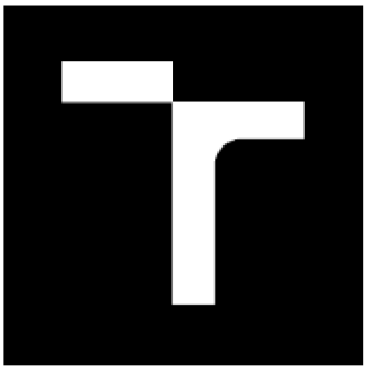
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a
komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2019 Ing. Štěpán Odstrčil



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

APLIKACE PRO PARSOVÁNÍ A ANALÝZU OBSAHU WEBOVÝCH STRÁNEK
TOOL FOR PARSING AND ANALYSING OF WEB PAGES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Ing. Štěpán Odstrčil**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. ZOLTÁN GALÁŽ**
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

Student: Ing. Štěpán Odstrčil ***ID:*** 146639
Ročník: 3 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Aplikace pro parsování a analýzu obsahu webových stránek

POKYNY PRO VYPRACOVÁNÍ:

V rámci bakalářské práce bude navržena, implementována a otestována aplikace pro parsování a analýzu textu z webových stránek. Aplikace bude vytvořena v programovacím jazyku Python, bude obsahovat grafické uživatelské rozhraní a možnost generování reportů založených na technikách zpracování přirozeného jazyka (tzv. natural language processing).

DOPORUČENÁ LITERATURA:

[1] BIRD, Steven, Ewan KLEIN a Edward LOPER. Natural language processing with Python. Beijing: O'Reilly, c2009. ISBN 978-0-596-51649-9.

[2] MITCHELL, Ryan. Web scraping with Python: collecting data from the modern web. Sebastopol, CA: O'Reilly Media, 2015. ISBN 1491910291.

Termín zadání: 1.2.2019 ***Termín odevzdání:*** 27.5.2019

Vedoucí práce: Ing. Zoltán Galáž
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Tato bakalářská práce se zabývá parsováním textu z HTML stránek a jejich analýze a rozboru. Byly použity techniky z Natural Language Processingu, neboli Zpracování Přirozeného Jazyka. Byla napsána knihovna v programovacím jazyce Python, za použitím nejnovějších technologií, postupů a knihoven. Byl zpracován popis těchto knihoven a tříd, jejich použití a příklady. Aplikace dále byla otestována unit testy. Aplikace obsahuje GUI (Graphical User Interface) pro snadnější používání a demonstraci funkcionalit.

KLÍČOVÁ SLOVA

Natural Language Processing, Parsování webových stránek, Parsování HTML stránek Zpracování Přirozeného Jazyka, Python, GUI

ABSTRACT

This bachelor's thesis is dealing with parsing of text in HTML pages and its analysis. Practices from Natural Language Processing were used. There were written libraries (or packages) in programming language Python, with use of modern practices, techniques and libraries. The usages and examples of these libraries and classes were made. All these libraries were tested using Unit tests. Application contains GUI (Graphical User Interface) for wasier usefulness and demonstration of functionality.

KEYWORDS

Natural Language Processing, WEB Parsing, HTML Parsing, Python, GUI

Bibliografická citace:

ODSTRČIL, Š. Aplikace pro parsování a analýzu obsahu webových stránek. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2019. 40 s. Vedoucí bakalářské práce Ing. Zoltán Galáž.

PROHLÁŠENÍ

„Prohlašuji, že svou diplomovou (bakalářskou) práci na téma Aplikace pro parsování a analýzu obsahu webových stránek jsem vypracoval samostatně pod vedením vedoucí/ho bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne 28.5.2019

Ing. Štěpán Odstrčil
autor práce

PODĚKOVÁNÍ

Rád bych poděkoval všem za podporu jak ve změně kurzu mého života, díky kterému jsem si vybral tuto fakultu a obor. Mé rodině, která i přes údivy, že studuji další školu mě podpořila a stále při mně při všech strastích a útrapách tohoto studia. Dále bych chtěl poděkovat mému vedoucímu bakalářské práce, Ing. Zoltánu Galážovi, za jeho vstřícnost, ochotu poradit a pomoct. Materiály, které dodal a samotný jeho vhled do problematiky.

Obsah

Seznam obrázků

1. ÚVOD

V dnešní době 21. století je internet a programování stále důležitější a perspektivnější než kdykoliv dřív. Trend zatím ukazuje, že v budoucnosti to nebude jinak a právě naopak, bude potřeba více odborníků v daném oboru. Nejen odborníků, ale i laiků, kteří pracují v jiném oboru, ale dokážou si určité úkony automatizovat. To je možné právě díky programování.

V zahraničí, jako například Dánsko, je normální, že studenti chemických, stavebních a dalších oborů umí programovat v Matlabu a používají ho pro své seminární práce. Díky němu zpracovávají data a prezentují grafy.

To samé platí i o zpracování textu. Internet je čím dál více zakořeněn v našich životech, čerpáme odtud nesčetné množství informací, a proto je nutné s ním umět náležitě pracovat. Nejen uživatelsky, ale právě pro zmíněnou automatizaci úkonů. Program, který dané webové stránky dokáže zpracovat a vytáhnout nám z nich patřičná data, která chceme v podstatě na pár kliknutí je cenným pomocníkem v práci, nebo i koníčku, který děláme.

V této bakalářské práci je vytyčen úkol napsat program pro zpracování HTML stránek, získání textu či určitých bloků, které uživatel může chtít hledat. Bloky mohou být třeba všechny odkazy, emaily nebo jen tagy, které hledá. Toto bude zpracování do oddělené knihovny, kterou bude využívat tzv. GUI, které je pro uživatele více přívětivější na použití. Výhoda oddělení uživatelského rozhraní od funkční knihovny je ten, že funkcionalitu může využít později kdokoliv ke svým účelům. Knihovna totiž daná data zpracuje a programátor s nimi naloží dle svého uvážení. Knihovna je popsána v kapitole 4.

Program je napsán v programovacím jazyce *Python*, který čím dál více nabývá na popularitě. Jak na poli *Data Science* (zpracování informací, vědecké účely), tak na poli programování webových aplikací, okenních aplikací či her.

2. CÍLE

Cíl bakalářské práce je navrhnout, implementovat a otestovat aplikaci pro parsování a analýzu textu z webových stránek. Aplikace je napsána v programovacím jazyce *Python* a možnost generování reportů založených na technikách zpracování přirozeného jazyka (tzv. natural language processing). V rámci bakalářské práce je vytvoření a otestování knihovny pro parsování textu a také vytvoření několika ukázek použití technik zpracování přirozeného jazyka.

Samotný program má oddělené knihovny pro parsování a analýzu textu, jeho testy a taktéž knihovnu pro jeho GUI (Graphical User Inteface – Grafické Uživatelské Rozhraní). Díky tomu je práce s programem snazší, jelikož místo příkazů GUI obsahuje tlačítka a textové výstupy. Všechny výstupy lze lehce uložit do textových souborů pro další použití.

Výsledný zdrojový kód je nahrán na web GitHUB, který slouží programátorům pro sdílení kódu, programů a prací pro různé účely. Samotný kód je pak na adrese:
https://github.com/StepanOdstrcil/bachelor-web-parsing-analysis-app.

3. TEORETICKÁ ČÁST

Pro parsování textu a webových stránek je potřeba několik nástrojů. První věc je získat nějakým způsobem webovou stránku. To buď přes prohlížeč a nechat si zobrazit její zdrojový kód, nebo právě přes nějaký program, který to udělá za uživatele.

Dále je potřeba samotný program pro parsování, který daný kód webové stránky vezme, projede svým algoritmem a rozparsuje dle přání uživatele. S tím se samozřejmě pojí samotný vstup uživatele, který musí dát algoritmu nějak vědět, co přesně chce provést a jaký výstup očekává.

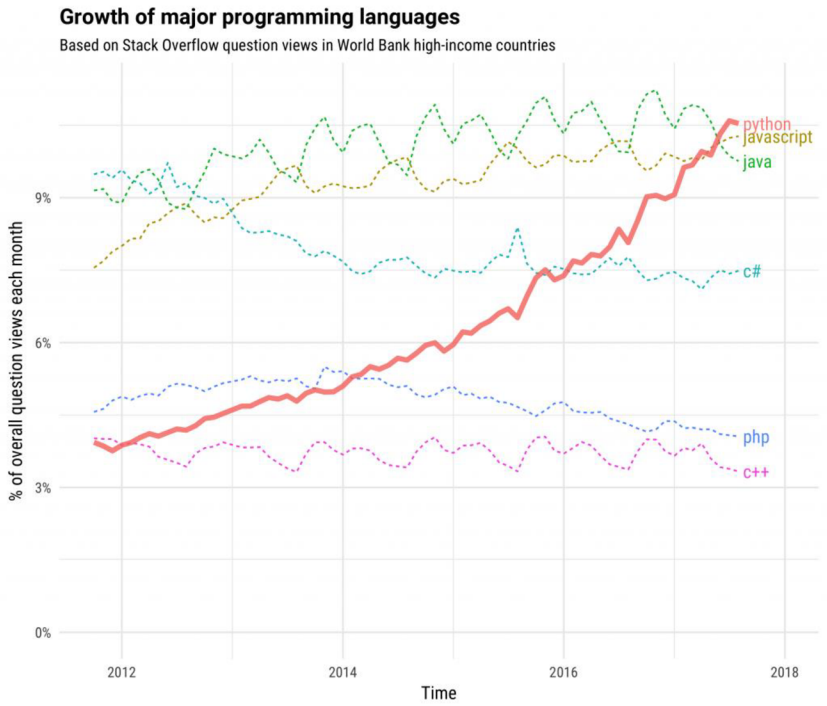
Už jen takový zevrubný popis naskýtá několik problémů k řešení. Proto byl zvolen programovací jazyk Python, který pro všechny tyto úkony již má vytvořené knihovny, které práci a tím pádem výsledek, usnadňují.

3.1 Python

Programovací jazyk Python byl vytvořen v roce 1991 Guido van Rossumem. Jedná se o skriptovací, za chodu interpretovaný jazyk. To znamená, že se nekompiluje předem a pak se spouští zkompilovaná aplikace napsaného programu, ale interpretuje za chodu, tzn. kompiluje se každý řádek kódu zvlášť po spuštění programu. To s sebou nese řadu výhod i nevýhod. Mezi výhody patří, že program je spustitelný ihned a stačí uživateli jen daný skript. Nevýhody jsou pak pomalejší výkon a nutnost přítomnost interpretru na daném zařízení [1].

Python získává na popularitě v průběhu času. Například dle webu www.stackoverflow.com, který slouží pro vývojáře na sdílení znalostí a řešení problémů, se počet dotazů na Python značně zvýšil, jak lze vidět na Obr. 1 a zatím trend ukazuje, že tomu v budoucnu nebude jinak [2].

Samotný jazyk nabízí vlastní prostředí pro instalaci knihoven, kterých je široká škála. Od základních, vylepšených funkcí, které již Python má, jako například lepší dotazování webových stránek, až po machine learning knihovny, určené pro strojové učení.



Obr. 1 Rostoucí popularita dotazů Pythonu na webu StackOverflow (převzato z [2]).

Python používá místo závorkování, jako je to u jiných programovacích jazyků, tabulátor nebo mezery. Takže to, že je nějaký kód v nějakém bloku, rozpoznává na základě jeho odsazení od zbytku kódu. To do jisté míry očistí kód od zbytečných znaků a textu navíc.

Dále umožňuje psát programy jak funkcionálně, tak i objektově. Je tudíž na uživateli, které paradigma využije a nemusí ihned balit svůj program do tříd nebo importovat hned knihovny.

3.1.1 Použití Pythonu v dnešní době

Python v dnešní době, v roce 2018, našel spoustu uplatnění napříč programovacím světem. Používá se pro psaní jednoduchých skriptů pro úkony v operačním systému, které uživateli zjednoduší práci se soubory a textem. Dále pro webové stránky, aplikace pro PC, aplikace pro android a také pro tzv. data science – zpracování dat ve vědecké sféře místo MATLABu. V poslední době vzniká i Micro Python, pomocí kterého se programují mikroprocesory [3].

3.2 Použité nástroje

V této bakalářské práci bylo použito několik nástrojů pro dosažení výsledku. Nejedná se jenom o Python, jako programovacího jazyka, ale i o program, v němž byl napsán, správu kódu, která umožňovala jeho sdílení a použité knihovny pro jeho správnou funkčnost.

3.2.1 PyCharm

PyCharm je profesionálnější nástroj pro programování v Pythonu. Je vydán ve dvou verzích, verze Community (zdarma) a Professional (placená). PyCharm nabízí spoustu funkcionalit. Některé jsou základní, jako je správa projektu, všech souborů, které do něj patří. Vytváření nových i s šablonou apod. Ukázka projektu v tomto IDE je vidět na Obr. 2.

Dále nabízí návrhy uživateli metod a funkcí, které uživatel již v projektu má, zvýraznění syntaxe nejen pro Python, ale i pro další jazyky, se kterými Python pracuje, jako HTML, CSS pro web, SQL pro databáze a další. Nabízí také možnosti založení projektu jako již hotového řešení, jako je Django nebo Flask pro webové stránky, Google App Engine atp… Má zakomponovaný terminál pro příkazy i výstup spuštěného programu, debugger pro krokování programu za chodu a odhalování chyb a/nebo obsahu proměnných. Nakonec má pak nástroje pro data science [4].

```
def get_all_following_links(self, level):
    """
    Gets all links defined by level. It gets all links in page. Then second level
    first level. Next level (third) is all links from all links at second level. E
    WARNING: It goes exponentially up!
    :param level: how deep should getting links go
    :return: list of lists of all links. Each list is level deeper. First is base
    links at first level.
    """
    following_links = [self._get_all_links(self._soup)]
    self._logger.info(f"Lvl:1/{level}|Links:1/1")

    for l in range(0, level - 1):
        links = following_links[l]

        current_level = l + 2
        current_link = 1
        found_links = []
        for link in links:
            if self.is_url_valid(link) and self._is_url_html(link, self._logger):
                found_links.extend(self._get_all_links_from_url(link))
            self._logger.info(f"Lvl:{current_level}/{level}|Links:{current_link}/{
            current_link += 1

        following_links.append(found_links)

    return [link for links in following_links for link in links]
```

Obr. 2 Ukázka IDE PyCharm.

3.2.2 GitHUB

GitHUB je webová služba, která slouží pro správu a sdílení kódu. Nabízí tzv. verzovací systém GIT. Ten umožňuje spravovat kód nejen jako samostatné soubory, ale i změny, které uživatel vytvořil. Takto vzniká historie celého projektu a každého souboru. Uživatel pak může tyto změny vracet a různě přehazovat mezi větvemi celého projektu [5].

3.3 Knihovna BeautifulSoup

Knihovna BeautifulSoup slouží právě pro parsování www stránek. Má řadu funkcí pro zpracování HTML a XML souborů, jejich procházení, parsování a filtrování jejich částí.

Na webu (dokumentaci) tvůrců této knihovny lze najít i několik příkladů, jak s knihovnou pracovat. Jeden takový příklad je použit a poupraven pro UNIT testy v této práci [6].

Zde, v této práci, je knihovna použita, avšak to nestačí. Je potřeba tyto univerzální funkce obalit vlastní třídou, která se bude starat o další funkce a bude právě používat funkce knihovny omezeným způsobem vhodným právě pro potřebnou funkcionalitu aplikace.

Knihovna pracuje s několika objekty HTML. Mezi základní a použité v této práci patří:

- *Tag* – jedná se o značku daného elementu v HTML. Jako příklad slouží odkaz, který má tag *a*, tag *input* sloužící pro textové pole vyplnitelné uživatelem. Dále *button*, což je tlačítko pro odesílání souborů.
- *Attributy* – zde se může jednat o jakoukoliv vnitřní vlastnost nějakého tagu. Může se jedná o *id*, což je unikátní identifikátor *tagu*. Dále například *style*, kde se píšou CSS styly pro daný tag. Dále *class*, kde se jedná o třídu CSS stylu daného tagu.

3.4 Natural Language Processing

Česky *Zpracování Přirozeného Jazyka* zažívá v dnešní době plné strojového učení velký boom. Celé strojové učení je moderní a zajisté budoucnosti na technologickém poli. Díky němu budou moct vznikat automatičtí komunikační roboti, kteří vyřídí objednávku, reklamaci anebo i lékařské vyšetření. Ve všech těch těchto případech je však důležitá komunikace s uživatelem. Proto stroj musí rozumět danému jazyku, sémantice řeči a významů slov, vět, idiomů a pořekadel. Proto je potřeba text řádně zpracovat.

3.4.1 Základní techniky zpracování textu

Existuje mnoho způsobů a technik pro zpracování jazyka. Od jednoduchých hledání slovíček v textu po rozpoznávání intonace hlasu v mluveném slovu.

Základní statistiky textu

Jako první za zmínku stojí asi logická analýza daného textu. Nejčastěji používaná slova, počet slov, vět, souvětí a další základní statistiky textu.

Rozpoznávání entit (Named Entity Recognition)

Jedná se o rozpoznání určitých slov, vět nebo uspořádání v textu, které něco znamenají. Jako příklad slouží třeba názvy lidí, jména, datum, čas, číslo, pořadí, název firem a známých značek. Mezi složitější patří potom pořekadla, idiomy apod. K tomuto

je potřeba většinou nějaký slovník těchto výrazů. Program pak daný text přečte a tyto výrazy hledá [7].

Modelování témat (Topic Modeling)

Jedná se o statistický model, který hledá sémantiku ve slovech a tím pádem dokáže získat témata, o kterých se v daném textu baví. To je pak zastoupeno v procentech. Jako příklad může sloužit, že pokud máme text o zvířatech, můžeme zjistit, jaké téma dominuje. Pokud je v textu často zmínka o psích granulích, kartáčích a hračkách, model poté ukáže větší procentuální zastoupení tématu *Pes*, než u ostatních zvířat, které se v textu také ukáží, ale ne v takové míře [8].

Sumarizace textu (Text Summarization)

Tato metoda je určená k vytvoření abstraktu z textu, abychom rychle a jednoduše zjistili, o čem text pojednává Funkce spočívá v najití nejvíce informativních vět z daného textu. Pro obrázky platí najití nejvíce reprezentativních obrázků. Dále pro kamerové záznamy se jedná o takové úseky, kde se toho děje nejvíce [9].

Podobnost textů (Word Movers)

Podobnost textů se dá získat pomocí metody anglicky „Word Movers“. Tato metoda porovná dva dané texty a vrátí nám reálné číslo od 0 do 1, kde nula znamená, že texty si nejsou vůbec podobné a jednička, že si jsou podobné na 100 %.

Vylepšení této metody tkví nejen v porovnání dvou textů jako celku, ale i jednotlivých slov. Každé slovo má svůj 300dimenzionální vektor, který když pomocí různých metod převedeme na 2D vektor a ten vyneseme do grafu, můžeme zjistit podobnost jednotlivých slov v našich textech a jak mají k sobě „blízko“.

3.5 Testování aplikací

Mezi správné praktiky psaní aplikací patří jejich testovatelnost. Toto hlavně platí u knihoven, které mají být pak používány pro jiné aplikace a tím pádem by měly být spolehlivé. Existuje několik typů testů. Každý se používá v jiné části aplikace a slouží k jinému účelu. Taktéž se jinak vyhodnocují výsledky těchto testů.

3.5.1 Základní typy testů

Zde je vypsáno několik typů testů. Aplikace této bakalářské práce obsahuje pouze unit (jednotkové) testy. Zbytek je popsán pouze pro ucelenější přehled testování aplikací.

3.5.1.1 Unit testy

Obvykle testují samotné knihovny. Nejsou tudíž plány pro implementaci knihoven nebo konkrétní aplikaci. Jednoduše řečeno jednotkové testy testují dané třídy, její metody anebo konkrétní funkce v dané knihovně. Projíždí jednu funkci po druhé, dávají jim

vstupy a zkoumají, zda daná funkce udělala, co měla. Na základě její výstupu a kontroly, zda je správný se pak test vyhodnotí jako úspěšný nebo neúspěšný [10].

Unit testy netestují připojení k databázi a správně by se k žádné databázi připojovat neměly. Pokud nějaká třída nebo funkce toto vyžaduje, nastává takzvané mockování/fakeování dat z databáze. To se obvykle dělá způsobem, že samotná třída má v sobě další třídu, která se stará o komunikaci s databází a o obstarávání dat. Takto před použitím (tedy testováním) metod třídy se může tato vnitřní třída nahradit jinou, podvrženou, která bude vracet data, která se pro test dopředu určí. Díky tomu se může i dobře rozhodnout o správnosti testu, protože se ví, jak má třída fungovat a jak má správně s podvrženými daty naložit. Takto se dá automaticky testovat mnoho případů dat z databáze, které mohou v reálné situaci nastat.

Správně by unit testy neměly využívat nejen data z databází, ale i jakékoliv konstanty, které záleží na konkrétní situaci. Jako příklad je třeba datum a čas. V mnoha případech se kontroluje nějaké datum s aktuálním datem. I toto by se v testu stát nemělo. Vše by mělo být pevně dané a tím pádem je třeba podvrhnout i nynější čas. Další příklad jsou soubory. Opět se v tomto případě využívá podvržených dat ze souboru, nebo se soubor v paměti vytvoří uměle.

Unit testy se řadí do tzv. *whitebox* testů. To znamená, že existuje přístup do testovaných funkcí a tříd a tím pádem se ví, co daná funkce dělá a jak funguje [10].

3.5.1.2 Akceptační testy

Tyto testy jsou opak od unit testů. Testují tzv. způsobem *blackbox*. Testují přímo funkčnost dané aplikace a jsou odstíněny od toho, jak je aplikace uvnitř napsaná. Mezi vlastnosti těchto testů patří třeba testování webů způsobem, že program simuluje uživatele a jeho klikání na tlačítka na stránce, vyplňování formulářů a testování výstupu aplikace [10].

Jelikož akceptační testy testují přímo aplikaci, zde již probíhá testování i s připojením na databázi. U velkých projektů se aplikační testy nachází na testovacích prostředích. To jsou kopie produkčního serveru, avšak s falešnými daty. To znamená, že testy mohou testovat i objednávky. Díky testovacímu serveru, objednávka nebude nikdy odeslána, protože od toho je testovací server [10].

3.5.1.3 Další testy

Mezi další testy patří testy **integrační**. Ty testují aplikace o vysoké komplexnosti. Jsou tím pádem rozděleny do menších částí a dohlíží, aby všechno do sebe zapadalo. Další testy jsou **systémové**. Ty zkouší další vlivy, jako například obsloužení tisíců zákazníků v jeden moment [10].

4. PRAKTICKÁ ČÁST

V praktické části práce byl hlavní úkol napsat program pro parsování a analýzu webových stránek. Program byl napsán v IDE PyCharm, který byl popsán v kapitole 3.2.1. Bylo použito moderních knihoven a postupů správného psaní kódu, a i přístupu ke sdílení tohoto kódu, k čemuž byla použita platforma GitHub, jehož stručný popis se nachází v kapitole 3.2.2. Odkaz do repozitáře je zde: https://github.com/StepanOdstrcil/bachelor-web-parsing-analysis-app.

Celá aplikace má v podstatě tři velké dílčí části. První slouží pro parsování webových stránek. Na to slouží hlavně knihovna Beatiful Soup, která mnoho problému s parsováním webových stránek řeší. Druhá část je natural language processing, zkráceně NLP a česky „Zpracování přirozeného jazyka“. Pro tyto praktiky byly použity dvě knihovny, každá se specializuje na určitou oblast NLP. Jsou to *Gensim* (https://radimrehurek.com/gensim) a *Spacy* (https://spacy.io). Dále knihovna, která využívá zmíněnou knihovnu *Spacy*, což je *Textacy* (https://chartbeat-labs.github.io/textacy/getting_started/quickstart.html). Třetí část aplikace je Grafické Uživatelské Rozhraní (GUI – Graphical User Interface). Zde se využívá knihovny PyQt5 (https://pypi.org/project/PyQt5/), který vychází z frameworku Qt. Ten je napsaný v C++. Zjednodušuje psaní multiplatformních aplikací (takže využitelná napříč Windows, Linuxem, Mac OS a jinde). Nabízí práci s okny, kontrolami jako jsou tlačítka, štítky, editační okna, rolovací seznamy atd. Zde má aplikace své vlastní okno, naimportovány již napsané knihovny pro parsování a analýzu, takže uživatel jen zadá patřičná data do daných polí a stiskne požadované tlačítko.

Samotné tyto knihovny potřebují další knihovny ke správnému fungování, všechny jsou rozepsány v kapitole 4.2.

4.1 Architektura aplikace

Aplikace je napsána objektově a v některých případech různé metody vrací další objekty, které zabalují vnitřní funkcionalitu daného výsledku. Metody jejich tříd jsou okomentovány, aby uživatel věděl, jak asi metoda funguje a co do ní má za argumenty dosadit.

Samotné třídy vlastnosti a metody drží privátně, pokud jsou uživateli nepotřebná. Toto zabraňuje změnění vnitřní struktury třídy a tím pádem jejímu špatnému fungování. Dále to také odstiňuje vnitřní pochody třídy a uživatel vidí pouze ty metody a vlastnosti, pro které byla třída napsána, aby splnila na venek svůj účel.

Všechny její metody jsou napsány a koncipovány tak, aby se žádný kód neopakoval a tím pádem změna jedné funkcionality se projevila ihned ve všech funkcionalitách, kde se daná metoda používá. To zvětšuje přehlednost v kódu a taky jeho udržitelnost.

Knihovna je rozdělena do dílčích celků nejen dle tříd, ale i dle knihoven, které zaobalují jednotlivé dílčí funkcionality celé aplikace. Dále umožňují uživateli si importovat přesně to, co zrovna chce používat. Takže nemusí importovat všechny třídy,

když chce například jen parsovat web. Třídy pro NLP by mu byly zbytečné. Díky tomu je aplikace přehlednější, udržitelnější a lépe rozšiřitelná.

4.2 Potřebné knihovny

Potřebné knihovny jsou nutné pro běh aplikace či knihovny, která řeší danou problematiku. Buď parsování webové stránky nebo NLP funkce. Bez těchto knihoven aplikace nepoběží a spadne při prvním volání chybějící knihovny.

Stručný výčet knihoven:

Requests

Knihovna určená pro dotazování dat z webových stránek. Je snazší na implementaci a práci s ní než integrovaná v pythonu. Umožňuje jak získávání dat z webové služby (get), tak posílání dat (post). Dále umožňuje nasimulovat webový prohlížeč. To je dobré zejména pro ty stránky, které zakazují přístup robotům a programům. Takto si webová služba bude myslet, že se jedná o normální prohlížeč nějakého uživatele.

Odkaz: http://docs.python-requests.org/en/master

BeautifulSoup

Tato knihovna se stará o parsování webových stránek. Umí číst, parsovat a upravovat HTML a XML formáty. Je jednoduchá na ovládání a používání. Tato knihovna je hlavní součástí webového parsování aplikace. Její funkcionalita je obalena do metod, které se dají snadněji používat uživatelem, a tak získat data přesně ta, která uživatel chce. Ten nemá o této knihovně ani zdání.

Odkaz: https://www.crummy.com/software/BeautifulSoup/bs4/doc

Lxml

Tato knihovna je zde nainstalována hlavně z důvodu, že obsahuje lepší parser pro čtení HTML knihovnou *Beautiful Soup*.

Odkaz: https://lxml.de

Spacy

Spacy je knihovna pro rozpoznávání entit v textu. To může být například číslo, datum, čas, jména lidí, organizací, měst a míst. Má několik jazykových mutací, kde bohužel čeština chybí. Pro zdejší testy byla tudíž použita angličtina a tím pádem anglické webové stránky.

Pro nainstalování anglického modelu je potřeba spustit skript, který toto umožní. Stačí otevřít příkazové okno jako administrátor ve složce, kde je nainstalován python a spustit tento příkaz (nainstaluje se středně velký model s názvem *en_core_web_md*.

Příkaz: *python.exe -m spacy download en_core_web_sm*

Odkaz: https://spacy.io

Gensim

Knihovna slouží pro *Topic modeling*, což je rozpoznání tématu, o kterém se mluví v daném textu. Dále pro sumarizaci textu.

Odkaz: https://radimrehurek.com/gensim

Nltk

Nltk slouží pro použití Gensimu pro *Topic modeling*, a to z toho důvodu, že dokáže rozeznat význam slov, jejich synonyma antonyma apod.

Odkaz: https://www.nltk.org

Textacy

Knihovna využívající *Spacy* obalující jeho funkcionalitu a přidává rozhraní pro analýzu textu. Využívá další knihovny, jako například Numpy. Ta je hojně využívána pro Data Science, jelikož nabízí rychlou práci s vektory, maticemi a jejich snadné používání. Textacy nabízí řadu řešení, jako například taktéž rozpoznání entit, hledání slov dle regexu, klíčová slova, termíny a metodu „Word Movers“, který byla popsána v kapitole 3.4.1.

Odkaz: https://chartbeat-labs.github.io/textacy/getting_started/quickstart.html

Matplotlib

Knihovna určená pro zobrazování grafů podle vzoru a syntaxe MatLABu. Používá se v podstatě stejně, jak grafy v tomto programu, akorát se využívá Python. Zde jde vidět snaha přiblížit Python i uživatelům, kteří přecházejí z MatLABu. Knihovna je využita pro NLP analýzu dvou textů.

Odkaz: https://matplotlib.org/

PyQt5

Knihovna určená pro tvorbu a spouštění GUI napsaných v Pythonu. Využívá se pro psaní multiplatformních aplikací, aby měly stejnou funkcionalitu GUI jak na Windows, tak na Linuxu či MacOS. Je využívána hlavně v C++.

Odkaz: https://pypi.org/project/PyQt5/

4.3 Rozložení projektu aplikace

Aplikace se konkrétně skládá z pěti knihoven. Jedna slouží čistě pro informování uživatele, co se děje. K takovým účelům slouží logování událostí. Jak chybných, tak těch informativních, aby uživatel věděl, v jakém stavu je daná operace. Takto může uživatel zjistit, kde třeba udělal chybu nebo co je špatně. Nebo také, jak dlouho daná operace bude ještě trvat. Logování probíhá tam, kde si uživatel zvolí. Může to být výstup souboru nebo prostá konzola pythonu.

Další je knihovna pro samotné parsování aplikace. Obsahuje jednu velkou třídu, která se o vše stará a obaluje danou funckionalitu čtení HTML stránek. Nabízí několik

metod, co s danou stránkou udělat a která data si uživatel přeje získat. Pracuje s knihovnou BeautifulSoup, jejíž metody využívá a jejíž instanci uchovává ve své paměti.

Třetí knihovna slouží pro NLP. Opět je vše obaleno v třídě, která se o NLP záležitosti stará. Kromě samotné hlavní třídy jsou zde i třídy pomocné, a to podle účelu, co si uživatel přeje s daným textem dělat. Pokud chce například rozpoznávání entit, tato NLP třída mu vrátí jinou, která v sobě uchovává data o tomto rozpoznávání. Uživatel má pak na výběr několik metod, které může nad tímto zpracováním použít a opět získat tak výsledek, který si přeje.

Čtvrtá knihovna je GUI pro danou aplikaci. To využívá již napsané a zmíněné knihovny a usnadňuje tak práci s celou aplikací. Takto je program přívětivý k uživateli, který může být lajk i s práci s počítačem, avšak díky intuitivním tlačítkům a popiskům dojde ke zdárnému konci.

Pátou knihovnou jsou samotné *Unit testy* celé aplikace. Ty jsou podrobněji popsány v kapitole 4.5.

4.3.1 Knihovna WebParsing

Knihovna *WebParsing* obsahuje pouze jediný soubor, *web_parser.py*, ve kterém se nachází třída *WebParser*. Tato poskytuje základní funkce, které uživatel chce pro parsování HTML stránek. Obsahuje také některé statické metody, které jsou privátní a slouží pouze pro lepší rozložení funkcionalit, aby byly napsány jednou na jednom místě.

Samotný konstruktor má dvě možnosti. Buď je bezparametrický, nebo přijme v parametru třídu *BeatifulSoup*, která v sobě má nahranou již nějakou HTML stránku pro parsování. Toto je využito pro získávání url linků ze webové stránky, navštívení těchto linků a získání dalších linků z této navštívené stránky. Více metoda *get_all_following_links* popsána v kapitole 4.3.1.1.

4.3.1.1 Veřejné metody

Knihovna *WebParsing* nabízí tyto metody:

load_page(url)

Tato metoda slouží pro nahrání webové stránky. Na vstupu je *url*, kterou stáhne knihovna *requests*. Zde se nejprve zjistí, zda je daná adresa vůbec webovou adresou a dále zda webová stránka je typu HTML. V případě, že obojí vyhovuje podmínce, získá se třída *BeautifulSoup* a uloží se do privátní proměnné obalující třídy *WebParser*.

get_all_tags ()

Metoda pro zanalyzování celé webové stránky a vrácení všech použitých tagů v ní. Uživatel, i když zná jazyk HTML, tak by musel ručně tuto stránku navštívit, prozkoumat její zdrojový kód a očima hledat, jaké tagy jsou zde použity. Může kterýkoliv přehlédnout nebo zapomenout. Tato metoda poslouží jako východisko pro výčet všech použitých tagů.

get_all_text()

Jednoduchá metoda pro vrácení čistého textu na webové stránce. Metoda ignoruje všechny tagy a formátování dané HTML.

get_items_by_tag(tag)

Získá všechen text obsažený v daném tagu zadaného uživatelem. Tag je zde míněno HTML značka. Například pokud chce uživatel všechen tučný text, zadá tag jako *strong*. Tato značka v HTML znamená, že daný text mezi otvírající a uzavírající značkou bude tučně. Metoda pak vrátí všechen text v těchto značkách, každý zvlášť v datovém typu tuple.

get_items_by_class(cls)

V konečném důsledku stejná funkcionalita, jako *get_items_by_tag(tag)*. Jediná funkce navíc je omezující podmínka třídy. V HTML může mít nějaký tag určitou třídu. Ta určuje různé styly toho tagu. Například zbarvení textu, velikost, ohraničení, pozici apod. Takto si uživatel může zvolit text určený právě touto třídou.

get_all_links()

Získá všechny url adresy na dané stránce. Url adresou může být i emailová adresa. Ty tato metoda ignoruje.

get_all_emails()

Získá všechny emailové adresy na webové stránce. Ignoruje to všechny normální url adresy.

get_all_following_links(level)

Získá všechny url adresy na dané stránce, navštíví tyto adresy a z daných, nově navštívených webových stránek získá opět všechny webové adresy. Uživatel si může zvolit až do jaké úrovně chce jít. První úroveň je daná stránka zadaná uživatelem, druhá je pak navštívení linků z první úrovně a zapsání nových linků. Třetí úroveň navštíví linky z druhé úrovně a opět zapíše. Linky stoupají geometrickou řadou a už třetí úroveň je obrovské množství linků.

Privátní statické metody

Další metody, které slouží pro správnou funkcionalitu třídy jsou privátní statické metody. Mezi ně patří kontrola, zda je daná webová adresa skutečně webovou adresou, a ne náhodným textem, zda je adresa HTML stránkou, vytvoření třídy *BeautifulSoup* z dané url adresy apod.

4.3.1.2 Příklady použití

Použití knihovny je relativně jednoduché. První je potřeba si vytvořit třídu pro samotné používání jejich metod.

```
wb = WebParser()
```

Následně je potřeba načíst webovou stránku do této třídy. Zde uveden příklad stránky *wikipedia*. Je důležité dát přes samotný string písmeno *r*, aby všechny znaky a symboly dané v url adrese byly zachovány přesně tak, jak jsou a neměly žádnou jinou funkci. Například \n by pak mohlo znamenat nový řádek, avšak v url adrese to znamená doslova tyto dva znaky za sebou.

```
wb.load_page(r"https://cs.wikipedia.org/wiki/Wiki")
```

Nyní stačí již volat veřejné metody a získat to, co si uživatel přeje. Zde je několik příkladů: získání všeho textu v tagu *p*, což je odstavec, získání textu ve třídě *tocnumber*. Získání všech linků, všech emailů a všech linků do úrovně druhé. Všechny tyto výsledky jsou v tomto příkladě rovnou vytisknuty, bez dalšího formátování. Metody samy nic netisknou a pouze vrací výsledek. Co s ním je už na uživateli.

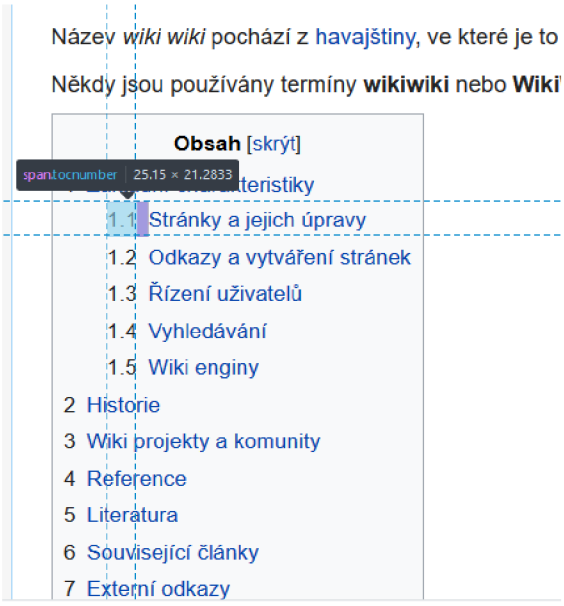
```
print(wb.get_items_by_tag("h2"))
print(wb.get_items_by_class("tocnumber"))
print(wb.get_all_links())
print(wb.get_all_emails())
print(wb.get_all_following_links(2))
```

Výsledky prvních dvou metod jsou:

```
('Obsah', 'Základní charakteristiky', 'Historie', 'Wiki projekty a
komunity', 'Reference', 'Literatura', 'Související články', 'Externí
odkazy', 'Navigační menu')

('1', '1.1', '1.2', '1.3', '1.4', '1.5', '2', '3', '4', '5', '6', '7')
```

První příkaz hledá všechny nadpisy druhé úrovně. Další najde všechen obsah dle třídy *tocnumber*. Pokud bychom se podívali na tuto stránku do jejího zdrojového kódu, zjistili bychom, že se jedná o číslování obsahu, jak lze vidět na Obr. 3.



Obr. 3 Třída pro číslování obsahu pro web wiki.

4.3.2 Knihovna NLP

Tato knihovna obsahuje dva soubory, první *nlp_service.py* s hlavní třídu celého NLP, *NLPService*. Ta obsahuje základní metody pro analýzu textu. Dále soubor *nlp_result.py*, jež obsahuje pomocné třídy, které některé metody z *NLPService* třídy vrátí. Tyto třídy zaobalují vnitřní funkcionalitu daného rozboru textu a umožňují uživateli s tímto výsledkem lépe a příjemněji pracovat.

Třída *NLPService* má buď bezparametrický konstruktor, nebo s argumentem textu pro zpracování. Ten se uloží do vnitřní vlastnosti a je pak použit pro další zpracování. Na rozdíl od *WebParser* třídy nemá metodu pro nahrání textu, neboť není potřeba tento text kontrolovat, zda je validní. Text je text a je na uživateli, který text chce zpracovat. Tudíž obsahuje veřejnou vlastnost *text*, do které může uživatel přímo nahrát předmět jeho zpracovávání.

Pomocné třídy pak mají vždy konstruktor s argumenty, do kterých si třída *NLPService* dosadí, co zrovna zpracovává. Tyto menší pomocné třídy totiž bez výsledku nemají žádný smysl, a proto je nutné ho do nich při vytváření nahrát.

4.3.2.1 Veřejné metody

Metody pro použití knihovny *NLPService* jsou následující:

get_named_entity_recognition()

Tato metoda získá rozpoznávání entit v textu. Nevrací žádnou pomocnou třídu, kterou by uživatel mohl používat nadále, ale tuple menších zabalovacích tříd pro lepší manipulaci s výsledky. Jedná se vždy o dvojici *text* a *označení*. *Text* je nalezený text v obsahu a *označení* vyjadřuje entitu, jako například osoba, číslo, datum atpod. Objekty v tuplu se neopakují.

get_topic_modeling_and_summarization()

Metoda vrátí pomocnou třídu *Gensim*, která je více popsána v kapitole 4.3.2.2. Metoda připraví text na zpracování knihovnou a inicializuje pomocnou třídu, kterou vrátí. S tou poté může uživatel pracovat a vytáhnout si z ní buď *Topic modeling* nebo *Sumarizaci textu*.

get_textacy_doc(text)

Metoda vrátí takzvaný „Doc“, což je třída, která si v sobě uchovává daný text a jeho analýzu. Sama si daný text zpracuje, naparsuje, rozčlení a nad touto instancí se pak volají její metody pro různé analýzy. Tato metoda je statická a jako vstup bere pouze text. Vrací tuple. Na prvním místě textacy doc a na druhém zpracovaný text pro jeho analýzu, aby uživatel věděl, jaký byl mezikrok pro zprocesování textu.

Metoda se přímo v GUI nevyužívá. Je veřejná, kdyby chtěl někdo knihovnu použít, jinak se využívá interně v rámci jiných metod uvnitř třídy NLPService.

get_n_grams()

N Gramy je různý počet po sobě jdoucích slovech. Tato metoda vrátí tuple, kde na prvním místě je tuple všech těchto slov vyskytujících se v daném textu. Na druhém místě zpracovaný text.

get_named_entity()

Jelikož knihovna textacy taktéž nabízí rozpoznání entit, je tato metoda do aplikace zařazena taktéž. Uživatel může porovnat více metod a vybrat si tu, která mu bude lépe vyhovovat. Metoda vrací tuple, kde na první místě je tuple všech rozpoznaných entit a na druhém zpracovaný text.

get_key_terms()

Metoda získá z daného textu různá klíčová slova a jejich váhu dle algoritmu SGrank, seřazeno dle ranku sestupně. Rank nabývá reálných hodnot od 0 až 1. Na první místě výsledného tuplu je opět tuple výsledků a na druhém zpracovaný text.

get_pos_regex()

Tato metoda nabízí získání slovních obratů, částí vět a slovních spojení na základě definovaného regexu v textacy knihovně. Na první místě tuplu je tuple těchto nálezů, na druhém zpracovaný text.

get_bag_of_terms()

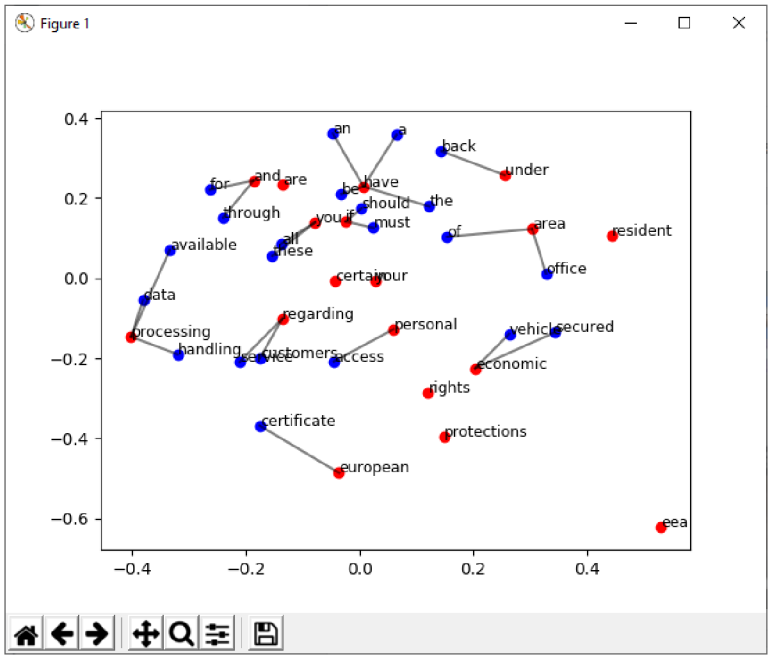
Získá z daného textu termíny a seřadí od nejvíce užívaných po nejméně užívané. Obsahuje vždy číslo užití a daný termín. Výsledek je opět jako tuple uvnitř obalujícího tuplu, který má na druhém místě zpracovaný text.

get_word_movers()

Statická metoda na porovnání dvou textů. Používá cosínovou vzdálenost mezi dvěma texty. Výsledek nabývá reálných hodnot od 0 do 1, kde nula značí, že texty nejsou vůbec podobné a jednička, že jsou naprosto shodné. Na prvním místě opět tento výsledek, na druhém text, kde jsou oba zpracované texty jdoucí po sobě.

show_word_movers_plot()

Tato metoda v sobě obsahuje upravený algoritmus word movers, kdy zobrazí daná slova v grafu a jejich vzdálenosti. Uživatel tak má možnost nejen kromě klasického word movers algoritmu zobrazit podobnost textů, ale i podobnost slov a výrazů v každém textu, jejich vzdálenosti a celkový obrázek jejich pozice ve 2D grafu.



Obr. 4 Ukázka 2D grafu při porovnávání dvou textů

4.3.2.2 Pomocné třídy

Knihovna NLP má dvě pomocné třídy.

NamedEntity

Tato třída slouží pouze pro lepší manipulaci s výsledky, pouze je zabaluje do dvou vlastností (*text*, *označení*) a také zajišťuje jejich jednoznačné označení. Díky tomu se tyto třídy budou nacházet v tuplu jen jedenkrát, pokud odkazují na stejnou kombinaci *textu* a *značky*.

Gensim

Třída slouží k uchování a získání výsledků pro *Topic modeling* a *Sumarizaci textu*. Obsahuje dvě metody, pro každý výsledek jinou.

První z nich je *get_topics(number_topic=5, number_words=4)*, která získá z knihovny *Gensim* témata a jejich procentuální vyjádření. Uživatel si taktéž může zvolit, kolik témat chce (přednastaveno na 5) a kolik chce slov ve výsledku (přednastaveno na 4).

Druhá metoda, *get_summarization()*, vrátí čistý text sumarizace textu. Zde není zapotřebí žádné obalování výsledku, protože samotný text je ten výsledek.

4.3.2.3 Příklady použití

Použití NLP knihovny je následující. Nejdříve je třeba inicializovat třídu *NLPService*. Zde v tomto příkladu je text získán z *WebParser*, z 15. odstavce www stránky https://en.wikipedia.org/wiki/Wiki. Je důležité upozornit, že stránka je nyní v angličtině. To z důvodu, že slovníky a celé knihovny *Spacy* a *Gensim* jsou v několika jazykových mutacích, čeština však do nich nepatří. V tomto odstavci se mluví o lincích, jakou mají syntaxi a pravidla pro wikipedii.

```
# Získání textu

wb = WebParser()
wb.load_page(r"https://en.wikipedia.org/wiki/Wiki")
text = wb.get_items_by_tag("p")[14]

# Inicializace třídy NLPService
nlp_service = NLPService(text)
```

Poté závisí, co uživatel chce za výsledek. V případě rozpoznávání entit, stačí zavolat metodu a nechat si vypsat výsledek. Díky pomocné třídě s výsledkem může pracovat i jinak, jako například ho třídit, seřadit a podobné. Jak *text*, tak *označení* má uložené uvnitř pomocné třídy. Všechny výsledky jsou rovnou tisknuty.

```
print(nlp_service.get_named_entity_recognition())
```

Výsledek pak vypadá v případě stránky wiki následovně:

```
(PERSON - WiKi, WORK_OF_ART - PopularMusic, ORG - CamelCase, ORG -
RichardWagner, PERSON - Richard Wagner, ORG - BeginnerQuestions,
PERSON - Wiki)
```

Následuje *Topic modeling* a *Sumarizace textu*. Na tu je tu pomocná třída, kterou lze získat metodou takto:

```
gensim = nlp_service.get_topic_modeling_and_summarization()
```

Nyní stačí zavolat nad instancí třídy *Gensim* metody pro danou činnost. Pro *Topic modeling* se zavolá metoda:

```
print(gensim.get_topics())
```

Výsledek této operace je následující:

```
[(0, '0.015*"richard" + 0.015*"reversal" + 0.015*"wagner" +
0.015*"richardwagner"'), (1, '0.054*"anchor" + 0.029*"reprocess" +
```

```
0.029*"reverting" + 0.029*"improve"'), (2, '0.079*"links" +
0.079*"camelcase" + 0.054*"create" + 0.029*"space"'), (3,
'0.054*"letters" + 0.029*"wikis" + 0.029*"capitalize" +
0.029*"spelling"'), (4, '0.053*"render" + 0.053*"camelcase" +
0.053*"example" + 0.053*"cause"')]
```

Pro *Sumarizaci textu* stačí zavolat metodu, která vrátí přímo text sumarizace.

```
print(gensim.get_summarization())
```

Výsledek:

```
Originally, most wikis[citation needed] used CamelCase to name pages
and create links.

These are produced by capitalizing words in a phrase and removing the
spaces between them (the word "CamelCase" is itself an example).
```

Jako ukázku získání klíčových slov poslouží tento text:

```
nlp_service.text = „Since the so-called 'statistical revolution' in
the late 1980s and mid 1990s, much Natural Language Processing
research has relied heavily on machine learning. Formerly, many
language-processing tasks typically involved the direct hand coding of
rules, which is not in general robust to natural language variation.
The machine-learning paradigm calls instead for using statistical
inference to automatically learn such rules through the analysis of
large corpora of typical real-world examples.“
```

Zadaný text nahrajeme do

```
key_terms, processed_text = nlp_service.get_key_terms()
```

Můžeme vidět, že máme k dispozici zpracovaný text. Ten si metoda zpracovala do této podoby:

```
since the so called statistical revolution in the late 1980s and mid
1990s much natural language processing research has relied heavily on
machine learning formerly many language processing tasks typically
involved the direct hand coding of rules which is not in general
robust to natural language variation the machine learning paradigm
calls instead for using statistical inference to automatically learn
such rules through the analysis of large corpora of typical real world
examples
```

Lze si všimnout, že všechna písmena jsou malými písmeny a chybí tam veškerá interpunkce.

Výsledek metody je:

```
(rule - 0.053859272473310314, machine - 0.049080775826764074, language
- 0.048601888374214926, statistical - 0.0407269733558363, natural -
0.039444225003814386, processing - 0.03822501581335402, world -
0.03715674162076043, real - 0.034785689429587575, typical -
0.03329984236476377, corpora - 0.03218779717067154)
```

Jelikož text se baví o machine learningu, dává tento výsledek smysl, nebo je alespoň logický a metoda nevrátila něco kompletně mimo toto téma.

4.3.3 Knihovna AdvancedLogging

Tato knihovna obsahuje pouze jediný soubor, *logger.py*, který v sobě má pouze jedinou třídu, *Logger*. Ta obsahuje metody pro logování do konzole. Obsahuje bezparametrický konstruktor nebo konstruktor s názvem události pro logování. Zde se obvykle uvádí jméno třídy nebo knihovny, která chce výstup zaznamenávat.

4.3.3.1 Veřejné metody

Veřejné metody jsou v podstatě jen logovací metody pro zápis výsledku. Obsahují vždy argument zprávy a pak volitelné argumenty, které se v této aplikaci nepoužívají. Lze zaznamenávat tyto události: *debug*, *info*, *warning*, *error*, *exception* a *critical*.

4.3.3.2 Příklady použití

Nejprve je potřeba vytvořit instanci třídy pro logování. Níže je uveden navíc i příklad, jak vložit název třídy, která chce logger používat. Tento přístup má tu výhodu, že kdyby se z nějakého důvodu volající třída přejmenovala, okamžitě se to projeví. To je lepší přístup než psát název jako string.

```
logger = Logger(self.__class__.__name__)
```

Zde je pak ukázka logování normální zprávy a výjimky.

```
logger.info(f"Nějaký text")
logger.exception(ex)
```

4.4 GUI

Velkou část aplikace tvoří GUI, které umožňuje uživateli snáze se zmíněnými knihovnami pracovat, a to až na takové úrovni, že o nich nemusí nic vědět. Stejně jako uživatel netuší, jak fungují programy word, excel a jiné, podobnou znalost nemusí mít uživatel při práci s GUI. Je samozřejmě nutné, aby věděl, jak daný program funguje a k čemu slouží.

GUI se skládá z několika tříd a oken. Každá plní svoji funkci a dílčí úkony jsou delegovány na místě, kde by je uživatel očekával. Je nutné si uvědomit, že jakýkoliv vývoj programu je takřka nekonečná práce a existují přímo specialisti, na UI (User Interface), neboli uživatelské rozhraní. Tito experti se zabývají veškerým pozicováním tlačítek, štítků, formulářů a celkovou přívětivostí daného zobrazení. Úkolem této bakalářské práce není hrát si na takového odborníka a jakékoliv (G)UI by se dalo několikrát vylepšit. V případě, kdy by bylo perfektní, vstupuje do hry pořekadlo „Tisíc lidí, tisíc chutí“.

4.4.1 Výchozí aplikace

Co se týče všech okenních aplikací, žádná nezačíná čistě hlavním oknem. Prvotní je takzvaný soubor s aplikací. Tento soubor je spouštěcí a obsahuje metody/funkce na spuštění samotného okna. Mezitím může inicializovat potřebné proměnné, načíst zdroje nutné k běhu programu a v případě nějaké chyby upozornit uživatele a vypnout spouštění programu.

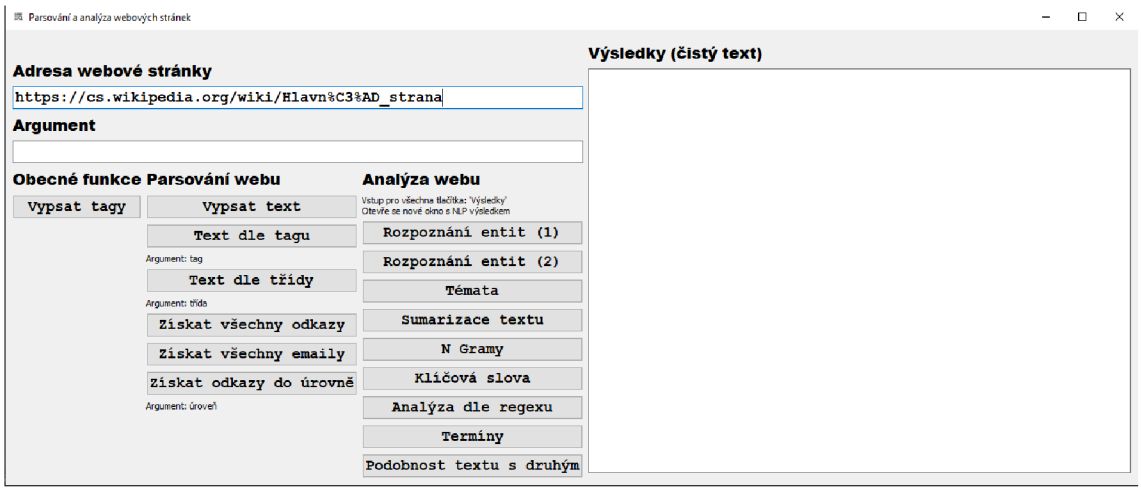
Takto je tomu i v tomto GUI pro aplikaci na parsování a analýzu webových stránek. V knihovně *GUI* se nachází soubor *App.py*, který dědí z třídy *QApplication* v kolekci *QtWidgets* z knihovny *PyQt5*. Lze si všimnou, že nedědí z okenní třídy, ale aplikační. Tato třída se postará o správné spuštění dané aplikace. Obsahuje konstruktor, který volá konstruktor předka a metodu *build()*, která využije proměnnou *main_form*, neboli *hlavní okno*. Tam vytvoří a uloží si instanci hlavního okna aplikace, které je popsáno v kapitole 4.4.2. Následně po uzavření hlavního okna ukončí celý chod programu příkazem:

```
sys.exit(self.exec_())
```

Lze vidět, že využívá systémovou funkci na ukončení programu a jako vstup volá vlastní metodu (umístěnou opět ve třídě předka). Díky tomu je práce s aplikací opravdu jednoduchá a programátor si díky vestavěným metodám v knihovně *PyQt* má značně ulehčenou práci.

4.4.2 Hlavní okno

Třída hlavního okna se nazývá *MainForm* a dědí ze třídy *QMainWindow* v kolekci *QtWidgets* z knihovny *PyQt5*. Tato třída obaluje většinu funkcionality programu. Hned v konstruktoru si inicializuje veškeré kontrolní prvky pro uživatele. Ty činí editační řádek pro webovou adresu, editační řádek pro argument dané funkcionality (jako například jaký tag chce uživatel v dané webové stránce najít), všechna tlačítka pro parsování a analýzu, a nakonec okno s výsledky dané funkce.



Obr. 5 Ukázka hlavního okna

Jelikož hlavní okno v sobě obsahuje většinu funkcionalit GUI, je dobré se mu věnovat podrobněji a popsat, jak funguje.

4.4.2.1 Vlastnosti a proměnné hlavního okna

Hlavní okno má několik vlastností a proměnných. Vlastnostmi je myšleno *properties*, což je v programování speciální druh proměnné. Není to jen prostá proměnná dané třídy, ale při jejím nastavování nebo získávání lze provádět kód, jako kdyby to byla metoda. Avšak místo volání metody programátor s vlastností pracuje jako s proměnnou, takže do ní přiřazuje hodnoty klasickým znaménkem „rovná se“ [12].

Jako proměnné slouží třídy pro parsování a analýzu textu. Jsou to instance tříd WebParser (kapitola 4.3.1) a NLPService (kapitola 4.3.2). Dále obsahuje proměnné potřebných kontrolek, aby si z nich mohla brát/zapisovat hodnoty. To jsou hlavně kontrolky jako editační okno pro webovou stránku, pro argument, štítek pro chybovou hlášku a editační okno pro výsledek. Dále je tu proměnná původní url webové stránky. Ta je využita při kontrole, zda se webová stránka změnila. Tato znalost slouží pro znovu načtení webové stránky.

Jako vlastnosti se používají hlavně možnosti snazšího získání hodnot z kontrolek. Ty se totiž získávají pomocí metod a je lepší to mít v jedné vlastnosti, která uvnitř sebe tuto metodu zavolá. Název vlastnosti je pak také příhodný. Takto se získává url, argument, výsledek a rozhodnutí, zda je url nová. Tato vlastnost porovná původní url s tou, která se nachází v kontrolce na webovou stránku. Více v kapitole 4.4.2.3.

4.4.2.2 Metody hlavního okna

Metody hlavního okna jsou především metody pro stisk tlačítek. Vždy, když se nějaké tlačítko stiskne, musí se jeho vnitřní metoda v *PyQt5* kontrolce propojit na danou metodu námi vytvořenou. Toto propojení probíhá v konstruktoru okna při vytváření tlačítek. Každé tlačítko má svou metodu stisknutí a v ní se volají příslušné instance tříd WebParser nebo NLPService a jejich metody, dle daného tlačítka. V případě, že je potřeba argument,

tak se načte z editačního řádku argumentu. Po zavolání příslušné metody třídy se vezme její výsledek a ten se zobrazí.

Lze si všimnou výhod oddělení logiky zpracovávání dat a komunikační. Třídy WebParser a NLPService vykonají svou práci, avšak nijak nevnucují své výsledky přímým vypisováním do konzole. Naopak výsledek jen vrátí a je na programátorovi, jak výsledky využije. Při testování se dají výsledky vypisovat do konzole a při využití GUI se výsledky zobrazí v daném okně. Třídy jsou tím pádem přenositelné, testovatelné a dělají jen jednu věc, pro kterou byly vytvořeny.

4.4.2.3 Wrappery hlavního okna

Při volání metod tlačítek je několik úkonů, které jsou pro každou metodu tlačítka stejné. Jedná se o tyto:

- Zjištění, zda se URL změnila
- Kontrola, zda je URL validní
- Zda je argument prázdný (záleží na typu metodu)
- Vymazání z chybového štítku předchozí chybu
- Zachycení chyby v případě pádu části kódu

Tyto metody se stanou vždy, nehledě na metody. Pro neopakování napsaného kódu a ani nevpisováni metod, které toto zajišťují do těla metod tlačítek byly použity takzvané *wrappery*. Jedná se o speciální druh metody/funkce, která obaluje zvnějšku jinou metodu/funkci. Využívá se v tomto případě tzv. dekorátorů, kde se napíše daný wrapper se zavináčem nad definici nějaké metody. Díky tomu se spustí kód wrapperu a on obalí danou funkci. Je to velmi efektivní, a hlavně se v kódu programátor lépe vyzná [12].

Jako příklad poslouží tento kus kódu [12]:

```
def p_decorate(func):
   def func_wrapper(name):
       return "<p>{0}</p>".format(func(name))
   return func_wrapper

@p_decorate
def get_text(name):
   return "Your name is: {0}. Hello!".format(name)

print(get_text("John"))
```

Zde funkce *get_text* bere jako argument jméno a vrátí větu s přivítáním daného jména. Obalující funkce *p_decorate* vloží okolo tohoto textu tagy odstavce a na konec. Zde vidíme uvnitř této funkce definici jiné funkce, která má stejný počet argumentů, jako funkce *get_text*, to proto, že tato funkce nahradí obalovanou funkci. Musí tudíž brát stejný počet argumentů. Uvnitř volá obalovanou funkci a provede ještě jiné úkony (přidáním tagů). Takto můžeme obalit jakoukoliv funkci tagy odstavce, aniž bychom zasahovali do původních funkcí [12].

Problematika wrapperů a jejich plné pochopení chce samozřejmě více studia a soustředění než dva krátké odstavce. Hlavně, jak to u programování bývá, je potřeba si tyto úkony vyzkoušet.

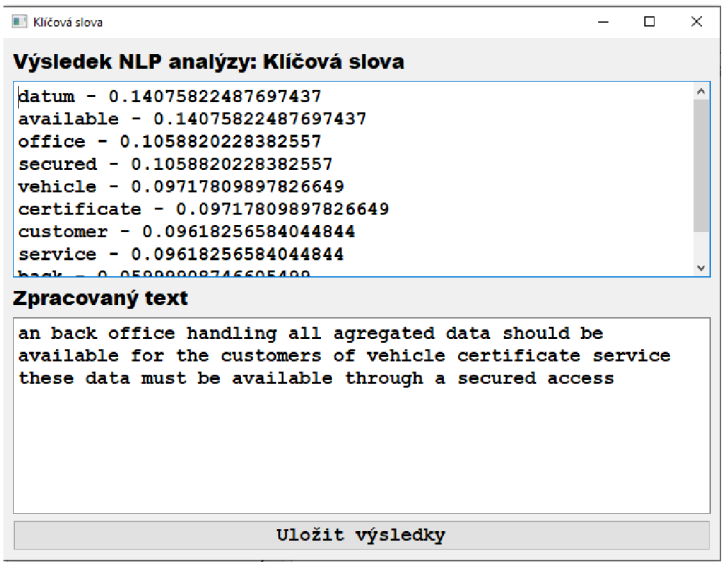
Dekorační metody slouží k odchycení problémů, kdy uživatel zadá něco špatně, nebo když se vyskytne chyba v běhu programu. Jako první dané metody obaluje wrapper pro odchycení výjimky (Exception). Ostatní metody sloužící pro kontrolu URL, argumentu výjimku vyvolají. Tato výjimka je zachycena nejvyšším wrapperem pro zachycení, který chybovou hlášku vypíše do štítku, umístěném nejvýše v celém okně. Takto všechny wrappery na kontrolu využívají přítomnost dalšího wrapperu a je zachován čistý chod aplikace.

4.4.3 Okno s výsledky NLP

Toto okno slouží pro výsledky NLP analýzy. Je tomu tak z důvodu, že bere jako argument text v editačním poli „Výsledky“ a tudíž výsledek NLP analýzy by tento původní text překryl. Proto veškeré tlačítko na NLP analýzu otevře nové okno s výsledkem. Takto uživatel bude mít pro každou analýzu nové okno a tím pádem se mu výsledky nebudou přepisovat editační formulář navzájem.

Okno s výsledky má hned dvě editační okna. První je zpracovaný text. Tato informace je zde z důvodu, aby uživatel věděl, jak se jeho text na vstupu zpracoval ještě před tím, než byl analyzován. Většinou se jedná o odstranění interpunkce a o změnu velkých písmen na malá. Avšak někdy toto zpracování může značně ovlivnit výsledek, a proto je lepší, aby zpracování textu byl uživatel informován.

Další výhoda tohoto okna je přítomnost tlačítka na uložení výsledků. Výsledek se vždy uloží do textového souboru s názvem analýzy a časem. Takto se nikdy nestane, že si několik analýz přepíšou stejný soubor. Dále v tomto souboru uloží i zpracovaný text před samotnou analýzou.

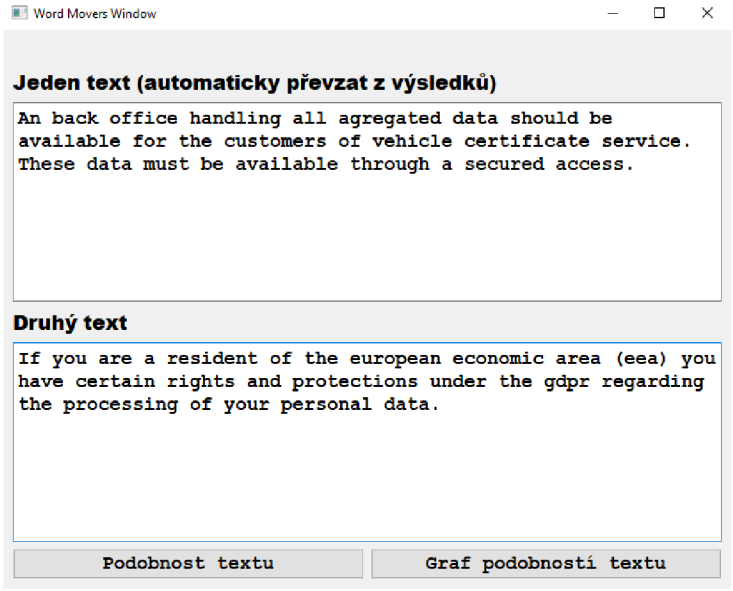


Obr. 6 Okno s výsledky pro analýzu klíčových slov

4.4.4 Okno pro analýzu word movers

Word movers okno je speciální, protože tato analýza zkoumá podobnost dvou textů. Tím pádem je potřeba dvou editačních polí a tím by se rušilo samotné okno hlavní aplikace, když by dané editační okno sloužilo jen pro jeden účel.

Okno je prosté, dvě textová pole pro vložení dvou textů. První text je předvyplněn z hlavního okna z pole výsledků. Uživatel však tento text může vymazat a libovolně měnit. Tato funkcionalita slouží pouze pro usnadnění práce. Dvě tlačítka, jedno pro celkovou podobnost textů. Tehdy se otevře klasické okno s výsledky popsané v kapitole 4.4.3. Druhé tlačítko slouží pro zobrazení grafu podobnosti slov v obou textech, jejich vzdálenosti a příbuznosti v 2D grafu. Metoda této analýzy je popsána podrobněji v kapitole 4.3.2.1.



Obr. 7 Ukázka okna „Word Movers“ s vloženými testovacími texty

4.4.5 Závěrem GUI

GUI je určitě vhodný způsob, jak distribuovat svoji aplikaci. Ulehčí to práci mnoha lidem a nepotřebují mít znalosti pythonu, jako samotný programátor. Navíc i pro samotného programátora je toto ulehčení, protože si časté chody programu může zautomatizovat do jednoho tlačítka, místo pamatování si všech příkazů, metod a funkcí.

4.5 Unit testy

Poslední knihovnou jsou pak unit testy. To se za knihovnu moc považovat nedá, protože neslouží k funkcionalitě aplikace, ale k otestování její správně funkčnosti. Kdyby se pak projekt rozrostl a někdo jiný chtěl změnit část kódu, díky unit testům zjistí, že tím sice vyřešil svůj problém, ale rozbil na nějakém jiném místě funkcionalitu. Toto ho varuje a taky ukáže, jak má daná metoda fungovat.

Unit testy se nachází v knihovně *UnitTests*, obsahují jeden soubor *WebParserUnitTests.py*. V něm se nachází jedna třída *WebParserTests*, která dědí z třídy pro testování aplikací *unittest.TestCase*. Díky tomu je testování aplikace snazší. Jakmile se totiž spustí tento soubor, třída *unittest.TestCase* si sama najde metody začínající na *test_*. Díky tomu dojde k automatickému spuštění testovacích scénářů. V případě, že nějaký test neprojde, dojde k chybové hlášce, které testy neprošly a co bylo jejich výsledkem.

Třída *NLPService* testována není, protože v podstatě obaluje funkcionalitu *Gensimu* a *Spacy*, které jsou s největší pravděpodobní otestovány jejich tvůrci. V tomto případě tím pádem testovat tuto třídu nemá moc smysl.

4.5.1 Pojmenování testovacích scénářů

Testovací scénáře (metody) jsou pojmenovány následujícím způsobem:

```
test__get_items_by_tag__with_valid_page__should_return_all_p_tags
```

Jako příklad je uvedena metoda pro testování třídy *WebParser*. Jednotlivé dílčí části jsou od sebe odděleny dvěma podtržítky, samotná slova pak jedním podtržítkem. První část je *test_*. To je pro samotnou otcovskou třídu, aby našla scénář a sama ho spustila. Další část je potom název testované metody. V tomto případě je to *get_items_by_tag*. Potom následuje podmínka, za které to bylo testování. V tomto případě se jedná o html stránku v řádném stavu. Není nijak schválně upravená, aby se otestovala odolnost metody proti chybám. Poslední část názvu říká očekávaný výsledek. Zde by měla metoda vrátit všechen text v odstavcích (tag *p*).

4.5.2 WebParser unit testy

Unit testy pro *WebParser*, obsahují 9 testovacích scénářů. Každý testuje určitou metodu třídy. Testování se provádí na falešné testovací stránce, která obsahuje mnoho různých tagů, tříd, linků a emailů. Díky tomu lze jednoduše zkontrolovat, co má být výstupem dané metody.

Dále obsahuje jednu privátní metodu pro inicializaci třídy *WebParser*. Důvodem této volby je snazší testování třídy, aby se v každém scénáři nemusela třída neustále inicializovat se stejnými parametry.

Jelikož je výstup znám, soubor pak obsahuje výstup jednotlivých unit testů pro porovnání výsledku samotného výstupu metody třídy *WebParsing*. V případě, že daná metoda třídy vrátí jiný výsledek, bude se tento výsledek lišit oproti očekávanému.

5. ZÁVĚR

V rámci aplikace pro analýzu a parsování webových stránek byly splněny cíle této bakalářské práce. Jako první bylo prozkoumáno pole knihoven pro *Python*, které by tuto práci zvládly nejlépe. Z toho se vybraly pro parsování knihovna *BeautifulSoup* a pro NLP knihovny *Spacy* a *Gensim*.

Aplikace byla rozdělena do několika knihoven, každá pro svou funkcionalitu, aby se daly jednoduše naimportovat uživatelem a použít pro daný úkol. Díky tomu se zbytečně nenačítají jiné funkce, které s tou, kterou chce uživatel použít, nemají, co dočinění. Takto byly vytvořeny dvě knihovny: *WebParsing* pro parsování textu a *NLP* pro zpracování přirozeného jazyka.

Knihovna *WebParsing* nabízí třídu *WebParser*, která nabízí několik metod pro získávání dat z webové stránky. Takto si uživatel může lehce navolit, co chce získat z obsahu. Patří mezi ně získání textu v určitém tagu (jako například nadpisy, odstavce, tučná písma apod.). Dále text v tagu, který má určitou třídu, která určuje vzhled a formátování obsahu. Následně lze získat všechny odkazy a emaily.

Knihovna *NLP* obsahuje třídu *NLPService*, díky které lze získat základní zpracování přirozeného jazyka. Je zde rozpoznávání entit. To vrátí z textu určitě entity, jako osoby, názvy firem, datum, čas, číslo apod. Dále pak lze získat pomocnou třídu, díky ní lze text zpracovat téma textu anebo jeho sumarizaci.

Samotné tyto třídy byly využity při psaní vlastního GUI pro celou aplikaci. Díky tomu je aplikace snadno ovladatelná i pro laika na poli Pythonu. Jako veškerý vývoj programu je pole na vylepšování GUI nedozírné a nikdy žádné UI nebude dokonalé. Proto taky zde existují experti, jejichž jedinou náplní práce je zkoumat UI aplikací a webových stránek, vylepšovat je a nabízet možná řešení, aby byly intuitivnější, snazší na používání a rozšiřování.

Nakonec byla otestována třída *WebParser* pomocí Unit testů. Díky tomu se vyzkoušela funkcionalita dané třídy na falešné stránce. Výsledky metod třídy byly porovnány s očekávanými a díky tomu šlo ověřit správnost jejich výsledků. Tyto automatické testy slouží jak k ověření, že třída funguje správně, tak ke kontrole, že pokud se v budoucnu něco změní, lze si ověřit, že se tím nerozbila původní funkcionalita.

Samotná aplikace je napsána dle moderních trendů na poli programování, mezi které patří neopakovat kód, objektově psát program, okomentované metody pro dokumentaci. Třídy jsou podrobeny automatickým testům. Samotný kód byl sdílen přes verzovací systém GitHUB, kde je nahrán na adrese: https://github.com/StepanOdstrcil/bachelor-web-parsing-analysis-app.

Samotné možnosti rozšíření aplikace jsou nedozírné. Jak bylo řečeno v předchozích kapitolách, vývoj softwaru je takřka nekonečný proces a platí to i zde. Jako první se nabízí rozšíření knihovny NLPService, která se stará o NLP analýzu textu. Zde by se dalo přidat nespočet funkcionalit a metod pro ještě větší a lepší statistiky na poli přirozeného zpracování jazyka. Další návrh by byl podpora více jazyků. Momentálně aplikace

podporuje pouze angličtinu, jako jazyk na zpracování. Vylepšit by se dala i část, kdy program parsuje webové stránky. Uživatel by si takto mohl ve výsledcích třeba filtrovat navíc dle vnořených parsovacích metodách. Tzn. by mohl použít nad daným výsledkem parsování jiné parsování. Nakonec přichází na řadu samotné GUI, kde by se implementovala všechna vylepšení v obou zmíněných knihovnách.

Literatura

[1] Kuhlman, Dave. "A Python Book: Beginning Python, Advanced Python, and Python Exercises". Archived from the original on 23 June 2012.

[2] The Incredible Growth of Python. Stack Overflow Blog [online]. New Your City: Stack Exchange, 2008, 6.9.2017 [cit. 2018-11-04]. Dostupné z: https://stackoverflow.blog/2017/09/06/incredible-growth-python

[3] MicroPython (na ESP8266/NodeMCU). Nauč se Python! [online]. Praha: 2016 [cit. 2019-05-20]. Dostupné z: https://naucse.python.cz/lessons/intro/micropython

[4] PyCharm features. Pycharm [online]. Praha, 2000 [cit. 2018-11-18]. Dostupné z: https://www.jetbrains.com/pycharm/features

[5] GitHUB Features. GitHUB [online]. Sunnyvale, 2008 [cit. 2018-11-18]. Dostupné z: https://github.com/features

[6] Beautiful Soup Documentation. Beautiful Soup [online]., 2004 [cit. 2018-11-20]. Dostupné z: https://www.crummy.com/software/BeautifulSoup/bs4/doc

[7] Elaine Marsh, Dennis Perzanowski, "MUC-7 Evaluation of IE Technology: Overview of Results", 29 April 1998

[8] BLEI, David M. (April 2012). "Introduction to Probabilistic Topic Models" (PDF). Comm. ACM. 55 (4): 77–84. doi:10.1145/2133806.2133826

[9] SHARMA, Mohit. Text Summarization. https://towardsdatascience.com [online], 15. 9. 2018 [cit. 2018-12-13]. Dostupné z: https://towardsdatascience.com/text-summarization-96079bf23e83

[10] Úvod do testování softwaru v C# .NET. ITNetwork [online]. Praha, [cit. 2018-11-20]. Dostupné z: https://www.itnetwork.cz/csharp/testovani/uvod-do-testovani-softwaru-v-csharp-net

[11] Properties vs. Getters and Setters. Python Course [online]. -, 2011 [cit. 2019-05-23]. Dostupné z: https://www.python-course.eu/python3_properties.php

[12] A guide to Python's function decorators. The Code Ship [online]., [cit. 2019-05-23]. Dostupné z: https://www.thecodeship.com/patterns/guide-to-python-function-decorators/

Seznam symbolů a zkratek

Zkratky:

NLP	Natural Language Proccessing (Přirozené zpracování jazyka)
ML	Machine Learning (Strojové učení)
GUI	Graphical User Inteface (Grafické uživatelské rozhraní)
UI	User Interface (Uživatelské rozhraní)

Seznam příloh

Příloha 1 - Obsah CD

K bakalářské práci bylo přiloženo CD s touto adresářovou strukturou:

- Soubor xodstr06.pdf – Tento dokument
- Složka BCWebParsingAnalysisApp – Soubor s aplikací napsané v PyCharmu
- Soubor BCWebParsingAnalysisApp/app.py – Výchozí spustitelný skript celé aplikace