VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

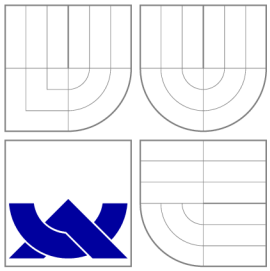
METODY SUMARIZACE DOKUMENTŮ NA WEBU

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

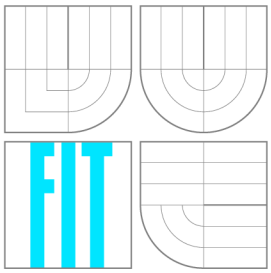
AUTOR PRÁCE
AUTHOR

Bc. MICHAL BELICA

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

METODY SUMARIZACE DOKUMENTŮ NA WEBU

METHODS OF DOCUMENT SUMMARIZATION ON THE WEB

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MICHAL BELICA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VLADIMÍR BARTÍK, Ph.D.

BRNO 2013

Abstrakt

Práce se zabývá sumarizací dokumentů ve formátu HTML. Jako jazyk webových dokumentů byla zvolena čeština. Práce je zaměřená na algoritmy sumarizace textů. Součástí práce je však i předzpracování sumarizovaného dokumentu a převod textu na reprezentaci vhodnou pro sumarizační algoritmy. Práce krátce pojednává o všeobecném dolování textů a později se zaměřuje na sumarizaci. Představené jsou dva jednoduché sumarizační algoritmy, přičemž práce se zaměřuje na pokročilý algoritmus využívající latentní sémantické analýzy. Výsledkem práce je návrh a implementace sumarizačního modulu pro jazyk Python. Souhrny generované implementovanými sumarizačními metodami jsou v závěrečné kapitole porovnány pomocí evaluačních metod i z pohledu subjektivního hodnocení autora práce.

Abstract

The work deals with automatic summarization of documents in HTML format. As a language of web documents, Czech language has been chosen. The project is focused on algorithms of text summarization. The work also includes document preprocessing for summarization and conversion of text into representation suitable for summarization algorithms. General text mining is also briefly discussed but the project is mainly focused on the automatic document summarization. Two simple summarization algorithms are introduced. Then, the main attention is paid to an advanced algorithm that uses latent semantic analysis. Result of the work is a design and implementation of summarization module for Python language. Final part of the work contains evaluation of summaries generated by implemented summarization methods and their subjective comparison of the author.

Klíčová slova

dolování z dat, sumarizace textů, redukce dat, extrakce dat z webu, Python, NLP, zpracovávání přirozeného jazyka, latentní sémantická analýza, LSA, singulární dekompozice, SVD

Keywords

data mining, text summarization, data reduction, web-data extraction, Python, NLP, natural language processing, latent semantic analysis, LSA, singular value decomposition, SVD

Citace

Michal Belica: Metody sumarizace dokumentů na webu, diplomová práce, Brno, FIT VUT v Brně, 2013

Metody sumarizace dokumentů na webu

Prohlášení

Prohlašuji, že jsem tuto práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Michal Belica
20. května 2013

Poděkování

Tímto bych rád poděkoval Ing. Vladimírovi Bartíkovi, Ph.D. za pomoc, kterou mi poskytl při řešení práce.

Obsah

Kapitola 1

Úvod

Automatická sumarizácia textových dokumentov je metóda, pri ktorej sa strojovo vytvára súhrn (abstrakt, extrakt, ...) zo sumarizovaného dokumentu pri súčasnom zachovaní hlavnej myšlienky pôvodného textu. To je okrem iného výhodné pre oboznámenie sa s hlavnými témami pôvodného textu v krátkom čase a bez jeho prečítania. Problémom pri automatickom vytváraní súhrnu je identifikácia hlavných tém dokumentu a následná extrakcia viet, ktoré identifikované témy najlepšie vystihujú. Tento problém v súčasnosti veľmi dobre rieši algebraická metóda LSA[1], ktorou sa práca zaoberá. V práci sú však popísané i dve jednoduchšie metódy sumarizácie a ich možné vylepšenia v rámci možností, ktoré vstupný formát dokumentu ponúka.

Text práce sa zaoberá tvorbou súhrnu z dokumentov vo formáte HTML[2], ktoré sú bežne používané vo webovom prostredí. Formát HTML znamená pre úlohu sumarizácie výhody i nevýhody, ktorým sa v rámci predspracovania dokumentu v práci venujem. Predspracovanie dokumentu veľmi úzko súvisí s prirodzeným jazykom, v ktorom je dokument napísaný. V mojej práci si kladiem za cieľ spracovávať dokumenty napísané v českom jazyku. To síce prináša komplikácie z dôvodu nedostatku nástrojov pre spracovanie prirodzeného jazyka, ale zároveň prínos pre českú NLP[3] komunitu.

Mojou úlohou bolo navrhnúť a následne v jazyku Python implementovať aplikáciu, ktorá automaticky vytvára súhrny z webových dokumentov. Hlavný prínos aplikácie vidím v automatickom spracovaní dokumentov vo formáte HTML, ktorý je dnes dominantným formátom na webe. Užívateľ aplikácie má tak možnosť pohodlne získať súhrnné informácie o témach, ktoré sú na internete k dispozícii.

Práca je členená do niekoľkých častí. V kapitole 2 sa venujem všeobecným témam dolovania textu, akými sú reprezentácia dokumentu a redukcia dimenzionality textových dát. Kapitola 3 popisuje úkony, ktoré je potrebné vykonať, aby bolo možné dokument čo najlepšie sumarizovať. Kapitola 4 sa zaoberá samotnou sumarizáciou, jej rozdelením a problémom hodnotenia. Kapitola 5 podrobne predstavuje vybrané metódy sumarizácie dôležité pre túto prácu. V ďalšej kapitole 6 je opísaná architektúra aplikácie spolu so špecifikáciou funkčnosti. V kapitolách 7 a 8, popisujúcich implementáciu a experimenty s implementovanou aplikáciou, som využil poznatky z predošlých kapitol vytvorených v rámci semestrálneho projektu. V záverečnej kapitole 9 sú zhrnuté poznatky z tejto práce, jej hlavné prednosti i nedostatky a navrhnuté možné rozšírenia.

[1] LSA - Latent Semantic Analysis, latentná sémantická analýza
[2] HTML - HyperText Markup Language
[3] NLP - Natural Language Processing

Kapitola 2

Dolovanie textových dát

Podľa Feldmana a Sangera [4] je dolovanie textu intenzívny poznávací proces, pri ktorom užívateľ pracuje s dokumentom alebo kolekciou dokumentov s využitím sady analytických nástrojov. Analogicky k všeobecnému dolovaniu dát sa v dokumente vyhľadávajú užitočné informácie na základe identifikácie zaujímavých *príznakov* (angl. *features*). Feldman so Sangerom [4] považujú za najčastejšie príznaky v textových dokumentoch nasledovné:

- **znaky** – Nie sú zvlášť zaujímavé a uvádzajú sa len pre úplnosť reprezentácie dokumentu.
- **n-gramy** – Sú viacej užitočné ako samostatné znaky, pretože obsahujú čiastočnú poziční informáciu. Väčšinou sa pracuje s počtom n-gramov, ktorý je rôzny v závislosti od jazyka alebo typu dokumentu.
- **slová** – Základný prvok dokumentu, ktorý nesie určitý význam. Niektoré slová však slúžia len ako syntaktický prvok a nenesú žiadnu hodnotnú informáciu.
- **slovné spojenia** – Jedná sa o spojenie viacerých slov, ktoré sú v jazyku zaužívané alebo majú ucelený význam.
- **vety** – Vyjadrujú ucelenú myšlienku.
- **koncepty** – Výrazy, ktoré sa nemusia nachádzať v dokumente a v niektorých prípadoch ich dokonca nie je možné jazykovými prostriedkami vyjadriť. Podľa Maternu [12] si koncept môžeme predstaviť napríklad ako tému dokumentu, ktorej reprezentáciou je vektor slov s priradenými váhami. Koncept „šport“ by mohol byť reprezentovaný ako $(1{,}5 \cdot \text{šport} + 0{,}8 \cdot \text{futbal} + 0{,}7 \cdot \text{hokej} + 0{,}7 \cdot \text{tenis} + \dots)$.

2.1 Štruktúra textových dát

Na rozdiel od klasického dolovania dát sú dáta v textových dokumentoch neštruktúrované prípadne polo-štruktúrované[1] (napr. HTML dokument). Avšak, ako sa uvádza v *The Text Mining Handbook* [4], v závislosti od pohľadu na textové dáta je v nich vždy možné vidieť druh štruktúry v podobe „mäkkého značkovania“. Objekty ako interpunkčné znamienka, veľkosť písmen, číslice a rôzne špeciálne znaky vyjadrujú syntaktickú štruktúru textu. Skúmanie tejto štruktúry a dokonca lexikálna či sémantická analýza textu sú techniky používané

[1] polo-štruktúrované dáta – dáta bez presne stanoveného formátu, ale s rozoznateľnými spoločnými znakmi

v rámci oblasti zvanej *spracovanie prirodzeného jazyka*. Pri dolovaní textov sa takmer nikdy nevyhneme použitiu syntaktickej, lexikálnej či sémantickej analýzy textu vo fáze predspracovania dokumentu. Hlavnou úlohou predspracovania je totiž previesť implicitnú štruktúru dokumentu na explicitnú tak, aby bolo možné s dátami rozumne pracovať a identifikovať v nich zaujímavé príznaky.

2.2 Reprezentácia dokumentu

Na základe vyššie spomenutých príznakov sa vytvárajú modely dokumentu. K základným modelom, ktoré boli používané ako prvé, patrí model nazývaný *Bag of Words* (skrátene BoW). Jedná sa o množinu slov vyskytujúcich sa v dokumente. Tento model sa tiež označuje ako *booleovský* prípadne *binárny*, pretože poskytuje len informáciu o tom či sa slovo v dokumente vyskytuje (booleovská hodnota `TRUE`) alebo nie (hodnota `FALSE`). Model BoW má nevýhodu v tom, že sa pri množinovej reprezentácii, ktorú používa, stráca informácia o usporiadaní slov v dokumente. Ďalšou nevýhodou je binárna informácia o výskyte slov v dokumente, čo sa často rieši priloženou dodatočnou informáciou o počte výskytov slova v dokumente.

Metrika, ktorej úlohou je zohľadňovať počet slov v dokumente, sa nazýva *term frequency* $\mathrm{TF}(t,d) = \mathrm{f}(t,d)$. Vyjadruje počet slov, respektíve všeobecne termov t, v dokumente d. Nevýhodou tejto metriky pre kolekciu dokumentov je to, že nezohľadňuje dĺžku dokumentu. Preto sa výsledok často normalizuje podľa vzorca 2.1, kde $\mathrm{MaxFreq}(d)$ vyjadruje počet výskytov najfrekventovanejšieho termu v dokumente d a s je parameter, ktorého úlohou je tlmiť príspevok druhého výrazu. V publikácii *Introduction to Information Retrieval* [11] sa uvádza, že parameter $0 \leq s < 1$ má zvyčajne hodnotu 0,4, pričom v skorších prácach sa používala hodnota 0,5. Normalizácia metriky $\mathrm{TF}(t,d)$ je však pre kolekciu dokumentov stále nedostatočná, pretože neuvažuje slová, ktoré sa v dokumentoch vyskytujú bežne, a teda nemajú pre jeden dokument veľkú významnosť. Riešením je metrika *inverse document frequency* $\mathrm{IDF}(t)$ vyjadrená vzorcom 2.2, kde N je celkový počet dokumentov a $\mathrm{DocFreq}(t)$ počet dokumentov, v ktorých sa vyskytuje term t. Výsledná váha termu v dokumente je potom daná ako výsledok výrazu $\mathrm{TF}(t,d) \cdot \mathrm{IDF}(t)$ (ďalej len TF*IDF).

$$\mathrm{TF}(t,d) = s + (1-s)\frac{\mathrm{f}(t,d)}{\mathrm{MaxFreq}(d)} \tag{2.1}$$

$$\mathrm{IDF}(t) = 1 + \log\left(\frac{N}{\mathrm{DocFreq}(t)}\right) \tag{2.2}$$

V súčasnosti je uspokojivou reprezentáciou dokumentu vektorový model spolu s metrikou TF*IDF. Vo vektorovom modeli je každý dokument reprezentovaný vektorom, v ktorom každý term vyjadruje jednu dimenziu dokumentu. Súradnice vo vektore potom obsahujú hodnotu metriky TF*IDF pre každý z týchto termov. Pri práci s reálnymi dokumentami je možné pozorovať, že vektory budú obsahovať počet dimenzií v ráde až stotisícok. Navyše každý vektor bude *riedky* (angl. *sparse*), pretože každý dokument bude obsahovať len malú časť celkovej množiny termov obsiahnutých v celej kolekcii dokumentov. Kvôli tomu je potrebné počet dimenzií vhodne redukovať. Materna [12, 3. diel] a Feldman so Sangerom [4] sa zhodujú, že existujú dva prístupy k zníženiu dimenzionality dokumentov.

2.3 Redukcia dimenzionality dokumentov

Výber príznakov (angl. *feature selection*) je postup, pri ktorom sa z pôvodnej množiny príznakov vyberú tie, ktoré majú pre úlohu najväčší význam. Vybrané príznaky však zostávajú nezmenené. Často sa pre vytvorenie množiny vhodných príznakov používa prístup založený na odstraňovaní príznakov s najmenším významom. Ako kandidáti pre odstránenie sú vhodné slová, ktoré nenesú žiaden sémantický význam. Mnoho systémov však vykonáva oveľa agresívnejšie filtrovanie príznakov. Medzi jednoduchšie metriky patrí napríklad DocFreq(t), kedy sa zahodí niekoľko percent najmenej frekventovaných slov. Ak sa na spôsob výberu príznakov pomocou DocFreq(t) pozrieme lepšie, zistíme, že sa jedná o jednoduchú myšlienku. Vo vektoroch dokumentov vyhľadáme a vynecháme termy, ktoré sa vo väčšine dokumentov nevyskytujú. Budeme pri tom predpokladať, že sa jedná o preklepy a chyby. Ak vynecháme slová s frekvenciou menšou ako 10, tak môžeme podľa Maternu [12, 3. diel] znížiť počet slov až na 17 %. Zo sofistikovanejších metrík filtrovania príznakov možno použiť napríklad *information gain* alebo *chi-square*, ktorých bližší popis je uvedený v publikácii *The Text Mining Handbook* [4].

Extrakcia príznakov (angl. *feature extraction*) je postup, pri ktorom sa pôvodné príznaky nahradia inými, ktorých počet je nižší. V podstate ide o úlohu zhlukovania, pri ktorej môžeme využiť znalosti o gramatických pravidlách prirodzeného jazyka. Bežne používanými prístupmi extrakcie príznakov sú *lematizácia* a *stemming* (bližší popis v sekcii 3.3), ktoré dokážu nahradiť rôzne tvary slov jediným spoločným tvarom. No použitie týchto metód so sebou prináša niektoré nové problémy, akými sú napríklad *nejednoznačnosť slov* alebo *synonymia*, kedy slovo „oko“ môže mať niekoľko rozličných významov, ale naopak slová ako „požiarnik“ a „hasič“ vyjadrujú rovnakú entitu.

Kapitola 3

Predspracovanie dokumentu

Pred samotným procesom sumarizácie je potrebné previesť HTML dokument do formy vhodnej pre spracovanie počítačom. Prvým problémom je odstránenie nežiadúcich prvkov z HTML dokumentu a extrakcia sumarizovaného textu. Následne je nutné z extrahovaného textu vytvoriť štrukturovaný dokument s jasne definovanými lingvistickými tokenmi (slová, vety, ...).

3.1 Extrakcia hlavného textu

Webové dokumenty obsahujú okrem primárneho obsahu aj prvky, tzv. *boilerplate*, ktoré do spracovávaného textu nepatria. Vo webových dokumentoch sa stretneme prevažne so štruktúrou stránky pozostávajúcej z hlavičky, menu, pätičky a iných pre spracovanie textu nepodstatných informácií. Webový dokument môže tiež obsahovať vložené bloky s reklamou alebo iným obsahom, ktoré primárne nepatria k hlavnému obsahu, hoci sa v ňom priamo nachádzajú. Hlavným obsahom je v kontexte mojej práce myslený text, ktorý obsahuje primárnu informáciu webového dokumentu. Hlavný obsah má vo väčšine prípadov formu článku. Extrahovanie môže prebiehať manuálne alebo automaticky, prípadne poloautomaticky. Manuálne extrahovanie je síce najspoľahlivejšia ale zároveň aj veľmi nepraktická metóda. Publikácia *A brief survey of web data extraction tools* [9] uvádza nástroje nazývané *wrappery*, ktoré umožňujú čiastočnú automatizáciu procesu extrakcie hlavného textu. Sú to nástroje, ktoré pristupujú k problému extrakcie obsahu rôznym spôsobom. Popis rôznych prístupov k tomuto problému je možné nájsť nižšie:

- **jazyky pre vývoj wrapperov** – Prvá iniciatíva pre riešenie problému extrakcie dát. Jazyky sú alternatívou k univerzálnym jazykom, akými sú napríklad Java a Python.
- **nástroje so znalosťou o HTML štruktúre** – Nástroje najprv prevedú HTML dokument na DOM[1]. Následne sú naň aplikované polo-automaticky alebo automaticky generované pravidlá. Ich nevýhodou je, že štruktúra DOM je často odlišná od štruktúry samotného dokumentu, tak ako ju vníma človek.
- **nástroje založené na NLP** – Nástroje využívajú techniky bežné pri spracovaní prirodzeného jazyka, na základe ktorých budujú vzťahy medzi lexikálnymi tokenmi. Zo vzťahov sú potom odvodené pravidlá na extrakciu relevantných častí dokumentu.

[1] DOM - Document Object Model, objektový model dokumentu

- **induktívne nástroje** – Extrahujú relevantné časti textu na základe oddeľovačov a značiek v texte (napr. HTML tagy). Pravidlá sú generované na základe formátovania dokumentu. Tento typ je bez použitia ďalších heuristík pre úlohu získavania hlavného textu z HTML dokumentu nevhodný.
- **nástroje založené na modelovaní** – Nástroje sa snažia nájsť časti dokumentu, ktoré prislúchajú preddefinovanej štruktúre objektov záujmu. Štruktúra objektov je modelovaná ako dátové primitíva (zoznam, n-tica, ...).
- **nástroje založené na ontológii** – Nespoliehajú sa na formátovanie a štruktúru extrahovaných dát, ale na dáta samotné. Pri poskytnutí špecifickej doménovej aplikácie môže byť ontológia použitá pre nájdenie konštánt na stránke a následné vytvorenie objektov z konštánt.

V publikácii *A brief survey of web data extraction tools* [9] sa uvádza mnoho implementácií týchto nástrojov a každým dňom sa viacej zvyšuje ich automatizácia. Mojim cieľom je vyhľadať vo webovom dokumente hlavný text, čo predstavuje len malú časť schopností, ktoré tieto nástroje dokážu. Pre jazyk Python som našiel tri vhodné projekty, ktoré umožňujú nájsť hlavné časti textu v HTML dokumente a zároveň sú dostatočne všeobecné na to, aby si poradili s množstvom rozličných štruktúr, ktoré sa vyskytujú na webových stránkach.

Posledným krokom extrahovania hlavného textu z dokumentu je odstránenie neužitočných HTML elementov z extrahovaného textu (napr. `<style>`, `<script>` a pod.). S odstránením objektov by nemal mať problém žiaden HTML parser, ktorý budem pri práci používať. Implementácia nižšie predstavenej platformy *Readability* dokonca vo výslednom dokumente už tieto elementy neobsahuje.

3.1.1 Platforma Readability

Formou wrapperu, ktorá využíva niekoľko prístupov, je v súčasnosti platforma Readability. Spája vlastnosti HTML-aware, induktívnych nástrojov a nástrojov založených na modelovaní. Spolieha sa na konvencie bežne používané pri tvorení webových dokumentov a takisto sémantických značiek, ak sú prítomné. Platforma *Readability* bola pôvodne len experimentom, ktorý mal ľuďom spríjemniť čítanie dlhších textov na webových stránkach. No po jej rozšírení vzniklo mnoho implementácií v rôznych jazykoch a bola integrovaná do veľkého množstva aplikácií. Pre jazyk Python existuje niekoľko funkčných implementácií, ktoré sa viac či menej inšpirujú referenčnou implementáciou v jazyku JavaScript.

Tento algoritmus okrem vyhľadania hlavného textu robí aj niekoľko užitočných transformácií. Identifikuje rôzne typy častí pôvodného textu a jednotne ich označí. Vo výslednom dokumente je potom veľmi ľahké identifikovať napríklad časti textu označujúce citácie, výpisy zdrojového kódu a iné.

3.1.2 Knižnica Body Text Extraction

Nástroj Body Text Extraction je založený na identifikácii hlavného textu webovej stránky pomocou hustoty textu a HTML tagov. Predpokladom k fungovaniu algoritmu je to, že hlavný text webového dokumentu obsahuje minimálne množstvo formátovacích značiek. Výstupom algoritmu je najdlhší blok textu nachádzajúci sa vo webovom dokumente. Z tejto skutočnosti vyplýva i najväčšia nevýhoda nástroja, ktorou je neschopnosť extrahovať text, ktorý v sebe obsahuje vložené bloky s reklamou alebo iným nežiadúcim obsahom.

3.1.3 Knižnica jusText

Nástroj jusText bol vyvinutý ako nástroj pre odstránenie *boilerplate* elementov vo webovom dokumente. Po odstránení je možné dokument zaradiť napríklad do textového korpusu pre potreby spracovania prirodzeného jazyka. Nástroj môže byť využitý aj na získanie hlavného textu z dokumentu, pretože po odstránení zbytočných objektov zostávajú na stránke len objekty, ktoré tvoria užitočný text. Projekt bol vytvorený v Natural Language Processing Centre na Masarykovej univerzite v Brne.

Algoritmus je založený na segmentácii. Iteratívne vyhľadáva bloky HTML dokumentu, ktoré majú nežiadúce charakteristiky, a rozdeľuje ich do skupín. Bloky v jednotlivých skupinách následne označí za vyhovujúce alebo nevyhovujúce (*boilerplate*) na základe ich okolitých elementov. Výsledkom sú bloky, ktoré obsahujú len užitočný textový obsah.

3.2 Syntaktická analýza textu

Ďalším krokom je vytvoriť zo získaného textového dokumentu reprezentáciu vhodnú na lingvistickú analýzu. Táto fáza predspracovania spočíva v rozdelení textu na odseky, vety a slová. Hoci je jazyk HTML určený pre prezentačné účely a nie je plnohodnotným sémantickým jazykom, môže delenie textu do značnej miery uľahčiť. Ak je dokument správne sémanticky označený, môžeme pomocou značiek rozpoznať užitočné časti textu:

- `<p>` – odsek (odstavec)
- `<h#>` – nadpis
- `<strong>` – fráza s dôrazom
- `<em>` – zvýraznená fráza

Tokeny ako veta či slovo bude nutné rozdeliť použitím metódy tokenizácie pracujúcej s neformátovaným textom. Pre jazyk Python existuje knižnica *nltk* [1], ktorá je na tieto účely vhodným kandidátom.

3.3 Identifikácia sémanticky podobných slov

Český jazyk má bohaté tvaroslovie, ktoré robí pri analýze dokumentu problémy. Pri sumarizácii budeme potrebovať označiť slová vyjadrujúce rovnaký objekt alebo jav, hoci textová reprezentácia slova je odlišná (skloňovanie, časovanie, ...). Pre riešenie tohto druhu problému je možné využiť rôzne prístupy. Žiaden nižšie uvedený prístup však nerieši *mnohoznačnosť slov* ani problém *synonymie*.

3.3.1 Stemming

Ide o proces algoritmického odstránenia nadbytočných prípon alebo predpôn v slovách. Nástroj, ktorým sa odstraňujú tvaroslovné prípony, sa nazýva *stemmer* a proces odstraňovania prípon sa nazýva *stemming*. Proces *stemmingu* so sebou prináša dva hlavné problémy v podobe **over-stemmingu** a **under-stemmingu**. Problém *over-stemmingu* spočíva v odstránení príliš dlhej prípony zo slova, ktoré zapríčiní, že dve slová s rozdielnym významom budú považované za rovnaké. Naopak problém *under-stemmingu* znamená odstránenie príliš krátkej časti prípony zo slova, ktoré spôsobí, že slová významovo patriace k sebe budú

považované za odlišné. Pri spracovaní textu sa s týmito problémami ráta, pretože sa úplne odstrániť nedajú. Pre reálne používané stemmery je dôležité hlavne to, aby sa slová s rovnakým koreňom transformovali na rovnaký tvar aj za cenu *over-stemmingu*. To je i dôvod, prečo sa v praxi používajú skôr agresívnejšie formy stemmerov.

Angličtina a podobné jazyky majú oveľa jednoduchšie tvorenie slov ako čeština, prípadne slovenčina. Je teda aj oveľa jednoduchšie vytvoriť stemmer pre tento jazyk ako pre vysoko flektívne jazyky. Najznámejší algoritmus pre stemming anglických slov sa nazýva *Porterov algoritmus*. Je to jednoduchá sada pravidiel, cez ktoré slovo prejde a upraví sa na požadovaný tvar. Hoc sa na český jazyk dá použiť podobný algoritmus s modifikovanými pravidlami, proces algoritmického stemmingu dosahuje oveľa horších výsledkov ako *Porterov stemmer*.

Porter navrhol a implementoval jazyk Snowball špeciálne určený pre potreby stemmingu. V jazyku Snowball existuje stemmer aj pre český jazyk. No pre jazyk Python existuje stemmer, ktorý dosahuje približne rovnaké výsledky, ale je prenositeľnejší a je podstatne jednoduchšie ho použiť. Ak sa neukáže, že implementácia v jazyku Python obsahuje nejaké zásadné nedostatky, v mojej implementácii ju uprednostním.

3.3.2 Prevod na slovníkový tvar

Ďalším prístupom k zjednoteniu tvarov slov, nazývaný *lematizácia*, je prevod slov na slovníkový tvar. Je to veľmi účinný a pri dostatočnej veľkosti slovníka aj spoľahlivý spôsob prevodu slov na jednotný tvar. Pre slová, ktoré sa v slovníku nenachádzajú, je možné použiť algoritmický stemmer. Najväčším problémom je potreba budovania dostatočne veľkého a kvalitného slovníka pre daný jazyk. Ďalším problémom je vysoká časová náročnosť hľadania slov v slovníku, takže je nutné myslieť aj na efektívnu implementáciu vyhľadávacích algoritmov.

Keďže nemám k dispozícii žiaden vyhovujúci slovník pre český jazyk, tak je v mojej implementácii použitá metóda algoritmického stemmingu. Výskumná skupina NLP na Fakulte informatiky VUT v Brne disponuje knižnicou *libma* a dostatočne veľkým slovníkom. Knižnica je však napísaná v jazyku C++, poskytnutá mi bola iba binárna verzia knižnice a wrapper do jazyka Python sa mi nepodarilo uviesť do činnosti.

3.3.3 Levenshteinova vzdialenosť

Existuje aj iný prístup ako určiť slová s podobným významom. Slová sa neprevádzajú na spoločný tvar, ale určuje sa ich podobnosť. Takýmto prístupom je aj *Levenshteinova editačná vzdialenosť*. Vyjadruje sa ako minimálny počet jednoznakových zmien potrebných k prevodu z prvého porovnávaného slova na druhé porovnávané slovo. Povolenými zmenami sú vloženie, zmazanie a náhrada znaku. Za podobné sú označené slová s počtom operácií menším ako prahová hodnota. Výhodou tohto prístupu je jeho nezávislosť na použitom jazyku. No z toho plynie aj hlavná nevýhoda tejto metódy. Pri zmenách sa neberú do úvahy spôsoby tvorby zmien, a tak sa náhrady vykonávajú aj v koreni slov a môžu sa ako podobné označiť aj slová s úplne odlišným sémantickým významom. Tento prístup sa používa veľmi vzácne.

3.4 Nevýznamné slová

Prirodzený jazyk je význačný tým, že obsahuje tzv. *stop slová*, ktoré nemajú sémantický význam, alebo lepšie povedané nie sú nositeľom informačnej hodnoty. Medzi tieto slová patria slová, pri ktorých sa predpokladá vysoký výskyt vo všetkých dokumentoch zo skúmanej kolekcie. Bežne sa medzi slovami vyskytujú spojky, predložky a podobne. Napriek tomu, že sa v textoch často vyskytujú významným počtom, čitateľ sa vďaka nim nedozvie žiadnu novú informáciu.

Stop slová sa líšia pre každý jazyk ale i pre skupinu ľudí či oblasť záujmu. Zoznam *stop slov* bude teda odlišný pre ľudí z akademickej a ekonomickej skupiny, i keď sa môže jednať o texty podobného formátu napísané v rovnakom prirodzenom jazyku. Pre každý jazyk však možno vytvoriť základný zoznam *stop slov*, ktorý sa potom doplní ďalšími slovami, ktoré v rámci riešenia konkrétnej úlohy plnia úlohu konkrétnejších *stop slov*.

Pri sumarizácii bude potrebné zostaviť zoznam *stop slov*, ktoré sa pri analýze dôležitosti slov nebudú brať do úvahy. Luhn vo svojej heuristike v článku *The automatic creation of literature abstracts* [10] priamo počíta s tým, že bude zoznam *stop slov* známy, pretože na jeho základe počíta hodnotu významnosti extrahovaných viet. Zoznam by mal obsahovať len slová, ktoré sú považované za *stop slová* bez ohľadu na typ dokumentov, aby mohol byť výsledný sumarizátor použitý pre čo najširšiu škálu dokumentov.

Kapitola 4

Sumarizácia

Pod pojmom sumarizácia budem v tejto práci rozumieť automatizovaný proces vytvárania súhrnu. S pojmom súhrn sa väčšina ľudí už stretla a intuitívne mu rozumie. V tejto práci budem tento pojem chápať rovnako ako v publikácii *Introduction to the special issue on summarization* [13], a teda ako text vytvorený z jedného alebo viacerých dokumentov, ktorý pokrýva hlavné myšlienky pôvodného textu a nie je dlhší ako polovica pôvodného textu. Na rozdiel od autorov publikácie *Introduction to the special issue on summarization* [13] nebudem uvažovať inú ako písanú formu dokumentu. Budem teda, podobne ako Das a Martins [2], predpokladať nasledujúce tri vlastnosti výsledného súhrnu:

- Súhrn môže byť vytvorený z jedného ale i viacerých textových dokumentov.
- Súhrn musí zachovávať hlavné myšlienky pôvodných dokumentov.
- Súhrn je podstatne kratší ako pôvodný rozsah dokumentov. Podľa typu súhrnu je dĺžka 3–25 % pôvodného textu.

Pri sumarizácii sa vždy pracuje s určitými entitami, ktoré tvoria výsledný súhrn. Tieto, nie vždy rovnaké, druhy entít sa pri sumarizácii vyberajú z textu pôvodného dokumentu a skladajú do podoby súhrnu. V závislosti od dĺžky pôvodného dokumentu sa môže jednať napríklad o slovné spojenia, vety ale i celé odseky. V ďalšom texte budem rozprávať o týchto entitách ako o **vetách**.

4.1 Vývoj metód

Prvé pokusy o sumarizáciu spísal vo svojej práci *The automatic creation of literature abstracts* [10] Luhn. Predpokladal, že vhodnými vetami do súhrnu sú tie s najviac frekventovanými frázami. Ďalšia významná práca patrila Edmunsonovi, ktorý vo svojej práci *New Methods in Automatic Extracting* [3] predstavil ďalšie tri prístupy k hodnoteniu viet v textových dokumentoch. Obe predchádzajúce práce popisujú metódy, ktoré sa dajú zaradiť medzi *heuristické*. Iným prístupom sú metódy *štatistické*, akou je napríklad metóda autorov Kupieca, Pedersena a Chena založená na Naivnej Bayesovskej klasifikácii, ktorú zverejnili v práci *A trainable document summarizer* [8], alebo neskôr sa rozvíjajúce metódy z oblasti *soft-computingu*, ktoré napríklad uvádzajú vo svojej práci [6] autori Steinberger a Ježek.

4.2 Typy sumarizácií

Podľa rozsahu kolekcie pôvodných dokumentov rozoznávame rozdelenie na jedno-dokumentovú a viac-dokumentovú sumarizáciu. Počet pôvodných dokumentov by nemal mať na výsledný súhrn žiaden vplyv, hoc metódy pre jedno a viac-dokumentovú sumarizáciu sa líšia. Podľa Dasa a Martinsa [2] sa sumarizácie bez ohľadu na počet pôvodných dokumentov delia formou nasledovne:

- **extrakt** – Identifikácia dôležitých častí (viet, odsekov) pôvodného dokumentu, a ich následné umiestnenie do súhrnu v nezmenenej forme. Vybrané vety často nemusia nadväzovať a môžu byť vytrhnuté z kontextu. Častým javom je prehliadanie *anaforických vzťahov*[1].
- **abstrakt** – Vyjadrenie hlavných myšlienok alebo tém pôvodného textu a vytvorenie nového originálneho obsahu.
- **fúzia** – Kombinácia dôležitých extrahovaných častí tak aby pôsobili súvisle. Fúzia je vlastne extrakt, v ktorom sa snažíme o súvislosť výsledného textu.
- **kompresia** – Odstránenie nedôležitých častí pôvodného textu (odsekov, viet, častí zložených viet, ...).

V dnešnej dobe je tvorba súhrnu založená prevažne na extrahovaní alebo kompresii pôvodného dokumentu. Tvorba plnohodnotného abstraktu zatiaľ stále zostáva doménou ľudí. Kvalitné automatizované vypracovanie abstraktu vyžaduje pokročilé metódy z oblasti spracovania prirodzeného jazyka. Problémom je sémantická analýza pôvodného textu a generovanie spisovného textu v prirodzenom jazyku.

V práci *Sumarizace textů* [6] sú spomenuté ďalšie hľadiská podľa ktorých je možné sumarizácie deliť:

- podľa úrovne spracovania súhrnu
  - **povrchné prístupy** – Informácie sú reprezentované povrchnými vlastnosťami. Povrchné vlastnosti sú napr. pozícia viet či frekvencia slov. Výsledkom je extrakt.
  - **hlbšie prístupy** – K určeniu významných častí používajú sémantické vlastnosti. Výstupom môže byť extrakt ale i abstrakt.
- podľa účelu vytvárania súhrnu
  - **hodnotiaci** – Vyjadrujú názor autora súhrnu o danom dokumente. V súčasnej dobe nie je možné tieto súhrny automaticky generovať.
  - **indikatívny** – Zhrnujú hlavné myšlienky pôvodného dokumentu. Ich úlohou je podať čitateľovi informáciu o vhodnosti čítania celého textu.
  - **informatívny** – Podobne ako indikatívna forma zhrňuje hlavné myšlienky pôvodného textu. Býva však dlhší a podáva aj niektoré detaily. Po prečítaní by mal byť čitateľ zoznámený s témou pôvodného dokumentu a nie je nutné aby ho prečítal celý.

[1] anaforický vzťah - odkazovanie sa (najčastejšie zámenom) na nejaký objekt v danom kontexte

- podľa užívateľov
  - **všeobecné** – V súhrne sú obsiahnuté všetky témy pôvodného textu. Je určený širokému okruhu čitateľov.
  - **opytovacie** – Súhrn je vytvorený s ohľadom na konkrétnu otázku, respektíve kľúčové slová.
  - **tematicky zamerané** – Súhrn obsahujúci len určitú tému.
  - **aktualizačné** – Zohľadňuje znalosti čitateľa a podáva mu len pre neho nové informácie.
  - **užívateľsky zamerané** – Obsahujú len témy, o ktoré majú čitatelia záujem.
- podľa použitej metódy
  - **heuristické** – Využívajú jednoduchú heuristiku na určenie dôležitosti viet.
  - **štatistické** – Využíva štatistické znalosti medzi abstraktami a textami v trénovacej množine dvojíc súhrn–plný text.
  - **grafové** – Inšpirované algoritmami z prostredia webu ako napríklad HITS[2] alebo *PageRank*.
  - **algebraické** – Napríklad metóda LSA[3].

4.3 Hodnotenie vytvoreného súhrnu

Ohodnotenie výsledného súhrnu je veľmi obtiažne. Najväčší problém pri ohodnotení je to, že neexistuje objektívne najlepší súhrn k danému dokumentu. Aby bolo napriek tomu možné porovnávať výsledky sumarizácií s inými sumarizátormi či ručne vytvorenými abstraktami, vznikli metódy na hodnotenie súhrnov. Steinberger a Ježek [15] uvádzajú niekoľko kritérií podľa ktorých je možné posudzovať kvalitu súhrnu. A to napríklad podľa:

- **sémantickej informovanosti** (angl. *content evaluation*) – Miera možnosti zrekonštruovať zo súhrnu pôvodný text. Hodnotí sa schopnosť súhrnu zachytiť hlavné myšlienky pôvodného textu.
- **množstva redundancie** (angl. *non-redundancy*) – Súhrn by nemal obsahovať redundantné informácie (hlavne nie redundantné vety).
- **gramatickej správnosti** (angl. *grammatical correctness*) – Text by nemal obsahovať netextový obsah akým sú citačné a rôzne iné značky, chybné slová alebo chyby interpunkcie.
- **súvislosti textu** (angl. *text quality evaluation*) – Miera s akou na seba nadväzujú jednotlivé časti súhrnu a vytvárajú integrovaný výsledný text. Vo výslednom súhrne sa hodnotí čitateľnosť a logická súvislosť súhrnu. Tu patrí aj požiadavka aby zámená ukazovali na objekt v kontexte výsledného súhrnu.
- **kompresného pomeru** – Podiel dĺžky súhrnu a dĺžky originálu.

[2] HITS - Hyperlink-Induced Topic Search, algoritmus je známy aj ako *Hubs and Authorities*
[3] LSA - Latent Semantic Analysis, latentná sémantická analýza

Základnými *vnútornými* (angl. *intrinsic*) metódami používanými pri hodnotení súhrnov sú *presnosť* $0 \leq P(E) \leq 1$ (angl. *precision*) a *úplnosť* $0 \leq R(E) \leq 1$ (angl. *recall*). Nevýhodou metód je, že sa dajú použiť len pre extrakty. Výhodou je, že sa ohodnotenie pomocou nich dá plne automatizovať. Myšlienka spočíva v porovnávaní extrahovaných viet oproti pôvodnému textu alebo ručne vytvorenému extraktu. Koeficienty hodnotenia sa vypočítajú podľa vzťahov 4.1 a 4.2, kde E predstavuje vyhodnocovaný extrakt, A je počet zhodných viet vo vyhodnocovanom a referenčnom extrakte, B je počet viet, ktoré sú vo vyhodnocovanom extrakte ale nie sú v referenčnom extrakte a C je počet viet, ktoré sú v referenčnom ale nie sú vo vyhodnocovanom extrakte.

$$P(E) = \frac{A}{A+B} \tag{4.1}$$

$$R(E) = \frac{A}{A+C} \tag{4.2}$$

Výsledky predchádzajúcich metód neposkytujú dostatočnú informáciu o kvalite súhrnov. Pri použití metriky *precision* je postačujúce, aby sumarizátor vybral jedinú vetu, ktorá sa zároveň vyskytuje v referenčnom extrakte, a dosiahne najvyššie možné hodnotenie $P(E) = \frac{1}{1+0} = 1$. Podobná situácia nastáva pre metriku *recall*, kde sumarizátoru dokonca stačí určiť ako dôležité všetky vety pôvodného dokumentu pre dosiahnutie najvyššieho možného hodnotenia $R(E) = \frac{n}{n+0} = 1$. Riešením týchto neduhov je metrika *F-score*, ktorá je kompozíciou metrík *precision* a *recall*. Jej základná varianta je vypočítaná ako vážený priemer týchto dvoch metrík (pozri vzťah 4.3). Zložitejšia varianta metriky *F-score*, vyjadrená vzťahom 4.4, uvažuje váhový faktor w, ktorý zvýhodňuje metriku *precision* ak $w > 1$ alebo metriku *recall* ak $w < 1$. Pri hodnote $w = 1$ je metrika zhodná s jej základnou verziou, určenou váženým priemerom metrík *precision* a *recall*.

$$F(E) = \frac{2 \cdot P(E) \cdot R(E)}{P(E) + R(E)} \tag{4.3}$$

$$F(E) = \frac{(w^2+1) \cdot P(E) \cdot R(E)}{w^2 \cdot P(E) + R(E)} \tag{4.4}$$

Steinberger a Ježek [15] tiež uvádzajú štatistickú mieru, ktorej výhodou je, že vylučuje náhodnú zhodu (úroveň kvality dosiahnutá náhodným výberom viet). Táto metrika je pomenovaná *Kappa*. Metrika *Kappa* taktiež umožňuje určenie pravdepodobnosti zhody medzi sudcami, čo je jej druhou výhodou. Koeficient *Kappa* je definovaný výrazom 4.5, kde $P(S)$ je pravdepodobnosť zhody medzi systémom (respektíve sudcom) a referenčným extraktom (respektíve iným sudcom) a $P(N)$ je pravdepodobnosť náhodnej zhody. Koeficient K je rovný nule ak je zhoda sumarizačného systému rovnaká s náhodnou zhodou, a $K = 1$ ak je hodnotený extrakt zhodný s referenčným. Ak je zhoda horšia ako sa očakáva od náhodného sumarizačného systému, tak môže byť koeficient K i záporný. Za vyhovujúcu hodnotu koeficientu K sa považuje číslo z rozsahu $\left\langle \frac{2}{3}; 1 \right\rangle$.

$$K = \frac{P(S) - P(N)}{1 - P(N)} \tag{4.5}$$

Vyššie popísané ko-selekčné prístupy používajú k hodnoteniu vety z referenčného extraktu. Ako som písať už skôr, tak referenčný extrakt nemusí byť nutne jediným vyhovujúcim súhrnom. Predpokladajme, že spracovávaný dokument má každú informáciu vyjadrenú

niekoľkými vetami, a jedna z nich sa objaví v referenčnom súhrne. Potom ak sumarizátor nevyberie presne túto vetu, prejaví sa to negatívne na hodnotení súhrnu, ktorý bol týmto sumarizátorom vytvorený. Sumarizátor mohol ale vybrať do súhrnu inú, nemenej vhodnú, vetu obsahujúcu tú istú informáciu. Steinberger a Ježek [16] popisujú takúto situáciu na jednoduchom príklade. Predpokladajme, že manuálny extrakt obsahuje vety [1, 2] pôvodného dokumentu. Predpokladajme tiež, že dva sumarizačné systémy A a B vybrali do extraktu porade vety [1, 2] a [1, 3]. Súhrn sumarizačného systému A bude hodnotený lepšie ako súhrn systému B, i keď je možné, že vety 2 a 3 sú rovnako dôležité a súhrny sumarizačných systémov by teda mali byť ohodnotené približne rovnako. Aby sa podobným nepresnostiam pri hodnotení súhrnov vyhlo, zaviedol sa systém hodnotenia nazývaný *relatívny prínos* (angl. *relative utility*). Sudcovia pri tomto spôsobe hodnotenia nemajú za úlohu vytvárať referenčný extrakt, ale hodnotia každú vetu dokumentu číslom v určitom rozsahu. Číslo udáva vhodnosť výberu danej vety do výsledného extraktu. Steinberger a Ježek [16] uvádzajú ako príklad dokument pozostávajúci z piatich viet [1/5, 2/4, 3/4, 4/1, 5/2]. Hodnota za znakom lomeno vyjadruje prínos (angl. *utility*) každej vety, ktorý udáva stupeň vhodnosti zaradenia danej vety do výsledného extraktu. Pri tomto spôsobe hodnotenia získajú dva súhrny [1, 2] a [1, 3] vytvorené rozdielnymi sumarizačnými systémami rovnaké skóre $5 + 4 = 9$, pretože oba tvoria podľa hodnotenia sudcu optimálny extrakt. Pre získanie hodnoty *relatívneho prínosu* je nutné požiadať $N > 0$ sudcov o priradenie prínosu všetkým n vetám dokumentu. Potom e najlepšie hodnotených viet nazveme extrakt viet veľkosti e. Výsledný *relatívny prínos* je ohodnotenie vypočítané pomocou výrazu 4.6, kde u_{ij} je prínos vety j získaný od sudcu i. Parameter ϵ_j je rovný 1 pre najlepšie ohodnotených e viet s ohľadom na súčet prínosov všetkých sudcov. V opačnom prípade je $\epsilon_j = 0$. Podobne $\delta_j = 1$ pre e najlepšie ohodnotených viet extrahovaných sumarizačným systémom, alebo $\delta_j = 0$ v opačnom prípade.

$$RU = \frac{\sum_{j=1}^{n} \delta_j \sum_{i=1}^{N} u_{ij}}{\sum_{j=1}^{n} \epsilon_j \sum_{i=1}^{N} u_{ij}} \tag{4.6}$$

Nevýhody vyššie uvedených metód sa snažia riešiť metódy založené na *podobnosti obsahu* (angl. *content-based*). Tie umožňujú porovnávať súhrn aj s ručne vytvoreným abstraktom, ktorý je často pre dokumenty k dispozícii. Ďalšou významnou odlišnosťou od ko-selekčných prístupov je to, že sa pri hodnotení môžu používať ľubovoľné *príznaky* (angl. *feature*), pričom pravdepodobne najčastejšie sú používané slová. Najznámejšou metrikou merania podobnosti vo vektorovom modeli je *kosínusová vzdialenosť* (angl. *cosine similarity*), ktorá je vyjadrená vzorcom 4.7, kde $\vec{x}$ a $\vec{y}$ sú vektory váh referenčného a hodnoteného súhrnu.

$$\cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| \|\vec{y}\|} = \frac{\sum x_i y_i}{\sqrt{\sum x_i^2} \sqrt{\sum y_i^2}} \tag{4.7}$$

Odlišnou metrikou uvedenou v publikácii *Hodnocení kvality sumarizátorů textů* [15] je *prekrytie obsahu* (angl. *unit overlap*). Ohodnotenie sa počíta podľa vzťahu 4.8, kde X a Y sú množinové reprezentácie textu. Ako jednotky reprezentácie textu sa používajú množiny slov.

$$\text{overlap}(X, Y) = \frac{|X \cap Y|}{|X| + |Y| - |X \cap Y|} \tag{4.8}$$

V publikácii *Hodnocení kvality sumarizátorů textů* [15] je predstavená ešte metrika nazvaná *najdlhšia subsekvencia* (angl. *longest common subsequence*). Metrika je vyjadrená formulou 4.9, kde length(X) je počet slov reťazca X a edit$_{di}(X, Y)$ je minimálny počet operácií vloženia a vymazania slov potrebný k transformácii X na Y.

$$\text{lcs}(X, Y) = \frac{\text{length}(X) + \text{length}(Y) - \text{edit}_{di}(X, Y)}{2} \tag{4.9}$$

Pomerne známou a často používanou, napríklad na medzinárodných súťažiach, je evaluačná metóda *Recall-Oriented Understudy for Gisting Evaluation* (ďalej len ROUGE). Ide vlastne o súbor rôznych metód používaných k hodnoteniu súhrnov, ktoré sú založené na podobnosti n-gramov. Ako n-gram sa pri ohodnotení touto metódou považuje postupnosť za sebou idúcich termov (najčastejšie slov). Predpokladajme, rovnako ako Steinberger a Ježek [16], množinu referenčných súhrnov RSS vytvorenú sudcami. Ohodnotenie súhrnu metódou ROUGE-n je potom dané vzťahom 4.10, kde Count$_{match}(gram_n)$ je maximálny počet n-gramov vyskytujúcich sa v referenčnom i hodnotenom súhrne a Count$(gram_n)$ je počet n-gramov v referenčnom súhrne. Čitateľ výrazu 4.10 udáva počet n-gramov nachádzajúci sa v hodnotenom súhrne i všetkých referenčných súhrnoch, zatiaľ čo menovateľ udáva počet n-gramov v referenčných súhrnoch. Metrika ROUGE-n je metrikou *úplnosti*. Okrem metriky ROUGE-n existuje niekoľko ďalších metrík. Metrika ROUGE-S (S ako *Skip bigram*) je benevolentnejšia k postupnostiam n-gramov, pretože dovoľuje ľubovoľný odstup medzi dvoma termami. Hodnota tejto metriky vlastne udáva nakoľko sú slová v texte správne zoradené. Metrika ROUGE-L (L ako *Longest common subsequence*) je založená na najdlhšej spoločnej postupnosti slov v referenčnom a hodnotenom súhrne. Metrika ROUGE-W (W ako *Weighted longest common subsequence*) je upravená verzia metriky ROUGE-L, ktorá v prípade zhody postupností n-gramov preferuje tie, ktoré sú neprerušené.

$$\text{ROUGE-}n = \frac{\sum_{C \in RSS} \sum_{gram_n \in C} \text{Count}_{match}(gram_n)}{\sum_{C \in RSS} \sum_{gram_n \in C} \text{Count}(gram_n)} \tag{4.10}$$

Najväčšia pozornosť je v práci *Hodnocení kvality sumarizátorů textů* [15] venovaná novo predstavenej metóde hodnotenia kvality súhrnov založenej na *singulárnej dekompozícii*. Ide o spôsob hodnotenia založený na *podobnosti obsahu*, takže je možné automaticky vytvorené súhrny porovnávať i s ručne vytvorenými abstraktami obsahujúcimi vety, ktoré sa nevyskytujú v pôvodnom dokumente. Hlavná myšlienka tkvie v porovnávaní jedného alebo viacerých hlavných tém sumarizovaného dokumentu metódou LSA. Viacej informácií o metóde LSA je možné nájsť v sekcii 5.3. Výsledky experimentov, vykonané autormi tejto práce, ukázali, že nová metóda je schopná rozlíšiť kvalitné extrakty od náhodných lepšie ako *kosínusová vzdialenosť*. Nevýhodou novo predstavenej metódy je závislosť na kvalitnom zozname *stop slov* a lematizačnom procese.

Úplne odlišným spôsobom pristupujú k hodnoteniu kvality sumarizácií *vonkajšie* (angl. *extrinsic*) metódy, medzi ktoré sa radia aplikačne založené metódy. Hodnotenie spočíva vo vyhodnocovaní sumarizácie pre nejakú praktickú úlohu. Z príkladov, ktoré spomínajú Steinberger a Ježek [15] je to napríklad *korelácia relevancie*, kde sa nahradí pôvodný text súhrnom, a potom sa zisťuje korelácia medzi originálnym dokumentom a súhrnom pre úlohy kategorizácie či vyhľadávania.

Relatívne najkvalitnejšou metódou hodnotenia stále zostáva manuálna kontrola a hodnotenie ľudskými recenzentmi (tzv. sudcami). No nevýhodou je okrem istej miery subjektivity pri hodnotení hlavne časová a finančná náročnosť manuálneho hodnotenia.

Kapitola 5

Sumarizačné metódy

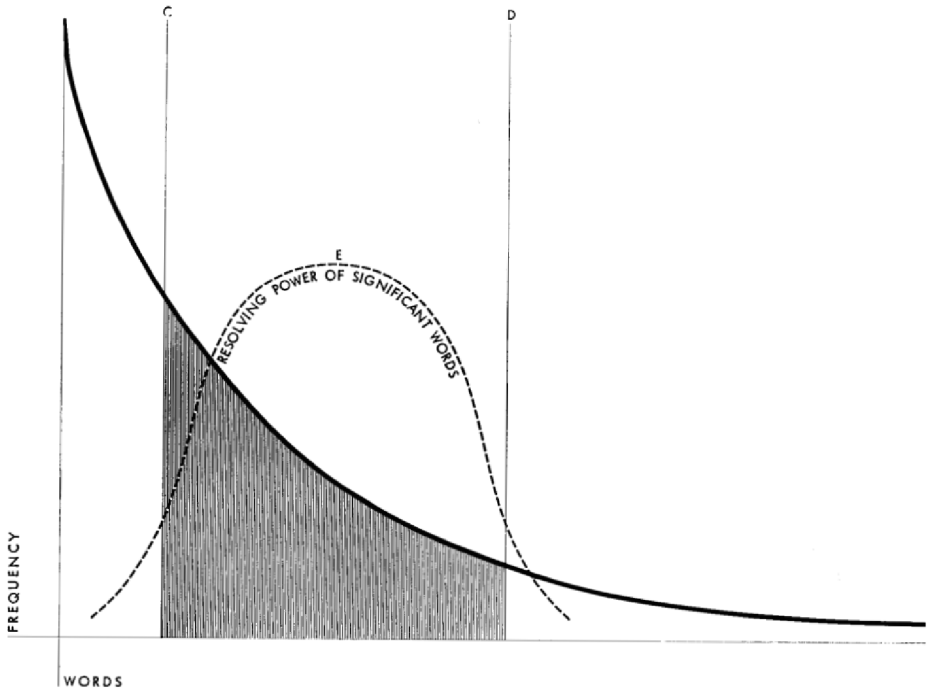
Táto kapitola detailne popisuje tri vybrané sumarizačné metódy. Pri každej metóde je uvedená úvaha o tom nakoľko je možné využiť formát HTML pre zlepšenie kvality vytvoreného súhrnu.

5.1 Luhnova metóda

Metóda sa radí medzi prvé sumarizačné metódy. Je založená na heuristickej analýze sumarizovaného textu. Hlavná myšlienka stavia na predpoklade, že časti textu, ktoré sú vhodnými kandidátmi do súhrnu zároveň obsahujú mnoho významných slov. Významné slová sú tie, ktoré sa v texte často opakujú ale zároveň nepatria medzi *stop slová* spomenuté v sekcii 3.4. Výsledným súhrnom je jednoduchý indikatívny extrakt zložený z viet pôvodného textu. Samotný algoritmus pozostáva z niekoľkých krokov uvedených nižšie:

1. Označ slová vstupného dokumentu obsiahnuté v zozname *stop slov* za nevýznamné.
2. Preveď slová v zozname významných slov na jednotný tvar, napríklad pomocou algoritmického stemmingu.
3. Spočítaj frekvencie jednotlivých slov v zozname významných slov.
4. Rozdeľ slová v súčasnom zozname významných slov na významné a nevýznamné podľa ich frekvencie výskytu vo vstupnom dokumente. Rozdelenie znázorňuje obrázok 5.1.
5. Vety rozdeľ na zhluky, ktoré obsahujú významné slová a zároveň nie sú od seba vzdialené viacej ako 4 nevýznamné slová.
6. Ohodnoť zhluky slov na základe výrazu 5.1, kde $count_{important}$ je počet významných slov v zhluku a $count_{total}$ je počet všetkých slov v zhluku.
7. Ohodnotenie vety je dané najlepšie ohodnoteným zhlukom vo vete.
8. Zoraď vety vzostupne podľa ich ohodnotenia.
9. Vyber prvých n najlepšie ohodnotených viet. Počet viet môže určiť užívateľ ako absolútne číslo. Často sa však pracuje s percentuálnym vyjadrením dĺžky súhrnu vzhľadom k dĺžke pôvodného dokumentu.
10. Usporiadaj vybrané vety podľa ich umiestnenia v pôvodnom dokumente.

$$rating_{luhn} = \frac{count^2_{important}}{count_{total}} \tag{5.1}$$



Obrázek 5.1: Diagram frekvencie slov v dokumente (prevzaté z práce *The automatic creation of literature abstracts* [10])

5.1.1 Príklad

Algoritmus je možné predviesť na nasledujúcom príklade[1]. Podčiarknuté slová sú v texte chápané ako významné. Časti obsiahnuté v hranatých zátvorkách [] predstavujú významné zhluky vo vetách.

> Jednalo se o případ [chlapce v 6. třídě], který měl problémy s učením. Přerostly až v reparát z jazyka na konci školního roku. Nedopadl bohužel dobře a tak musel opakovat 6. [třídu, což se chlapci] ani trochu nelíbilo. Připadal si, že je mezi malými dětmi a realizoval se tím, že si ve [třídě o rok mladších dětí] budoval vedoucí pozici. Dost razantně. Fyzickou převahu měl, takže to nedalo až tak moc práce.

Ako je možné vidieť, text obsahuje tri zhluky. Zhluk **třídě o rok mladších dětí** má hodnotu $\frac{3^2}{5} = 1{,}8$ a zvyšné dva zhluky majú rovnaké ohodnotenie $\frac{2^2}{4} = 1{,}0$. Ak by sme teda vyberali z uvedeného textu napríklad 2 vety, tak uvedený extrakt by vyzeral nasledovne:

> Jednalo se o případ chlapce v 6. třídě, který měl problémy s učením. Připadal si, že je mezi malými dětmi a realizoval se tím, že si ve třídě o rok mladších dětí budoval vedoucí pozici.

[1] Úryvok textu prevzatý z http://www.prevko.cz/dite/skutecne-pribehy-deti

5.1.2 Použitie pre HTML dokument

Pri použití Luhnovej metódy v prostredí webových dokumentov sa ponúka možnosť zvýhodniť aj iné slová ako najfrekventovanejšie. Ak budeme predpokladať správne použitie značiek `<em>`, `<strong>` a podobných, tak je možné identifikovať významné slová. Zhluky vo vetách obsahujúce takéto slová by mali mať vyššie hodnotenie ako významné slová určené len na základe frekvencie výskytu v dokumente, pretože ich významnosť je určená priamo autorom textu. Otázkou ale je do akej miery alebo akým spôsobom by mali byť spomínané slová zvýhodnené pred frekventovanými slovami. Prípadne sa možno zamyslieť nad tým, či by to prinieslo vôbec nejaký úžitok.

5.2 Edmundsonova metóda

Edmundson vo svojej práci [3] popisuje nový druh metód využiteľných pre automatickú sumarizáciu. Jeho práca je založená na experimentoch s vybraným korpusom dokumentov. Dokumenty obsahovali ručne vytvorený abstrakt, na základe ktorého hľadal významné atribúty, ktoré vystihujú reláciu medzi vetami v abstrakte a v pôvodnom dokumente. Iteratívnym vyhodnocovaním ručne vyhotovených a strojovo získaných súhrnov prišiel na nové heuristiky určovania významnosti viet v dokumente, tzv. *clues*. Na rozdiel od predchádzajúcich prác, kde sa kládol dôraz len na prítomnosť vysoko frekventovaných slov, Edmundson zavádza hneď štyri metódy:

- **Cue** – Hľadá sa výskyt tzv. pragmatických slov.
- **Key** – Hľadajú sa kľúčové slová s vysokou frekvenciou výskytu. V podstate sa jedná o modifikovanú verziu Luhnovej metódy.
- **Title** – Hľadá sa výskyt takých slov vo vetách, ktoré sa zároveň vyskytujú v nadpisoch.
- **Location** – Štrukturálne indikátory (umiestnenie extrahovaných termov).

Pre metódy je potrebné vybudovať **slovník význačných slov**, čo je podobne ako *stop slová* pri Luhnovej metóde (sekcia 5.1) na jazyku závislá komponenta. Slovník môže byť vybudovaný na základe vhodného textového korpusu. Skladá sa z podslovníkov pre tri druhy slov uvedených nižšie.

- **bonus slová** (angl. *bonus words*) – Pozitívne relevantné slová, ktoré majú významnú informačnú hodnotu alebo kladú dôraz. Prispievajú k vyššej váhe pre výber vety do súhrnu. Patrí sem 2. a 3. stupeň prídavných mien, príslovky vyjadrujúce úsudok, hodnotné slová v danom dokumente a kauzálne termy.
- **stigma slová** (angl. *stigma words*) – Negatívne relevantné slová, ktoré prispievajú k nižšej váhe pre výber vety do súhrnu. Patria sem slová, ktoré odbočujú od hlavnej témy dokumentu alebo rozoberajú zbytočné detaily.
- **null slová** (angl. *null words*) – Irelevantné slová, ktoré neobsahujú žiadnu informačnú hodnotu pre dokument. Sú to podobné slová ako *stop slová* používané v Luhnovej metóde. Patria sem predložky, spojky, zámená, prídavné mená, atď.

5.2.1 Metóda Cue

Metóda pracuje so slovníkom význačných slov. Najprv porovná zoznam slov so zoznamom *bonus slov* a priradí im váhu $b > 0$, *stigma slovám* váhu $s < 0$ a *null slovám* váhu $n = 0$. Výsledná váha vety je daná sumou váh všetkých slov obsiahnutých vo vete. Následne je proces prakticky rovnaký ako v Luhnovej metóde. Vety sa zoradia podľa váhy a do súhrnu sa vyberie vopred zadaný počet najlepšie ohodnotených viet usporiadaných podľa poriadia v pôvodnom dokumente.

5.2.2 Metóda Key

Metóda pracuje na podobnom princípe ako Luhnova metóda. Je založená na predpoklade, že slová vyskytujúce sa v dokumente s vysokou frekvenciou môžu byť považované za *pozitívne relevantné* (kľúčové slová). Všetky slová sa porovnajú so slovníkom *bonus slov*, čím sa vybuduje slovník dôležitých slov, pre ktoré sa spočíta ich frekvencia v dokumente. Za kľúčové slová sa označia slová s parametrom $\mathrm{TF}(t, d) > w$, kde w je hraničná hodnota zadaná užívateľom. Váha každej vety je tvorená sumou frekvencií slov v nej obsiahnutých. Nakoniec sa rovnako ako v predchádzajúcich metódach vyberú vety s najvyšším ohodnotením.

5.2.3 Metóda Title

Metóda je založená na úvahe, že autor dokumentu tvorí titulok, nadpisy a štruktúru dokumentu tak aby tým vyjadril hlavné myšlienky dokumentu. Nadpisy tak tvoria vlastne krátku sumarizáciu. Hypotéza, že slová z nadpisov sú v dokumente významné, je podľa Edmundsona [3] štatisticky potvrdená na 99 %. Algoritmus ohodnotenia viet pre svoje správne fungovanie vyžaduje **slovník nadpisov**, ktorý obsahuje slová z titulku, nadpisov a vedľajších nadpisov, ktoré sa zároveň nenachádzajú medzi *null slovami*. Slovám v tomto slovníku sa priradia kladné váhy a váha vety je vyjadrená ako suma váh slov v nej obsiahnutých. Edmundson priradil slovám z rôznych typov nadpisov váhy $w_t > w_h > w_s$, kde w_t je váha pre slová z titulku, w_h je váha pre slová z nadpisov a w_s je váha pre slová z vedľajších nadpisov.

5.2.4 Metóda Location

Metóda je založená na predpoklade, že vety pod nadpismi sú pozitívne relevantné. Takisto tematické vety sa často nachádzajú na začiatku alebo na konci dokumentu či odsekov. Táto hypotéza bola experimentálne potvrdená Edmundsonom. Algoritmus ohodnotenia viet najprv vybuduje slovník obsahujúci slová, ktoré sa často vyskytujú v nadpisoch. Všetky slová v dokumente, ktoré sa vyskytujú vo vybudovanom slovníku ohodnotí váhou w_h. Všetky vety v prvom a poslednom odseku ohodnotí váhami w_{p1} a w_{p2}. Vety, ktoré sú umiestnené ako prvé alebo posledné v odseku ohodnotí váhami w_{s1} a w_{s2}. Váha každej vety v dokumente je vyjadrená ako suma váh uvedených vyššie.

5.2.5 Kvalita metód

Podľa experimentov, ktoré Edmundson vykonal dopomohli tieto heuristiky k tvorbe lepších súhrnov ako metódy založené len na frekvencii kľúčových slov. Edmundson vo svojej práci taktiež ukazuje, že najlepšie výsledky sa dosahujú kombináciou vyššie uvedených metód. Pre ohodnotenie viet použil lineárnu kombináciu štyroch navrhovaných metód ktorú vyjadruje výraz 5.2, kde w_1 je váha ohodnotenia C metódy *Cue*, w_2 váha ohodnotenia K metódy

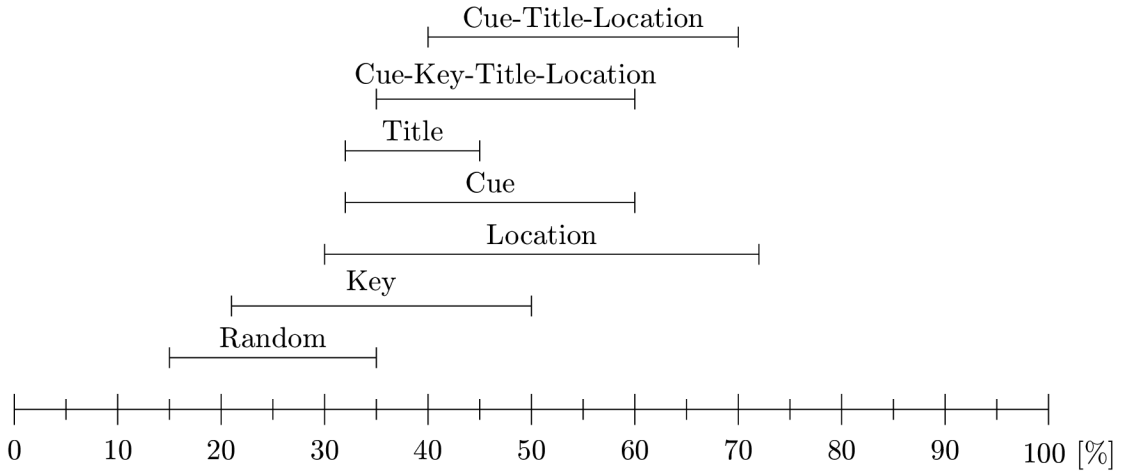
Key, w_3 váha ohodnotenia T metódy *Title* a w_4 váha ohodnotenia L metódy *Location*. Úspešnosť rôznych kombinácií je znázornená na obrázku 5.2 ako rozsahy percentuálnej zhody vygenerovaných súhrnov s referenčným abstraktom. Z obrázku 5.2 je tiež pomerne jasne vidieť, že najlepšie výsledky z testovaných prináša kombinácia pozostávajúca z metód *Cue*, *Title* a *Location*.

$$w_1 C + w_2 K + w_3 T + w_4 L \tag{5.2}$$

5.2.6 Použitie pre HTML dokument

Rovnako ako v prípade Luhnovej metódy 5.1 tak aj pri tejto metóde je možné využiť značkovanie jazyka HTML. Pri predpoklade, že webový dokument obsahuje správne značenia nadpisov a odsekov je možné využiť heuristiku na základe slov z titulkov a nadpisov a heuristiku založenú na štrukturálnych indikátoroch. Pre heuristiku využívajúcu významné slová je potrebné vybudovať slovník s *bonus slovami*. Pri použití v prostredí HTML je možné uvažovať nad variantou kedy slovník týchto slov nahradia práve slová zvýraznené značkami `<em>` prípadne `<strong>`.

Ďalšou heuristikou by mohlo byť sledovanie URL adresy odkazov. Vetu v ktorej je umiestnený hypertextový odkaz mieriaci na inú doménu možno považovať za znak odchýlenia sa od témy dokumentu. Slová obsiahnuté v odkaze by sa teda dali označiť za *stigma slová*, pretože odbiehajú od hlavnej témy dokumentu.



Obrázek 5.2: Porovnanie úspešnosti Edmundsonových algoritmov pre ohodnotenie viet

5.3 Metóda založená na analýze latentnej sémantiky

Text tejto sekcie vychádza prevažne z prác autorov Steinbergera a Ježka [6, 14]. *Latentná sémantická analýza* (ďalej len LSA) je algebraická metóda, ktorá implicitne analyzuje vzťahy medzi slovami či slovnými spojeniami a vetami. V literatúre sa často používa aj termín *Latentné sémantické indexovanie* kvôli pôvodnej aplikácii metódy v súvislosti so spracovaním textu v oblasti *information retrieval*. Prvé zmienky využitia LSA pre potreby sumarizácie možno nájsť v publikácii *Generic text summarization using relevance measure and latent*

semantic analysis [5]. Algoritmus využíva metódu rozkladu matice pomocou *singulárnej dekompozície* (ďalej len SVD[2]). Metóda SVD slúži k redukcii dát. Avšak existujú algoritmy, pri ktorých sa metóda uplatňuje i pri klasifikácii či vyhľadávaní v textových dokumentoch.

5.3.1 Singulárna dekompozícia

Proces algoritmu SVD podľa Steinbergera a Ježka [6] začína vytvorením matice $A = [A_1, A_2, \ldots, A_s]$, kde každý stĺpcový vektor A_i reprezentuje frekvencie slov vo vete i pôvodného dokumentu. Pre dokument obsahujúci m slov a n viet vznikne matica rozmerov $m \times n$. Keďže slov je v pomere k vetám v dokumente mnoho, tak výsledná matica je riedka (angl. *sparse*). Dekompozícia je potom definovaná podľa rovnice 5.3.

$$A = U\Sigma V^T \tag{5.3}$$

Matica $U = [u_{ij}]$ rozmeru $m \times n$ je stĺpcovo ortogonálna a jej stĺpce sa nazývajú ľavé singulárne vektory. Diagonálna matica $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$ rozmeru $n \times n$ obsahuje nezáporné singulárne čísla zoradené zostupne (diagonálne prvky). Ak r je rád matice A, potom platí vzťah 5.4. Nakoniec $V = [v_{ij}]$ je ortogonálna matica rozmeru $n \times n$, ktorej stĺpce sa nazývajú pravé singulárne vektory. Rozmer matíc je redukovaný na k dimenzií, kde $k < n$. Z toho vyplýva, že matice sú redukované nasledovne: U na $m \times k$, Σ na $k \times k$ a V^T na $k \times n$ (obrázok 5.3).

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0 \tag{5.4}$$

$$A = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 1 & 1 & \cdots & 1 \\ 0 & 0 & \cdots & 1 \\ 1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 1 & \cdots & 1 \end{bmatrix} = \overset{U}{\begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1m} \\ u_{21} & & & \\ & & & \\ \vdots & & \ddots & \\ & & & \\ & & & \\ u_{m1} & & & u_{mm} \end{bmatrix}} \overset{\Sigma}{\begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & 0 & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & 0 & \sigma_r \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}} \overset{V^T}{\begin{bmatrix} v_{11} & \cdots & v_{1n} \\ v_{21} & & \\ \vdots & \ddots & \\ & & \\ v_{n1} & & v_{nn} \end{bmatrix}}$$

Obrázek 5.3: Singulárny rozklad matíc s naznačeným redukovaným priestorom

Okrem matematického hľadiska sa môžeme na rozklad matice pozerať ako na rozklad pôvodného dokumentu do k lineárne nezávislých bázových vektorov reprezentujúcich hlavné témy dokumentu. Výsledkom je, že podobná kombinácia slov sa bude vyskytovať pozdĺž rovnakého singulárneho vektoru. Potom matica A mapuje slová do jednotlivých viet a matice U a V mapujú slová resp. vety do k najvýznamnejších tém. Jednou vetou by sa to dalo vyjadriť ako schopnosť sémanticky zhlukovať slová a vety. Sémantické zhlukovanie je veľmi užitočná vlastnosť, pretože metóda dokáže na základe výskytu slov v podobnom kontexte rozpoznať aj slová, ktoré majú syntakticky úplne iný tvar. Takýmito slovami môžu byť synonymá, ale i slová, ktoré spolu tematicky súvisia. Ako príklad uvediem slová hasič, záchranár, požiar a iné.

[2]SVD - Singular Value Decomposition

5.3.2 Ohodnotenie a výber viet

Samotná sumarizačná metóda tak ako bola navrhnutá v publikácii *Using latent semantic analysis in text summarization and summary evaluation* [14] využíva rozklad matice V^T slov proti vetám. Matica popisuje mieru významnosti viet v témach dokumentu. Následne sa vyberie veta s najväčšou dĺžkou vektorovej reprezentácie $\Sigma^2 \times V^T$. Násobením Σ^2 sa zohľadní štatistická významnosť hlavných tém, ktorá je úmerná druhej mocnine príslušného singulárneho čísla. Formálne sa jedná o výpočet v k rozmernom latentnom priestore tém dĺžky vektoru s_r pre r-tú vetu podľa vzorca 5.5. Tento algoritmus má oproti algoritmu, ktorý predstavili Gong a Liu [5] výhodu v tom, že témy majú rozdielnu významnosť identifikovanú maticou Σ.

$$s_r = \sqrt{\sum_{i=1}^{k} v_{ri}^2 \sigma_i^2} \tag{5.5}$$

Do výsledného súhrnu sú následne zaradené vety, ktoré majú najvyššie hodnoty s. Dôležitá téma môže byť potom v súhrne vyjadrená niekoľkými vetami. Metóda LSA môže byť použitá v mnohých modifikáciách. Niektoré sú v krátkosti popísané v práci Steinbergera a Ježka [6]. Treba podotknúť, že SVD nie je jediná metóda uplatňujúca sa pri spracovaní textu. Ďalšou podobnou metódou je napríklad metóda *Non-negative matrix factorization*, ktorá rozkladá maticu A na dve matice.

5.3.3 Príklad

Postup práce algoritmu predvediem na príklade[3] uvedenom nižšie. Rovnaký úryvok textu bol použitý ako názorná ukážka princípu Luhnovej metódy v sekcii 5.1.1. Tabuľka 5.1 obsahuje počet výskytov významných slov, ktorých frekvencia je v ukážkovom texte väčšia ako jeden výskyt. Z tejto tabuľky je následne vygenerovaná matica A, ktorá reprezentuje text ukážkového príkladu. Zaujímavú časť matice je možné vidieť vo výraze 5.6. Každá veta v ukážkovom úryvku je očíslovaná (tučne vyznačené číslo v zátvorkách pred začiatkom vety), aby sa na ňu dalo jednoducho odkazovať. Číslo slúži ako jedinečné ID a zároveň ako index do stĺpca matice A z výrazu 5.6. Na prvý pohľad je vidieť, že matica A je veľmi riedka. Obsahuje len jednotky a nuly až na jeden prípad, kedy sa slovo „dítě“ objavilo vo vete 3 dvakrát.

> **(0)** Jednalo se o případ chlapce v 6. třídě, který měl problémy s učením. **(1)** Přerostly až v reparát z jazyka na konci školního roku. **(2)** Nedopadl bohužel dobře a tak musel opakovat 6. třídu, což se chlapci ani trochu nelíbilo. **(3)** Připadal si, že je mezi malými dětmi a realizoval se tím, že si ve třídě o rok mladších dětí budoval vedoucí pozici. **(4)** Dost razantně. **(5)** Fyzickou převahu měl, takže to nedalo až tak moc práce.

[3] Úryvok textu prevzatý z http://www.prevko.cz/dite/skutecne-pribehy-deti

index riadku v matici A	*slovo*	*počet výskytov*
0	třída	3
1	chlapec	2
2	dítě	2
3	rok	2

Tabulka 5.1: Tabuľka počtu výskytov významných slov v ukážkovom úryvku

$$A = \begin{bmatrix} \mathbf{1} & 0 & \mathbf{1} & \mathbf{1} & 0 & 0 \\ \mathbf{1} & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{2} & 0 & 0 \\ 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{5.6}$$

$$s = \begin{bmatrix} 3{,}187 & 3{,}187 & 3{,}440 & 2{,}988 & 2{,}607 & 2{,}912 \end{bmatrix} \tag{5.7}$$

Položky matice A sa najskôr normalizujú podľa výrazu 2.1. Normalizácia však prebehne pre každú vetu (stĺpcový vektor matice A) zvlášť. Na normalizovanú maticu A sa aplikuje algoritmus singulárnej dekompozície a z výsledných matíc Σ a V^T sa podľa vzťahu 5.5 vypočíta ohodnotenie každej vety. Výsledný vektor s ohodnotenia viet pre ukážkový úryvok je uvedený vo výraze 5.7. Z neho je vidieť, že najvyššie ohodnotenie dosiahli prvé tri vety. Ak by sme chceli do extraktu vybrať, rovnako ako v príklade pre Luhnovu metódu, len dve vety, tak by výsledný extrakt vyzeral nasledovne:

> Jednalo se o případ chlapce v 6. třídě, který měl problémy s učením. Nedopadl bohužel dobře a tak musel opakovat 6. třídu, což se chlapci ani trochu nelíbilo.

5.3.4 Použitie pre HTML dokument

Metóda LSA patrí medzi pokročilé metódy analýzy textu. Pri sumarizácii sa nevyužíva žiadna explicitne viditeľná heuristika čo znamená, že nemožno využiť značkovacích schopností jazyka HTML pre jej zlepšenie. Možnosťou, ktorá ma napadá je zvýhodniť váhy slov v značkách `<strong>`, `<em>`, ... v matici A. Tým by sa vo výsledku podporili témy, do ktorých tieto slová spadajú. Problém by však mohol nastať vtedy, ak by na základe tejto úpravy niektorá z tém bola taká výrazná, že by prekryla ostatné významné témy v dokumente. Do výsledného súhrnu by sa potom vybrali len vety s jednou témou hoci ich je v pôvodnom dokumente obsiahnutých viacej.

Kapitola 6

Návrh sumarizátoru

Téma sumarizácie textu nie je nová, a preto existuje niekoľko funkčných implementácií. Vo väčšine implementácií sa predpokladá, že prirodzeným jazykom dokumentu je angličtina, prípadne ďalšie rozšírené jazyky ako nemčina, španielčina a podobne. Spomínané implementácie často pracujú ako samostatné aplikácie, ktoré ako vstup uvažujú neformátovaný text alebo špecifický formát súboru XML. Mojim cieľom je navrhnúť modul, ktorý bude použiteľný ako samostatná aplikácia i knižnica a dokáže spracovať dokumenty v českom jazyku. Modul bude pracovať s formátom HTML, dokáže automatizovane vyhľadať hlavný text v dokumente a s využitím metainformácií získaných z HTML dokumentu vytvorí súhrn pomocou modernej sumarizačnej techniky.

6.1 Existujúce implementácie

Ako som spomínal na začiatku tejto sekcie, tak existuje niekoľko implementácií sumarizátorov. Veľké množstvo z nich sú ale veľmi jednoduché implementácie ako napríklad `https://github.com/thavelick/summarize/`, ktoré sú pre serióznu prácu nevyhovujúce.

Je možné nájsť aj implementácie, ktoré využívajú pokročilejšie techniky. Implementácia sumarizátora Open Text Summarizer poskytuje podporu pre množstvo jazykov a medzi nimi aj češtinu. Využíva sumarizačnú metódu založenú na heuristike $\mathrm{TF}(t, d)$, čo je stále veľmi jednoduchá metóda. Knižnica je napísaný v jazyku C a obsahuje tiež jednoduché GUI[1]. Nevýhodou je, že ako vstup podporuje len neformátovaný text a nemôže teda využiť žiadne metainformácie o dokumente. Disponuje wrapperom v jazyku Python, no jej hlavná časť napísaná v jazyku C je preložiteľná len na platforme UNIX.

Pokročilými sumarizátormi sú knižnice Musutelsa a Almus vyvíjané na Západočeskej univerzite v Plzni v jazyku Java. Sumarizačný algoritmus v oboch je založený na metóde LSA spomenutej v sekcii 5.3. Umožňujú viac-dokumentovú sumarizáciu a *Almus* okrem tradičnej sumarizácie disponuje aj algoritmom pre aktualizačnú sumarizáciu. Vstupným dokumentom je anglický text vo formáte XML[2]. Text je teda nutné pred vstupom do sumarizátoru previesť na kompatibilný XML formát, čo je pre užívateľa značne nepohodlné. Obe knižnice je možné rozšíriť o sumarizáciu ďalších jazykov, pričom *Musutelsa* disponuje takýmto rozšírením pre český jazyk. Nevýhodou knižnice *Musutelsa* je, že vývoj sa zdá byť zastavený už od roku 2007 a dokumentácia je neúplná. Pre knižnicu *Almus* zase nie sú k dispozícii zdrojové kódy. Obe knižnice by som označil za akademické projekty, ktoré majú

[1] GUI - Graphical User Interface, grafické užívateľské rozhranie
[2] XML - eXtensible Markup Language, rozšíriteľný značkovací jazyk

za cieľ len čo najpresnejšie otestovať kvalitu implementovaného sumarizačného algoritmu, a nie sú preto vhodné pre reálne využitie.

Najkomplexnejším riešením sumarizátoru je platforma MEAD. Sumarizátor je napísaný v jazyku Perl a umožňuje modulárne pridávanie rôznych metód sumarizácie ako aj modulov pre podporu nových jazykov. Balík už obsahuje množstvo typov sumarizačných metód pripravených na použitie. Rovnako ako v predchádzajúcich knižniciach je vstupom dokument vo formáte XML. Okrem sumarizačných algoritmov táto platforma obsahuje aj nástroje pre hodnotenie výsledných súhrnov.

6.2 Špecifikácia funkcionality

Sumarizátor bude fungovať ako modul jazyka Python. Modul bude podporovať vstupný dokument vo formáte HTML, pričom bude schopný sám identifikovať hlavný text. Zároveň by však mala byť ponechaná možnosť pridania podpory pre nový vstupný formát. Výstupom aplikácie bude súhrn v textovom formáte, ktorý bude kvôli prehľadnosti obsahovať na každom riadku jednu extrahovanú vetu.

Implementované budú všetky metódy predstavené v kapitole 5. Nemalo by byť však ťažké pridať novú implementáciu metódy. Algoritmus každej metódy sa bude snažiť v čo najväčšej miere využiť možnosti, ktoré mu poskytuje formát HTML k zlepšeniu svojej činnosti, tak ako to bolo spomenuté v kapitole 5. Najväčšia pozornosť bude venovaná implementácii metódy LSA (sekcia 5.3), ktorej predpokladané problémy preberiem v nasledujúcom texte.

Prvým problémom metódy LSA je pomerne náročný algoritmus singulárnej dekompozície. Algoritmus pracuje s maticami veľkých rozmerov, ktoré predstavujú pre aplikáciu vysokú pamäťovú náročnosť. Knižnica *gensim*, ktorá bude s najväčšou pravdepodobnosťou použitá, síce pracuje s riedkymi maticami pomerne efektívne, ale treba sa pripraviť na možné problémy pri spracovaní objemných dokumentov.

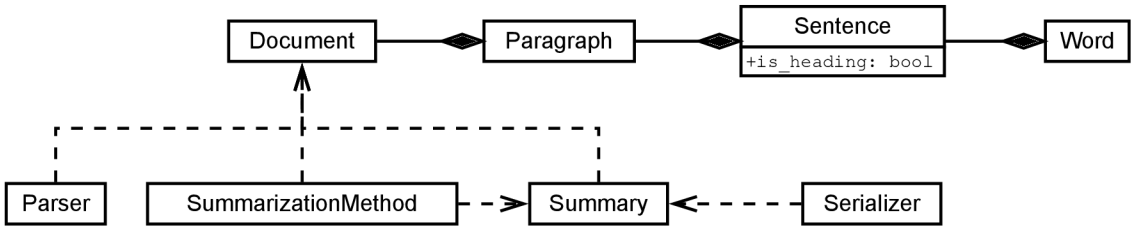
V sekcii 5.3 som rovnako ako pre ostatné metódy navrhol určité vylepšenie metódy LSA, tak aby boli preferované slová, ktoré autor článku označil za „zaujímavé“. To je možné pomocou zvýhodnenia váh metriky TF*IDF. Váhu bude potrebné určiť na základe experimentov s implementovaným sumarizátorom. Pravdepodobne zvolím, podobne ako je uvedené v práci k projektu *Musutelsa* [7], zvýhodnenie pomocou multiplikatívnej konštanty.

V práci k projektu *Musutelsa* [7] sa ukázalo, že sumarizátor považoval dlhé vety, za dôležitejšie. To je dané tým, že dlhé vety obsahujú aj viacej kľúčových slov. Ohodnotenie viet je preto potrebné normalizovať. Preto sa Křišťan [7] rozhodol normalizovať ohodnotenie vety s podľa výrazu 6.1, kde $rank$ je pôvodné ohodnotenie vety, $\text{length}(s)$ je dĺžka vety a $treshold \geq 0$ je premenná ovplyvňujúca zvýhodnenie kratších viet. Hodnota 0 znamená, že sa dĺžka vety nebude brať do úvahy a s rastúcou hodnotou rastie aj šanca výberu kratších viet do súhrnu.

$$rank_{norm}(s) = \frac{rank}{\text{length}(s)^{treshold}} \tag{6.1}$$

6.3 Architektúra modulu

Vytvorený modul som sa snažil navrhnúť tak, aby ho bolo možné jednoducho rozšíriť o nové metódy sumarizácie, alebo jazyk. Takisto je možné vymeniť formát vstupného i výstupného dokumentu. Architektúra hlavnej časti modulu je naznačená na obrázku 6.1.



Obrázek 6.1: Architektúra modulu pre sumarizáciu

Základnou triedou je trieda `Document`, ktorá predstavuje objektovú formu dokumentu nezávislú na vstupných dátach. Trieda `Document` sa skladá z inštancií triedy `Paragraph` reprezentujúcej odsek a obsahujúcej vety (trieda `Sentence`). Každá veta potom obsahuje slová ako inštancie triedy `Word`. Nadpisy sú modelované ako vety, pričom ich je možné odlíšiť od bežných viet atribútom `Sentence.is_heading`. S týmto modelom dokumentu sa pracuje v celom module. Trieda `Parser` má za úlohu previesť vstupný formát dokumentu na inštanciu triedy `Document`. Jednotlivé položky modelu dokumentu (odseky, vety, slová) môžu obsahovať rôzne metainformácie, ktoré boli súčasťou vstupného formátu dokumentu. Tieto môžu byť následne využité pri ďalšom spracovaní.

Objekt triedy `Document` má dve nepovinné závislosti, ktorými sú *stemmer* a množina *stop slov*. Hoci sú tieto dve závislosti pre dokument nepovinné, tak sú samozrejmou súčasťou takmer každého sumarizačného algoritmu, pretože sa pomocou nich značne zvyšuje kvalita výsledných súhrnov.

Trieda `Parser` môže implementovať prevod z akéhokoľvek formátu na inštanciu triedy `Document` a tým je celý modul nezávislý na vstupnom formáte. Pre potreby tejto práce bude modul obsahovať minimálne implementáciu pre prevod z HTML formátu. Pre prevod bude potrebné v HTML dokumente najprv vyhľadať hlavný text, čo je možné pomocou jedného z modulov spomenutých v sekcii 3.1, prípadne ich kombináciou.

Vytvorená inštancia triedy `SummarizationMethod` závisí na inštancii triedy `Document`. Trieda implementuje vybranú metódu sumarizácie. V základnej implementácii modulu sa predpokladá prítomnosť metód spomenutých v kapitole 5. Výsledkom je objekt triedy `Summary`, ktorý obsahuje pôvodný dokument a výsledok sumarizácie spolu so štatistikami priebehu a výsledku aplikovanej sumarizačnej metódy. Tento objekt môže okrem iného slúžiť pre účely vyhodnocovania kvality sumarizácie.

Výsledný objekt triedy `Summary` je predaný objektu triedy `Serializer`, ktorá prevedie výsledok sumarizácie na vhodný výstupný formát. Základným výstupným formátom by mal byť neformátovaný text. Kedykoľvek by malo byť možné implementovať triedu vykonávajúcu prevod do iného vhodného výstupného formátu.

Po vytvorení inštancie triedy `Document` a pred samotnou sumarizáciou je možné dokument ľubovoľne upraviť. Na dokument je možné použiť algoritmus pre *rezolúciu anaforických vzťahov* alebo nahradiť v dokumente skratky ich plným tvarom.

Kapitola 7

Implementácia modulu

V tejto kapitole sa snažím popísať implementáciu funkcionality navrhnutej v kapitole 6. Taktiež popisujem zmeny, ktoré som vykonal pri riešení implementačných detailov či rôznych problémov. Implementovaný modul vychádza hlavne z návrhu znázorneného na obrázku 6.1.

Pre implementáciu modulu som zvolil programovací jazyk Python. Je to skriptovací jazyk, ktorý je nezávislý na platforme a umožňuje veľmi rýchle prototypovanie. Pre tento jazyk existuje niekoľko prekladačov i interpretov. Referenčná implementácia interpretu, zvaná *CPython*, spolu s ďalšími masovo rozšírenými implementáciami, je pomerne rýchla, a v prípade nevyhovujúceho výkonu umožňuje prepísať kritické časti kódu do jazyka C. Jazyk poskytuje užitočné knižnice pre prácu s textom (*nltk*) i s matematickými štruktúrami (*NumPy*, *SciPy*). Jazyk bol ale zvolený aj preto, že s ním mám veľmi dobré skúsenosti a nemalú prax.

Implementácia bola rozdelená na dva samostatné moduly. Modul *readability.py* zaobstaráva extrakciu hlavného textu z HTML dokumentu. Modul *sumy* je implemetáciou sumarizačných a evaluačných algoritmov. Zdrojové kódy oboch modulov sú dostupné na serveri GitHub pod licenciami *The BSD 2-Clause License* a *Apache License, Version 2.0*. Pri písaní zdrojových kódov som sa snažil v čo najväčšej miere rešpektovať konvencie popísané v dokumente PEP8. To umožňuje pomerne ľahké zorientovanie sa v kóde programátorom, ktorí už jazyk Python poznajú. Pre správu implementovaných modulov je vytvorený skript *setup.py*, ktorý umožňuje jednoduchú inštaláciu i testovanie a obsahuje metadáta potrebné na to aby mohol byť modul ľahko dohľadateľný na PyPI.

Modul je testovaný pomocou *Continuous Integration* serveru Travis CI a podporuje verzie jazyka Python 2.6, 2.7, 3.2, 3.3. Modul pracuje jednotne v celom kóde s reťazcami typu `unicode` po vzore jazyka Python verzie 3. Pre testy sa využíva knižnica *nosetests* s modulom *coverage* pre sledovanie pokrytia kódu, ktoré bolo v čase písania práce 82%. Skutočné pokrytie kódu je vyššie, pretože prevzaté kódy nie sú testované, a takisto nie je možné na 100 % otestovať kódy starajúce sa o prenositeľnosť kódu medzi jednotlivými verziami jazyka v jednej jeho verzii.

7.1 Použité nástroje

Ako som spomínal vyššie tak knižnica *sumy* je závislá na niekoľkých ďalších knižniciach. Väčšinou je využitá len malá časť funkcionality, ktorú tieto knižnice ponúkajú, čo môže byť vnímané negatívne. No algoritmy v týchto knižniciach sú optimalizované a dobre otestované, a preto som ich uprednostnil pred pokusmi o vlastnú implementáciu. Alternatívy v podobe menších špecializovaných modulov neboli pre jazyk Python k dispozícií.

7.1.1 Tokenizer textu na vety a slová

Knižnica *nltk* je balíkom nástrojov bežne používaných pri spracovaní prirodzeného jazyka. Z tejto knižnice je v mojej práci využitý len tokenizer neštruktúrovaného textu na slová a vety. Pre jazyk Python existuje i menší balík text-sentence, ktorý obsahuje len tokenizer. Tento tokenizer vznikol ako na jazyku nezávislá samostatná náhrada za tokenizer obsiahnutý v knižnici *nltk*, a poskytuje dokonca niektoré dodatočné informácie o texte. Dokáže rozpoznať začiatky a konce viet a v nich následne deteguje, okrem bežných slov, mená, skratky, výpustku i niekoľkonásobnú interpunkciu. Medzi ďalšie prednosti patrí schopnosť vysporiadať sa so skráteným tvarom slov, ktorý je často používaný v anglickom jazyku (napr. *isn't*). Najväčšia pozornosť je venovaná rozpoznaniu čísiel v rôznych formátoch. Knižnica dokonca správne rozpozná rímske číslice. No po zhliadnutí zdrojových kódov som zistil, že je v čase písania tejto práce implementácia tokenizeru ešte nedokončená, a preto som sa rozhodol zatiaľ využívať tokenizer z balíka nástrojov *nltk*.

7.1.2 Knižnica pre prácu s maticami

NumPy je numerická knižnica vychádzajúca z knižníc *Numeric* a *Numarray*. Jej reprezentácia vektoru (všeobecne matíc) slúži ako základ pre mnoho knižníc. Poskytuje formát polí používaných v jazyku C i Fortran a algoritmy pre ich efektívnu manipuláciu. Okrem bežných implementácií vektorov a matíc ponúka knižnica i niektoré špecializované reprezentácie vektorových objektov. Knižnicu používam v mojej práci pri reprezentácii dokumentu v sumarizátore založenom na metóde LSA.

7.1.3 Knižnica pre singulárnu dekompozíciu

SciPy je open source modul pre jazyk Python, ktorý obsahuje balík programov pre vedecko-technické výpočty. Z tohto balíku som v mojej práci využil len algoritmus určený pre singulárnu dekompozíciu matíc. Balík je závislý na knižnici *NumPy*, z ktorej využíva hlavne objekty pre reprezentáciu multi-dimenzionálnych polí. Hoci sú pre Python dostupné aj samostatné knižnice implementujúce singulárnu dekompozíciu matíc, tak všetky mnou vyskúšané moduly trpeli nedostatkami v podobe chýb alebo vysokej časovej a pamäťovej náročnosti.

Ako prvá bola podrobená testovaniu knižnica *sparsesvd*, ktorá však v čase písania práce nepodporovala Python verzie 3. Ešte počas písania práce bola táto podpora pridaná, no pri dôkladnejšom testovaní knižnice sa ukázalo, že obsahuje niekoľko závažných chýb. Ďalším možným kandidátom je knižnica *SDV.py*, ktorá je ale napísaná v čistom Pythone, a preto sú algoritmy v nej implementované vysoko časovo a pamäťovo náročné. Tretiu v poradí som sa rozhodol preskúmať knižnicu *gensim*, ktorá využíva knižnicu *sparsesvd*, no ak nie je dostupná, tak použije vlastnú implementáciu singulárnej dekompozície. No keďže knižnica

je určená na iné účely aké potrebujem v mojej práci ja, a pretože sama závisí na knižnici *SciPy*, tak som sa rozhodol jej použitiu vyhnúť.

7.1.4 Knižnica pre terminálové užívateľské rozhranie

Knižnica *docopt* umožňuje pomocou formalizovaného popisu terminálového užívateľského rozhrania (ďalej len CLI[1]) automaticky zostrojiť parser argumentov príkazového riadku a vrátiť rozpoznané parametre programu. V mojej práci je táto knižnica použitá na vytvorenie CLI pre skript, ktorý umožňuje pohodlné spustenie sumarizácie pre ľubovoľný dokument alebo URL adresu z príkazového riadku. Jazyk Python síce obsahuje moduly pre parsovanie argumentov príkazového riadku v štandardnej knižnici, ale ich používanie je omnoho zložitejšie a menej udržovateľné. Navyše knižnica *docopt* vykonáva aj jednoduchú kontrolu niektorých parametrov a sama sa postará o obslúženie náležitých chybových stavov.

7.2 Rozdelenie na časti

Implementácia funkcionality z kapitoly 6 bola rozdelená na dva samostatné moduly jazyka Python. Prvý z modulov *readability.py* zabezpečuje rozpoznávanie a parsovanie hlavného textu z HTML dokumentu. Jedná sa o port pôvodnej implementácie platformy Readability v jazyku Python. Podobných portov existuje niekoľko, a preto som jeden vybral, upravil ho pre potreby extrakcie anotovaného textu a zároveň pridal podporu pre jazyk Python verzie 3. Anotáciou je myslené sémantické značkovanie textu dostupné v HTML dokumente.

Na module *readability.py* je závislý druhý z modulov, *sumy*. Modul *sumy* sa zaoberá samotnou sumarizáciou pomocou metód popísaných v kapitole 5. Modul je rozdelený na viacero význačných podmodulov, ktoré sú bližšie popísané v nasledujúcich sekciách práce.

7.2.1 Modul models

Modul `sumy.models` obsahuje modely dokumentu. V čase písania tejto práce boli implementované dva modely. Prvým modelom je objektový model z modulu `sumy.models.dom`. Je základným modelom a pracuje sa s ním v celej aplikácií. Je to zjednodušená implementácia modelu navrhnutého v sekcii 6.3. Ako sa pri implementácii ukázalo, tak slovo nie je potrebné modelovať ako samostatný objekt, pretože stačí ak je modelované ako reťazec obsahujúci toto slovo. Navyše model slova ako samostatného objektu so sebou prinášal nadbytočnú réžiu v podobe veľkej spotreby pamäte. Takisto sa zmenil model vety, ktorý už nie je kompozíciou slov, ale je to samostatný reťazec obsahujúci celú vetu. Parsovanie slov z vety je potom úlohou samotnej vety. Táto zmena bola vykonaná z dôvodu nedostatočnej kvality tokenizera z balíku *nltk*, ktorý považoval za slová aj čiarky, zátvorky a iné nevhodné znaky.

Ďalším implementovaným modelom je vektorový model dokumentu postavený na metrike *term frequency* umiestnený v module `sumy.models.tf`. Tento model pracuje s množinou termov (slov). Model poskytuje informácie ako frekvencia slov, veľkosť vektorového modelu a podobne.

[1] CLI - Command Line Interface

7.2.2 Modul evaluation

Modul `sumy.evaluation` slúži na hodnotenie súhrnov. V module sú implementované metódy založené na *podobnosti obsahu* (angl. *content-based*) a ko-selekčné prístupy k hodnoteniu súhrnov. Aby sa vyhodnocovanie dalo plne automatizovať tak boli implementované metódy, ktoré vyžadujú k svojej činnosti len referenčný súhrn. Z metód založených na podobnosti obsahu sú implementované *cosine similarity* a *unit overlap*. Obe pracujú s vektorovým modelom dokumentu, ktorý je implementovaný ako trieda `sumy.models.TfDocumentModel`. Metódy využívajúce ko-selekčný prístup pracujú s množinami viet referenčného a vyhodnocovaného extraktu. Implementované sú metódy *precision*, *recall* a *F-Score*.

7.2.3 Modul nlp

Modul `sumy.nlp` obsahuje nástroje potrebné pri spracovaní prirodzeného jazyka. Týmito nástrojmi sú stemmer a tokenizer pre český jazyk. Implementácia stemmeru bola prevzatá a v súlade s licenciou mierne upravená. Tokenizer je založený na knižnici *nltk*. Je preto rovnako ako tokenizer z tejto knižnice závislý na jazyku. Vhodnejším riešením by bol nejaký na jazyku nezávislý tokenizer, no pre jazyk Python nebol v čase písania práce dostupný žiaden vhodnejší.

7.2.4 Modul parsers

Modul `sumy.parsers` obsahuje parsery textového tvaru dokumentu a následný prevod dokumentu na objektový model. Každý parser je implementáciou entity `Parser` zobrazenej na obrázku 6.1. Najjednoduchším parserom je parser pre neštruktúrovaný text (ďalej len *plaintext*). Ten rozdelí vstupný dokument na odseky, vety a slová, pričom je schopný rozoznať nadpisy (riadky so všetkými písmenami veľkými). Jednotlivé parsery môžu poskytovať dodatočné informácie dostupné vo vstupnom dokumente. Príkladom takéhoto parseru je trieda `HtmlParser`, ktorá je schopná poskytnúť, okrem samotného objektového modelu dokumentu, informácie o sémantickom značkovaní jednotlivých častí dokumentu. Práve tento parser je závislý na knižnici *readability.py*, ktorá tvorí jeho takmer celú aplikačnú logiku.

7.2.5 Modul summarizers

Modul obsahujúci implementácie jednotlivých sumarizačných metód. Každá implementácia metódy v podobe sumarizátoru reprezentuje entitu `SummarizationMethod` z obrázku 6.1. Najjednoduchší z implementovaných algoritmických sumarizátorov predstavuje inštancia triedy `sumy.summarizers.LuhnSummarizer`. Jeho implementácia je založená na princípe Luhnovej metódy popísanej v sekcii 5.1. Sumarizátor pracuje s modelom dokumentu reprezentovaným inštanciou triedy `sumy.models.TfDocumentModel` a okrem jednej voliteľnej závislosti, zoznamu *stop slov*, nepotrebuje žiadne dodatočné závislosti.

Ďalší implementovaný sumarizátor vychádza z popisu Edmundsonovej metódy, ktorý je uvedený v sekcii 5.2. a je reprezentovaný triedou `sumy.summarizers.EdmundsonSummarizer`. Sumarizátor v sebe obsahuje všetky štyri Edmundsonove sumarizačné metódy (*clues*). Hodnoteniu viet pre každú z týchto metód je možné priradiť váhu, pričom implicitne sú váhy nastavené podľa experimentov vykonaných Edmundsonom tak, aby sa dosahovalo čo možno najlepších výsledkov. Sumarizátor k svojej činnosti vyžaduje neprázdne zoznamy *null slov*, *stigma slov* a *bonus slov*.

Posledným implementovaným je sumarizátor založený na metóde LSA, ktorej bližší popis je uvedený v sekcii 5.3. Je reprezentovaný triedou `sumy.summarizers.LsaSummarizer`. Sumarizátor je funkčný až po nainštalovaní knižníc *NumPy* a *SciPy*. Knižnica *NumPy* sa stará o efektívnu reprezentáciu a prácu s maticami a z knižnice *SciPy* je využitý algoritmus pre singulárnu dekompozíciu. Sumarizátor je implicitne nastavený tak aby neredukoval latentný priestor dokumentu a teda zachoval všetky jeho dimenzie. Pre redukciu dimenzionality je potrebné nastaviť konštantu `LsaSummarizer.REDUCTION_RATIO` na hodnotu v intervale $(0; 1)$. Sumarizátor je ďalej závislý na zozname *stop slov* a modeli dokumentu dodanom ako inštancia triedy `sumy.models.TfDocumentModel`.

7.3 Tok programu

Program začína parsovaním vstupného dokumentu. Dokument môže byť zadaný ako *plaintext* alebo ako súbor HTML či URL adresa. Ak je vstupný dokument vo formáte HTML, tak sa za pomoci knižnice *readability.py* extrahuje hlavný text dokumentu. Text dokumentu sa prevedie na objektovú reprezentáciu vhodnú pre použitie v programe za pomoci tokenizácie textu na vety a slová. Slová sa často podrobia dodatočnému spracovaniu (stemming a vyradenie *stop slov*). Objektová reprezentácia dokumentu sa potom využíva vo vybranom sumarizátore, kde sa však môže pracovať aj s inou reprezentáciou dokumentu, najčastejšie založenou na metrike $\mathrm{TF}(t, d)$. Sumarizátor následne vráti súhrn s počtom viet, ktoré užívateľ zadal buď ako absolútne číslo alebo ako percentuálne vyjadrenie vzhľadom k dĺžke pôvodného dokumentu.

Aby bol užívateľ od zložitosti procesu sumarizácie čo najviac odtienený a mohol jednoduchým spôsobom používať sumarizačné funkcie, bola vytvorená utilita *sumy*. Tá umožňuje z pohodlia príkazového riadku vyvárať sumarizácie len zadaním názvu sumarizačnej metódy, URL adresy a voliteľne dĺžky výsledného súhrnu (implicitná dĺžka je nastavená na 20 % dĺžky pôvodného dokumentu).

Kapitola 8

Vyhodnotenie sumarizácií

V tejto kapitole popisujem experimenty, ktorých cieľom bolo medzi sebou porovnať kvalitu súhrnov generovaných za pomoci implementovaných sumarizačných metód. Pre vyhodnocovanie sumarizácií som použil tri metódy založené na podobnosti výberu viet, a to *precision*, *recall*, *F-score* a metódy *Cosine similarity* a *Unit overlap* založené na podobnosti obsahu. Všetky tieto metódy sú súčasťou evaluačného modulu knižnice *sumy*. Pre pohodlné vyhodnocovanie z prostredia príkazového riadku bola implementovaná utilita *sumy_eval*, ktorá dokáže spustiť všetky evaluačné metódy nad zadaným dokumentom. Pre účely experimentovania bol dodatočne implementovaný sumarizátor, ktorý extrahuje vety do súhrnu náhodne. Cieľom bolo porovnať implementované metódy aj s týmto sumarizátorom a zistiť tak, či algoritmy z modulu *sumy* pracujú lepšie ako náhodný výber viet. Náhodný sumarizátor teda slúžil ako spodná hranica ohodnotenia súrnu. Pretože náhodný sumarizátor pracuje nedeterministicky, tak boli experimenty s ním vždy 100-krát opakované a výsledky spriemerovanié.

8.1 Dátové sady

Väčšina korpusov pre vyhodnocovanie sumarizácií sa zameriava na hodnotenie sumarizácií neštruktúrovaného textu v anglickom jazyku, a preto som vytvoril vlastný korpus. Referenčné extrakty som vytvoril taktiež sám, pretože pre zadané články neboli k dispozícii. To so sebou prináša nevýhodu v podobe zanesenia určitej subjektivity do hodnotenia. Lepšie riešenie, ktoré je možné nájsť aj v práci *Hodnocení kvality sumarizátorů textů* [15], by bolo zhromaždiť súhrny od niekoľkých sudcov a vety, ktoré boli vybrané najväčším počtom sudcov zahrnúť do referenčného extraktu. Bohužiaľ sa mi nepodarilo zohnať dostatočný počet sudcov na to aby boli výsledky relevantné. Malý počet sudcov totiž zapríčinil, že extrakty vytvorené sudcami zdieľali málo viet a referenčný extrakt bol potom príliš krátky.

Pri experimentovaní som použil 2 dátové sady HTML dokumentov s textom napísaným v českom jazyku. Prvá dátová sada je získaná zo serveru `http://zdrojak.cz/` (ďalej len **ZDROJAK13**) a druhá je tvorená textami zo serveru `http://cs.wikipedia.org/` (ďalej len **WIKI13**). Prvá dátová sada obsahuje HTML dokumenty s textom zameraným na oblasť IT (návrhové vzory). Tieto dokumenty boli vybrané hlavne preto, že dokument HTML obsahuje v dostatočnom množstve kvalitné sémantické značkovanie a členenie textu. Články v tejto sade sa venujú v texte vždy jednej až trom témam. Články dátovej sady **ZDROJAK13** obsahovali aj krátke úvodníky, ktoré sa dali použiť ako abstrakty dokumentov pri evaluačných metódach založených na podobnosti obsahu. Dátová sada **WIKI13** obsahuje

texty z rôznych oblastí a s rôznorodou dĺžkou. Texty v tejto sade ale obsahujú mnoho častí (referencie, zoznamy odkazov na príbuzné témy, . . .), ktoré algoritmus extrakcie textu určí ako nevhodné, a preto je výsledná dĺžka sumarizovaných dokumentov omnoho kratšia. Táto dátová sada neobsahuje sémantické značkovanie v takom množstve a kvalite ako sada **ZDROJAK13**, ale stále poskytuje dostatočné informácie o základnom členení textu. Parametre oboch dátových sád je možno vidieť v tabuľkách 8.1 a 8.2. Pre dátovú sadu **WIKI13** neboli k dispozícii abstrakty, a preto vyhodnocovanie prebiehalo len oproti referenčným extraktom.

počet dokumentov	12
minimálny počet viet v dokumente	44
maximálny počet viet v dokumente	113
priemerný počet viet v dokumente	65
priemerný počet slov v dokumente	987
priemerný počet významových slov v dokumente	698
priemerný počet unikátnych významových slov v dokumente	447

Tabulka 8.1: Prehľad parametrov dátovej sady **ZDROJAK13**

počet dokumentov	21
minimálny počet viet v dokumente	21
maximálny počet viet v dokumente	114
priemerný počet viet v dokumente	54
priemerný počet slov v dokumente	788
priemerný počet významových slov v dokumente	583
priemerný počet unikátnych významových slov v dokumente	440

Tabulka 8.2: Prehľad parametrov dátovej sady **WIKI13**

8.2 Porovnanie metód

Evaluácia sumarizácií bola uskutočnená pre všetky implementované metódy v module *sumy*. Konfigurácia jednotlivých sumarizátorov bola ponechaná na východzie hodnoty. Náhodný sumarizátor používal rovnomerné rozloženie. Luhnova metóda brala do úvahy zhluky s maximálne 4 slovami medzi významnými slovami a mala k dispozícii zoznam českých *stop slov*. Edmundsonova metóda používala metódy *Cue*, *Title* a *Location* s váhami nastavenými na hodnotu 1. Slovník *bonus slov* bol nastavený významnými slovami označenými v HTML dokumente, slovník *stigma slov* bol nastavený na slová z odkazov umiestnených v HTML dokumente a slovník *null slov* obsahoval české *stop slová*. Nastavenia parametrov jednotlivých metód *Cue*, *Title* a *Location* boli ponechané na východzie hodnoty. Metóda LSA redukovala latentný priestor na $\frac{1}{10}$ pôvodnej veľkosti pričom minimálny počet dimenzií bol nastavený na hodnotu 3.

Ako prvé boli vyhodnotené sumarizácie pre indikatívne extrakty, ktoré predstavovali text v rozsahu približne 3 % rozsahu pôvodného dokumentu (1–3 vety). Výsledky indikatívnych súhrnov pre obe dátové sady sú uvedené v tabuľkách 8.3 a 8.4. Čísla v bunkách tabuliek udávajú priemernú hodnotu metódy pre danú metriku získanú z ohodnotenia jednotlivých dokumentov.

Z tabuliek 8.3, 8.4 je vidieť, že najlepšie výsledky (tučne vyznačené hodnoty) pre indikatívny extrakt dosiahla v prevažnej väčšine metrík Edmundsonova metóda. To je spôsobené tým, že metóda dokáže využiť najviac metadát z HTML dokumentu. Slovníky *bonus slov* a *stigma slov* metóda získava na základe sémantických značiek dodaných priamo autorom textu, slová v nadpisoch sú korektne rozpoznané a je im priradená zodpovedajúca významnosť a metóda je tiež schopná využiť ohodnotenie viet na základe ich pozície. Preto je metóda schopná vybrať hodnotné vety už pri malej dĺžke súhrnu.

V prípade metriky *kosínusovej vzdialenosti* s pôvodným dokumentom je pre obe dátové sady najúspešnejšia Luhnova metóda, čo je dané tým, že metóda vyberá vety obsahujúce zhluky najfrekventovanejších slov (metrika založená na $\mathrm{TF}(t, d)$). Na princípe $\mathrm{TF}(t, d)$ zároveň pracuje aj evaluácia pomocou metriky *kosínusovej vzdialenosti* a metóda logicky vyberá vety, ktoré túto metriku ovplyvňujú v dokumente najvýraznejšie čím sú aj vety v súhrne spätne ohodnotené vysokou mierou zhody.

Hodnotiaca metrika/Typ sumarizátoru	*Random*	*Luhn*	*Edmundson*	*LSA*
Precision	0,153	0,000	**0,250**	0,083
Recall	0,018	0,000	**0,027**	0,007
F-score	0,032	0,000	**0,049**	0,013
Cosine similarity (abstract)	0,110	0,113	**0,196**	0,159
Cosine similarity (extract)	0,271	0,257	**0,381**	0,306
Cosine similarity (document)	0,321	**0,476**	0,465	0,435
Unit overlap (abstract)	0,062	0,050	**0,093**	0,077
Unit overlap (extract)	0,073	0,078	**0,118**	0,087
Unit overlap (document)	0,046	0,073	**0,073**	0,073

Tabulka 8.3: Výsledky evaluácie indikatívneho súhrnu (3 % pôvodného textu) pre dátovú sadu **ZDROJAK13**

Hodnotiaca metrika/Typ sumarizátoru	*Random*	*Luhn*	*Edmundson*	*LSA*
Precision	0,230	0,143	**0,437**	0,214
Recall	0,026	0,017	**0,054**	0,028
F-score	0,047	0,031	**0,095**	0,048
Cosine similarity (extract)	0,369	0,450	**0,457**	0,404
Cosine similarity (document)	0,530	**0,792**	0,660	0,621
Unit overlap (extract)	0,117	0,145	0,140	**0,149**
Unit overlap (document)	0,083	**0,137**	0,094	0,136

Tabulka 8.4: Výsledky evaluácie indikatívneho súhrnu (3 % pôvodného textu) pre dátovú sadu **WIKI13**

Druhá sada experimentov predstavovala vyhodnotenie sumarizácie pre informatívne extrakty, ktoré predstavovali text v rozsahu približne 10 % rozsahu pôvodného dokumentu (priemerne 8 viet). Výsledky informatívnych súhrnov pre obe dátové sady sú uvedené v tabuľkách 8.5 a 8.6. Čísla v bunkách tabuliek udávajú, rovnako ako v predchádzajúcom prípade, priemernú hodnotu metódy pre danú metriku získanú z ohodnotenia jednotlivých dokumentov.

Tabuľka 8.5 ukazuje, že pre HTML dokument, ktorý má bohatú sémantické štruktúru dosahuje Edmundsonova metóda najlepšie výsledky i pre informatívne extrakty. Avšak v nasledujúcej tabuľke 8.6, kde boli použité dokumenty, ktoré obsahujú len základné značkovanie už dosahuje lepších výsledkov metóda LSA. Tá si totiž vystačí so samotnou štruktúrou textu a sémantické značkovanie pre jej činnosť nepredstavuje významnú výhodu.

Hodnotiaca metrika/Typ sumarizátoru	*Random*	*Luhn*	*Edmundson*	*LSA*
Precision	0,157	0,199	**0,372**	0,151
Recall	0,077	0,102	**0,177**	0,071
F-score	0,102	0,134	**0,238**	0,096
Cosine similarity (abstract)	0,253	0,204	0,254	**0,263**
Cosine similarity (extract)	0,512	0,560	**0,650**	0,566
Cosine similarity (document)	0,524	0,725	**0,731**	0,712
Unit overlap (abstract)	0,092	0,079	**0,093**	0,085
Unit overlap (extract)	0,226	0,259	**0,308**	0,220
Unit overlap (document)	0,182	0,222	0,222	**0,244**

Tabulka 8.5: Výsledky evaluácie informatívneho súhrnu (10 % pôvodného textu) pre dátovú sadu **ZDROJAK13**

Hodnotiaca metrika/Typ sumarizátoru	*Random*	*Luhn*	*Edmundson*	*LSA*
Precision	0,130	0,160	0,225	**0,264**
Recall	0,050	0,067	0,101	**0,117**
F-score	0,069	0,092	0,136	**0,157**
Cosine similarity (extract)	0,428	0,506	0,500	**0,537**
Cosine similarity (document)	0,554	**0,700**	0,572	0,639
Unit overlap (extract)	0,147	0,196	0,188	**0,226**
Unit overlap (document)	0,148	0,217	0,147	**0,223**

Tabulka 8.6: Výsledky evaluácie informatívneho súhrnu (10 % pôvodného textu) pre dátovú sadu **WIKI13**

Z výsledkov hodnotenia sumarizácií je vidieť, že dokonca aj náhodný sumarizátor, ktorý mal slúžiť ako spodná hranica ohodnotenia, dosahuje hodnôt blízkych k súhrnom, ktoré boli získané z algoritmických sumarizátorov. Je to spôsobené tým, že uvedené evaluačné metódy ohodnocujú súhrny veľmi povrchne. Kvôli tomu je potrebné hľadať nie len nové sumarizačné ale i evaluačné algoritmy pre automaticky tvorené súhrny. Steiberger a Ježek [15] predstavil nový prístup k hodnoteniu založený na LSA. Ten by však mohol kvôli použitým algorimom zvýhodňovať mnou implementovaný sumarizátor založený taktiež na LSA. Ak by sa podarilo vyvinúť algoritmus, ktorý by spoľahlivo detegoval kvalitný extrakt, tak vytvorenie spoľahlivého extraktívneho sumarizátoru by sa redukovalo na kombinatorický problém. Dnešný výskum sa však pokúša vyvinúť spoľahlivý evaluátor, ktorý je použiteľný i pre iné ako extraktívne sumarizátory.

8.3 Subjektívne hodnotenie

Keďže všetky algoritmy produkujú súhrn vo forme extraktu, tak nie je vhodné hodnotiť štruktúru viet z pohľadu ich zrozumiteľnosti. V nasledujúcom texte sa kvôli tomu zameriam na hodnotenie súvislosti textu, teda toho ako vety na seba nadväzujú ale hlavne na to či súhrny obsahujú hlavné myšlienky pôvodného textu. Hodnotiť budem iba informatívne extrakty, pretože dĺžka indikatívnej formy súhrnu mi neposkytuje dostatočný priestor na hodnotenie.

Ešte pred samotným hodnotením by som chcel spomenúť, že pri hodnotení extraktov dátovej sady **WIKI13** som narazil na niekoľko viet, ktoré boli veľmi nezrozumiteľné. Je to dané tým, že tokenizer viet pre český jazyk nevedel korektne spracovať citačné a iné špeciálne značky, ktoré sa v dokumentoch tejto dátovej sady používajú. Tým sa dostali do výsledného súhrnu polovičaté vety, alebo len nič nehovoriace zhluky špeciálnych znakov. Takéto problémy mali len súhrny produkované sumarizátorom založeným na Edmundsonovej metóde. Je to spôsobené tým, že hodnotenia metódy sú z časti tvorené metrikami, ktoré neberú obsah viet do úvahy. Preto sa dostanú do súhrnu aj vety obsahovo nežiadúce. Pred sumarizáciou dokumentov je na to treba myslieť a buď vhodne nastaviť váhy pre jednotlivé Edmundsonove metódy alebo použiť inú, vhodnejšiu, sumarizačnú metódu.

Pre obe dátové sady boli subjektívne najhoršie, po obsahovej stránke i z hľadiska súvislosti textu, súhrny vytvorené Luhnovou sumarizačnou metódou. Súhrny sa líšili len málo od súhrnov vytvorených náhodným sumarizátorom a v niektorých prípadoch obsahovali aj vety bez informačnej hodnoty.

Porovnanie metódy LSA s Edmundsonovou metódou nie je jednoznačné, pretože kvalita súhrnu závisí na použitej dátovej sade. Pre dátovú sadu **ZDROJAK13** produkuje jednoznačne najkvalitnejšie súhrny Edmundsonova metóda. Toto nie je vôbec prekvapivé, pretože metóda sa ukázala ako najkvalitnejšia aj pri automatizovanom hodnotení. Avšak pre dátovú sadu **WIKI13** sa kvalita súhrnov vytvorených Edmundsonovou metódou značne zhoršila. Metóda LSA síce neprodukuje významne lepšie súhrny ako pre dátovú sadu **ZDROJAK13**, ale pretože Edmundsonova metóda nemá k dispozícii kvalitné metadáta o dokumente, sú pre dátovú sadu **WIKI13** výstupy sumarizátoru založeného na metóde LSA najkvalitnejšie z hodnotených.

8.4 Porovnanie s konkurenčnými implementáciami

Pri pokusoch o porovnanie kvality výstupov modulu *sumy* s konkurečnými aplikáciami som narazil na niekoľko prekážok. Jediný využiteľný program, hoci funkčný len na platforme Linux, bola aplikácia *Open Text Summarizer*. Tá však ako svoj vstup prijíma len neštruktúrovaný text a používa veľmi jednoduchú sumarizačnú techniku (hoci dokumentácia hovorí o metrike TF*IDF). Zvyšné aplikácie spomenuté v sekcii 6.1 sa mi nepodarilo úspešne použiť. Dôvodom bola hlavne nedostatočná dokumentácia vstupných súborov a konfigurácie. Hoci knižnica *Almus* obsahovala ukážkové súbory s konfiguráciou a vstupným dokumentom tak po spustení sumarizácie zamrzla. Keďže k aplikácii nie sú k dispozícii zdrojové kódy tak nie je jednoduché zistiť príčinu chyby.

Kapitola 9

Záver

V práci boli predstavené hlavné pojmy a princípy potrebné k návrhu modulu pre sumarizáciu textových dokumentov. Popis bol rozšírený o možnosti a problémy súvisiace so vstupným formátom HTML. Taktiež boli predstavené niektoré nástroje potrebné či už pre všeobecné spracovanie textu alebo úzko súvisiace s dolovaním na webe.

Preskúmaných bolo niekoľko sumarizačných metód a tri z nich boli hlbšie vysvetlené v kapitole 5. Tu spomínané metódy sú implementované ako súčasť modulu *sumy* pre jazyk Python. Implementovaný modul je šírený ako open-source knižnica dostupná na URL adrese https://github.com/miso-belica/sumy.

V záverečnej kapitole 8 sú uvedené poznatky z vyhodnotenia sumarizácií jednotlivých metód a ich vzájomné porovnanie. Z vykonaných experimentov je možné pozorovať, že najvhodnejšou metódou pre sumarizáciu bohato sémanticky značkovaných HTML dokumentov je Edmundsonova metóda. Pre texty bez alebo s minimálnym množstvom metadát o dokumente sa ako vhodnejšia varianta javí metóda založená na LSA.

Prínos práce spočíva v preskúmaní sumarizačných metód a algoritmov potrebných pri predspracovaní HTML dokumentov. Poznatky z vykonaného prieskumu sú využité pri implementácii voľne dostupného modulu, ktorý je schopný jednoduchým spôsobom sumarizovať ľubovoľný HTML dokument v českom jazyku. Pri pokusoch o porovnanie s inými implementáciami sa mi nepodarilo úspešne vyskúšať žiadnu podobnú aplikáciu, ktorá by disponovala modernou sumarizačnou metódou a korektne pracovala pre reálne HTML dokumenty.

Pri implementácii som mal najväčšie ťažkosti s dostupnosťou kvalitných nástrojov pre spracovanie prirodzeného českého jazyka. Kvôli tomu sa mi nakoniec nepodarilo implementovať *rezolúciu anaforických vzťahov*. Ako pokračovanie práce je preto vhodné najskôr implementovať kvalitné nástroje pre spracovanie prirodzeného českého jazyka. Ďalšie vhodné pokračovanie práce vidím v implementácii parserov vstupného textu pre formáty súborov používaných v konkurenčných sumarizátoroch pracujúcich s anglickým jazykom. Pridanie podpory pre anglický jazyk do implementovaného modulu by nemalo byť náročné, pretože s podporou pre ďalšie jazyky sa od začiatku návrhu modulu počítalo. Pridanie parserov umožní porovnávanie implementovaného modulu s aplikáciami tretích strán.

Literatura

[1] Bird, S., Klein, E. a Loper, E. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. 1. vyd. Beijing: O'Reilly Media, červenec 2009. Dostupné na: <http://www.nltk.org/book>. ISBN 978-0-596-51649-9.

[2] Das, D. a Martins, A. F. T. *A Survey on Automatic Text Summarization*. listopad 2007.

[3] Edmundson, H. P. New Methods in Automatic Extracting. *J. ACM*. Duben 1969, roč. 16, č. 2. S. 264–285. Dostupné na: <http://doi.acm.org/10.1145/321510.321519>. ISSN 0004-5411.

[4] Feldman, R. a Sanger, J. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data* [Hardcover]. Cambridge, MA, USA: Cambridge University Press, prosinec 2006. ISBN 978-0-521-83657-3.

[5] Gong, Y. a Liu, X. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA: ACM, 2001. S. 19–25. SIGIR '01. Dostupné na: <http://doi.acm.org/10.1145/383952.383955>. ISBN 978-1-58113-331-6.

[6] Ježek, K. a Steinberger, J. Sumarizace textů. In DATAKON. 2010.

[7] Křišťan, M. *Multidokumentový sumarizátor textů založený na latentní sémantické analýze*. Plzeň: Západočeská univerzita v Plzni, 2007. Diplomová práce.

[8] Kupiec, J., Pedersen, J. a Chen, F. A trainable document summarizer. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA: ACM, 1995. S. 68–73. SIGIR '95. Dostupné na: <http://doi.acm.org/10.1145/215206.215333>. ISBN 978-0-89791-714-6.

[9] Laender, A. H. F., Ribeiro Neto, B. A. et al. A brief survey of web data extraction tools. *SIGMOD Rec.* červen 2002, roč. 31, č. 2. S. 84–93. Dostupné na: <http://doi.acm.org/10.1145/565117.565137>. ISSN 0163-5808.

[10] Luhn, H. P. The automatic creation of literature abstracts. *IBM Journal Res. Dev.* Duben 1958, roč. 2, č. 2. S. 159–165. Dostupné na: <http://dx.doi.org/10.1147/rd.22.0159>. ISSN 0018-8646.

[11] Manning, C. D., Raghavan, P. a Schütze, H. *Introduction to Information Retrieval* [Hardcover]. 1. vyd. New York, NY, USA: Cambridge University Press, červenec 2008. Dostupné na: <http://nlp.stanford.edu/IR-book/>. ISBN 978-0-521-86571-9.

[12] Materna, J. *Sémantická analýza textů* [online]. srpen 2011, 14.2.2012 [cit. 2. ledna 2013]. Dostupné na: <http://fulltext.sblog.cz/category/vyzkum/>.

[13] Radev, D. R., Hovy, E. a McKeown, K. Introduction to the special issue on summarization. *Computational Linguistics*. Prosinec 2002, roč. 28, č. 4. S. 399–408. Dostupné na: <http://dx.doi.org/10.1162/089120102762671927>. ISSN 0891-2017.

[14] Steinberger, J. a Ježek, K. Using latent semantic analysis in text summarization and summary evaluation. In *In Proceedings ISIM '04*. 2004. S. 93–100.

[15] Steinberger, J. a Ježek, K. Hodnocení kvality sumarizátorů textů. In Západočeská univerzita v Plzni. *Znalosti 2005*. [b.m.]: VŠB - Technická univerzita Ostrava, únor 2005. S. 96–107. ISBN 978-80-248-0755-6.

[16] Steinberger, J. a Ježek, K. Evaluation Measures for Text Summarization. *Computing and Informatics*. 2009, roč. 28, č. 2. S. 251–275. ISSN 1335-9150.

Příloha A

Obsah CD

- `bin/` – Priečinok s binárnymi súbormi.
  - `python-2.7.5.msi` – Inštalátor interpretu jazyka Python verzie 2.7 pre 32-bitové systémy Windows.
  - `python-3.3.2.msi` – Inštalátor interpretu jazyka Python verzie 3.3 pre 32-bitové systémy Windows.
  - `numpy-MKL-1.7.1.win32-py2.7.exe` – Inštalátor knižnice *NumPy* pre jazyk Python verzie 2.7 a 32-bitové systémy Windows.
  - `numpy-MKL-1.7.1.win32-py3.3.exe` – Inštalátor knižnice *NumPy* pre jazyk Python verzie 3.3 a 32-bitové systémy Windows.
  - `scipy-0.12.0.win32-py2.7.exe` – Inštalátor knižnice *SciPy* pre jazyk Python verzie 2.7 a 32-bitové systémy Windows.
  - `scipy-0.12.0.win32-py3.3.exe` – Inštalátor knižnice *SciPy* pre jazyk Python verzie 3.3 a 32-bitové systémy Windows.
  - `lxml-3.2.1.win32-py2.7.exe` – Inštalátor knižnice *lxml* pre jazyk Python verzie 2.7 a 32-bitové systémy Windows.
  - `lxml-3.2.1.win32-py3.3.exe` – Inštalátor knižnice *lxml* pre jazyk Python verzie 3.3 a 32-bitové systémy Windows.
  - `nltk-2.0.4.win32-py2.7.exe` – Inštalátor knižnice *NLTK* pre jazyk Python verzie 2.7 a 32-bitové systémy Windows.
- `src/` – Priečinok so zdrojovými súbormi modulov *readability.py* a *sumy*.
- `experiments/` – Priečinok s dátovými sadami použitými pri experimentovaní so sumarizačnými algoritmami.
- `doc.pdf` – Text tejto práce v elektronickej podobe.
- `tex/` – Priečinok so zdrojovými súbormi textu tejto práce.