Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Text miningové nástroje pro tematickou analýzu česky psaných textů

Diplomová práce

Autor: Bc. Jakub Nevyhoštěný
Studijní obor: Datová věda

Vedoucí práce: Mgr. Jiří Haviger, Ph.D.

Hradec Králové Duben 2024

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 16.4.2024 Jakub Nevyhoštěný

Poděkování:

Děkuji vedoucímu diplomové práce Mgr. Jiřímu Havigerovi, Ph.D. za metodické vedení práce, ochotu a cenné rady. Dále děkuji své rodině, přítelkyni a přátelům za jejich důvěru a podporu po celou dobu studia.

Abstrakt

Tato diplomová práce se zaměřuje na průzkum specifik českého jazyka ve spojitosti se strojovým zpracováním textu. Dále poskytuje komplexní přehled metod a nástrojů dílčích kroků pro předzpracování českých a anglických textů, a jejich převod do vektorové reprezentace. S důkladným rozborem problematiky tematické analýzy a jejích podoblastí představuje několik nástrojů v podobě modelů pro modelování témat a klasifikátorů pro klasifikaci témat. Po kompletním uvedení do oblasti problematiky jsou postupně tyto nástroje pro předzpracování, vektorizaci a klasifikaci či modelování aplikovány na skutečná česká textová data. Následně jsou zkoumány a porovnány jejich výhody, nevýhody a výsledky. V závěru práce je na základě výsledků navržen postup pro uživatele, kteří mají zájem o provedení tematické analýzy česky psaných textů.

Abstract

Title: Text mining tools for topic discovery of Czech written texts

This Diploma Thesis focuses on the specifics of the Czech language in relation to machine text processing. Furthermore, it provides a comprehensive overview of methods and tools for preprocessing Czech and English texts and their conversion into vector representation. With a thorough study of topic analysis and its sub-areas, it presents several tools in the form of topic modelling models and topic classifiers. After a complete introduction to the subject area, these tools for preprocessing, vectorization and classification or modeling are gradually applied to real Czech text data. Subsequently, their advantages, disadvantages and results are examined and compared. The paper concludes by proposing a procedure based on the results for users interested in performing topic analysis of Czech written texts.

Klíčová slova: český jazyk, tematická analýza, modelování témat, klasifikace témat, předzpracování textu, vektorizace, python

Key words: Czech language, topic analysis, topic modeling, topic classification, text preprocessing, vectorization, python

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Vzhledem k narůstající popularitě informačních technologií se v posledních desetiletích výrazně navýšil objem světových dat. V roce 2020 dosáhl objem dat přibližně 44 zettabytů a odhaduje se, že do roku 2025 se tento objem zvýší více než čtyřnásobně. Tato data pocházejí z rozmanitých zdrojů, jako jsou sociální sítě, e-maily, zpravodajské portály, odborné databáze, lékařské záznamy, bankovní informace, obchodní platformy a mnoho dalších. [26] S takovým obrovským množstvím dat je pro člověka nemožné je rychle a efektivně ručně zpracovat, neboť by to bylo časově náročné, finančně nákladné, neefektivní, a více náchylné k chybám a nesrovnalostem. Právě zde vstupuje do hry strojové zpracování textu, které umožňuje automatickou extrakci významu z textových dat identifikací opakujících se slov.

Jedním z příkladů může být filtrování e-mailů. V roce 2023 přicházelo a odcházelo denně přibližně 340 miliard e-mailů, z nichž 56,5 % tvořily nevyžádané zprávy (jako reklama nebo obsah pro dospělé). [3] Představa, že by takové množství emailů měl přebírat, pročítat a posuzovat člověk, je přinejmenším nereálná. Stroj takový email či recenzi prozkoumá pomocí analýzy klíčových slov a odkazů, ověří adresu odesílatele a posoudí autenticitu e-mailu, a to v řádu milisekund.

Automatizace těchto procesů prostřednictvím tematické analýzy řízené umělou inteligencí výrazně usnadňuje, urychluje a zpřesňuje analýzy nestrukturovaného textu. Tento přístup je aplikovatelný na různé typy dat, včetně emailové komunikace, příspěvků na sociálních sítích a recenzí. Tímto způsobem mohou firmy identifikovat témata, která jejich zákazníci nejčastěji zmiňují v souvislosti s jejich produktem, značkou nebo službou.

V současné době existuje řada nástrojů určených k provedení tematické analýzy. Každý z těchto nástrojů má své specifické vlastnosti a zaměřuje se na rozdílné aspekty analýzy textových dat, a proto je výběr vhodného nástroje klíčový. Tato práce hodnotí a porovnává různé implementace modelů a klasifikátorů pro tematickou analýzu, zkoumá a porovnává jejich výhody, nevýhody a výsledné výkony. Cílem je usnadnit uživatelům rozhodování při výběru optimálního nástroje pro jejich specifické potřeby v oblasti tematické analýzy textových dat.

2 Cíl a metodika práce

Cílem této práce je poskytnout podrobný přehled metod a nástrojů určených pro předzpracování a vektorizaci textových dat v češtině a angličtině. Dále detailně popsat problematiku tematické analýzy a představit modelovací a klasifikační modely, klasifikátory a nástroje, které jsou k jejímu provedení určené. Tyto nástroje následně aplikovat na skutečná česká textová data a porovnat jejich výhody, nevýhody a výsledné výkony. Nakonec na základě těchto výsledků navrhnout postup, kterým by se měl ubírat uživatel, jenž chce provést tematickou analýzu česky psaných textů.

V oblasti klasifikace témat budou v této práci zkoumány metody strojového učení zastoupené dvěma knihovnami: Natural Language Toolkit a Scikit-learn. Dále bude aplikován jeden zástupce hlubokého učení, konkrétně Keras. Pro strojové učení budou využity klasifikátory Naivní Bayesovský klasifikátor, Metoda podpůrných vektorů, Logistická regrese a Rozhodovací strom. Poslední zmíněný klasifikátor v případě knihovny Scikit-learn při testování končil s chybou odkazující na nedostatek paměti pro běh programu, a nebude proto součástí výsledků této knihovny. V kontextu modelování témat budou zkoumány tři knihovny, a to Gensim, Scikit-Learn a Tomotopy, a jejich implementace dvou předních modelů v oblasti detekce témat: Latent Semantic Analysis (často označováno též jako Latent Semantic Indexing) a Latent Dirichlet Allocation. Existují ovšem i další způsoby provedení dílčích částí tematické analýzy, například pomocí Large Language Model, ale těmi se tato práce nezabývá. K dispozici jsou také platformy typu Software as a Service (SaaS), vhodné pro uživatele bez programovacích dovedností, které dokáží provést klasifikaci či modelování témat, ale ty nejsou předmětem této práce.

Testování vybraných nástrojů bude realizováno v programovacím jazyce Python, který je široce uznáván pro svou efektivitu v oblasti analýzy dat a strojového učení. Konkrétně bude použita verze Pythonu 3.10, avšak tuto verzi v době vypracovávání této práce knihovna Keras zatím nepodporuje, a proto pro ni bude použita kompatibilní verze 3.8. Pro ladění chyb v kódu bude využita podpora umělé inteligence, konkrétně nástroj Chat-GPT 3.5 vyvinutý společností OpenAI.

3 Teoretický úvod

V této kapitole budou podrobně popsána specifika českého jazyka v kontextu strojového zpracování textu. Bude zde poskytnut detailní přehled teoretických základů a metod pro tematickou analýzu, která zahrnuje klasifikaci a modelování témat. Dále budou rozebrány jednotlivé kroky předzpracování a vektorizace textů.

3.1 Specifika českého jazyka

Čeština, patřící do západoslovanského jazykového kontextu indoevropské jazykové rodiny, je ovlivněna latinským a německým jazykem. Je charakterizována jako flektivní jazyk s diakritickými znaky a komplexním systémem skloňování a časování.

Čeština rozlišuje deset slovních druhů — Podstatná a přídavná jména, zájmena, číslovky, slovesa, příslovce, předložky, spojky, částice a citoslovce. Příslovce jsou často odvozena od přídavných jmen odstraněním koncovky "ý" nebo "í" a nahrazením jinými koncovkami, jako jsou "e", "ě" nebo "o". Záporná slova, která se v jedné větě mohou objevovat vícekrát, jsou vytvořena přidáním předpony "ne" k danému slovu.

Posledních pět slovních druhů jsou neohebné, zatímco prvních pět slovních druhů jsou ohebné a jejich význam může být upraven skloněním do jednoho ze sedmi pádů — nominativu, genitivu, dativu, akuzativu, vokativu, lokálu a instrumentálu, přičemž nominativ singuláru slouží jako základní tvar (lemma).

Slovesa jsou časována do tří časů — minulého, přítomného a budoucího, s infinitivem jako základním tvarem. Vid, který rozlišuje ukončení děje, může být dokonavý nebo nedokonavý. Slovesa se také rozlišují podle způsobů — oznamovací, podmiňovací, rozkazovací, a podle vztahu k podmětu — činný nebo trpný.

Čeština také rozlišuje 3 jmenné rody — mužský, který se dále dělí na živý a neživý, ženský a střední. Rod může být mimo jiné ovlivněn také koncovkami slov v minulém čase.

Rozlišuje se také mluvnické číslo na jednotné a množné, avšak mohou se také vyskytovat duály. [24]

V porovnání s angličtinou je čeština v mnoha ohledech složitější. Pořadí slov v češtině je flexibilní, protože k vyjádření funkce slova ve větě využívá pád namísto pořadí slov jako angličtina. Angličtina nemá žádné pády a tedy žádné skloňovací vzory, nemá gramatické rody, slovesné třídy, neboť se (až na pár výjimek) slovesa časují stejně, a neustále se zjednodušuje a zbavuje zbytečných pravidel. [25] Tyto rozdíly ukazují, že při zpracovávání českých textů není možné na ně aplikovat stejné metody

předzpracování, klasifikace či modelování jako na anglické texty, neboť některé tyto metody, které jsou pro anglické texty původně určeny, nepočítají s tolika složitostmi, které s sebou čeština nese.

Je zapotřebí odlišného způsobu porozumění přirozenému jazyku strojem, při kterém se snaží analyzovat strukturu věty, extrahovat její význam, identifikovat a kvantifikovat subjektivní informace atd; což jsou metody, které mohou být při tematické analýze (především při modelování témat) užitečné. [10] Výzkumy ukazují, že význam textu lze odvodit z různých jazykových úrovní. [13]

Například morfologická úroveň se zabývá vnitřní strukturou slov, tedy jejich tvaroslovím, a zkoumá všechny typy morfémů z hlediska jejich kombinací, forem a funkcí. Morfém je nejmenší jazyková jednotka nesoucí význam (je tvořen kombinací fonémů) a spolu s ohýbáním (skloňováním) či časováním může měnit tvar slova prostřednictvím předpon (prefix), přípon (sufix), vpon (infix) atd. Stroj může morfologicky analyzovat dané slovo, rozpoznat význam každého z morfémů a následně i slova samotného. V angličtině je například takto možné pomocí přípony "-ed" u většiny sloves rozpoznat, že se děj slovesa odehrál v minulosti. [13] V češtině je například možné pomocí předpony "ne-" identifikovat záporná slova.

Lexikologická úroveň se zase zabývá všemi variantami slov a slovních spojení neboli slovní zásobou jazyka. [13] Například slovo „jazyk“ může referovat k jazyku v ústech, části obuvi nebo také k řeči. Správný význam slova je odvozen až na základě kontextu, ve kterém je použito v rámci celé věty.

Na syntaktické úrovní se provádí analýza, která umožňuje rozpoznat odlišné významy ve větách, které mohou sdílet identická slova. Jedním z důležitých aspektů je schopnost rozlišit mezi subjektem a předmětem ve větě, tedy identifikovat, kdo konkrétní činnost provádí a koho se týká. [13] Například věty „Pes honí kočku.“ a „Kočka honí psa.“ sdílejí sice stejná slova, ale jejich význam je vzhledem k rozdílné syntaxi odlišný. Syntaktická analýza umožňuje stroji identifikovat strukturální vztahy mezi slovy, což vede k rozlišení významu vět a poskytuje přesnější a vhodnější interpretaci obsahu. [13] Na česky psaném textu je možné tuto analýzu provést například s využitím knihovny Stanza obsahující model pro češtinu, jehož výstupem pro jednotlivá slova mohou být slovní druh a závislostní vztahy.

Další metodou analýzy je sémantická analýza, která umožňuje stroji interpretovat skutečný význam slov na základě jejich kontextu, ve kterém se vyskytují, a tím identifikovat synonyma, antonyma a další jazykové vztahy. [13] Například, pokud se

slova „učitel“ a „pedagog“ často objevují v podobných kontextech, stroj s velkou pravděpodobností určí, že mají podobný význam. Poslední, pragmatická (někdy též logická nebo praktická) úroveň, se zabývá významem slov a vět s využitím situačního povědomí, znalostí z reálného světa a porozuměním toho, co je sdělováno a jaký význam by mohl nejlépe zapadnout do daného kontextu. Hlavním cílem je porozumění kontextu za hranicemi pouhého textu a vyčíst z něho další význam, který v něm není explicitně vyjádřen. [13] Například věta „Ty jsi k sežrání.“ nenaznačuje, že autor věty doslovně hodnotí něčí chuťové kvality či má záměr dotyčnou osobu konzumovat. Skrytý význam v tomto kontextu naznačuje, že autor danou osobu, které je věta adresována, považuje za atraktivní.

3.2 Tematická analýza

Tematická analýza (angl. topic analysis) představuje metodu strojového učení, která má za cíl porozumět obsáhlým textovým souborům a kategorizovat je do tematických celků. Za použití technik zpracování přirozeného jazyka identifikuje vzory a sémantické struktury v textu, které následně přiřazuje jednotlivým tématům. Tato analýza využívá pokročilé algoritmy k extrakci slov a identifikaci různých skupin na základě podobných slovních vzorců. [27] Tematickou analýzu lze provádět na různých úrovních:

- Úroveň dokumentu — Tento model identifikuje témata, která se vyskytují napříč celým textem, jako jsou témata obsažená v e-mailových zprávách nebo v novinových článcích.
- Úroveň věty — Tento model zaměřuje svou analýzu na témata obsažená v jednotlivých větách, například témata zahrnutá v nadpisech novinových článků.
- Úroveň dílčích vět — Tento model se zaměřuje na extrakci témat z menších částí jednotlivých vět, jako jsou různá témata obsažená v rámci uživatelských recenzí produktů. [11]

Tematická analýza nachází uplatnění v situacích, kde je potřeba rychle analyzovat velké objemy textových dat, jejichž manuální klasifikace nebo modelování by bylo časově náročné, finančně nákladné nebo méně přesné. Firmy využívají tuto techniku k optimalizaci svých interních procesů, automatizaci obchodních operací či získávání důležitých poznatků. Existují dva hlavní přístupy k tematické analýze: klasifikace témat

a modelování témat. Volba mezi těmito přístupy závisí na konkrétním úkolu a dostupných datech. [27] Ke správnému rozhodnutí je důležité porozumět hlavnímu rozdílu mezi nimi, kterým je typ strojového učení, jež je základem jejich fungování:

- Učení bez učitele je proces, kdy algoritmus identifikuje skryté struktury, vzory, vztahy a souvislosti v datech bez předchozího zadání informací. [8] Tento typ strojového učení se využívá nejen pro modelování témat, ale také pro úkoly jako je shlukování.
- Naopak učení s učitelem zahrnuje proces, kde je algoritmus předem natrénován na označených datech (angl. labeled data), tzv. trénovacích datech, kterým jsou na vstupu přiřazeny odpovídající štítky (v tomto případě názvy jednotlivých témat). Čím více trénovacích dat je k dispozici, tím přesnější a efektivnější je model. Po natrénování může algoritmus klasifikovat nová, dosud neoznačená data. [8]

3.3 Klasifikace témat

Klasifikace témat (v obecné rovině klasifikace textu) představuje proces tematické analýzy založený na metodě učení s učitelem, kde je cílem kategorizovat textové dokumenty do předem definovaných témat. Obtížnost určení témat závisí na konkrétních datech, a proto je důležité provést základní analýzu textu a porozumět mu, aby byla vybrána adekvátní témata. V některých případech je vhodné provést nejprve modelování témat (viz 3.4) a na jeho výsledky navázat. V praxi se stává, že jsou témata odhalena až během vytváření samotného modelu. Pro efektivní fungování modelu je důležité mít pevně stanovená témata a konzistentní trénovací data. [11] Jakmile je seznam témat sestaven, použije se k označení trénovacích dat, na kterých se model naučí témata rozpoznávat. Tento proces je nazýván trénování, během něhož se model učí rozpoznávat vzory a charakteristiky textu, které jsou typické pro jednotlivá témata, což mu umožňuje klasifikovat nové, dosud nezaznamenané dokumenty. Trénovací a testovací data by měla být rozdělena přibližně v poměru 80:20. Příliš velké množství trénovacích dat může vést k přetrénování modelu a snížení jeho úspěšnosti v klasifikaci. Natrénovaný model lze poté použít na reálných datech. Klasifikace témat umožňuje správu a analýzu velkých kolekcí textových dat, například analýzu recenzí produktů a jejich rozdělení do kategorií "pozitivní" a "negativní". Klasifikaci témat lze rozdělit do tří skupin podle počtu tříd (témat), do kterých může být text zařazen:

- Binární klasifikace — Tento proces kategorizuje text do dvou tříd (témat), například určení spamové zprávy, zdravého či nemocného pacienta atd.
- Multi-class klasifikace — Tento typ klasifikace přiřazuje text právě jedné z více než dvou tříd (témat), přičemž se tyto třídy vzájemně nepřekrývají. Příkladem může být kategorizace recenzí zákazníků podle jejich hodnocení.
- Multi-label klasifikace — Tento proces přiřazuje textu více než jednu možnou třídu (téma), například rozpoznání žánru filmu na základě jeho popisu.

3.3.1 Klasifikátory

Naivní Bayesovský klasifikátor

Naivní Bayesovský klasifikátor představuje pravděpodobnostní algoritmus, který využívá Bayesovu větu a teorii pravděpodobnosti k predikci tématu textu. Principem je výpočet pravděpodobnosti zařazení každého dokumentu do určitého tématu pomocí Bayesovy věty, tedy na základě znalosti předchozích podmínek. Poté je dokument přiřazen do tématu s nejvyšší vypočítanou pravděpodobností. [11] Tento klasifikátor lze použít jak pro binární, tak pro multi-klasifikaci. Pro stanovení tématu dokumentu je pro každé téma vypočítána pravděpodobnost, že se o dané téma jedná. Odpovědí je pak vyšší z těchto hodnot. Tento klasifikátor nezohledňuje pořadí slov a strukturu věty, a každý dokument chápe pouze jako soubor nějakého počtu slov. Právě četnost slov je zde hlavním faktorem, který je využit pro výpočet požadovaných pravděpodobností. Jak již bylo zmíněno výše, kromě některých základních vlastností pravděpodobnosti se zde používá Bayesova věta pro podmíněnou pravděpodobnost. Pravděpodobnost výskytu každého slova v testovacím dokumentu v každém z témat je vypočtena jako podíl takové četnosti a celkového počtu slov v daném tématu. Avšak ne každé slovo se vyskytuje ve všech tématech, což by mohlo vést k celkové nulové pravděpodobnosti. K prevenci této situace se využívá Laplaceovo vyhlazování (angl. Laplace smoothing), které ke každé četnosti přidává jedničku a k děliteli přičítá počet unikátních slov v trénovacích datech.

Metoda podpůrných vektorů

Metoda podpůrných vektorů (angl. Support Vector Machines) je o něco složitější než předchozí metoda a nabízí lepší výsledky, nicméně je také náročnější na implementaci a výpočetní zátěž. [11] Základní princip spočívá v mapování slov v numerické podobě (viz 3.6) do N-dimenzionálního prostoru (často 2D pro binární klasifikaci), kde je nalezena dělící hranice, oddělující příklady příslušející k různým

třídám (tematům). Tato hranice je označována jako nadrovina (angl. separating hyperplane nebo decision boundary), kterou může být například lineární přímka ve 2D prostoru. Při klasifikaci nového textu stroj na základě příznaků z trénovacích dat určuje, na které straně přímky (nadroviny) se bude text nacházet. Optimální nadrovina je taková, která má maximální odstup mezi příklady a nadrovinou (tzv. maximum margin). Příklady, které jsou nejblíže nadrovině, definují klasifikátor a nazývají se podpůrné vektory (angl. support vectors). To, jakým způsobem lze nebo nelze data rozdělit určuje tzv. separabilita dat. Při nerozdělitelných datech se může použít jádrová transformace z 2D do 3D, což převede prostor příznaků do vyšší dimenze, čímž se data stávají separovatelnými a je možné nalézt dělící nadrovinu. Tento algoritmus je vhodný spíše pro menší datasety s dobře separovatelnými třídami.

Logistická regrese

Naivní Bayesovský klasifikátor využívá pravděpodobnosti na základě výskytu příznaků v jednotlivých třídách, zatímco logistická regrese kategorizuje příznaky podle jejich důležitosti. Cílem logistické regrese je vytvořit model, který co nejlépe popisuje vztah mezi vysvětlovanou (predikovanou) proměnnou a skupinou vysvětlujících (predikujících) proměnných a poté nalézt logistickou funkci, která nejlépe odděluje jednotlivé třídy v trénovacích datech. V případě binární klasifikace má logistická funkce tvar sigmoidu. Tento algoritmus umožňuje odhadnout pravděpodobnost výskytu sledovaného jevu na základě hodnot vysvětlujících proměnných a následně posoudit míru jejich vlivu. Přizpůsobením nových dat logistické funkci algoritmus předpovídá pravděpodobnost výskytu daného jevu. Logistická regrese může být použita jak pro binární, tak pro multi-class nebo multi-label klasifikaci. [14]

Rozhodovací strom

Rozhodovací strom (angl. Decision Tree) je jednoduchý algoritmus, jehož název odkazuje na jeho grafickou podobu. Hypotézu tohoto algoritmu představuje graf typu strom, což znamená, že je souvislý a neobsahuje cykly. Strom se skládá z větví a listů, které reprezentují zařazení objektů do tříd. [11] Při tvorbě rozhodovacího stromu se data postupně rozdělují na podmnožiny na základě hodnot jejich příznaků. Příznak s největším informačním přínosem, tedy ten, který nejlépe rozděluje příklady, je zvolen jako kořen stromu. Poté je množina dat rozdělena do podmnožin (větví) podle hodnot kořenového příznaku. Tento proces se opakuje v každé další podmnožině až do vyčerpání příznaků. Výsledný strom by neměl být příliš rozsáhlý, neboť by nebyl snadno srozumitelný a klasifikace by trvala příliš dlouho. Příliš velký strom je možné

zjednodušit pomocí tzv. prořezávání. Je důležité vyhnout se přetrénování, kdy je strom příliš přesný a přesné odpovídá trénovacím datům, což může ovlivnit jeho schopnost kategorizace nových dat. Klasifikací se rozumí postupné procházení stromu, při kterém se na každé větvi provádí test určující, do které větve patří testovaný objekt. Takto se pokračuje, dokud není objekt přiřazen do příslušného listu. Rozhodovací strom je velmi jednoduše pochopitelný a interpretovatelný algoritmus, pro jehož využití není potřeba nikterak náročná předpříprava dat.

Neuronové sítě

Neuronová síť představuje výpočetní model, který se volně inspiruje strukturou a funkcí lidských neuronů. Základní stavební jednotkou jsou neurony, které přijímají vstupní signály (číselné hodnoty představující vlastnosti dat), zpracovávají je a generují výstupní signály. Každý neuron má několik vstupů ohodnocených váhami (násobiteli vstupu) a jeden výstup. Pro úpravu vstupních hodnot na základě těchto vah používá neuron aktivační funkci, čímž zavádí do sítě nelinearitu a umožňuje jí učit se složitým vztahům v datech. Výstup jednoho neuronu slouží jako vstup pro jeden nebo více dalších neuronů. Neurony jsou organizovány do jedné ze tří vrstev:

- Vstupní vrstva — První vrstva sítě, do které jsou přiváděna vstupní data.
- Skrytá vrstva — Mezivrstva umístěná mezi vstupní a výstupní vrstvou. Skryté vrstvy se starají o učení a reprezentaci komplexních vzorů v datech. Síť může být mělká (angl. shallow) s jednou nebo dvěma skrytými vrstvami, nebo hluboká (angl. deep) s více vrstvami.
- Výstupní vrstva — Poslední vrstva sítě, která vytváří výstup. Počet neuronů v této vrstvě závisí na konkrétní úloze. V klasifikačních úlohách může každý neuron reprezentovat jedno téma.

Neurony v sousedních vrstvách jsou propojeny hranami s váhami, které určují sílu spojení mezi neurony. Tyto váhy se upravují během trénování sítě pomocí procesu zpětné propagace. Tento proces zahrnuje úpravu vah na základě rozdílu mezi predikovanými hodnotami sítě a skutečnými hodnotami v trénovacích datech. Architektura sítě, tedy způsob, jak jsou neurony propojeny, definuje topologii sítě. [11] Neuronové sítě se často trénují na velkém množství dat a jsou schopny se adaptovat na širokou škálu problémů.

3.3.2 Evaluace

Pro posouzení výkonnosti modelu je nezbytné provést testování na předem označených datech, aby bylo možné porovnat predikované a skutečné hodnoty. Vhodným přístupem je křížová validace, která rozděluje celou datovou sadu s označenými daty na trénovací a testovací části v poměru přibližně 80:20. Model je trénován na trénovací části a jeho výkonnost je testována na zbývající testovací části. Kvalita modelu je poté vyhodnocena pomocí klasifikačních metrik, které vychází z matice záměn (angl. confusion matrix). Tato matice poskytuje přehled o tom, jak přesně model predikuje pozitivní a negativní případy. [11]

Tab. 1 Matice záměn. Zdroj: Autor

Klasifikátor		Predikce (výsledek klasifikace)	
		Pozitivní	Negativní
Skutečnost	Pozitivní	SP	FN
	Negativní	FP	SN

Matice záměn, jejíž podobu lze vidět v Tab. 1, obsahuje čtyři základní pole:

- Skutečně pozitivní (SP) — Správně klasifikovaný pozitivní případ,
- Skutečně negativní (SN) — Správně klasifikovaný negativní případ,
- Falešně pozitivní (FP) — Falešně klasifikovaný pozitivní případ,
- Falešně negativní (FN) — Falešně klasifikovaný negativní případ.

Využitím těchto výsledků lze vypočítat následné hodnotící metriky:

- Správnost (angl. accuracy): Podíl správně klasifikovaných případů. Výpočet:
$$\frac{SN + SN}{SP + FP + SN + FN}$$
- Přesnost (angl. precision): Podíl správně klasifikovaných pozitivních případů mezi všemi predikovanými pozitivními případy. Výpočet:
$$\frac{SP}{SP + SN}$$
- Senzitivita (angl. recall): Podíl správně klasifikovaných pozitivních případů mezi všemi skutečně pozitivními případy. Výpočet:

$$\frac{SP}{SP + FN}$$

- Specificita (angl. specificity): Podíl správně klasifikovaných negativních případů mezi skutečně negativními případy. Výpočet:

$$\frac{SN}{FP + SN}$$

- Chybovost (angl. error rate): Podíl falešně klasifikovaných případů. Výpočet:

$$\frac{FN + FP}{SP + FN + FP + SN}$$

- F-skóre (angl. F1-Score): Harmonický průměr mezi přesností a senzitivitou. Výpočet:

$$2 * \frac{Přesnost * Senzitivita}{Přesnost + Senzitivita}$$

3.3.3 Nástroje

V současné době je k dispozici řada volně dostupných nástrojů pro implementaci klasifikátorů textů, a mnoho z nich je orientováno na programovací jazyk Python. Tato kapitola se zaměří pouze na tyto klasifikační nástroje. I když existuje široká škála nástrojů, níže jsou uvedeny ty nejčastěji používané a z pohledu autora ty nejintuitivnější. Zároveň budou tyto nástroje použity v praktické části práce.

- Natural Language Toolkit (NLTK) je volně dostupná knihovna, zaměřená na zpracování přirozeného jazyka, která nabízí mnoho metod pro tokenizaci, stemování, tagování, parsing a klasifikaci textu. Díky své výkonnosti a uživatelsky přívětivé povaze je vhodná jak pro začátečníky, tak i pro pokročilé v oblasti zpracování textu. [2]
- Scikit-learn je rychlá a snadno použitelná knihovna, postavená na knihovnách NumPy a SciPy, která poskytuje kompletní sadu metod pro klasifikaci textu. Pokrývá celý proces od předzpracování a čištění dat, přes trénování modelů až po vyhodnocení výsledků. [21]
- Keras je flexibilní a výkonná knihovna pro hluboké učení, poskytující jednoduché rozhraní pro vytváření a trénování neuronových sítí v jazyce Python. I když není primárně zaměřena na zpracování textových dat, umožňuje efektivní práci s neuronovými sítěmi a je kompatibilní s několika hlavními frameworky pro hluboké učení, jako jsou Jax, TensorFlow nebo PyTorch. [7]

3.4 Modelování témat

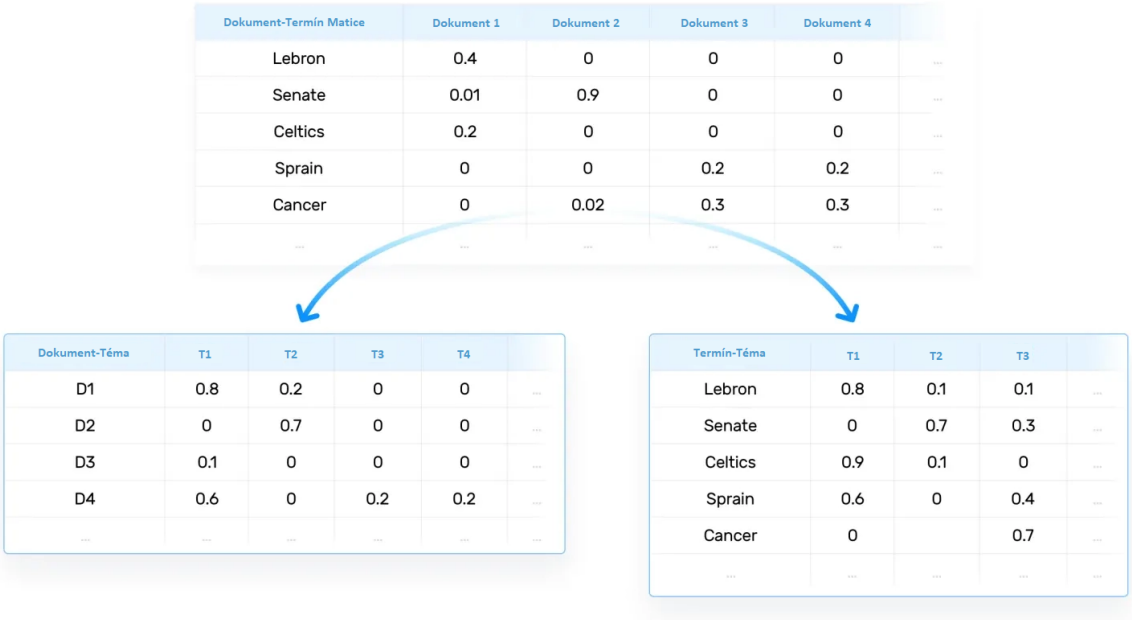
Modelování témat (angl. topic modeling) představuje techniku využívající statistického modelování a technik strojového učení k odhalování skrytých témat v nestrukturovaných textech. Jedná se o metodu strojového učení bez učitele, takže nemá k dispozici strojově čitelné štítky jako vodítko pro rozlišení témat dokumentů. Tyto algoritmy fungují na předpokladu, že každý dokument je složen ze směsi různých témat, pro které se snaží zjistit sílu jejich zastoupení v daném dokumentu. Toto vyhodnocení se provádí na základě frekvence slov v dokumentech a vzájemných vztahů mezi nimi. Po natrénování modelu je každé téma reprezentováno pravděpodobnostním rozdělením nad slovy. Tato slova představují nejpravděpodobnější výrazy, které se objevují v dokumentech o daném tématu. Uživatelé mohou tato rozdělení slov interpretovat a porozumět tak tematickému obsahu souboru dokumentů. Tato technika je velmi blízká shlukové analýze, přičemž zde mohou jednotlivé shluky reprezentovat určité téma. Modelování témat nachází uplatnění zejména v situacích, kdy uživatelé disponují rozsáhlými textovými daty, jako jsou například emaily, zákaznické recenze, výsledky dotazníků a podobně, a chtějí identifikovat témata, která se v těchto dokumentech vyskytují. [10] Při analýze zákaznických recenzí mohou tato témata zahrnovat například rozlišení mezi pozitivními a negativními recenzemi. Na základě tohoto rozlišení mohou společnosti přijímat vhodná opatření, například zveřejňování pozitivních recenzí pro přilákání nových zákazníků, a zasílání negativních recenzí zákaznické podpoře s cílem vyřešit vzniklé problémy. Oblíbenými algoritmy pro modelování témat jsou Latent Semantic Analysis a Latent Dirichlet Allocation.

3.4.1 Modely

Latent Semantic Analysis

Latent Semantic Analysis (LSA), někdy také Latent Semantic Indexing (LSI), je jednou z nejčastějších metod modelování témat. Tato metoda se opírá o tzv. distribuční sémantiku, která tvrdí, že slova nebo fráze, které se objevují ve stejných kontextech, mají tendenci mít podobný význam. LSA analyzuje frekvence slov v dokumentech a předpokládá, že dokumenty zabývající se podobnými tématy budou mít podobné distribuce frekvencí určitých slov. Nepřihlíží se přitom k pořadí slov, jejich významu ani sémantice. Pro výpočet frekvencí se často používají metody jako Bag of Words nebo TF-IDF, přičemž se častěji volí druhá varianta, neboť lépe zohledňuje důležitost slov

spolu s jejich četností (viz 3.6). Po výpočtu frekvencí se sestaví tzv. matice dokument-termín (DTM z angl. Document-term matrix), kde každý řádek reprezentuje jeden dokument a každý sloupec slovo (termín) vyskytující se v celém korpusu. Každá buňka této matice obsahuje frekvenci výskytu daného slova v daném dokumentu. Hlavním cílem LSA je zachytit základní sémantickou strukturu textu a reprezentovat ji v méně rozměrném prostoru. K tomu je matice rozložena na součin tří matic (U, S a V) pomocí singulárního rozkladu, kde matice U představuje matici dokument-téma, matice V matici termín-téma a matice S diagonální matici singulárních hodnot představujících potenciální témata nalezená v dokumentech. LSA vybírá t největších singulárních hodnot spolu s prvními t sloupci matice U a prvními t řádky matice V, čímž identifikuje t nejvýznamnějších témat v původní matici. t je hyperparametr určující počet témat, které algoritmus nalezl. Přestože tato metoda poskytuje cenné poznatky o sémantické struktuře textových dat, má některá omezení, například obtíže při zpracování polysémie (slova s více významy) a synonym (různá slova s podobným či stejným významem). [15]

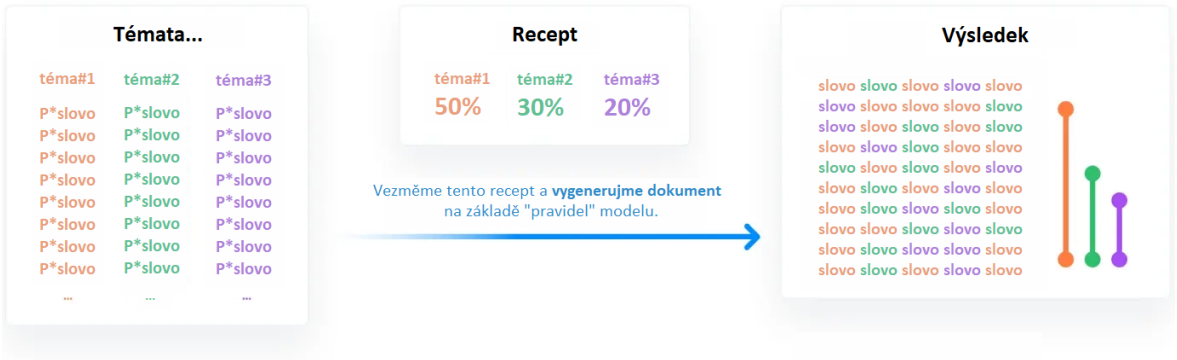


Obr. 1 Rozklad matice dokument-termín na matice dokument-téma a termín-téma. Zdroj: podle [15] upravil autor.

Latent Dirichlet Allocation

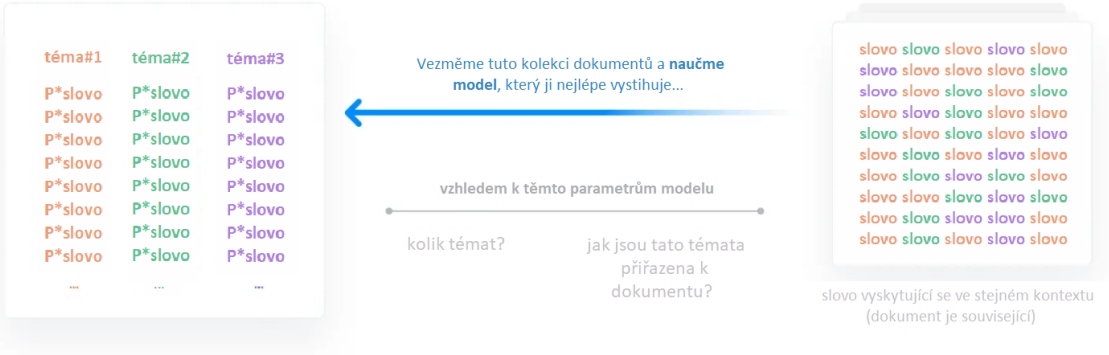
Latent Dirichlet Allocation (LDA) představuje rozšíření LSA modelu. Podobně jako LSA, i LDA je založena na distribuční sémantice. Navíc však předpokládá, že dokumenty jsou složeny ze směsi témat a že každé slovo má určitou pravděpodobnost příslušnosti k jednotlivým tématům. Účel LDA je přiřadit všechny známe dokumenty v korpusu

k neznámým tématům tak, aby slova z každého dokumentu byla z větší části zachycena těmito tématy. Také LDA ignoruje syntaktické informace a sémantiku textu. Pro své fungování LDA předpokládá, že dokumenty jsou strukturovány tak, že jsou vybrána určitá témata, a poté jsou vybrána slova, která do těchto témat spadají (viz Obr. 2).



Obr. 2 Tvorba dokumentu z pohledu algoritmu LDA. Zdroj: podle [15] upravil autor.

Na základě toho algoritmus zpětně iteruje přes každé slovo v každém dokumentu a zkouší pro každé slovo odhadnout nejvhodnější téma (viz Obr. 3).



Obr. 3 Proces modelování nového dokumentu algoritmem LDA. Zdroj: podle [15] upravil autor.

Pro trénování modelu existují dva hyperparametry, které ovlivňují podobnost dokumentů a témat:

- alfa — Určuje podobnost dokumentů. Čím vyšší hodnota alfa, tím více témat je přiřazeno každému dokumentu a naopak, čímž se snižuje nebo zvyšuje jejich podobnost.
- beta — Určuje podobnost témat. Čím vyšší hodnota beta, tím více slov bude patřit ke každému tématu a naopak, čímž se snižuje nebo zvyšuje jejich odlišnost.

Také je nutné specifikovat počet témat, které má model nalézt, a někdy také počet slov, která má pro každé téma extrahovat. Výstupem LDA je vektor čísel udávajících hodnotu míru pokrytí jednotlivých témat pro každý dokument. Porovnáním a analýzou těchto vektorů je možné získat přehled o tematických okruzích korpusu. [15]

3.4.2 Evaluace

Výstupem algoritmů pro modelování témat je soubor slov, která s největší pravděpodobností reprezentují dané téma. Z těchto slov lze vypočítat metriku nazývanou koherence, která slouží k celkovému vyhodnocení natrénovaného modelu. Existuje několik druhů koherence, ale varianta CV vykazuje nejvyšší korelaci s lidským hodnocením [19], a tato práce se zaměřuje především na tuto variantu. Výpočet koherence zahrnuje následující kroky:

1) Segmentace — Prvním krokem je segmentace, kdy jsou vytvořeny dvojice podmnožin slov pro hodnocení koherence tématu. Z nejdůležitějších slov daného tématu se vytvoří sada dvojic, kde druhá část dvojice potvrzuje první část. Pro variantu CV se používá metoda S-one-set, kde je míra potvrzení počítána nad jedním slovem a zbytkem seznamu.
2) Výpočet pravděpodobnosti — V tomto kroku se vypočítá pravděpodobnost na základě četnosti výskytu jednotlivých částí dvojic z tématu v textovém korpusu. Pro variantu CV se používá metoda klouzavého okna o velikosti 110, což znamená, že pravděpodobnosti se počítají nad klouzavým oknem o velikosti 110 sousedních slov, které se pohybuje napříč textem.
3) Míra potvrzení — V tomto kroku se vypočítá tzv. NPMI (angl. Normalized Pointwise Mutual Information) mezi slovy prvků každé dvojice s ostatními slovy ze seznamu nejdůležitějších slov daného tématu. Poté se vypočte podobnost mezi oběma těmito vektory měr pomocí kosinové podobnosti.
4) Agregace — V posledním kroku se vypočte aritmetický průměr konfirmačních měr, který představuje výsledné skóre koherence. V případě, že je více témat, je konečným výsledkem průměrná hodnota koherence jednotlivých témat. [19]

3.4.3 Nástroje

Existuje řada nástrojů pro implementaci modelů pro modelování témat, z nichž některé jsou dostupné pro programovací jazyk Python. Tato kapitola se zaměřuje především

na tyto nástroje. Byly vybrány nejčastěji používané a z pohledu autora nejintuitivnější nástroje, které budou aplikovány v praktické části této práce.

- Gensim je bezplatná open-source knihovna pro jazyk Python, která poskytuje širokou škálu funkcí souvisejících s tematickým modelováním. Umí efektivně zpracovávat nestrukturovaná textová data a automaticky odhalovat jejich sémantickou strukturu pomocí různých algoritmů. Kromě toho umožňuje výpočet různých evaluačních metrik pro identifikované vzory v textech. [18]
- Dále je možné použít knihovnu Scikit-learn, která kromě klasifikace textu nabízí také širokou škálu metod pro modelování témat. Obsahuje implementace algoritmů pro LDA i LSA, avšak nenabízí vestavěné nástroje pro výpočet evaluační koherence. [21]
- Tomotopy je pythonovské rozšíření nástroje Tomoto (Topic Modeling Tool), což je knihovna pro modelování témat napsaná v jazyce C++. Aktuální verze Tomoto podporuje několik tematických modelů, mezi něž patří také LDA, avšak nikoliv LSA či LSI. [12]

3.5 Předzpracování

Získaná textová data zřídka splňují požadavky pro okamžité využití v modelování či analýze. Je proto nutné je nejprve předpřipravit, než budou použita pro další zpracovávání. Tento proces zahrnuje identifikaci a následnou korekci chyb či nesrovnalostí, řešení problémů souvisejících se šumem, nekonzistencemi nebo nežádoucími prvky v textových datech, a jejich následnou transformaci a úpravu s cílem zlepšit jejich kvalitu a použitelnost pro analýzu, modelování či jiné potřeby uživatele.

Kroků, které lze v této fázi provést je velmi mnoho, avšak není potřeba je provádět všechny. Výběr konkrétních kroků je vždy podmíněn charakterem řešeného problému, a různé úlohy mohou vyžadovat odlišné postupy předzpracování.

3.5.1 Regulární výrazy

Nástrojem, který by měl být součástí pracovního postupu každého uživatele pracujícího s rozsáhlým objemem textových dat, jsou regulární výrazy. Regulární výraz, známý též jako regexp, regex nebo regexes, je posloupnost znaků definující specifický textový vzor, který je následně porovnáván s vybraným textem a umožňuje v něm vyhledávat a manipulovat s ním na základě konkrétních kritérií. Regulární výrazy umožňují extrahovat požadovaná data z textu, přetvářet je do potřebné podoby, rozdělovat je na základě daného vzoru nebo provádět jejich ověřování. S jejich znalostí je možné vytvořit efektivní nástroje, které při správném použití pomáhají uživatelům nalézt požadované informace a výrazně šetří jejich čas. Konstrukce regulárních výrazů zahrnuje název metody a kombinaci doslovných znaků (písmen, číslic a symbolů) spolu se speciálními znaky, které nesou unikátní význam v rámci daného vzoru (například tečka, která reprezentuje libovolný znak, nebo stříška, označující začátek řetězce atd.). [4] Regulární výrazy disponují širokou škálou již zmíněných metod, zde je uvedeno několik z nich:

- re.match(vzor, text) — Metoda vyhledá vzor pouze na začátku sekvence textu a pokud jej nalezne, vrátí jej.
- re.search(vzor, text) — Metoda vyhledá vzor v celé sekvenci textu a pokud jej nalezne, vrátí jeho první výskyt.
- re.findall(vzor, text) — Metoda prohledá celý text a vrátí všechny výskyty nalezeného vzoru.

- re.finditer(vzor, text) — Metoda funguje obdobně jako "re.findall", ale v případě nalezení vrátí iterátor obsahující pozice nalezeného vzoru.
- re.split(vzor, text) — Metoda vyhledá vzor v textu a vrací seznam prvků původního textu rozdělených podle zadaného vzoru.
- re.sub(vzor, substitut, text) — Metoda vyhledá vzor v textu a v případě nalezení jej nahradí zadaných substitutem, poté vrací upravený text.

Regulární výrazy zahrnují rozsáhlou paletu dalších speciálních znaků, předdefinovaných skupin znaků, kvantifikátorů a logických spojek. Díky nim je možné provádět širokou škálu operací nad původním textem, jako je například identifikace a úprava nadpisů, emailových adres, hypertextových odkazů a dalších typů údajů. Regulární výrazy jsou často implementovány ve formě knihoven a balíčků pro jednotlivé programovací jazyky (například pro Python je zde modul "re").

3.5.2 Tokenizace

I přes provedené očištění textu, jeho podoba nestrukturovaného textového řetězce nenesoucího žádné informace je pro stroj stále nepochopitelná. Je proto nutné převést tento text do lépe zpracovatelného formátu. K tomu slouží proces zvaný tokenizace. Jedná se o proces, během kterého je text v přirozeném jazyce rozdělen na menší jednotky, označované jako tokeny, které lze považovat za již samostatné prvky. Rozdělení textu může probíhat na úrovni vět, slov nebo znaků. Při rozdělení na úrovni vět lze využít více přístupů, například rozdělení pomocí interpunkčních znamének. Avšak v některých případech, kdy se v textu vyskytují zkratky nebo jiné specifické znaky ukončené tečkami, může být tato metoda nevhodná. Alternativně lze využít knihoven a nástrojů s předtrénovanými modely, které by měly tyto nedostatky eliminovat. [9] Rozdělení na úrovni slov nebo znaků představuje méně komplikovanou záležitost a je možné ho provést v "čistém" Pythonu. Pro anglické texty existuje několik knihoven a nástrojů, které je možné využít pro tokenizaci, jako jsou NLTK, spaCy, Tokenizers a další. Některé z předchozích uvedených knihoven podporují také češtinu, a je tedy možné je využít také pro tokenizaci česky psaných textů. Pro české texty jsou ovšem k dispozici přímo české nástroje jako UDPipe nebo morphoDiTa, které jsou lépe přizpůsobeny specifikám českého jazyka. Na Obr. 4 lze vidět porovnání z výstupu tokenizace vět mezi knihovnou NLTK a českým nástrojem UDPipe v Pythonu. Z něho je patrné, že knihovna NLTK nedokázala rozeznat zkratku "angl." jako označení pro

angličtinu a nesprávně jej identifikovala jako ukončení věty. Narozdíl od nástroje UDPipe, který zkratku rozpoznal a ze vstupu správně vytvořil dva tokeny (věty).

```
Vstup:
Tento text slouží k ukázce tokenizace (angl. tokenization). A toto by měla být druhá věta!

NLTK:
1. věta: Tento text slouží k ukázce tokenizace (angl.
2. věta: tokenization).
3. věta: A toto by měla být druhá věta!

UDPipe:
['# sent_id = 1', '# text = Tento text slouží k ukázce tokenizace (angl. tokenization).']
['# sent_id = 2', '# text = A toto by měla být druhá věta!']
```

Obr. 4 Porovnání tokenizace vět mezi NLTK a UDPipe. Zdroj: Autor

3.5.3 Eliminace stop slov

V každém textu se běžně vyskytují slova, která nemají sémantickou hodnotu a nepřispívají k celkovému významu. Tato slova se označují jako "stop slova" a zahrnují neplnohodnotné výrazy, jako jsou předložky, spojky, částice, zájmena a další. [9] V češtině mezi ně mohou patřit například slova "by", "se", "a", "toho", "z", "to", "i", "když" a mnohé další. Je zcela evidentní, že například v jakémsi pomyslném článku o nové AI technologii budou slova jako "AI", "technologie" a "stroj" více přispívat k významu textu než výše zmíněna stop slova. Jejich odstraněním zůstanou v textu pouze relevantní slova. Současně se tím sníží náročnost programu, neboť čas potřebný pro zpracování celého textu je díky redukci jeho množství kratší. Implementace odstranění stop slov není složitá, postačuje mít seznam stop slov daného jazyka a odstranit všechny výskyty těchto slov v textu. Avšak hlavním problémem je neexistence univerzálního seznamu stop slov. Nabízí se několik řešení:

- Vytvoření vlastního seznamu — Toto řešení může být pracné a méně přesné, v závislosti na pečlivosti tvůrce seznamu. V seznamu mohou některá slova chybět a jeho samotné vytváření může zabrat zbytečně příliš mnoho času (ačkoliv je v dnešní době možné využít pomoc generativní umělé inteligence).
- Využití knihoven — Existují volně dostupné knihovny, které obsahují již vytvořené seznamy stop slov z různých jazyků včetně češtiny (např. knihovna "stopwords-iso").
- Využití externích nástrojů — Lze použít již vytvořené programy, které mají v sobě seznam již zabudovaný a umí stop slova v zadaném textu identifikovat

a případně odstranit. Pro česky psané texty je to například nástroj MTA software (konkrétně aplikace "Stop slova") od NLP Mendelu Group.

Eliminaci stop slov není nutné provádět vždy, ale je klíčová pro úlohy, jako je klasifikace textu, vyhledávání informací nebo tematická analýza, kde pomáhá eliminovat šum a zaměřit se na klíčové výrazy vyjadřující hlavní myšlenky v textu. V některých případech, například u strojového překladu, by však odstranění stop slov mohlo výrazně snížit přesnost programu.

3.5.4 Stemování a Lemmatizace

Mnoho jazyků, včetně češtiny, má slova v různých tvarech, přestože mají stejný význam. Často se stává, že je nad textem zavolán dotaz pro nalezení určitého slova, jako například "pes", ale vyskytuje se v něm pouze v jiných tvarech, jako jsou "psích", "psa" a podobně. V takovém případě by výsledek dotazu signalizoval, že dané slovo v textu není přítomno, což však úplně neodpovídá skutečnosti. Aby se tento nedostatek vyřešil, provádí se nad textovým souborem operace, které převádějí slova na jejich základní formu. Z podstatných jmen se získává nominativ jednotného čísla, ze sloves infinitiv a podobně. Předpony a přípony slov jsou také odstraněny. Slova upravená tímto způsobem jsou pro analýzu jejich významu vhodnější a usnadňují manipulaci s nimi. Pro tyto procesy existují dva různé přístupy, a to stemování a lemmatizace. [9]
Stemování (angl. stemming) je proces identifikace kořene slova odstraněním předpon a morfologických koncovek. Hlavním rozdílem mezi stemováním a lemmatizací je, že stemovací algoritmy nemusí vracet skutečná slova s významem. Tyto algoritmy mohou slovo pouze zkrátit na tzv. stemy. Přestože se požaduje, aby různé tvary slova se stejným významem byly zkráceny na stejný stem a aby žádná rozdílná slova neměla stejný stem, mohou se vyskytnout chyby, kdy tato pravidla nejsou dodržena. Porušení prvního pravidla, kdy různé tvary slova se stejným významem nejsou zkráceny na jeden stejný stem, se nazývá under-stemming. [9] To může nastat, když je odebrána příliš malá část slova. Například slova "velikými" a "velikou" by měla mít stejný stem "velik", ale při under-stemmingu by mohly mít stemy podobu "veliky" a "veliko". Porušení druhého pravidla, kdy slova s různým významem jsou zkrácena na stejný stem, se nazývá over-stemming. [9] To může nastat, když je odebrána příliš velká část slova. Například slova "velké" a "velitel" by správně měla mít různé stemy, ale při over-stemmingu by oba jejich stemy mohly mít podobu "vel". Stemování je sice rychlejší než lemmatizace, ale méně přesné. Pro stemování anglických textů je možné využít

například knihovnu NLTK, která disponuje stemovacími algoritmy Porter stemmer či Snowball stemmer. Pro české texty je možné použít například Czech stemmer od Luís Gomes.

Lemmatizace (angl. lemmatization) je proces podobný stemování, který také odstraňuje předpony a morfologické koncovky slov. Na rozdíl od stemování však lemmatizace vyžaduje slovník, pomocí kterého porovnává zkrácené slovo s odpovídajícím kořenem slova s příslušným významem. Tento proces je sice náročnější než stemování, avšak z hlediska zachování sémantiky slov je přesnější, neboť výsledné lemma nese skutečný význam. [9] Pro lemmatizaci anglických textů v jazyce Python je možné využít knihovny NLTK nebo spaCy. Pro zpracování českých textů mohou být užitečné knihovny Stanza či simplemma, které podporují češtinu, nebo alternativní nástroje jako UDPipe a MorphoDiTa od Milana Straky a Jany Strakové, či Lemmatizer od NLP Mendelu Group, které lze využít bez znalosti programování.

3.5.5 Normalizace

Normalizace, někdy označovaná též jako standardizace, představuje klíčový krok obsahující několik dílčích podprocesů zaměřených na transformaci textu do jednotné podoby. Stejně jako u jiných technik předzpracování textu je důležité si uvědomit, že volba konkrétních operací závisí na charakteru a požadavcích řešeného problému, původu zdrojových dat a jejich obsahu. [20] Některé z těchto operací lze také provést prostřednictvím regulárních výrazů (viz 3.5.1). Následující seznam představuje několik klíčových podprocesů normalizace textu:

- Převod na velká nebo malá písmena — V mnoha textech lze opomenout použití velkých písmen tam, kde to není nezbytné (mimo vlastní jména a podobně). Obdobně je vhodné v některých případech převést malá písmena na velká (například u zkratek, jako n.y. na N.Y.). Oba procesy v sobě má většina programovacích jazyků již implementované. V Pythonu jsou k dispozici metody ".lower()" pro konverzi na malá písmena a ".upper()" pro konverzi na velká písmena.
- Odstranění interpunkčních znamének — Během tohoto procesu dochází k odstranění interpunkčních znamének, neboť nenesou žádnou sémantickou informaci (tečka, vykřičník, otazník, závorky a další), například u akronymů.

- Odstranění prázdných znaků ("\n", "\t" atd.) — V textu, který je následně podroben analýze nebo jiným operacím, jsou v některých případech zalomené řádky nepotřebné, a proto je vhodné je odstranit.
- Odstranění speciálních symbolů (např. smajlíků) — V případě, že zdrojem dat je textová zpráva, chat ze sociální sítě a podobně, mohou se v něm vyskytovat smajlíky či jiné speciální symboly, které stroj nemůže interpretovat tak, aby v nich našel význam. Je proto vhodné tyto znaky odstranit (pokud nejsou klíčové pro řešený problém).
- Sjednocení tvarů čísel a datumů — V případě, že se v textu objevují číslice a zároveň čísla vyjádřená slovy, je žádoucí převést jednu z těchto variant na druhou, aby byla zachována jednotná podoba v celém textu. Tuto operaci je také vhodné aplikovat na formátování datumů.
- Převod zkratek na plné znění — Některé lemmatizační nebo stemovací programy nemusí být schopny rozpoznat zkratky uvedené v textu. Tento problém lze řešit převedením zkratek na jejich plné znění.

3.6 Vektorová reprezentace

Jakmile jsou data podrobena procesu předzpracování, lze přistoupit k následující fázi, a to vektorizaci. Tento krok představuje proces převodu textových dokumentů, jako jsou věty nebo odstavce, do formátu numerického vektoru, který umožňuje efektivní zpracování a analýzu strojem, neboť algoritmy strojového učení pracují v číselném prostoru. [9] Tato matematická reprezentace nejen usnadňuje manipulaci s daty, ale také snižuje výpočetní náročnost programu. Modely jsou založené na mapování jednotlivých textových jednotek na vektorové reprezentace, které jsou obvykle vyjádřeny souborem reálných čísel v n-dimenzionálním prostoru. Velikost dimenze závisí na konkrétním modelu a určuje, kolik detailů mezi slovy vektor zachycuje. Vektor slova je získáván prostřednictvím učení z rozsáhlého spektra textů, přičemž je kladen důraz na výskyt slova v kontextu s ostatními. Slova s podobným významem mají také podobnou vektorovou reprezentaci. Existuje několik odlišných přístupů, jejichž největší rozdíl tkví v podobě a kvalitě zachycení lingvistických rysů textové jednotky ve výsledném vektoru. [22] Tab. 2 představuje fiktivní pracovní korpus, který bude využit pro ilustraci těchto přístupů.

Tab. 2 Pracovní korpus. Zdroj: Autor

Dokument ID	Text
D1	Kočka není pes.
D2	Pes není kočka.
D3	Pes je doma.
D4	Kočka je venku.

3.6.1 Kódování 1 z N

Jednoduchou metodou vektorizace je tzv. Kódování 1 z N (angl. One-Hot encoding), která převádí data na binární vektory. Každý vektor reprezentující jedno slovo se skládá z nul a jedniček, zaznamenávajících pouze přítomnost (1) nebo nepřítomnost (0) daného slova ze slovníku v daném textu. Tato metoda vytváří unikátní vektor pro každé slovo, avšak s sebou nese vyšší výpočetní náročnost. Velikost každého vektoru koresponduje s velikostí slovníku. [23] Přestože je tato metoda snadno implementovatelná a nabízí intuitivní přístup, může se vyskytnout nevýhoda v podobě "Out of vocabulary". Tento problém nastává, když je modelu předložen text obsahující slovo, které není zahrnuto v korpusu, na němž byl model trénován. V takovém případě je nezbytné rozšířit slovník o dané slovo a znovu natrénovat model. Tato metoda se používá především tam, kde je důležité zachovat informaci o existenci nebo neexistenci určité kategorie, ale není důležité zachovat pořadí, frekvenci nebo vzdálenost mezi kategoriemi.

Z korpusu je vytvořen soubor jeho jedinečných slov – slovník ve formě vektoru o šesti dimenzích:

$$S = [kočka, není, pes, je, doma, venku]$$

Vektorová reprezentace provedená metodou Kódování 1 z N na pracovním korpusu má podobu:

$$D1 = [[1\ 0\ 0\ 0\ 0\ 0], [0\ 1\ 0\ 0\ 0\ 0], [0\ 0\ 1\ 0\ 0\ 0]]$$
$$D2 = [[0\ 0\ 1\ 0\ 0\ 0], [0\ 1\ 0\ 0\ 0\ 0], [1\ 0\ 0\ 0\ 0\ 0]]$$
$$D3 = [[0\ 0\ 1\ 0\ 0\ 0], [0\ 0\ 0\ 1\ 0\ 0], [0\ 0\ 0\ 0\ 1\ 0]]$$
$$D4 = [[1\ 0\ 0\ 0\ 0\ 0], [0\ 0\ 0\ 1\ 0\ 0], [0\ 0\ 0\ 0\ 0\ 1]]$$

Každý řádek reprezentuje jednu větu, a jednotlivé seznamy oddělené čárkami představují vektory pro jednotlivá slova. Prochází se každé slovo ve větě a porovnává se s obsahem slovníku. Pokud slovo odpovídá některému záznamu ve slovníku, je na

odpovídajícím indexu vytvořeného vektoru nastavena hodnota jedna; v opačném případě je nastavena hodnota nula.

3.6.2 Bag of Words

Další metodou je tzv. Batoh slov (angl. Bag of Words), který reprezentuje dokument jako neuspořádanou množinu čísel, přičemž ignoruje jejich pořadí a zaměřuje se na frekvenci slov. Podobně jako předchozí metoda One-Hot Encoding, i Bag of Words je intuitivní a snadno implementovatelný. Ignoruje vztahy mezi slovy a stále se potýká s problémem "Out-of-vocabulary". Na rozdíl od One-Hot Encoding se však neomezuje pouze na binární reprezentaci přítomnosti nebo nepřítomnosti slova. Místo toho zaznamenává i četnost jednotlivých výskytů daného slova v textu. Výsledný vektor zachycuje slova v pořadí, v jakém se nacházejí ve slovníku, a jejich četnosti v jednotlivých větách. [22] Vzhledem k omezením této metody může nastat situace, kdy dvě věty s totožným počtem stejných slov, ale v různém pořadí, budou mít podobnou nebo stejnou vektorovou reprezentaci (viz následující příklad).

Podobně jako v předchozím případě je z korpusu vytvořen soubor jeho jedinečných slov – slovník ve formě vektoru o šesti dimenzích:

$$S = [kočka, není, pes, je, doma, venku]$$

Vektorová reprezentace provedená metodou Bag of words na pracovním korpusu bude mít podobu:

$$D1 = [1\ 1\ 1\ 0\ 0\ 0]$$
$$D2 = [1\ 1\ 1\ 0\ 0\ 0]$$
$$D3 = [0\ 1\ 1\ 1\ 1\ 0]$$
$$D4 = [1\ 0\ 1\ 1\ 0\ 1]$$

Každý řádek reprezentuje jednu větu, a jednotlivé číselné hodnoty odpovídají frekvenci výskytu jednotlivých slov ze slovníku ve větě. Pro každou větu se postupně prochází slovníkem; pokud se dané slovo ve větě vyskytuje, zaznamená se číslo označující počet jeho výskytů.

3.6.3 Množina n-gramů

Batoh N-gramů (angl. Bag of N-grams) představuje rozšíření modelu Bag of Words (BoW). Zatímco BoW uvažuje text jako množinu čísel reprezentujících jednotlivá slova a jejich frekvenci, Bag of N-grams jej rozšiřuje o reprezentaci pořadí slov v dokumentu. Tento model text rozděluje do sekvencí n-sousedících slov, symbolů nebo tokenů,

známých jako n-gramy. V případě, že hodnota "n" je větší než 1, model je schopný zachytit kontextovou informaci obsaženou v textu a reprezentovat pořadí slov ve vektoru. Model tak dokáže identifikovat slova, která se vedle sebe vyskytují častěji než slova jiná, a tím i lépe identifikovat sousloví, například "Český dub" oproti "Český" a "Dub". Čím vyšší je hodnota "n", tím širší kontext slov bude reprezentován. Je třeba však vzít v úvahu, že s rostoucí hodnotou "n" může docházet k výraznému zvýšení dimenzionality prostoru příznaků, což může ovlivnit výpočetní složitost. Běžnými volbami pro "n" jsou 1 (unigramy), 2 (bigramy) a 3 (trigramy). Velikost vektoru v tomto modelu odpovídá počtu n-gramů. Tímto způsobem je možné také identifikovat téma dokumentu, neboť častěji vyskytující se n-gramy lépe vystihují jeho obsah než ty, které se v něm vyskytují méně. Zároveň takto umožňuje nalézt sémantickou podobnost mezi dokumenty, protože dokumenty pojednávající o podobném tématu budou mít podobné n-gramy. [6] Další využití je možné nalézt například v automatickém doplňování či našeptávání slov při psaní. Volba konkrétní hodnoty "n" závisí na konkrétní úloze v oblasti zpracování přirozeného jazyka (z anglického Nature Language Processing, NLP) a vlastnostech analyzovaných dat. Uživatel by měl na svých datech vyzkoušet různé n-gramy, aby určil optimální hodnotu pro konkrétní úkol a data.

V ukázce je zvolena hodnota n = 2 (bigramy). Z pracovního korpusu je tentokrát namísto slovníku vytvořen bigram model o 8 dimenzích:

$$[kočka\ není, není\ pes, pes\ není, není\ kočka, pes\ je, je\ doma, kočka\ je, je\ venku]$$

Vektorová reprezentace provedená metodou Bag of N-grams na pracovním korpusu bude mít podobu:

$$D1 = [1\ 1\ 0\ 0\ 0\ 0\ 0\ 0]$$
$$D2 = [0\ 0\ 1\ 1\ 0\ 0\ 0\ 0]$$
$$D3 = [0\ 0\ 0\ 0\ 1\ 1\ 0\ 0]$$
$$D4 = [0\ 0\ 0\ 0\ 0\ 0\ 1\ 1]$$

Podobně jako v případě modelu BoW, každý řádek reprezentuje jednu větu, přičemž jednotlivé číselné hodnoty odpovídají frekvenci výskytů jednotlivých bigramů v dané větě. Pro každou větu je postupně procházen bigramový model a pokud se daný bigram ve větě vyskytuje, je do vektoru zaznamenán počet jeho výskytů.

3.6.4 TF-IDF

Metoda Bag of Words (BoW) zaznamenává frekvenci výskytů slov v dokumentu, což zvýhodňuje slova, která se v něm často opakují. Taková slova mohou být ovšem pro význam dokumentu méně důležitá než ta, která se vyskytují ojediněle (slova "a", "je" a podobně), a je tedy vhodné jim přiřazovat menší váhu. [22] Metoda TF-IDF (angl. Term Frequency — Inverse Document Frequency) slouží ke kvantifikaci důležitosti slova vzhledem k celému korpusu dokumentů. [9] Základním principem je, že pokud se slovo vyskytuje často v jednom dokumentu korpusu a zároveň se tak často nevyskytuje v ostatních dokumentech, má toto slovo pro daný dokument větší důležitost. Dále se očekává, že důležitost daného slova vzroste s četností jeho výskytů v daném dokumentu, ale klesne se zvýšením výskytů v ostatních dokumentech. Vyšší hodnota TF-IDF naznačuje, že slovo má významnou roli v daném dokumentu ve srovnání s ostatními dokumenty v korpusu. Tuto hodnota lze vyjádřit kombinací hodnot TF a IDF podle následující rovnice:

$$TFIDF = TF * IDF$$

TF (angl. Term Frequency) měří, jak často se určité slovo vyskytuje v celém dokumentu. Čím častěji se slovo v dokumentu objevuje, tím vyšší bude jeho TF hodnota. [9] Tato hodnota může být vyjádřena jako přímý součet výskytů daného slova v dokumentu, avšak u delšího textu je pravděpodobnost výskytu daného slova vyšší než u kratšího textu, proto je vhodnější tuto hodnotu normalizovat vydělením celkovým počtem všech slov v dokumentu [22]:

$$TF = \frac{počet\ výskytů\ daného\ slova\ v\ určitém\ dokumentu}{celkový\ počet\ všech\ slov\ v\ určitém\ dokumentu}$$

Stop slova jsou důkazem toho, že často se vyskytující slova nemusí být nezbytně klíčová pro význam dokumentu. Čím častěji se tedy slovo v dokumentech vyskytuje, tím je považováno za méně důležité. IDF (angl. Inverse Document Frequency) měří, jak vzácné nebo unikátní je dané slovo napříč všemi dokumenty v korpusu. [9] Čím vzácněji se slovo objevuje ve více dokumentech, tím vyšší je jeho IDF hodnota. Její výpočet lze provést podle následujícího vzorce [22]:

$$IDF = \log \frac{celkový\ počet\ dokumentů\ v\ korpusu}{počet\ dokumentů\ v\ korpusu\ obsahujících\ dané\ slovo}$$

Pro ovlivnění výsledných TF-IDF hodnot je možné zahrnout parametry maximální a minimální frekvence výskytu slova. Tím se stanoví horní a spodní hranice pro to, kolikrát se dané slovo musí nebo nesmí vyskytovat mezi všemi dokumenty. V případě,

že je toto předem definované omezení překročeno (horní hranice) nebo nesplněno (spodní hranice), je dané slovo považováno za nerelevantní a je vynecháno z výstupu. Je nezbytné pečlivě volit hodnoty těchto parametrů, aby nedošlo k odstranění důležitých slov a tím ke snížení účinnosti metody. I když definování těchto parametrů není nutné, jejich implementace přispívá ke snížení paměťové náročnosti celého procesu. [8]

Při aplikaci na pracovním korpusu je opět nejprve vytvořen slovník ve formě vektoru o šesti dimenzích:

$$S = [kočka, není, pes, je, doma, venku]$$

Při vytváření vektorové reprezentace každého z dokumentů z korpusu jsou pro každé slovo vypočítány hodnoty TF, IDF a výsledné TF-IDF. Ukázku výpočtů pro dokument D1 z pracovního korpusu lze vidět v Tab. 3.

Tab. 3 Ukázka výpočtu metodou TF-IDF. Zdroj: Autor

Slovo ze slovníku	Počet (D1)	TF (D1)	IDF	TFIDF (D1)
kočka	1	1/3 = 0,33	log(4/3) = 0,12	0,33 * 0,12 = 0,04
není	1	1/3 = 0,33	log(4/2) = 0,3	0,33 * 0,3 = 0,10
pes	1	1/3 = 0,33	log(4/3) = 0,12	0,33 * 0,12 = 0,04
je	0	0/3 = 0	log(4/2) = 0,3	0 * 0,3 = 0
doma	0	0/3 = 0	log(4/1) = 0,6	0 * 0,6 = 0
venku	0	0/3 = 0	log(4/1) = 0,6	0 * 0,6 = 0

Vektorová reprezentace celého pracovního korpusu provedená metodou TF-IDF bude mít podobu:

$$D1 = [0{,}04\ 0{,}1\ 0{,}04\ 0\ 0\ 0]$$
$$D2 = [0{,}04\ 0{,}1\ 0{,}04\ 0\ 0\ 0]$$
$$D3 = [0\ 0\ 0{,}04\ 0{,}1\ 0{,}2\ 0]$$
$$D4 = [0{,}04\ 0\ 0\ 0{,}1\ 0\ 0{,}2]$$

4 Metody

V následujících kapitolách budou popsány postupy a způsoby řešení pro aplikování jednotlivých příslušných nástrojů pro předzpracování, vektorizaci, klasifikaci a modelování, zmíněné v předchozích kapitolách, na datovou sadu (v případě klasifikace) a korpus (v případě modelování) s českými texty v programovacím jazyce Python.

4.1 Klasifikace

V následujících kapitolách budou detailně popsány postupy a metody, které byly použity při aplikaci nástrojů pro klasifikaci témat, jak byly představeny v teoretickém úvodu, na reálných českých textech.

4.1.1 Data

Pro aplikaci nástrojů pro předzpracování textu a klasifikaci témat bude využita datová sada ČSFD[1] obsahující přes 91 tisíc filmových recenzí od uživatelů z webového portálu Česko-Slovenské filmové databáze. [5] Obsahuje recenze, které lze kategorizovat do tří různých témat: pozitivní, neutrální a negativní. Množství recenzí přiřazených k daným tématům je rozloženo přibližně rovnoměrně, přičemž každá recenze může patřit pouze do jedné z těchto kategorií. Jedná se tedy o problematiku multi-class klasifikace. Datová sada se skládá ze tří samostatných textových souborů, z nichž každý obsahuje recenze zaměřené na jednotlivá témata. Náhled datové sady lze vidět na Obr. 5.

[1] https://corpora.kiv.zcu.cz/sentiment

```
         label                                                text
2509     neutral  Tuctová americká komedie, kde je víc klišé než...
9237    positive  Šustivé róby, večeře plné lesku, pikniky v Hyd...
14987    neutral  Z čiré nostalgie dob dětství. Teď jsem pár epi...
3431    positive  Žena u detektoru lži: ,,Provedeme několik test...
27819    neutral  ...Ach jo. Tenhle anděl je falešný jak panák n...
...          ...                                                ...
6265    positive  S U P E R představení, dokonalé obsazení a gen...
23989   negative  Samoúčelná brutalita - zabíjení zvířat, znásil...
16207    neutral  Další z řady béčkových krimithrillerů se zamot...
860     positive  Takovou skvělou řezbu jsem nečekal. Pro mě jed...
15795   positive  Tenhle film je hned od začátku bomba. Al Pacin...

[91381 rows x 2 columns]
```

Obr. 5 Ukázka datové sady ČSFD. Zdroj: Autor

4.1.2 Předzpracování

Tyto kroky předzpracování budou aplikovány na všechny české texty napříč popsanými nástroji.

Prvním krokem předzpracování textu budou regulární výrazy, které budou aplikovány pomocí metody "regexPreprocessing", jejíž implementaci lze vidět na Obr. 6.

```python
def regexPreprocessing(df_column):
    # Odstranění emailových dadres
    df_column = df_column.str.replace(r'^.+@[^\.].*\.[a-z]{2,}$', '', regex=True)
    # Odstranění URLs
    df_column = df_column.str.replace(r'^https?://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$', '', regex=True)
    # Odstranění telefonních čísel
    df_column = df_column.str.replace(r'(\+420\s?\d{3}\s?\d{3}\s?\d{3}|\b\d{3}\s?\d{3}\s?\d{3}\b)', '', regex=True)
    # Odstranění více mezer mezi slov a nahrazení jednou mezerou
    df_column = df_column.str.replace(r'\s+', ' ', regex=True)
    # Odstranění interpunkčních znamének
    df_column = df_column.str.replace(r'[!\"#\$%&\'\(\)\*\+,-\./:;<=>\?@\[\\\]\^_`{\|}~]', "", regex=True)

    return df_column
```

Obr. 6 Metoda pro předzpracování regulárními výrazy. Zdroj: Autor

Tato metoda přijímá jako vstupní argument "df_column", což je sloupec dataframe obsahující texty recenzí. Na text bude postupně aplikováno 5 regulárních výrazů:

1) Odstranění emailových adres, jehož rozbor je:
   - "^" označuje začátek řetězce,
   - ".+@" označuje minimálně jeden jakýkoliv znak, po kterém následuje zavináč,
   - "[^\.]" označuje jakýkoliv znak kromě tečky,

- ".*" označuje jakýkoliv (i prázdný) řetězec,
- "\." označuje tečku,
- "[a-z]{2,}" označuje jakékoliv alespoň dvě malá písmena,
- "$" označuje konec řetězce.

2) Odstranění URL odkazů, jehož rozbor je:
   - "^" označuje začátek řetězce,
   - "https?://" označuje výskyt "http://" nebo "https://",
   - "[a-zA-Z0-9\-\.]+" označuje minimálně jeden výskyt jakéhokoliv velkého nebo malého písmene, čísla, pomlčky či tečky,
   - "\." označuje tečku,
   - "[a-zA-Z]{2,3}" označuje minimálně dva, ale maximálně tři výskyty jakéhokoliv velkého nebo malého písmene,
   - "(/\S*)?$" označuje nepovinný výskyt lomítka následovaného libovolnými neprázdnými znaky.
3) Odstranění telefonních čísel obsahujících a neobsahujících českou předvolbu, jehož rozbor je:
   - "\+420" označuje výskyt české telefonní předvolby,
   - "\s?" označuje nepovinný výskyt mezery,
   - "\d{3}" označuje tři libovolná čísla,
   - tento vzor se opakuje až do znaku "|" označující konec první možnosti shody a začátek druhé, který je označen znakem "\b".
4) Odstranění sekvencí mezer, tabulátorů a podobně, a jejich nahrazení jednou mezerou. Výraz obsahuje pouze "\s+" označující výskyt minimálně jednoho bílého znaku, a následně "' '" představující nahrazovaný vzorec.
5) Odstranění interpunkčních znamének. Celý tento výraz obsahuje jen soubor znaků, které se z textu odstraní. Některé z nich jsou obklopeny zpětnými lomítky, které zde představují tzv. escape znak, což znamená, že následující znak je chápan doslovně.

Metoda na konci vrací upravený dataframe.

Dalším krokem, který je identický u všech popsaných nástrojů, je lemmatizace. Knihovny NLTK, Scikit-learn ani Keras nenabízejí žádnou metodu pro lemmatizaci českých textů. Pro tyto účely bude vytvořena metoda "simplemmatizace",

která využívá knihovnu simplemma, jež dokáže provést lemmatizaci českých textů. Implementace metody je znázorněna na Obr. 7.

```
def simplemmatizace(df):
    df_list = df.tolist()
    lemmaSent = []

    for sentence in df_list:
        lemmaWords = []
        tokens = sentence.split()

        for token in tokens:
            lemmaWords.append(simplemma.lemmatize(token, lang='cs'))
        lemmaSent.append(' '.join(lemmaWords))

    lemma_df = pd.DataFrame(lemmaSent)

    return lemma_df[0]
```

Obr. 7 Metoda pro lematizaci českých textů pomocí knihovny simplemma. Zdroj: Autor

Tato metoda přijímá vstupní parametr "df", který představuje sloupec z dataframe obsahující texty recenzí. Nejprve je dataframe převeden na seznam a následně jsou postupně procházeny jednotlivé věty a jejich slova. Pro každé z těchto slov je volána lemmatizační funkce a výsledné lemma je přidáno do seznamu "lemmawords", který reprezentuje lemmatizovaná slova jedné věty. Na konci věty jsou tato slova spojena zpět do věty, která je následně přidána do seznamu "lemmaSent". Po dokončení procházení všech vět je seznam opět převeden na dataframe, který je pak ve formě sloupce vrácen jako výstup.

Všechny tyto kroky budou aplikovány na jednotlivé soubory z datasetu zvlášť, neboť v rámci lemmatizace se prohazuje pořadí řádků a data by se stala nekonzistentními.

Další kroky předzpracování, jako je odstranění stop slov, převod na malá písmena, a vektorizace jsou prováděny v každém z nástrojů odlišně. To je způsobeno tím, že některé nástroje mají vlastní implementaci těchto procesů, zatímco pro jiné bylo nutné vytvořit metodu z externích knihoven.

4.1.3 NLTK

NLTK neposkytuje vlastní metodu pro odstranění stop-slov v češtině. K tomuto účelu bude využit seznam českých stop slov "Stopwords Czech" z kolekce "Stopwords ISO"

obsahující několik vícejazyčných stop slov seznamů. Jednotlivá slova z tohoto seznamu budou porovnávána s textem recenzí a v případě, že se v něm nějaké z nich vyskytne, budou z textu odstraněna.

Během extrakce příznaků jsou nejprve popisná témata převedena na číselné hodnoty pomocí třídy "LabelEncoder" z knihovny Scikit-learn. Před samotným převedením textů recenzí je vytvořen seznam nejčastěji se vyskytujících slov v textu, který je následně porovnáván se samotnými texty. Výsledný seznam příznaků je utvořen na základě existence nebo neexistence daných slov mezi těmi nejčastěji se vyskytujícími slovy, což přispívá k větší přesnosti klasifikátoru. K tomuto účelu jsou využity dvě metody.

První metodou je "extractWordFeatures", jejíž implementace je zobrazena na obrázku.

```
def extractWordFeatures(preprocessed_texts, num_features):
    all_words = []
    for text in preprocessed_texts:
        words = word_tokenize(text, language='czech')  # Tokenizace
        all_words.extend(words)

    # Počet výskytů
    word_freq_dist = nltk.FreqDist(all_words)

    # Nejčastější slova
    word_features = []
    for word, _ in word_freq_dist.most_common(num_features):
        word_features.append(word)

    return word_features
```

Obr. 8 Metoda pro extrakci nejfrekventovanějších slov. Zdroj: Autor

Tato metoda slouží k extrakci nejčastěji se vyskytujících slov z textu recenzí a má dva vstupní parametry. Prvním z nich je již předzpracovaný text recenzí a druhým je číselná hodnota určující počet výsledných nejčastějších slov. Uvnitř metody je procházena každá jednotlivá recenze a provádí se tokenizace pomocí funkce "word_tokenize(), která rozděluje text na jednotlivá slova. Všechna tato slova jsou přidána do seznamu "all_words". Po dokončení procházení všech textů je spočítána frekvence výskytu jednotlivých slov pomocí třídy "nltk.FreqDist()". Nakonec jsou vybrána nejčastěji se vyskytující slova metodou "most_common". Tato metoda vrací seznam dvojic

obsahující slovo a jeho frekvenci, ale funkce extrahuje pouze slova. Seznam těchto vybraných slov je vrácen jako výstup metody.

Druhá metoda, nazvaná "createFeatureSet", slouží k vytvoření sady příznaků pro každý text z datové sady na základě výstupního seznamu předchozí metody. Implementace této metody je zobrazena na obrázku.

```
def createFeatureSet(data, word_features):
    feature_set = []
    for text, label in data:
        words = word_tokenize(text, language='czech')  # Tokenizace
        features = {}
        for word in word_features:
            if word in words:
                features[word] = True
            else:
                features[word] = False
        feature_set.append((features, label))

    return feature_set
```

Obr. 9 Metoda pro vektorizaci. Zdroj: Autor

Metoda přijímá dva vstupní parametry. Prvním z nich je seznam dvojic obsahující předzpracované texty recenzí a jejich témata, druhým je výstupní seznam z předchozí metody. Pro každý text recenze se ověřuje, zda se jeho slova (tokeny) nacházejí v seznamu nejčastěji se vyskytujících slov. Pokud se slovo v seznamu nachází, je mu přiřazena hodnota "True", jinak je přiřazena hodnota "False". Slova a jejich hodnoty jsou poté uloženy do výsledného seznamu, který metoda vrací. Tento seznam je následně využit pro trénování a testování modelu. I když výstupem funkce nejsou pouze čísla, proces stále zahrnuje převod textových dat na číselnou reprezentaci, což je podstatou vektorizace.

Pro rozdělení dat na trénovací a testovací sadu bude využita metoda "train_test_split" z balíčku "sklearn", která data rozdělí v poměru 75:25 (trénovací : testovací).

Pro trénování modelu obsahuje knihovna NLTK třídu "SklearnClassifier()", která slouží k integraci klasifikátorů z knihovny Scikit-learn do NLTK frameworku. Tato třída umožňuje využívat klasifikátory poskytované v knihovně Scikit-learn jako součást rozhraní NLTK. Klasifikátory jsou do této třídy předávány jako vstupní parametry.

Pro testování a evaluaci modelu knihovna NLTK poskytuje implementované funkce jako "nltk.classify.accuracy()", která slouží k výpočtu přesnosti klasifikátoru.

Tato funkce porovnává predikce klasifikátoru s očekávanými hodnotami a vrací podíl správně klasifikovaných instancí na celkovém počtu instancí. Dále je k dispozici metoda "classify_many()", která umožňuje klasifikovat více instancí najednou. Tato metoda přijímá seznam instancí a vrací seznam jejich predikcí, který je dále využit pro vytvoření výsledné matice záměn. Pro samotné zobrazení výsledných metrik nemá knihovna NLTK žádné třídy ani metody, tudíž bude využito tříd "classification_report" a "confusion_matrix" z balíčku "sklearn.metrics".

4.1.4 Scikit-learn

Knihovna Scikit-learn neposkytuje implementaci pro lemmatizaci obecně, tudíž ani pro české texty. Proto je pro tento krok předzpracování využita metoda "simplemmatizace" (viz Obr. 7). Odstranění stop slov a převedení textu na malá písmena lze provést současně během vektorizace textu, neboť metoda "CountVectorizer" umožňuje, kromě samotného převodu slov na číselnou reprezentaci, provedení obou těchto kroků. Knihovna Scikit-learn není vybavena seznamem českých stop slov, a proto je použit seznam "Stopwords Czech" z kolekce "Stopwords ISO". Vektorizace je prováděna již zmíněnou BOW (viz 3.6.2) metodou "CountVectorizer", která převádí text na vektory čísel, kde každý vektor obsahuje počet výskytů jednotlivých slov ve zpracovávaném textu. Tato metoda přijímá několik vstupních argumentů:

- "max_features" — Určuje maximální počet slov vytvořených z textu. V tomto případě bude používat pouze 1500 nejčastějších slov.
- "min_df" — Určuje minimální počet vět ve kterých se musí slovo vyskytovat, aby mohlo být přidáno do výsledného seznamu. Zde musí slovo být obsaženo v 5 nebo více větách.
- "max_df" — Určuje maximální relativní frekvenci dokumentů, ve kterých se slovo vyskytuje, aby mohlo být přidáno do výsledného seznamu. Zde je slovo zařazeno, pokud se vyskytuje v méně než 70 % dokumentů. Určení parametru max_df má za cíl filtrovat slova, která jsou příliš častá a nejsou relevantní pro analýzu.
- "stop_words" — Tato proměnná obsahuje seznam slov, která mají být ignorována při vytváření seznamu. Zde je metodě předán seznam stop slov "Stopwords Czech".

- "lowercase" Určuje, zda mají být slova převedena na malá písmena před zpracováním. V tomto případě je nastaveno na True.

Nakonec je využita metoda "fit_transform", která provádí dva hlavní kroky. Nejdříve naučí CountVectorizer na základě poskytnutého textu (část "fit"), a poté provede převod textu na vektory čísel (část "transform"). Výstup je uložen v proměnné "text" jako pole (array) vektorů čísel, které reprezentují jednotlivé texty.
Dále je provedena další transformace vektorů čísel, které byly vytvořeny předchozí metodou, tentokrát pomocí metody TF-IDF (viz 3.6.4), která využívá třídu "TfidfTransformer". Bude využita metoda "fit_transform", která se používá k přizpůsobení transformátoru datům a jejich následné transformaci. V tomto případě přijímá vstupní matici "text" (obsahující vektory čísel z předchozí metody) a provede TF-IDF transformaci. Tímto způsobem bude získáno pole vektorů čísel, které obsahují TF-IDF hodnoty namísto jednoduchých početních hodnot, které byly vytvořeny metodou BOW.
Výsledné vektory spolu s popisnými daty jsou rozděleny na trénovací a testovací data a jejich odpovídající popisky pomocí metody "train_test_split" v poměru 75:25 (trénovací : testovací). Trénovací data a příslušné popisky jsou následně předány metodě "fit" pro natrénování modelu. Testovací data jsou poté předána metodě "predict", která pro ně vytvoří predikované popisky témat. Výsledné metriky a matice záměn jsou zobrazeny pomocí tříd "classification_report" a "confusion_matrix", kterým jsou předány skutečné a predikované hodnoty témat testovacích dat.

4.1.5 Keras

Keras nenabízí žádné třídy ani metody pro lemmatizaci textu či odstranění stop slov. Proto je nutné opět využít metodu "simplemmatizace" (viz Obr. 7) pro lemmatizaci, a seznam českých stop slov "Stopwords Czech" z kolekce "Stopwords ISO" pro odstranění stop slov. Nicméně Keras nabízí třídy a metody pro tokenizaci a vektorizaci textu. K tomuto účelu slouží funkce "tav", jejíž implementaci lze nalézt na Obr. 10.

```
def tav(texts):
    # Tokenizace a vektorizace textu
    tokenizer = Tokenizer(num_words=1500)
    tokenizer.fit_on_texts(texts)
    sequences = tokenizer.texts_to_sequences(texts)

    # Padding sekvencí
    maxlen = []
    for text in sequences:
        maxlen.append(len(text))
    maxlen = max(maxlen)
    padded_sequences = tf.keras.utils.pad_sequences(sequences, maxlen=maxlen)

    return padded_sequences, maxlen
```

Obr. 10 Metoda pro tokenizaci a vektorizaci s využitím Keras. Zdroj: Autor

Funkce provádí tokenizaci textu a vytváří sekvence čísel z jednotlivých slov v textu. Na začátku je vytvořen objekt Tokenizer, kterému je předán parametr "num_words", určující maximální počet unikátních slov, které Tokenizer zachová při tokenizaci textu. V tomto případě bude hodnota nastavena na 1500, stejně jako v předchozích knihovnách. Dále se pomocí metody "fit_on_texts" Tokenizer učí na základě poskytnutých textů, a během tohoto procesu vytváří slovník, který mapuje slova na číselné identifikátory (tokeny). Metoda "texts_to_sequences" převede textová data na sekvence čísel podle vytvořeného slovníku Tokenizeru, kde každé slovo v textu je nahrazeno odpovídajícím číselným identifikátorem z tohoto slovníku. Následně je hledána maximální délka textu v kolekci textů recenzí, která bude použita k nastavení maximální délky výstupních sekvencí při paddingu. Padding je proces přidání nulových hodnot nebo speciálního znaku na začátek nebo na konec datové sekvence tak, aby dosáhla požadované délky. To je důležité pro vstup do neuronové sítě, která vyžaduje, aby všechny vstupní data měla stejný rozměr. Bez paddingu by se sekvence různých délek musely zpracovávat odděleně, což by mohlo způsobit problémy při vytváření nebo trénování modelu. Nakonec metoda "pad_sequences" provede padding textových sekvencí na stejnou délku. Pokud je text kratší než maximální délka (maxlen), budou do něj na začátek přidány nulové hodnoty. Parametr "padding" v této metodě umožňuje určit, zda se nulové hodnoty přidají na začátek nebo na konec. Výchozí hodnota je "pre", což znamená, že nulové hodnoty budou přidány na začátek. Na výstupu metoda vrací Numpy pole s vektorizovanými a vyplněnými sekvencemi textů, a číslo označující maximální délku sekvence textu nalezenou v datasetu.

Následuje definice modelu neuronové sítě. Bude použit sekvenční model, což je lineární způsob definice vrstev neuronové sítě, kde každá vrstva má jeden vstup a jeden výstup. Model bude obsahovat celkem 4 vrstvy. První vrstvou bude tzv. embedding vrstva, která transformuje vstupní čísla na vektorové reprezentace, jež se dále učí během trénování modelu. Parametry této vrstvy jsou následující:

- "input_dim" — Velikost slovníku, v tomto případě nastavena na hodnotu 1500.
- "output_dim" — Velikost výstupního vektoru, tedy dimenze vektoru pro každé slovo. V tomto případě je nastavena na hodnotu 64.
- "input_length" — Délka vstupní sekvence, což je maximální délka sekvence vstupních dat. V tomto případě bude použita hodnota proměnné "maxlen".

Druhou vrstvou je tzv. flatten vrstva, která redukuje data do jednorozměrného pole. Tato vrstva slouží k transformaci vícerozměrné vektorové reprezentace na jeden dlouhý vektor, který může být vstupem pro další vrstvy. Třetí vrstvou je tzv. dense vrstva s 32 neurony a ReLU aktivační funkcí, která přiřazuje váhy každému neuronu ve vstupním vektoru. Aktivační funkce ReLU umožňuje modelu naučit se nelineární vzory v datech. Poslední vrstvou je opět dense vrstva, tentokrát s třemi neurony (podle počtu témat), a aktivační funkcí softmax. Tato vrstva produkuje pravděpodobnostní rozložení mezi více třídami. Aktivační funkce softmax zajistí, že výstupní hodnoty jsou normalizované do rozmezí (0, 1), a součet pravděpodobností výstupních tříd je rovno jedné.
Samotný model je nakonec nutné zkompilovat. K tomu slouží metoda "compile", které je nutné nastavit několik parametrů:

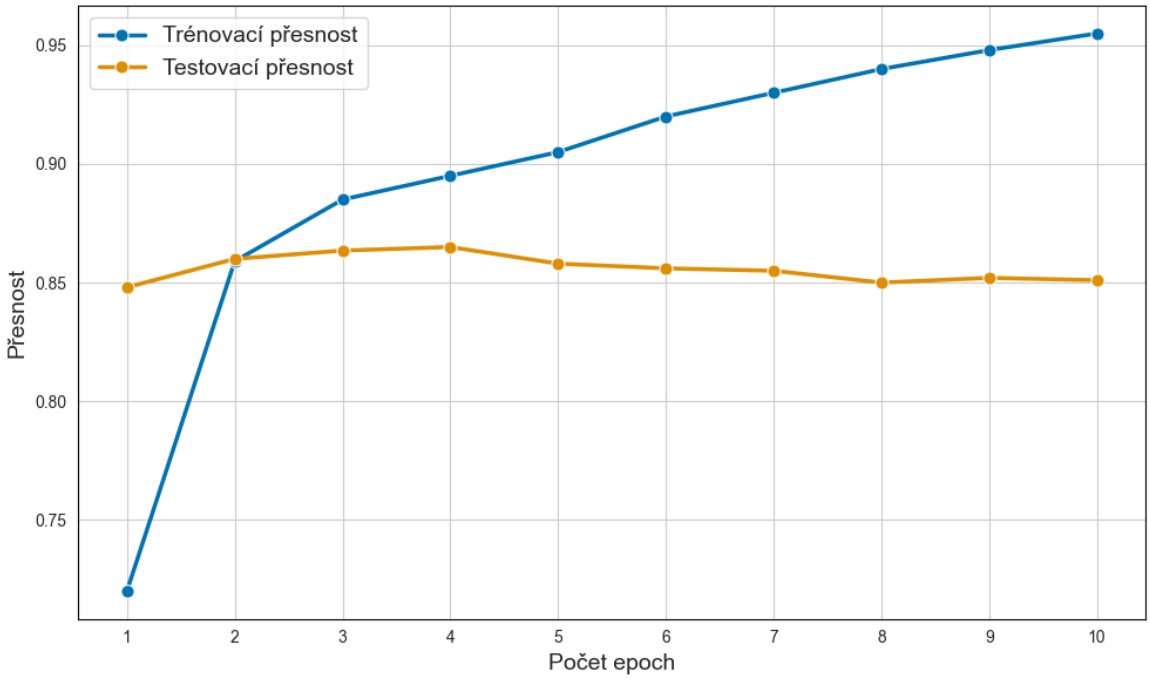
- "optimizer" určuje algoritmus, který bude použit pro optimalizaci vah modelu během trénování. V tomto případě je použit RMSprop, což je variantní metoda gradientního sestupu.
- "loss" (ztrátová) funkce určuje způsob, jakým se měří chyba mezi skutečnými a predikovanými hodnotami. V tomto případě je ztrátová funkce "categorical_crossentropy", která se běžně používá pro problémy klasifikace, kde existují více než dvě třídy. Tato ztrátová funkce vyžaduje kategorické cílové proměnné ve formátu one-hot encoding. To znamená, že namísto jednoduchého seznamu hodnot, který bude obsahovat 0 pro pozitivní texty, 1 pro negativní texty atd., je nutné mít pro každou třídu seznam, kde je na indexu

odpovídajícímu třídě jednička a na všech ostatních nula. K tomu bude využita metoda "to_categorical", které jsou předány původní číselné popisky a počet tříd (v tomto případě 3).

- "metrics" je hodnota, která je použita k vyhodnocení výkonu modelu během trénování. Zde je uvedena metrika 'acc', což znamená přesnost (accuracy).

Pro rozdělení dat na trénovací a testovací sadu bude využita metoda "train_test_split" z balíčku "sklearn", která data rozdělí v poměru 75:25 (trénovací : testovací).
Pro trénování modelu bude použita metoda "fit", které jsou kromě trénovacích dat předány další dva vstupních parametry:

- "epochs" určuje, kolikrát celý model projde všechna trénovací data. Je zásadní správně určit tuto hodnotu, aby model nebyl nenatrénovaný ani přetrénovaný. Obr. 11 ukazuje vývoj přesnosti modelu při provádění 10 opakování.



Obr. 11 Graf vývoje přesnosti modelu při trénování. Zdroj: Autor

Lze si všimnout, že trénovací přesnost stoupá s každou epochou, nicméně testovací přesnost roste pouze do chvíle, kdy je předčena trénovací přesností. Tento jev naznačuje případ přetrénování, kdy si model vede lépe na trénovacích datech než na datech, která nikdy předtím neviděl. Po dosažení tohoto bodu se model příliš specializuje na trénovací data a učí se pro ně specifické

reprezentace, které se nezobecňují na testovací data. Optimální počet epoch bude tedy hodnota před tímto zlomem.

Keras disponuje implementovanými metodami nazývanými "Callbacky" pro ukončení trénování v optimálním okamžiku. Konkrétně bude využit callback s názvem "EarlyStopping", který je konfigurován tak, aby monitoroval validační ztrátu (val_loss) a ukončil trénování, pokud se tato ztráta nezlepší během dvou po sobě jdoucích epoch (parametr "patience"). Na konci trénování model vrátí hodnotu přesnosti o 2 epochy zpět.

- "batch_size" specifikuje velikost dávky použité během trénování. Zde bude nastavena hodnota 32, což znamená, že váhy sítě se aktualizují po každých 32 trénovacích příkladech.

Dále bude vytvořen seznam pravděpodobností pro každý vstupní vzorek, který model přiřadil k jednotlivým třídám pomocí metody "predict". Z těchto pravděpodobností je poté sestaveno pole indexů tříd s nejvyššími pravděpodobnostmi pro každý vstupní vzorek.

Nakonec proběhne výpočet přesnosti a ztráty pomocí metody "evaluate" na testovacích datech. Výsledky budou zobrazeny spolu s maticí záměn pomocí balíčků "confusion_matrix" a "classification_report" z knihovny sklearn.

4.2 Modelování

V následujících kapitolách budou podrobně popsány postupy a metody použité při aplikaci nástrojů pro modelování témat, jak byly uvedeny v teoretickém úvodu, na reálných českých textech.

4.2.1 Data

Pro aplikace nástrojů pro modelování témat bude využit Korpus českých veršů[2], který je budován v Ústavu pro českou literaturu Akademie věd ČR. Zdrojový repositář obsahuje 1305 knih poezie z Korpusu českých veršů ve formě JSON souborů, kde každý soubor představuje básně z jedné básnické knihy. Tyto soubory obsahují nejen samotné texty básní, ale také metadata, jako jsou anotace rýmů, fonetický přepis,

[2] https://github.com/versotym/corpusCzechVerse

tokenizovaná a lemmatizovaná slova, morfologické značení a další. [17][16] Ukázka části jednoho ze souborů lze vidět na Obr. 12.


```
[
  {
    "p_author": {
      "born": 1841,
      "died": 1900,
      "name": "Albert, Eduard",
      "identity": "Albert, Eduard"
    },
    "biblio": {
      "motto_aut": null,
      "b_subtitle": "B\u00e1sn\u011b",
      "publisher": "\u0160im\u00e1\u010dek, Franti\u0161ek; Unie",
      "edition": "[1.]",
      "motto": null,
      "p_title": "JAROSLAVU VRCHLICK\u00c9MU",
      "place": "Praha",
      "dedication": null,
      "b_title": "Na zemi a na nebi",
      "pages": "[XVI]+104",
      "year": "1900",
      "signature": "N\u00e1rodn\u00ed knihovna \u010cR, Praha; 54 H 2287"
    },
    "book_id": "0001",
    "poem_id": "0001-0000-0000-0001-0000",
    "b_author": {
      "born": 1841,
      "died": 1900,
      "name": "Albert, Eduard",
      "identity": "Albert, Eduard"
    },
    "body": [
      [
        {
          "text": "Tv\u00e1 lo\u010f jde po vysok\u00e9m mo\u0159i,",
          "punct": {
            "6": ","
          },
          "words": [
            {
              "token_lc": "tv\u00e1",
              "xsampa": "tva:",
              "morph": "PSFS1-S2------1-",
              "phoebe": "tvA",
              "token": "Tv\u00e1",
              "lemma": "tv\u016fj"
            },
            {
              "token_lc": "lo\u010f",
              "xsampa": "loc",
              "morph": "NNFS1-----A-----",
              "phoebe": "loT",
              "token": "lo\u010f",
              "lemma": "lo\u010f"
            },
```


Obr. 12 Ukázka souboru z Korpusu českých veršů. Zdroj: Autor

4.2.2 Předzpracování

Vzhledem k tomu, že korpus obsahuje již předzpracovaná textová data, proces předzpracování bude výrazně usnadněn, neboť se bude skládat jen z načtení dat a odstranění stop slov. Pro tento účel bude využita metoda "poemsExtraction", jejíž implementaci lze vidět na Obr. 13. Tato metoda postupně načte všechny zdrojové soubory a provede extrakci lemmat ze slov jednotlivých básní. Výstupem této metody bude seznam básní, kde každá báseň obsahuje seznam lemmat jednotlivých slov. V rámci tohoto procesu bude rovněž provedeno odstranění stop slov s využitím "Stopwords Czech" z kolekce "Stopwords ISO".

```
def poemsExtraction():
    with open("stopwords-cs.txt", "r", encoding="utf-8") as stop_words_file:
        stop_words = stop_words_file.read()

    corpus_folder_path = "datasets/corpusCzechVerse-master/ccv"
    corpus_files = os.listdir(corpus_folder_path)

    poems = []
    for corpus_file in corpus_files:
        corpus_file_path = os.path.join(corpus_folder_path, corpus_file)
        with open(corpus_file_path) as file:
            data = json.load(file)

        for poem_data in data:
            poem = []
            poem_body = poem_data['body']
            for line in poem_body:
                for word_info in line:
                    for word in word_info['words']:
                        lemma = word['lemma']
                        if lemma not in stop_words:
                            poem.append(lemma)
            poems.append(poem)
    return poems
```

Obr. 13 Metoda pro přípravu dat z Korpusu českých veršů. Zdroj: Autor

Metoda začíná načtením českého seznamu stop slov do proměnné "stop_words". Následně cyklus postupně prochází obsah složky se zdrojovými soubory, které jsou ve formátu JSON načteny. Z každého souboru jsou postupně extrahovány hodnoty s klíči "body", "words" a nakonec "lemma". Pokud se hodnota klíče "lemma" nenachází v seznamu stop slov, je uložena do seznamu "poem", který obsahuje všechna slova

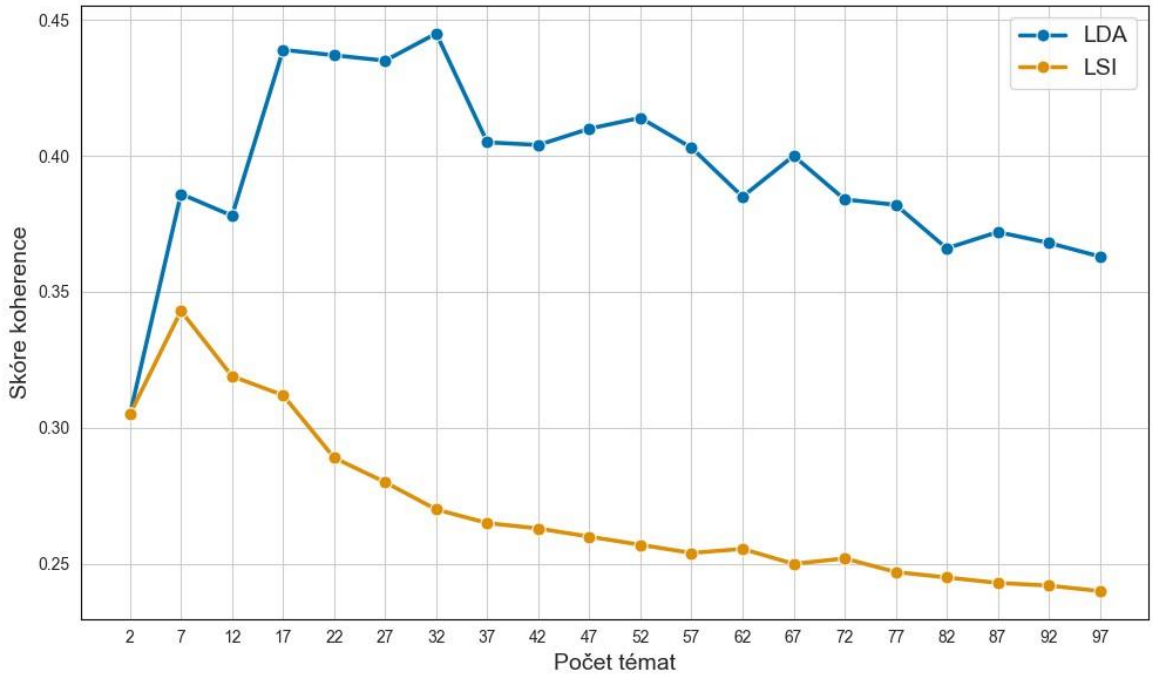
jedné básně. Po dokončení zpracování básně je obsah tohoto seznamu přidán do seznamu "poems". Po zpracování všech básní ze všech souborů je seznam seznamů lemmat vrácen na výstupu metody.

4.2.3 Gensim

Gensim poskytuje kompletní sadu metod a nástrojů pro implementaci a evaluaci LDA a LSI modelů.

Nejprve bude použita třída "Dictionary" k namapování slov na unikátní identifikátory, a poté metoda "doc2bow", která převádí namapovaná slova na seznam slov a jejich četností v daném dokumentu ve formátu (ID slova, četnost), čímž bude vytvořená BOW reprezentace.

Výstupy těchto kroků budou předány třídám pro inicializaci LDA a LSI modelů, spolu s počtem témat, které mají být vygenerovány. Pro model LDA je možné specifikovat také alfa a beta parametry. Optimální počet témat lze nalézt spuštěním evaluace modelu pro různé hodnoty počtu témat. Výsledek této evaluace pro modely LDA a LSI je zobrazen na Obr. 14.



Obr. 14 Graf vývoje koherence Gensim LDA a LSI modelů pro různé hodnoty počtu témat. Zdroj: Autor

Cyklus byl spuštěn s počátečním počtem dvou témat a inkrementován po pěti tématech až k téměř sto tématům. U modelu LDA se skóre koherence zvyšovalo s počtem témat

až do dosažení hodnoty 37 (počet témat), po níž začalo klesat. Před dosažením této hodnoty však skóre koherence již dvakrát po sobě kleslo, takže je vhodné zvolit poslední hodnotu před těmito poklesy, a to 17. Pro model LSI bylo nejvyšší skóre koherence dosaženo se sedmi tématy. Pro výsledný model této práce bude v kódu implementováno automatické ukončení cyklu po detekci dvou po sobě jdoucích klesajících hodnotách skóre koherence. Pokud by tato podmínka nenastala, bude vybrán počet témat s nejvyšším skóre koherence. V případě, že je tato podmínka splněna, ale dosažené skóre koherence je menší než maximální hodnota dosud nalezená, bude použita hodnota počtu témat, která odpovídá dosaženému maximu.

Hodnoty parametrů alfa a beta lze nalézt obdobným způsobem, tedy spuštěním evaluace modelu pro různé kombinace hodnot alfa a beta parametrů, při optimálním počtu témat zjištěném v předchozím kroku. Výstupem budou optimální hodnoty s nejvyšší koherencí, která bude představovat výslednou přesnost modelu. V tomto případě jsou to hodnoty:

$$alfa = 0{,}1$$

$$beta = 0{,}9$$

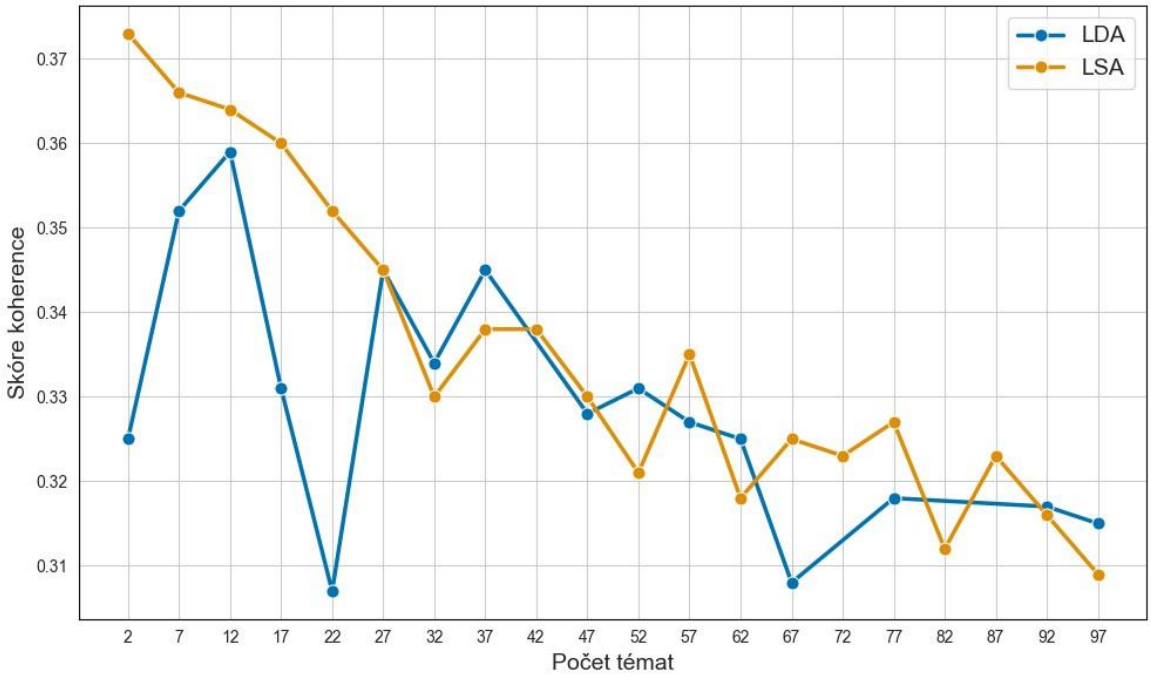
Koherenci modelu lze vypočítat pomocí třídy "CoherenceModel", které stačí předat inicializovaný model, seznam básní, slova s identifikátory a požadovanou evaluační metriku, v tomto případě koherenci CV. Výslednou hodnotu lze poté získat pomocí metody "get_coherence".

4.2.4 Scikit-learn

Scikit-learn poskytuje kompletní implementaci LDA a LSA modelů, avšak neobsahuje vestavěné metody pro jejich předpřípravu, evaluaci a výpočet koherence.

Nejprve je nutné vytvořit TF-IDF matici, která bude později převedena na dokument-termín matici. Pro vytvoření slovníku bude použita třída "Dictionary" z knihovny Gensim. Z tohoto slovníku budou extrahována unikátní slova, ze kterých bude vybráno deset nejdůležitějších slov pro každé téma. Seznam seznamů těchto slov bude později použit jako vstupní parametr pro evaluační metodu namísto modelu.

Následuje inicializace LDA a LSA modelů, kde je nutné určit počty témat. U modelu LDA jsou také specifikovány parametry alfa a beta. Podobně jako v předchozím případě bude optimální počet témat nalezen spuštěním evaluace modelu pro různé hodnoty počtu témat. Výsledky této evaluace pro modely LDA a LSA jsou zobrazeny na Obr. 15.



Obr. 15 Graf vývoje koherence Sklearn LDA a LSA modelů pro různé hodnoty počtu témat. Zdroj: Autor

Cyklus byl spuštěn s počátečním počtem dvou témat a inkrementován po pěti až k téměř stovce témat. U modelu LDA skóre koherence s rostoucím počtem témat kolísá s klesajícím trendem. Nejvyšší skóre koherence bylo dosaženo se dvanácti tématy. U modelu LSA bylo nejvyšší skóre dosaženo již při první hodnotě dvou témat. Pro výsledný model této práce bude v kódu implementováno automatické ukončení cyklu po detekci dvou po sobě jdoucích klesajících hodnot skóre koherence. Pokud by tato podmínka nenastala, bude vybrán počet témat s nejvyšším skóre koherence. V případě, že je tato podmínka splněna, ale dosažené skóre koherence je menší než maximální hodnota dosud nalezená, bude použita hodnota počtu témat, která odpovídá dosaženému maximu.

S optimálním počtem témat lze provést další evaluaci modelu za účelem nalezení optimálních hodnot parametrů alfa a beta. Kombinace hodnot těchto parametrů s nejvyšším skóre koherence bude představovat výslednou přesnost modelu. V tomto konkrétním případě jsou to následující hodnoty:

$$alfa = 0{,}8$$

$$beta = 0{,}9$$

Jak již bylo zmíněno, knihovna Scikit-learn neposkytuje nástroje pro výpočet skóre koherence, proto bude použita evaluační třída z knihovny Gensim, která umožňuje

evaluaci modelů i z jiných knihoven. Této třídě bude předán seznam seznamů nejdůležitějších slov pro daná témata, slovník a texty básní.

4.2.5 Tomotopy

Tomotopy nabízí komplexní sadu metod a nástrojů pro přípravu, použití a evaluaci pouze LDA modelů.

Po načtení dat je možné okamžitě začít s modelováním, neboť veškeré výpočty aprocesy, jako je vytvoření dokument-termín matice, slovníku a tak dále, jsou interně prováděny při trénování modelu. Opět je nutné specifikovat počet témat a parametry alpha a beta. Díky vysoké rychlosti modelu lze spustit evaluační cyklus pro všechny kombinace těchto parametrů a vybrat ty, které dosáhnou nejvyššího skóre koherence. V tomto případě dosáhl model nejvyššího skóre koherenci s 87 tématy a parametry alfa a beta:

$$alfa = 0{,}9$$

$$beta = 0{,}9$$

Po inicializaci modelu jsou mu jednotlivě předány texty básní a následně je model trénován.

Pro výpočet koherence je využita třída "Coherence", které je předán pouze model a žádaná evaluační metrika, v tomto případě CV.

5 Výsledky

V této kapitole budou prezentovány výsledky dosažené pomocí metod a nástrojů popsaných v předchozí kapitole. Všechny nástroje automaticky zaznamenávají výsledky do souboru CSV. Pro klasifikaci jsou výsledky ukládány do souboru "Klasifikace_výsledky.csv", zatímco pro modelování do souboru "Modelování_výsledky.csv". Je důležité mít na paměti, že každý běh programu může vrátit různé, ač podobné výsledky, které se mohou lišit od těch dosažených autorem.

5.1 Klasifikace

Výsledky jednotlivých klasifikátorů jsou zobrazeny v Tab. 4 a uspořádány sestupně podle dosažené přesnosti.

Tab. 4 Výsledná přesnost klasifikátorů. Zdroj: Autor

Klasifikátor	Přesnost
ScikitLearn Metoda podpůrných vektorů	0,753
ScikitLearn Logistická regrese	0,748
Keras Neuronové sítě	0,742
ScikitLearn Naivní Bayesovský klasifikátor	0,739
NLTK Logistická regrese	0,736
NLTK Metoda podpůrných vektorů	0,733
NLTK Naivní Bayesovský klasifikátor	0,727
NLTK Rozhodovací strom	0,582

Z tabulky je zřejmé, že nejvyšší přesnosti dosáhl klasifikátor "Metoda podpůrných vektorů" z knihovny ScikitLearn, který dosáhl hodnoty přesnosti 0,753. S výjimkou rozhodovacího stromu z knihovny NLTK vykázaly všechny klasifikátory velmi podobné výsledky, které se lišily pouze v rozmezí desetin a v některých případech i setin. Lze pozorovat, že klasifikátory implementované v rámci knihovny NLTK dosáhly nižších výsledků. V kontrastu k tomu ty samé klasifikátory v rámci knihovny ScikitLearn dosáhly lepších výsledků.

Pro lepší vizualizaci efektivity jednotlivých klasifikátorů v rozdělení testovacích dat do specifických témat jsou na Obr. 16 zobrazeny matice záměn jednotlivých klasifikátorů.

ScikitLearn Metoda podpůrných vektorů		Predikce		
		Pozitivní	Negativní	Neutrální
Skutečnost	Pozitivní	0,748	0,178	0,075
	Negativní	0,205	0,693	0,102
	Neutrální	0,083	0,098	0,819

ScikitLearn Logistická regrese		Predikce		
		Pozitivní	Negativní	Neutrální
Skutečnost	Pozitivní	0,733	0,184	0,083
	Negativní	0,197	0,687	0,116
	Neutrální	0,077	0,099	0,824

Keras Neuronové sítě		Predikce		
		Pozitivní	Negativní	Neutrální
Skutečnost	Pozitivní	0,796	0,140	0,064
	Negativní	0,261	0,631	0,108
	Neutrální	0,107	0,090	0,802

ScikitLearn Naivní Bayesovský		Predikce		
		Pozitivní	Negativní	Neutrální
Skutečnost	Pozitivní	0,713	0,188	0,099
	Negativní	0,193	0,686	0,121
	Neutrální	0,079	0,103	0,818

NLTK Logistická regrese		Predikce		
		Pozitivní	Negativní	Neutrální
Skutečnost	Pozitivní	0,756	0,171	0,073
	Negativní	0,230	0,659	0,111
	Neutrální	0,100	0,105	0,795

NLTK Metoda podpůrných vektorů		Predikce		
		Pozitivní	Negativní	Neutrální
Skutečnost	Pozitivní	0,745	0,184	0,071
	Negativní	0,225	0,663	0,112
	Neutrální	0,111	0,098	0,791

NLTK Naivní Bayesovský		Predikce		
		Pozitivní	Negativní	Neutrální
Skutečnost	Pozitivní	0,737	0,164	0,098
	Negativní	0,229	0,645	0,126
	Neutrální	0,104	0,095	0,801

NLTK Rozhodovací strom		Predikce		
		Pozitivní	Negativní	Neutrální
Skutečnost	Pozitivní	0,598	0,264	0,138
	Negativní	0,287	0,520	0,193
	Neutrální	0,167	0,205	0,628

Obr. 16 Matice záměn klasifikátorů. Zdroj: Autor

Matice jsou uspořádány podle předchozí tabulky a zobrazují podíly správně a nesprávně klasifikovaných textů v rámci každého tématu. Z obrázku lze pozorovat, že matice velmi dobře reflektují celkové přesnosti uvedené v Tab. 4. Dále je patrné, že všechny klasifikátory nejlépe rozřadily recenze do tématu "neutrální" a nejhůře do tématu "negativní".

5.2 Modelování

Výsledky jednotlivých modelů jsou prezentovány v Tab. 5 a uspořádány sestupně podle dosaženého skóre koherence.

Tab. 5 Výsledná skóre koherence modelů. Zdroj: Autor

Model	Skóre koherence	Počet témat	Alfa	Beta
Tomotopy LDA	0,954	87	0,9	0,9
Gensim LDA	0,624	17	0,1	0,9
ScikitLearn LDA	0,414	7	0,9	0,8
ScikitLearn LSA	0,383	2		
Gensim LSI	0,344	7		

Z tabulky je patrné, že nejvyšší skóre koherence bylo dosaženo pomocí modelu Tomotopy LDA, kde bylo dosaženo hodnoty 0,954 s 87 tématy a nastavením parametrů

alpha a beta na 0,9. Společně s modelem Gensim LDA jsou to jediné modely, které v koherenci překročily hranici 0,5. Dále je zřejmé, že modely LDA obecně dosahují vyššího skóre koherence než modely LSA (LSI).
Skóre koherence představuje užitečnou metriku pro srovnání výkonu různých modelů, avšak pro hlubší interpretaci modelů a pochopení člověkem je klíčový výběr nejreprezentativnějších slov, která charakterizují jednotlivá témata. Pro ukázku toho, jak dobře tyto modely popisují hledaná témata byla analyzována báseň "Jaroslavu Vrchlickému" od Eduarda Alberta. Jedná se o první báseň korpusu a její text je následující:

Tvá loď jde po vysokém moři,
v ně brázdu jako stříbro reje,
svou příď v modré vlny noří
a bok svůj pěnné do peřeje.

Tvá lana sviští, plachty duní
a třepe vlajka. V noční chvíli
zříš magický svit mořských tůní,
a ve snu, Albatros jak pílí.

Já samotním, jsem na ostrově,
ohýnek topím, rybku lově
zasedám na břeh za večera.

Dým v kotoučích se modrých krade,
kdes písklo ptáče, ještě mladé,
tma na mne hrozí z pološera. [1]

Tab. 6 zobrazuje deset nejvýznamnějších slov (uspořádaných podle významnosti) daného tématu, která modely určily pro jedno téma. Modely jsou opět seřazeny sestupně dle dosaženého skóre koherence uvedeného v předchozí tabulce. Vzhledem k tomu, že báseň reflektuje více tematických rovin, je zřejmé, že žádný z modelů nedokázal komplexně shrnout celou báseň pod jedno dominující téma. Všechny modely se zaměřují na klíčové prvky básně, jako jsou moře a přírodní prostředí spojené

s mořskou plavbou. Některé modely, jako jsou Gensim LDA a Scikitlearn, zdůrazňují i tajemství a magii moře, zatímco ostatní modely se více zaměřují na akci a pohyb na lodi či rybolov. Lze si všimnout, že model Tomotopy LDA se zdá preferovat primárně první odstavec básně, což odpovídá jeho lepším výsledkům při vyšším počtu témat, kdy zahrnuje větší část básní. Z tohoto pohledu, pokud je vyžadována detekce pouze jednoho tématu, se zdají být vhodnějšími modely pro celkovou analýzu Gensim LDA a ScikitLearn LDA, které zdůrazňují širší škálu klíčových prvků básně, což poskytuje komplexnější obraz tématu básně. Všechny modely poskytují různé perspektivy na hlavní téma básně a zachycují různé aspekty textu.

Tab. 6 Témata básně "Jaroslavu Vrchlickému" podle různých modelů. Zdroj: Autor

Model	10 nejvýznamnějších slov tématu
Tomotopy LDA	modrý, loď, jít, vysoký, moře, brázda, stříbro, rej, přijít, vlna
Gensim LDA	modrý, magický, tůň, lovit, kotouč, stříbro, pěnné, rybka, jít, hrozit
ScikitLearn LDA	modrý, zřít, zasedat, nořit, noční, mořský, moře, mladý, magický, loď
ScikitLearn LSA	modrý, albatros, zasedat, nořit, noční, mořský, moře, mladý, magický, loď
Gensim LSI	modrý, albatros, dunět, lovit, jít, loď, kdes, kotouč, lano, břeh

6 Diskuze

V oblasti klasifikace témat dosáhl nejvyšší přesnosti klasifikátor založený na metodě podpůrných vektorů, implementovaný v knihovně ScikitLearn, jehož přesnost 0,753 představuje nejvyšší dosaženou přesnost v rámci této práce. S ohledem na autorovy předpoklady o nadřazenosti hlubokého učení, reprezentovaného knihovnou Keras, je jeho umístění na třetím místě překvapivé. Přesnost klasifikátoru je však závislá na řadě faktorů, jejichž optimalizace má na výsledky různý vliv. V první řadě je důležité zdůraznit kvalitu poskytnutých dat. Analýza datasetu ukázala, že texty tématu "neutrální" se od textů z ostatních témat příliš neliší. V některých recenzích je dokonce možné nalézt významově téměř totožné věty, což ztěžuje jejich správné klasifikování. Dále by optimalizace předzpracování textů mohla být posílena rozšířením seznamu stop slov. V případě hlubokého učení v kontextu klasifikátoru Keras, úpravy v architektuře neuronové sítě, konkrétně přidání nebo odebrání vrstev, by také mohly významně přispět k zlepšení výsledků klasifikace.

V rámci modelování témat se model Tomotopy LDA jeví jako dominantní v porovnání s ostatními testovanými knihovnami a jejich modely. Vyšší skóre koherence modelů LDA ve srovnání s modely LSA (LSI) bylo očekáváno, jelikož modely LDA reprezentují pokročilejší formu základních modelů LSA (LSI). Lze polemizovat o tom, zda by se výsledky modelů výrazně lišily, kdyby bylo ladění hyperparametrů provedeno stejným způsobem jako u modelu Tomotopy LDA. Jistě by se kvalita všech modelů zvýšila využitím rozšířeného seznamu stop slov, neboť je možné si v Tab. 6 všimnout, že mezi deseti nejvýznamnějšími slovy modelu Gensim LSI je také slovo „kdes“, které samo o sobě nenese žádnou významovou hodnotu a lze ho tedy považovat za stop slovo. Rozdílnost výstupů modelů je navíc pravděpodobně ovlivněna použitím různých forem dokument-termínových matic, kde některé modely využívají metodu BOW, zatímco jiné přesnější TF-IDF reprezentaci.

7 Závěry a doporučení

Nelze jednoznačně popsat správný obecný postup provedení kroků předzpracování textu, neboť ten musí být vždy podřízen povaze zdrojových textových dat, konkrétnímu řešenému problému či oblasti aplikace. V některých případech mohou data vyžadovat minimální úroveň úprav, což bylo pozorováno například při zpracování korpusu českých veršů v této práci, naopak jiná data mohou vyžadovat komplexní předzpracování. Je rovněž vhodné provádět analýzu s různými konfiguracemi předzpracování a nejen po provedení všech. Existuje možnost, že klasifikační model dosáhne lepších výsledků bez určitých úprav, například bez lemmatizace, než s ní a podobně.

V rámci vektorové reprezentace textu dominují metody Bag-of-Words (BOW) a TF-IDF. Metoda BOW je obvykle preferována pro menší datové sady díky své jednoduchosti a nižší výpočetní náročnosti. Naopak, metoda TF-IDF se osvědčuje při práci s rozsáhlejšími datovými soubory. Avšak s narůstajícím objemem dat se zvyšuje i výpočetní náročnost, neboť tento model bere v úvahu frekvenci výskytu slov v jednotlivých dokumentech v porovnání s celkovou frekvencí v korpusu dokumentů. Pro dosažení nejlepších výsledků je vhodné provést analýzu pomocí obou metod a následně vybrat tu, která poskytuje lepší výsledky. Ačkoliv je TF-IDF sofistikovanější než BOW, nemusí vždy přinášet lepší výsledky.

Z dosažených výsledků je patrné, že kromě klasifikátoru založeného na rozhodovacím stromě jsou výkony všech klasifikátorů velmi srovnatelné. Lze tedy konstatovat, že výběr jakéhokoli z těchto klasifikátorů by byl vhodný. Pro implementaci strojového učení je výhodnější využít knihovny Scikit-learn, která tyto klasifikátory nabízí přímo, neboť NLTK používá pouze tzv. wrappers pro Scikit-learn k integraci těchto klasifikátorů, což může, jak naznačují výsledky, ovlivnit jejich přesnost.

Při modelování témat je volba optimálního modelu na základě generovaných slov do jisté míry subjektivní. Avšak na základě dosažených výsledků lze s jistotou tvrdit, že model LDA výrazně předčil model LSI (LSA), a je tudíž určitě lepší volbou. Přestože model LDA od Tomotopy dosáhl nejvyššího skóre koherence, doporučeným postupem je využít více implementací modelů LDA, a na základě nejvýznamnějších slov generovaných těmito modely následně vybrat ten, který podle subjektivního posouzení nejlépe reflektuje téma daného textu.

I přes výše zmíněná doporučení je stále nutné pamatovat na to, že výběr klasifikátoru či modelu závisí na specifikách výzkumu, dat a požadavcích na analýzu témat.

8 Seznam použité literatury

[1] Albert, E. (1900). JAROSLAVU VRCHLICKÉMU: Básně. [1. ed.]. Šimáček, F.; Unie. Praha: Národní knihovna ČR; 54 H 2287.

[2] Bird, Steven, Edward Loper and Ewan Klein (2009), *Natural Language Processing with Python*. O'Reilly Media Inc.

[3] CVETICANIN, Nikolina. What's On the Other Side of Your Inbox – 20 SPAM Statistics for 2023 [online]. 2023 [cit. 2023-11-09]. Dostupné z: https://dataprot.net/statistics/spam-statistics/

[4] FRIEDL, Jeffrey E. F. *Mastering regular expressions*. 3rd ed. Sebastapol, CA: O'Reilly, c2006. ISBN 978-059-6528-126.

[5] Habernal, I., Ptáček, T., & Steinberger, J. (2013). Sentiment analysis in Czech social media using supervised machine learning. In Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (pp. 65-74). Retrieved from https://aclanthology.org/W13-1609.pdf

[6] HUANG, Qi; CHEN, Zhanghao; LU, Zijie a YE, Yuan. Analysis of Bag-of-n-grams Representation's Properties Based on Textual Reconstruction. Online. 2018. Dostupné z: https://doi.org/https://doi.org/10.48550/arXiv.1809.06502. [cit. 2024-02-01].

[7] Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras

[8] KANDYBA, Jan. Detekce tématu dokumentu. Plzeň, 2019. Bakalářská práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra kybernetiky.

[9] KATRÁKOVÁ, Lenka. Text mining [online]. Brno, 2020 [cit. 2023-11-13]. Dostupné z: https://is.muni.cz/th/vvbow/DP_Text_mining_final.pdf. Diplomová práce. MASARYKOVA UNIVERZITA, Přírodovědecká fakulta, Ústav matematiky a statistiky. Vedoucí práce RNDr. Radim Navrátil, Ph.D.

[10] KHERWA, Pooja a Poonam BANSAL. Topic Modeling: A Comprehensive Review. *EAI Endorsed transactions on scalable information systems* [online]. 2019, **7**(24) [cit. 2024-02-22].

[11] KOWSARI, Kamran; JAFARI MEIMANDI, Kiana; HEIDARYSAFA, Mojtaba; MENDU, Sanjana; BARNES, Laura et al. Text Classification Algorithms: A Survey.

Online. *Information*. 2019, roč. 10, č. 4. Dostupné z: https://doi.org/10.3390/info10040150. [cit. 2024-02-22].

[12] Lee, Minchul. (2022). bab2min/tomotopy: 0.12.3 [Software]. Zenodo. https://doi.org/10.5281/zenodo.6868418

[13] LIDDY, Elizabeth D. Natural Language Processing: In Encyclopedia of Library and Information Science. Online. 2nd Edition. NY: Marcel Decker, 2001. Dostupné z: https://surface.syr.edu/cgi/viewcontent.cgi?article=1043&context=istpub. [cit. 2023-11-22].

[14] PARK, Hyeoun-Ae. An Introduction to Logistic Regression: From Basic Concepts to Interpretation with Particular Attention to Nursing Domain. *Journal of Korean Academy of Nursing* [online]. College of Nursing and System Biomedical Informatics National Core Research Center, Seoul National University, Seoul, Korea., 2013, 43(2), 154-164 [cit. 2024-02-22]. Dostupné z: doi:https://doi.org/10.4040/jkan.2013.43.2.154.

[15] PASCUAL, Frederico. Topic Modeling: An Introduction. Online. MonkeyLearn. 2019. Dostupné z: https://monkeylearn.com/blog/introduction-to-topic-modeling/. [cit. 2024-02-22].

[16] Plecháč, P. (2016). Czech Verse Processing System KVĚTA -- Phonetic and Metrical Components. Glottotheory, 7(2), 159-174. DOI: 10.1515/glot-2016-0013.

[17] Plecháč, P., & Kolár, R. (2015). The Corpus of Czech Verse. Studia Metrica et Poetica, 2(1), 107-118. DOI: 10.12697/smp.2015.2.1.05.

[18] Radim Řehůřek and Petr Sojka, "Software Framework for Topic Modelling with Large Corpora," in Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, pp. 45-50, ELRA, Valletta, Malta, May 22, 2010. [Online]. Available: http://is.muni.cz/publication/884893/en.

[19] RÖDER, Michael, Andreas BOTH a Alexander HINNEBURG. Exploring the Space of Topic Coherence Measures. *WSDM '15: Proceedings of the Eighth ACM International Conference on Web Search and Data Mining* [online]. New York, USA: Association for Computing Machinery, 2015, 399–408 [cit. 2024-02-26]. Dostupné z: doi: https://doi.org/10.1145/2684822.2685324.

[20] RYNT, Lukáš. Dvojjazyčné vyhledávání v dokumentech. Bakalářská práce, vedoucí Ing. David Bernhauer. Praha: České vysoké učení technické v Praze, Fakulta informačních technilogií, Katedra teoretické informatiky, 2022.

[21] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

[22] SMUTKA, Miroslav. Určování blízkost pojmů v oblasti informačních technologií. Bakalářská práce, vedoucí Doc. RNDr. SMRŽ PAVEL, Ph.D. Brno: Vysoké učení technické v Brně, Fakulta informačních technologií, Ústav počítačové grafiky a multimédií, 2016.

[23] T. HANCOCK, John a M. KHOSHGOFTAAR, Taghi. Survey on categorical data for neural networks. Online. *Journal of Big Data*. 2020, roč. 28, č. 7. Dostupné z: https://doi.org/https://doi.org/10.1186/s40537-020-00305-w. [cit. 2024-04-16].

[24] TESAŘÍKOVÁ, Anna. Topic Modeling for Corpus of Czech Verse. Bachelor's thesis. Praha: Czech technical university in Prague, Faculty of Information Technology, Department of Applied Mathematics, 2022.

[25] THOROVSKÝ, Martin. V čem je angličtina jednodušší než čeština? Online. Svobodná Angličtina. 2015. Dostupné z: https://svobodnaanglictina.cz/v-cem-je-anglictina-jednodussi-nez-cestina/. [cit. 2024-02-26].

[26] VULETA, Branka. How Much Data Is Created Every Day? +27 Staggering Stats [online]. 2021 [cit. 2023-11-09]. Dostupné z: https://seedscientific.com/how-much-data-is-created-every-day/

[27] WOLFF, Rachel. What Is Topic Analysis? Examples & Tools. Online. MonkeyLearn. 2020. Dostupné z: https://monkeylearn.com/blog/what-is-topic-analysis/. [cit. 2024-02-07].

9 Přílohy

9.1 Příloha 1 – Zdrojové kódy tematické analýzy

https://github.com/Jakub-Nevyhosteny/DP_Tematicka_Analyza

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání diplomové práce

Autor: **Bc. Jakub Nevyhoštěný**

Studium: I2100848

Studijní program: N0688A140019 Datová věda

Studijní obor: Datová věda

Název diplomové práce: **Text miningové nástroje pro tématickou analýzu česky psaných textů**

Název diplomové práce AJ: Text mining tools for topic discovery of Czech written texts

Cíl, metody, literatura, předpoklady:

Cíle práce:
Poskytnout komplexní přehled metod a nástrojů pro předzpracování a vektorizaci textů psaných v češtině a angličtině, jakož i pro jejich klasifikaci a modelování témat. Tyto nástroje aplikovat na skutečných datech a porovnat jednotlivé výsledky. Nakonec na základě výsledků navrhnout postup, kterým by se měl ubírat výzkumník, jenž chce provést tematickou analýzu česky psaných textů.

Osnova práce:

1. Teoretický úvod - Rešerše a představení jednotlivých kroků pro předzpracování, vektorizaci, klasifikace a modelování.
2. Metody - Praktické představení metod pro splnění dílčích úkonů představených v úvodu.
3. Výsledky - Interpretace a vizualizace výsledků dosažených v předchozí části.
4. Diskuze a závěr - Diskuzace nad dosaženými výsledky, a závěrečný výstup práce.

- https://www.nltk.org/
- https://keras.io/
- https://spacy.io/
- https://scikit-learn.org/stable/
- https://radimrehurek.com/gensim/
- https://monkeylearn.com/
- https://lindat.mff.cuni.cz/
- KATRÁKOVÁ, Lenka. Text mining [online]. Brno, 2020. Diplomová práce. MASARYKOVA UNIVERZITA, Přírodovědecká fakulta, Ústav matematiky a statistiky. Vedoucí práce RNDr. Radim Navrátil, Ph.D.
- KANDYBA, Jan. Detekce tématu dokumentu. Plzeň, 2019. Bakalářská práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra kybernetiky.
- KOHOUT, Lukáš. Identifikace pojmenovaných entit v textu. Bakalářská práce, vedoucí RNDr. Zuzana Nevěřilová. Brno: Masarykova Univerzita, Fakulta informatiky, 2012.
- Richter Michal, Straňák Pavel and Rosen Alexandr. Korektor – A System for Contextual Spell-checking and Diacritics Completion In Proceedings of the 24th International Conference on Computational Linguistics (Coling 2012), pages 1-12, Mumbai, India, 2012.
- SMUTKA, Miroslav. Určování blízkost pojmů v oblasti informačních technologií. Bakalářská práce, vedoucí Doc. RNDr. SMRŽ PAVEL, Ph.D. Brno: Vysoké učení technické v Brně, Fakulta informačních technologií, Ústav počítačové grafiky a multimédií, 2016.

Zadávající pracoviště: Katedra informatiky a kvantitativních metod, Fakulta informatiky a managementu

Vedoucí práce: Mgr. Jiří Haviger, Ph.D.

Datum zadání závěrečné práce: 15.10.2021