VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

APLIKACE PRO SUMARIZACI TEXTU

APPLICATION FOR TEXT SUMMARIZATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JAKUB MIČKA**
AUTHOR

VEDOUCÍ PRÁCE **VLADIMÍR BARTÍK, Ing., Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Mička Jakub**
Program: Informační technologie
Název: **Aplikace pro sumarizaci textu**
Application for Text Summarization
Kategorie: Data mining

Zadání:

1. Prostudujte problematiku dolování v textu, zaměřte se na téma sumarizace textu.
2. Seznamte se s programovacím jazykem Python a knihovnami podporujícími práci s textem a jeho zpracování.
3. Analyzujte požadavky a navrhněte webovou aplikaci provádějící sumarizaci textu pomocí zvolené metody. Volbu metody konzultujte s vedoucím.
4. Navrženou aplikaci implementujte a proveďte testování.
5. Zhodnoťte dosažené výsledky a diskutujte další možné pokračování tohoto 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.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1-2, částečně bod 3.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 28. října 2018

Abstrakt

V této práci jsem se zaměřil na implementaci webové aplikace, která slouží jako prostředek pro automatickou tvorbu souhrnů v anglickém jazyce. Automatická tvorba souhrnů je v řešení prováděna pomocí metody TextRank a Latentní sémantické analýzy. Obě tyto metody jsou vylepšeny o rozpoznávání pojmenovaných entit. Přínosem této práce je zjištění, že využití rozpoznávání pojmenovaných entit u Latentní sémantické analýzy a především u metody TextRank, vede k vytváření kvalitnějších souhrnů. Tato kvalita souhrnů byla ověřena pomocí metrik ROUGE.

Abstract

This work is focused on an implementation a web application, which is a tool for automatic English text summarization. In result, automatic text summarization is made by TextRank and Latent semantic analysis method. Both of these methods are improved by named entity recognition. The main benefit of this work is proving that using the named entity recognition with Latent semantic analysis and especially with TextRank method leads to creation of higher quality summaries. This quality of the summaries was verified by ROUGE metrics.

Klíčová slova

sumarizace textu, zpracování přirozeného jazyka, ROUGE, Python, TextRank, Latentní sémantická analýza, lematizace, stematizace, pojmenované entity, stop slova, tokenizace

Keywords

text summarization, natural language processing, ROUGE, Python, TextRank, Latent semantic analysis, lemmatization, stemmatization, named entities, stop words, tokenization

Citace

MIČKA, Jakub. *Aplikace pro sumarizaci textu.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Vladimír Bartík, Ing., Ph.D.

Aplikace pro sumarizaci textu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Vladimíra Bartíka, Ing., Ph.D. Další informace mi poskytl Amir Hossein Esmaildokht Mamaghani, Ing. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jakub Mička
7. května 2019

Poděkování

Rád bych poděkoval vedoucímu práce panu Vladimíru Bartíkovi, Ing.,Ph.D. za podporu a cenné rady při tvorbě této práce. Dále pak Amiru Hossein Esmaildokht Mamaghani, Ing., který mi pomohl se znalostmi zpracování přirozeného jazyka. Kristýně Tulisové, Filipu Habánovi a Markovi Laryšovi děkuji za čas, který strávili nad tvorbou referenčních souhrnů pro účely testování. A nakonec děkuji Barboře Blažíčkové za korekci textu.

Obsah

Kapitola 1

Úvod

Tato práce je zaměřena na jeden z procesů zpracování přirozeného jazyka. Tímto procesem je automatická tvorba souhrnů, neboli sumarizace textu. Pro účely této práce byl vybrán ke zpracování a použití pouze text v anglickém jazyce.

Tvorba souhrnů je problém, který je znám už velice dlouhou dobu. Především v dnešní době je potřeba automatické tvorby souhrnů velká a tato potřeba bude i nadále růst. Většina infromací je totiž dostupná na internetu v textové podobě. Tento text je ve velké většině napsán v anglickém jazyce a z tohoto důvodu je tato práce založena na angličtině.

Těchto informací, zpráv a článků neustále přibývá. Pro člověka je velice náročné pročítat tak velké množství textu a vybírat pouze užitečné informace, které mu daný text poskytuje. Většina těchto článků totiž obsahuje mimo důležitých informací, které jsou zastoupeny menší částí, informace zbytečné, které slouží pouze jako výplň. Z tohoto důvodu jsem se rozhodl pro tvorbu webové aplikace, která tyto články a jednotlivé informace v nich obsažené dokáže zkrátit a předat uživateli pouze nejpodstatnější hodnotu z daného článku.

Hlavním cílem této práce je představit metody, které dokáží provádět automatickou tvorbu souhrnů a dále pak zhodnotit a porovnat výsledky těchto metod mezi sebou. Zhodnocení výsledků daných metod je důležité nejen pro člověka, který se pohybuje v oblasti IT, ale i pro člověka, který nemá v této oblasti žádné zkušenosti. Důvodem pro to je, že hodnocení výsledků metod představuje hodnocení kvality vytvořených souhrnů, což je velice podstatná informace. Mojí motivací je dosáhnutí co nejkvalitnějších souhrnů a to především z hlediska uživatele, tedy čtenáře daných textů. Tohoto výsledku chci dosáhnout pomocí úpravy či vylepšení metod, které jsou v oblasti zpracování přirozeného jazyka a sumarizace známé.

Tato práce je rozdělena do několika kapitol. Kapitola 2 popisuje obecné informace o tvorbě souhrnů, jednotlivé metody, které tyto souhrny vytváří a metody, které se využívají k vyhodnocení kvality jendotlivých souhrnů. Kapitola 3 obsahuje popis prvků, které by měla obsahovat výsledná aplikace. V kapitole 4 je popsána implementace metod pro tvorbu souhrnů, které využívá výsledná aplikace a problémy s nimi spojené. Dále tato kapitola obsahuje popis implementačního jazyka a použitých knihoven, implementaci předzpracování textu a nakonec implementaci samotné webové aplikace. V kapitole 5 je pak shrnuto testování implementovaných metod a zhodnocení jejich výsledků.

Kapitola 2

Sumarizace textu a její metody

Sumarizace textu je rozsáhlý problém k jehož vyřešení existuje řada metod, které jsou méně, či více úspěšné. Jelikož se sumarizace dělí do několika typů, musí existovat i mnoho různých metod k vyřešení každého typu zvlášť. V této kapitole jsou popsány základní informace o samotné sumarizaci, o jejím dělení, o přípravě textu k sumarizaci, metodách vhodných k vyřešení této problematiky a možnostech testování jednotlivých metod.

2.1 Obecné informace o sumarizaci textu

Tato část textu je zaměřena na automatickou sumarizaci textu, která spadá do oblasti zpracování přirozeného jazyka. Prvním využitím sumarizace v historii byly knihovní katalogy v roce 1674 a později, v roce 1898, se začala sumarizace používat pro tvorbu abstraktů vědeckých prací [4]. Tyto sumarizace byly prováděny manuálně do doby, než byl v padesátých letech 20. století vytvořen první sumarizační systém na počítač od firmy IBM [4]. Od tohoto okamžiku začal počet automatických sumarizátorů přibývat.

Co je to sumarizace

Sumarizace, českým názvem shrnutí, je metoda zpracování textu, při níž dochází ke zkrácení vstupního textu s co nejmenší ztrátou obsahově důležitých informací [17]. Všechny infromace, které jsou ve vstupním textu podstatné, se po dokončení celé sumarizace vyskytují i ve výsledném shrnutí. Velice podstatnou částí shrnutí je zkracování textu, s čímž bojují všechny sumarizační metody. Je důležité si uvědomit, že čím více chceme, aby výsledné shrnutí bylo kratší, tím více dochází k redukci podstatného obsahu. Z tohoto důvodu je důležité, aby sumarizační metody vhodně volily délku tohoto shrnutí s co nejmenší ztrátou podstatných zpráv.

Využití sumarizace v praxi

V dnešní době se na celém internetu nachází veliké množství volně dostupných textů. Většina z nich obsahuje pouze pár důležitých informací a zbytek je jenom přebytečná část, která neobohacuje čtenáře o žádné nové, ani zajímavé poznatky. V tomto případě je na místě využití sumarizačních metod k získání podstatných zpráv, aby člověk neztrácel čas čtením nepodstatných informací.

Mezi další využití, které je sice extrémní, patří tvorba názvu článku nebo knihy. V tomto případě se jedná o sumarizaci, která z velikého množství textu dokáže vybrat pouze pár slov a tím pojmenovat celý dokument.

Druhy sumarizací textu

Existuje velice mnoho rozdílných dělení. V této podkapitole jsou popsány pouze ty nejzákladnější a nejpodstatnější pro tuto práci. Aby bylo zřejmé, jak se jednotlivé druhy sumarizací od sebe odlišují, je vždy u každého typu uveden příklad, který tuto skutečnost názorně představuje.

Typy souhrnů se můžou dělit podle formy na extrakt a abstrakt. Extrakt je souhrn, který je tvořen pouze ze sekvencí slov původního textu. Tyto úseky mohou být buď jednotlivé fráze, věty nebo odstavce. Problémem extrakce je možnost špatné návaznosti sekvencí jedné na druhou a také možné nesrovnalosti a odlišnosti v samotném významu celého textu oproti textu původnímu. Tyto významové odlišnosti mohou vzniknout v důsledku nezačlenění určité věty, nebo vět, do výsledného souhrnu a tím změnit původní kontext celého dokumentu. [16]

Příklad 2.1.1 *Policie ČR zatkla dne 24. 5. 2018 v ulici Palackého dva pachatele. Tito pachatelé se dopustili krádeže osobního automobilu značky Fiat. Ten byl odcizen ve večerních hodinách předchozího dne ze sousední vesnice. Pachatelé udělali chybu, když nechali přes noc stát automobil zaparkovaný před jejich domem. Za krádež vozidla hrozí pachatelům až půl roku odnětí svobody.*

Řešení 2.1.1 *Policie ČR zatkla dne 24. 5. 2018 v ulici Palackého dva pachatele. Tito pachatelé se dopustili krádeže osobního automobilu značky Fiat. Za krádež vozidla hrozí pachatelům až půl roku odnětí svovody.*

Jak je vidět v řešení 2.1.1, konečný souhrn je vytvořen z nepozměněných vět, které se vyskytovaly v příkladu 2.1.1.

Abstrakt je narozdíl od extraktu souhrn, jenž nemusí obsahovat, ve většině případů neobsahuje, sekvenci slov z originálního dokumentu. V dnešní době je tento typ sumarizace stále obtížný. Jedním z důvodů je problém generování textu, který by byl po obsahové stránce stejný jako původní text, ale délka celého dokumentu by byla co nejkratší a samotné věty vyskytující se v něm by se od originálu lišily.[16]

Příklad 2.1.2 *Příklad pro abstrakt je stejný jako příklad 2.1.1.*

Řešení 2.1.2 *Policie zatkla 24. 5. 2018 dva pachatele, kteří spáchali čin odcizení osobního automobilu. Pachatelům hrozí půl roku odnětí svobody.*

Řešení 2.1.2 obsahuje souhrn příkladu 2.1.2, ve kterém nejsou jednotlivé věty totožné s větami daného příkladu.

Další možný způsob dělení je na základě rozsahu. A to na souhrny jednodokumentové a vícedokumentové. Jednodokumentový souhrn je vytvořen na základě jednoho dokumentu. Jako vstup pro jednodokumentovou sumarizační metodu je zvolen pouze jediný dokument, ze kterého vznikne jako výstup abstrakt, nebo extrakt.

Příklad 2.1.3 *Dne 24. 5. 2018 v 7 hodin ráno policie České republiky zadržela dva pachatele podezřelé z krádeže osobního automobilu. Osobní automobil byl podle očitých svědků*

odcizen 23. 5. 2018 okolo 18 hodiny v obci Křečkov ležící nedaleko města Poděbrady. Právě do Poděbrad s odcizeným autem pachatelé dojeli. Policisté ráno dalšího dne nalezli auto zaparkované před domem v ulici Palackého a počkali si zde na zloděje. Oběma pachatelům hrozí trest odnětí svobody na dobu 6 měsíců. Pachatelé byli zatím převezeni do vazby a čekají na rozhodnutí soudu.

Řešení 2.1.3 *Dne 24. 5. 2018 v 7 hodin ráno policie České republiky zadržela dva pachatele podezřelé z krádeže osobního automobilu. Osobní automobil byl podle očitých svědků odcizen 23. 5. 2018 okolo 18 hodiny v obci Křečkov ležící nedaleko města Poděbrady. Oběma pachatelům hrozí trest odnětí svobody na dobu 6 měsíců.*

Řešení 2.1.3 představuje jednodokumentovou extrakci textu z příkladu 2.1.3.

Vícedokumentová sumarizace je založena na vytvoření abstraktu, či extraktu z více dokumentů, které mají velice podobný obsah. Tímto vznikne pouze jediný text, který obsahuje všechny důležité poznatky a zprávy ze všech textů najednou. Jako příklad může být uvedeno několik článků, které se zabývají růstem bitcoinu ve stejném čase. Po vícedokumentové sumarizaci vznikne jediný dokument, který bude obsahovat všechny stejné, ale i individuální podstatné informace, které byly obsaženy v jednotlivých článcích.

Příklad 2.1.4 *Příklad pro extrakt vícedokumentové sumarizace je tvořen z příkladu 2.1.1 a 2.1.3.*

Řešení 2.1.4 *Policie zatkla 24. 5. 2018 dva pachatele, kteří spáchali čin odcizení osobního automobilu. Osobní automobil byl podle očitých svědků odcizen 23. 5. 2018 okolo 18 hodiny v obci Křečkov ležící nedaleko města Poděbrady. Pachatelé byli zatím převezeni do vazby a čekají na rozhodnutí soudu. Za krádež vozidla hrozí pachatelům až půl roku odnění svobody.*

Řešení 2.1.4 obsahuje vícedokumentový extrakt, který je tvořen větami z textu příkladů 2.1.3 a 2.1.1.

2.2 Příprava textu k sumarizaci

Příprava textu pro účely sumarizace je podstatná část, která by neměla být podceňována. Každá metoda má svoje vlastní požadavky na podobu přípravy textu a není možné obecně popsat všechny. V této kapitole budou popsány základní úpravy textu, které se využívají u převážné většiny sumarizačních metod.

Tokenizace

Podle [8] je tokenizace proces členění textu na slova, fráze nebo další smysluplné prvky, kterým se říká tokeny. Tento proces automaticky provádí program zvaný tokenizátor. Cílem tokenizátoru je rozdělení vět na jednotlivé tokeny. Seznam tokenů je poté použit jako vstup jiných metod pro přípravy textu k sumarizaci [8].

Je důležité si uvědomit, že provádění tokenizace je velice závislé na použitém jazyce daného textu. Pokud budeme uvažovat češtinu, francouzštinu nebo angličtinu, jednotlivá slova tohoto jazyka lze od sebe ve větě rozeznat pomocí bílých znaků, většinou mezery. U jiných jazyků, jako je čínština nebo thajština, není vždy jasné, kde se v dané větě nachází hranice mezi jednotlivými slovy. Tyto jazyky proto potřebují ke své tokenizaci další lexikální a morfologické informace.[8]

Stematizace

Stematizace je proces, při kterém dochází k určování základu slova, kterému se říká stem [6]. Příkladem stematizace můžou být slova „connection“,„connected“ a „connecting“, jejichž stem je slovo „connect“. Stematizace je založena na odstraňování předpon (prefixů) a přípon (sufixů) [6]. Při odstraňování afixů, což je souhrnný název pro předpony a přípony, může dojít k několika problémům:

1. Při odebrání prefixu nebo sufixu může dojít k vytvoření slova, které v daném jazyce neexistuje (destination → destin).
2. Slova, která si jsou podobná lexikálně, ale ne sémanticky, mohou být zkrácena na stejný stem (ledový a leda → led).
3. Pokud při ohýbání slova dochází ke změně písmen v kořeni, stem nemusí být správně určen (mouse a mice).

Lematizace

Lematizace je proces, při kterém dochází k určování základního tvaru slova, kterému se říká lemma [6]. Základem lematizace je vyhledání lemmatu daného slova v korpusu, souboru textů [6]. Výstupem lematizace, narozdíl od stematizace, je slovo, které v daném jazyce existuje. Lematizace je oproti stematizaci ve většině případů pomalejší. Jedním z příkladů použití lematizace mohou být slova „going“, „gone“ a „went“, jejichž nalezené lemma je „go“.

Stop slova

Mnoho slov se v textu vyskytuje s vysokou frekvencí, ale jejich vliv na samotný obsah textu je nulový. Tato slova je potřeba z textu odstratnit. Důvodem pro jejich odstranění je možné ovlivnění dalších metod kvůli jejich vysokému počtu výskytů v textu. V anglickém jazyce se jedná o slova typu „a“, „the“ nebo „this“. [8]

Seznam všech stop slov, které by se v daném textu měly odstranit je různorodý, nekonzistentní a těchto seznamů může vzniknou nekonečně mnoho [8]. Důvodem pro to je fakt, že v jednom případě jsou stop slova v seznamu slovy důležitými a v druhém případě jsou zase tyto slova nedůležitá a důležitá jsou jiná slova z jiného seznamu. Při odstraňování těchto slov je zapotřebí vybrat si správný seznam stop slov, nebo si pro specifický úkol vytvořit svůj vlastní.

Velká a malá písmena

Dalším důležitým krokem k předzpracování textu je rozhodnutí, zda budou veliká písmena zachována, nebo dojde k jejich přepsání na malé písmeno. Tento problém je potřeba řešit zvláště u počátečních písmen vět [7]. V některých případech při změně velkého písmene na malé k žádné sémantické chybě nejdojde, jako například v případě slov „Card“ a „card“ (v obou případech jde o stejný význam slova karta).

Někdy se ovšem může stát, že změna velikosti písmene ovlivní i sémantickou stránku daného slova. Tento případ ilustrují slova „rose“ a „Rose“. V prvním případě jde o význam květiny a v druhém pak slovo „Rose“ představuje vlastní jméno. [7]

Větné příznaky pro sumarizaci

Následující text byl převzat z [10] a popisuje základní možné příznaky, které lze použít pro výběr klíčových vět.

Počet výskytů slov: Slova, která jsou nejvíce významná pro obsah celého textu se podle statistik vyskytují v textu nejvíce. Klíčové věty, které by se měly zařadit do výsledného souhrnu by tudíž měly obsahovat co nejvíce těchto významných slov, v co největším počtu. Počet výskytů významného slova pro daný obsah dokumentu udává skóre pro danou větu ve které se nachází.

Výskyt věty v dokumentu: Důležité věty pro výsledný souhrn se často vyskytují na významných místech dokumentu. Mezi tyto významné části patří začátek a konec celého textu nebo odstavce.

Slova z nadpisů: Nadpis ve většině případů udává informaci o tom, o čem celý text pojednává. Téma obsahu dokumentu je pro sumarizaci velice důležité, proto by věty, které obsahují daná slova z nadpisů, měly být významnější než ostatní.

Vlastní jména: Věty, které obsahují vlastní jména, jako například jména míst, osob nebo organizací, jsou vhodnými kandidáty k výběru do konečného souhrnu.

Délka věty: Délka věty je jedním z indikátorů počtu informací, které se ve větě mohou nacházet. Pokud je věta velmi krátká, je větší pravděpodobnost, že tato věta v sobě nese méně informací než věta středního rozsahu. Naopak velice dlouhá věta není taktéž vhodná do výsledného souhrnu. Problém je určení vhodné délky věty.

Podobnost: Pro každou větu je dobré určit její podobnost s ostaními větami vyskytujícími se v dokumentu. Věty si mohou být podobné na základě slov, ze kterých jsou tvořeny, nebo podle jejich obsahu ze sémantického hlediska.

Klíčová slova: V některých případech je vyhledání klíčových slov velice podstatné. Některé z klíčových slov mohou pozitivně, ale i negativně ovlivnit výsledné skóre věty, ve které se právě nacházejí. Klíčová slova a jejich vliv mohou být zvolena pro každý dokument odlišná nebo mohou být použita obecná klíčová slova. K těmto obecným slovům patří například „in summary“, „in conclusion“ nebo „describes“.

Tyto příznaky patří k těm nejběžnějším a nejvíce používanějším. Nelze ovšem říci, že je nutné využít všechny tyto příznaky a nelze využít žádné jiné. Je hlavně velice důležité vybrat si správné příznaky, které pomohou metodám k řešení konkrétního problému.

2.3 Metody sumarizace textu

Tato kapitola se zabývá obecnými poznatky o sumarizačních metodách, algoritmech, které využívají a způsobem jakým vytváří výsledný souhrn. První dvě metody popsané v této kapitole budou následně využívány i v dalších kapitolách a bude na ně odkazováno. Ostatní metody budou představeny z důvodu většího proniknutí do problematiky sumarizace a poskytnutí důležitého přehledu o metodách, které se v dnešní době pomalu začínají čím dál více vylepšovat a využívat.

Latentní sémantická analýza

Latentní sémantická analýza, zkráceně LSA, je metoda založená na rozkladu matice, která je vytvořena ze vstupního textu [17]. Tato metoda nepotřebuje být trénována a spadá do kategorie algoritmů učení bez učitele, což znamená, že k zadaným vstupům nejsou potřebné žádné korektní výstupy [5].

Podstatnou vlastností této metody je využití vztahů mezi jednotlivými větami a předpoklad, že slova, která mají stejný význam, se vyskytují v podobných větách. Vztah mezi větami a slovy je analyzován pomocí řídké matice výskytů.

LSA se dá rozdělit do tří částí, které budou popsány níže. Těmito částmi jsou správné vytvoření matice výskytů, singulární rozklad, neboli singulární dekompozice, a výběr vět na základě singulárního rozkladu matice výskytů [5].

Matice výskytů

Obecně lze za vstup metody LSA považovat jakákoliv část dokumentu nebo textu. Každá matice se skládá z řádků a sloupců. Řádky matice výskytů jsou reprezentovány jednotlivými slovy. Každé slovo se může na řádku matice vyskytnout pouze jednou. Sloupce poté představují všechny věty vstupního dokumentu. Prvky dané matice mohou být reprezentovány mnoha způsoby. [5]

Jedním ze způsobů je využití frekvence slov. Každému slovu v dokumentu, které reprezentuje řádek matice výskytů, je vypočítán pro každou větu vstupního dokumentu počet jeho výskytů v konkrétní větě a tuto hodnotu poté reprezentuje prvek indexovaný danou větou (sloupcem) a slovem (řádkem). Tento odstavec je parafrázován z práce [5].

Takto vytvořená matice je řídká. Důvodem pro to je fakt, že slova se velice často vyskytují jenom v pár větách a v ostaních nikoliv. Tím pádem je častou hodnotou jednotlivých prvků matice hodnota 0. [17]

SVD

SVD, neboli singulární rozklad, je metoda využívaná k rozkladu matice A na součin tří matic $U\Sigma V^T$ [17]. Tento rozklad lze provést pro jakoukoliv $m \times n$ matici. Singulární rozklad pro matici A o velikosti $m \times n$, kde $m \geq n$ je definován takto (2.1):

$$A = U\Sigma V^T \tag{2.1}$$

Matice U je $m \times n$ sloupcově ortonormální matice, jejíž sloupce jsou nazývány levé singulární vektory [14]. Řádky této matice odpovídají slovům v dokumentu a sloupce jednotlivým tématům obsažených v daném textu.

Σ je $n \times n$ diagonální matice obsahující kladné singulární hodnoty, které jsou seřazeny od největšího po nejmenší [14]. Jelikož se nenulové hodnoty nevyskytují jinde, než na diagonále, je možné tuto matici zapsat jako vektor, kde jednotlivé prvky vektoru udávají váhu každého tématu vyskytujícího se v dokumentu.

Matice V je $n \times n$ ortonormální a její sloupce jsou nazývány pravé singulární vektory [14]. Každému sloupci této matice odpovídá věta z textu a každému řádku téma, které se vyskytuje v dokumentu [5].

Pokud dojde k situaci, kdy počet řádků matice A bude menší, než počet sloupců této matice ($m < n$), tedy matice A nesplňuje podmínku pro SVD, musí dojít k transponování této matice [17]. Tímto způsobem je možné SVD použít i na matici o rozměrech $m \times n$, která nesplňuje podmínku $m \geq n$.

Ortonormální matice je čtvercová matice A, jejíž transponovaná matice je zároveň inverzní a tedy platí $AA^T = I$, kde I je jednotková matice. Jednotková matice je matice, jejíž prvky ležící na diagonále nabývají hodnoty 1 a ostatní prvky hodnoty 0.

Výběr vět

Způsobů, metod a algoritmů, jak vybrat věty do výsledného souhrnu na základě singulárního rozkladu je mnoho.

Jedním z prvních přístupů popsaných v [5] je přístup Gong a Liu, kteří k výběru doporučují použít matici V^T. Jak už bylo zmíněno dříve, řádky této matice představují témata obsažená v dokumentu a sloupce jednotlivé věty. Hodnoty prvků matice v jednotlivých řádcích udávají, jak moc věta, reprezentovaná sloupcem, odpovídá konkrétnímu tématu daného řádku. Na základě tohoto tvrzení je pro každé téma vybrána věta s největší hodnotou. Konečný počet vět je dán parametrem. Nevýhody přístupu Gong a Liu jsou následující:

1. Pokud pro jeden koncept, neboli téma, bude mít více vět velikou hodnotu, bude vybrána pouze věta s hodnotou nejvyšší a ostatní budou zanedbány.

2. Pokud bude parametr počtu vět veliký, budou některé věty ve výsledném souhrnu méně důležité.

Dalším ze způsobů výběru vět je přístup Steinbergera a Jezeka popsaný v [5]. Tento přístup se snaží odstranit všechny nedostatky přístupu Gong a Liu. Steinberger a Jezek používají, stejně jako Gong a Liu, matici V^T. Místo výběru vět pro každé téma využívá tento přístup výpočtu délky vektoru věty pomocí hodnot matice Σ, která odpovídá síle tématu v dokumentu. Vzorec 2.2 zobrazuje výpočet délky vektoru dané věty. V_{ji} představuje prvek matice V^T s indexem (j, i), s_i je délka vektoru věty i a Σ_{jj} reprezentuje hodnotu na diagonále matice Σ. Hodnota n značí počet všech témat v dokumentu, tedy počet řádku matice V^T.

$$s_i = \sqrt{\sum_{j=1}^{n} V_{ji} \cdot \Sigma_{jj}} \tag{2.2}$$

Jednou z variant tohoto přístupu je využití druhých mocnin hodnot V_{ij} a Σ_{jj}. Tato varianta je popsána v [17]. Vzorec 2.3 využívá stejného značení jako vzorec 2.2.

$$s_i = \sqrt{\sum_{j=1}^{n} V_{ij}^2 \cdot \Sigma_{jj}^2} \tag{2.3}$$

Vzorce 2.2 a 2.3 platí pro podmínku matice A s rozměry $m \times n$, uvedenou v kapitole 2.3, kde $m \geq n$. Pokud $m < n$, v obou vzorcích je potřeba nahradit V_{ij} za U_{ij}, jež představuje prvek matice U^T. Nutnost použití matice U^T vychází z popisu metody LSA v kapitole 2.3, kdy z důvodu nesplnění podmínky $m \geq n$ matice A muselo dojít k jejímu transponování a tudíž i k záměně hodnot a významu výstupních matic U a V^T.

TextRank

Metoda TextRank je jedna z grafových metod založená na algoritmu PageRank, jenž slouží k ohodnocování významnosti uzlů hypertextové struktury Webu [16]. Aby bylo možné pochopit funkcionalitu metody TextRank, je nejprve nutné vysvětlit základy metody PageRank.

PageRank

PageRank je grafový algoritmus, který byl využíván firmou Google ve webových prohlížečích pro hodnocení významnosti internetových stránek [17]. K jeho využití při tvorbě sumarizace přispěla nepotřebnost tohoto algoritmu lingvistických znalostí a jazyková nezávislost [16]. PageRank patří k iteračním metodám, což znamená, že při každé nové iteraci je využita iterace předchozí.

Následující popis algoritmu je založen na textu [16]. Základem PageRanku je orientovaný graf $G = (V, H)$ tvořený množinou vrcholů V, kde každý vrchol reprezentuje stránku, a množinou hran H, $H \in V \times V$. Dále je definována množina $In(V_i)$ pro daný vrchol V_i, jež obsahuje všechny vrcholy, z kterých vede hrana do vrcholu V_i, a množina $Out(V_i)$, která obsahuje všechny vrcholy do kterých vede hrana z vrcholu V_i.

Algoritmus určuje pro každý vrchol V_i PR skóre, neboli PageRank, které udává pravděpodobnost výskytu v daném uzlu V_i v konkrétním čase. Výpočet tohoto skóre je vyobrazen v následující rovnici 2.4.

$$PR(V_i) = (1 - d)/N + d * \sum_{V_j \in In(V_i)} \frac{PR(V_j)}{|Out(V_j)|}, \tag{2.4}$$

kde N označuje počet všech vrcholů, d je parametr, který nabývá hodnot z intervalu $\langle 0, 1\rangle$ a představuje pravděpodobnost přechodu z vrcholu V_i přes některou z hran vedoucí z tohoto vrcholu. V reálném prostředí to znamená, že náhodný návštěvník stránky bude pokračovat, proklikem přes odkaz, na další stránce. Podle [16] se hodnota d doporučuje volit okolo 0.8. Hodnota $(1 - d)$ naopak udává pravděpodobnost, se kterou dojde k přechodu z vrcholu V_i na libovolný vrchol grafu, neboli náhodný uživatel začne na úplně nové stránce. Výraz $\frac{V_j}{|Out(V_j)|}$ reprezentuje pravděpodobnost přechodu z uzlu V_j do uzlu V_i. Suma tohoto výrazu představuje existenci více uzlů, ze kterých lze s určitou pravděpodobností provést přechod do uzlu V_i.

Jak už bylo zmíněno, algoritmus PageRank je iterační a PR skóre se v každé iteraci počítá pro každý vrchol. Počáteční hodnoty všech PR skór vrcholů jsou voleny stejně, tak aby jejich součet byl dohromady roven 1. Algoritmus je ukončen, pokud rozdíl všech hodnot PR a PR předchozí iterace dosáhne hodnoty menší, než je hodnota parametru ε. Hodnota ε je volena před zahájením výpočtu algoritmu.

Pokračování metody TextRank

V přechozím textu byl vysvětlen algoritmus PageRank, který je základem pro metodu s názvem TextRank. Vstupem do sumarizačního procesu TextRanku je dokument. Jednou z důležitých fází tohoto procesu je konstrukce grafu vazeb vět dokumentu, jak je popsáno v [16]. Každý vrchol tohoto grafu reprezentuje větu ze vstupního textu. Z každého vrcholu vede ohodnocená hrana do všech ostatních vrcholů, včetne sebe samého. Ohodnocení hrany udává podobnost vět, které daná hrana spojuje.

Výpočet podobností

Podobnost vět lze vypočítat více způsoby. Jedním z používaných vzorců je vzorec 2.5 [16]. V_i a V_j označují jednotlivé věty pro které se podobnost počítá. W_k označuje jedno konkrétní slovo. Čitatel zlomku představuje počet všech slov vyskytujících se zároveň v obou větách, pro které se podobnost počítá. Ve jmenovateli zlomku se počítá součet logaritmů délek obou

vět.

$$Podobnost(V_i, V_j) = \frac{|\{W_k; W_k \in V_i \wedge W_k \in V_j\}|}{\log(|V_i|) + \log(|V_j|)} \tag{2.5}$$

Dalším ze způsobů výpočtů podobnosti dvou vět je kosinova podobnost 2.6. Tato podobnost je založena na výpočtu úhlu mezi dvěma vektory. Každý vektor představuje jednu větu z dokumentu. Tvorba vektorů může být následující:

1. Věty, pro které má být spočítána podobnost, jsou vybrány.
2. Jsou vybrána všechna slova vyskytující se v obou větách a následně odstraněny všechny duplicity. Vznikne tak množina slov W.
3. Vektor je v lineární algebře matice o velikosti $m \times 1$ (sloupcový vektor) nebo $1 \times m$ (řádkový vektor). Hodnota m je dána velikostí množiny slov W. Dále budeme uvažovat použití řádkového vektoru.
4. Řádek matice reprezentuje jednu větu a sloupce všechna slova z dané množiny W. Pro každé slovo je pak spočítán počet jeho výskytů v dané větě. Tento proces je proveden pro obě dvě věty.

Po dokončení všech těchto bodů vzniknou dva vektory, na které lze aplikovat kosinova podobnost. Tato podobnost je počítána podle vzorce 2.6. Výsledkem jsou hodnoty v rozmezí od 1 (oba dva vektory, obě dvě věty, jsou totožné) do 0 (oba dva vektory, obě dvě věty, jsou naprosto odlišné).

$$KosinovaPodobnost(A, B) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2}\sqrt{\sum_{i=1}^{n} (B_i)^2}} \tag{2.6}$$

Po výpočtu podobností všech vět mezi sebou můžeme aplikovat algoritmus PageRank. Změna oproti původní verzi je ta, že vrcholy představují jednotlivé věty dokumentu, hrany jsou ohodnocené podobností a graf není orientovaný. S neorientovaností grafu se lze vypořádat rovností množin $In(V_i)$ a $Out(V_i)$, tedy $In(V_i) = Out(V_i)$. Ohodnocení hran, tedy podobnost vět V_i a V_j, které jsou spojeny touto hranou, je označováno jako váha w_{ij}. Je tedy nutné původní rovnici 2.4 patřičně upravit. Upravená rovnice 2.7 je rozšířena právě o podobnosti vět. [16]

$$PR(V_i) = (1 - d)/N + d * \sum_{V_j \in In(V_i)} w_{ij} \frac{PR(V_j)}{\sum_{V_k \in Out(V_j)} w_{kj}} \tag{2.7}$$

Samotný algoritmus PageRank není změněn. Na začátku jsou hodnoty všech vrcholů nastaveny na $1/N$, kde N je počet vrcholů. V každé iteraci dochází k výpočtu PR skóre pro každou větu, vrchol grafu. Na konci iterace dojde k zjištění, zda změna nějaké z hodnot vrcholů oproti předchozí iteraci je menší než zadaná mez (ε). Pokud ano, výpočet je ukončen [16]. V opačném případě dochází k další iteraci. Do výsledného souhrnu jsou poté vybrány věty s nejvyšším ohodnocením, tedy nejvyšším PR skóre [16]. Počet vět souhrnu je zadán jako parametr.

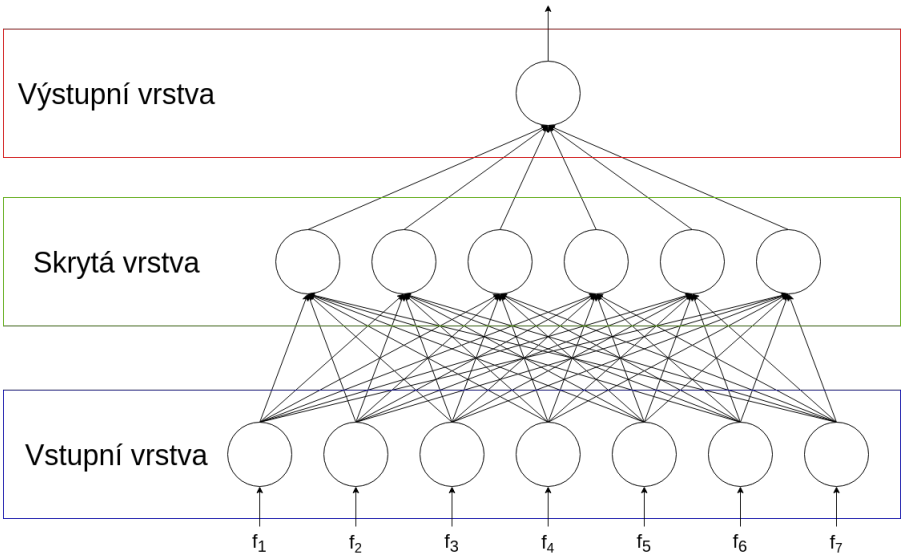
Další metody

Neuronové sítě

Jedny z nejnovějších metod pro sumarizaci textu jsou neuronové sítě. V této podkapitole nebude popsána teorie neuronových sítí obecně, ale způsob jejich využití v oblasti tvorby souhrnu.

Neuronové sítě umožňují vytvářet extrakt, ale zároveň dovolují experimentování s vytvářením abstraktu. Vytvoření exktraktu pomocí neuronové sítě může být rozděleno na 3 důležité kroky. Mezi tyto kroky patří trénování modelu, fúze příznaků a výběr vět [9].

Jednou z nevýhod této metody je nutnost trénování modelu neuronové sítě. Trénování spočívá v naučení neuronové sítě ropoznávat věty, které by měly být zahrnuty do výsledného souhrnu. S trénováním souvisí potřeba trénovacího korpusu, což jsou trénovací data potřebná k natrénování daného modelu. Trénovací sada, neboli korpus, je tvořena množinou dokumentů. O každé větě z této množiny je rozhodnuto, věta je označena, zda patří, nebo nepatří do výsledného souhrnu. Toto rozohdnutí a označení je prováděno člověkem. [9]



Obrázek 2.1: Vizualizace neuronové sítě pro tvorbu souhrnu

V [9] je využita třívstvá neuronová síť typu feed-forward. Tato neuronová síť, zobrazená na obrázku 2.1, se skládá ze sedmi neuronů ve vstupní vrstvě, šesti neuronů ve skryté vrstvě a jednoho neuronu ve výstupní vrstvě. Cílem trénování sítě je dosažení globálního minima energetické funkce, která je kombinací chybové a penalizační funkce. Pro dosažení globálního minima energetické funkce je použita optimalizační metoda konjugovaných gradientů.

Dalším krokem pro dosažení extraktu je fúze příznaků. V [9] je popsáno sedm hlavních příznaků vět, které mohou být pozměněny nebo přidány zcela nové. Všech sedm příznaků je zobrazeno v tabulce 2.1. Značení f označuje příznak (feature). Každá věta dokumentu je reprezentována vektorem $[f_1, f_2, ..., f_7]$ těchto příznaků.

Pro příznak f_6, počet tematických slov ve větě, je nejdříve proveden výběr deseti nejčastěji vyskujících se slov v dokumentu, které představují tematická slova [9].

Po trénování se neuronová síť naučí příznaky, které musí obsahovat vhodné věty výsledného souhrnu. Spojení mezi neurony neuronové sítě, které mají po trénování velice nízké

Tabulka 2.1: Větné příznaky pro neuronovou síť

f_1	Odstavec následující za nadpisem
f_2	Umístění odstavce v dokumentu
f_3	Umístění věty v odstavci
f_4	První věta odstavce
f_5	Délka věty
f_6	Počet tematických slov ve větě
f_7	Počet nacházejících se slov z nadpisu ve větě

váhy mohou být zanedbány, tudíž odstraněny bez ovlivnění správného fungování neuronové sítě. Pokud po odstranění těchto spojení mezi neurony zůstane neuron bez jakékoliv vazby s přechozí vrstvou, může být tento neuron odstraněn. Zanedbání těchto vezeb patří do části fúze příznaků. [9]

Posledním krokem je výběr vhodných vět. Po natrénování modelu neuronové sítě a odstranění zbytečných spojení a neuronů dokáže tato síť sama ohodnotit věty a odlišit, které věty jsou podstatné pro výsledný souhrn a které nikoliv [9].

Další

Neuronové sítě nejsou jedinou metodou, která je vylepšována a zdokonalována z důvodu snahy o zhotovení dokonalého abstraktu. Některé z těchto metod jsou velice náročné a využívají lingvistiku jako hlavní prostředek pro tvorbu abstraktu. Tyto metody se snaží o dokonalé pochopení kontextu vět, významu jednotlivých slov a provedení důkladné sémantické analýzy celého textu.

2.4 Metody pro vyhodnocování sumarizace textu

Metod pro vyhodnocení souhrnu vygenerovaného počítačem není mnoho. Tvorba souhrnu je velmi individuální a subjektivní. Každý v textu dokážet najít podstatné věci, které ostatním lidem důležité vůbec nepřipadají. Z tohoto důvodu je samotné vyhodnocení, zda je daný vytvořený souhrn dobrý nebo ne, stejně obtížný problém, jako provedení sumarizace, ne-li obtížnější.

K nejadekvátnějším posouzením kvality souhrnu stále patří lidský faktor. I když každý člověk hodnotí souhrn velice subjektivně, dokáže posoudit z hlediska návaznosti textu a jednotlivých významů vět zda daný souhrn pojednává o stejné informaci, která byla obsažena v původním textu. Problémem tohoto posuzování je časová náročnost. Člověk nedokáže provést tisíce posudků za hodinu. Z tohoto důvodu je potřeba nalézt jiné řešení, které by tuto lidskou práci mohlo nahradit.

Řešením je automatické vyhodnocování kvality souhrnů. Toto automatické vyhodnocování provádí počítač pomocí nástroje, který je k této činnosti přizpůsoben. Mezi takovýto nástroj patří ROUGE, který sice neprovádí hlubokou sémentickou analýzu textu, ale je pro tento účel velice rozšířen a dosahuje kvalitních výsledků (viz [12]).

Přesnost, Úplnost a F-skóre

Podle [15] patří mezi hlavní metriky, nejen pro vyhodnocování kvality souhrnů, přesnost (*precision*), úplnost (*recall*) a F-skóre (*F-score*). Následující vystvětlení těchto pojmů bude vztaženo k sumarizaci textu.

Mějme kandidátní souhrn a množinu referenčních souhrnů. Přesnost vyjadřuje, jak velká část kandidátního souhrnu obsahuje relevantní informace (vztaženo k množine referenčních souhrnů). Přesnost je poté počítána jako počet vět vyskytující se zároveň v kandidátním souhrnu a množině ideálních souhrnů poděleno počtem vět v množině referenčních souhrnů [15].

Úplnost vyjadřuje jaký zlomek množiny referenčních souhrnů je obsažen v kandidátním souhrnu. Její výpočet je proveden jako počet vět vyskytující se zároveň v kandidátním souhrnu a množině ideálních souhrnů poděleno počtem vět v kandidátním souhrnu [15]. F-skóre je kombinací metrik přesnosti a úplnosti [15]. Udává jak moc je kandidátní souhrn kvalitní. Jeho výpočet má několik variant. Základním z nich je výpočet harmonického průměru přesnosti a úplnosti. Ten je znázorněn pomocí rovnice 2.8 [15].

$$\text{F-score} = \frac{2 * recall * precision}{recall + precision} \tag{2.8}$$

Dalším ze způsobů výpočtu F-skóre je váhování úplnosti a přesnosti (rovnice 2.9). Hodnota β je váhový faktor, jenž vyzdvihuje hodnotu úplnosti, pokud $\beta > 1$ nebo vyzdvihuje hodnotu přesnosti pokud $\beta < 1$ [15].

$$\text{F-score} = \frac{(\beta^2 + 1) * recall * precision}{recall + precision * \beta^2} \tag{2.9}$$

ROUGE

ROUGE je akronymem anglického názvu *Recall-Oriented Understudy for Gisting Evaluation*. Je to balíček, který zahrnuje metriky pro automatickou kontrolu kvality souhrnu pomocí jeho porovnání s ideálními souhrny vytvořenými lidmi [12]. Většina těchto metrik je založena na počítání společných n-gramů, sekvencí slov či dvojic slov souhrnu vygenerovaného počítačem a souhrnů vytvořených člověkem [12]. Jednotlivé metriky ROUGE využívají metrik úplnosti, přesnosti a F-skóre, ale jejich výpočty jsou u každé ROUGE metriky odlišné. Mezi metriky ROUGE patří například ROUGE-N, ROUGE-L, ROUGE-W a ROUGE-S. Následně budou tyto metriky vysvětleny podrobněji.

ROUGE-N

Metrika je založena na výpočtu úplnosti podle výskytů n-gramů v kandidátním souhrnu a množině ideálních souhrnů [12]. Jelikož N v názvu metriky ROUGE-N reprezentuje přirozené číslo udávající velikost n-gramů, existuje nekonečně mnoho variant této metriky. Pro příklad lze uvést metrika ROUGE-1, která pracuje s 1-gramy (unigramy), ROUGE-2 pracuje s 2-gramy atd. Metrika ROUGE-N je počítána následovně:

$$\text{ROUGE-N} = \frac{\sum_{S \in \{ReferenceSummaries\}} \sum_{\text{n-gram} \in S} Count_{match}(\text{n-gram})}{\sum_{S \in \{ReferenceSummaries\}} \sum_{\text{n-gram} \in S} Count(\text{n-gram})} \tag{2.10}$$

S v rovnici představuje větu nacházející se v množině referenčních souhrnů, která je v rovnici označena jako $\{ReferenceSummaries\}$. Čitatel provádí výpočet pro nalezení počtu výskytů všech n-gramů, které se vyskytují v kandidátním souhrnu a zároveň ve větách z množiny referenčních souhrnů. Počet výskytů určitého n-gramu v jednom referenčním souhrnu je omezen počtem jeho výskytů v souhrnu kandidátním. Například, pokud se určitý n-gram vyskytuje v kandidátním souhrnu pětkrát a pro tento souhrn existují dva referenční souhrny, kde v prvním se daný n-gram vyskytuje dvakrát a v druhém osmkrát, je výsledný počet výskytu n-gramu v obou referenčních souhrnech roven sedmi. Jelikož omezující hodnotou je hodnota pět, v prvním souhrnu se započítají oba dva výskyty a ve druhém pouze pět, z důvodu toho, že počet výskytů daného n-gramu převyšuje omezující hodnotu. Ve jmenovateli je počítán počet všech n-gramů, které se vyskytují v množině referenčních souhrnů. [13]

Pokud je využívána metrika ROUGE-1, tak ve jmenovateli je počítán počet všech slov, které se vyskytují ve všech větách všech referenčních souhrnů. Hodnota jmenovatele je vždy zvýšena s přidáním nového referenčního souhrnu [12]. Pokaždé, když je přidán nový souhrn do množiny referenčních souhrnů, provádí metodika ROUGE-N vyhodnocení na základě nových aspektů [12]. To je důležité z důvodu toho, že pro jeden text neexistuje pouze jeden jediný souhrn, ale řada variant a alternativ.

Metrika ROUGE-N upřednostňuje kandidátní souhrny jejichž n-gramy se vyskytují ve více referenčních souhrnech [12]. To je velice intuitivní, protože po kandidátním souhrnu požadujeme, aby byl co nejvíce podobný souhrnům referenčním [12].

V některých článcích (například [12]) rovnice 2.10 pracuje pouze s jedním referenčním souhrnem [13]. Pokud je tedy v množině referenčních souhrnů více, jak jeden souhrn, je na každý z nich aplikována rovnice 2.10 a poté vybrána největší výsledná hodnota. Při testování jsem využil první ze zmíněných dvou variant, tedy rovnici 2.10, která využívá celou množinu referenčních souhrnů.

ROUGE-L

Metrika ROUGE-L se narozdíl od ROUGE-N dívá na větu více jako na celek. Důležitým procesem této metriky je vyhledání nejdelší společné podposloupnosti dvou vět, která je označována jako LCS (*Longest Common Subsequence*).

Uvažujme dvě věty A a B. LCS je pro tento problém definována jako nejdelší posloupnost slov, která se vyskytuje ve větě A a zároveň ve větě B [13]. Tato posloupnost nemusí být souvislá. Pokud věta A obsahuje slova $[w_1,w_2,w_3,w_4]$ a B slova $[w_1,w_2,w_5,w_4,w_7]$, tak LCS je tvořena slovy $[w_1,w_2,w_4]$. Vzorců a přístupů k výpočtu F-skóre, úplnosti a přesnosti je několik. V následujícím textu budou rozebrány pouze ty, které jsou využity při testování sumarizačních metod, které byly implementovány v této práci.

Mějme kandidátní souhrn K o počtu slov n a množinu ideálních souhrnů R. Výpočet úplnosti (vzorec 2.11) a přesnosti (vzorec 2.12) je založen na výpočtu LCS pro každou větu každého referenčního souhrnu z R a každou větu z K. Hodnota x_k je počet vět konkrétního souhrnu z množiny referenčních souhrnů R, a m je počet souhrnů vyskytujících se v této množině. Následně y označuje počet vět v souhrnu K. Pro každou větu ze souhrnu z R je vypočítána LCS s každou větou z K a určena největší hodnota. Tyto hodnoty jsou následně sečteny a poděleny buď součtem délek všech vět z množiny R pro výpočet úplnosti (*Recall*), nebo počtem slov z kandidátního souhrnu vynásobených počtem souhrnů v R pro výpočet přesnosti (*Precision*).

Například pro věty kandidátního souhrnu K_1 = [Zítra, bude, pršet], K_2 = [Nesmím, si, zapomenout, vzít, deštník] a věty z množiny referenčních souhrnů R_1 = [Zítra, může, být, škaredě], R_2 = [Vezmu, si, deštník], R_3 = [Zítra, bude, pršet], R_4 = [Nezapomenu, si, vzít, deštník]. Výpočet úplnosti podle 2.11 bude probíhat následovně. Hodnota LCS pro větu R_1 a K_1 je 1, jelikož nejdelší společná podposloupnost je slovo *Zítra*. Pro větu R_1 a K_2 je hodnota LCS rovna 0, žádná společná podposloupnost neexistuje. Za větu R_1 bude vybrána hodnota 1, jakožto největší LCS hodnota pro tuto větu a věty z K. Tento krok je proveden pro všechny ostatní věty z R. Pro představu například hodnota LCS věty R_4 a K_2 je 2, i když tato posloupnost není souvislá, skládá se ze slov *si* a *vzít*. Součet maximálních hodnot LCS pro věty z referenčních souhrnů je 7 (pro R_1 jedna, pro R_2 dva, R_3 tři a R_4 jedna). Hodnota úplnosti je poté 9/14.

Pro výpočet F-skóre se v tomto případě používá rovnice 2.13, která využívá jak úplnost, tak přesnost. Hodnota α určuje významnost přesnosti a úplnosti a může nabývat hodnot mezi 0 a 1. Pokud nabývá hodnoty větší než 0.5, tak větší významnost má přesnost a pokud menší než 0.5, tak významnější je úplnost. Jestliže hodnota α je rovna 0.5 jsou pro f-skóre přesnost a úplnost významné stejnou mírou.

$$Recall = \frac{\sum_{k=1}^{m}\sum_{i=1}^{x_k} max_{j=1}^{y} LCS(R_{ki}, K_j)}{\sum_{k=1}^{m}\sum_{i=1}^{x_k} |R_{ki}|} \tag{2.11}$$

$$Precision = \frac{\sum_{k=1}^{m}\sum_{i=1}^{x_k} max_{j=1}^{y} LCS(R_{ki}, K_j)}{n * m} \tag{2.12}$$

$$\text{F-score} = \frac{Recall * Precision}{\alpha * Recall + (1-\alpha) * Precision} \tag{2.13}$$

ROUGE-W

LCS, popsaný v přechozím textu, poskytuje hezké vlastnosti pro testování sumarizačních metod [12]. Jedním z jeho problémů je nerozlišování mezi podposloupností, která je souvislá, a která je naopak nesouvislá [12]. Mějme Například větu K_1 z kandidátního souhrnu a dvě věty R_1 a R_2 ze souhrnu referenčního, které vypadají následovně:

K_1 : [$\boldsymbol{w_1}$, $\boldsymbol{w_2}$, $\boldsymbol{w_3}$, $\boldsymbol{w_4}$, w_5 ,w_6]

R_1 : [$\boldsymbol{w_1}$, $\boldsymbol{w_2}$, $\boldsymbol{w_3}$, $\boldsymbol{w_4}$, w_8 ,w_9]

R_2 : [$\boldsymbol{w_1}$, $\boldsymbol{w_2}$, w_{10}, w_9, $\boldsymbol{w_3}$,$\boldsymbol{w_4}$]

Metrika ROUGE-L v tomto případě vyhodnotí, že R_1 a R_2 mají vůči K_1 stejnou hodnotu LCS. Věta R_1 má ale narozdíl od R_2 podposloupnost souvislou, tudíž by tato věta měla být více relevantní vůči K_1. Tento problém řeší metrika ROUGE-W, jenž využívá váhování nejdelší společné podposloubnosti (dále jen WLCS, neboli *Weighted Longest Common Subsequence*). [12]

WLCS zdokonaluje základní metodu LCS pomocí váhování jednotlivých podposloupností, které jsou ve větách nalezeny. Aplikace váhy se provádí váhovou funkcí, která může

mít několik různých podob. V implementované metrice ROUGE-W je využita váhová funkce $f(k) = k^w$, která pro daný vstup vrátí jeho w-tou mocninu [12]. Tato funkce je poté využita pro váhování podposloupností tak, že pro každou podposloupnost ve větě je určena její délka na kterou je aplikována funkce f a následně všechny tyto hodnoty sečteny. Pro předchozí příklad můžeme uvažovat váhu w rovnou 2, poté hodnota $WLCS(R_1, K_1, 2) = 4^2 = 16$ a $WLCS(R_1, K_1, 2) = 2^2 + 2^2 = 8$ [13]. Na tomto příkladu je vidět, že při využití váhování WLCS je možné od sebe rozeznat souvislé a nesouvislé podposloupnosti.

Jedním z důležitých kroků při použití metriky ROUGE-W je určení váhy pro váhovou funkci. Váha dokáže velikým způsobem ovlivnit jednotlivé výpočty, jak jsem se sám mohl přesvědčit při testování této metriky. V oficiální implementaci je doporučená hodnota pro váhu $w = 1, 2$ [13].

Vzorce pro výpočet úplnosti a přesnosti jsou oproti metrice ROUGE-L trochu pozměněny. Výpočet F-skóre je nadále stejný (rovnice 2.13). Obecné vzorce pro výpočet úplnosti a přesnosti vět X a Y reprezentují rovnice 2.14 a 2.15 [12].

$$Recall = f^{-1}\left(\frac{WLCS(X,Y)}{f(|X|)}\right) \tag{2.14}$$

$$Precision = f^{-1}\left(\frac{WLCS(X,Y)}{f(|Y|)}\right) \tag{2.15}$$

Tyto rovnice fungují pouze pro dvě věty a ne celý souhrn. Lze je ovšem upravit do tvaru 2.16 a 2.17 pokud využijeme váhové funkce $f(k) = k^w$ použité v implementaci. Výpočet úplnosti reprezentuje rovnice 2.16. Čitatel představuje výpočet součtu hodnot WLCS počítané pro každou větu z každého souhrnu obsaženého v množině referenčních souhrnů a pro každou větu ze souhrnu kandidátního. Hodnota w reprezentuje váhu, jenž je parametr pro váhovou funkci. Ve jmenovateli je počítán součet délek všech vět každého souhrnu z množiny ideálních souhrnů, kde každá věta je umocněna na váhu w a celkový součet délek vět každého souhrnu je také umocněn na w. Jelikož je v obecném vzorci celý tento výpočet předán jako parametr inverzní váhové funkci, celý tento zlomek je umocněn na $1/w$.

Jediným rozdílem mezi výpočtem úplnosti a přesnosti je jmenovatel. Pro výpočet přesnosti (rovnice 2.17) je ve jmenovateli počítán počet slov v kandidátním souhrnu (n) umocněný na váhu w a vynásobený celkovým počtem referenčních souhrnů.

$$Recall = \left(\frac{\sum_{k=1}^{m}\sum_{i=1}^{x_k} max_{j=1}^{y} WLCS(R_{ki}, K_j, w)}{\sum_{k=1}^{m}(\sum_{i=1}^{x_k}(|R_{ki}|)^w)^w}\right)^{1/w} \tag{2.16}$$

$$Precision = \left(\frac{\sum_{k=1}^{m}\sum_{i=1}^{x_k} max_{j=1}^{y} WLCS(R_{ki}, K_j, w)}{m * n^w}\right)^{1/w} \tag{2.17}$$

ROUGE-S

Metrika je založena na společném výskytu tzv. skip-bigramů v kandidátním souhrnu a referenčních souhrnech [13]. Skip-bigram je dvojice slov, které se vyskytují za sebou ve větě a na jejich vzdálenosti od sebe nezáleží [12]. Následující příklad je převzat z [12]. Mějme věty:

S_1 : police killed the gunman

S_2 : police kill the gunman

S_3 : the gunman kill police

Každá z těchto vět obsahuje šest skip-bigramů. Skip-bigramy první věty jsou *police killed*, *police the*, *police gunman*, *killed the*, *killed gunman* a *the gunman*. Věta S_2 obsahuje tři skip-bigramy, které se nachází i ve větě první a to *police gunman police the* a *the gunman*. Poslední věta má s větou S_1 společný pouze skip-bigram *the gunman*.

Tato metrika má stejně jako ROUGE-N několik různých variant. Tyto varianty jsou založeny na rozdílné maximální vzdálenosti mezi jednotlivými slovy, které tvoří skip-bigramy. Například ROUGE-S1 využívá skip-bigramy tvořené slovy, které jsou od sebe vzdáleny maximálně ob jedno slovo. Věta S_1 pro tento případ obsahuje pouze pět skip-bigramů (*police killed*, *police the*,*killed the*,*killed gunman*,*the gunman*). ROUGE-S2 využívá skip-bigramy tvořeny slovy, které jsou od se vzdáleny maximálně dvě slova, ROUGE-S5 maximálně pět slov atd. Speciálním případem je ROUGE-S*, kde slova, která vytváří skip-bigram, mohou být jakkoliv od sebe vzdálená [13]. Opačným případem je potom metrika ROUGE-S0, která je ekvivaletní s metrikou ROUGE-2, jelikož využívá pro tvorbu skip-bigramů pouze slova, která leží vedle sebe (jsou to obyčejné bigramy) [13].

Výpočet úplnosti a přesnosti je velice podnobný jako u metriky ROUGE-N. Místo počítání s n-gramy zde dochází k využití skip-bigramů, které se nachází jak v kandidátním souhrnu, tak v souhrnu z množiny referenčních souhrnů. Pro úplnost je poté v děliteli na místo počtu slov v množině referenčních souhrnů počet skip-bigramů obsažený v této množině a pro přesnost na místě počtu slov v kandidátním souhrnu počet jeho skip-bigramů.

Rozšířením metriky ROUGE-S, která je implementována v balíčku použitém při vyhodnocování sumarizačních metod, je metrika ROUGE-SU.

ROUGE-SU

Metrika ROUGE-SU řeší problém metriky ROUGE-S, který nastává v případě, kdy kandidátní věta neobsahuje žádná dvojice slov, které by se vyskytovaly v ideálních souhrnech [12]. Tato věta poté získá nulové ohodnocení, i když by v určitých případech měla mít skóre větší. Například věta S_4 *gunman the killed police* je obrácená věta k větě S_1. Obě dvě mají stejný význam [12]. Metrika ROUGE-S nenajde žádné společné skip-bigramy mezi S_4 a S_1. Řešením je tedy metrika ROUGE-SU, která využívá společně se skip-bigramy i unigramy, tedy jednotlivá slova [12].

Kapitola 3

Návrh a specifikace aplikace

Zadáním této práce je vytvoření webové aplikace provádějící sumarizaci textu pomocí zvolené metody. Na základě konzultace a dostupných nástrojů jsem zvolil jako jazyk pro vstupní text a výstupní souhrn angličtinu. Dalším důvodem byl fakt, že většina dostupného textu na internetu, ať už jde o knihy, zprávy nebo články, je napsána v anglickém jazyce a proto je příhodné zvolit tento jazyk.

Uživatel zadá vstupní text v anglickém jazyce, ze kterého chce vytvořit souhrn. Před spuštěním samotného procesu sumarizace je uživateli umožněno nastavení několika parametrů dané sumarizace, které určitým způsobem ovlivňují podobu výsledného souhrnu. Rozhodl jsem se k poskytnutí větší uživatelské svobody, aby výsledný souhrn byl co nejvíce přizpůsobem požadavkům uživatele.

Jedním z parametrů, které uživatel může měnit, je volba sumarizační metody. Mezi metody, které jsem se rozhodl využít patří metoda TextRank a Latentní sémantická analýza (obě popsané v kapitole 2.3). Jelikož jde o metody, které provádějí extraktivní sumarizaci, je tato aplikace určena k tvorbě extraktů. Dalším parametrem je volba použití rozpoznávání pojmenovaných entit (podkapitola 4.5). Uživatel tímto parametrem může určit, zda pojmenované entity, vyskytující se ve vstupním textu, budou ovlivňovat výběr vět do konečného souhrnu. Více o využití pojmenovaných entit při sumarizaci textu je popsáno v implementační části, viz kapitola 4.

Pro souhrn je podstaná co nejkratší délka a co nejvíce důležitý obsah ze vstupního textu. Někdy ovšem uživatel potřebuje co nejpřesnější počet vět v daném souhrnu. Kvůli tomu je uživateli umožněno volit si počet vět ve výsledném souhrnu. Pokud uživatel nepotřebuje konkrétní počet vět, ve svém konečném souhrnu, může nastavit tzv. práh. Hodnota prahu neurčuje kolik vět bude obsahovat výsledný souhrn, ale určuje na základě ohodnocení vět, které provedla sumarizační metoda, jaké věty se budou vyskytovat v konečném souhrnu. Více o prahu je popsáno v podkapitole 4.6.

Po volbě těchto parametrů je umožneno uživateli spustit proces sumarizace. Po dokončení tohoto procesu je uživateli zobrazen výsledný souhrn. Aby mohl uživatel využít přechozího nastavení parametrů a mohl změnit pouze vstupní text, rozhodl jsem pro využití technologie AJAX. V případě této aplikace jde o asynchronní zobrazení výsledného souhrnu, po dokončení procesu sumarizace, aniž by došlo k obnovení celé stránky. Tím je zajištěno, že vstupní text a nastavení parametrů není změněno a konečný souhrn je zobrazen. Technologie AJAX je poté popsána v podkapitole 4.7.

Kapitola 4

Implementace aplikace pro sumarizaci textu

Tato kapitola se zabývá popisem implementace webové aplikace, která slouží pro vytváření souhrnu z daného textu. Kapitola obsahuje popis programovacího jazyka Python, který byl použit k samotné implementaci. Dále představuje využití knihoven tohoto jazyka pro práci s textem a jejich přínos pro tvorbu souhrnů.

Jedním z dalších témat zastoupených v této kapitole je implementace a popis výběrů vět do výsledné sumarizace. S tím souvisí i nejdelší části této kapitoly, kterými jsou implementační detaily jednotlivých sumarizačních metod, které výsledná aplikace využívá k dosažení co nejlepších souhrnů. Nakonec je popsána implementace výsledné aplikace, jejíž návrh byl popsán v kapitole 3.

4.1 Programovací jazyk Python a jeho knihovny

Jak již bylo řečeno v úvodu této kapitoly 4, programovací jazyk Python byl zvolen jako implementační jazyk výsledné webové aplikace. To je důvodem k představení obecného popisu tohoto jazyka. Dále pak seznámení se s jeho výhodami a nevýhodami pro jeho využití k implementaci sumarizačních metod. A následné představení možností, které nám pro tuto práci samotný jazyk společně s jeho knihovnami nabízí.

Obecný vhled do jazyka Python

Jazyk Python byl vynalezen v devadesátých letech minulého století počítačovým programátorem Guidem van Rossumem[18]. Jméno tohoto jazyka bylo inspirované britským televizním seriálem Monty Pythonův létající cirkus (v originále Monty Python's Flying Circus).

Python je široce používaný vysokoúrovňový programovací jazyk [18]. Tento jazyk je interpretovaný. Nejříve přeložený do bajtkódu a poté interpretovaný na virtuálním stroji [18]. Python podporuje více paradigmat, jako je například procedurální nebo objektově-orientované programování. V tomto jazyce je všechno reprezentováno jako objekt [18]. To znamená, že skoro vše v Pythonu má své atributy a metody.

Python je dynamicky typovaný jazyk, z čehož vyplývá, že během kompilace se provádí pouze syntaktická analýza a typová kontrola probíhá až za běhu programu [18]. I když to tak na první pohled nevypadá, jazyk Python je silně typovaný, provádí kontrolu typů a zjišťuje jejich případné chybné použití. Nelze tedy sčítat jablka a hrušky dohromady, jako například v jazyce Javascript.

Po syntaktické stránce je jazyk Python velice jasný a jednoduchý [1]. Díky jeho čisté syntaxi je tento jazyk vhodný pro začátečníky. Jeho zvláštností je využívání bílých znaků, především tabulátorů, k odsazování. Narozdíl od jazyků typu C nejsou příkazy oddělovány či ukončovány středníkem, ale pouze znakem konce řádku. Z toho vyplývá, že se na jednom řádků nachází primárně pouze jeden příkaz. Toto lze obejít využitím už dříve zmiňovaného středníku, při jehož použití lze od sebe oddělovat příkazy na jednom řádku.

Python podporuje automatickou správu paměti pomocí tzv. garbage collectoru [18]. Tímto je zajištěno uvolňování paměti, o které se programátor nemusí starat. Mezi jeho další klady patří jeho přenosnost. Běží na mnoha variantách Linuxu, operačním systemu Windows i počítačích MAC [1].

Python má mnoho variant implementací, ke kterým patří CPython, Pypy, Jython nebo IronPython. Jeho aktuální verze je Python 3.x, který se od verze Python 2.x poměrně hodně odlišuje. V dnešní době se ovšem stále využívají obě dvě verze. [1]

Výhodou Pythonu je jednoduchá syntaxe, možnost využití reflexe, mnoha paradigmat, generického programování a užitečných standartních knihoven [18]. Je jednoduchý na naučení, poskytuje modulový systém a vestavěné typy [18].

Mezi nevýhody Pythonu patří nekompatibilita mezi verzemi 2 a 3 [18]. Není vhodný pro velice náročné výpočetní operace a úkoly, které výrazně využívají paměťový prostor [18].

Knihovny pro práci s textem

Python umožňuje použití širokého množství knihoven, mezi kterými se nachází knihovny pro práci s textem. Jednou z těchto knihoven, která byla použita pro tvorbu aplikace (kapitola 3), je knihovna NLTK.

NLTK

NLTK (Natural Language Toolkit) poskytuje soubor knihoven, které umožňují efektivní práci s textem v anglickém jazyce. Tyto knihovny implementují funkce a metody pro zpracování přirozeného jazyka [2]. Anglický jazyk není jediným jazykem, se kterým tyto knihovny dokáží pracovat. Je to ovšem jazyk, který je podporován všemi knihovnami, které NLTK poskytuje. NLTK je dostupný na platformách Windows, Mac OS X i Linux. Je to zcela bezplatný open-source projekt [2].

Knihovny z NLTK umožňují například provádět klasifikaci textu, tokenizaci, stematizaci, parsování, rozpoznávání pojmenovaných entit, či tagování [2]. NTLK tedy poskytuje velice dobré metody pro předzpracování textu, to je jedním z důvodů proč byl tento toolkit vybrán k použití při implementaci této aplikace (kapitola 3). NLTK také poskytuje rozhraní pro získání souborů textů, zvaných korpusy, které mohou být použity pro trénování modelů neuronových sítí nebo k vyzkoušení si práce s NLTK knihovnami [2].

Scikit-learn

Tato Python knihovna je vhodná pro efektivní dolování dat a datovou analýzu [3]. Je stejně jako NLTK open-source a může být komerčně využívána [3]. Scikit-learn je knihovna, která je primárně určena pro strojové učení [3]. Z tohoto důvodu nebyla nutnost jejího použití při implementování aplikace (3). Poskytuje možnost klasifikace, shlukování, regrese nebo využití předpřipravených modelů neuronových sítí [3].

NumPy

Tato knihovna je určena pro vědecké výpočty. Neslouží tedy primárně k práci s textem. Při zpracování textu je velice důležité převést text do jeho číselné reprezentace, se kterou dokáže počítač lépe pracovat. NumPy dokáže s touto číselnou reprezentací provádět efektivní výpočty, především s reprezentací vektorů a matic. Z tohoto důvodu je tato knihovna využita při implementaci sumarizačních metod, které aplikace využívá (viz kapitola 3). I NLTK využívá pro svoje výpočety právě tuto knihovnu NumPy [2].

4.2 Využití stematizace a odstranění klíčových slov

Před samotnou implementací sumarizačních metod je potřeba text či dokument, který tyto metody berou jako vstup, předzpracovat. Všechny důležité metody pro předzpracování textu se v implementaci nachází ve složce *utils* v souboru *document_preprocessing.py*.

Jedním z prvních procesů při předzpracování textu je rozdělení dokumentu na věty a následně tyto věty na tokeny. K tomuto byla v obou případech použita knihovna NLTK a její metody *sent_tokenize* a *word_tokenize*, které podporují anglický jazyk, jenž je pro tuto práci stěžejní. Samotná tokenizace byla popsána v kapitole 2.2.

Dalším důležitým krokem při předzpracování textu je stematizace, která je taktéž popsána v kapitole 2.2. Ke stematizaci je v implementaci využita knihovna NLTK, která implementuje *SnowballStemmer*. Algoritmus Snowball (někdy je označován jako Porter2), který je implementován jako *SnowballStemmer*, je optimalizovanější a provádí lepší stematizaci, než jeho předchůdce algoritmus Porter. *SnowballStemmer* je výrazně mírnější pro stematizaci, než například Lancaster, který je taktéž součastí knihovny NLTK. Snowball byl vybrán zejména kvůli nenásilnému určování základu slova, narozdíl od již zmiňovaného Lancaster algoritmu a efektivnějšímu způsobu provádění stematize, než algoritmus Porter.

Lematizace, popsaná v kapitole 2.2, je také součastí NLTK knihovny. NLTK poskytuje jedinou implementaci lematizace v podobě *WordNetLemmatizeru*. Tento lematizer je proto použit k provedení lematizace i v této aplikaci.

Posledním důležitým krokem pro předzpracování vstupního dokumentu je odstranění tzv. stop slov (kapitola 2.2). K tomuto je taktéž využita NLTK knihovna. NLTK poskytuje korpus, který obsahuje anglická stop slova určená pro jejich odstranění z dokumentu při analýze textu. Tyto slova jsou v korpusu zastoupena vždy s malým písmenem, tudíž před samotným odstraňováním jednotlivých stop slov je nutné pro každé slovo ze vstupního dokumentu provést převod všech velkých znaků na znak malý.

4.3 Implementace Latentní sémantické analýzy

Obecně byla metoda Latentní sématincké analýzy rozebrána v kapitole 2.3. Konečná verze této sumarizační metody je implementována ve složce *summarizers* v souboru s názvem *latent_semantic_analysis_final.py*. V tomto souboru se náchází třída LSAFinal, která reprezentuje tuto sumarizační metodu. Nejdůležitejší metodou třídy LSAFinal je metoda summary, která obstarává veškerou činnost tvorby souhrnu ze vstupního textu.

Nejdříve se na základě předchozího rozboru předzpracuje vstupní dokument. Jakým způsobem je toto provedeno je voleno na základě configuračních parametrů. Pokud jsou všechny parametry nastaveny na hodnotu *True*, provede se rozdělení dokumentu na věty. Následně věty na tokeny, u kterých se provede odstranění stop slov, lematizace a následně

stematizace. Tyto slova i věty, vzniklé zpracováním vstupního textu, jsou uloženy do třídních proměnných.

Dále je pro každé slovo vypočítána jeho frekvence výskytů v jednotlivých větách. Tato informace je taktéž uložena v třídní proměnné třídy LSAFinal. Důležitou metodou před aplikováním SVD algoritmu je metoda *build_matrix*. Tato metoda vytváří matici (s použitím knihovny NumPy vzniká speciální numpy pole, které tuto matici reprezentuje) a to tak, že jednotlivé řádky jsou tvořeny slovy a sloupce větami. Prvky matice představují počet výskytů daného slova v dané větě. Jelikož je informace o počtu výskytů jednotlivých slov uložena ve třídní proměnné datového typu slovník, lze ji pomocí názvu slova a pořadí věty zaindexovat.

Následuje singulární rozklad popsaný v kapitole 2.3. Knihovna NumPy obsahuje implementaci metody SVD. Pokud matice obsahuje stejně, nebo více řádků než sloupců, je použita implementace SVD takto (4.1):

$$u, sigma, vt = numpy.linalg.svd(matrix, full\_matrices = True) \tag{4.1}$$

Parametr *full_matrices* nastavený na hodnotu *True* znamená, že výsledná matice U a V^T budou čtvercové. Vektor Σ bude mít v tomto případě vždy dimenzi stejnou, jako počet řádků či sloupců čtvercové matice V^T. Tohoto lze využít při implementaci výpočtu ohodnocení jednotlivých vět ze vstupního dokumentu.

Výpočet tohoto ohodnocení je prováděn metodou *calculate_sentence_rank*. V tomto případě jsou jako parametry předány matice Σ a V^T. Ohodnocení je prováděni pro každou větu zvlášť a matice V^T je čtvercová, proto je po každé indexaci věty matice V^T výsledkem vektor (dále bude označován jako V_i), který má stejné rozměry jako vektor Σ. Ohodnocení vět je prováděno podle vzorce 2.3, jehož implementace byla provedena následovně:

$$sqrt(sum(vt[:, sent_{idx}] * *2 * sigma * *2) \tag{4.2}$$

, kde vt značí matici V^T, sigma představuje vektor Σ, $sent_idx$ reprezentuje index věty pro kterou je ohodnocení počítáno.

V případě, kdy má matice počet řádků menší, než sloupců je tato matice transponována pomocí metody knihovny NumPy a její singulární rozklad je proveden s jinými parametry.

$$u, sigma, vt = numpy.linalg.svd(matrix, full_matrices = False).$$

Parametr *full_matrices* nastavený na hodnotu *False* znamená, že výsledná matice U bude mít rozměry $m \times k$ a matice V^T rozměry $k \times n$, kde k je menší z hodnot m a n. Jelikož byla matice před provedením SVD transponována, důležité informace pro ohodnocení vět se nevyskytují v matici V^T, ale matici U. Ta je společně s vektorem Σ, jehož dimenze má hodnotu k, předána jako pamater funkci *calculate_sentence_rank*. Dalším parametrem je parametr *transposed* nastavený na hodnotu *True*, což značí, že matice byla před výpočtem singulárního rozkladu transponována a musí se použít jíný postup výpočtu ohodnocení vět. Tento postup je zobrazen v následující rovnici 4.3.

$$sqrt(sum(vt[sent_{idx}, :] * *2 * sigma * *2). \tag{4.3}$$

Jediná změna oproti rovnici 4.2 nastává v indexování matice U, která je reprezentována proměnnou vt. Důvodem toho je zastoupení vět v řádcích narozdíl od rovnice 4.2, kde jsou věty zastoupny ve sloupcích.

4.4 Implementace algoritmu TextRank

Algoritmus TextRank je implementován v souboru *summarizers* ve složce, která nese název *text_rank_single_document_final.py*, podle popisu této metody v kapitole 2.3. Základ této implementace tvoří třída TextRankSingleDocumentFinal, která obsahuje všechny důležité metody. Hlavní metodou je metoda summary.

V této metodě je vstupní dokument předán funkcím pro předzpracování textu v podobě tokenizace, stematizace, lematizace a odstranění stop slov na základě konfiguračního souboru *config.json*, stejně jako u Latentní sémantické analýzy viz 4.3.

Změnou oproti LSA (4.3) je reprezentace každé věty v podobě pole. Toto pole obsahuje jako svoje prvky slova dané věty. Tyto věty jsou parametrem pro metodu *build_similarity_matrix*, která, jak už název napovídá, provádí vytvoření matice podobnosti. Metoda *build_similarity_matrix* je metodou třídy SimilarityMetricsSingleDocument, která je instanciována při inicializaci třídy TextRankSingleDocumentFinal.

Základem metody *build_similarity_matrix* je vytvoření dvourozměrného pole představující matici podobnosti. Řádky i sloupce této matice představují věty z dokumentu a prvky matice reprezentují podobnost vět mezi sebou. Prvkům ležící na diagonále matice podobnosti je přiřazena hodnota jedna (hodnota 1 reprezentuje stoprocentní podobnost, naopak hodnota 0 podobnost nulovou). Jelikož každá věta z dokumentu odpovídá jednomu řádku i sloupci, tak prvek na diagonále je důsledkem indexace matice podobnosti právě identickými větami, a proto je tomuto prvku přidělena hodnota 1. Ostatní prvky matice podobnosti jsou vypočítány na základě podobností dvou indexujících vět pomocí metody *sentence_similarity*.

Parametry této metody jsou dvě věty. Pro každou z nich metoda provádí vytvoření vektoru, jehož dimenze je rovna celkovému počtu slov obsažených v obou větách (počítáno bez duplicit). Prvek tohoto vektoru představuje jedno slovo z daných dvou vět. Každému prvku daného vektoru je poté vypočítán počet jeho, tedy konkrétního slova, výskytů v dané větě.

Takto vytvořené vektory jsou následně předány jako parametr metodě *cosine_distance* z knihovny NLTK. Tato metoda provádí výpočet kosinovy podobnosti (vzorec 2.6) s tím rozdílem, že pro identické vektory vrací hodnotu 0 a pro zcela rozdílné hodnotu 1. V této implementaci jsou hodnoty podobnosti uvažovány opačné, a proto je návratová hodnota z metody *cosine_distance* odečtena od hodnoty 1.

Tímto způsobem je vytvořena matice podobnosti. Poslední úpravou prvků této matice, před použitím algoritmu PageRank, je jejich normalizace. Výraz 4.4, který je součástí rovnice pro výpočet PageRanku 2.7, představuje normalizaci jednoho prvku matice podobnosti. Hodnota w_{ij} reprezentuje právě jeden prvek matice, kde i je index řádku a j index sloupce.

$$\frac{w_{ij}}{\sum_{V_k \in Out(V_j)} w_{kj}} \tag{4.4}$$

Výraz $\sum_{V_k \in Out(V_j)} w_{kj}$ přestavuje součet všech prvků matice podobnosti ve sloupci j. Jelikož je matice podobnosti symetrická, bylo v implementaci normalizace použito místo součtu všech prvků matice ve sloupci j součtu všech prvků v řádku i. Normalizace je tímto způsoben provedena pro celou matici. Touto normalizací prvků matice podobnosti je matice připravena k použití algoritmu PageRank.

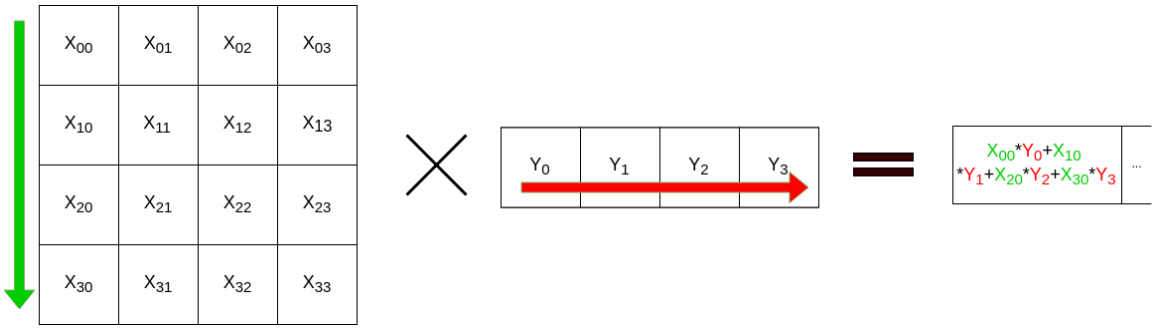
PageRank je popsán v kapitole 2.3. Implementace tohoto algoritmu je provedena maticově. Nejdříve je pomocí NumPy knihovny vytvořen vektor, jehož dimenze je rovna počtu řádků matice podobnosti. Složky tohoto vektoru představují jednotlivá PageRank skóre

(dále pak PR). Pro dané věty je hodnota PR rovna jedné. Jelikož je potřeba inicializovat tento vektor tak, aby součet hodnot všech jeho prvků byl roven jedné, musí být tyto hodnoty poděleny velikostí dimenze daného vektoru.

Po incializaci následuje cyklus, jehož nejdůležitější část je tvořena výpočtem PR skóre. Ve vzorci 2.7 je počítáno toto skóre pouze pro jednu větu. V této implementaci je použito maticové násobení, které umožnuje výpočet PR pro všechny věty najednou. Tato implementace je zobrazena v následujícím kódu 4.5.

$$new_P = np.ones(len(A)) * (1 - self.d)/len(A) + self.d * A.T.dot(P) \tag{4.5}$$

Příkaz $np.ones(len(A))$ vytvoří stejný vektor jako v inicializační části. $A.T$ reprezentuje transponovanou matici podobností a P vektor obsahující PR skóre všech vět. Poslední výraz rovnice 4.5, tedy $A.T.dot(P)$, provádí skalární součin transponované matice A a vektoru P. Jelikož NumPy při výpočtu tohoto skalárního součinu bere vektor P jako řádkový, je výpočet tohoto součinu prováděn podle obrázku 4.1. Z tohoto důvodu musí být matice podobnosti A transponována, jelikož normalizace této matice proběhla po řádcích, a ne po sloupcích.



Obrázek 4.1: Výpočer skalárního součinu matice a vektoru v prostředí NumPy

Kontrola konce algoritmu je provedena výpočtem rozdílu vektoru P a P z předchozí iterace, po kterém vznikne vektor jejich rozdílu. Pokud tento vektor obsahuje hodnotu menší nebo rovnu hodnotě ε, je algoritmus ukončen.

4.5 Využití rozpoznávání pojmenovaných entit

Aby bylo možné popsat využití pojmenovaných entit při tvorbě sumarizace, je důležité pochopit, co slovní spojení *pojmenované entity* označuje. Podle [11] je pojmenovaná entita slovo či slovní spojení, jež jednoznačně identifikuje osobu, místo, časový úsek, firmu atd.

Pojem pojmenovaná entita se může často plést s pojmem vlastní jméno. Oba výrazy však nejsou úplně stejné a jsou mezi nimi určité rozdíly. Například vlastní jména se píší s velkým písmenem na začátku slova, zatímco pojmenované entity mohou začínat i písmeny malými. [11]

Dalším rozdílem může být rozsah, kdy vlastní jména jsou většinou tvořeny dvěma slovy, kdežto pojmenované entity mohou být tvořeny i pěti slovy najednou. Pojmenovaná entita může být i zároveň vlastní jméno, a naopak. Příklad, který toto ilustruje je převzat z [11]. Mějme větu: *Václav Klaus byl zvolen prezidentem České republiky v roce 2003. Václav Klaus*

je vlastní jméno, ale zároveň i pojmenovaná entita (identifikace osoby). Druhou pojmenovanou entitou, která se v tomto příkladu nachází je *prezidentem České republiky v roce 2003*. Uvnitř druhé pojmenované entity se nachazí další vnořené pojmenované entity a to *Česká republika* (identifikuje místo) a *rok 2003* (identifikuje časový úsek).

K rozpoznání pojmenovaných entit v textu lze využit několik přístupů. Jedním z nich může být vyhledání konkrétních slov podle slovníku [11]. Tento přístup může mít řadu problémů, jako je například slovník, který neobsahuje všechny pojmenované entity, nebo velké písmeno u slova na začátku věty (*Kohout* jako jméno a *kouhout* jako zvíře) [11]. Dalšími přístupy mohou být statistické modely nebo neuronové sítě. Popis těchto metod a přístupů není součástí této práce.

V další části textu je popsáno, jakým způsobem jsou využity pojmenované entity při provádění sumarizace textu a jak bylo jejich využití implementováno v této aplikaci.

Aplikace (3) umožňuje uživateli vybrat, zda budou, nebo nebudou využity pojmenované entity k ovlivnění ohodnocení jednotlivých vět. Pokud se uživatel rozhodne tyto pojmenované entity využít, je volána metoda *apply_ner_weights*.

Důležitou součástí této metody je počet pojmenovaných entit v konkrétních větách. Pro každou větu je při předzpracování textu, které bylo pospáno v kapitole 4.2, spočítán počet pojmenovaných entit, které se v ní nachází. K zjišťování a rozpoznávání těchto entit jsou v této implementaci využity tři metody z knihovny NLTK. Těmito metodami jsou *pos_tag*, *ne_chunk* a *tree2conlltags*.

Metoda *pos_tag* slouží k určování slovních druhů. POS je akronymem anglického slovního spojení part of speech, což znamená slovní druh. Tato metoda určuje pro každé slovo jeho slovní druh a výsledkem je pak dvojice *(slovo, slovní druh)*. Metoda je aplikována na každé slovo v každé větě. V implementaci aplikace je tato metoda využívána vždy na jednotlivé věty, kdy vstupem do ní je seznam slov a výstupem seznam dvojic.

Výstup metody *pos_tag* vstupuje do metody *ne_chunk*. Tato metoda zpřístupňuje natrénovaný model pro rozpoznání jednotlivých pojmenovaných entit na základě slovního druhu slova a slovních druhů slov vyskytujících se ve větě okolo něho. Výstupem metody *ne_chunk* je strom reprezentující větu. Ke slovům, které byly rozpoznány jako pojmenované entity je přidáno její označení. Ostatním slovům není přídáno žádné označení.

Nakonec je použita metoda *tree2conlltags*, která vytvořený strom, pomocí předchozí metody, konvertuje na seznam trojic. Každá trojice obsahuje slovo, slovní druh a označení pojmenované entity. Toto označení je při konverzi stromu pozměněno a to tak, že pokud je tvořena pouze jedním slovem, je před toto slovo přidáno *B-* (z anglického slova *beginning*). Je-li pojmenovaná entita tvořena více slovy, prvnímu z nich je přidána předpona *B-* a ostatním předpona *I-* (z anglického slova *inside*). Pokud slovo nebylo rozpoznáno jako pojmenovaná entita, neobsahuje tudíž žádné označení a místo toho je použito speciální značení písmenem O (vychází z anglického slova *outside*).

Po získání tohoto seznamu trojic je zjištěn počet vyskytujících se pojmenovaných entit v konkrétní větě. To je provedeno spočítáním všech entit kromě těch, které jsou označeny písmenem O. Toto je provedeno pro všechny věty vstupního textu. Takto připravený počet pojmenovaných entit ve větách je použit k upravení ohodnocení vět.

Ohodnocení vět je upraveno podle vzorce 4.6. Proměnná *new_weight* reprezentuje upravené ohodnocení věty, *sentence_weight* představuje ohodnocení věty před jeho změnou, *ner_weight* je počet pojmenovaných entit v dané větě, pro kterou je počítána změna jejího ohodnocení, a hodnota $self.num\_all\_named\_entities$ představuje počet všech pojmenovaných entit, které se vyskytly v celém vstupním dokumentu.

$$new\_weight = sentence\_weight * (ner\_weight / self.num\_all\_named\_entities) \quad (4.6)$$

4.6 Výběr vět do výsledné sumarizace

U metody Latentní sémantické analýzy a PageRanku je výběr vět pro výsledný souhrn prováděn stejným způsobem. Aplikace umožňuje, aby uživatel zvolil jednu ze dvou možností pro výběr vět. Mezi těmito možnostmi je zvolení prahu, anglicky *threshold* (metoda *apply_threshold*), nebo přímého vybrání počtu vět (metoda *get_top_n_sentence_idx*), který by se měl podle uživatele vyskytnout v konečném souhrnu.

Do obou metod jsou věty předány jako parametr s datovým typem seznam. Tento seznam obsahuje všechny věty ze vstupního dokumentu, kde každá z nich je reprezentována dvojicí hodnot. První hodnota odpovídá indexu věty ve vstupním dokumentu a druhá představuje ohodnocení, které každá věta získala aplikováním jedné ze sumarizačních metod (Latentní sémantické analýzy, nebo PageRanku).

Po výběru vět na základě prahu, nebo hodnoty počtu vět je získán nový seznam, který v sobě obsahuje věty do výsledného souhrnu, stále reprezentovány dvojicí hodnot. Prvky tohoto seznamu jsou seřazeny podle první hodnoty, tedy indexu věty ve vstupním dokumentu, v sestupném pořadí. Díky těmto seřazeným indexům jsou vybrány věty ze vstupního textu v pořadí takovém, ve kterém se v něm nacházejí a tyto věty následně tvoří výsledný souhrn.

Threshold

Hodnota prahu, kterou uživatel může zadat, se pohybuje v intervalu $\langle 0, 1 \rangle$. Na základě této hodnoty je počítána skutečná hodnota prahu, podle které probíhá výběr vět.

Metoda LSA i PageRank provádí ohodnocení vět vstupního dokumentu. Skutečný práh je počítán pomocí nejvyššího (*maximal rank*) a nejnižšího ohodnocení (*minimal rank*). Tento výpočet představuje rovnice 4.7. Hodnota *threshold_ratio* reprezentuje práh, který volí uživatel.

$$threshold = max\_rank - (max\_rank - min\_rank) * threshold\_ratio \quad (4.7)$$

Po výpočtu prahu je pak pro každou ohodnocenou větu provedeno porovnání, zda její ohodnocení je větší nebo rovno tomuto prahu. Pokud ano, je daná věta zařazena do výsledné sumarizace. V opačném případě je věta zanedbána.

Počet vět

Druhou možností pro výběr vět je určení celkového počtu vět (dále pak n), který by se měl objevit v souhrnu. Aplikace dovoluje zadat hodnotu počtu vět pouze větší nebo rovnu nule, nemůže tak dojít k zvolení záporných hodnot, které by v tomto případě nedávaly žádný smysl.

Ohodnocené věty jsou seřazeny podle ohodnocení od nejvyššího po nejnižší a následně je vybráno top n vět s nejvyšší hodnotou. Pokud je hodnota n větší, než počet vět ve vstupním dokumentu, výsledný souhrn je totožný se vstupním textem.

4.7 Implementace webové aplikace

V této poslední části textu budou popsány implementační detaily dané webové aplikace, jejíž návrh byl představen v kapitole 3.

Implementačním jazykem této webové aplikace je jazyk Python, pro který existuje mnoho webových frameworků. Mezi ně patří například Django, CherryPy, PyramidTornado či Flask. Jelikož je daná aplikace navržena jednoduše, nepotřebuje rozsáhlý framework, proto byl vybrán Flask, jako implementační webový framework pro tuto webovou aplikaci.

Pro samotnou aplikaci jsou důležité složky *static*, *templates*, *views* a také soubory *app.py* a *___main___.py*, které jsou uloženy mimo tyto složky. Složka *static* slouží k uchování souborů určených ke stylizaci html dokumentů. Obsahuje tedy css soubory pro jednotlivé stránky webové aplikace.

Složka *templates* obsahuje html soubory, jež definují obsah jednotlivých stránek. Poslední důležitou složkou je složka *views*, která uchovává soubory, které umožňují zobrazení stránek a poskytují různé funkce, které budou popsány dále v textu. K tomuto využívají již zmíněný framework Flask. Soubor *___main___.py* slouží ke spuštění celé aplikace.

Flask umožňuje vytvářet tzv. *blueprinty*, které slouží jako šablony pro generování jednotlivých sekcí aplikace. Tyto blueprinty jsou vytvořeny pro každou stránku pomocí konstruktoru obsaženým v jednotlivých souborech složky *views*. Tomuto konsktruktoru je předán i parametr pro nalezení složky, ve které se vyskytuje popis dané stránky (tedy složka *templates*), pro kterou je *blueprint* vytvořen. Tyto *blueprinty* jsou poté zaregistrovány v souboru *app.py* pomocí metody frameworku Flask. Bez jejich registrace by aplikace o daných stránkách, reprezentovaných *blueprinty*, nevěděla.

Pro každou stránku je ve složkách *templates* a *views* vytvořen jeden soubor. Tyto soubory by měly mít stejný název, jelikož jsou mezi sebou provázány. Důležitým souborem pro aplikaci je soubor s názvem *homepage.py* a *homepage.html*.

Flask umožňuje pomocí dekorátoru zaregistrovat tzv. pohled (anglicky *view*). Tento pohled představuje funkci, která vrací obsah pro danou cestu v URL. Spojení cesty a pohledu se říká routa (anglicky *route*). Příkladem pro to může být tato část kódu.

```
@app.route('/')
def index():
    return 'Index Page'
```

Proměnná *app* reprezentuje *blueprint*, pro který je zaregistrovaný pohled, kde na cestě *'/'* bude přístupný obsah funkce *index()*. Tímto způsobem je v souboru *homepage.py* umožněno, aby na cestě *'/homepage'* byl dostupný obsah funkce *show*. Tato funkce je velmi důležitá a to z důvodu, že vrací vyrenderovanou stránku popsanou v souboru *homepage.html*. Jinými slovy umožňuje vyobrazit danou stránku. Vyrenderování je provedeno pomocí funkce *render_template*, jež poskytuje Flask. Jako jediný parametr je předán název html dokumentu, jehož obsah se má zobrazit.

Soubor *homepage.py* obsahuje načtení a zkonstruování objektů všech tříd, které provádějí sumarizace (viz 4.3 a 4.4). Tyto objekty jsou využity metodou *summarize*, jejíž volání je spjato s kódem v html souboru *homepage.html*.

V souboru *homepage.html* je tedy uložen popis stránky pomocí html elementů. Stránka obsahuje dvě textová pole pro zobrazení vstupního textu a výsledného souhrnu. Dále pak selektor pro výběr sumarizační metody, posuvník k určení hodnoty prahu, vstupní pole pro určení počtu vět do výsledného souhrnu a zaškrtávací políčko k určení, zda budou využity

pojmenované entity. Posledním důležitým elementem stránky je tlačítko *Summarize*, jež slouží k provedení sumarizace vstupního textu.

Aby tento proces fungoval asynchronně a nebyla obnovována celá stránka, ale pouze obsah textová oblast určená pro výsledný souhrn, implementace využívá technologie AJAX (asynchronní Javascript a XML). AJAX umožňuje dynamicky měnit obsah stránky bez nutnosti jejího znovunačtení za pomoci asynchronního zpracování webových stránek pomocí knihovny napsané v JavaScriptu. Tato knihovna se jmenuje jQuery. JQuery je tedy knihovna jazyka JavaScript, jež umožňuje vytvářet stránky více dynamicky pomocí výměny dat mezi serverem a klientem ve formátu JSON. JSON formát nahradil formát XML, který se vyskytuje v názvu technologie AJAX.

Pomocí knihovny jQuery je pak napsána funkce, která je vykonána při stisku tlačítka *Summarize*, která již byla v tomto textu zmíněna. Tato funkce zasílá požadavek typu GET na URL, jehož cesta je spojena s funkcí *summarize*, implementovanou v souboru *homepage.py*. Požadavek obsahuje hodnoty jednotlivých možností, které uživatel navolil před kliknutím na tlačítko *Summarize*, např. výběr sumarizační metody, počet vět atd. Po kliknutí na tlačítko *Summarize* je zavolána funkce, která pomocí požadavku GET předá jednotlivé hodnoty funkci *summarize*.

Tato funkce na základě obdržených hodnot provede tvorbu souhrnu vstupního textu podle zvolené sumarizační metody a výsledný souhrn vrátí ve formátu JSON funkci, která byla vytvořena pomocí jQuery. Tímto je zajištěn asynchronní přístup. Data přijatá od funkce *summarize* jsou následně předána textové oblasti, jejíž obsah se aktualizuje a zobrazí uživateli na klientské straně. K jiným změnám při tomto procesu nedochází.

Aby bylo možné využívat jednotlivé implementované sumarizační metody přímo ve webové aplikaci, bylo nutné vytvořit speciální třídu, která tento problém řeší. Tato třída se nazývá *processor* a obsahuje dvě důležité metody. Metoda *load_summarizers*, jak už název napovídá, načte a vytvoří instance všech sumarizačních tříd, které se vyskytují v konfiguračním souboru, do třídní proměnné třídy *processor*. Tato metoda je volána v souboru *homepage.py* před prvním renderováním stránky.

Druhá metoda s názvem *run* je volána z funkce *summarize*. Metodě *run* jsou jako parametry předávány: vstupní text, název sumarizační metody, hodnota prahu, počet vět a využití pojmenovaných entit. Všechny tyto parametry jsou důležité pro tvorbu souhrnu a zároveň jsou očekávanými parametry jednotlivých implementovaných sumarizačních metod. Na základě instancí, které byly do třídní promenné třídy *processor* přiřazeny v metodě *load_summarizers*, je porovnáno jejich jméno s názvem sumarizační metody, která má provést souhrn z daného vstupního textu. Pokud se názvy shodují, je nad danou instancí sumarizační metody volána metoda *summary*. Jelikož všechny třídy sumarizačních metod využívají stejné rozhraní, tak metoda *summary* je implementována pro každou z nich.

Kapitola 5

Testování metod pro aplikaci

Testování sumarizačních metod jsem provedl ve dvou fázích. V první fázi byly jednotlivé souhrny, vytvořené sumarizačními metodami, podrobeny mému vlastnímu subjektivnímu posouzení kvality těchto souhrnů. Tato fáze spočívala v přečtení všech vstupních textů a obodování jednotlivých souhrnů. Bodoval jsem známkami od jedné do pěti. Známka jedna představuje velice špatnou kvalitu souhrnu na základě obsahu a návaznosti vět. Známka pět naopak reprezentuje kvalitu dobrou, až překvapující.

V druhé fázi byl k vyhodnocení souhrnů, vytvořených pomocí sumarizačních metod, použit nástroj ROUGE (2.4). Jelikož originální implementace nátroje ROUGE není napsána v jazyku Python, ale v jazyku Perl, využil jsem dostupného obalu napsaného v Pythonu, který zprostředkovává rozhraní pro využití metrik ROUGE. Samotný nástroj ROUGE obsahuje více metrik, jak bylo popsáno v 2.4, a proto jsem se rozhodl využít ty metriky, které podle [12] dosahují kvalitních výsledků pro jednodokumentovou sumarizaci, která je náplní této práce. Je to metrika ROUGE-2, ROUGE-L, ROUGE-SU4 a ROUGE-W s váhou $w = 1,2$. Po tomto procesu jsem nakonec srovnal mé bodové hodnocení jednotlivých sumarizačních metod s hodnocením, které vytvořil nástroj ROUGE.

Jako vstupní texty pro jednotlivé sumarizační metody byly zvoleny internetové zprávy z kanálu CNN. Těchto zpráv, napsaných v anglickém jazyce, bylo využito celkem pět. Pro každou tuto zprávu byly vytvořeny vždy tři referenční souhrny, každý z nich jinou osobou. Referenční souhrny jsem nevytvářel já, abych zabránil případnému ovlivňení porovnání výsledků metrik ROUGE a mým hodnocením kvalit kandidátních souhrnů. Tyto referenční souhrny jsou vytvořeny z vět kandidátního souhrnu, tudíž jde o extrakty vytvořené člověkem. Důvodem pro zvolení extraktů a nepoužití abstraktů je ten, že testované metody vytvářejí extrakty a proto mi přišlo nevhodné porovnávat extrakty s abstrakty.

Testovány byly pouze sumarizační metody, které jsou implementované v samotné aplikaci. Tedy metoda TextRank a Latentní sémantická analýza, obě dvě popsané v podkapitole 2.3. Tyto dvě metody jsem testoval ve fázích, které byly popsány výše. Navíc k tomu byla každá z těchto metod testována ve více variantách. První varianta byla samotná metoda bez použití rozpoznávání pojmenovaných entit (dále pak NER) s výběrem vět na základě prahu. Druhá varianta spočívala v metodě bez použití NER a s výběrem vět podle počtu vět, který má být ve výsledném souhrnu. Další dvě varianty jsou totožné s prvními dvěma variantami s tím rozdílem, že u obou dvou je využit NER. Pro práh byly vybrány hodnoty $0,2$, $0,5$ a $0,8$, aby mohly být testovány krátké, střední i dlouhé souhrny. Ze stejného důvodu, jako u výběru hodnot pro práh, byly vybrány hodnoty pro počet vět, tedy $0,2$, $0,5$ a $0,7$. Nejde ovšem přímo o počet vět, ale o procentuální vyjádření počtu vět z celého vstupního dokumentu.

5.1 Výsledky testování a vyhodnocení sumarizačních metod

Jak bylo řečeno v kapitole 5, k testování bylo použito 5 článků. Každý tento článek je o jiném rozsahu a jiném počtu pojmenovaných entit, jak je vidět v tabulce 5.1. Důvodem pro to je zkoumání, zda počet vět a pojmenovaných entit výrazným způsobem ovlivní výsledný souhrn, nebo nikoliv.

Samotný program ROUGE dovoluje využití stematizace a odstranění stop slov při evaluaci jednotlivých souhrnů. Odstranění stop slov je samozřejmostí, protože ty nevytvářejí důležitý obsah textů. Problém nastává u lematizace, zda tento proces provádět, či nikoliv. Abych zjistil, zda jednotlivé metriky ROUGE dosahují lepších výsledků při využití procesu lematizace provedl jsem testování s jeho využitím i bez něho. Výsledky vypovídaly zcela jasně ve prospěch využití lematizace. Odchylky nebyly sice markatního rozměru, ale pro každou metriku ROUGE byla její hodnota po provedení hodnocení souhrnu s využitím lematizace lepší, o trochu vyšší, než bez využití lematizace. Z tohoto důvodu jsem pro výsledné hodnocení metod využil hodnoty metrik ROUGE, u kterých byl použit proces lematizace.

Při testování variant s využitím prahu ovlivňovalo výsledek skóre jednotlivých vět. Pokud jedna věta měla například o řád vyšší skóre než ostatní, ve výsledném souhrnu se mohla, i při zvolení prahu $0,8$, vyskytnou pouze ona jediná. Tím pádem byla za kandidátní souhrn považována pouze jedna jediná věta. Což vedlo k velice nízkému ohodnocení metrikami ROUGE.

Pro každou testovanou metodu jsem provedl 60 pokusů. Abych zjistil nejlepší kombinaci parametrů pro jednolivé metody, u každého pokusu jsem zaznamenal hodnoty F-skóre jednotlivých metrik nástroje ROUGE a následně i svoje vlastní hodnocení. Po zaznamenání těchto hodnot jsem pro každý článek označil tři největší hodnoty u každého typu hodnocení, tzn. jak pro ROUGE-N, ROUGE-W, ROUGE-L, ROUGE-SU4, tak pro moje vlastní hodnocení. A tato označení jsem rozlišil na první, druhé a třetí místo. Pro každou variantu metody jsem spočítal počet všech takto označených hodnot. Pokud byla hodnota označena prvním místem, počítala se tato hodnota 3krát, pokud druhým místem, tak 2krát a pokud místem třetím, tak 1krát. Porovnání výsledného počtu určilo nejlepší variantu pro každou metodu. Jednotlivé výsledky testování metod budou následně představeny.

Tabulka 5.1: Rozbor testovacích článků

	Počet vět	**Počet pojmenovných entit**
1. článek	51	116
2. článek	54	135
3. článek	26	43
4. článek	21	19
5. článek	23	40

Testování metody TextRank

Na začátku testování jsem pro každý článek vytvořil souhrn pomocí této metody. Jak bylo řečeno v úvodu kapitoly 5, k testování bylo využito celkem 12 variant dané metody a proto je pro každý článek vytvořeno 12 souhrnů. Tyto souhrny jsem poté ohodnotil, na základě mého vlastního posouzení kvality, body od jedné do pěti (hodnota jedna je nejhorší a pět

nejlepší). Jednotlivé body, které jsem přidělil daným souhrnům, jsou zobrazeny v tabulce 5.2.

Tabulka 5.2 ukazuje, že na základě mého posouzení byly nejkvalitnější souhrny provedeny s použitím ropozpoznávání pojmenovaných entit, a to jak s využitím prahu o hodnotě $0,8$, tak s využitím počtu vět, kde v souhrnu byla polovina vět původního textu. Další kvalitní souhrny poskytovala varianta s využitím pojmenovaných entit, kde souhrn obsahoval pouze 20 % vět originálního textu a také varianta bez použití pojmenovaných entit s prahem $0,8$. Tyto souhrny byly do jisté míry tvořeny podstatnými informacemi a věty na sebe z větší části navazovaly. Poměrně podobných výsledků bylo dosaženo pomocí metrik ROUGE.

Tabulka 5.2: Subjektivní posouzení kvality souhrnů vytvořených metodou TextRank

	Varianta	**1. článek**	**2. článek**	**3. článek**	**4. článek**	**5. článek**
A	NER práh 0,2	3	4	1	1	1
B	NER práh 0,5	3	4	1	1	2
C	NER práh 0,8	4	3	1	3	2
D	NER věty 0,2	3	3	1	2	3
E	NER věty 0,5	2	3	2	3	3
F	NER věty 0,7	2	2	2	2	2
G	práh 0,2	3	1	1	1	1
H	práh 0,5	2	2	1	2	2
I	práh 0,8	2	2	3	3	2
J	věty 0,2	2	2	1	1	2
K	věty 0,5	2	2	2	2	2
L	věty 0,7	2	2	2	2	2

Jelikož celková tabulka hodnocení metody TextRank pomocí metrik ROUGE je obsáhlá, je pro názornou ukázku zobrazeno v tabulce 5.3 pouze hodnocení prvního souhrnu. Celkové hodnocení všech souhrnů je poté součástí přílohy (A).

V tabulce 5.3 je vidět, že pro první článek dosahuje nejlepšího hodnocení varianta metody TextRank s použitím NER, kde souhrn obsahuje pouze 20 % vět z celkového počtu vět vstupního textu. Tohoto hodnocení dosahuje u všech metrik. V případě prvního článku dopadly nejhůře varianty s využitím prahu $0,2$ a $0,5$ s využitím rozpoznávání pojmenovaných entit. Jejich hodnocení pomocí každé metriky je naprosto stejné, viz tabulka 5.3.

V celkovém testu této metody pomocí metrik ROUGE dopadla nejlépe varianta s rozpoznáváním pojmenovaných entit bez použití prahu, kde souhrn obsahoval polovinu vět ze vstupního dokumentu (varianta E). Ve většině případů byla tato varianta ohodnocena nejlépe. Druhou nejlépe hodnocenou variantou je využití prahu $0,8$ bez použití NER (varianta I). Na třetím místě dopadla varianta F, s využitím NER a 70 % počtem vět. Varianty E a I byly nejlépe hodnoceny už v první fázi, tedy mnou.

Po zhodnocení variant metody TextRank pomocí ROUGE a zároveň pomocí mého subjektivního hodnocení (viz úvod podkapitoly 5.1) jsem dospěl k následujícím výsledkům. Varianta, která provádí nejkvalitnější souhrny je varianta E. Dále pak varianta I zůstává na svém druhém místě. Varianta F neměla dost dobré subjektivní hodnocení, proto na třetím místě je varianta C, s využitím rozpoznávání pojmenovaných entit a prahem o hodnotě $0,8$.

Z porovnání výsledků metrik ROUGE a mého subjektivní hodnocení lze usoudit, že využití rozpoznávání entit u metody TextRank vede k lepším výsledkům. Tyto lepší výsledky jsou ovšem dosaženy za předpokladu, že počet vět ve výsledném souhrnu se pohybuje okolo poloviny počtu vět vstupního textu, nebo pokud jsou jednotlivá skóre vět rozložená tak, že práh o hodnotě $0,8$ dokáže vybrat pouze věty s nejdůležitějším obsahem.

Tabulka 5.3: Hodnocení souhrnu prvního článku, vytvořeného metodou TextRank, pomocí metrik ROUGE

	Varianta	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**	**ROUGE-W**	**ROUGE-SU4**
A	NER práh 0,2	0,31481	0,25234	0,30555	0,16303	0,24947
B	NER práh 0,5	0,31481	0,25234	0,30555	0,16303	0,24947
C	NER práh 0,8	0,41353	0,25462	0,39614	0,18522	0,26104
D	NER věty 0,2	0,42136	0,2607	0,40356	0,18769	0,267
E	NER věty 0,5	0,52453	0,45202	0,52201	0,28514	0,43181
F	NER věty 0,7	0,51035	0,46002	0,50725	0,28125	0,43337
G	práh 0,2	0,44271	0,33071	0,4323	0,20248	0,32092
H	práh 0,5	0,46439	0,35479	0,45726	0,22568	0,34438
I	práh 0,8	0,49689	0,45171	0,49586	0,27625	0,42728
J	věty 0,2	0,48903	0,3428	0,47231	0,21544	0,34042
K	věty 0,5	0,46954	0,35602	0,4627	0,2289	0,34579
L	věty 0,7	0,49689	0,45171	0,49586	0,27625	0,42728

Testování metody Latentní sémantická analýza pomocí metrik ROUGE

Stejně jako v případě testování metody TextRank bylo i pro metodu LSA vytvořeno dvanáct souhrnů z důvodu dvanácti variant této metody. Tabulka 5.4 obsahuje moje hodnocení jednotlivých vytvořených souhrnů. Podle mého posouzení byly nejlépe vytvořeny souhrny pomocí varianty C, tedy s využitím rozpoznávání pojmenovaných entit (NER) a prahem s hodnotou $0,8$. Další zdařilou variantou, která produkovala jedny z kvalitnějších souhrnů, byla varianta K, viz tabulka 5.4. Velice špatné souhrny, z hlediska výběru nedostatečného počtu vět s nepodstatnými informacemi, vytvářely varianty A a G. Obě tyto varianty využívaly práh $0,2$. Z toho lze usoudit, že tato velikost prahu je pro tvorbu souhrnů, na základě ohodnocení vět metodou LSA, nedostatečná.

Celkové hodnocení metody LSA pomocí metrik ROUGE je stejně jako v případě testování metody TextRank součástí přílohy (B) z důvodu rozsáhlosti tabulky. Pro názorný příklad výsledků metrik ROUGE jsou v tabulce 5.5 zobrazeny výsledeky testování prvního článku.

Tabulka 5.4: Subjektivní posouzení kvality souhrnů vytvořených metodou LSA

	Varianta	**1. článek**	**2. článek**	**3. článek**	**4. článek**	**5. článek**
A	NER práh 0,2	3	1	1	1	1
B	NER práh 0,5	3	3	1	3	2
C	NER práh 0,8	3	3	2	4	3
D	NER věty 0,2	4	3	1	1	2
E	NER věty 0,5	2	2	2	3	3
F	NER věty 0,7	2	2	3	2	2
G	práh 0,2	3	1	1	1	1
H	práh 0,5	4	3	1	2	1
I	práh 0,8	2	2	3	2	2
J	věty 0,2	3	3	1	1	1
K	věty 0,5	2	3	4	1	3
L	věty 0,7	2	2	3	2	2

Tabulka 5.5: Hodnocení souhrnu prvního článku, vytvořeného metodou LSA, pomocí metrik ROUGE

	Varianta	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**	**ROUGE-W**	**ROUGE-SU4**
A	NER práh 0,2	0,31481	0,25234	0,30555	0,16303	0,24947
B	NER práh 0,5	0,31481	0,25234	0,30555	0,16303	0,24947
C	NER práh 0,8	0,38508	0,24727	0,37063	0,17163	0,25317
D	NER věty 0,2	0,5673	0,44505	0,55646	0,2874	0,44118
E	NER věty 0,5	0,54409	0,47583	0,54159	0,29436	0,44999
F	NER věty 0,7	0,50968	0,47137	0,50764	0,28243	0,44227
G	práh 0,2	0,31481	0,25234	0,30555	0,16303	0,24947
H	práh 0,5	0,59477	0,50657	0,58823	0,28399	0,48928
I	práh 0,8	0,50717	0,43408	0,50373	0,27722	0,4142
J	věty 0,2	0,61156	0,53977	0,61156	0,32031	0,51819
K	věty 0,5	0,52044	0,4316	0,51797	0,2756	0,4151
L	věty 0,7	0,49085	0,45464	0,48983	0,27476	0,42982

Nejlepší hodnocení souhrnu prvního článku pomocí metrik ROUGE obdržela varianta *J*. Jak jde vidět v tabulce 5.5, jedná se o variantu bez použití rozpoznávání pojmenovaných entit, kde souhrn obsahuje pouze 20 % vět z celkového počtu vět vstupního textu. Nej-

hůře byly hodnoceny varianty A, B a G, a to tak, že všechny tři varianty obdržely stejné hodnocení.

V celkovém hodnocení pomocí metrik ROUGE vyhrála varianta bez použití ropoznávání pojmenovaných entit se 70% počtem vět ze vstupního textu, tedy varianta L. Na druhém místě byla nejlépe hodnocena varianta K a na třetím místě varianta E. Oproti metodě TextRank využití pojmenovaných entit nevede k zásadnímu ovlivnění kvality souhrnů vytvořených metodou LSA.

Po provedení porovnání mých hodnocení s hodnocením metrik ROUGE jsem došel k závěru, že nejhoršími variantami, na kterých se shodují obě dvě hodnocení, jsou varianty A a B. Ty využívají práh o hodnotě $0,2$. Nejlepší hodnocenou variantou je varianta K. Hned za ní je varianta L, tudíž po přidání mého subjektivního hodnocení došlo ke změně pořadí na prvních dvou místech, oproti hodnocení pouze metrikou ROUGE. Na třetím místě zůstává varianta E.

Zhodnocení testování

Jelikož bylo moje testování prováděno na úplně odlišných datech, než například v [12] nebo v [13], nemohu porovnávat jejich dosažené výsledky s výsledky, kterých jsem dosáhl já. Z tohoto důvodu jsem se rozhodl pouze zhodnotit výsledky mého testování.

Jak již bylo řečeno v úvodu podkapitoly 5.1, jedním z hledisek, které jsem zkoumal při testování jednotlivých metod, byl vliv velikosti vstupního textu na konečný souhrn. Podle jednotlivých výsledků na rozsahu vstupního textu nezáleží, spíše záleží na vhodném výběru počtu vět, či prahu. Problém ale nastává při určování tohoto počtu. Podle mého názoru není jednotný způsob, jak určit konkrétní počet vět pro co nejkvalitnější souhrn z důvodu subjektivnosti každého souhrnu.

Pokud bych měl zhodnotit jednotlivé metody, tak metoda TextRank ve spojení s rozpoznáváním pojmenovaných entit vytvářela kvalitní souhrny. Jak podle metrik ROUGE, tak podle mého hodnocení. Metodě LSA k lepším souhrnům rozpoznávání entit nepomohlo. U obou dvou metod bylo využití prahu podle metrik ROUGE spíše ke škodě. Podle mého hodnocení bylo využití prahu v určitých případech vhodné a jeho použití vedlo ke tvorbě kvalitního souhrnu, zejména při použití NER. Tohoto kvalitního souhrnu, ve více případech, dosáhly metody pouze při použití prahu o hodnotě $0,5$ a $0,8$. Z tohoto usuzuji, že použití menšího prahu vede opravdu ke zhoršení kvality výsledného souhrnu.

Kapitola 6

Závěr

Cílem této práce bylo vytvořit webovou aplikaci, která pomocí sumarizačních metod dokáže vytvářet co nejkvalitnější souhrny z pohledu uživatele. Tohoto cíle bylo dosaženo s použitím metod Latentní sémantické analýzy a TextRanku, které jsem vylepšil pomocí rozpoznávání pojmenovaných entit. Především metoda TextRank dosahovala s využitím rozpoznávání pojmenovaných entit, jak na základě subjektivního posouzení, tak s využitím hodnocení pomocí metrik ROUGE, kvalitních výsledků.

V průběhu tvorby této webové aplikace jsem si osvojil práci s programovacím jazykem Python a jeho knihovnami, především pro zpracování přirozeného jazyka. Prozkoumal jsem a naučil se způsoby a metody, kterými lze dosáhnout automatické sumarizace textu. Navrhl jsem webovou aplikaci, která by uživateli měla dávat co největší volnost při tvorbě souhrnu k obrazu svému. Nakonec jsem jednotlivé výstupy implementovaných sumarizačních metod, které webová aplikace využívá pro tvorbu souhrnů, otestoval na reálných článcích. Toto testování ověřilo funkčnost této aplikace provádět sumarizaci nad texty, které se běžně vyskytují na internetu.

Jednou z možností pro budoucí zdokonalení této webové aplikace je využití metod pro tvorbu abstraktů. Tyto metody by mohly pomoci při tvořbě kvalitnějších, kratších a uživatelsky příjemnějších souhrnů, než tomu je při použití metod pro tvorbu extraktů, které byly implementovány v této práci.

Závěrem bych chtěl říci, že metody, které jsem implementoval, společně s mým vylepšením, předčily moje očekávání, které jsem měl před samotnou implementací. A to jak z hlediska dobré návaznosti vět, i když jsem vytvářel pouze extrakty, tak z hlediska výběru podstatných vět do výsledného souhrnu.

Literatura

[1] General Python FAQ. https://docs.python.org/2/faq/general.html, vid. 17.3.2019.

[2] Natural Language Toolkit. https://www.nltk.org/, vid. 17.3.2019.

[3] Scikit-learn. https://scikit-learn.org/stable/, vid. 17.3.2019.

[4] Anttila, P.: *Automatic Text Summarization*. Diplomová práce, Univerzita v Turku, Fakulta budoucích technologií, 2018.

[5] Badry, R. M.; Eldin, A. S.; Elzanfally, D. S.: Text Summarization within the Latent Semantic Analysis Framework: Comparative Study. *International Journal of Computer Applications*, ročník 81, č. 11, Listopad 2013: s. 81(11):40–81(11):45, ISSN 0975-8887, doi:10.5120/14060-2366.
URL https://www.researchgate.net/publication/260456506

[6] Chmelař, P.; Hellebrand, D.; Hrušecký, M.; aj.: Nalezení slovních kořenů v češtině. In *Znalosti 2011: Sborník příspěvků 10. ročníku konference*, VŠB-Technical University of Ostrava, 2011, ISBN 978-80-248-2369-0, s. 66–77.
URL http://www.fit.vutbr.cz/research/view_pub.php.cz?id=9473

[7] Denny, M.; Spirling, A.: Text Preprocessing For Unsupervised Learning: Why It Matters, When It Misleads, And What To Do About It. *Political Analysis*, ročník 26, 03 2018: s. 1–22, doi:10.1017/pan.2017.44.

[8] Gurusamy, V.; Kannan, S.: Preprocessing Techniques for Text Mining. 10 2014.

[9] Kaikhah, K.: Text Summarization Using Neural Networks. *Faculty Publications-Computer Science*, 03 2019.

[10] Khan, A.; Salim, N.: A Review on Abstractive Summarization Methods. *Journal of Theoretical and Applied Information Technology*, ročník 59, č. 1, Leden 2014: s. 59(1):64–59(1):72, ISSN 1992-8645.
URL https://www.researchgate.net/publication/287206659_A_Review_on_Abstractive_Summarization_Methods

[11] Kohout, L.: *Identifikace pojmenovaných entit v textu*. Bakalářská práce, Masarykova univerzita, Fakulta informatiky, Brno, 2012.

[12] Lin, C.-Y.: ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, 2004, str. 10.
URL http://aclweb.org/anthology/W04-1013

[13] Machovec, P.: *Automatická sumarizace textu*. Diplomová práce, Masarykova univerzita, Fakulta informatiky, Brno, 2015.

[14] Steinberger, J.; Ježek, K.: Text Summarization and Singular Value Decomposition. In *Advances in Information Systems*, editace T. Yakhno, Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, ISBN 978-3-540-30198-1, s. 245–254.

[15] Steinberger, J.; Jezek, K.: Evaluation Measures for Text Summarization. *Computing and Informatics*, ročník 28, 01 2009: s. 251–275.

[16] Steinberger, J.; Ježek, K.: Sumarizace textů. In *DATAKON 2010: sborník databázové konference, Mikulov, 16.-19.10.2010*, 2010, ISBN 978-80-7368-424-2, str. 1, [Online; navštíveno 12.11.2018].
URL http://textmining.zcu.cz/publications/SumarizDATAKON.pdf

[17] Vandas, K.: *Metody sumarizace textu*. Bakalářská práce, Univerzita Karlova v Praze, Matematicko-fyzikální fakulta, Praha, 2009.

[18] Zemek, P.: Introduction to Python. 2018, prezentace z předmětu IPP 5.3.2018, VUT FIT.

Příloha A

Celkové výsledky testování metody Latentní sémantická analýza

1. Článek

Tabulka A.1: Hodnocení souhrnu prvního článku, vytvořeného metodou LSA, pomocí metrik ROUGE

	Varianta	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**	**ROUGE-W**	**ROUGE-SU4**
A	NER práh 0,2	0,31481	0,25234	0,30555	0,16303	0,24947
B	NER práh 0,5	0,31481	0,25234	0,30555	0,16303	0,24947
C	NER práh 0,8	0,38508	0,24727	0,37063	0,17163	0,25317
D	NER věty 0,2	0,5673	0,44505	0,55646	0,2874	0,44118
E	NER věty 0,5	0,54409	0,47583	0,54159	0,29436	0,44999
F	NER věty 0,7	0,50968	0,47137	0,50764	0,28243	0,44227
G	práh 0,2	0,31481	0,25234	0,30555	0,16303	0,24947
H	práh 0,5	0,59477	0,50657	0,58823	0,28399	0,48928
I	práh 0,8	0,50717	0,43408	0,50373	0,27722	0,4142
J	věty 0,2	0,61156	0,53977	0,61156	0,32031	0,51819
K	věty 0,5	0,52044	0,4316	0,51797	0,2756	0,4151
L	věty 0,7	0,49085	0,45464	0,48983	0,27476	0,42982

2. Článek

Tabulka A.2: Hodnocení souhrnu druh článku, vytvořeného metodou LSA, pomocí metrik ROUGE

	Varianta	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**	**ROUGE-W**	**ROUGE-SU4**
A	NER práh 0,2	0,32559	0,30323	0,32336	0,15341	0,29988
B	NER práh 0,5	0,6129	0,54012	0,60369	0,28024	0,52592
C	NER práh 0,8	0,58444	0,49387	0,58111	0,28056	0,48038
D	NER věty 0,2	0,6094	0,51917	0,60206	0,27961	0,50842
E	NER věty 0,5	0,60767	0,53846	0,60472	0,30821	0,51965
F	NER věty 0,7	0,57898	0,55898	0,57898	0,30556	0,53405
G	práh 0,2	0,03357	0,00241	0,02878	0,0139	0,00652
H	práh 0,5	0,54074	0,45536	0,53629	0,26258	0,44978
I	práh 0,8	0,56678	0,53473	0,56678	0,30052	0,51084
J	věty 0,2	0,55099	0,46789	0,54642	0,2663	0,46225
K	věty 0,5	0,59259	0,52191	0,59164	0,30481	0,50718
L	věty 0,7	0,5721	0,54211	0,5721	0,30281	0,51855

3. Článek

Tabulka A.3: Hodnocení souhrnu třetího článku, vytvořeného metodou LSA, pomocí metrik ROUGE

	Varianta	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**	**ROUGE-W**	**ROUGE-SU4**
A	NER práh 0,2	0,25143	0,158	0,25143	0,12639	0,16765
B	NER práh 0,5	0,25143	0,158	0,25143	0,12639	0,16765
C	NER práh 0,8	0,65998	0,54321	0,65998	0,31792	0,53538
D	NER věty 0,2	0,50523	0,37406	0,50224	0,26024	0,38029
E	NER věty 0,5	0,70208	0,60377	0,70208	0,35042	0,58545
F	NER věty 0,7	0,66846	0,61081	0,66846	0,3688	0,58515
G	práh 0,2	0,325	0,31646	0,325	0,20323	0,30172
H	práh 0,5	0,55712	0,48025	0,55712	0,29618	0,48207
I	práh 0,8	0,67915	0,63262	0,67915	0,37556	0,60789
J	věty 0,2	0,58297	0,492	0,58297	0,31155	0,49188
K	věty 0,5	0,69959	0,62733	0,69959	0,37818	0,60878
L	věty 0,7	0,67915	0,63262	0,67915	0,37556	0,60789

4. Článek

Tabulka A.4: Hodnocení souhrnu čtvrtého článku, vytvořeného metodou LSA, pomocí metrik ROUGE

	Varianta	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**	**ROUGE-W**	**ROUGE-SU4**
A	NER práh 0,2	0,31152	0,22222	0,31152	0,1673	0,21749
B	NER práh 0,5	0,5614	0,47837	0,5614	0,31655	0,44734
C	NER práh 0,8	0,57971	0,5055	0,57609	0,33599	0,47699
D	NER věty 0,2	0,5614	0,47837	0,5614	0,31655	0,44734
E	NER věty 0,5	0,57971	0,5055	0,57609	0,33599	0,47699
F	NER věty 0,7	0,62798	0,57057	0,625	0,37679	0,52744
G	práh 0,2	0,05304	0,03101	0,05304	0,03283	0,0215
H	práh 0,5	0,51497	0,4202	0,51098	0,30049	0,38213
I	práh 0,8	0,62022	0,56198	0,61749	0,37191	0,5121
J	věty 0,2	0,51323	0,46237	0,51323	0,29692	0,42081
K	věty 0,5	0,58757	0,50286	0,5838	0,341	0,45501
L	věty 0,7	0,64897	0,58929	0,64601	0,38662	0,5428

5. Článek

Tabulka A.5: Hodnocení souhrnu pátého článku, vytvořeného metodou LSA, pomocí metrik ROUGE

	Varianta	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**	**ROUGE-W**	**ROUGE-SU4**
A	NER práh 0,2	0,27368	0,24307	0,26947	0,15723	0,23022
B	NER práh 0,5	0,46417	0,3931	0,45734	0,24622	0,38948
C	NER práh 0,8	0,64685	0,59162	0,64685	0,3336	0,56647
D	NER věty 0,2	0,60123	0,5387	0,60123	0,29396	0,52935
E	NER věty 0,5	0,67015	0,62815	0,67015	0,35612	0,60651
F	NER věty 0,7	0,64317	0,61116	0,64317	0,36877	0,59101
G	práh 0,2	0,34944	0,29699	0,34572	0,16113	0,28097
H	práh 0,5	0,44257	0,38908	0,43919	0,21258	0,37211
I	práh 0,8	0,65527	0,62625	0,65527	0,37454	0,60578
J	věty 0,2	0,51772	0,46657	0,51464	0,25565	0,44339
K	věty 0,5	0,67396	0,632	0,67396	0,38088	0,61717
L	věty 0,7	0,66092	0,63834	0,66092	0,37652	0,60834

Příloha B

Celkové výsledky testování metody TextRank

1. Článek

Tabulka B.1: Hodnocení souhrnu prvního článku, vytvořeného metodou TextRank, pomocí metrik ROUGE

	Varianta	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**	**ROUGE-W**	**ROUGE-SU4**
A	NER práh 0,2	0,31481	0,25234	0,30555	0,16303	0,24947
B	NER práh 0,5	0,31481	0,25234	0,30555	0,16303	0,24947
C	NER práh 0,8	0,41353	0,25462	0,39614	0,18522	0,26104
D	NER věty 0,2	0,42136	0,2607	0,40356	0,18769	0,267
E	NER věty 0,5	0,52453	0,45202	0,52201	0,28514	0,43181
F	NER věty 0,7	0,51035	0,46002	0,50725	0,28125	0,43337
G	práh 0,2	0,44271	0,33071	0,4323	0,20248	0,32092
H	práh 0,5	0,46439	0,35479	0,45726	0,22568	0,34438
I	práh 0,8	0,49689	0,45171	0,49586	0,27625	0,42728
J	věty 0,2	0,48903	0,3428	0,47231	0,21544	0,34042
K	věty 0,5	0,46954	0,35602	0,4627	0,2289	0,34579
L	věty 0,7	0,49689	0,45171	0,49586	0,27625	0,42728

2. Článek

Tabulka B.2: Hodnocení souhrnu druhého článku, vytvořeného metodou TextRank, pomocí metrik ROUGE

	Varianta	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**	**ROUGE-W**	**ROUGE-SU4**
A	NER práh 0,2	0,40753	0,35752	0,40125	0,18127	0,35139
B	NER práh 0,5	0,61135	0,51282	0,60993	0,27924	0,50526
C	NER práh 0,8	0,61227	0,53077	0,60907	0,29641	0,51713
D	NER věty 0,2	0,61262	0,5279	0,60961	0,28424	0,51823
E	NER věty 0,5	0,61484	0,54842	0,6128	0,30993	0,52937
F	NER věty 0,7	0,58344	0,54109	0,58259	0,30256	0,51685
G	práh 0,2	0,01569	0,00527	0,01569	0,00851	0,00357
H	práh 0,5	0,5187	0,39869	0,51219	0,22313	0,39649
I	práh 0,8	0,60074	0,53994	0,59798	0,30424	0,51876
J	věty 0,2	0,49428	0,40849	0,489	0,22023	0,39934
K	věty 0,5	0,61121	0,49043	0,60428	0,28223	0,47522
L	věty 0,7	0,59991	0,5392	0,59717	0,30391	0,51804

3. Článek

Tabulka B.3: Hodnocení souhrnu třetího článku, vytvořeného metodou TextRank, pomocí metrik ROUGE

	Varianta	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**	**ROUGE-W**	**ROUGE-SU4**
A	NER práh 0,2	0,17316	0,0921	0,16882	0,08097	0,0979
B	NER práh 0,5	0,25143	0,158	0,25143	0,12639	0,16765
C	NER práh 0,8	0,67793	0,5669	0,67793	0,33569	0,55313
D	NER věty 0,2	0,35447	0,19047	0,35122	0,15766	0,20756
E	NER věty 0,5	0,68134	0,59283	0,68134	0,3649	0,57747
F	NER věty 0,7	0,66846	0,61081	0,66846	0,3688	0,58515
G	práh 0,2	0,23457	0,125	0,23045	0,10745	0,13262
H	práh 0,5	0,4914	0,39179	0,4914	0,24507	0,3772
I	práh 0,8	0,70105	0,66667	0,69914	0,38438	0,63433
J	věty 0,2	0,46566	0,36549	0,46566	0,22896	0,36202
K	věty 0,5	0,58304	0,49591	0,58072	0,30613	0,46864
L	věty 0,7	0,69574	0,66484	0,69574	0,38385	0,63441

4. Článek

Tabulka B.4: Hodnocení souhrnu čtvrtého článku, vytvořeného metodou TextRank, pomocí metrik ROUGE

	Varianta	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**	**ROUGE-W**	**ROUGE-SU4**
A	NER práh 0,2	0,31152	0,22222	0,31152	0,1673	0,21749
B	NER práh 0,5	0,45014	0,35942	0,45014	0,24912	0,33631
C	NER práh 0,8	0,57197	0,47126	0,56818	0,32539	0,44466
D	NER věty 0,2	0,50793	0,43011	0,50793	0,28098	0,40423
E	NER věty 0,5	0,57197	0,47126	0,56818	0,32539	0,44466
F	NER věty 0,7	0,52764	0,45295	0,52449	0,31378	0,42356
G	práh 0,2	0,11347	0,02898	0,10639	0,05618	0,03634
H	práh 0,5	0,46296	0,35681	0,45833	0,25786	0,32772
I	práh 0,8	0,55327	0,47222	0,54983	0,32396	0,44229
J	věty 0,2	0,42778	0,34463	0,42778	0,24136	0,31395
K	věty 0,5	0,47742	0,36165	0,47311	0,26382	0,34002
L	věty 0,7	0,47359	0,36832	0,46994	0,26828	0,34271

5. Článek

Tabulka B.5: Hodnocení souhrnu pátého článku, vytvořeného metodou TextRank, pomocí metrik ROUGE

	Varianta	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**	**ROUGE-W**	**ROUGE-SU4**
A	NER práh 0,2	0,27368	0,24307	0,26947	0,15723	0,23022
B	NER práh 0,5	0,59945	0,5376	0,59945	0,29728	0,5193
C	NER práh 0,8	0,64685	0,59162	0,64685	0,3336	0,56647
D	NER věty 0,2	0,4586	0,37621	0,44904	0,24198	0,36819
E	NER věty 0,5	0,67015	0,62815	0,67015	0,35612	0,60651
F	NER věty 0,7	0,63955	0,59372	0,63955	0,36084	0,57412
G	práh 0,2	0,08373	0,0	0,08373	0,03782	0,02013
H	práh 0,5	0,57856	0,47739	0,57856	0,31296	0,47625
I	práh 0,8	0,64488	0,61279	0,64488	0,36933	0,59558
J	věty 0,2	0,32225	0,25515	0,44602	0,22794	0,35641
K	věty 0,5	0,61099	0,52535	0,61099	0,33318	0,51399
L	věty 0,7	0,63702	0,59854	0,63702	0,36465	0,58257