR, je možné také určit, kolik slov ze zadaných musí článek minimálně obsahovat, aby byl vyhodnocen jako relevantní. Jedná se tedy o kompromis mezi uvedenými operátory.

Z těchto parametrů se sestaví dotaz na primární index podle následujícího schématu

```
{
  ,,query‘‘: {
    ,,multi_match‘‘: {
      (,,minimum_should_match‘‘: minimum z klíčových slov,)
      ,,operator‘‘: AND/OR,
      ,,query‘‘: ,,klíčová slova‘‘,
      ,,fields‘‘: [,,title‘‘(, #text)]
        }}}
```

Prvky uvedené v kulatých závorkách se nemusí nutně vyskytnout.

Použití more like this

Ještě před tím, než budeme schopni vůbec vytvořit nějaký dotaz na podobnost, musíme znát množinu článků, podle kterých budeme hledat podobné. Použít celou *referenční množinu* však není moudré. Přestože jejím smyslem je co nejlépe reprezentovat hledanou tématickou oblast, stále obsahuje nemalé množství článků, které jsou relevantní málo nebo vůbec.

Hledání podobnosti pomocí `more_like_this` se ukázalo jako poměrně citlivé na méně relevantní články, a proto za tímto účelem potřebujeme, aby maximum v použitém výběru tvořily ty skutečně nejlepší. Proto se vždy pro tento dotaz použije přibližně 17 % článků z *referenční množiny*. Jak byla tato hodnota stanovena je popsáno v části 3.5.3.

Dále je ještě vhodné specifikovat množinu slov, která nebudou při vyhodnocení brána v úvahu (*stop slova*). Jedná se především o často se vyskytující předložky, spojky, zájmena, ale i některá příslovce a slovesa, která, přetože nenesou žádný význam, mohou díky svým četnostem negativně ovlivnit výsledek. Tato slova jsou uložena ve zvláštním souboru `enwiki-stopwords`.

Když máme vybrány články, podle kterých budeme určovat podobnost, můžeme sestavit `more_like_this` query:

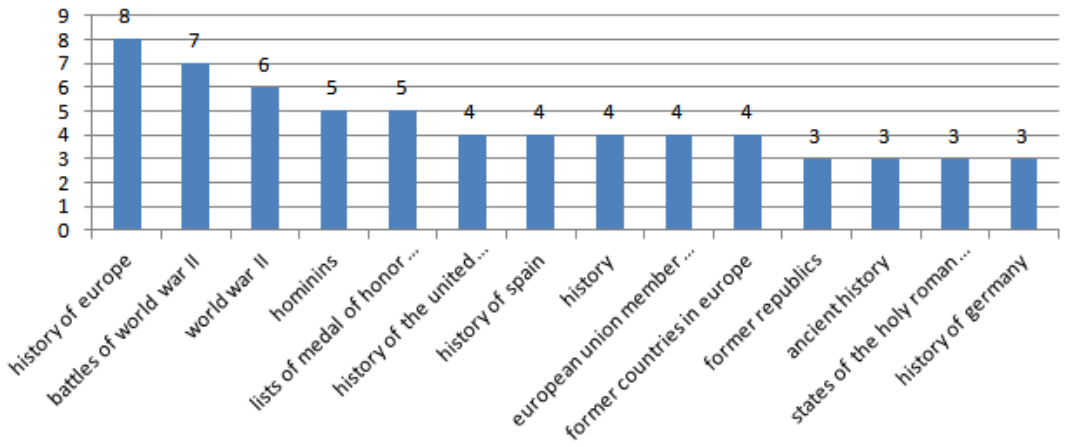
```
{
  ,,query‘‘: {
    ,,more_like_this‘‘: {
      ,,fields‘‘: [,,title‘‘, #text],
      ,,ids‘‘: [seznam id vybranych clanku],
      ,,min_term_freq‘‘: 1,
      ,,min_doc_freq‘‘: 5,
      ,,max_query_terms‘‘: 25,
      ,,stop_words‘‘: [seznam stop words]
        }}}
```

Určení hodnot číselných parametrů `more_like_this` je opět vysvětleno v sekci 3.5.3.

Nejčastější kategorie

O jaké články se uživatel zajímá můžeme odhadovat také podle toho, do jakých kategorií spadala většina článků z *referenční množiny*. Během jejího zpracování si budeme značit,

jaké kategorie se u článků vyskytly a kolikrát. Následně je seřadíme podle počtu výskytů a z několika nejlepších přidáme jejich články.



Obrázek 3.2: Histogram nejčastěji se vyskytujících kategorií pro dotaz `history of europe`

Problémy při tomto postupu způsobují obecné, nic neříkající kategorie jako `days`, `former good articles`, `language-related lists` atd. Některé z nich se mohou vyskytovat poměrně často, což se negativně projeví ve výsledném výběru. Z tohoto důvodu jsou takovéto kategorie ignorovány a články z nich nejsou přidávány.

Jako vhodná optimalizace by se mohlo zdát započítání i kategorií článků získaných na základě dotazu `more_like_this`. Při experimentech se ale ukázalo, že tento krok ovlivní výsledek spíše negativně.

Odkazované články

Pokud se článek Wikipedie odkazuje na nějaký jiný, dá se předpokládat, že tyto články spolu souvisejí. Můžeme tedy rozšířit výběr relevantních článků o ty, které jsou z tohoto výběru odkazované. Abychom ale nezpůsobili velký nárůst velikosti výsledného souboru, nebudeme zahrnovat všechny odkazované články, nýbrž jen ty odkazované z reprezentativních článků, konkrétně z *referenční množiny*.

Při jejím zpracování si budeme z polí `links` postupně tvořit množinu všech článků, na které z ní vedou odkazy. Na závěr celého vyhledávacího procesu se pak přidají články z této množiny, které ještě nebyly do výsledku zařazeny. Vzhledem k tomu, že velká část odkazovaných relevantních článků již byla přidána některým z předcházejících mechanismů, většina takto přidaných článků pravděpodobně bude s hledaným tématem souviset jen okrajově či vůbec. Přesto ale můžeme tímto způsobem získat i uspokojivé množství relevantních článků.

3.5.3 Určení interních parametrů

Vnitřní parametry systému, které uživatel nemá možnost modifikovat, byli určeny experimentálně. Stejné dotazy byly spouštěny s různými hodnotami těchto parametrů a jejich výsledky byly vzájemně srovnávány. Konečné hodnoty byly zvoleny jako kompromis mezi nejlepšími konfiguracemi pro jednotlivé dotazy.

Počet článků pro more like this

Pro nejlepší výsledky `more_like_this` query je třeba zvolit optimální množství článků z *referenční množiny*. Jako příklad si uveďme dotaz hledající články o starověkém Římu, kde klíčová slova budou `ancient rome`. *Referenční množina* obsahuje téměř 110 článků seřazených podle jejich skóre.

Na začátku množiny se vyskytují články jako `Ancient Rome`, `Culture of ancient Rome`, `Military of ancient Rome` apod., které jsou jistě vysoce relevantní pro náš dotaz. Již kolem 20. místa se ale začínají vyskytovat i články jako `History of Armenia`, které s Římem příliš nesouvisí. Na druhou stranu `Colosseum`, další velmi relevantní článek, se vyskytuje až těsně před 40. místem, takže bychom neměli končit s výběrem příliš brzy.

Je tedy potřeba určit hranici, kdy se do výběru začíná míchat příliš velké množství nerelevantních výsledků. Hodnoty tohoto parametru byli testovány v kombinaci s různými hodnotami parametrů `more_like_this`. Napříč experimenty téměř vždy vykazovalo nejlepší výsledky zahrnutí přibližně 17 % článků ze začátku *referenční množiny*. Tato hodnota je použita i ve výsledném řešení.

Parametry more like this

Kromě vhodné množiny článků, na jejichž základě se hledá podobnost, je nutné stanovit také vhodné hodnoty pro parametry uvedené v části 3.3.2.

Pro parametr `min_term_freq` byla zvolena hodnota 1, což znamená, že každý term ve vstupních dokumentech bude při zpracování uvažován. Již při zvýšení této hodnoty na 2 dochází k ignorování termů, které se vyskytly jen jednou, což může v případě kratších článků vyloučit i důležité termy s vysokým tf-idf. V případě již zmiňovaného dotazu `ancient rome` se to projevilo velkým množstvím obecně historických článků a také článků o osobách jménem „Roman“ kvůli mnohoznačnosti tohoto slova. U dotazu `football players` pak výsledek obsahoval mimo konkrétních fotbalistů také podstatně více článků o fotbale samotném.

Při zvolení hodnoty 1 pro parametr `min_doc_freq` byly výsledky velmi dobré, avšak po jeho zvýšení na 5 došlo ke zpřesnění celého výběru. Na rozdíl od `min_term_freq` tak zvýšení neznamenalo nechtěné vyloučení důležitých termů jen proto, že se vyskytly v málo dokumentech, ale naopak došlo k eliminaci těch, které jako významné jen vypadaly.

Poslední parametr, `max_query_terms`, obecně ovlivňuje přesnost celého dotazu. Čím více termů zahrneme jako reprezentativních, tím lepší představu získáme o vstupních dokumentech, ale také zvýšíme náročnost celého dotazu. Při práci s hodnotou 12 bylo ve výsledku i 6x více článků než pro hodnotu 25, která byla později zvolena jako optimální. Zvýšení na tuto hodnotu také pomohlo snížit rozptýlení relevantních článků. Její další zvyšování sice opět zmenšilo celkový výběr, nepřinášelo však další relevantní články, naopak spíše vyřazovalo ty, jež byly na hranici relevantnosti.

Nejlepší články z podobných

Z článků získaných na základě `more_like_this` je nutné ještě vybrat ty, které jsou skutečně relevantní k původnímu dotazu, abychom neplýtvali pamětí přidáním množství zbytečností. K dispozici máme jejich skóre, vezmeme tedy určité množství nejlépe hodnocených článků a budeme doufat, že jsme ze všech relevantních ve výběru vynechali jen naprosté minimum.

Při analýze vrácených článků v jednotlivých experimentech bylo zjištěno, že i při zahrnutí 90 % nejlépe hodnocených stále ještě získáváme relevantní články. Přestože v koncových

částech výběru jsou již dost rozptýlené, zahrnutí několika stovek zbytečných článků způsobí jen minimální nárůst velikosti. Ten se dá vzhledem k zahrnutí více relevantních článků tolerovat.

Počet nejčastějších kategorií

Počet kategorií, ze kterých se zahrnou články do výsledku, je hodnota, představující minimální procento všech výskytů kategorií, které budou obsaženy ve výběru nejlepších kategorií. Pokud bychom uvažovali, že graf 3.2 představuje úplný histogram kategorií pro daný dotaz, měli bychom celkem 63 výskytů. Při hodnotě parametru 50 % bychom tak vzali prvních 6 kategorií, které dohromady obsahují 35 výskytů.

Pro každý z experimentálních dotazů byly zjištěny četnosti jednotlivých kategorií a vypočítána optimální hodnota parametru. Na jejich základě pak byla stanovena jeho výsledná hodnota. Vzhledem k tomu, že rozptyl hodnot byl kvůli rozptylu relevantních kategorií v histogramu velký (20 - 70 %), je zvolená hodnota 40 % velmi volným kompromisem.

Kapitola 4

Vyhodnocení práce

Pro zhodnocení celého systému použijeme především metriky pro měření jeho efektivity vysvětlené v části 2.1.7, jmenovitě *přesnost* a *úplnost*. Pro testovací dotazy nemáme k dispozici seznamy dokumentů, které by měly být vráceny, tudíž tyto hodnoty nejsme schopni určit přesně. Vyhodnocení pomocí *přesnosti* a *úplnosti* tak bude spíše orientační.

Dále také srovnáme implementovaný systém s existujícími systémy řešící podobně orientované problémy a navrhneme možné způsoby, jak celý systém zefektivnit.

4.1 Zhodnocení efektivity systému

Požadavek na výsledný výběr obvykle nejspíš bude, abychom v něm našli maximum článků týkajících se námi specifikovaného tématu, přičemž nám nebudou vadit ty, které s ním vůbec nesouvisí, pokud jejich vinou nebude výsledný soubor příliš velký. Většinou je pro nás tedy klíčová spíše *úplnost* než *přesnost*.

Jak bylo řečeno v úvodu této kapitoly, je v našem případě obtížné tyto hodnoty určit přesně. Není v lidských silách procházet celou Wikipedii a identifikovat všechny články relevantní k danému dotazu. Můžeme se pouze podívat na výstup systému, podle článků přítomných ve výběru přibližně odhadnout *přesnost*, následně v něm zkusit namátkou vyhledat několik článků, které považujeme za relevantní a odhadnout, jestli je i *úplnost* dostatečně uspokojivá.

4.1.1 Efektivita při hledání podle kategorií

Chování systému při tomto způsobu vyhledávání článků již bylo naznačeno v části 3.5.1 včetně vyhodnocení efektivity pro dotaz `football`. Podle tohoto dotazu bychom mohli celkovou efektivitu hodnotit pozitivně, avšak je nutné uvědomit si, co za tímto výsledkem skutečně stojí.

Téma fotbal je mezi kategoriemi Wikipedie poměrně dobře vymezené a téměř všechny kategorie týkající se tohoto tématu obsahují ve svém názvu klíčové slovo `football`. Při hledání jediného klíčového slova také získáme daleko více kategorii, než při hledání kombinace více slov. Dalším pozitivním činitelem je i fakt, že samotné slovo `football` je dostatečně jednoznačné. Jeho jediný další význam je ve smyslu již zmiňovaného amerického fotbalu, což se stále dá považovat za relevantní oblast.

Pokud bychom použili stejný přístup například pro vyhledání slova `health`, jistě bychom opět měli vysokou *přesnost*, podstatně horší už by ale byla *úplnost*. Vzhledem k tomu, že

podkategorie nejsou automaticky zahrnovány, by se ve výsledku neobjevily články z kategorií jako `Diseases and disorders`, `Drugs` apod., přestože se dají považovat za velmi relevantní. Pro dobrý výsledek by tedy bylo nutné v dotazu kombinovat více klíčových slov s operátorem OR.

Dotazy na kategorie jsou obecně orientovány spíše na *přesnost* než *úplnost* a využijí je především uživatelé, kteří se orientují v kategoriích Wikipedie a jsou pomocí nich schopni specifikovat svůj předmět zájmu.

4.1.2 Efektivita při hledání v článcích

Pro zhodnocení tohoto přístupu se podívejme podrobněji na dotaz `football players` bez zahrnutí nejčastějších kategorií a odkazovaných článků nad Simple English Wikipedia. Jako výstup bychom čekali primárně články týkající se konkrétních hráčů, jako přijatelné pak můžeme brát i články o fotbalu obecně.

Referenční množina obsahuje přes 300 článků, které se týkají převážně fotbalistů, fotbalových týmů a fotbalu samotného, a je tedy velmi dobrým reprezentativním vzorkem toho, co hledáme. Dotaz `more_like_this` vrátil 3 262 podobných článků, z nichž kolem 2 900 bylo zahrnuto do výsledku. Na začátku této množiny se vyskytuje překvapivě mnoho článků o hokejových hráčích. Za nimi však následuje velké množství fotbalistů, občas klubů apod.

Na základě experimentování s Explain API v Elasticsearch bylo zjištěno, že nepřesnost výběru je způsobena především obecnějšími sportovními slovy jako `game`, `play`, `league`, `player` atd. I přes to odhaduji, že 60 – 70 % všech vybraných článků je relevantních k původnímu dotazu. Na druhou stranu byli nalezeni i poměrně známí hráči kteří ve výběru chyběli.

Velikost výsledného souboru je necelých 8 MB, což je přibližně 5 % velikosti celé Simple Wikipedie. Pro tento konkrétní dotaz se tedy systém jeví jako efektivní.

Trochu odlišné bylo chování v případě dotazu `ancient rome`. Při něm jsme z `more_like_this` zahrnuli 770 článků, přičemž ale jen malé množství (odhadem okolo 30 %) bylo skutečně relevantních. Jednalo se především o články o Římu, bitvách s jeho účastí, osobách spojených s Římem apod. Články na hranici relevance pak byli nejčastěji o papežích a letopočtech z období Římské říše (nejvíce z období 500 př.n.l – 500 n.l).

Podstatné zlepšení pro dotaz znamenalo přidání nejčastějších kategorií, mezi kterými byly `roman emperors`, `wars and battles of ancient rome` apod. Z celkem 24 přidaných kategorií bylo 14 velmi nebo alespoň dostatečně relevantních.

Celková *přesnost* se tak může pohybovat mezi 30 a 40 %, *úplnost* pro tento dotaz bych opět hodnotil jako přinejmenším dostačující, přestože ve výsledku chybí například většina článků z římské mytologie.

4.2 Návrhy na vylepšení

4.2.1 Zlepšené zpracování vstupních dat

Abychom zvýšili přesnost celého systému, bylo by vhodné provádět změny již na úrovni analýzy vstupních dat. Pozitivní dopady by pravděpodobně mělo například inteligentní přidávání synonym k termům, případně optimalizovaní jejich tvorby pro potřeby vyhledávání mezi články Wikipedie.

Problémy dělá především značkování použité v tělech textů, díky kterému vznikají i termy jako `==` nebo {{`link`. Aby se minimalizoval jejich negativní vliv především na

more like this query, nejsou tyto termy při jeho zpracování uvažovány, jedná se však pouze o potlačení problému, nikoliv jeho řešení.

Dále by na úrovni analýzy mohl být aplikován také vhodný seznam *stop slov*, aby nám dotazy zbytečně neovlivňovaly předložky, členy apod. V implementován řešení jsou tato slova odfiltrována pouze ze zadávaných dotazů.

Stejný seznam je použit také pro vyhledávání podobných článků pomocí more like this query a jeho obsah by mohl být předmětem diskuze.

4.2.2 Skóre relevantních článků

Další prostor pro zlepšení je i u samotného hodnocení článku. Obyčejné `multi_match` query řeší pouze to, jestli se zadaná slova v dokumentu vyskytla či nikoli. Bylo by ale jistě vhodné, aby dokument obsahující přímo frázi `united kingdom` byl hodnocen lépe, než dokument pojednávající o nějakém království (`kingdom`) obecně, které bylo kdysi sjednoceno (`united`).

Lepší hodnocení by dále mohly mít také články, které zadaná slova obsahují v nadpisech, případně kategoriích. Vzhledem k tomu, že pole `categories` v indexu není analyzované (`not_analyzed`), museli bychom mít u každého článku i duplicitní pole s kategoriemi rozdělenými na termy.

4.2.3 Zahrnutí podkategorií

Dosud uvedené návrhy by měly za efekt především zvýšení *přesnosti* dotazů. Pro zvýšení ukazatele *úplnost* se nabízí zahrnout do výsledku i podkategorie, a to jak při vyhledávání přímo v kategoriích, tak i v případě přidávání nejčastěji se vyskytujících kategorií v *referenční množině*.

Nemůžeme však jen slepě zahrnout všechny podkategorie ze získaného výběru, byť většina bude s velkou pravděpodobností relevantní. V části 2.2.2 jsme uvedli, že v hierarchii nejsou vyloučeny cykly, tudíž bychom museli buď omezit hloubku zanoření, což by ale mohlo vést k nezahrnutí velkého množství důležitých článků, nebo být schopni případný cyklus detekovat.

Další problém by nastal při vybrání kategorie příliš blízko u kořene celé struktury (například `Culture`). To by mohlo mít za následek přidání daleko větší části Wikipedie, než bychom čekali, a výsledná úspora paměti by byla příliš malá. Jako řešení se opět nabízí omezit hloubku zanoření. Rovněž by bylo vhodné zamyslet se nad heuristikami, které by umožnily vybrat nejrelevantnější z podkategorií, a tím by umožnily bezpečně zvětšit hloubku maximálního zanoření.

Při výběru nejčetněji zastoupených kategorií by si rozšíření zasloužil i seznam obecných kategorií, které nikdy nebudou přidávány. Ve výsledném řešení je implementován spíše pro demonstrační účely a jeho obsah by opět mohl být předmětem diskuze. Vzhledem k velkému počtu kategorií Wikipedie je však obtížné všechny takové kategorie identifikovat.

4.2.4 Příchozí odkazy

V části 3.5 jsme popisovali přidání článků, které jsou odkazované z *referenční množiny*. Stejným způsobem bychom mohli přidat i články, které se na referenční množinu odkazují.

Vzhledem k tomu, že z dumpu nelze přímo vyčíst, z jakých článků je konkrétní článek odkazován, bylo by nutné dump při zpracování projít dvakrát. Prvním průchodem bychom pro každý článek určili množinu článků, které se na něj odkazují, druhý by pak provedl zaindexování článků i s polem pro příchozí odkazy.

4.2.5 Výstup pro Aard 2

Během psaní této práce byla vydána beta verze čtečky Aard 2, která, na rozdíl od svého předchůdce a původní cílové čtečky Aard Dictionary, využívá formát `.slob`.

Z hlediska budoucí použitelnosti systému by tedy bylo vhodné upravit generování výstupu právě do tohoto formátu, případně do něj automaticky převádět vznikající `.aar` slovníky.

4.3 Porovnání se systémy řešící podobné problémy

4.3.1 Book creator na Wikipedii

Wikipedie samotná nabízí uživatelům možnost sestavit si z jejích článků něco jako knihu. Vybrané články lze organizovat do kapitol, řadit abecedně, případně je uspořádat jak uznáme za vhodné. Knihu je pak možné stáhnout ve formátu PDF nebo dokonce i objednat v tištěné verzi. Součástí PDF jsou i obrázky, funkční odkazy v textu vedoucí na online verze článků, externí odkazy, seznam zdrojů atd.

Tvorba knihy

Po spuštění Book creator je možné volně procházet Wikipedií a vybírat si články, které bychom chtěli mít v naší knize. Na každé stránce se zobrazuje rámeček ukázaný na obrázku 4.1, díky němuž je možné daný článek jednoduše přidat. Zároveň v něm také vidíme, kolik článků již v knize máme a můžeme se do ní rychle přepnout.

Obrázek 4.1: Rozhraní nástroje Book Creator pro tvorbu knihy z článků Wikipedie

Pokud najedeme na některý z odkazů na jiný článek v textu, zobrazí se malý pop-up box nabízející přidat odkazovaný článek. Pro přidání článku tak ani nemusíme navštívit jeho stránku.

Dalším prvkem usnadňujícím výběr je možnost přidat do knihy celou kategorii. Na stránkách kategorií se možnost `Add this page to your book` změní na `Add this category to your book`. Po kliknutí na tuto možnost se do knihy přidají všechny články z této kategorie. Zahrnutí podkategorií je nutné udělat manuálně.

Book creator dále nabízí také doporučení podobných článků. Po kliknutí na možnost `Suggest pages` je uživateli předloženo až 10 článků, týkající se podobného tématu jako články již zařazené v knize. Při vybrání `Biology`, `Organism` a `Life` mi byly doporučeny články `Cell (biology)`, `Species` apod. Z deseti takto vybraných článků považuji devět za vysoce relevantní, pouze článek `Earth` je na hranici relevance. Při přidávání dalších článků týkajících se genetiky a evoluce se v doporučení častěji objevovaly články relevantní i k tomuto zúženému tématu. Přesnost doporučovaných článků tedy hodnotím jako velmi vysokou.

Porovnání

Book creator, který je přímo součástí stránek Wikipedie, umožňuje uživatelům vytvářet knihu z jednotlivých článků a tu si následně stáhnout a uložit. K usnadnění této práce nabízí kromě přidávání jednotlivých článků také možnost zahrnout všechny články z vybrané kategorie a nabídku článků zaměřených podobně jako množina již vybraných.

Hlavní nevýhodou je, že celý výběr musí uživatel provést manuálně, což jej bude stát poměrně dost času. Sice z toho logicky plyne, že ve výběru nebudou zbytečné články (uživatel nezahrne, co jej nezajímá), je ale velká šance, že při výběru přehlédne nemalé množství článků, které by ve výsledku byly rovněž užitečné.

Výstup v podobě PDF pak očividně není myšlen, aby šetřil místo v paměti oproti kompletní offline verzi Wikipedie. Soubor obsahující pouhých 25 článků má velikost přes 25 MB. V sekci 4.1.2 jsme uváděli, že výstup z našeho systému obsahoval přibližně 3 000 článků a jeho velikost byla necelých 8 MB. Nutno ovšem připomenout, že jeho součástí nebyly obrázky a že formát `.aar` je navržen tak, aby byl paměťově úsporný.

4.3.2 The Wikipedia Corpus

The Wikipedia Corpus[1] obsahuje kompletní texty článků anglické Wikipedie, ze kterých umožňuje vytvářet vlastní virtuální korpusy zaměřené na vybrané téma. Z těchto korpusů se pak dají extrahovat seznamy klíčových slov, frekventovaných slovních spojení atd.

Při vytváření korpusu je možné buď zadat klíčová slova a vyhledat je v textu článků, nebo použít vyhledání slov v nadpisech článků. Součástí druhého přístupu je také možnost specifikovat, která slova se naopak nesmí vyskytnout v článcích nebo jejich nadpisech a která se mají vyskytnout v textu. Při zadávání klíčových slov je možné použít i znak „*“ známý z regulárních výrazů zastupující libovolný text. Korpus je možné později manuálně upravovat, přidávat a odebírat články.

Z vytvořených korpusů pak lze získávat statistické údaje, jako například jaká slova se v nich nejčastěji vyskytují. Aby výsledek nebyl zkreslený slovy, která jsou frekventovaná v rámci celé Wikipedie, máme možnost definovat míru specifičnosti slov pro daný korpus. Můžeme tak získat slovní zásobu vystihující vybranou tématickou oblast, které se korpus týká. Kromě nejčastějších slov můžeme hledat také nejčastější dvojice podstatných jmen, případně nejčastější spojení podstatného jména s přídavným jménem.

Korpusy dále umožňují vyhledání údajů o konkrétním slově. Můžeme tak opět vyhledat informace o jeho výskytech, zároveň ale můžeme získat i informace o jeho kontextu, tedy jaká slova se nejčastěji vyskytují v jeho zadaném okolí nebo v jakých slovních spojeních obvykle vystupuje.

Porovnání

Smyslem The Wikipedia Corpus pochopitelně není vybírat z Wikipedie články za účelem co nejpřesnějšího vymezení zadaného tématu a jeho offline uložení. Jeho hlavní použití je spíše pro analýzu jazyka vybrané tématické oblasti, její slovní zásoby atd.

Vybírání článků je tedy založeno jen na základních mechanismech posuzujících, jestli se zadaná slova v článku vyskytla nebo ne, a nenabízí nic jako výběr podobně zaměřených článků. Pokud chceme získat přesněji vymezenou množinu článků, je nutné buď chytře nastavit, která klíčová slova se mají vyskytovat a která ne, nebo výběr manuálně upravit.

[1] http://corpus.byu.edu/wiki/

Kapitola 5

Závěr

V rámci této práce byl navrhnut a naimplementován systém pro výběr tematicky zaměřených článků z Wikipedie. Tento systém standardně funguje jako jednoduchá webová služba, které uživatel předá parametry, po vyřízení jeho požadavku je mu zaslán zpět email s odkazem pro stažení výsledného souboru. Je však možné jej používat i jako konzolovou aplikaci.

Výběr článků lze realizovat buď na základě kategorií Wikipedie, nebo podle výskytu klíčových slov v nadpisech a textech jednotlivých článků. V případě druhého přístupu se systém snaží určit, o jakou tématickou oblast se uživatel zajímá, a přidává do výsledného výběru i další články z této oblasti, přestože se v nich přímo nevyskytují zadaná slova. K tomu využívá především mechanismy pro určení podobných dokumentů a zahrnutí všech článků z kategorií, které se ve výběru často opakují.

Výstup celého systému je ve formátu `.aar`, který je čitelný offline čtečkou Wikipedie Aard Dictionary. Jedním ze základních požadavků na tento výstupní soubor byla minimální velikost, aby se uspořila paměť, kterou články z Wikipedie v offline podobě zabírají. Velikost výstupních souborů je závislá na velikosti hledané tematické oblasti. I v případě testovacích dotazů na rozsáhlejší oblasti Simple English Wkipedie se však pohybovala pod 30 MB.

Literatura

[1] Abiteboul, S.; Manolescu, I.; Rigaux, P.; aj.: *Web Data Management*. Cambridge University Press, 2011, iSBN 11-07012-43-0.

[2] Bellomi, F.; Bonato, R.: Network analysis for Wikipedia. In *proceedings of Wikimania*, 2005.

[3] Bird, S.; Klein, E.; Loper, E.: *Natural language processing with Python*. O'Reilly, 2009, iSBN 05-96516-49-5.

[4] Buckley, C.; Voorhees, E. M.: Retrieval evaluation with incomplete information. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, 2004, s. 25–32.

[5] Gormley, C.; Tong, Z.: *Elasticsearch: The Definitive Guide*. O'Reilly Media, 2015, iSBN 978-1-4493-5854-9.

[6] Hiemstra, D.: Information retrieval models. *Information Retrieval: searching in the 21st Century*, 2009: s. 2–19.

[7] Holloway, T.; Bozicevic, M.; Börner, K.: Analyzing and visualizing the semantic coverage of Wikipedia and its authors. *Complexity*, ročník 12, č. 3, 2007: s. 30–40.

[8] Manning, C. D.; Raghavan, P.; Schütze, H.: *Introduction to Information Retrieval*. Cambridge University Press, 2008, iSBN 05-21865-71-9.

[9] Miller, M. S.: Safe Serialization Under Mutual Suspicion. [online], Naposledy navštíveno 11. 4. 2015.
URL http://erights.org/data/serial/jhu-paper/intro.html

[10] Pilászy, I.: Text Categorization and Support Vector Machines. In *6th International Symposium of Hungarian Researchers on Computational Intelligence*, 2005.

[11] Powers, D.: Evaluation: from Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation. *Journal of Machine Learning Technologies*, ročník 2, č. 1, 2011: s. 37–63.

[12] van Rijsbergen, C. J.: *Information Retrieval*. Butterworths, druhé vydání, 1979.

[13] Salton, G.; Buckley, C.: Term-weighting approaches in automatic text retrieval. *Information processing & management*, ročník 24, č. 5, 1988: s. 513–523.

[14] Schönhofen, P.: Identifying document topics using the Wikipedia category network. *Web Intelligence and Agent Systems*, ročník 7, č. 2, 2009: s. 195–207.

[15] Sebastiani, F.: Machine learning in automated text categorization. *ACM Computing Surveys*, ročník 35, č. 1, 2002: s. 1–47.

[16] Singhal, A.: Modern Information Retrieval: A Brief Overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, ročník 24, č. 4, 2001: s. 35–43.

[17] Voorhees, E. M.: The philosophy of information retrieval evaluation. In *Evaluation of cross-language information retrieval systems*, Springer, 2002, s. 355–370.

[18] Zesch, T.; Gurevych, I.: Analysis of the Wikipedia Category Graph for NLP Applications. In *Proceedings of the TextGraphs-2 Workshop*, 2007.