VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

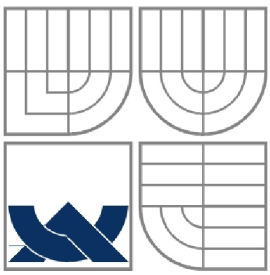
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SUMARIZACE DOKUMENTŮ NA WEBU

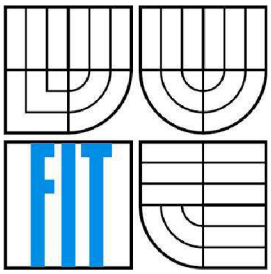
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. JÁN ŠKURLA
AUTHOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SUMARIZACE DOKUMENTŮ NA WEBU

SUMMARIZATION OF DOCUMENTS FROM THE WEB

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JÁN ŠKURLA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VLADIMÍR BARTÍK, Ph.D.

BRNO 2012

Abstrakt

Tématem této diplomové práce je sumarizace dokumentu na webu. Nejdříve se věnuje problematice získávání textu z webu pomocí wrapperu. Je zde uveden přehled jednotlivých wrapperu použitých pro inspiraci k budoucí implementaci. Práce také obsahuje jednotlivé metody tvorby souhrnu (Luhnova, Edmundsonova a KPC) z textových dat. Součástí práce je návrh a implementace aplikace na extrakci textových dat s následnou tvorbou souhrnu. Aplikace je postavena na platformě Java s využitím grafické knihovny Swing.

Abstract

Topic of this master's thesis is a summarization of the documents on the web. First, it deals with the issues of acquiring text from the web using wrapper. An overview of wrappers used as an inspiration for the future implementation is stated. This paper also includes various methods for creating summary (Luhn`s, Edmundson`s and KPC) from the text data. Application design for the text data extraction and summarization is also part of this paper. Application is based on Java platform and Swing graphic library.

Klíčová slova

Extrakce textu z WWW, HTML wrapper, sumarizace textu, Edmundson, Luhn, Kupiec, KPC, Java

Keywords

Text mining from web, HTML wrapper, text summarization, Edmundson, Luhn, Kupiec, KPC, Java

Citace

Ján Škurla: Sumarizace dokumentů na webu, diplomová práce, Brno, FIT VUT v Brně, 2012

Sumarizace dokumentů na webu

Prohlášení

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

.....
Ján Škurla
21.5.2012

Poděkování

Chtěl bych poděkovat vedoucímu své diplomové práce, panu Ing. Vladimíru Bartíkovi, Ph.D., za odbornou pomoc, připomínky a trpělivost při tvorbě tohoto projektu.

Zadání diplomové práce/13382/2011/xskurl00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2011/2012

Zadání diplomové práce

Řešitel: **Škurla Ján, Bc.**
Obor: Informační systémy
Téma: **Sumarizace dokumentů na webu**
Summarization of Documents from the Web
Kategorie: Data mining

Pokyny:

1. Seznamte se s problematikou dolování textu, zaměřte se přitom na algoritmy sumarizace textu.
2. Prostudujte možnosti využití metod sumarizace pro webové sránky. Po dohodě s vedoucím zvolte algoritmus sumarizace, který bude k tomuto účelu vhodný.
3. Navrhněte koncepci programu, který bude provádět sumarizaci webových dokumentů.
4. Navržený program implementujte.
5. Funkčnost programu ověřte na vhodném vzorku dat a výsledky metod porovnejte.
6. Zhodnoťte dosažené výsledky a další možná rozšíření projektu.

Literatura:

- Das, D., Martins, A.: A Survey on Automatic Text Summarization. Carnegie Mellon University, 2007.
- Feldman, R., Sanger, J.: The Text Mining Handbook. Cambridge University Press, 2007.

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1-3.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Bartík Vladimír, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 19. září 2011
Datum odevzdání: 23. května 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Obsah

1 Úvod

1.1 Textové dáta a sumarizácia

Sumarizácia textu je metódou dolovania dát, presnejšie dolovania textu pomocou vytvorenia súhrnu prostredníctvom extrakcie viet z pôvodného textu, ktorá nám umožňuje zoznámiť sa so základnou myšlienkou dokumentu bez toho, aby sme ho prečítali. Potrebujeme si vedieť vybrať dopredu text, ktorý sa nám naozaj oplatí čítať, ktorý nám prinesie informácie, ktoré sme hľadali. Masívne sprístupnenie internetu celosvetovej populácii totiž spôsobilo, že k užívateľovi sa dostane, pomerne ľahkým spôsobom obrovské množstvo informácii. V prevažnej väčšine sú ich zdrojom webové stránky. Tento stav môžeme považovať za zaplavenie informáciami. Preto musíme riešiť naliehavý problém ako sa v tomto mori nových textov nestratiť a vybrať si iba tie najdôležitejšie a najcennejšie dokumenty pre nás. Je prirodzené keď sa inšpirujeme v reálnom svete a zoberieme si príklad v rokmi overených praktikách. Vyberanie toho správneho dokumentu na internete by sme mohli prirovnať vyberaniu knihy. Neprídeme do kníhkupectva, a nekúpime všetky knihy, ale vyberieme si tú, ktorá nás zaujala, alebo ju aktuálne potrebujeme, alebo jej niektorú kapitolu. Vyberáme si podľa názvu, obsahu a v neposlednom rade podľa grafickej úpravy. Rovnaký prístup využijeme aj pri internetových stránkach, kde umožníme používateľovi zoznámenie s obsahom dokumentu pomocou súhrnu. Na základe tohto súhrnu sa môže užívateľ rozhodnúť, či vybraná stránka vyhovuje jeho kritériám, a či mu prinesie požadované informácie, vo veľmi krátkom čase.

1.2 Analýza požiadaviek

Hlavným cieľom tejto diplomovej práce je vytvorenie aplikácie využívajúcej vybranné metódy sumarizácie na tvorbu súhrnu pre webové stránky. Po dohode s vedúcim tejto diplomovej práce boli zvolené základné sumarizačné metódy, ktoré budú podrobnejšie vysvetlené v kapitole 5. Rovnako bol vybraný aj programovací jazyk, v ktorom daná aplikácia umožňujúca sumarizáciu bude implementovaná. Vytváranie súhrnov z webových stránok, ktoré bude predstavovať približne 20% pôvodného rozsahu stránky v textovej podobe. Aby sme sa dostali ku žiadanému výsledku, môžme prácu rozčleniť do 2 logických častí, kde prvá časť predstavuje extrakciu textových dát zo zvolenej webovej stránky a v druhej časti vytvoríme z týchto dát súhrn pomocou vybraných metód.

1.3 Rozčlenenie práce

Diplomová práca je rozčlenená s výnimkou úvodu a záveru do 7 kapitol, ktoré sú pre väčšiu prehľadnosť ďalej členené na jednotlivé podkapitoly. Na začiatku práce, kapitola 2 obsahuje základný popis protokolov a technológií použitých pri získavaní textu z webu. Kapitola 3 stručné spracováva problematiku dolovania dat, a to hlavne dolovania v texte a na webe. Následne sú spomenuté možné metódy použité pri dolovaní textu z webových stránok. Kapitola 4 obsahuje krátke zoznámenie s metódami sumarizácie, ich rozdelením a vzájomným prekrytím jednotlivých metód. Taktiež sú v tejto kapitole vysvetlené jednotlivé pojmy, ktoré sa budú využívať v nasledujúcom texte. Kapitola 5 oboznamuje s vybranými sumarizačnými metódami, ktoré sú použité v tejto diplomovej práci. Po dohode s vedúcim diplomovej práce boli vybrané jednoduché sumarizačné metódy založené na štatistickej a heuristickej analýze. Pričom pri každej metóde je podrobne na príklade vysvetlený spôsob činnosti tejto metódy. Kapitola 6 sa zaoberá analýzou a návrhom programovej časti tejto

diplomovej práce. Nachádza sa tu rozdelenie aplikácie na jednotlivé programy a ďalšie menšie časti, spoločne s ich vzájomnou komunikáciou, ktorá je reprezentovaná vhodne zvolenými modelmi UML. Kapitola 7 reprezentuje implementáciu sumarizačnej aplikácie s popisom riešení zvolených pri jej vytváraní. Kapitola 8 je venovaná testovaniu aplikácie pri použití jednotlivých sumarizačných metód a zhodnotenie ich možného využitia na vybraný druh internetovej stránky. Záver práce nachádzajúci sa v kapitole 9 obsahuje zhrnutie dosiahnutých výsledkov a ich možné vylepšenie pri ďalšom rozšírení práce.

1.4 Náväznosť na predchádzajúcu prácu

Táto práca nadväzuje na semestrálny projekt, ktorý bol použitý ako základ pre teoretickú časť diplomovej práce. Kapitoly 2, 3, 4 a 5 vychádzajú zo semestrálneho projektu, ale boli v priebehu vytvárania programovej časti diplomovej práce obohatené o niekoľko poznatkov, ktoré slúžia k lepšiemu pochopeniu danej problematiky. Kapitoly pojednávajúce o návrhu a implantácii aplikácie boli vypracované odznova, pričom bola využitá iba malá časť vytvorená v rámci semestrálneho projektu.

2 Základné protokoly a jazyky na WWW

Keďže táto práca je založená na extrakcii a následnej sumarizácii textu z webu, takže určitým spôsobom spracováva webové stránky, je dôležité stručne popísať technológie, ktoré sa pri tom využívajú [1].

2.1 HTTP

Hypertext transfer protocol (HTTP) je univerzálny textový protokol na prenos súborov, pracujúci na aplikačnej vrstve referenčného OSI modelu, založený na komunikácii medzi servermi a klientmi služby WWW. Bol navrhnutý organizáciou W3C (World Wide Web Consortium) a skupinou IETF (Internet Engineering Task Force), pričom aktuálna verzia HTTP/1.1 je publikovaná ako RFC dokument 2616 [2].

HTTP definuje požiadavky klienta (user agent) a odpovede servera. Klient začne požiadavku nadviazaním spojenia na určitý port vzdialeného zariadenia (serveru) a server mu pošle odpoveď. Spojenie musí zaručovať spoľahlivý prenos, pričom väčšinou ide o protokol TCP, využívajúci port 80. Keďže server neuchováva informácie o predchádzajúcich požiadavkách, jedná sa o bezstavový protokol, čo je v určitých prípadoch nežiaduce, preto sa zavádzajú náhradné riešenia (cookies, parameter v URI atď.).

2.1.1 Adresovanie dokumentov na WWW pomocou URI

URI (Universal Resource Identifier) je zhrnujúcim pomenovaním pre URL (Universal Resource Locator) a URN (Universal Resource Name). URL špecifikuje miesto v sieti, kde sa nachádzajú dáta a je teda závislé na umiestení. Na druhú stranu URN poskytuje unikátny názov pre dáta a teda je nezávislý na umiestení v sieti.

Keďže v tejto práci je najdôležitejšie umiestenie dokumentu s textom, tak sa pod pojmom URI chápe URL a URN sa nebude vôbec využívať.

Formát URL adresy

Keďže táto práca používa na získanie textu URL adresy, bolo by vhodné si predstaviť, z čoho sa taká URL adresa skladá:

`<protokol>://<adresa_serveru>[:<port>]/<path>[?<dotaz>]`

- protokol – použitý spôsob prístupu k zdroju (http, ftp …).
- adresa_serveru – kanonické meno (fit.vutbr.cz, localhost), alebo IP adresa.
- port – presné číslo portu, pokiaľ nie je vyplnené tak sa používa typická hodnota (napr. http má 80).
- path – cesta k dokumentu.
- dotaz – údaje odosielané serveru.

Príklad URL adresy:

`http://www.fit.vutbr.cz:80/index.php`

2.1.2 HTTP požiadavka

Každá požiadavka má predpísaný základný formát:

```
<metóda><URI><verzia protokolu>
<hlavičky>
<prazdny riadok>
<telo žiadosti>
```

Vybrané metódy

Protokol HTTP má niekoľko základných metód. Iba dve z nich – GET a HEAD musí obsahovať každá implementácia a spoločne s POST patria k metódam použitým v tejto práci.

- **Metoda GET:** Príkaz načíta zdrojové dáta zo serveru a odošle ich klientovi. Syntax príkazu vyzerá nasledovne:
  `GET url HTTP/1.0`
- **Metoda HEAD:** Príkaz nevracia súbor požadovaný URL adresou, ale iba metainformácie o tejto adrese. Využíva sa pri testovaní dostupnosti URL adresy.
- **Metoda POST:** Slúži na zasielanie údajov z klienta na server, pričom dĺžka dát nie je ničím obmedzená.

2.1.3 HTTP odpoveď

```
<verzia protokolu><stavový kód><popis>
<hlavičky>
<prazdny riadok>
<telo odpovede>
```

Vybrané stavové kódy:
Popisujú výsledok spracovania požiadavky od klienta, pričom podporujú vlastnú definíciu chybových oznamov. Definíciu všetkých stavových kódov je možné nájsť na [2]. Stavové kódy sa delia na päť základných skupín, pričom pri každej sú uvedené príklady :

- **1XX Informačné**
  `100 Continue`
- **2XX Úspech**
  `200 Ok`
- **3XX Presmerovanie**
  `301 Moved Permanently`
  `304 Not Modified`
- **4XX Chyba klienta**
  `403 Forbidden`
  `404 Not Found`
- **5XX Chyba servera**
  `500 Internal Server Error`

2.2 Jazyky použité pri tvorbe www stránok

Pretože podstatnou časťou tejto práce je získavať textové dáta z webových stránok, je vhodné si popísať aspoň tie najzákladnejšie jazyky, ktoré sa využívajú na ich tvorbu. Väčšina z nich vychádza zo značkovacieho jazyka SGML [3].

2.2.1 Značkovacie jazyky

Tieto jazyky predstavujú nástroj pre vkladanie dodatočných informácií do normálneho textu. Pričom základná myšlienka vychádza z prípravy rukopisu do tlače vyznačovaním budúceho vzhľadu jednotlivých častí textu ako napríklad veľkosť a druh písma. Na rozdiel od programovacích jazykov, kde prevláda štruktúrovaný text, v značkovacích jazykoch je najviac zastúpený obecný text, v ktorom je štruktúrovaný text umiestený pomocou značiek. Značky sú od textu oddelené pomocou uhlových zátvoriek napr.:

```
<BIG>veľký</BIG>
```

Existuje veľké množstvo značkovacích jazykov ako je TEX, RTF (Rich Text Format), PS (PostScript) alebo PDF (Portable Document Format), ktoré pracujú hlavne s textom. Univerzálnejšími jazykmi je rodina SGML (Standard Generalized Markup Language), kam patrí HTML (Hypertext Markup Language), ktorý sa využíva hlavne pri tvorbe www stránok.

2.2.2 SGML

SGML (Standard Generalized Markup Language) štandardizovaný normou ISO 8879:1996, je metajazyk určený na popis značkovacích jazykov. Definuje základné syntaktické vlastnosti značkovacích jazykov a to hlavne:

- **spôsob zápisu značiek**
  Značky sú označené menom a zapisované v uhlových zátvorkách. Značka s menom „BIG“ sa zapíše ako:

  ```
  <BIG>
  ```

  SGML pripúšťa dva druhy značiek – párové a nepárové:

  ```
  <parova> Text medzi parovymi značkami.</parova> <neparova>
  ```

  Navyše SGML pripúšťa neobmedzené vnorovanie značiek
  ```
  <znacka><vnorena>Text vo vnorenej značke.</vnorena> <znacka/>
  ```

- **spôsob zápisu atribútov**
  Každá značka môže obsahovať ľubovoľný počet atribútov, pričom každý z nich je označený menom a môže mať nejakú hodnotu:

  ```
  <prvy atribut = ”prvy” druhy atribut = ”2”>
  ```

- **znakové entity**
  Spôsob zápisu špeciálnych symbolov, ktoré nejde napísať normálnym spôsobom.

2.2.3 XML

Keďže univerzálnosť a komplikovanosť SGML sa stali najväčšou prekážkou nasadenia SGML do reálneho používania, tak bol vytvorený jazyk XML (eXtensible Markup Language). XML vychádza z rovnakého princípu, ale je podstatne zjednodušený (odstránenie nepárových značiek, jediná možnosť zápisu syntaxe ...). XML sa využíva nielen pri tvorbe dokumentov, ale aj pri ukladaní dát, komunikácii a v ďalších aplikáciach.

Rovnako ako pri SGML, tiež pri XML sa musia definovať mená značiek a ďalšie informácie pre vytvorenie značkovacieho jazyka, ktorý by bol založený na XML.

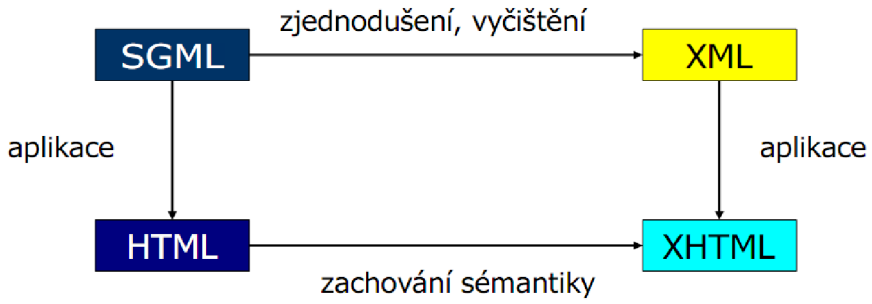
2.2.4 HTML

Predstavuje najviac využívanú aplikáciu SGML, ktorá je určená na tvorbu hypertextových dokumentov na webe. HTML je medzinárodným štandardom, pričom špecifikáciu jazyka má na starosti W3C (World Wide Web Consortium). Jedná sa o textový súbor označený príponou .html, alebo .htm. HTML dokument sa delí na definíciu DTD, hlavičku a telo dokumentu. Pre lepší prehľad a použitie tagov je možno použiť [4], alebo podrobne [5].

2.2.5 XHTML

Je podobne ako HTML najviac využívaná aplikácia na tvorbu dokumentov na www, ale na rozdiel od nej je postavená na XML. Z toho plynie, že sú dovolené tagy, ktoré sa nachádzajú v pároch, čím vzniká na základe prísnejších pravidiel XML čistejšia náhrada za HTML. Viac pre záujemcov o XHTML sa nachádza v [6], alebo v podrobne popísanej špecifikácii [7].

Súvislosť medzi SGML, XML, HTML a XHTML reprezentuje nasledujúci obrázok 2.1 .



Obrázok 2.1 značkovacie jazyky z rodiny SGML [3]

3 Dolovanie dát

Dolovanie dát (data mining) podľa literatúry [8] je netriviálny proces zisťovania platných, neznámych, potencionálne užitočných a ľahko pochopiteľných informácií z dát. V rámci tejto diplomovej práce sú z tejto oblasti dôležité dve časti: dolovanie v texte (text mining) a dolovanie na webe (web mining).

3.1 Dolovanie v texte

Dolovanie v texte podľa literatúry [9] patrí medzi špeciálnu metódu pre dolovanie dát. Pri dolovaní máme zväčša k dispozícii štrukturované dáta, s ktorými následne pracujeme, napr. uložené v databázových systémoch vo forme tabuliek. Pri dolovaní v texte, ktorý sa nachádza v textových dokumentoch (e-maily, HTML stránky, súbory vo formáte PDF atď.) sú typicky v podobe prirodzeného jazyka v neštrukturovanej či semištrukturovanej podobe.

Tieto dokumenty obsahujú veľké množstvo informácií uložených v presne danej šablóne, alebo formáte. Pre dolovanie je dôležitá čo najväčšia kolekcia dokumentov, pričom každý dokument môže obsahovať obrovské množstvo viet, fráz, slov a typografických elementov.

Medzi základné úlohy dolovania v texte patria: klasifikácia dokumentov, vyhľadávanie textov a informácií, či v prípade tejto diplomovej práce sumarizácia textov. Tieto metódy pre získavanie znalostí z textu svoje použitie nájdu rovnako ako v teoretickom výskume, taktiež v reálnom svete pre získanie nových informácií z pôvodných dát.

3.2 Dolovanie na webe

Dolovanie na webe je v mnohom podobné s dolovaním v texte, avšak web predstavuje oveľa rozsiahlejší zdroj informácií pre dolovanie, keďže z webovej stránky sa dá získať nielen text ale aj iné podstatné informácie napr. o štruktúre odkazov medzi stránkami, o použití stránky atď.. Pričom získavanie textu z internetových stránok je v súčasnej dobe veľmi zložité, pretože stránky neobsahujú len textové informácie k danej téme, ale nachádzajú sa na nich aj iné, pre čitateľa menej dôležité údaje, čo predstavuje veľký problém pri automatickej extrakcii.

Štruktúra týchto stránok je rok od roku zložitejšia a predstavuje ďalší významný problém. Preto sa nedá využiť jednoduché skopírovanie textu zo stránky, ako sa to robilo v minulosti, ale musíme využiť zložitejšie techniky na transformáciu z HTML na XML, alebo na čistý text. Táto technika je známa ako extrakcia informácii z webu alebo ako HTML wrapping.

3.2.1 HTML Wrapper

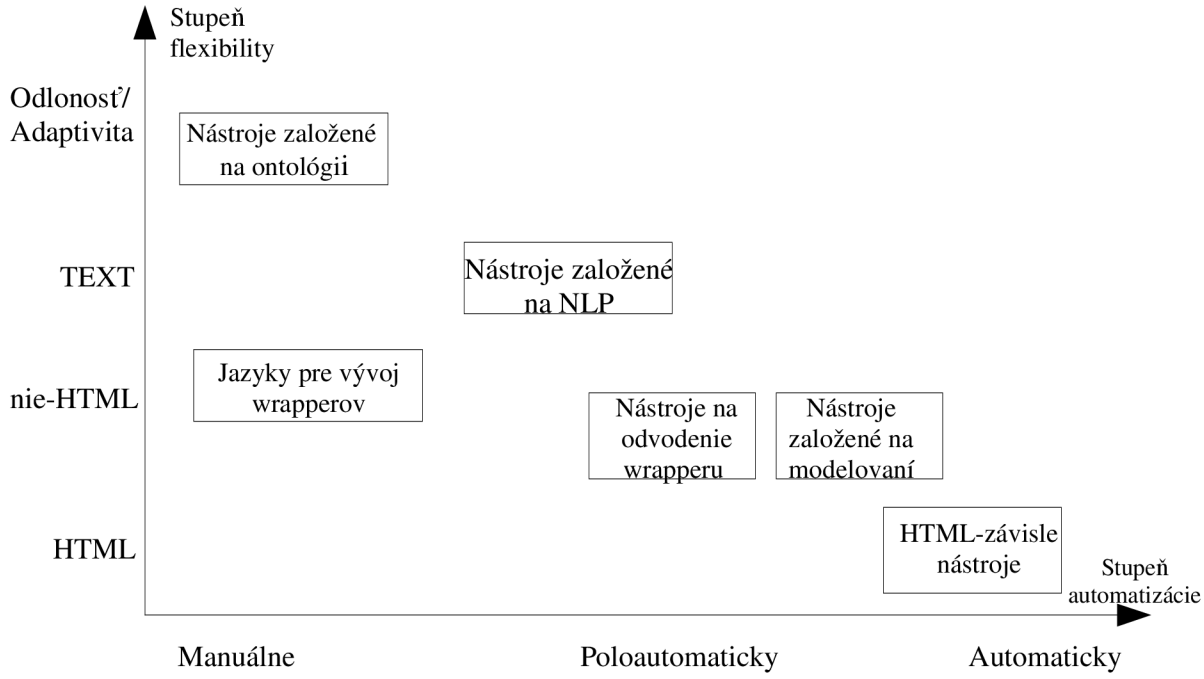
Wrapper identifikuje zaujímavé časti dokumentu a sprístupní ich v použiteľnom formáte pre následné strojové spracovanie, v našom prípade vytvorenie súhrnu. Podľa literatúry [10] a [11], z ktorej bude táto časť vychádzať, existuje viacero typov wrapperov, ktoré môžeme rozdeliť do nasledujúcich skupín:

- **Jazyky pre vývoj wrapperov :** Jedna z prvých iniciatív pre riešenie problému tvorenia wrapperov bol vývoj jazykov, špeciálne určených na pomoc používateľom pri stavbe

wrapperov. Tieto jazyky boli navrhnuté ako alternatívy k univerzálnym jazykom, ako Perl a Java, ktoré boli dovtedy najviac využívané pre túto úlohu.

- **HTML-závisle nástroje :** Skupina nástrojov, ktoré využívajú na extrakciu dát štrukurálných vlastností HTML dokumentov. Pred samostatným procesom extrakcie tieto nástroje vytvoria stromovú reprezentáciu (strom) štruktúry dokumentu, ktorá odráža jeho HTML značkovaciu (tag) hierarchiu. Pravidlá na extrakciu sú generované poloautomaticky alebo automaticky a postupne aplikované na predtým vytvorený strom.
- **Nástroje založené na NLP :** Techniky spracovania prirodzeného jazyka (NLP) boli využívané niekoľkými nástrojmi, aby sa naučili extrakčné pravidlá pre dolovanie relevantných informácií, ktoré existujú v dokumentoch v podobe prirodzeného jazyka. Tieto nástroje zvyčajne využívajú techniky, ako je filtrovanie alebo lexikálne sémantické značkovanie, na vybudovanie vzťahov medzi frázami a vetnými členmi, pričom z týchto vzťahov následne odvodzujú extrakčné pravidlá. Tieto pravidlá sú založené na syntaktických a sémantických obmedzeniach, ktoré pomáhajú identifikovať relevantné informácie v dokumente. Nástroje založené na NPL sú zvyčajne vhodnejšie pre webové stránky, ktoré sa skladajú z gramatického textu.
- **Nástroje na odvodenie wrapperu :** Nástroje na odvodenie wrapperu generujú extrakčné pravidlá založené na oddeľovačoch, odvodených z danej sady testovacích príkladov. Hlavný rozdiel medzi týmito nástrojmi a nástrojmi založenými na NLP je, že tieto nie sú závislé na jazykových obmedzeniach, ale skôr na formátovaní, ktoré implicitne vymedzuje štruktúru kusov nájdených dát. Preto sú tieto nástroje vhodnejšie pre HTML dokumenty ako predchádzajúca skupina.
- **Nástroje založené na modelovaní** : Táto kategória zahŕňa nástroje, ktorým keď je daná štruktúra obsahujúca objekty záujmu, tak sa snaží na weboch nájsť časti dát, ktoré implicitne zodpovedajú danej štruktúre. Daná štruktúra je poskytnutá na základe sady modelovacích primitív (napr. n-tíc, zoznamov atď.), ktoré sa prispôsobia podriadenému dátovému modelu. Následne, algoritmy podobné tým, ktoré používajú nástroje na odvodenie wrapperu, identifikujú objekty s príslušnou štruktúrou na daných stránkach.
- **Nástroje založené na ontológii :** Predchádzajúce nástroje sa spoliehali výlučne na štrukturálne vlastnosti dát v dokumente, na základe ktorých vytvorili pravidlá, alebo vzory na vykonanie extrakcie. Existuje ale aj iný prístup k extrakcii a to na základe dát. Ak je poskytnutá špecifická doménová aplikácia, ontológia môže byť použitá pre nájdenie konštánt na stránke a následne pre vytvorenie objektov z týchto konštánt.

Ako vidíme tak predchádzajúce metódy poskytujú odlišnú mieru automatizácie a taktiež sú rozdielne ich možnosti pri spracovaní webových stránok, čo dokumentuje nasledujúci obrázok 3.1 .



Obrázok 3.1 Klasifikácia používajúca stupeň flexibility a automatizácie na HTML wrappery z [11]

Vyberieme si najvhodnejšiu skupinu a niektoré nástroje z tejto skupiny následne použijeme, alebo ich využijeme ako predlohu pri vytváraní vlastnej aplikácie, pri implementácii aplikácie na extrakciu textu. Z obrázku vyplýva, že najvhodnejšou skupinou pre túto diplomovú prácu budú nástroje HTML-aware, preto by bolo vhodné si popísať nástroje z tejto skupiny.

3.2.2 HTML-aware nástroje

Existuje viacero týchto nástrojov, ktoré sa medzi sebou líšia licenciou, použitím ale aj komplexnosťou. Náš výber bude na základe použitého programovacieho jazyka a licencie. Zameriame sa na bezplatné nástroje naprogramované v jazyku Java [10].

Roadrunner - Projektom škôl Universita di Roma Tre a Universita della Basilicata. Skúma techniky na extrakciu dát z HTML stránok pomocou automaticky generovaných wrapperov. Tento nástroj generuje wrapper na analýzy podobnosti a zložitosti na základe niekoľkých vzoriek z rovnakej triedy.

XWRAP Elite - Nástroj založený na XML, ktorý dokáže automaticky generovať wrappery pre weby. Tento nástroj sa skladá z troch častí: extrakcia objektov a elementov, získanie pomocou rozhrania filtru a generovanie kódu. Wrappery vytvorené pomocou XWRAP Elite majú dve odlišné funkcie:

- **Dátový wrapper –** mení HTML do XML s popisom štruktúry.
- **Funkcionálny wrapper** – filtruje webové stránky a poskytuje filtrované výsledky v XML.

WebHarvest - Ponúka spôsob, ako získavať užitočné dáta z webových stránok. Využíva k tomu používané techniky a technológie pre manipuláciu s textom a XML, ako je XSLT, XQuery a použitie regulárnych výrazov. Web-Harvest sa zameriava predovšetkým na webové stránky, ktoré sú založené na HTML alebo XML. Taktiež umožňuje pridávať Java knižnice pre zlepšenie schopností extrakcie.

4 Sumarizácia

Sumarizáciu je možné neformálne definovať [12] ako redukčnú transformáciu zdrojového textu na výsledný súhrn, pomocou redukcie (skrátenia) obsahu zovšeobecnením a (alebo) výberom toho, čo je v zdrojovom texte dôležité. Pritom existujú situácie, v ktorých nie je možné sumarizáciu vhodne využiť.

Proces sumarizácie môžeme rozdeliť na tri základné časti:

- **interpretácia textu** - zo zdrojového textu sa interpretuje textová reprezentácia.
- **transformácia textu** - textová reprezentácia sa transformuje na reprezentáciu súhrnu.
- **generovanie súhrnu** - vygenerovanie výsledného súhrnu z reprezentácie súhrnu.

4.1 Typy sumarizácie

Sumarizácia sa dá rozdeliť na viacero základných typov podľa toho, čo je vstupom (zdrojovým textom) a výstupom sumarizácie (súhrnom) [13].

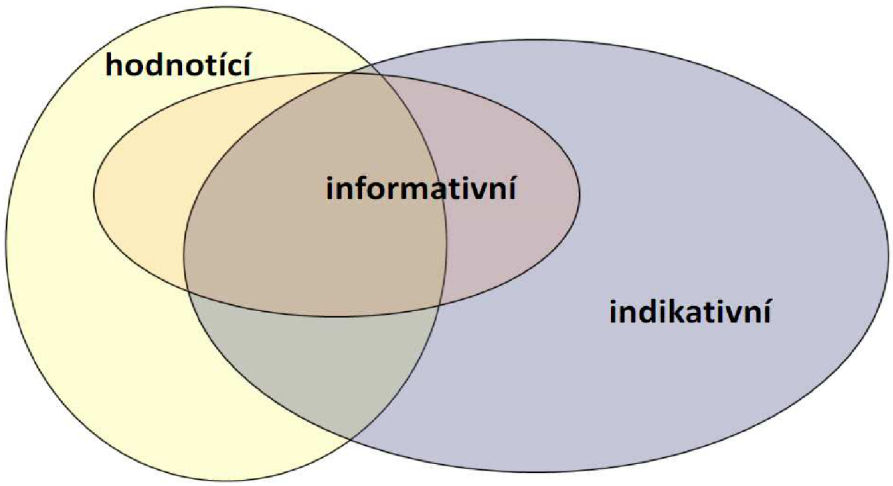
Podľa formy výsledného súhrnu:

- **extrakt:** Základným princípom pri jeho tvorbe je, že „čo vidíte, to aj dostanete“, čo predstavuje, že zdrojový text je iba mechanicky, bez akejkoľvek logickej spojitosti, transformovaný na súhrn. Využívajú sa hlavne štatistické a heuristické metódy na získavanie sekvencií slov, ktoré sa nachádzajú v pôvodnom texte. Pričom sa jedná o jednotlivé časti vety, celé vety, ale takisto sem môžu patriť aj celé odseky pôvodného textu. Takto vytvorený extrakt sa prejavuje hlavne slabšou logickou súvislosťou medzi jednotlivými časťami vo výslednom texte, čo je spôsobené výberom slov a viet, bez ohľadu na kontext v pôvodnom zdrojovom texte.
- **abstrakt:** Podľa literatúry ho taktiež môžeme nazvať ako extrakcia faktov. Hlavnou myšlienkou je nájsť logické spojitosti v zdrojovom texte. Jedná sa o súhrn, ktorý zväčša neobsahuje sekvencie slov z pôvodného textu. Výsledný text zachováva najpodstatnejšie informácie a logické súvislosti zo zdrojového textu, avšak jedná sa o veľmi zložitú úlohu, ktorá je na naprogramovanie zatiaľ veľmi náročná.

Podľa účelu :

- **indikatívny :** Jediným cieľom indikatívneho súhrnu je uľahčiť čitateľovi rýchle rozhodnutie o tom, či má čítať celý text. Prinášajú skrátenou formou podstatné informácie o hlavných témach v dokumente pričom zachovávajú jeho najpodstatnejšie časti. Ich dĺžka zväčša nepredstavuje viac ako desatinu dĺžky pôvodného textu.
- **informatívny :** Taktiež slúži na časovú úsporu pre čitateľa, pretože poskytuje rýchle zoznámenie sa s danou témou. Tvorí ho približne 25 až 33% pôvodného textu, z ktorého je vytvorený obsah so zachovanými vybranými detailmi.
- **kritický** : Vyhodnocuje zdrojové texty a v nich obsiahnuté témy podľa dôležitosti pre tvorcu sumarizácie. Ide o subjektívne videnie danej problematiky.

Určitá najpodstatnejšia časť výsledného súhrnu, sa nachádza vo všetkých súhrnoch podľa účelu, čo reprezentuje nasledujúci obrázok.



Obrázok 4.1 Vzťah medzi typmi súhrnu [13].

Podľa použitého princípu:

- **heuristicke metódy :** Luhn, Edmundson
- **štatistické metódy** : KPC
- **grafové metódy** : TextRank, LexRank
- **algebraické metódy** : LSA (Latentná sémantická analýza), NMF (Non-Negative Matrix Factorization)

Podľa zamerania :

- **obecné** : Sú určené pre čo najväčší počet užívateľov s diametrálne odlišnými záujmami. Všetky témy nachádzajúce sa v zdrojovom dokumente sú pre čitateľov rovnako dôležité.
- **zamerané na čitateľa :** Presne zodpovedajú potrebám a záujmom jednotlivých skupín čitateľov, ale aj jednotlivcov.

Takisto existuje ešte veľa ďalších kritérií, na základe ktorých môžeme rozdeliť typy sumarizácie, ako sú napr. jazyk (jedno-jazykové, viac-jazykové), počet dokumentov (jeden dokument, viacero dokumentov), médium (text,audio,video...), úroveň(povrchné, hlbšie) atď..

4.2 Základné pojmy

Slúži k vysvetleniu jednotlivých pojmov, ktoré sa budú vyskytovať v nasledujúcich kapitolách. Rozdelenie týchto pojmov je podľa toho v akej fáze úpravy slov sa používajú. Pri každom pojme sa nachádza jeho anglický názov a čo najpresnejšie vysvetlenie tohto pojmu.

4.2.1 Predspracovanie slov

- **Stemming** [10] - technika na vytvorenie základného tvaru, ktorá pre časované, skloňované, alebo inak upravené slovo pomocou lexikografických pravidiel daného jazyka vráti koreň slova. Môžeme ho aj inak nazvať ako normalizovanie slova. Pri stemmingu sa odstránia

všetky morfologické koncovky, pripadne pokiaľ je to nutné taktiež aj predpony. Stemming je úzko spätý s jazykom, pretože základom tejto metódy je zoznam koncoviek a spôsob ich odstránenia.

- **Stemmer** [10] – algoritmus ktorý, prevádza stemming. Pre anglický jazyk existuje viacero typov stemmerov napr. Porterov, Lovinsonov, alebo Paice/Husk stemmer, pričom každý z nich využíva inú metodiku pri vytváraní koreňa slov. Je možné tieto stemmeri využiť aj na iné jazyky, ale výsledok závisý od ich podobnosti s anglickým jazykom.

4.2.2 Ohodnotenie slov

- **Stop words** [9] - stop slová, sú slová ktoré majú v danom jazyku vysoký výskyt, zväčša majú iba syntaktický význam, čiže neobsahujú žiadnu významnú informáciu. Medzi stop slová zaraďujeme napr. spojky, predložky atď..
- **Bonus words** [16] **-** slová, ktoré majú pozitivný vplyv pri výbere daného slova, alebo vety obsahujúcej tieto slová do výsledného súhrnu. Do tejto skupiny slov patria hlavne tieto slová: druhý a tretí stupeň prídavných mien (komparatív a superlatív), príslovky vyjadrujúce úsudok, dôležité slová pre daný text (profit, benefit atd.), opytovacie slová (who, what atď.) a slová vyjadrujúce kauzalitu.
- **Stigma words** [16] - majú negativný vplyv pri výbere slova a vety, v ktorej sa tieto slová nachádzajú do výsledného súhrnu. Patria sem slovné spojenia popisujúce zľahčovanie danej témy, alebo odkaz na inú časť textu. Ďalej sa tu nachádzajú slovné spojenia opisujúce nedôležité detaily, slová obiehajúce od danej témy.
- **Null words** [16] - rovnaká skupia slov ako Stop words reprezentujúca slová nemajúce žiadnu informačnú hodnotu, ktorá bola použitá v Luhnovej metóde.

5 Vybrané sumarizačné metódy

V tejto kapitole sú teoreticky spracované sumarizačné metódy, ktoré budú implementované v programovej časti tejto diplomovej práce. Prvé dve metódy[15,16] predstavujú ranú fázu sumarizácie, čiže sa jedná o klasické sumarizačné metódy pre jeden dokument, ktoré sú založené na heuristických metódach a posledná KPC metóda je založená na štatistických metódach.

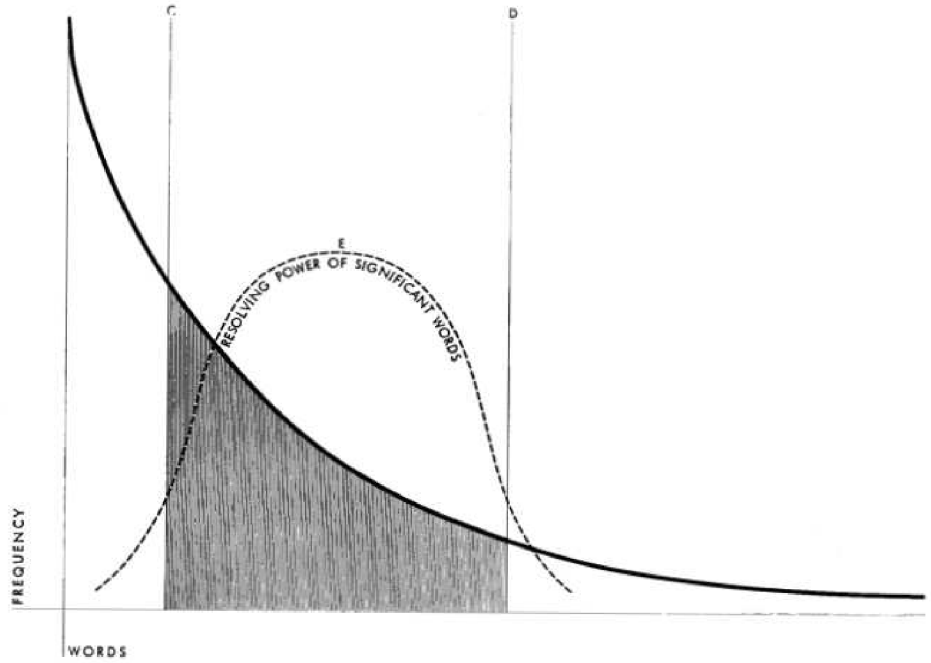
5.1 Luhnova metóda

Predstavuje najstaršiu a najjednoduchšiu metódu (1958) automatickej sumarizácie textu [15], ktorá je založená na heuristickej analýze spracovaného textu, kde výsledkom je jednoduchý extrakt z viet nachádzajúcich sa v texte. Vychádza z myšlienky, že slová, ktoré sa najčastejšie vyskytujú v texte, sú obsahovo dôležité. Čiže čím viac vysokofrekvenčných slov sa vyskytuje vo vete, tím viacej informácií veta obsahuje a je vhodné ju zahrnúť do výsledného súhrnu.

5.1.1 Popis fungovania metódy

Metódu môžeme rozdeliť na niekoľko základných fáz:

1. Filtrovanie jednotlivých slov na základe zoznamu stop slov.
2. Stemming slov, ktorý vráti normalizovaný tvar tohto slova.
3. Vypočítanie frekvencie výskytu normalizovaných slov.
4. Odstránenie málo sa vyskytujúcich slov. Čiže zostali iba dôležité slová, čo nám znázorňuje nasledujúci obrázok 5.1.



Obrázok 5.1 Závislosť dôležitosti slov od ich frekvencie výskytu [15].

5.Ohodnotenie jednotlivých viet na základe dôležitosti slov v týchto vetách a miery ich hustoty. Pričom mieru hustoty môžeme chápať tak, že:

- Každú vetu rozdelíme na skupiny dôležitých slov, ktoré nie sú od seba vzdialené viac ako štyri nedôležité slová.
- Každá takáto skupina je ohodnotená ako druhá mocnina počtu dôležitých slov, deleno celkový počet slov v tejto skupine. Prehľadne to môžeme zapísať ako: ohodnotenie_segmentu = dôležité_slová^2 / všetky_slová
- Veta je ohodnotená na základe najvyššieho ohodnotenia skupiny v tejto vete.

6. Vybranie najlepšie ohodnotených viet do výsledného súhrnu. Počet viet v súhrne môže byť obmedzený maximálnym počtom najlepších slov, alebo percentuálnym pomerom k pôvodnému textu.

5.1.2 Príklad použitia Luhnovej metódy

Časť pôvodného textu:
The Americans say they freed the Iranian fishermen in the Arabian Sea after more than a month in captivity, and provided fuel and food for them to return home. The rescue was carried out by forces aircraft carrier group, which recently left the Gulf to assist US military operations in Afghanistan. Earlier in the week, the US rejected an Iranian warning to keep its forces out of Gulf waters after Western powers unveiled new sanctions on Iran over its nuclear programme.

Z ktorého sme po filtrovaní pomocou stop slov, po normalizovaní a odstránení málo sa opakujúcich slov dostali výsledný text, v ktorom nedôležité slová nahradíme hviezdičkou.

* American * * freed * Iranian * * * Arabian Sea * * * * * * captivity, * provided fuel * * * * * return *. * rescue * carried * * force aircraft carrier *, * * * * Gulf * * US military operations * Afghanistan. * * * *, * US rejected * Iranian * * * * force * * Gulf water * * power * * sanction * Iran * * nuclear programme.

Ďalej uvedieme príklady výpočtov pre najlepšie ohodnotené segementy v danej vete.

Segment [American * * freed * Iranian * * * Arabian Sea] má ohodnotenie 5^2 / 11 = 2,7.
Segment druhej vety [rescue * carried * * force aircraft carrier] má ohodnotenie 5^2 / 8 = 3,1.
Najvýznamejšia časť tretej verty [force * * Gulf water * * power * * sanction * Iran * * nuclear programe] má výsledné ohodnotenie 8^2 / 17 = 3,7.

Čiže pokiaľ by sme do výsledného súhrnu vyberali najlepšie ohodnotenú vetu, na základe tejto metódy by sme vybrali nasledujúcu vetu:
Earlier in the week, the US rejected an Iranian warning to keep its forces out of Gulf waters after Western powers unveiled new sanctions on Iran over its nuclear programme.

5.2 Edmundsonova metóda

Podľa literatúry [16] táto metóda popisuje nové spôsoby vytvárania extrakcie dokumentov pomocou automatického výberu viet. Automatickým výberom sú vyberané vety, ktoré majú na zoznámenie čitateľa s významom dokumentu najväčší potenciál. Kým predchádzajúce práce [15] boli zamerané iba na jednu súčasť dôležitosti vety, presnejšie, prítomnosť vysokofrekvenčných slov s informačným obsahom (kľúčové slová), táto metóda pridáva ďalšie tri súčasti: praktické slová (význačné slová); slová z nadpisov a titulky; a štrukturálne indikátory (poloha viet). Pomocou ich kombinácie získavame ohodnotenie vety, pričom rovnako ako v predchádzajúcej metóde sa do súhrnu vyberie určité percento najlepšie ohodnotených viet [16].

5.2.1 Spôsoby ohodnocovania viet a ich hodnotiace algoritmy:

Význačný spôsob – Tento spôsob je založený na skutočnosti, že dôležitosť vety je založená na prítomnosti vecných slov ako sú napr. ťažký, dôležitý alebo nemožný. Tento spôsob ohodnotenia využíva pred spracovaný slovník význačných slov, ktorý sa skladá z 3 podslovníkov (kap. 4.2.2):

- bonusových slov - kladne hodnotené slová
- negatívnych slov – záporne hodnotené slová
- bezvýznamných slov – nie sú nijako hodnotené

Pseudoalgoritmus ohodnotenia pomocou význačného spôsobu:

1. Porovnaj každé slovo textu so slovníkom význačných slov.
2. Ohodnoť všetky bonusové slová s hodnotou b > 0, všetky priťažujúce s hodnotou s < 0, všetky bezvýznamné s hodnotou n = 0.
3. Vypočítaj význačnú hodnotu C každej vety sčítaním ohodnotenia slov b, s a n.
4. Vytvor poradie všetkých slov v klesajúcom poradí hodnôt.
5. Vyber vety, ktorých poradie je menšie ako parameter P, ktorý reprezentuje percentuálny počet viet v dokumente.
6. Vyber všetky nadpisy.
7. Zoraď vybrané vety pod ich príslušné nadpisy.
8. Na výstup priraď názov dokumentu, mená autorov a výsledok 7. kroku.

Spôsob pomocou kľúčových slov- Princíp, avšak nie algoritmus tejto metódy bol prvý krát využitý v Luhnovej metóde. Je založený na hypotéze, že slová, ktoré sa často nachádzajú v dokumente sú dôležité, čiže sú kľúčové. Môžeme ich definovať ako nevýznačné slová z najlepšie ohodnotených viet pomocou význačného spôsobu a ich kľúčového ohodnotenia daného ich frekvenciou v tomto zozname. Spôsob ohodnotenia je vysvetlený pomocou nasledujúceho pseudoalgoritmu:

1. Porovnaj každé slovo textu so slovníkom význačných slov.
2. Vytvor tabuľku výrazných, neporovnateľných textových slov, čím vznikajú kandidáti na kľúčové slová.
3. Vypočítaj frekvenciu výskytu k každého kandidáta na kľúčové slovo.
4. Vytvor poradie všetkých slov v klesajúcom poradí hodnôt.

5. Vytvor slovník kľúčových slov zo všetkých kandidátov na kľúčové slovo, ktorých frekvencia je väčšia ako hraničná hodnota W, ktorá reprezentuje percentuálny pomer výskytu slova k počtu slov v dokumente (reprezentuje jeho frekvenciu výskytu).
6. Porovnaj každé slovo v texte so slovníkom a označ každé slovo, ktoré sa v ňom nachádza pomocou hodnoty k, ktorá zodpovedá frekvencii výskytu v texte.
7. Vypočítaj hodnotu kľúčových slov K každej vety ako sumu kľúčových slov hodnoty k v tejto vete.
8. Vytvor poradie všetkých slov v klesajúcom poradí hodnôt.
9. Vyber vety, ktorých poradie je menšie ako parameter P, ktorý reprezentuje percentuálny počet viet v dokumente.
10. Vyber všetky nadpisy.
11. Zoraď vybrané vety pod ich príslušné nadpisy.
12. Na výstup priraď názov dokumentu, mená autorov a výsledok 11. kroku.

Spôsob pomocou nadpisov - Tento spôsob je založený na hypotéze, že autor má predstavu, že titulka dokumentu ohraničuje tematiku dokumentu. Taktiež keď autor rozdelí telo dokumentu na hlavné časti tým tieto časti sumarizuje pomocou vhodne zvolených nadpisov. Hypotéza, že slová z titulky a nadpisov sú dôležité, je štatisticky potvrdená na 99 percent. Čiže nasledujúci pseudoalgoritmus využíva princíp, že najdôležitejšie informácie sa nachádzajú v špecifických častiach textu (názov dokumentu, slová v úvode, slová v závere …):

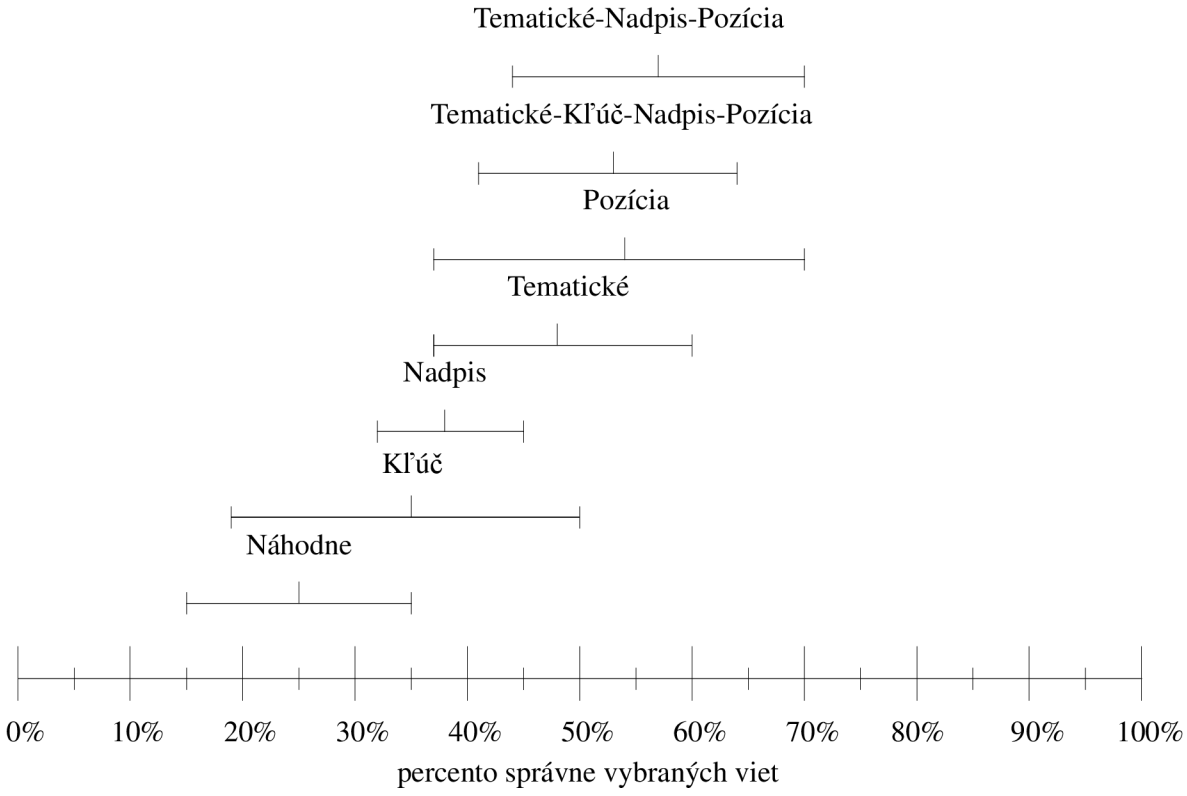
1. Porovnaj každé slovo názvu a nadpisov so slovníkom bezvýznamných slov.
2. Vytvor slovník nadpisov neporovnateľných slov.
3. Ohodnoť všetky slová zo slovníka tak, že pokiaľ sa nachádzajú v názve dokumentu, budú ohodnotené konštantou t_1. Ohodnoť všetky slová ktoré pochádzajú z podtitulu, alebo z nadpisov pomocou t_2.
4. Porovnaj každé slovo z textu so slovníkom. Ohodnoť každé slovo, ktoré sa nachádza aj v slovníku prislúchajúcu hodnotou t_1 alebo t_2.
5. Vypočítaj hodnotu nadpisu T každej vety ako sumu ohodnotení slov v tejto vete.
6. Vytvor poradie všetkých slov v klesajúcom poradí hodnôt.
7. Vyber vety, ktorých poradie je menšie ako parameter P.
8. Vyber všetky nadpisy.
9. Zoraď vybrané vety pod ich príslušné nadpisy.
10. Na výstup priraď názov dokumentu, mená autorov a výsledok 9. kroku.

Pozičný spôsob – Je založený na princípe, že vety nachádzajúce sa pod nadpisom sú dôležité, taktiež sú dôležité vety, ktoré sa nachádzajú na začiatku alebo konci dokumentu, alebo odseku. Čiže nasledujúci pseudo algoritmus využíva to, že dôležité informácie sa vyskytujú na špecifických miestach v texte (začiatok, koniec), čo je využité pri ohodnotení textu pomocou tejto metódy:

1. Porovnaj každé slovo nadpisov so slovníkom slov nadpisov.
2. Ohodnoť všetky zhodujúce sa slová z textu so slovami zo slovníka hodnotou h, ktorá je zo slovníka.
3. Vypočítaj hodnotu nadpisu H pre každý nadpis, sčítaním hodnôt jednotlivých slov nachádzajúcich sa v danom nadpise.

4. Ohodnoť každú vetu pomocou hodnoty nadpisu H.
5. Ohodnoť vetu prvého odseku hodnotou O_1 a posledného hodnotou O_2. Prvú vetu každého odseku ohodnoť hodnotou O_3 a poslednú ako O_4.
6. Spočítaj celkovú hodnotu O každej vety sčítaním O_1,O_2,O_3 a O_4.
7. Spočítaj pozičnú hodnotu L každej vety sčítaním O a H
8. Vytvor poradie všetkých viet v klesajúcom poradí hodnôt.
9. Vyber vety, ktorých poradie je menšie ako parameter P.
10. Vyber všetky nadpisy.
11. Zoraď vybrané vety pod ich príslušné nadpisy.
12. Na výstup priraď názov dokumentu, mená autorov a výsledok 11. kroku.

Kombinácia predchádzajúcich spôsobov – Pomocou kombinácie štyroch predchádzajúcich spôsobov môžeme podľa literatúry [16] dosiahnuť lepšie výsledky viď. Obrázok 5.2, ako keby sme použili iba jednu, alebo náhodne vybrali vetu.



Obrázok 5.2 Kombinácie Edmundsonových spôsobov ohodnotení viet [16].

Najlepšia kombinácia podľa literatúry[16] je kombinácia význačnej metódy, metódy nadpisov a pozičnej metódy.

1. Výber požadovanej hodnoty pre parametre a_1,a_2,a_3,a_4 a P.
2. Výpočet totálneho ohodnotenia $a_1C + a_2K + a_3T + a_4L$ pre každú vetu pomocou jednotlivých ohodnotení C, K, T, L.

3. Vytvor poradie všetkých viet v klesajúcom poradí hodnôt.
4. Vyber vety, ktorých poradie je menšie ako parameter P.
5. Vyber všetky nadpisy.
6. Zoraď vybrané vety pod ich príslušné nadpisy.
7. Na výstup priraď názov dokumentu, mená autorov a výsledok 6. kroku.

5.3 KPC metóda

Nazvaná podľa ich autorov Kupiec, Pedersen, Chen, ktorú prezentovali [17] v roku 1995. Využívajú možnosť klasifikovať každú na vetu na základe jej rysov a podľa toho ju zaradiť, alebo nezaradiť do výsledného súhrnu. Navyše ku každému súhrnu je vytvorený zoznam kľúčových fráz, ktoré sa nachádzajú v danom sumarizovanom texte. Pričom metóda je schopná sa učiť na základe korpusov odborných článkov obsahujúcich originál a ručný súhrn.

5.3.1 Klasifikačné rysy

- **Dĺžka vety:** Krátke vety zväčša nie sú zahrnuté do súhrnu, pričom všetky nadpisy sa považujú za krátke vety. Nastaví sa hraničná hodnota (podľa autorov je najlepšia hodnota 5) a pre vety, ktoré sú dlhšie ako zadaná hodnota je tento rys pravdivý.
- **Ustálené slovné spojenia (frázy):** Vety obsahujúce čokoľvek zo zoznamu ustálených slovných spojení, prevažne dlhých dve slová (napr. this letter, in conclusion …), alebo vety vyskytujúce sa okamžite po nadpisoch, ktoré obsahujú kľúčové slová (conclusions, results, summary ...) sa oveľa častejšie vyskytujú vo výslednom súhrne. Tento rys je pravdivý pre vety, ktoré obsahujú akúkoľvek z 26 fráz, alebo nasledujú za odsekmi, ktoré obsahujú špecifické kľúčové slová.
- **Odsek:** Tento spojitý rys zaznamenáva informácie z prvých desiatich odsekov a posledných piatich v dokumente. Pričom jednotlivé vety sú rozdelené podľa toho, či sa nachádzajú na začiatku, konci (odseky musia byť dlhšie ako jedna veta) alebo v strede odseku (odseky musia byť dlhšie ako dve vety). Tento rys vychádza z podobného princípu použitého v Edmundsonovej metóde, pričom zjednodušuje pozičný spôsob ohodnotenia vety.
- **Tematické slová:** Najčastejšie sa vyskytujúce obsahové slová možno definovať tiež ako tematické (väzby pre slová s rovnakou frekvenciou sú vyriešené na základe dĺžky slova). Je vybraná malá vzorka tematických slov a každá veta je ohodnotená funkciou frekvencie. Táto vlastnosť je binárna - závislá na tom, či je veta v databáze najvyššie ohodnotených viet. V tejto metóde boli vykonané pokusy, pri ktorých boli upravené ohodnotenia viet použité ako potenciálne náhodné, avšak toto riešenie prinieslo horšie výsledky.
- **Veľké slová:** Správne pomenovania sú často rovnako dôležité ako vysvetľujúci text pre skratky napr., "... ASTM (American Society for Testing and Materials)". S týmto rysom sa počíta podobne ako s predchádzajúcim rysom, s tým obmedzením, že tematické slovo, zložené z veľkých písmen nesmie byť prvé slovo vety a musí začínať veľkým písmenom. Navyše sa musí niekoľkokrát vyskytnúť a nesmie byť skrátenou mernou jednotkou (napr. F, C, Kg, atď.). Vety, v ktorých sa tieto slová vyskytujú prvý raz sú ohodnotené dvojnásobne oproti neskorším výskytom.

5.3.2 Kľúčové frázy

KPC metóda navyše od ostatných metód ku každej sumarizácii pridáva zoznam kľúčových fráz, ktoré pozostávajú najmenej z dvoch susediacich slov, ktoré sú väčšinou frázy zložené z podstatných mien a sú prezentované podľa poradia výskytu. Ich výpočet je založený na frekvenčnej analýze sekvencie slov v dokumente. Na identifikovanie kľúčových slov sa používa stop list, ktorý slúži na rozdelenie slov vo vetách na fráze.

5.3.3 Klasifikácia

Pre každú vetu vypočítame pravdepodobnosť, že bude zahrnutá do súhrnu S, pomocou k klasifikačných rysov F_j; j = 1 … k, ktoré môžu byť vyjadrené pomocou nasledujúceho Bayesovho klasifikátora :

$$P(s \in S | F_1, F_2, \ldots F_k) = \frac{P(F_1, F_2, \ldots F_k | s \in S)}{P(F_1, F_2, \ldots F_k)} \quad (5.1)$$

Za predpokladu štatistickej nezávislosti funkcie :

$$P(s \in S | F_1, F_2, \ldots F_k) = \frac{\prod_{j=1}^{k} P(F_j | s \in S)}{\prod_{j=1}^{k} P(F_j)} \quad (5.2)$$

kde $P(s \in S)$ predstavuje pravdepodobnosť, že veta s sa nachádza nepodmienene v súhrnu S, $P(F_j | s \in S)$ je pravdepodobnosť hodnoty príznaku F_j vo vete súhrnu a nakoniec je pravdepodobnosť hodnoty príznaku F_j nepodmienene.

5.3.4 Príklad výpočtu použitím Bayesového klasifikátora

Majme 100 rôznych trénovacích viet, z ktorých ručne vytvoríme 20% súhrn. Majme vybrané klasifikačné rysy viet z metódy KPC: dĺžka vety (F_1), ustálené slovné spojenia (F_2) a tematické slová (F_3).

Podľa počtu rysov v trénovacích vetách štatisticky zistíme:
Rys F_1 sa vyskytuje v 80% vetách, čiže $P(F_1) = 0,8$ a $P(notF_1) = 0,2$.

Rys F_2 sa vyskytuje v 30% vetách, čiže $P(F_1) = 0,3$ a $P(notF_1) = 0,7$.

Rys F_3 sa vyskytuje v 40% vetách, čiže $P(F_1) = 0,4$ a $P(notF_1) = 0,6$.

$P(F_1 | s \in S) = 0,9$ (180 viet z 200)
$P(F_2 | s \in S) = 0,4$ (80 viet z 200)
$P(F_3 | s \in S) = 0,6$ (120 viet z 200)
$P(notF_1 | s \in S) = 0,1$ (20 viet z 200)
$P(notF_2 | s \in S) = 0,6$ (120 viet z 200)
$P(notF_3 | s \in S) = 0,4$ (80 viet z 200)

$P(s \in S)$ je konštantou, čo znamená že pre 20% súhrn má veľkosť 0,2

Majme 4 vety S_1, S_2, S_3 a S_4, ktorým pre zaradenie viet do výslednej sumarizácie vypočítame $P(s \in S | F_1, F_2, F_3)$

Pre vetu S_1 s rysmi F_1 = áno, F_2 = áno, F_3 = áno

$$P(s_1 \in S | F_1 = ano, F_2 = ano, F_3 = ano) = \frac{0,2 * 0,9 * 0,4 * 0,6}{0,8 * 0,3 * 0,4} = 0,45$$

Pre vetu S_2 s rysmi F_1 = áno, F_2 = áno, F_3 = nie

$$P(s_2 \in S | F_1 = ano, F_2 = nie, F_3 = ano) = \frac{0,2 * 0,9 * 0,6 * 0,4}{0,8 * 0,7 * 0,4} = 0,19$$

Pre vetu S_3 s rysmi F_1 = áno, F_2 = nie, F_3 = nie

$$P(s_3 \in S | F_1 = ano, F_2 = nie, F_3 = nie) = \frac{0,2 * 0,6 * 0,6 * 0,4}{0,8 * 0,7 * 0,6} = 0,09$$

Pre vetu S_4 s rysmi F_1 = nie, F_2 = nie, F_3 = nie

$$P(s_4 \in S | F_1 = nie, F_2 = nie, F_3 = nie) = \frac{0,2 * 0,1 * 0,6 * 0,4}{0,2 * 0,7 * 0,6} = 0,06$$

Podľa vypočítaných podmienených pravdepodobností by sme postupne do súhrnov vybrali tieto vety:

- 25% súhrn - veta S_1
- 50% súhrn - veta S_1 a S_2
- 75% súhrn - veta S_1, S_2 a S_3.
- 100% súhrn - veta S_1, S_2 potom S_3 a nakoniec S_4.

Problém výpočtu pomocou Bayseovskej klasifikácie môže nastať v prípade, že niektorý z rysov by sa v trenovacích dátach vôbec nenachádzal, čiže by obsahoval nula prvkov. Potom by aj podmienená pravdepodobnosť bola nulová. Riešením takto vzniknutej chyby je použitie Laplaceovskej korekcie, ktorá pridá do všetkých množín jeden prvok.

6 Analýza a návrh aplikácie

Pred samotnou implementáciou aplikácie je dôležité vytvoriť návrh ako celá aplikácia bude pracovať, a z ktorých častí sa bude skladať. Taktiež by sa mali vymedziť niektoré základné ciele, ktoré chceme aby výsledná aplikácia obsahovala, čiže určiť funkčnosť, použitie a v neposlednom rade tiež spôsob ovládania a formáty vstupov a výstupov. Všetky tieto časti sú z programátorského hľadiska veľmi dôležité pre vytvorenie programov a ich samostatných tried, ktoré budú medzi sebou vzájomne v jednotlivých programoch komunikovať.

6.1 Analýza

Funkčnosť
Mala by byť schopná bezproblémovo automaticky extrahovať dôležité textové informácie z webovej stránky, zadanej pomocou URL, alebo uloženej na disku, do formátu TXT, alebo do vektora reťazcov s parametrami.

Nasleduje ďalšie spracovanie textu použitím vybraných metód sumarizácie, ktoré môžu byť modulárne pridávané do hlavného programu, na tvorbu súhrnu. Pričom užívateľ si môže vybrať zvolenú metódu, môže jej nastaviť požadované parametre a použité slovníky. Program zobrazuje priebeh extrakcie a odhadovaný čas ukončenia, pričom dovoľuje vytvárať log o priebehu extrakcie a tvorbe súhrnu so štatistickými a heuristickými údajmi. Program dovoľuje nastaviť formát výstupu pri výslednom súhrne (XML, HTML, TXT).

Použitie
Aplikácia bude zlepšovať možnosť rozhodovania užívateľa (čitateľa), či mu stojí za to si pozrieť danú stránku podrobne. Bude určená predovšetkým na získavanie súhrnov z webových stránok. Pričom bude možné vytvárať textové extrakty z týchto stránok v kompatibilných formátoch (XML, TXT) pre možné ďalšie spracovanie v iných programoch. Taktiež bude umožňovať vytváranie štatistickej a heuristickej analýzy nad danou stránkou.

Ovládanie
Jeden z cieľov tejto aplikácie je čo najjednoduchšie a intuitívne ovládanie. Preto v návrhu bolo vybrané ako najvhodnejšie ovládanie myšou a vkladanie URL webovej stránky do formulára pomocou klávesnice. Pre daný charakter aplikácie sa zvolené ovládanie javí ako dostatočné. Je vhodné vytvoriť aspoň jednu variantu programu, ktorá bude obsahovať čo najmenej užívateľských prvkov pre čo najľahšie ovládanie.

6.2 Návrh aplikácie

Aplikácia sa bude skladať z viacero programov, ktoré budú slúžiť na extrahovanie textu z webových stránok a následne na vytvorenie súhrnu z tohto textu. Budeme rozlišovať dva hlavné programy a balík viacerých modulov, ktorý bude slúžiť na rozšírenie funkčnosti programov.

- Program na experimentovanie s metódami
  Program na prevádzanie experimentov sa bude skladať z dvoch logických celkov. Prvý celok bude predstavovať hlavnú časť, ktorá bude slúžiť na extrahovanie textu z webovej stránky a jej následnej úpravy, pričom je možné využiť externé aplikácie. Poskytne funkčnú časť programu a užívateľské GUI pričom umožní do tohto programu pridávať moduly. Druhý logický celok sa bude skladať z balíku jednotlivých modulov, ktoré budu poskytovať vybrané sumarizačné metódy a budú vytvárať súhrny z vyextrahovaného textu.
- Program na rýchlu tvorbu súhrnu
  Logické celky z predchádzajúceho programu sú spojené, čiže daný program neumožňuje vkladanie modulov, ale obsahuje vybrané module s najlepšími nastaveniami, aby vytvárali čo najkvalitnejšie súhrny. Pričom GUI tohto programu je čo najjednoduchšie a obsahuje čo najmenej prvkov.
- Balík modulov
  Obsahuje vybrané sumarizačné metódy v podobe modulov, ktoré umožňujú rozšíriť GUI experimentálneho programu. Pričom obsahujú rovnaké metódy na prácu s modulom, čo umožňuje, aby využívali spoločné rozhranie a mohli byť dynamicky pridávané, alebo odoberané z programu.

Pokiaľ technické možnosti pri implementácií dovolia, bude navyše vytvorený applet, ktorý bude umožňovať užívateľom vytvárať súhrny na webovej stránke.

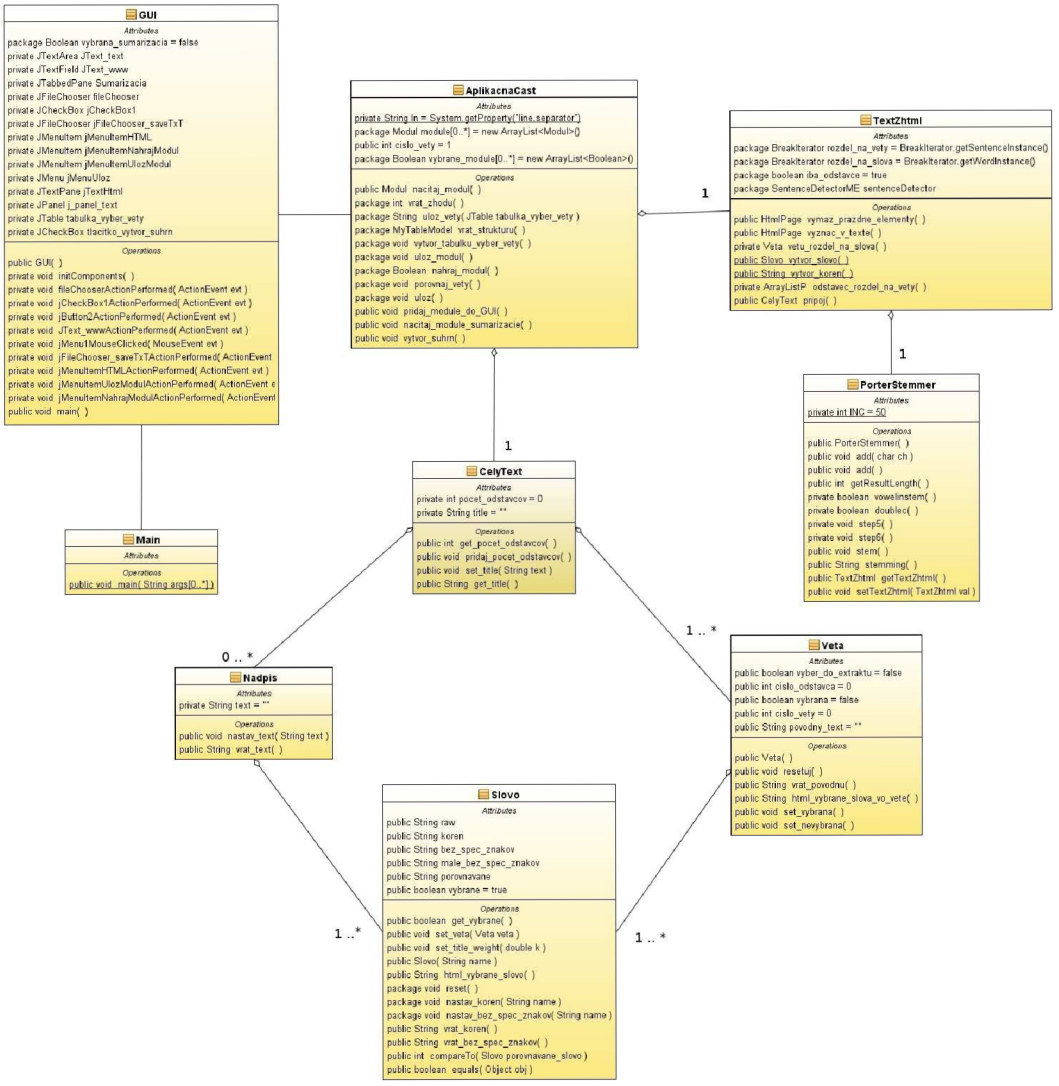
V nesledujúcom prehľade sú vybrané iba tie najdôležitejšie triedy programov pre chod celej aplikácie, pričom boli pomenované tak, aby najlepšie zodpovedali činnosti, ktorú prevádzajú. Triedy sú v prehľade zoradené logicky, podľa svojej naviazanosti na predchádzajúce triedy a príslušnosti k programu, v ktorom sa budú nachádzať.

Hlavné triedy experimentálneho programu:

- `Main` - bude vstupná trieda programu, ktorá vytvára GUI.
- `GUI` - bude slúžiť na ovládanie celého programu, skladať sa z viacero obrazoviek, ktoré umožňujú užívateľovi vytvárať extrakty a súhrny. Ďalej umožní zobrazenie textov a nastavení jednotlivých modulov. Pri vytváraní GUI sa vytvorí časť, ktorá bude slúžiť na vykonávanie akcií zvolených v GUI programu nazvaná `AplikacnaCast`.
- `AplikacnaCast` - bude určená na oddelenie programovej časti od GUI, pričom bude umožňovať zobrazenie svojej činnosti pomocou vybraných komponentov z GUI.
- `TextZhtml` - bude slúžiť na nahranie webovej stránky, podľa zadaného URL, jej následného spracovania a vrátenia textu vo forme vhodného vektoru, alebo podobnej dátovej štruktúry.
- `Text` - bude predstavovať vhodnú reprezentáciu celého textu rozdeleného na slová, vety, nadpise a ďalšie vhodné štruktúry, ktoré budú využité pri práci zo zadaným textom.
- `Utility` - bude obsahovať dôležité jednoduché metódy pre prácu funkčnej časti programu a ďalších tried.

Diagram tried experimentálneho programu

V nasledujúcom diagrame sa nachádzajú najdôležitejšie triedy a pre názornosť ich základné atribúty a operácie uvedené bez parametrov.



Obrázok 6.1 Diagram tried experimentálneho programu

Triedy programu na tvorbu súhrnu:

Podobne ako v predchádzajúcich triedach, s tým rozdielom, že trieda GUI nebude pracovať s načítanými modulmi, ale budú v programe už obsiahnuté.

Triedy balíčkov:

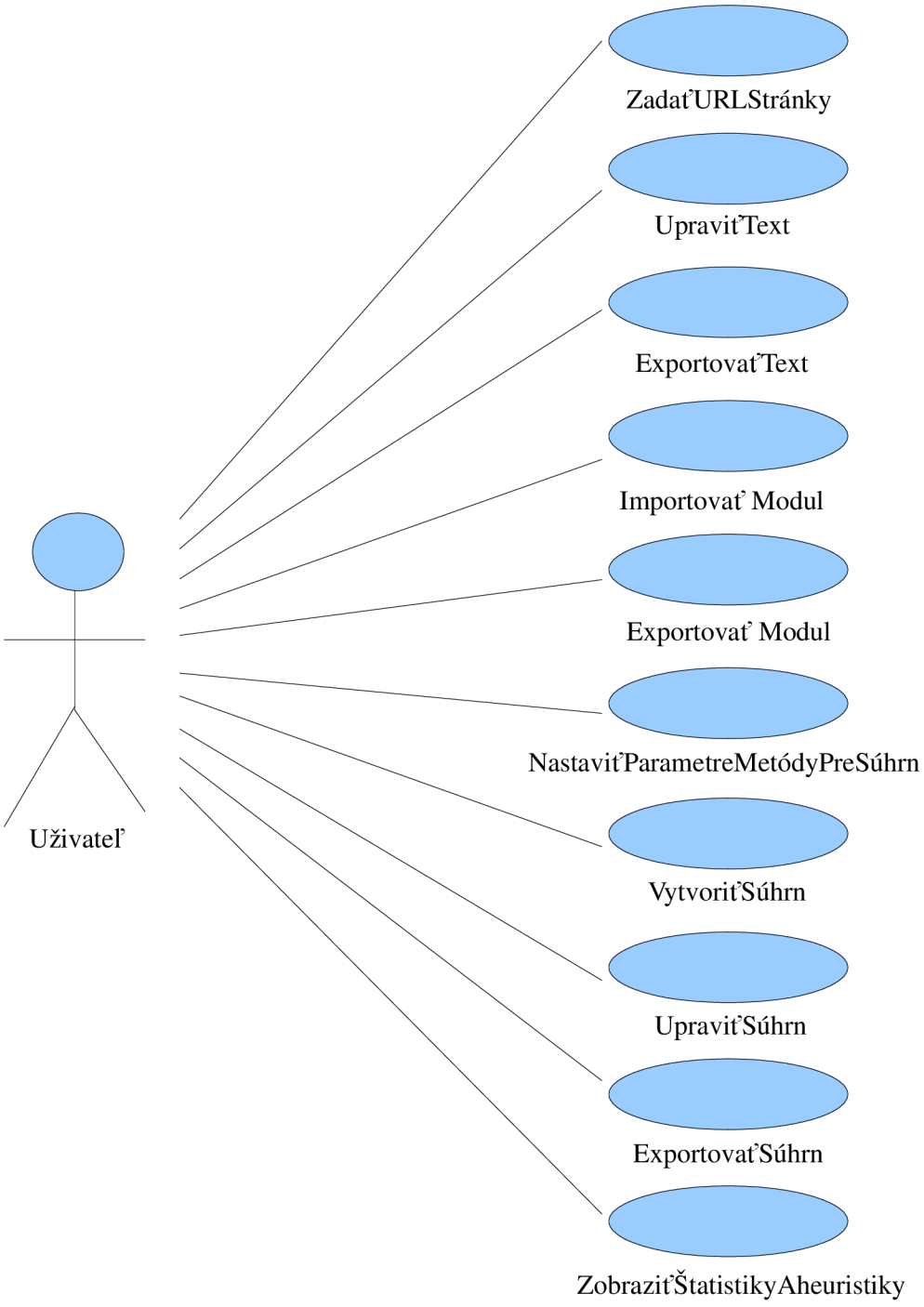
- `Suhrn` – bude predstavovať vstupný bod do modulu, ktorý bude obsahovať rozhranie pre prácu s týmto modulom.

- `GUI` – bude slúžiť na ovládanie daného balíčka a umožní integráciu do GUI experimentálneho programu. Umožní zobrazenie heuristických a štatistických ohodnotení textu pomocou vybranej metódy.
- `AplikacnaCast` – taktiež ako v prípade experimentálneho programu bude určená na oddelenie programovej časti od GUI a bude zobrazovať svoju činnosť pomocou komponentov z GUI.
- `Metoda` – v tejto triede bude obsiahnutá vybraná metóda pomocou, ktorej sa bude vytvárať výsledný súhrn.

6.3 Obmedzenia aplikácie

Najpodstatnejším obmedzením je použitie anglického jazyka, čiže na správnu prácu program potrebuje stránky napísané výlučne v anglickom jazyku. Pričom aplikácia nie je navrhnutá tak aby rozoznávala, či je zadaný jazyk správny. Taktiež aplikácia neumožňuje vytvárať súhrn z iných informačných zdrojov ako sú textové (napr. video, audio formáty, obrázky atď.), pokiaľ neobsahujú vhodnú textovú interpretáciu. Program neumožňuje extrakciu zložitých matematických vzorcov, alebo iných komplikovaných matematických zápisov a nepodporovaných znakov. Pokiaľ by sa jednalo o veľmi rozsiahle stránky, tak sa predpokladá vysoká časová náročnosť kvôli viac násobnému spracovaniu (stemming, porovnávanie so slovníkmi, zobrazenie atď..) vyextrahovaného textu na výsledný súhrn.

6.4 Diagram užitia

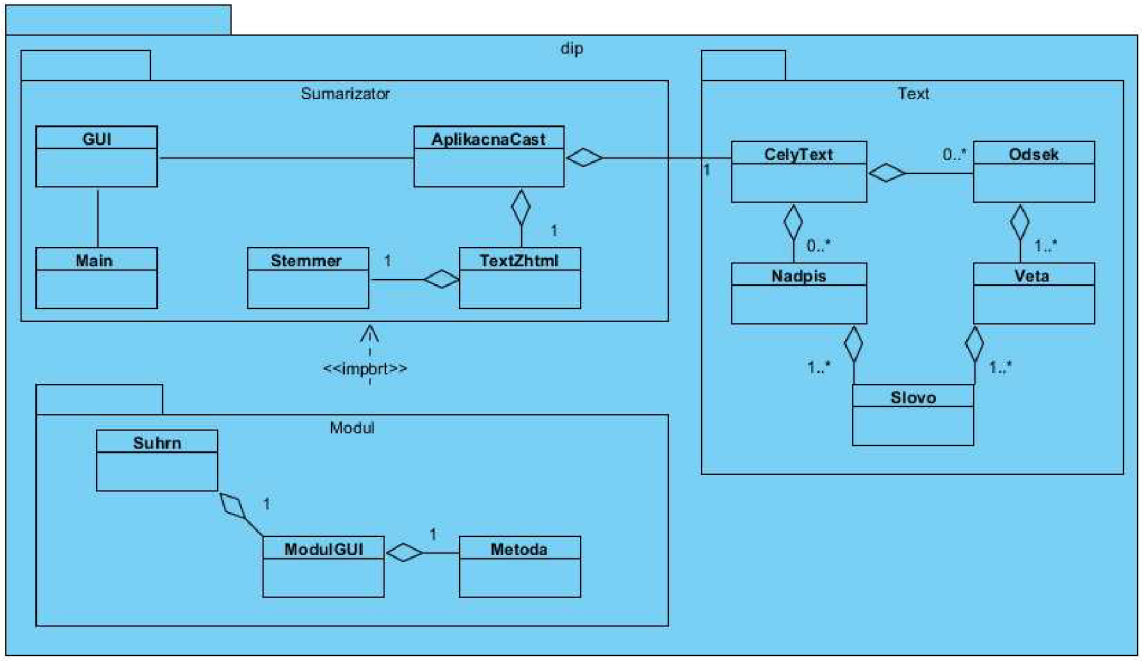


Obrázok 6.2 Diagram užitia

6.5 Diagram baličkov

V tejto časti kapitoly je zobrazené (Obrázok 6.3) predpokladané prepojenie jednotlivých tried experimentalného programu výslednej aplikácie, s možnosťou importu jednotlivých modulov. Pre

väčšiu prehľadnosť sa v triedach nenachádzajú ich atribúty a ani metódy. Navyše pre zjednodušenie je zobrazený iba jeden modul importovaný do programu.



Obrázok 6.3 Diagram balíčkov.

6.6 Zhrnutie požiadaviek

Zhrnutie základných požiadaviek na vytvorenú aplikáciu.

- Rýchla extrakcia textu a tvorba súhrnu.
- Jednoduché a prehľadné ovládanie.
- Možnosť spustenia na bežnom osobnom počítači.
- Možnosť nastavenia jednotlivých parametrov daných metód.
- Možnosť zmeny slovníkov použitých pri tvorbe súhrnu.
- Možnosť zapnutia výpisov o prebiehajúcich operáciách.
- Možnosť pridávania jednotlivých modulov, pokiaľ možno dynamicky.
- Možnosť ukladania získaných údajov do kompatibilných formátov.
- Súhrn poskytuje obstojnú informačnú hodnotu pre užívateľov.

7 Implementácia aplikácie

7.1 Implementačné prostredie a použité nástroje

Na implementáciu výslednej aplikácie bol zvolený programovací jazyk Java, ktorého jednoduchý popis sa bude nachádzať v nasledujúcej časti textu (kap. 7.2).

Aplikácia bola vytvorená vo vývojovom prostredí NetBeans 7.1.1, kde bola zároveň aj testovaná. Bolo vybraná pre svoju jednoduchosť a prehľadnosť pri tvorbe zdrojového kódu, predošlými skúsenostiam pri tvorbe bakalárskej práce a hlavne kvôli tomu, že sa vývojové prostredie nachádza na školskom serveri Merlin.

Nakoniec bolo po zhodnotení dosiahnutých výsledkov upustené od použitia HTML wrapperov, ktoré boli oveľa viacej špecializované, než ako by bolo potrebné pre túto aplikáciu a namiesto nich, bola v záverečnej implementácii použitá knižnica `htmlUnit`, umožňujúca extrahovať text z HTML.

7.2 Java

Keďže na naprogramovanie aplikácie bude použitý tento programovací jazyk vo verzii 6.0, tak by bolo vhodné spomenúť aspoň jeho základne vlastnosti, na základe ktorých bol vybraný.

Java 6.0 [18] je objektovo orientovaný programovací jazyk vyvinutý vo firme Sun Microsystems pôvodne pod menom Oak v roku 1991. Pôvodným zámerom Javy malo byť hlavne poskytnutie programovacieho jazyka pre spotrebnú elektroniku, aby program bolo možné napísať, skompilovať a spustiť na rôznych platformách bez opätovnej kompilácie. Java je objektovo orientovaná, jej syntax je odvodená (upravená a zjednodušená) od jazyka C++. Bola odstránená nielen väčšina konštrukcií, ktoré spôsobovali programátorom problémy, ale aj pribudli niektoré užitočné rozšírenia. Java sa taktiež delí o podobné znaky s jazykmi ako sú Lisp a SmallTalk a tiež so skriptovacími jazykmi ako Perl a Tcl. V súčasnosti je Java vyvíjaná spoločnosťou Oracle.

Základné vlastnosti JAVY:

- **jednoduchá** – hlavná výhoda oproti ostatným programovacím jazykom je v jednoduchosti. Umožňuje programovať bez predchádzajúceho profesionálneho tréningu, za pomoci bežných programovacích praktík pri tvorbe softvéru. Programátori majú dostupný veľký počet knižníc na tvorbu objektov. Od základný typov dát cez I/O objekty, sieťové rozhranie až po nástroje pre tvorbu grafického používateľského rozhrania. Tieto knižnice môžu byť rozširované, aby poskytli nové funkčnosti.
- **objektovo orientovaná** – je navrhnutá tak, aby bola od základu objektovo orientovaná. Ide o techniku programovania, pri ktorej sa zameriava na objekty (triedy) a interface k nim. Všetky dátové typy sú objektové, okrem ôsmych základných dátových typov.
- **interpretovaná** – na rozdiel od C++ sa nevytvára strojový kód na špecifickú platformu, ale na nezávislý medzikód (byte code). Vytvorený medzikód je nezávislý na architektúre zariadenia. Java interpret môže tento medzikód interpretovať priamo na zariadení, pre ktoré je určený. V súčasnosti sa na urýchlenie používajú technológie, ktoré byte code najprv

interpretujú a na základe analýzy štatistik získaných z tejto interpretácie vykonajú preklad často používaných častí do strojového kódu.

- **distribuovaná** – je navrhnutá pre podporu aplikácií v sieti (obsahuje obrovskú knižnicu rutín zameraných na TCP/IP protokoly). Pracuje so vzdialenými súbormi pomocou URL, taktiež má prístup k objektom na lokálnom súborovom systéme.
- **robustná** – je určená pre tvorbu vysoko spoľahlivého softvéru. Poskytuje skoré kontrolovanie problémových situácií, po ktorom nasleduje dynamické (run time) kontrolovanie. Neumožňuje používať niektoré programátorské konštrukcie, ktoré sú častou príčinou chýb (napr. GOTO, odkazy ...). Obsahuje Garbage collector, ktorý slúži na automatickú správu pamäti. Automaticky vyhľadáva už nepoužívané časti pamäti a uvoľňuje ich pre ďalšie použitie. Tento jednoduchý model správy pamäte odstraňuje všetky druhy programátorských chýb.
- **bezpečná** – je vytvorená pre sieťové prostredie, kde je bezpečnosť nesmierne dôležitá. Java umožňuje konštruovať aplikácie, ktoré nemôžu byť infikované zvonku. Má aj vlastnosti, na celkovú ochranu počítača, kde je program vykonávaný, pred nebezpečnými operáciami nepriateľského kódu.
- **platformovo nezávislá** – na každom počítači s Java Virtual Machine (JVM) môže bežať akýkoľvek Java program, kdekoľvek bol napísaný. Nezávislosť jazyka funguje preto, lebo zdrojové programy sa nekompilujú do špecifického strojového kódu, ale na nezávislý byte code. Tento byte code je interpretovaný pomocou JVM na zvolenej platforme. Podľa konkrétnej platformy sa môže prispôsobiť vzhľad a chovanie aplikácie.
- **viacvláknová** – podporuje spracovanie viacvláknových aplikácií na úrovni programovacieho jazyka s pridanými sofistikovanými synchronizačnými metódami, čo umožňuje vykonávať viac vecí naraz. Knižnice poskytujú Thread8 triedy a run-time systém umožňuje sledovať stav a upravovať zamykacie primítivy. Systémové knižnice boli napísané, tak aby vlákno bolo zabezpečené a používalo funkcie, ktoré nespôsobujú konflikty viacerých paralelných vlákien.

7.3 Implementácia

Hlavným cieľom tejto práce bolo vytvoriť aplikáciu, ktorá by spĺňala požiadavky uvedené v predchádzajúcom návrhu. Implementácia prebiehala postupne vo viacerých na sebe nezávislých etapách, čo umožňovalo rýchly postup. Tak ako bolo uvedené už v návrhu tak celá aplikácia bola implementovaná ako dva nezávislé programy, kde jeden je určený na experimentovanie s metódami a bol na implementáciu náročnejší a druhý slúži na jednoduché vytváranie súhrnov a jeho implementácia nepredstavovala žiadne väčšie problémy. Moduly nakoniec boli implantované ako adresár obsahujúci binárny súbor reprezentujúci danú metódu s GUI a slovníky využívané touto metódou.

Taktiež sa pri implementácií objavili určité problémy, tie budu zhrnuté v samostatnej kapitole 8.6. Medzi hlavné časti implementácie, ktoré budú vysvetlené podrobnejšie, patrili:

- Vytvorenie základných objektov použitých v aplikácii pre prácu z textom ako je slovo, veta, celý text a ich vzájomné prepojenie a práca s týmito objektami.
- Spracovanie webovej stránky na text, následne jeho rozdelenie na základné objekty.
- Implementácia modulu obsahujúceho Luhnovu metódu.

- Implementácia modulu obsahujúceho Edmundsonovu metódu.
- Implementácia modulu obsahujúceho KPC metódy.
- Vytvorenie experimentálneho programu, ktoré umožňuje nastavovať parametre a zobrazuje jednotlivé výsledky predchádzajúcich metód a umožňuje ukladať výsledný text.
- Vytvorenie zjednodušeného programu, s možnosťou voľby členitosti vstupného textu.

7.4 Implementované triedy

Programy a moduly boli rozdelené do jednotlivých tried, ktoré sú v nasledujúcich podkapitolách usporiadané podľa jednotlivých programov a ich vývojových etáp, postupne ako boli jednotlivé triedy naprogramované. Názvy tried každého programu sú pomenované výstižne, aby už z názvu bolo na prvý pohľad jasné, na čo daná trieda slúži. Každá ďalej vysvetľovaná trieda je zjednodušená a obsahuje iba niektoré najdôležitejšie funkcie a premenné, ktoré budú ďalej objasnené.

7.4.1 Základné objekty aplikácie

V prvotnej fáze implementácie boli vytvorené základné triedy `Slovo`, `Veta`, `Odsek`, `Napdis` a `CelyText`, ktoré reprezentujú jednotlivé časti spracovávaného textu. Všetky tieto triedy obsahujú základné atribúty, pre ich využitie pri sumarizácii v jednotlivých metódach.

Každé `Slovo` obsahuje tri hlavné atribúty, ktoré reprezentujú jeho nespracovanú podobu, podobu bez špecialných znakov a podobu po stemmingu, čo využívajú zvolené sumarizačné metódy. Taktiež obsahuje odkaz na vetu, v ktorej sa nachádza, čo urýchľuje pri zmene vlastnosti slova aj následnú zmenu vo vete. Slovo a veta umožňuje reštartovanie svojho ohodnotenia pomocou metódy `reset()`, čo sa používa pri opakovanom ohodnotení slov pomocou vybraných metód. Pokiaľ je to nutné, tak na zvýraznenie vybraného slova v metóde sa používa `html_vybrane_slovo()`, čo spôsobí využitie html syntaxe v komponentoch Swingu a zobrazí dané slovo červenou, alebo zelenou farbou podľa jeho atribútu `vybrane`. Trieda `Veta` sa skladá z pola jednotlivých slov, premennej pre výber pomocou jednotlivých sumarizačných metód tejto vety na základe jej slov.

7.4.2 Spracovanie webovej stránky

Na túto úlohu je určená trieda `TextZhtml`, hlavne jej metóda `pripoj()`, kde na základe užívateľom zadaného URL z GUI programu, sa pomocou objektu `client` z triedy `WebClient` z knižnice `htmlunit`, vytvorí pripojenie na server. `Client` stiahne danú webovú stránku pričom emuluje zadaný webový prehliadač, konkrétne v tejto aplikácii to je CHROME vo verzii 16. Následne trieda `TextZhtml` pomocou metódy `vyznac_v_texte()` nájde zvolené atribúty `(h1-h6, p, b)`, pomocou ktorých následne daný text obalí párovou značkou v preddefinovanom formáte: `<!atribut!>Nejaký text.</!atribut!>`, čo je využité pri spracovaní čistého textu, ktorý získame pomocou `asText()` z `htmlunit`.

Takto získaný text je na základe html atribútov rozdelený na nadpisy a odseky. Do úvahy sa neberú samostatné vety nenachádzajúce sa v odsekoch, pretože na základe testovania viacerých významných webových stránok sa zistilo, že tieto vety zväčša nereprezentovali relevantný text, ktorý by prinášal nejaké konštruktívne informácie potrebné do výslednej sumarizácie.
Všetky odseky boli následne rozdelené na vety a vety na slová. Pričom najprv na vytvorenie viet bola využitá trieda `BreakIterator` obsiahnutá v Jave SE, ale kvôli horším výsledkom boli použité

nástroje `opennlp`. Nadpisey skladajúce sa iba zo slov a odseky, sú spolu s názvom dokumentu vložené do `CelyText`.

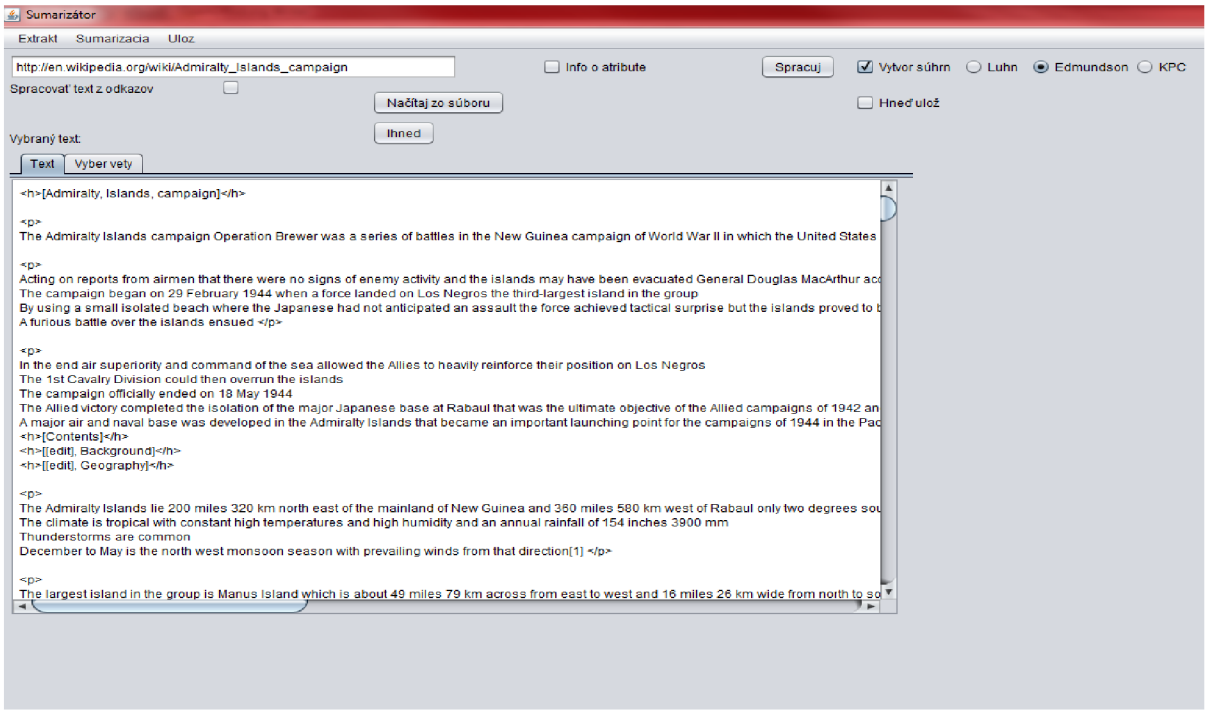
7.4.3 Experimentálny program

Je rozdelený na 2 časti na GUI a na aplikačnú logiku, kde navzájom tieto 2 časti komunikujú.

GUI progamu

Tvorí vstupno-výstupné rozhranie medzi aplikáciou a užívateľom. Skladá sa z veľkého počtu `Swing` komponentov, pričom medzi najdôležitejšie patria:

- `j_panel_text`, ktorý slúži na zadanie `URL` webovej stránky, z ktorej sa bude vytvárať sumarizácia alebo vyberie uloženú stránku zo zadanej pozície na disku. Umožňuje nastaviť, ktoré sumarizačné metódy majú byť použité na vytvorenie súhrnu, alebo iba zobrazí výpis vyextrahovaného textu z webovej stránky, poprípade očíslovanie jednotlivých viet, čo sa využilo v praxi pri ručnej extrakcii, kde sa vybrané vety nemuseli opisovať celé, ale bolo použité iba ich poradové číslo v dokumente.
- `j_panel_sumarizacia` slúži na sprístupnenie GUI modulov do GUI hlavného programu, čím s nimi umožňuje užívateľovi pracovať.



Obrázok 7.1 GUI

Aplikačná časť progamu

Tvorí funkčnú časť programu, s ktorou užívateľ komunikuje pomocou GUI. Umožňuje vytváranie súhrnov, prácu s modulami a ukladanie výsledkov sumarizácie. Obsahuje viacero metód, pričom najdôležitejšie sú nasledujúce:

- `vytvor_suhrn` – pomocou triedy `TextZhtml` vytvorí textový extrakt, ktorý slúži ako vstupný parameter vybranej sumarizačnej metódy, ktorá z tohto textu vytvorí v danom moduly súhrn.
- `nahraj_modul()`- slúži na nahranie (import) vybraného modulu do adresára moduls, pričom tento modul je sprístupnený po opätovnom pustení experimentálneho programu.
- `uloz_modul()` - umožní uloženie daného modulu do súboru s príponou `.suma`
- `uloz()` - ukladá textové dáta do súboru s vhodnou koncovkou, ktorá reprezentuje určenie tohto súboru.

7.4.4 Modul pre Luhnovú, Edmundsonovú a KPC metódu

V tejto časti budú spoločne vysvetlené triedy a metódy implementované pre jednotlivé moduly spolu s podobou ich GUI, ktorý sa vkladá do `j_panel_sumarizacia` experimentálneho programu. Všetky moduly majú jednu rovnakú triedu `Suhrn`, ktorá slúži na komunikáciu s hlavným programom. Obsahuje viacero metód umožňujúcich prácu s týmto modulom:

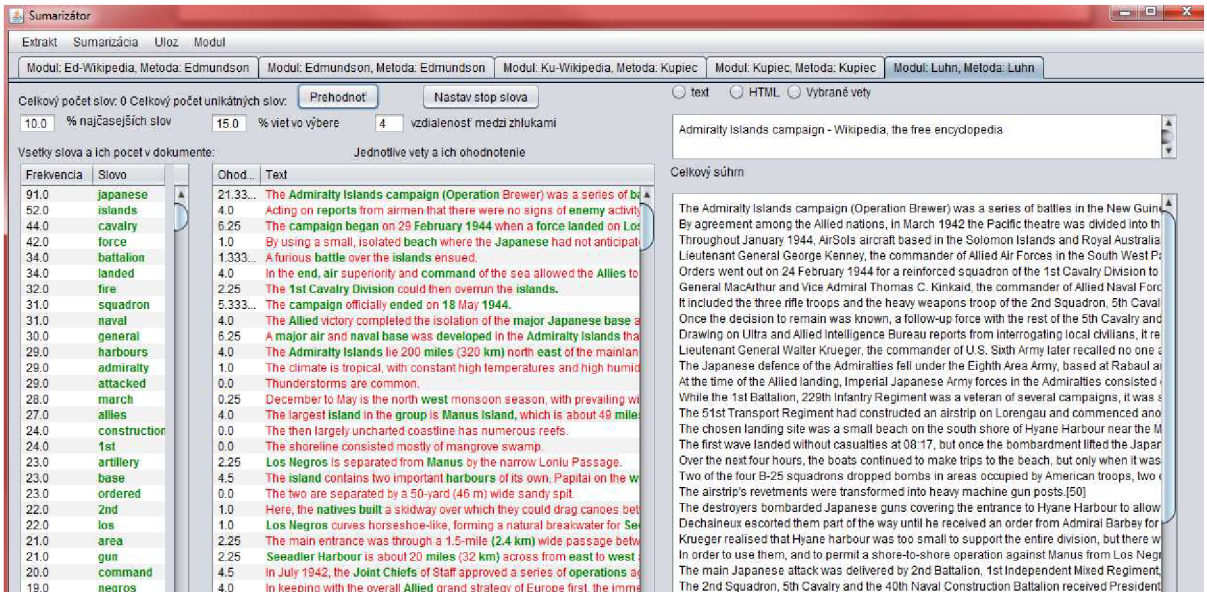
- `nastav_text()` : danému modulu sprístupní vyextrahovaný text z programu vo forme nadpisov, viet a slov.
- `get_suhrn()` : vráti súhrn vytvorený danou metódou vo forme Stringu.

Každý modul obsahuje triedu `ModulGUI`, ktorá zobrazuje užívateľské rozhranie na prácu s týmito sumarizačnými metódami. Pričom každý modul využíva v tejto triede iné objekty a metódy na zobrazenie heuristickej a štatistickej analýzy a výsledného súhrnu. `ModulGUI` navyše dovoľuje nastavovať dôležité parametre, ktoré ovplyvňujú výber viet do výsledného extraktu (pre každý modul Obrázok 8.2, 8.3 a 8.4). Hodnoty týchto parametrov sa pri prvom spustení aplikácie načítajú z textového súboru `config`. Pri všetkých metódach je navyše možnosť meniť slová použité v jednotlivých slovníkoch. Ďalšou funkčnou možnosťou je usporiadať si dané slová a vety podľa ich ohodnotenia a z toho zistiť hraničné hodnoty, kedy už neboli dané vety a slová vybrané do výslednej sumarizácie. Pri novom vykonaní zadanej metódy, alebo pri zmene parametrov sa vždy reštartuje ohodnotenie viet aj slov.

Navyše každý modul sa skladá aj z triedy nazvanej podľa zadanej sumarizačnej metódy.

Modul Luhn

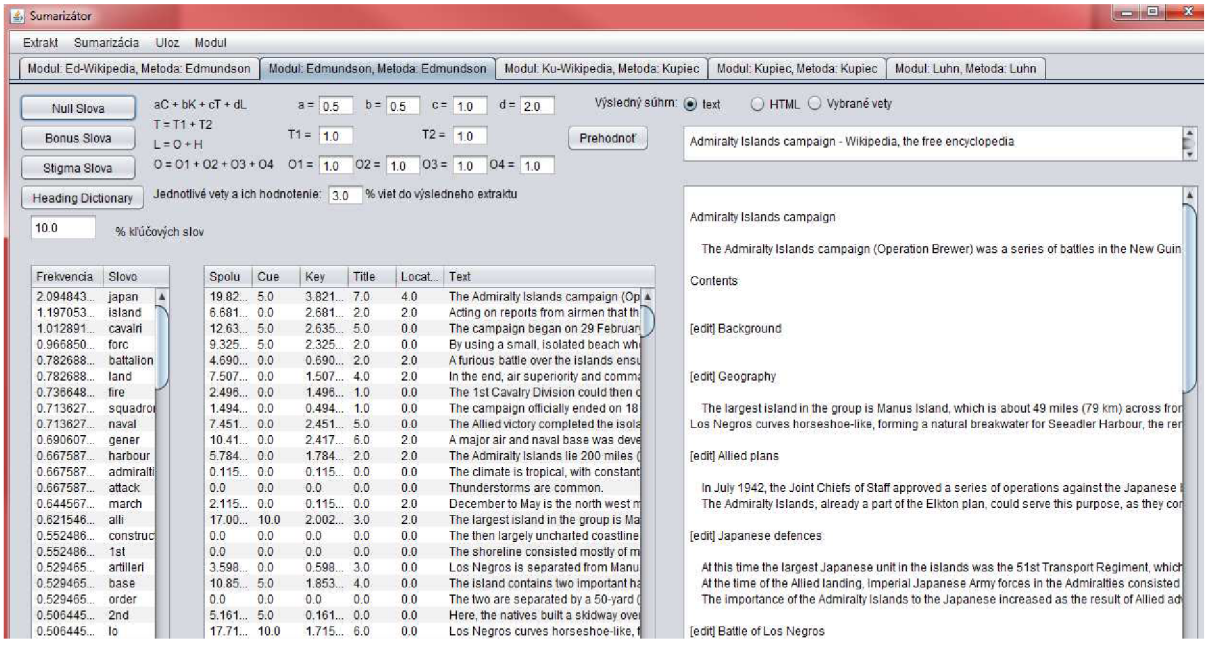
Z textu uloženého v `CelyText` vytvorí výslednú sumarizáciu pomocou metódy `vytvor_extrakt()`, ktorá sa skladá z viacerých etáp. Najprv `nacitaj_stop_slova()` vytvorí slovník stop-slov. Ďalej metóda `spracuj_vety()` najprv porovnáva slová vo vete so stop-slovami. Zo slov, ktoré nie sú stop-slová, vytvorí koreň slov, a takto upravené slová vloží do slovníka, v ktorom sú uložené jednotlivé slová a ich výskyt v texte. Zo slovníka pomocou `vyber_slova()` označí zvolené percento najpočetnejších slov a priradí im parameter vybrané. Následne `ohodnot_vety()` vypočíta ohodnotenie pre všetky vety podľa zvolenej metódy, tieto vety sa následne zoradia a `vyber_vety()` vyberie určené percento viet s najlepším ohodnotením. Ktoré sa pomocou `vypis_extrahovany_text()` môžu predávať iným objektom ako string.



Obrázok 7.2 GUI – Modul Luhn

Modul Edmundson

Pracuje analogicky ako predchádzajúca metóda pričom využíva k svojej činnosti viacej slovníkov (`stigma`, `bonus`, `heading` a `null`) a navyše ohodnocuje slovo a následne vetu viacerými parametrami, ktorých váhu pri konečnom ohodnotení viet ide zmeniť pomocou metódy `ohodnot_vety_koeficientom()`. Tak ako predchádzajúca metóda tak aj táto poskytuje výpis jednotlivých viet, ale navyše pridáva do výsledného extraktu aj všetky nadpisy.

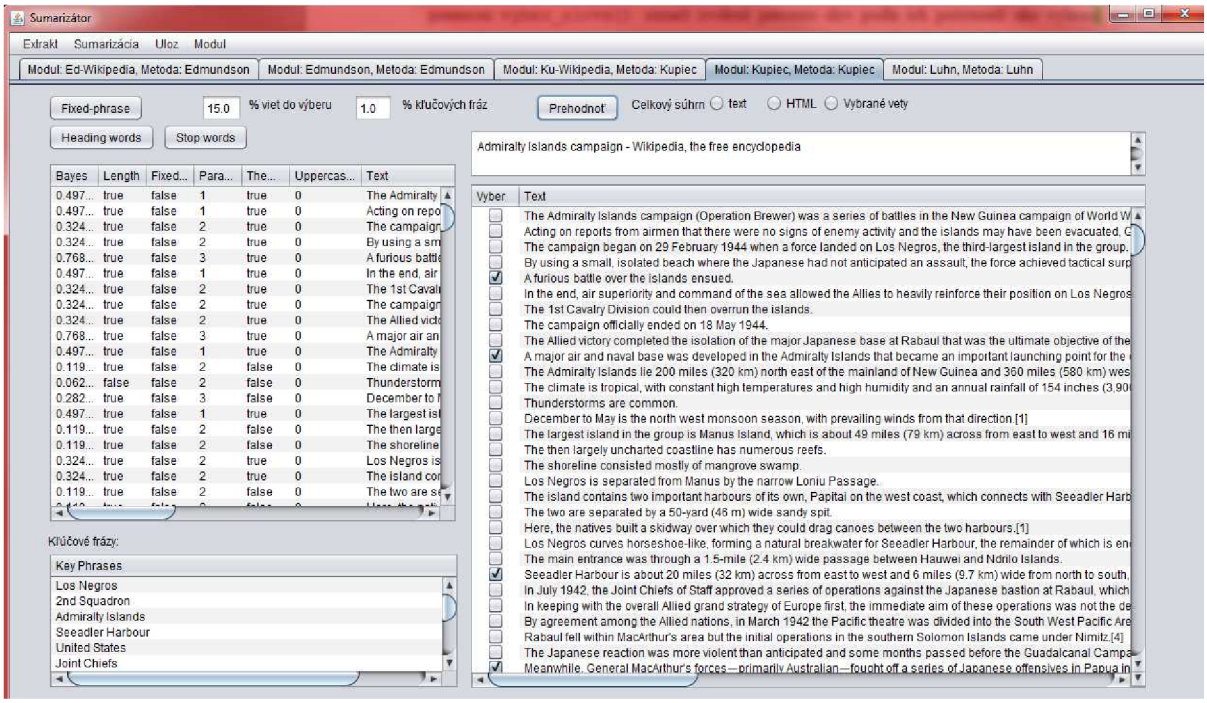


Obrázok 7.3 GUI – Modul Edmundosn

Modul KPC

Je postavený na inom princípe, kde využíva predchádzajúce ručné ohodnotenie sumarizácie textu, ktoré načíta z textového súboru prostredníctvom `nacitaj_ohodnotenie()`.Táto funkcia vypočíta koľkokrát sa daný rys nachádza pri vybranej alebo nevybranej vete do celkového ručného súhrnu. Následne modul KPC nastaví vetám jednotlivé rysy metódou `nastav_rysy_viet()` a pomocou `ohodnotVety()` vypočíta bayesovým klasifikátorom na základe zistených rysov vety číselné ohodnotenie každej vety. Následne metódou `vyber_vety()` vyberie najlepšie ohodnotené vety.

Navyše tento modul počíta najčastejšie slovné spojenia v texte upravenom pomocou stop listu ktoré reprezentujú kľúčové frázy, pričom na obrazovku zobrazuje vždy 1% slovných spojení podľa počtu ich výskytov.



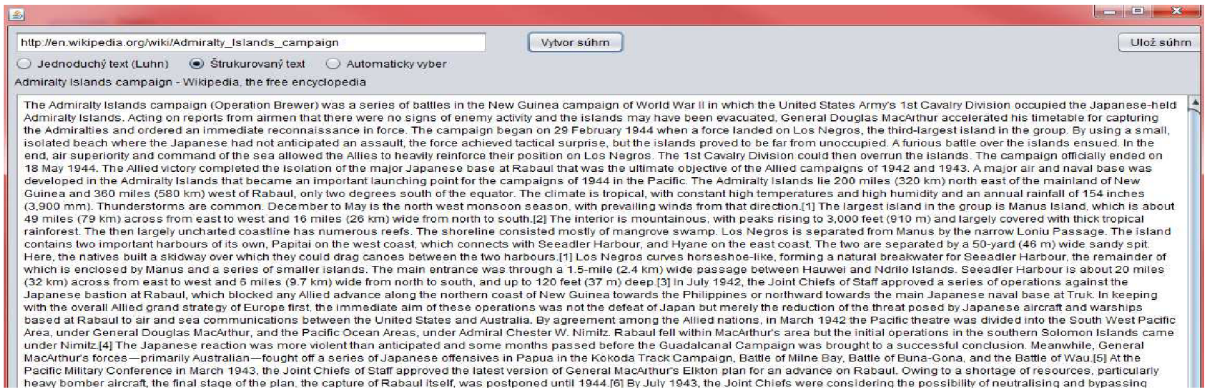
Obrázok 7.4 GUI – Modul KPC

7.4.5 Zjednodušený program

Pracuje obdobne ako experimentálny program, pričom neobsahuje zbytočné nastavenia. Vďaka jednoduchosti umožňuje nasadenie aj na mobilných platformách podporujúcich Javu a Html. Všetky parametre sumarizácie sú prednastavené, pričom poskytuje tri možnosti sumarizácie podla textu:

- štrukturovaný – na sumarizáciu sa použije KPC metóda
- jednoduchý – na sumarizáciu sa využije Luhnová metóda
- automatický – aplikácia sama zistí aká je štruktúra textu a podľa toho aplikuje najvhodnejšiu sumarizačnú metódu

Umožňuje automaticky ukladať vytvorený súhrn ako HTML súbor.



Obrázok 8.5 GUI zjednodušeného programu – Výsledok sumarizácie

7.5 Načítanie modulov

Keďže hlavný experimentálny program je založený na dynamickom načítavaní modulov, tak v tejto časti bude vysvetlené ako bolo načítanie implementované, a na akom spôsobe pracuje. Pričom k riešeniu boli použité tieto informačné zdroje [19].

Riešenie je založené na dynamickom načítavaní `.jar` súborov a Java tried z týchto súborov.

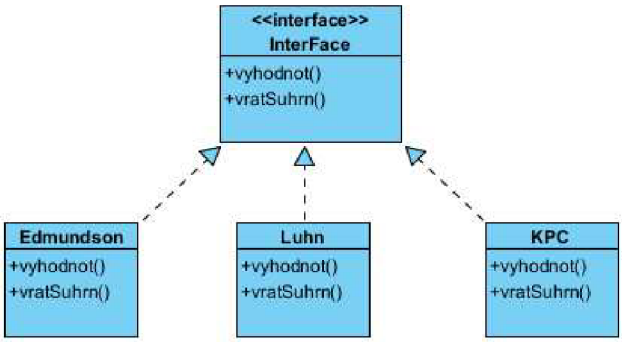
7.5.1 Návrhový vzor

Pri riešení načítania modulov bol využitý návrhový vzor adaptér, ktorý podľa literatúry [20] slúži na poskytnutie stabilného rozhrania pre podobné komponenty, ktoré majú rôzne rozhrania. Toto riešenie sa skladá z troch logických častí, ktoré sa nachádzajú v týchto programoch:

- Loader – dynamicky načítáva a používa `.jar` súbore a Java triedy
- Komunikačné rozhranie – definuje rozhranie, ktoré môže byť použité na komunikáciu medzi programom a dynamickou knižnicou
- Dynamické knižnice – Loader načíta a následne použije tieto knižnice pomocou komunikačných rozhraní

Loader, komunikačné rozhranie a dynamické knižnice musia byť rozdelené v rôznych projektoch, kvôli nasledujúcim podmienkam:

- Loader importuje komunikačné rozhrania
- Dynamické knižnice taktiež importujú komunikačné rozhrania
- Loader by nemal vedieť nič o dynamických knižniciach
- Dynamické knižnice nemajú vedieť o Loaderovi



Obrázok 8.6 Využitie adaptéru v experimentálnom programe

7.5.2 Príklad implementácie načítania modulu

V tejto časti bude na jednoduchom príklade zdrojového kódu predchádzajúcich častí vysvetlená činnosť tohto návrhového vzoru.

Vytvorenie komunikačného rozhrania

```
package interface;
public interface Pozdrav {
  String povedz(String name);}
```

Vytvorenie dynamickej knižnice

```
package pozdrav;
import interface.Pozdrav ;
public class povedzAhoj implements Pozdrav {
  public String povedz(String name){
    return "Ahoj " + name;}}
```

Vytvorenie Loaderu

Loader využíva k načítaniu nami vybranej triedy triedu `Class`, ktorá sa štandardne nachádza v Jave SE.

```
package pozdravProgram;
import interface.Pozdrav;
..
public class Main {
  public static void main(String[] args) throws Exception {
    //Potrebujeme URL k nacitaniu suboru jar file.
    URL u = new File(args[0]).toURI().toURL();
    //Nacitanie suboru jar pouzitim URLClassLoader.
    URLClassLoader cl = new URLClassLoader(new URL[]{u});
    //Nacitanie nazvu triedy
    String class = args[1];
```

```
Pozdrav ahoj;
//Nacitanie triedy Pozdrav
ahoj =(Pozdrav) Class.forName(class, true, cl).newInstance();
//Vypisanie Ahoj Jano
System.out.println(ahoj.povedz("Jano"));}}
```

7.6 Vytvorenie slovníkov

Predstavovalo dôležitú a veľmi časovo náročnú časť implementácie jednotlivých metód, keďže Luhnová, Edmundsonová a aj KPC metóda k svojej činnosti nevyhnutne potrebovali slovníky pomocou, ktorých môžu byť ohodnocované jednotlivé slová a vety. Pričom tieto slovníky neboli verejnou súčasťou jednotlivých zverejnených prác autorov [15,16,17], preto k činnosti sumarizačných metód ich bolo nevyhnutné vytvoriť. V nasledujúcej časti textu je opísaný stručný popis vytvárania jednotlivých slovníkov. Na testovanie bolo vytvorených viacero sád, podľa problematiky, z ktorej je tvorený súhrn.

7.6.1 Luhn – Stop slová

Dôležitý slovník, ktorý sa používa vo všetkých troch metódach. Sú v ňom obsiahnuté najfrekventovanejšie slová anglického jazyka bez znateľnej informačnej hodnoty. Spracovanie nebolo náročne, pretože na webe sa nachádza veľa podobných dokumentov.

7.6.2 Edmundson - Význačné slová

Do tejto skupiny môžeme podľa [16] zaradiť tri slovníky tj.:

- **bezvýznamné slová** – obsahuje rovnaké položky ako slovník stop slov. Patria sem napr.: číslovky, sloveso byť, predložky, zámena, prídavné mená, členy atď.
- **stigma slová** - zhoršujú ohodnotenie vety, ktorá ich obsahuje, sú to napr. tieto: znevažujúce a význam zhoršujúce slová.
- **bonus slová** – vety s týmito slovami majú väčšiu šancu nachádzať sa v súhrne. Medzi tieto slová patria: porovnávacie prídavné mená, superlatívy, príslovky záveru, slovesá kauzality atď.

Všetky slová boli vybrané z jednotlivých vedeckých článkov a webových stránok, pričom bolo určené s akou pravdepodobnosťou zlepšuje, alebo zhoršuje možnosť výskytu danej vety v súhrne. Navyše boli pridané synonymá pomocou slovníka synonym.
Pričom je možné, aby jednotlivé slová mali inú kladnú, alebo zápornú hodnotu. Poprípade aby mali slová z jednotlivých druhov slovníkov rovnaké hodnoty.

7.6.3 Edmundson – Slovník nadpisov

Do tohto slovníka boli vybrané názvy nadpisov podľa pravdepodobností výberu daných viet pod týmto nadpisom do výsledného súhrnu, ktorý môže nadobúdať hodnoty v rozsahu `<0.0;5.0>`, pričom väčšinou je rozsah `<0.0;2.0>`, ale pre slová, ktorých význam je zhrnutie, alebo synonymum tohto slova je hodnota `5.0`. Hodnota `5.0` znamená určite vybrané a `0.0` nevybrané.

7.6.4 KPC - Slovník nadpisov a fráz

Prvá časť vychádza z prechádzajúceho riešenia, pričom je do tohto slovníka vybraný určitý počet najlepšie ohodnotených nadpisov. Vybrané frázy predstavujú slovné spojenia, alebo slová ktoré sa najčastejšie vyskytujú v ručných súhrnoch.

7.6.5 Práca so slovníkmi

Na načítanie zvolených slovníkov sa využívajú podľa potreby metódy `nacitaj_slovnik()`, pokiaľ nás nezaujímajú hodnoty zadaných slov, alebo `nacitaj_slovnik_s_hodnotami()`. Ukážka formátovania súboru, s ktorými pracujú predchádzajúce metódy:

```
#uvod
introducing;;1.0#vysvetlenie
assumption;;0.5#predpoklad
assumptions;;0.6#predpoklady
#Popis
describe;;0.4#popis
descriptions;;0.4#popisy
definition;;0.5#definicia
effect;;0.3#ucinok
#Vseobecne zhrnutie
resume;;2.0
aggregate;;2.0
total;;2.0
```

7.7 Vstup a výstup aplikácie

Celá aplikácia je vytvorená tak aby bolo možné všetky nastavenia jednotlivých modulov (slovníky a konfiguračný súbor) a výsledky extrakcie a sumarizácie uložiť do vhodných formátov (txt, alebo HTML). Takto uložené výsledky, je možné zobraziť vo vhodnom textovom, alebo internetovom prehliadači.

7.7.1 Import a export rozširujúcich modulov

Aplikácia umožňuje import modulov zo zvoleného umiestenia na disku, čo predstavuje prekopírovanie vhodného adresára, v ktorom sa nachádza zadaný modul s vhodnými slovníkmi a súborom `config` do adresára `moduls` nachádzajúceho sa v adresári aplikácie.

Taktiež umožňuje export modulu do zvoleného umiestenia, pričom pomocou exportu je možné vytvárať viacero kombinácii pre nastavenie modulu danej metódy.

7.7.2 Nahratie spracovávaného dokumentu

V experimentálnom programe je tvorený zadaním URL umiestenia dokumentu na webe, alebo výberom z lokálneho umiestenia. Zjednodušený program umožňuje iba pomocou URL. Vstupný súbor musí byť vo formáte HTML, alebo XHTML, pričom môže obsahovať CSS, JavaScript a ďalšie prvky, avšak tieto musia byť správne implementované.

7.7.3 Export výsledného súhrnu

Je možný vo formáte HTML, alebo txt. Každý modul používa k vytvoreniu súboru iné pravidlá a niektoré moduly delia text na nadpisy, odseky a vety, ale iné ukladajú výsledný súhrn iba vo forme viet.

- **txt**
  Súbory v formáte txt sú uložené, tak že nadpisy sú vždy vo formáte nadpis, po ktorom nasleduje jeden volný riadok a po ňom text. Titulka dokumentu sa nachádza na prvom riadku napísaná veľkými písmenami.
- **HTML**
  Výsledné súbory v HTML formáte obsahujú text uložený pomocou HTML tagov, kde nadpis a odsek je v tagoch `<H2></H2>` pre nadpis a `<p></p>` pre odsek. Titulka dokumentu je uložená v hlavičke dokumentu v tagoch `<title></title>`.

7.8 Problémy počas implementácie a ich riešenie

Táto časť sa zaoberá popisom závažnejších problémov, ktoré sa vyskytli pri implementácii programovej časti tejto diplomovej práce a spôsob, akým by mali byť vyriešené.

Keďže všetky popisy využitých metód neobsahovali nikde slovníky, tak museli byť vytvorené. Pričom pri slovníkoch, ktoré využívajú ohodnotenie jednotlivých slov a slovných spojení bola určená pravdepodobnosť, s ktorou by boli vybrané, alebo nevybrané do výslednej sumarizácie textu.

Taktiež nastali problémy pri implementácii metódy KPC, keďže sa nikde nedali zohnať presné postupy ako túto metódu naprogramovať tak museli byť vytvorené vlastné pravidlá pri výbere počtu frekventovaných slov, rovnako pri určovaní slov, ktoré sú skratkami a ich počte skratiek. A taktiež pri ohodnocovaní jednotlivých viet na základe príslušnosti k odsekom, v ktorých sa nachádzajú.

Pri vytváraní programu vo forme appletu, došlo k problému, že základná aplikácia k svojej činnosti využívala externé knižnice, ale applet kvôli bezpečnostným obmedzeniam nedovoľuje prácu s týmito knižnicami, čo by sa dalo riešiť pomocou vytvorenia unixového serveru na verejnej IP adrese, kde by bol daný applet podpísaný. Avšak nakoniec sa od vytvorenia appletu upustilo.

Pri spracovaní webových stránok bolo upustené od vlastného riešenia extrakcie textu, keďže niektoré stránky využívali opätovného presmerovania a vytvorenia textu za použitia JavaScriptu. Čo malo za následok otestovanie viacero voľne dostupných knižníc pričom najlepšou bola `htmlUnit` umožňujúca vytvorenie textu, ktorej najnovšia verzia je využitá v tejto aplikácii.

8 Experimenty s metódami

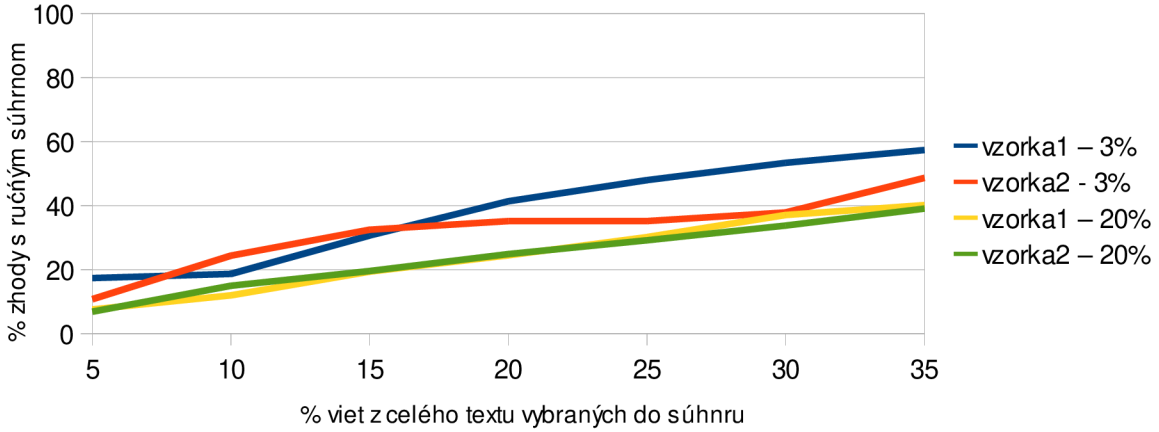
Táto kapitola popisuje testovanie. Testovanie bolo uskutočnené pomocou experimentov na vytvorenie súhrnu postupne pomocou každej sumarizačnej metódy. A takto vytvorené súhrny boli porovnané s vytvorenými ručnými extraktmi, a to indikatívny (podľa KPC metódy cca. 3 percentný súhrn, ktorý predstavuje najdôležitejšie informácie) a informačný (20 percentný súhrn) ručný súhrn. Za účelom testovania boli vytvorené dve testovacie vzorky z rôznych dokumentov:

- vzorka1, ktorá predstavuje 10 vedeckých článkov zameraných na oblasť informačných technológií s priemerným rozsahom približne 250 viet. Všetky články sú prepísané do jazyka HTML pričom sú odstránené tabuľky, obrázky a použitá literatúra. Táto vzorka má simulovať pôvodné využitie vybraných metód a mal by prinášať celkovo najlepšie výsledky pre indikatívny súhrn pomocou Edmundsonovej a KPC metódy.
- vzorka2 sa skladá z 10 článkov zo serveru Wikipedia.org dlhých cca. 130 viet. Obsahuje štrukturovaný text s nadpismi a odsekmi. Slúži na simulovanie najčastejšieho využitia tejto aplikácie.

Pričom ku každej metóde je pre lepšiu názornosť priložený graf, ktorý zodpovedá percentuálnej zhode výsledného automatického súhrnu (s hodnotami 5, 10, 15, 20, 25, 30 a 35%) vytvoreného touto metódou a ručne vytvoreného informatívneho a indikatívneho súhrnu.

8.1 Luhnová metóda

Z grafu (graf 8.1) je vidieť, že vytvárané súhrny nie sú kvalitné. Hlavnou príčinou zlých výsledkov tejto metódy bolo, že nevyužíva k svojej činnosti žiadne slovníky, okrem slovníka obsahujúceho stop slová. Navyše ani nevyužíva ohodnocovanie pomocou nadpisov, preto je pre túto metódu celkom problematické nájsť podstatné informácie v štrukturovanom texte, ktorý obsahuje veľa odlišných slov a nedokáže vytvoriť kvalitný slovník na ohodnotenie viet a následne vybrať najdôležitejšie vety do koncového súhrnu, čo je vidieť v nasledujúcom grafe (viď. graf 8.1). Taktiež je pre túto metódu takmer nemožné vybrať podstatné informácie, ktoré sa nachádzajú iba pod určitými nadpismi ako je Abstract, Summary atď, čo umožňujú Edmundsonova a Kupiecova metóda.

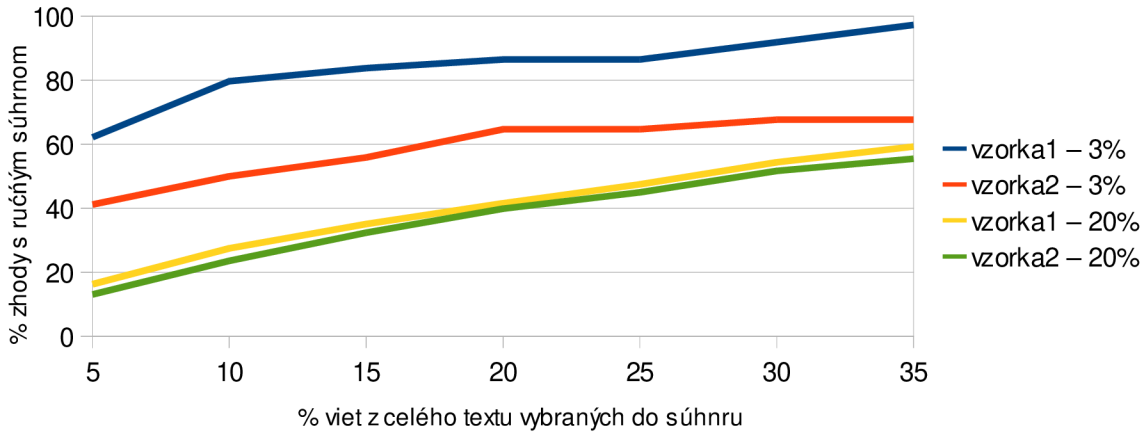


Graf 8.1 Luhnová metóda v porovnaní s ručnými súhrnmi

8.2 Edmundsonová metóda

Podľa nasledujúcich grafov táto metóda v porovnaní z Luhnovou metódou prináša približne 100% zlepšenie výsledkov vo všetkých prípadoch sumarizácie, čo je spôsobené tým, že Edmundsonova metóda je úzko závislá od ohodnotenia jednotlivých bonusových slov, ale aj od pozície viet v texte, prípadne pod ktorým nadpisom sa daná veta nachádza. Pri dobre vytvorených slovníkoch určených špeciálne pre daný typ stránok dosahuje veľmi dobré výsledky, avšak pri použití na inom type stránok podobne ako v prípade KPC metódy sa táto výhoda stráca.

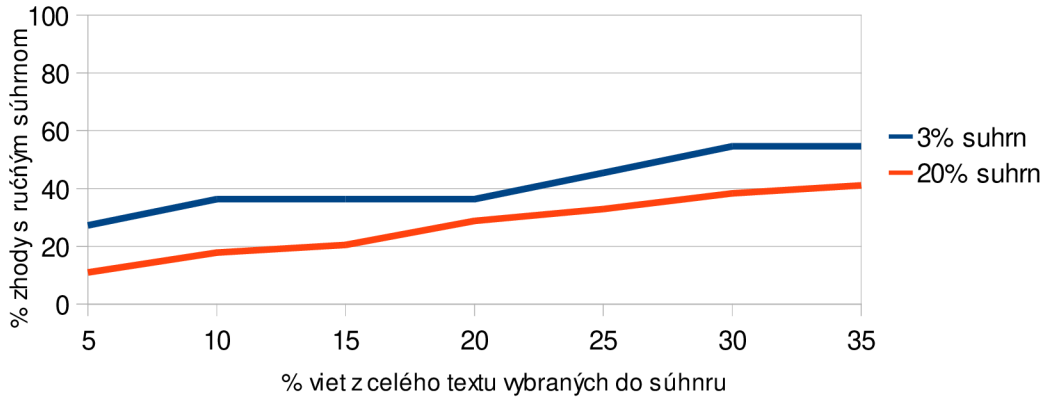
Už pri 10% výslednom súhrne pomocou Edmundsonovej metódy podáva dostatočné informácie z indikatívneho ručného súhrnu zo sumariazovaného textu v prípade vzorky 1, avšak pri vzorke 2 sú dostatočné informácie až pri 20% súhrne. Pri porovnaní s informačným ručným súhrnom sú tieto výsledky oveľa horšie, čo dokazuje graf, ale stále lepšie ako pri použití Luhnovej metódy, avšak to nepredstavuje problém, keďže táto metóda je primárne určená na vytváranie indikatívneho súhrnu.



Graf 8.2 Edmundsonová metóda v porovnaní s ručnými súhrnmi

8.2.1 Použitie nevhodných slovníkov

Nasledujúci graf má dokázať, že kvalita výsledného súhrnu záleží na správne zvolenom slovníku, vytvorenom pre danú problematiku, a následnom ohodnotení viet. Preto si ukážeme, že pokiaľ využijeme ohodnotenie, ktoré na to nie je určené, nevykazuje výsledný súhrn dostatočnú kvalitu. V prípade tohto testu bude na vytvorenie súhrnu z vybraných dokumentov zo vzorky1 využité ohodnotenie určené pre vzorku2.



Graf 8.3 Edmudnsonová metóda - nevhodné ohodnotenie pre danú tému

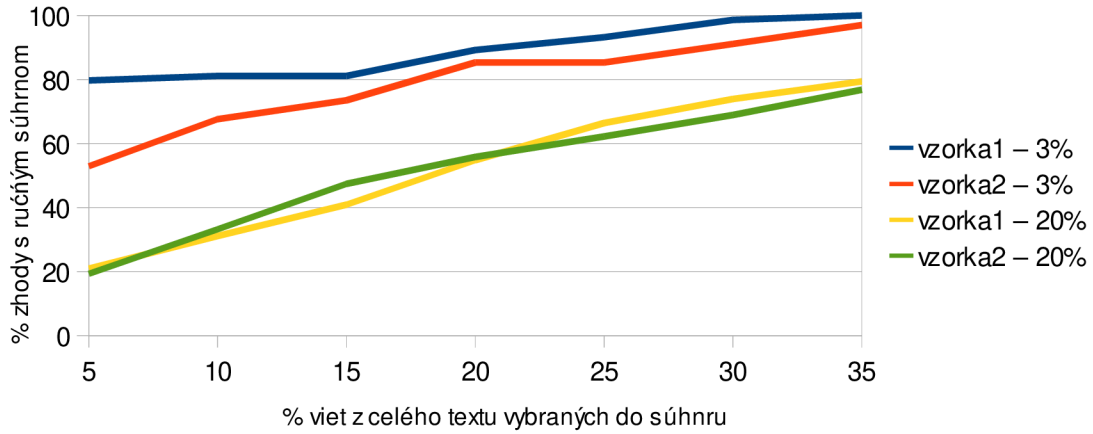
Z predchádzajúceho grafu (Graf 8.3) je vidieť, že pri použití nesprávneho slovníka, je pre Edmundsonovu metódu problematické vytvoriť vhodný indikatívny súhrn pre dané dokumenty. Taktiež informatívny súhrn nie je veľmi kvalitný a dosahuje zhoršenia až o 50%.

8.3 KPC metóda

Prináša najlepšie výsledky z predchádzajúcich metód, čo umožňuje hlavne to, že rovnako ako Edmundsonova metóda využíva k výberu viet hlavne ich pozíciu v sumarizovanom texte. Presnejšie umiestenie na začiatku, alebo konci textu, prípadne polohu viet pod vybranými nadpismi.

Pri primárnom určení tejto metódy na vytváranie indikatívnych súhrnov z vedeckých prác dosahuje už pri 5% súhrne výborné výsledky, ktoré sú o 30% lepšie ako v prípade Edmundsonovej metódy. Pri vzorke 2, reprezentujúcej články zo serveru Wikipedia, tieto výsledky dosahuje, až okolo 15% súhrnu, čo je tiež dobrý výsledok, ale je spôsobený, že táto metóda je primárne určená na vedecké práce s určitou pevne danou štruktúrou textu, ktorá sa avšak na webových stránkach, z ktorých sa vytvárala vzorka 2 nenachádza.

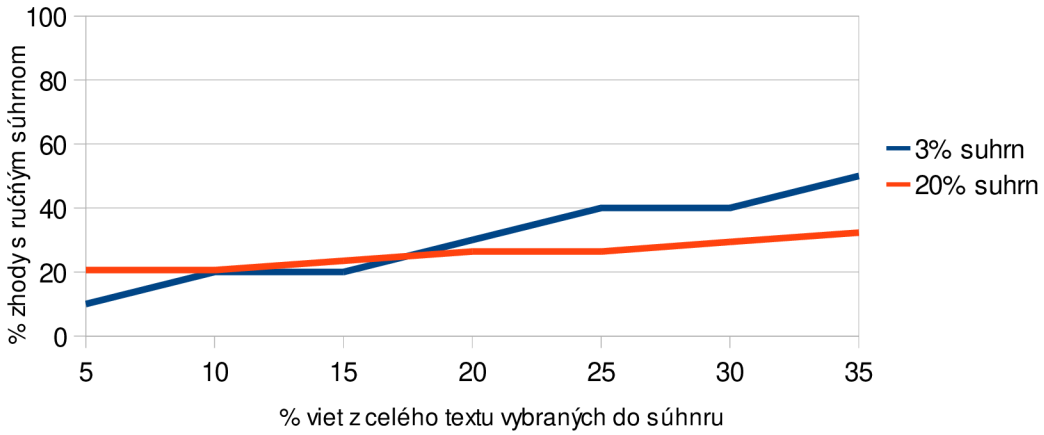
Pri porovnaní s ručným informačným súhrnom, poskytuje dobré výsledky až pri 25% súhrnu, čo je v porovnaní s Edmudnosnovej metódy zlepšenie o viac ako 10%. Čiže túto metódu je možné použiť aj pri vytváraní informačných súhrnov z webových stránok, ale pri použití vhodných slovníkov, pomocou ktorých aplikácia vie určiť ktoré časti textu obsahujú najviacej podstatných informácií (napr. plot, summary, career atď.).



Graf 8.4 KPC metóda v porovnaní s ručnými súhrnmi

8.3.1 Použitie nevhodných slovníkov

Rovnako ako v predchádzajúcej metóde bude aj tu využité ohodnotenie, ktoré nebolo vytvorené pre danú problematiku. Preto bude pre vybrané dokumenty zo vzorky1 využité ohodnotenie pre vzorku2.



Graf 8.5 KPC metóda - nevhodné ohodnotenie pre danú tému

Z grafu (Graf 8.5) je zrejmé, že pre KPC metódu je veľmi obtiažne vytvoriť vhodný indikatívny a informatívny súhrn pre daný dokument na podklade nevhodného ohodnotenia. Zhoršenia sú oproti dobre vytvoreným ohodnotením viet až 2-3 násobné.

8.4 Zhodnotenie testov

Z predchádzajúcich testov, ktoré prezentujú jednotlivé grafy vyplýva, že výsledky použitia Luhnovej metódy v porovnaní s ostatným sumarizačnými metódami sú v štrukturovanom texte obsahujúceho nadpisy oveľa horšie. Dá sa povedať, že vybrané vety do výslednej sumarizácie iba tipuje na základe najviac frekventovaných slov.

Edmudnsonová metóda ponúka pri správne vytvorených slovníkoch a ohodnoteniach pre zadanú problematiku prekvapivo dobré výsledky pri indikatívnom súhrne, avšak pri informatívnom súhrne sú výsledky oveľa horšie avšak stále vhodne slúžia na získanie prehľadu o danom dokumente.

KPC metóda s predchádzajúcich metód prináša najlepšie výsledky pri porovnaní s indikatívnym, ale aj informatívnym ručným súhrnom. Je pri vhodnom vytvorení ohodnotenia pre danú tému a problematiku taktiež možná použiť pri vytváraní súhrnov z štrukturovaných webových dokumentov.

9 Záver

Hlavným cieľom tejto diplomovej práce bolo vytvorenie aplikácie, ktorá vytvára sumarizáciu dokumentov na webe. K dosiahnutiu stanoveného cieľa bolo dôležité v prvom rade zoznámenie sa s danou problematikou dolovania textových dát so zameraním na algoritmy sumarizácie textu. Ďalej bolo nutné naštudovať metódy sumarizácie textu a vybrať najvhodnejšie z nich pre sumarizáciu na webových stránkach. Postup, ktorý nás doviedol k výsledku sa dá rozčleniť na dve časti.

Prvou je získanie textových dát z webovej stránky, k čomu bolo potrebné nájsť vhodné nástroje a určiť, ktoré z nich sa využijú pri výslednej implementácii buď ako inšpirácia vlastného kódu, alebo ako medzičlánok pri získavaní dát. Ako vhodná sa javila trieda nástrojov závislých na HTML, z ktorých najlepším riešením pre náš problém boli nástroje zdarma, ktoré môžu pracovať aj bez GUI. Avšak nakoniec bolo od tohto zámeru opustené a bola použitá knižnica htmlUnit, ktorá umožňuje priamu extrakciu textu z webových stránok.

Druhou časťou bolo naštudovanie vhodných metód, ktoré boli použité pri implementácii programu na tvorbu výsledného súhrnu. Po dohode s vedúcim tejto diplomovej práce boli vybrané metódy jednoduchšie na implementáciu. Zvolené boli tieto tri sumarizačné metódy (Luhnová, Edmudnsonvá a KPC metóda), ktorých činnosť je podrobne popísaná v kapitole o sumarizácii.

Následne bola navrhnutá koncepcia aplikácie, ktorá bude robiť sumarizáciu textových dokumentov. Aplikácia bola v návrhu rozdelená na dve logické časti, kde prvá doluje textové dáta a sprístupňuje ich vo forme extraktu druhej logickej časti, ktorá na nich prevádza sumarizáciu na základe zvolenej sumarizačnej metódy.

Ďalšia časť tejto práce prestavuje implementáciu navrhnutého riešenia. Pričom k implementácii tejto aplikácie som využil programovací jazyk Java 6.0. Hlavná aplikácia umožňuje vkladanie modulov, ktoré pridávajú nové sumarizačné metódy, alebo umožňujú využívanie viacero rovnakých metód s iným nastavením, alebo inými slovníkmi.

Nasledujúcou časťou bolo experimentovanie s navrhnutou aplikáciou, kde boli uskutočnené experimenty s jednotlivými sumarizačnými metódami a ich použitím na vybranej vzorke dát. Na vybraných dátach boli testované ich pôvodné zameranie, ako aj ich nové, významné pre sumarizáciu z webových stránok. Pri experimentoch bolo zistené, že pokiaľ daná metóda využíva k svojej činnosti slovník, tak je možné pri vhodne vytvorenom slovníku podľa druhu a zamerania webovej stránky vytvoriť kvalitný extrakt.
Toto umožňovali Edmundsonova a KPC metóda, narozdiel od nich Luhnová metóda neposkytovala vôbec uspokojivé výsledky, ktoré by sa dali využiť v praxi.

Táto diplomová práca ukázala, že je možné na sumarizáciu dokumentov na webe využívať aj jednoduchšie sumarizačné metódy na tvorbu súhrnov, ale nevýhodou je, že musia k svojej činnosti používať vhodne ohodnotené slovníky s vybraným zameraním pre danú problematiku.

Existuje viacero spôsobov rozšírenia tejto práce. Keďže aplikácia umožňuje vkladanie modulov, prvým možným možným rozšírením je jednoduché pridanie ďalších pokročilých sumarizačných metód. Taktiež je možné do aplikácie v budúcnosti pridať vybranú lexikálnu databázu pre anglický jazyk, pomocou ktorej by mohli byť vytvárané kvalitnejšie súhrny na základe ohodnotení pomocou obsiahnutých slovných druhov vo vete, alebo ich vzájomnou kombináciou. Ďalším rozšírením je možnosť práce s iným značkovacím jazykom (napr. Tex, Word atď.).

Literatura

[1] Burget, R., Zeman, D.: Studijní opora ITW, VUT Brno, 2006.

[2] Fielding, R. a kol.: The Internet Society (1999) : Hypertext Transfer Protocol – HTTP/1.1, [online], [cit 2012-03-01], URL:http://www.ietf.org/rfc/rfc2616.txt

[3] Burget, R.,Hruška, T.: Studijní opora Internetové aplikace (WAP) II. část SGML, HTML, CSS, DOM, VUT Brno, 2007.

[4] Janovský, D.: Jak psát web, [online], [cit 2012-03-01], URL:http://www.jakpsatweb.cz/

[5] W3C.: HTML 4.01 Specification, [online], [cit 2012-03-01], URL: http://www.w3.org/TR/html401/

[6] Honza, P.: Jak na Web, [online], [cit 2012-04-01], URL: http://www.jaknaweb.com/clanky/xhtml

[7] W3C.: XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition), [online], [cit 2012-03-01], URL: http://www.w3.org/TR/html401/

[8] Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: From Data Mining to Knowledge Discovery in Databases. AI- Magazine, 1996, [on-line], [cit 2012-06-04], URL: http://www.kdnuggets.com/gpspubs/aimag-kdd-overview-1996-Fayyad.pdf .

[9] Feldman, R. Sanger, J.: The Text Mining HandBook, Cambridge University Press 2007.

[10] Grefenstette, G. Hull, D.A.: A Detailed Analysis of English Stemming Algorithms, [online], [cit 2012-04-02], URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.21.3856&rep=rep1&type=pdf

[11] Boronat, A: A comparison of HTML-aware tools for Web Data extraction, [online], [cit 2012-04-01], URL:lips.informatik.uni-leipzig.de/files/2008-9.pdf

[12] Leander, A: A Brief Survey of Web Data Extraction Tools, [online], [cit 2012-04-01], URL: http://homepages.dcc.ufmg.br/~berthier/books_journal_papers/sigmod_record_2002.pdf

[13] Jones, K.: Automatic summarising: factors and directions, [online], [cit 2012-05-01], URL: http://www.cl.cam.ac.uk/archive/ksj21/ksjdigipapers/summbook99.pdf

[14] Ježek, K., Steinberger, J.: Sumarizace textů, [online], [cit 2012-05-01], URL: http://textmining.zcu.cz/publications/SumarizDATAKON.pdf

[15] Luhn, H.: The Automatic Creation of Literature Abstracts*, [online], [cit 2012-06-01], URL: https://text-analysis.googlecode.com/files/luhn58.pdf

[16] Edmundson, H.: New Methods in Automatic Extracting, [online], [cit 2012-07-01], URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.83.5638&rep=rep1&type=pdf

[17] Kupiec, J., Pedersen J., Chen F.: A Trainable Document Summarizer, [online], [cit 2012-08 01], URL: http://www.jopedersen.com/Publications/kupiec95trainable.pdf

[18] Škurla, J.: Výukový program pro demostraci principů indexování, VUT Brno, 2009.

[19] Jenkov, J.: Dynamic Class Loading and Reloading in Java, [online], [cit 2012-05-04], URL: http://tutorials.jenkov.com/java-reflection/dynamic-class-loading-reloading.html#classloader

[20] Zendulka, J.: UP – rozpracování (objektový návrh), prednáška k predmetu AIS. [cit 2012-06-04], URL: https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/AIS-IT/lectures/10_OOnavrh.pdf

Zoznam príloh

Príloha 1. DVD obsahujúce zdrojové kódy, aplikáciu, testovacie dáta a inštalačný súbor vývojového prostredia.