Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Metody modelování témat

Diplomová práce

Autor: Jiří Kánský
Studijní obor: Datová věda

Vedoucí práce: Ing. Martina Husáková, Ph.D.

Hradec Králové Duben 2024

Prohlášení:

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

V Hradci Králové dne

_____ Bc. Jiří Kánský

Poděkování:

Děkuji vedoucí diplomové práce Ing. Martině Husákové, Ph.D. za metodické vedení práce a cenné rady.

Abstrakt

Cílem práce je implementovat různé metody modelování témat v rámci extrakce informací z textu. Práce obsahuje praktické použití několika různých algoritmů na modelování témat, které jsou v jejím závěru porovnány. Práce nejdříve sblíží čtenáře se zpracováním přirozeného jazyka a extrakcí informací. Poté rozebírá teorii, o kterou se implementované metody opírají. Implementace jednotlivých algoritmů proběhla v rámci programovacího jazyku Python s využitím převážně knihoven Gensim, Top2Vec a BERTopic. Testovací dataset se skládá z 18.000 dokumentů v podobě novinových článků. Ve výsledném porovnání bylo zjištěno několik výhod a nevýhod jednotlivých algoritmů. Na základě výsledků potom práce vyhodnocuje, jaká z metod byla nejadekvátnější pro vybraný dataset.

Klíčová slova:

Zpracování přirozeného jazyka, extrakce informací, modelování témat, latent semantic analysis, latent Dirichlet allocation, non-negative matrix factorization, Top2Vec, BERTopic

Abstract

Title: Topic modeling methods

The goal of Diploma Thesis is implementation of various methods of topic modeling, which is a part of discipline known as information extraction from texts. The thesis contains practical elaboration on how to carry out various topic modeling algorithms and makes comparison between them afterwards. At first the thesis introduces natural language processing and information extraction. Afterwards it talks about theory behind each of the concerning algorithms. Implementation of individual algorithms is done via programming language Python, mostly using libraries Gensim, Top2Vec and BERTopic. Data used for testing contains about 18.000 documents in the form of news articles. Finally, during comparison various advantage and disadvantage of algorithms were found out. Based on these results, the thesis makes an assumption which method was best fit for the chosen dataset.

Key words:

Natural language processing, information extraction, topic modeling, latent semantic analysis, latent Dirichlet allocation, non-negative matrix factorization, Top2Vec, BERTopic

Obsah

1 Úvod

Nápad, že by počítače někdy byly schopny porozumět běžnému jazyku a dokonce i souvisle držet krok při konverzaci s člověkem, je součástí lidského myšlení už od první poloviny dvacátého století. Tato myšlenka byla také uvedena v článku od Alana Turinga v říjnu 1950 jako důsledek existence výpočetní inteligence . [24]

Od počátku dvacátého prvního století se tato vize stávala více a více pravděpodobnější, převážně díky spojení technik umělé inteligence s výzkumem přirozených jazyků v rámci aplikovatelnosti na průmysl a obchod. Nyní skoro každá webová stránka obsahuje možnost překladu jazyka, chytré mobily jsou schopné porozumět mluveným příkazům a otázkám. Vyhledávače typu Google jsou schopny využívat jednoduchých lingvistických technik k automatické úpravě či k doplnění dotazů, také k vyhledávání platných výsledků na základě dotazu nebo výsledků, které jsou nejblíže podobné. Zpracování přirozeného jazyka je oborem počítačové vědy, převážně umělé inteligence, který dokáže počítač naučit, jak pracovat s přirozenými jazyky ve formě psaného či mluveného slova, videí, obrázků a podobně.

Právě díky tomuto porozumění je možné vytvářet různé algoritmy, které jsou aplikovatelné pro praktické účely, jako například filtrování spamu, predikce psaného textu, analýza sémantiky, využití překladačů, chytrých asistentů apod. Aktuálním příkladem užitečnosti zpracování přirozeného jazyka je chatbot ChatGTP, který dokáže držet krok v konverzaci, anebo je využitelný k inspiraci, obecně automatizaci.

V rámci zpracování přirozeného jazyka lze také odpovědět na otázku: *"Jak je pro jednoho člověka možné se vyznat v milionech dokumentů?"* Jedná se o celkem převažující problém. V této souvislosti lze hovořit o prohlížení e-mailů celé organizace, pochopení desetiletí starých dokumentů nebo novin, nebo jednoduše charakterizovat rozsáhlé obory studia.

Všechny tyto problémy lze řešit pomocí modelování témat, což je metoda zpracování přirozeného jazyka, která je využitelná ke zpracování velkého množství dokumentů.

Tato práce se bude převážně soustřeďovat na modelování témat, což je součástí zpracování přirozeného jazyka. Jedná se o dolování dat z textu a využívá se k vyhledávání skrytých struktur, které se vyskytují v obsáhlých textech.

Tyto struktury se řadí do témat, které model rozpozná pomocí identifikace vzorů a frází v dokumentech a jejich podobnosti. Pomocí nalezených témat můžeme posoudit, o jaký typ textu jde, jaký má emoční náboj, nebo pozorovat procentuální

zastoupení konkrétních slov. Modelování témat má celou řadu praktických využití ve všech oblastech, kde se člověk potkává s obrovským množstvím textových dat, a dokáže značně usnadnit práci. Cílem práce je představit čtenáři různé metody modelování témat s důrazem na odlišnosti mezi nimi.

2 Zpracování přirozeného jazyka

Zpracování přirozeného jazyka (dále jen NLP) je interdisciplinární podobor počítačové vědy a lingvistiky, který se primárně zabývá učením počítačů, jak mají rozpoznat a případně manipulovat s přirozenými jazyky. V rámci lingvistiky a filozofie jazyka považujeme přirozený jazyk za jakýkoliv, který se přirozeně vyskytuje v lidských společenstvích v podobě využívání, opakování a měnění se bez jakéhokoliv vědomého plánování či rozhodování. [1] NLP kombinuje počítačovou lingvistiku
s modely statistiky, strojového učení a hlubokého učení. Dohromady tyto technologie umožňují počítačům zpracovávat přirozené jazyky ve formě textových či zvukových dat a pochopit jejich plné mínění, včetně úmyslů a sentimentu mluvčího či autora. [1] V rámci NLP lze také rozlišovat pochopení přirozeného jazyka (dále jen NLU) a generování přirozeného jazyka (dále jen NLG), kde NLU znamená schopnost počítače pochopit přirozený jazyk a NLG znamená schopnost jazyk generovat. NLG poskytuje srozumitelné vyjádření toho, co se právě stalo v počítači tak, aby to byl člověk schopný pochopit. NLU v minulosti znamenalo NLP. Jde o pochopení struktury a kontextu všech možných lidských jazyků počítačem, viz blíže. [4]

Počítačová lingvistika je vědní obor, který zkoumá počítačové aspekty lidského jazyka, přičemž NLP je inženýrská disciplína, která se snaží vybudovat výpočetní zdroje, které chápou, generují a manipulují přirozeným jazykem. Použitím vektorizace textu dochází k transformaci textu do formy, která je pochopitelná pro počítač, a následně zadává trénovací data a očekávané výstupy do algoritmů strojového učení, aby se naučili vytvářet asociace mezi patřičnými vstupy a výstupy. Stroje potom dokážou využitím statistických metod rozhodnout, jaké vlastnosti nejlépe reprezentují text, a následně predikovat výsledky pro nová data. Běžný proces NLP začíná předzpracováním textu, aby stroj dokázal porozumět textu nebo optimalizoval výkon algoritmu. Po předzpracování dochází k extrakci vlastností, což znamená, že se z textu „vydoluje“ charakteristická vlastnost, která správně reprezentuje řešenou problematiku. Následuje modelování, kdy se upravený text vkládá do algoritmů, které již vytváří požadovaná řešení. [4] NLP dokáže pochopit a analyzovat obrovská množství nestrukturovaných textů, čímž napomáhá ušetřit velké množství času, protože by jinak tyto procesy trvaly dny nebo týdny. Zároveň je také analýza objektivnější a přesnější, než kdyby byla prováděna člověkem, protože stroj

není náchylný k chybám a nepočítá s předsudky. Běžné úlohy NLP jsou vzájemně propleteny, takže je lze rozdělit do kategorií různými způsoby.

Následující rozdělení třídí známé úlohy NLP do kategorií podle toho, s jakou částí nestrukturovaného textu právě pracuje, viz blíže.

2.1 Zpracování psaného a mluveného slova

Tokenizace – tento proces spočívá v rozdělení souborů plynulého textu do individuálních slov. V rámci přirozeného jazyka, který využívá ve své psané formě latinskou abecedu a obsahuje mezery, je tento proces celkem triviální. Problém nastává s jazyky, které neobsahují oddělovače slov, jako například čínština, japonština, thaiština atd. Pomocí tokenizace je text rozdělen do menších částí a lze ho jednodušeji analyzovat, což napomáhá strojům chápat konkrétní jazyk. [4]

Optické rozpoznávání znaků – na základě obrázků reprezentujících psaný text dokážeme získat korespondující text. Využívá se pro vstup dat do strojů (např.: pas, faktury, bankovní výkaz aj.). [7]

Text-to-speech – na základě textu transformuje psané jednotky a vytvoří mluvenou reprezentaci. [5]

Rozpoznávání řeči – na základě zvukového záznamu vypíše textovou reprezentaci, což je opak text-to-speech, je mnohem složitější, např. mohou nastat problémy s přízvukem. [5]

2.2 Morfologická analýza

Stematizace – v rámci lingvistické morfologie a získávání informací je stematizace proces získání základního kmene (kořene) z odvozených slov. Algoritmus se nazývá stemmer. [6]

Lematizace – je v lingvistice procesem seskupování odvozených slov, aby bylo možné je analyzovat jako jeden předmět, jejich lemma (slovníkový tvar slova). Algoritmus se nazývá lematizátor, viz blíže. [4]

Morfologická segmentace – rozděluje slova na jednotlivé morfémy a identifikuje jejich třídu. Složitost je vysoce závislá na daném jazyku. Turečtina je například zcela nemožná, protože jedno slovo obsahuje obrovské množství forem. [7]

Part-of-speech tagging – na základě vět určuje slovní druhy daných slov. Využívá se ve vyhledávačích, chatbotech, virtuálních asistentech, analýze sentimentu, kategorizaci textu atd. [5]

2.3 Syntaktická analýza

Gramatická indukce – proces učení formální gramatiky na základě pozorovaní, podle něhož algoritmus vytvoří model, který zohledňuje vlastnosti pozorovaných objektů. To znamená, že se počítač učí jazyk na základě příkladů z jazyka. (metoda pokus-omyl, genetické algoritmy, rozpoznávání vzorů atd.) [7]

Segmentace vět – rozhodování o tom, kde věty končí a začínají. Běžně se rozhoduje na základě interpunkce, seznamu běžně používaných zkratek, a jestli následující slovo začíná velkým písmenem. Tento přístup má většinou 95 % úspěšnost. Existují však přístupy, které mají úspěšnost vyšší. [6]

Syntaktická analýza – proces analyzuje posloupnost formálních prvků textu, aby zjistil jejich gramatickou strukturu vůči formální gramatice. Jde o rozdělení vět nebo jiných slovních struktur do jejich složek, čímž vzniká tzv. derivační strom, který ukazuje syntaktickou podobnost složek, a může obsahovat i informace o jejich sémantice. Algoritmus se nazývá parser. [7]

2.4 Lexikální sémantika

Rozpoznávání pojmenovaných entit (NER) - na základě textu kategorizuje vlastní jména, jako třeba jména lidí nebo míst, a rozděluje je do typů. Rozpoznávání hlavních jmen lze provádět jednoduše na základě velkých písmen, což ale neumožňuje rozdělení daných entit do kategoriálních typů, a nemusí být zrovna přesné nebo postačující. Jde o problém, protože každý jazyk má jiná gramatická pravidla, viz blíže. [6]

Analýza sentimentu – získávání subjektivního vyjádření na základě textu. Většinou využívá online recenze k určení polarity. Velmi užitečné to je pro marketing v rámci pozorování trendů a názorů veřejnosti na sociálních sítích. [5]

Desambiguace lexikálních významů (WSD) - identifikace smyslu slova na základě kontextu. Využívá slovníkové metody, metody učení s učitelem pomocí předem známých významů slov a učení bez učitele, které funguje na základě shlukování výskytů slov. [6]

Spojování pojmenovaných entit – jde o proces přiřazování unikátních identit jmenovaným entitám. (Místo a jméno můžou být stejné, ale označují něco jiného např.: Paris jako jméno x Paris jako město) [7]

2.5 Diskurz

Koreferenční rozklad – identifikace slov v rámce velkého množství textu, které označují stejnou entitu. Anaphora rozklad je specifický příklad, kde jde o přiřazování zájmen k podstatným jménům. Součástí koreference je vyhledávání vztahů mezi jednotlivými slovními spojeními. [4]

Diskurzivní analýza – obsahuje několik různých úloh, např. syntaktická analýza celého textu nebo rozpoznávání a klasifikace mluvního aktu na základě řeči. [7]

Modelování témat – typ statistického modelování, který využívá učení bez učitele k rozpoznávání shluků nebo skupin podobných slov v textu. Shluky dělí na témata, které lze využít k pochopení nestrukturovaných dat. [12]

2.6 Složité aplikace

Automatická sumarizace – automaticky vytvoří čitelné shrnutí velkého kusu textu. [7]

Korektor gramatických chyb – zahrnuje valnou většinu problémů spojenou se všemi druhy lingvistické analýzy. Má dopad na stovky miliónů životů prakticky každý den. [7]

Strojový překlad – automaticky překládá text jednoho jazyka do jazyka jiného. Jeden z nejsložitějších problémů, který vyžaduje všechny možné druhy lidských znalostí k řešení. [7]

Natural language understanding (NLU) - přeměňuje velké části textu na formální reprezentace, které lze snadněji manipulovat pomocí počítače (predikátová logika atd.). [7]

Natural language generation (NLG) - přeměňuje informace z počítačové databáze na čitelné pro lidské porozumění, viz blíže. [4]

3 Extrakce informací

Extrakce informací (dále jen IE) je podobor NLP, který se soustřeďuje na automatickou identifikaci a extrakci strukturovaných informací z nestrukturovaných nebo semi-strukturovaných dat, což jsou převážně texty.

Běžný formát strukturovaných dat jsou čísla a text, přičemž nestrukturovaná data mají velké množství podob, jako například audio, video, email nebo výstupy z různých pozorovacích zařízení. Pro nestrukturovaná data tedy neexistuje předdefinovaný datový model, tudíž se ukládají způsobem, který nevyžaduje žádnou transformaci. Cílem IE je umožnit použití výpočetních technologií na dříve nestrukturovaná data, specificky tato data transformovat na strukturovaná, která je možné jednoduše analyzovat, prohledávat a vizualizovat. [8]

V dnešní době lidstvo generuje ohromné množství informací, které neustále roste. To znamená, že IE je velmi důležitým nástrojem ke zpracování dat. Většina organizací tedy spoléhá na IE k využití NLP algoritmů na automatizaci banální jednoduché práce.

V rámci NLP je možné se setkat s termínem získávání informací (IR), což zní celkem podobně jako extrakce informací (IE). Mezi těmito disciplínami však existuje značný rozdíl. [9] Rozdíl je patrný již od jejich názvů, protože něco získat znamená, že se snažíme dostat k datům, která již existují v případné databázi, tedy jde spíše o úlohu typu vyhledávání. Něco extrahovat v tomto smyslu znamená dolovat informace ze souboru dokumentů, které se samostatně v souboru nevyskytují.

Součástí IE je identifikace specifických entit, relací různých částí textu mezi sebou, nebo rozpoznávání důležitých částí textu, jako například pojmenovaných entit, a transformovat je do strukturovaného formátu. Můžeme tedy říci, že IE je součástí IR. [9]

V rámci datových vstupů IE uvažuje o existenci souboru dokumentů tak, že každý z dokumentů ctí určitou šablonu, tedy konkrétní dokument popisuje jednu nebo více entit nebo událostí podobně v ostatních dokumentech, ale liší se v detailech.

3.1 Typy extrahovaných struktur

Extrahované struktury lze zpravidla kategorizovat do entit, relací, zájmen popisujících entitu a struktur vyššího stupně.

Entity jsou běžné jmenné fráze a skládají se z jednoho nebo několika málo tokenů v nestrukturovaném textu. Nejznámější formou entit jsou pojmenované entity, jako například jména osob, míst a firem. Metoda Named entity recognition (NER) byla prvně zveřejněna na šesté konferenci MUC (Message Understanding Conferrence) [25] a skládala se ze tří podúloh: **ENAMEX**, tj. vlastní jména a akronymy osob, míst a organizací; **TIMEX**, tj. absolutní časové názvy; **NUMEX**, tj. peněžní a jinné numerické výrazy. [10]

Relace jsou definovány jako dvě nebo více spolu souvisící entity předem definovaným způsobem. Například "X je zaměstnancem Y" jako relace mezi osobou a firmou, "X nákazy Y" relace mezi nemocí a místem, "X stojí Y" relace mezi věcí a peněžní cenou. [10] Mezi extrakcí entit a extrakcí relací existuje jeden zásadní rozdíl. Zatímco entity jsou sekvence slov, které lze zvýraznit na zdrojovém dokumentu, relace vyjadřují asociace mezi dvěma oddělenými částmi textu, které reprezentují entity. Extrakce více než dvousměrných relací se nazývá extrakce záznamu, přičemž nejpopulárnějším typem extrakce záznamu je extrakce události. Jako příklad lze uvést teroristický útok, kde se bude extrahovat "jméno útočníka", "jména místa zasažení", "počet poznamenaných osob", "počet mrtvých osob" a "datum". [10]

Zájmena popisující entity jsou považovány za důležitou strukturu proto, že ve většině aplikací IE je potřeba danou entitu spojit s hodnotou zájmena, které tuto entitu popisuje. Hodnota daného zájmena bývá odvozena na základě kombinace slov okolo dané entity. Například při snaze vyhodnotit zájmena popisující nějaký typ restaurace je nutné získat relevantní data, většinou ve tvaru recenzí či jiných vyjádření, převážně z internetu. Následuje zjištění, zda jsou názory negativní či pozitivní. Spadá to tedy pod analýzu sentimentu. [10]

Struktury vyššího stupně jsou například seznamy nebo tabulky. V dnešní době se rozsah možností IE rozšířil i na složitější struktury. [10]

3.2 Typy nestrukturovaných dat

Nestrukturovaná data jsou většinou rozdělena na základě míry rozdrobení, na které extraktor pracuje, a podobnosti formátu skrze soubor nestrukturovaných dokumentů. [10]

Záznamy a věty reprezentují malé úryvky textu, které jsou nejpopulárnější formou textu využívanou k extrakci dat. V případě nestrukturovaných záznamů lze data chápat jako zřetězení souboru strukturovaných dat. To znamená, že během extrakce stačí pouze segmentovat text na základě hraničních mezí entit. Naopak ve větách existuje celá řada slov, které nejsou součástí entit, takže segmentace není dostačující. [10]

Paragrafy a dokumenty se skládají z velké řady záznamů a vět. Většina metod IE potřebuje zpracovat velké množství vět nebo dokonce celých dokumentů, aby vůbec dokázaly extrahovat relevantní informace. Mezi příklady patří extrakce událostí z novinových článků, extrakce částí emailů, extrakce strukturovaného životopisu z dokumentu, extrakce titulu, času a místa z různých oznámení a extrakce titulů a citací z článků. [10]

Strojem generované stránky se většinou pevně drží standardizovaných šablon. Zdrojem těchto dat jsou převážně html dokumenty generované na stránkách, které využívají databáze. Extraktor pro tyto dokumenty nazýváme wrapper. [10]

Zdroje semi-strukturovaných dat na základě specifické domény jsou zdroje nestrukturovaných textů, které mají dobře definované rámce, jako například novinové články, inzeráty, citace nebo životopisy. Ve všech předchozích příkladech existuje neformální styl a formát, který je obecně dodržován. Díky těmto pravidlům lze vytvořit slušný extrakční model v případě dostačujících dat, ale mezi jednotlivými vstupy existuje mnohem větší rozptyl než v případě strojem generovaných stránek. [10]

Nejasné zdroje jsou stránky, na nichž nelze z počátku zpozorovat nějaký druh homogenity či konzistence. V těchto případech je důležité využívat nadbytečnost získaných informací z velkého množství zdrojů. [10]

3.3 Aplikace IE

Využití extrakce informací je nespočetně mnoho. Níže jsou popsány příklady využití: byznys inteligence, vědecký výzkum, monitorování medií a zdravotnictví.

Byznys inteligence je koncept, který spojuje architekturu, úložiště dat, analytické nástroje, počítačové aplikace, a metodologie, které transformují data do využitelných a relevantních informací. Tyto informace jsou nezbytné pro zlepšení úspěšnosti byznysu. Díky moderním technologiím je možné získat strukturované informace z ohromného množství dat, které pak napomáhají rozvíjet ukazatele výkonosti a dashboardy s informacemi v reálném čase, což zaměstnancům a investorům umožňuje provádět informovaná rozhodnutí. [31]

Vědecký výzkum je složitý proces, který zabírá velké množství času. Výzkum se provádí za účelem přispět vědě pomocí systematického shromažďování, interpretace a evaluace velkého množství dat. Díky IE je dnes možné pro výzkumníky automaticky získávat reference a relevantní doporučení v podobě vědeckých článků. [31]

Monitorování medií je sběr relevantních informací pomocí průběžného sledování mediální výstupů, jako jsou noviny, televize nebo sociální media. Provádí se pomocí analýzy rozmanitého výběru mediálních platforem pro identifikaci trendů, které lze využít z různých důvodů, jako třeba politických, komerčních nebo vědeckých. Pomocí IE lze vyloučit nepotřebná data a získávat pouze data relevantní. [31]

Zdravotnictví – způsob zlepšování zdraví pomocí prevence, diagnózy a léčby. V rámci zdravotnictví si instituce, které poskytují výše zmíněné služby, udržují vlastní databázi pacientů. To znamená, že nemocnice většinou pracují s velkým množstvím dat. Extrakce informací umožňuje zdravotním institucím organizovat zdravotní záznamy na základě informací o konkrétních pacientech a jejich případných lékařských předpisech. [31]

Zmíněné aplikace neobsahují všechny možné případy, kdy by byla IE užitečná, ale je zřejmě patrné, že má smysl IE využívat v rámci jakékoliv činnosti, která za sebou zanechává velkou spoustu dat.

4 Modelování témat

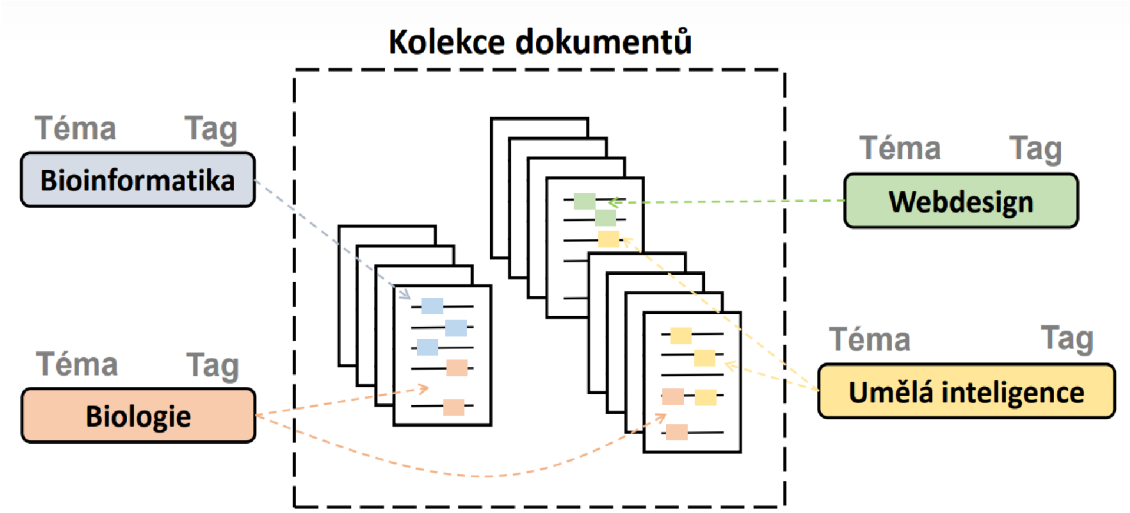
Modelování témat je jedna z nejčastěji využívaných úloh v rámci NLP. Je také technikou strojového učení bez učitele, která dokáže prozkoumávat soubory dokumentů, nacházet mezi nimi slovní a frázové vzory a automaticky provádět shlukování slov do skupin a podobných výrazů, které správně vystihují tyto soubory dokumentů. Snaží se tedy odkrývat skryté struktury v korpusu nestrukturovaných textů. [11]

Vzhledem k tomu, že modelování témat je učení bez učitele, tedy nepotřebuje předdefinovaný seznam vstupních dat provázaných s cílovými proměnnými, tak není při modelování potřebné mít před trénovaný model, lze rychle a jednoduše začít analýzu vybraných dat. [12]

Algoritmus modelování témat spočívá v odvozování abstraktních témat na základě slov v dokumentech, konkrétně dle jejich frekvence výskytu nebo vzdálenosti mezi nimi.

Existuje zde předpoklad, že každý dokument obsahuje určitou kolekci témat, proto se tedy technika podobá shlukové analýze, přičemž vytvořené shluky odpovídají nalezeným tématům a nesou určitou váhu, na jejímž základě je lze porovnávat. [13]

Pomocí získaných informací lze usoudit, o čem daný úryvek textu vypovídá.

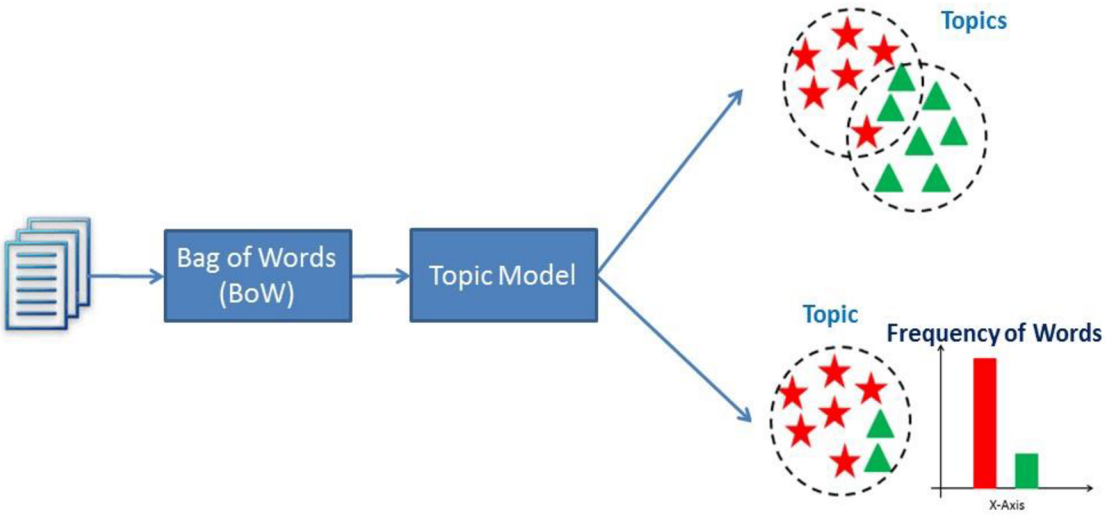


Obrázek č. 1: Reprezentace výskytu různých témat v dokumentech [26]

4.1 Proces Modelování témat

- Příprava dat
  - viz Extrakce informací **3.2** Typy nestrukturovaných dat. Data k modelování lze získat právě z těchto zdrojů.
- Předzpracování dat
  - při předzpracování dat je využívána technika NLP, konkrétně začíná se tokenizací, která rozdělí text na dílčí tokeny (slova); pokračuj se stemmatizací a lemmatizací, tedy získávají se základní tvary slov z lexikologického hlediska; následuje odstranění stop-slov, což je soubor běžně používaných slov, které neobsahují relevantní informace při modelování (v angličtině např.: a, is, the, are); následuje určení bigramů či trigramů, což jsou slovní spojení, jejichž zjištění napomáhá v přesnosti modelu.
- Vektorizace textu
  - během vektorizace jsou textová data v podobě vět nebo dokumentů transformována do numerických vektorů, které lze využívat při aplikaci algoritmů modelování témat.
- Určení počtu témat
  - vzhledem k tomu, že jde o učení bez učitele, tak není známo, jaký počet témat by byl optimální při modelování, nejlepším způsobem je změřit koherenci modelu při různém počtu témat a vybrat ten s nejlepším skóre, což však může zabrat velkou spoustu času, obzvlášť když se pracuje s velkým množstvím dat.
- Modelování
  - na základě charakteru nestrukturovaných textů jsou vybrány vhodné modely, několik z nich bude podrobně popsáno níže.
- Evaluace + optimalizace
  - při evaluaci modelů lze využít dvou metrik: koherence a relevance. Koherence měří, jak dobré mají mezi sebou jednotlivá slova v rámci dokumentu relace na základě sémantické podobnosti nebo jejich výskytu. Relevance měří, jak dobře získaná témata charakterizují dokumenty na základě jejich významu a specificity. Při evaluaci také lze využít pouze lidské oko a rozumně rozhodnout.

Na obrázku je příklad procesu modelování témat. Bag of Words je textový model, který reprezentuje text jako neuspořádanou sbírku slov, většinou v podobě matice. Využívá se běžně v rámci NLP a klasifikace dokumentů.



Obrázek č. 2: Jednoduchá reprezentace procesu modelování témat [27]

4.2 Aplikace strojového překladu (dále jen MT) v rámci technologií

Od jeho zrození před více než dvaceti lety se modelování témat stalo důležitou součástí
několika vědeckých disciplín, jako například vědecký výzkum, bioinformatika, analýza sociálních sítí, softwarové inženýrství a jiné. [12]

Vědecký výzkum

Jeden z prvních modelů MT byl popsán Papadimitriou, Raghavanem, Tamakim a Vempalou v roce 1998. Následující model s názvem Probabilistic latent semantic analysis (PLSA) byl vytvořen v roce 1999 Thomasem Hoffmanem. První generativní model Latent Dirichlet Analysis (LDA) byl představen v roce 2002 Davidem M. Bleiem, Andrewem Y. Ng, Michaelem I. Jordanem. V rámci LDA zkoumají 16.000 dokumentů z podsouboru korpusu TREC AP se 100 tématy. [12]

Thomas L Griffiths a Mark Steyvers využili modelování témat na abstraktní PNAS dataset a zkoumali trendy ve výzkumných článcích, které pak dělili na tzv. horká a studená témata. Témata nalezená pomocí LDA využitím vměšování Gibbsova

vzorkovače dokázala najít smysluplné struktury a odhalit relace mezi vědeckými články v jiných disciplínách. Nikolaos Aletras analyzoval oblast interdisplinárního výzkumu vědecké nadace pomocí algoritmů modelování témat, což pomohlo správcům vědecké nadace pochopit obsah a kontext financování portfolií. Poté se zvýšila podpora financování vědeckého výzkumu. [12]

Další z příkladů praktického využití modelování témat v rámci vědeckého výzkumu náleží Michaelu Paulovi, který pomocí LDA klasifikoval vědecké články na základě témat a jazyka. Díky tomu byly objeveny různé zajímavé statistiky a korelace mezi lingvistikou, počítačovou lingvistikou a vzděláváním. [12]

Bioinformatika

Modelování témat je užitečná metoda k porovnávání tradičních způsobů, jako je klasifikace a shlukování v bioinformatice. Napomáhá odborníkům interpretovat biologické informace. V roce 2006 studium statistického modelování biomedicínských korpusů Davidem Bleiem ukázalo, že model LDA může být využit pro odvození skrytých faktorů prostupujících biomedicínskými texty k syntetizaci a organizaci informací o složitých biologických jevech. V roce 2010 byl využit algoritmus PLSA na dataset čipů k extrakci shluků. Tento model seskupuje geny a vzorky zároveň. V roce 2013 Andrew Rider vytvořil model pro sdílení zdravotních dat a předpovídání rizik onemocnění. V tomto případě model umožňuje sdílet užitečné informace a zároveň si uchovat vlastní soukromí. [12]

John Barnett a Tommy Jakkola využili modelování témat pro analýzu genové exprese. Profilování genové exprese umožňuje náhled na vnitřní fungování buněk. Léčivé látky mají vliv na různé cesty v buňkách. V případě, že přesněji dokážeme posoudit, jaké cesty jsou ovlivňovány léčivými látkami ve specifických buňkách, umožní přesněji zacílit léky. [12]

Analýza sociálních sítí

Velké množství volně dostupných dat na sociálních webových platformách poskytuje možnost dolování informací o reálném světě. Z důvodu velké popularity a široce rozsáhlého využívání je možné tyto informace využívat k vytváření úsudků o uživatelích daných platforem a o věcech, co se dějí okolo nich.

Youngchul Cha napsal analýzu sociálních sítí pomocí modelování témat. Konkrétně využil LDA k analýze grafu vztahů ve velké sociální síti (Twitter, dnes X). [12] V roce 2009 Justin Grimmer předložil Bayesovský hierarchální model k politické analýze záměrů senátorů v novinách. [12] Debashis Naskar vytvořil SentLDA, což je generativní model identifikující sentiment uživatelů sociálních sítí na základě jejich konverzací. [12] Byl vytvořen Multi attribute latent Dirichlet allocation (MA-LDA) model. Pomocí MA-LDA lze začlenit čas a hashtag příspěvky do obyčejného LDA. Metoda dokazuje, že při uvažování nad aktuálními trendy je časový faktor důležitý. [12]

Softwarové inženýrství

S pokrokem lidstva softwarový průmysl stále jen roste. Důsledkem je obrovské množství dat, které se vyskytuje v softwarových úložištích, přičemž jsou tato data z většiny nestrukturovaná. Například zdrojové kódy, dokumentace, modelové případy, úložiště chyb atd.

Dolování z těchto nestrukturovaných úložišť může odhalit zajímavé a použitelné informace k podpoře různých úloh softwarového inženýrství. Základním předpokladem využití modelování témat je sdílení stejných charakteristik textu jako mají přirozené jazyky.

LDA-GA byl vytvořen pro softwarové inženýrství k identifikaci optimální konfigurace LDA. LDA-GA využívá jako parametry koherence dokumentů vztahujících se na stejné téma k odvození evoluce genetického přístupu (GA). [12]

4.3 Latent Semantic Analysis

LSA je teorie a metoda extrakce a zastoupení kontextuálních významů slov pomocí statistických výpočtů nad velkým korpusem textu.

Základní myšlenkou LSA je, že význam slova je tvořen postupným sbíráním zkušeností s konkrétním jazykem. Toto je sociolingvistický pohled, který je kompatibilní

s pohledem Etienna Wengera, který uvádí, že význam slova je vytvořen postupným vyjednáváním. V letech 1990 až 2000 LSA demonstrovala schopnost modelování různých kognitivních funkcí, jako třeba schopnost učit se a chápat význam slov, epizodickou paměť, sémantickou paměť, koherence diskurzu, a chápání metafor. [15]

Na základě těchto schopností byla LSA implementována jako metodologie v kvantifikaci textových dat, což vyústilo ke zlepšení získávání informací, srovnávání dokumentů, kategorizaci dokumentů a kvantifikaci dat jako součástí předzpracování dat v prediktivní analýze. Matematický základ LSA je model vektorového prostoru (dále jen VSM = Vector Space Model), což je algebraický model reprezentující dokumenty jako vektory v prostoru, kde termíny jsou dimenze tohoto prostoru. [15]

Prvním krokem LSA je v tomto prostoru vytvořit tzv. *Document-term matrix.* Jde o matici, která se skládá z *n* dokumentů a *m* termínů. (*A = n x m)*

V nejjednodušším případě by se do matice zahrnoval pouze počet výskytů termínů (např.: kolikrát se j-tý termín vyskytnul v i-té matici). Prakticky však využívání pouze počtů výskytů není úplně nejlepší, protože se nepočítá s významem každého slova v dokumentu. Například slovo "atom" informuje o tématech daného dokumentu více než slovo "dnes".

Ve většině případů tedy LSA modely místo jednoduchého počtu výskytů slov v matici využívají TF-IDF skóre (Term Frequency – Inverse Document Frequency). Měří, jak důležitý je termín v jednom dokumentu vůči všem dokumentům v korpusu.

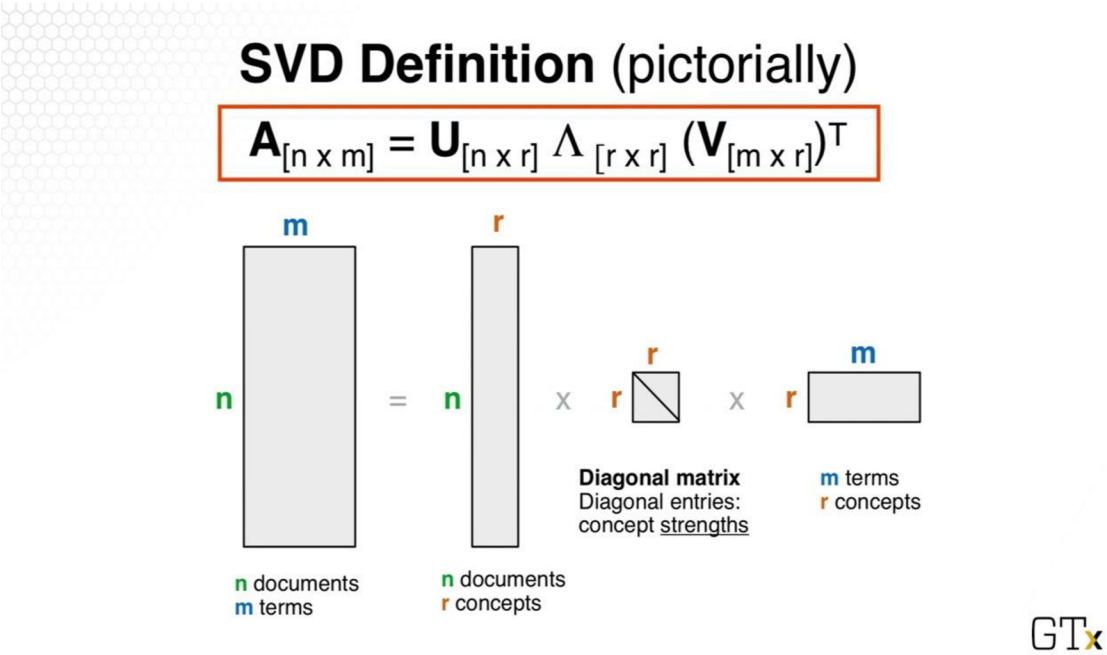
$$w_{i,j} = tf_{i,j} \times \log \frac{N}{df_j},$$

kde *w* je TF-IDF skóre, *tf* je počet výskytů termínu v dokumentu, *N* je počet všech dokumentů a *df* je počet dokumentů, který obsahuje daný termín.

Lze intuitivně usoudit, že váha termínu se snižuje s velkým výskytem a naopak.

V následujícím kroku je provedena dekompozici matice A, protože v aktuálním stavu není použitelná. Pravděpodobně je místy prořídlá, obsahuje data, která nejsou

relevantní, a spoustu šumu. Dekompozici provádíme pomocí singulárního rozkladu (dále jen SVD). [15]



Obrázek č. 3: Grafická reprezentace SVD rozkladu [28]

Na obrázku č. 3 je znázorněn SVD rozklad, kde matice U se skládá z *n* dokumentů a *r* konceptů (témat), diagonální matice Λ reprezentuje asociaci konceptů mezi sebou, matice V se skládá z *m* termínu a *r* konceptů.

Dalším krokem SVD je zmenšit rozměry pro jednoduší zpracování. Po výběru *k* největších singulárních hodnot, které jsou reprezentovány na diagonále matice Λ, a k nim korespondujících vektorů z matic U a V, je dosaženo *k* aproximace matice A s minimální chybou. Po zmenšení rozměrů vznikne daná aproximace:

$$A_k = U_k \Lambda_k V_k^T$$

Vzniká tedy sémantický prostor, kde U_k a V_k^T jsou redukované matice, které reprezentují termíny a dokumenty v menším dimenzionálním prostoru.

V tomto novém sémantickém prostoru lze spočítat kosinovou podobnost mezi určitými dvěma vektory, čímž je zjištěna podobnost dvou dokumentů, které reprezentují. [15]

4.4 Latent Dirichlet Allocation

LDA je generativní pravděpodobnostní model, který analyzuje dokumenty za účelem objevit skrytá témata, která se vyskytují napříč textovým korpusem.

V rámci modelování témat je LDA jedna z nejpoužívanějších metod. Byla představena v roce 2003, autory jsou David M. Blei, Andrew Y. Ng a Michael I. Jordan. Metoda uvažuje nad dokumenty jako nad směsicí témat a nad tématy jako nad směsicí slov. Pomocí zpětné analýzy předpokládaného generativního procesu tvorby dokumentů LDA dokáže odhalit témata, která nejlépe charakterizují soubory textů. [14]

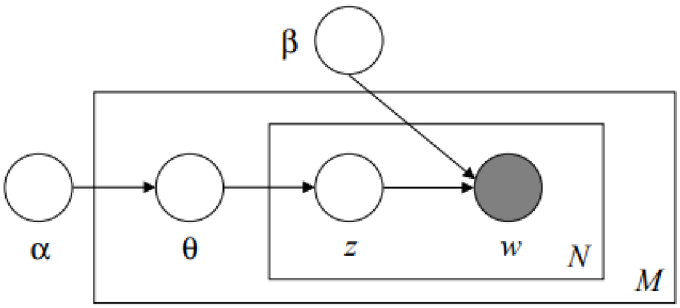
LDA předpokládá následující generativní proces pro každý dokument W v korpusu D:

1. Vyber $N \sim$ Poisson (ζ).
2. Vyber $\theta \sim$ Dir (α).
3. Pro každé slovo w_n z dokumentu W:
    a. Vyber téma $z_n \sim$ Multinomial (θ).
    b. Vyber slovo w_n z pravděpodobnosti $p\ (w_n|z_n, \beta)$, což je multinomická pravděpodobnost na základě tématu z_n.

V tomto základním modelu se předpokládá několik zjednodušujících okolnostní, z nichž budou některé odstraněny v následujících sekcích. Je uvažováno, že rozměr k Dirichletova rozložení (a tudíž rozměr témat z) je nám již znám. [14]

Je nezbytné LDA odčlenit od jednoduchého Dirichlet-multinomiálního shlukovacího modelu. Běžný shlukovací model počítá s dvouúrovňovým modelem, kde se Dirichlet využívá jednou pro korpus. Multinomiální shlukovací proměnná se vybírá jednou pro každý dokument z korpusu a soubor slov je vybírán na základě závaznosti dokumentu k shlukové proměnné. Takový model umožňuje dokumentu přidělit pouze jedno téma.

LDA je tříúrovňový model, uzel zodpovědný za přidělování témat je využíván opakovaně i v jednom dokumentu. To znamená, že pod modelem LDA se může k dokumentům vztahovat více než jen jedno téma. [14]



Obrázek č. 4: Pravděpodobnostní grafický model [14]

Podle obrázku je zřetelné, že existují tři úrovně reprezentace LDA. Grafické modely, jako na obrázku, jsou převážně využívány v rámci Bayesovských statistických modelů, kde se nazývají hierarchickými modely.

De Finettiho teorie říká, že spojené rozdělení nekonečně zaměnitelných posloupností náhodných proměnných je to samé, jako kdyby byly vybrány náhodné parametry z nějakého rozdělení a vybrané náhodné proměnné by byly navzájem nezávislé a identicky rozdělené na základě tohoto parametru. [14]

LDA počítá s tím, že slova jsou generována tématy (podmíněním rozdělením), že jsou tato témata nekonečně zaměnitelná v jednom dokumentu. Podle de Finettiho teorie pravděpodobnost posloupnosti slov a témat vypadá následovně:

$$p(W, Z) = \int p(\theta) \left(\prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n) \right) d\theta,$$

Rozdělení LDA dokumentů je získáno marginalizací (viz. výše) proměnných témat a aplikací Dirichletova rozdělení na θ. [16]

Odvození

Aby mohla být využívána LDA, je nutné nejdříve spočítat posteriorní rozdělení skrytých proměnných na základě dokumentu :

$$p(\theta, Z|W, \alpha, \beta) = \frac{p(\theta, Z, W|\alpha, \beta)}{p(W|\alpha, \beta)}.$$

Toto rozdělení je bohužel neřešitelné. Naštěstí existuje celá řada algoritmů aproximační interference, které lze využívat pro LDA, jako například Laplaceova aproximace, variační aproximace nebo Markovovy řetězce. [14]

Odvození α a β

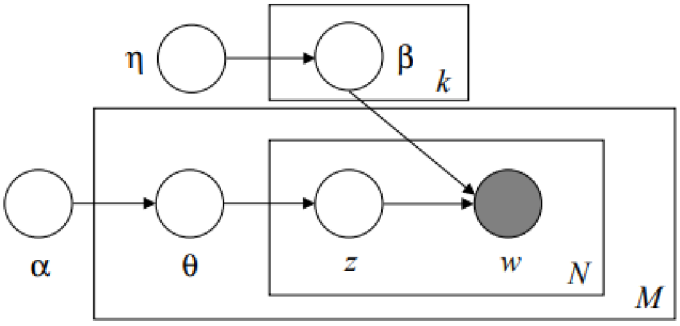
Jak bylo zmíněno dříve, pravděpodobnost p(W|α, β) je neřešitelná. Pomocí variační interference je možné získat dolní hranici logaritmické pravděpodobnosti, která je řešitelná. Tuto hranici lze maximalizovat vůči parametrům α a β. [14]

Střídáním variačního EM algoritmu, který maximalizuje dolní hranici pomocí parametrů γ a ϕ, lze najít Bayesovské odhady pro LDA model, a potom pro fixní hodnoty těchto parametrů lze maximalizovat dolní hranice podle modelových parametrů α a β. [14]

EM algoritmus probíhá následovně:

1. E-krok: V každém dokumentu najdi optimální hodnoty variačních parametrů $\{\gamma_d^*, \phi_d^*: \mathrm{d} \in D\}$.
2. M-krok: Maximalizuj nalezenou dolní hranici logaritmické pravděpodobnosti podle parametrů α a β.

Tyto dva kroky se opakují, dokud dolní hranice logaritmické pravděpodobnosti nezačne konvergovat. [14]



Obrázek č. 5: Grafický model reprezentující upravenou LDA [14]

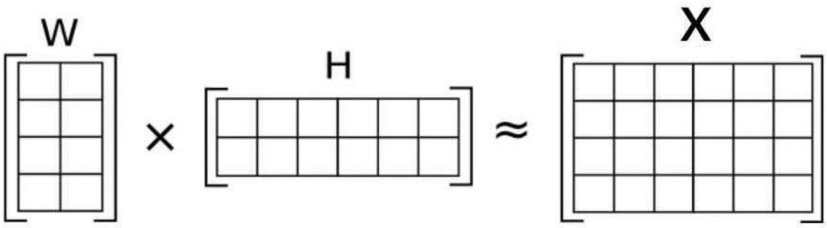
V rámci LDA je získán upravený model jako je na obrázku 5, přičemž parametr β je považován za náhodnou matici *k x V*, kde je každá řada nezávisle určena záměnným Dirichletovským rozdělením. [14]

Tudíž zbývají pouze hyperparametr η v záměnném Dirichletovu rozdělení a hyperparametr α. Hyperparametry jsou určovány na základě empirického Bayesu, tedy je využit EM k nalezení odhadů těchto parametrů s maximální pravděpodobností na základě marginální pravděpodobnosti. [14]

4.5 Non Negative Matrix Factorization

Faktorizace nezáporných matic (dále jen NMF) je matematická a výpočetní technika, která se využívá v datové analýze, strojovém učení a řade jiných vědeckých disciplín. Spočívá ve faktorizaci příslušných nezáporných dat do dvou nebo více nezáporných matic s menšími rozměry. NMF je schopná odhalovat ukryté struktury, vzory a vlastnosti v nestrukturovaných datech. [17]

Matematika používaná při NMF spočívá ve formulování a řešení optimalizačního problému faktorizace dané nezáporné matice X do dvou nezáporných matic W a H, kde W je nazývána maticí báze a H maticí koeficientů. [16][17]



Obrázek č. 6: Reprezentace NMF [17]

NMF je algoritmus učení bez učitele, který najednou provádí snížení dimenze dat a shlukování. V rámci modelování témat toto může být využito společně s TF-IDF k hledání témat napříč dokumenty. [17]

Nechť X je nezápornou maticí, je hledána k-dimenzionální aproximace nezáporných faktorů W a H. Je aproximován každý sloupec (objekt/dokument) matice X pomocí lineární kombinace k-zmenšených dimenzí, nebo také bazických vektorů W. Každý z těchto bazických vektorů je považován za shluk. Zastoupení dokumentů v těchto shlucích je zakódováno faktorem H. [16]

Platí $X \approx W \times H$,

kde pro:

$X = n \times m$, n jsou termíny a m jsou dokumenty (slova), jde tedy o *document-term* matici, v níž jsou váhy opět reprezentovány pomocí TF-IDF
$W = n \times k$, kde n jsou termíny a k je počet vybraných témat (bazických vektorů)
$H = k \times m$, kde m jsou dokumenty a k je opět počet témat. [16]

Vzhledem k tomu, že je využita pouze aproximace, nejdříve musí být nalezeny optimální hodnoty W a H. Je využita účelová funkce reprezentující chybu, která vznikne při rekonstrukci A pomocí $W \times H$ [16]:

$$\frac{1}{2}||X - WH||_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{m}\left(X_{ij} - (WH)_{ij}\right)^2$$

EM algoritmus je použit k optimalizaci W a H, protože cílem je minimalizovat účelovou funkci. Iterace probíhá alternativně mezi dvěma aktualizačními pravidly, dokud nedojde ke konvergenci [16]:

$$H_{cj} \leftarrow H_{cj} \frac{(W\boldsymbol{X})_{cj}}{(W\boldsymbol{W}\boldsymbol{H})_{cj}}$$

$$W_{ic} \leftarrow W_{ic} \frac{(\boldsymbol{X}H)_{ic}}{(\boldsymbol{W}\boldsymbol{H}H)_{ic}}$$

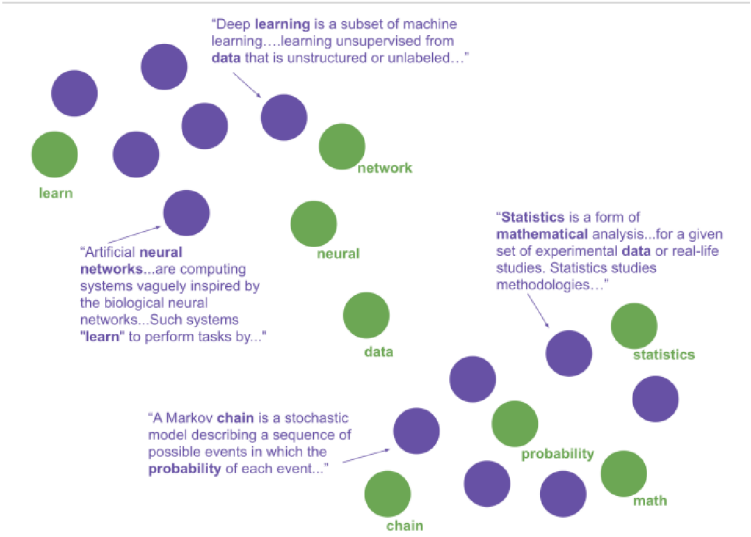
Po konvergenci jsou získány nezáporné matice W a H. V matici W lze zpozorovat rozdělení jednotlivých slov do vybraných témat a v matici H je vidět zastoupení témat v jednotlivých dokumentech. [16]

4.6 Top2Vec

Top2Vec je relativně nový algoritmus, který v roce 2020 prezentoval datový vědec Dimo Angelov. [18]

K tomu, aby top2vec mohl extrahovat témata, potřebuje vzájemně vnořené dokumenty a vektory reprezentující slova s určitými vlastnostmi. Nejvhodnější je vnoření, kde vzdálenosti mezi vektory dokumentů a vektory slov reprezentují sémantickou podobnost. Podobné dokumenty by měly být posazeny blízko k sobě ve vnořovacím prostoru a odlišné dokumenty by od sebe měly být posazeny daleko. Slova, která nejlépe popisují dané dokumenty, by měla být k nim blízko. [18]

Jestliže jsou k dispozici takto vnořená slova a dokumenty, lze vypočítat vektory témat. Prostor, který je vytvořen na základě těchto vlastností, se nazývá sémantický. Říká se, že sémantický prostor je nepřetržitá reprezentace témat. [18]



Obrázek č. 7: Sémantický prostor [18]

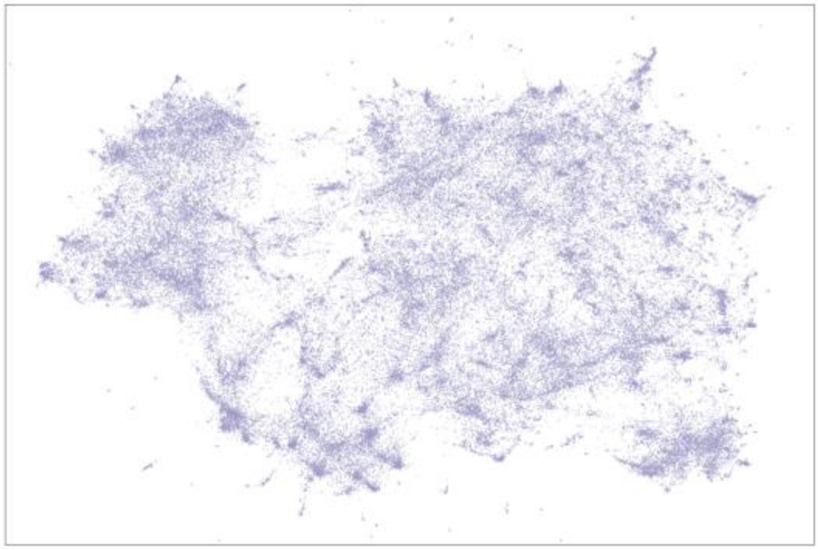
Fialové body jsou dokumenty a zelené jsou slova. Slova jsou nejblíže k dokumentům, které vystihují, a podobné dokumenty jsou blízko u sebe.

Pro zjišťování vzájemně vnořených dokumentů a vektorů top2vec se používá model doc2vec. Doc2vec je algoritmus strojového učení, který využívá umělých neuronových sítí k naučení reprezentace dokumentů jako vektorů. [18]

Top2vec pracuje s Distributed Bag of Words (dále jen DBOW) verzí modelu doc2vec, která používá vektory dokumentů k predikci slov v rámci významu v dokumentu. DBOW architektura je podobná word2vec skip-gram modelu, který využívá kontextuální slova k predikci slov okolních. DBOW místo toho k predikci využívá vektory dokumentů, což mu umožňuje se učit vektory dokumentů a vektory slov zároveň, které jsou tudíž vzájemně vnořené. [18]

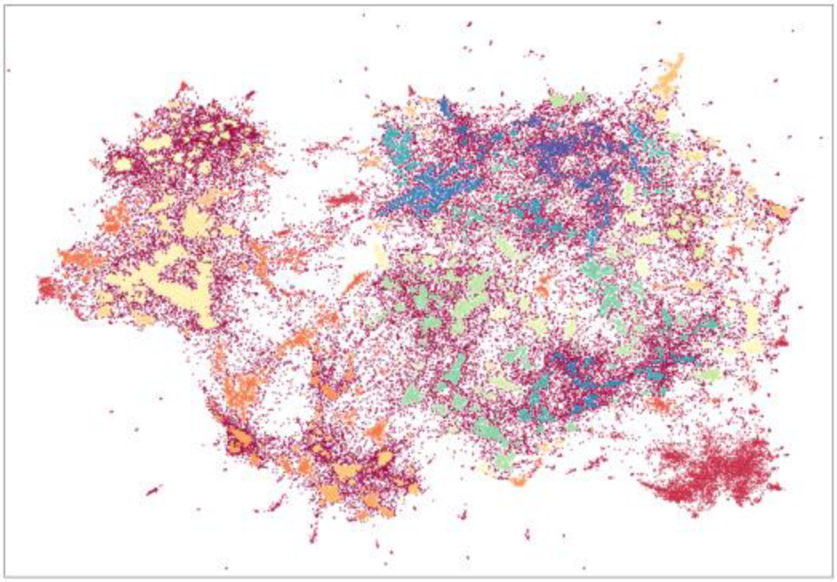
V sémantickém prostoru se můžou vyskytovat oblasti, kde je hodně dokumentů u sebe. Tyto oblasti většinou indikují, že existuje nějaké téma, které je obsaženo ve všech dokumentech. Vzhledem k tomu, že vektory dokumentů reprezentují témata dokumentů, tak lze vypočítat centroid (průměr) těchto témat. Tento centroid je právě vektorem tématu, který reprezentuje celou oblast, ze které byl vypočítán. Slova, která jsou nejblíže k tomuto vektoru, jsou slova, která ho nejlépe sémanticky vystihují. [18]

Hlavní myšlenkou top2vec je, že počet oblastí s velkou hustotou dokumentů se rovná počtu vynikajících témat. Tohle je jednoduchá možnost, jak oddělovat témata, protože lze nalézt téma pro skupinu dokumentů, které sdílejí téma společné. K vyhledávání těchto oblastí top2vec využívá HDBSCAN algoritmus, který na základě bodů v prostoru vytváří shluky a označuje odlehlé body ležící v oblastech s malou hustotou jako anomálie. [18] Problém je, že při práci s obrovským množstvím dat dochází k tomu, že data nejsou dostatečně hustá, takže by celý algoritmus trval dlouho a zabíral zbytečně moc paměti. Proto se předtím top2vec snaží snížit dimensionalitu dat pomocí algoritmu UMAP. [18]



Obrázek č. 8: Graf vytvořen pomocí algoritmu UMAP [18]

Pomocí algoritmu UMAP byl dokument, který se skládal z 300 dimenzí, vnořen do 2 dimenzí.

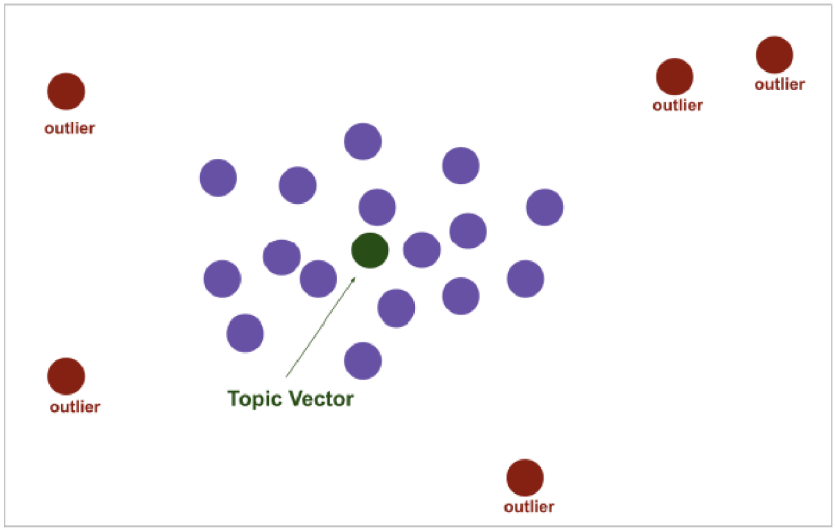


Obrázek č. 9: HDBSCAN graf [18]

Každá ze zbarvených oblastí reprezentuje husté shluky dokumentů identifikované pomocí HDBSCAN, červené body jsou šum.

Husté shluky dokumentů a dokumenty označené jako šum pomocí HDBSCAN v snížené dimenzi pomocí UMAP odpovídají umístění v originálním sémantickém prostoru. Využití těchto algoritmů lze brát jako proces, který označuje každý dokument v sémantickém prostoru jako anomálie nebo součásti shluku, do kterého patří. [18]

Pomocí těchto označení lze vypočítat vektory témat. Vektory se počítají několika způsoby z vektorů dokumentů. Nejjednodušší je vypočítat centroid, tedy aritmetický průměr všech vektorů dokumentů ve stejném shluku. Lze také využívat geometrický průměr nebo pravděpodobnosti na základě shluků. Výsledky pro všechny metody jsou skoro stejné, proto top2vec využívá metodu centroidu. [18]



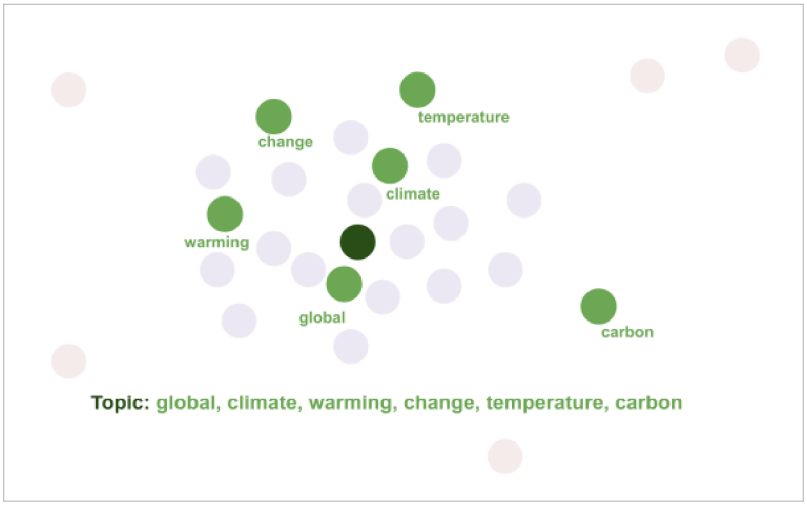
Obrázek č. 10: Metoda centroidu [18]

Vektor tématu je centroid hustého shluku fialových bodů, přičemž se červené body při výpočtu centroidu nevyužívají

Centroid je spočítán pro každou množinu vektorů dokumentů, které jsou součástí stejného shluku, čímž generují vektor tématu pro každou množinu. Počet oblastí s hustými shluky se rovná počtu prominentních témat v korpusu. [18]

V sémantickém prostoru, každý bod reprezentuje téma, které lze nejlépe sémanticky popsat pomocí nejbližších vektorů slov. Tudíž vektory slov, které jsou nejblíž k vektorům témat, je ze sémantického pohledu reprezentují nejlépe. Vzdálenost mezi každým vektorem slova a vektorem tématu ukazuje, jak je slovo sémanticky podobné tématu. Slova nejblíže k vektoru tématu lze považovat za slova, která jsou nejbližší všem dokumentům ve shluku, protože vektor tématu je centroidem tohoto shluku. [18]

Běžně využívaná slova se objevují ve většině dokumentech, takže se většinou vyskytují na stejně odlehlém místě od všech dokumentů v sémantickém prostoru. To znamená, že slova nejblíže k vektorům tématu nebudou zpravidla stop-slovy, tudíž není nutné se zbavovat stop-slov. [18]



Obrázek č. 11: Zvýraznění jednotlivých slov [18]

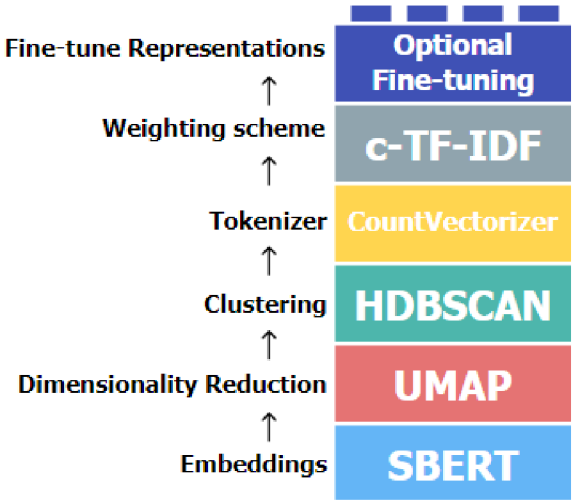
Slova vystihující téma jsou nejbližší vektory k vektoru témata.

4.7 BERTopic

BERTopic je jedním z nejmodernějších algoritmů modelování témat, který byl představen Maartenem Grootendorstem v roce 2019. [19] Název BERTopic je zkratka pro BERT a slovo „topic“, které odkazuje na záměr algoritmu, tedy extrakci smysluplných témat z nestrukturovaných dat.

Model využívá BERT k vnořování slov, což transformuje slova do kontextualizovaných reprezentací pomocí velkého korpusu dat využitím hlubokého učení neuronových sítí. Díky vnořeným slovům lze získat bohaté informace o významech každého slova z dokumentů, čímž dovolují BERTopic nacházet sémantiku a kontextuální relace mezi slovy. Tím je získán mnohem přesnější model témat. Kromě toho BERTopic také využívá c-TF-IDF k filtrování nepodstatných slov a klade důraz na slova, která jsou k modelování témat důležitá. Tato metodika přiřazuje váhy tříd ke slovům na základě výpočtu TF-IDF. Napomáhá tak BERTopic upřednostňovat slova s více informacemi. [19]

BERTopic lze chápat jako sérii kroků pro modelování témat.



Obrázek č. 12: Výchozí model BERTopic [21]

Tyto kroky jsou převážně výchozí, ale v rámci BERTopic existuje modularita. Každý krok je pečlivě vybírán tak, aby byly všechny navzájem nezávislé. Například tokenizace není ovlivněna vnořováním. [21]

Vnořování

BERTopic začne tím, že transformuje dokumenty do jejich numerických reprezentací pomocí vnořování slov. Výchozí metoda využívá Sentence-BERT (dále jen SBERT), což je zlepšená modifikace originální před trénované BERT sítě. Bidirectional Encoder Representations from Transformers neboli BERT, byl zveřejněn v roce 2018 Jacobem Devlinem, Ming-Wei Changem, Kentonem Lee, Kristinou Toutanovou. [20]

Z názvu lze odvodit, že se podobá transformátorům, hlavní rozdíl je však v tom, že BERT využívá pouze kodér bez dekodéru.

Cílem BERT je naučení se jazyka pomocí vytváření významných representací textů. Navíc je také obousměrný, takže kodér využívá informací z obou stran. BERT také využívá WordPiece vnořování, které obsahuje slovník zhruba 30.000 slov. [20] BERT dokáže extrahovat hluboko skryté sémantické informace z vět. Je trénován na velkém množství nestrukturovaných textových dat učením bez učitele pomocí dvou hlavních úloh, Masked Language Modeling (dále jen MLM) a Next Sentence Prediction (dále jen NSP). [20]

Při provádění úlohy MLM je využíván obousměrný přístupu. Během běžných přístupů zleva doprava, nebo zprava doleva, se snažíme predikovat následující slovo a při využívání obousměrného přístupu vzniká možnost pohledu na následující slovo, což je z praktického hlediska podvod. [20] Metoda MLM nahrazuje zhruba 15 % slov slovy [MASK], což nám umožňuje využívání obousměrného přístupu bez podvádění. Toto donutí model, aby se naučil predikovat slova na základě významů okolních slov, a dokáže naučit BERT obousměrnou reprezentaci jazyka. [20]

Metoda NSP se snaží naučit predikci, zda jsou v textu dvě věty po sobě jdoucí nebo ne, což napomáhá k pochopení relací na úrovni vět. Mějme dvě věty, X a Y. Při trénování jde o to, aby bylo zjištěno, zda věta Y následuje správně za větou X. Polovinu času vybíráme Y správně a druhou polovinu je nahrazeno náhodnou větou z trénovacího korpusu. [20]

Po natrénování lze BERT upravit na specifických menších datasetech. Během upravování se parametry přizpůsobí konkrétní úloze a vzniklá vnoření slov jsou využita jako vstup. [20]

Bylo prokázáno, že BERT sám o sobě není optimální pro úlohy typu podobnosti vět, které využívají standartní metriky podobnosti jako euklidovskou vzdálenost nebo kosinovou podobnost. Při výpočtu podobnosti mezi 10.000 větami by BERT strávil zhruba 65 hodin, protože by to vyžadovalo zhruba 50 milionu výpočtů. Tento problém vyřešili v roce 2019 Nils Reimers a Iryna Gurevych. Dokázali BERT modifikovat, aby používal siamskou neurální síť k odvozování významných vnořování, a zredukovali čas výpočtu z 65 hodin na 5 vteřin. [22]

Vzniklý model byl pojmenován Sentence-BERT (SBERT), a je využíván ve výchozím nastavení modelu BERTopic.

Redukce dimenzí

Redukce dimenzí je důležitým krokem algoritmu, protože napomáhá v efektivní a přesné identifikaci témat ve velkém množství nestrukturovaných dat. Ve většině případech jsou datasety obsáhle s vysokým počtem dimenzí, kde je každé slovo reprezentováno vektorem. S vyšším počtem unikátních slov roste dimenze celého výpočetního prostoru. Ke zmenšení komplexity BERTopic využívá UMAP, který dokáže zmenšit dimenze dat a zároveň zachovat struktury, které se v datech vyskytují. Algoritmus UMAP byl zmíněn dříve a funguje na stejném principu jako při modelu top2vec. [21]

Shlukování

Upravená data shlukujeme za účelem seskupování podobných dokumentů na základě jejich sémantické významnosti. Proces hraje hlavní roli v přesnosti určování reprezentací témat správně, takže efektivita shlukovacího algoritmu má velký vliv na kvalitu konečného výsledku. Jako výchozí algoritmus BERTopic využívá HDBSCAN, který provádí shlukování na základě hustoty. Vzniklé shluky lze využívat při identifikaci témat v souborech dokumentů, což umožní náhled do obsahu dokumentů a jakýchkoliv skrytých struktur v datech. [21]

Tokenizér

Před tvorbou schématu vah je nutné vybrat způsob, který umožní zachovat modularitu při vyhledávání témat. Při používání modelu HDBSCAN mohou mít shluky odlišné tvary a úrovně hustoty, což znamená, že reprezentace témat na základě centroidů není nejlepší. V tomto případě je ideální vybrat techniku, která nevyužívá žádných předpokladů o strukturách shluků, jako například metodu bag-of-words. Aby mohla být aplikována metodu bag-of-words na úrovni shluků, všechny dokumenty v rámci jednoho shluku musí být brány jako jeden dokument, čehož dosáhneme zřetězením. Poté stačí spočítat frekvenci slov v každém shluku. [21]

Protože kvalita reprezentace témat je základem chápání různých vzorů, interpretace témat a komunikování výsledků, je potřeba vybrat nejlepší metodu pro konkrétní dataset. Výchozí metoda BERTopic je CountVectorizer. CountVectorizer je metoda v knihovně pythonu scikit-learn, která konvertuje soubory textových dokumentů do

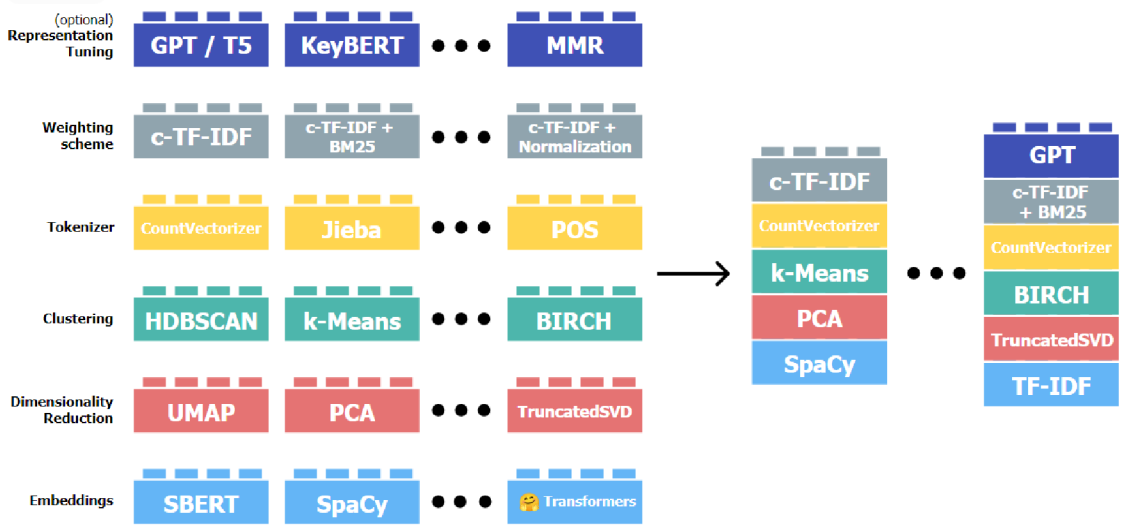
matice počtů tokenů za účelem extrakce vlastností textu. Každý dokument je reprezentován vektorem, který počítá frekvenci každého slova v dokumentu. [23]

Schéma vah

Na základě generované bag-of-words reprezentace je snaha zjistit, co rozlišuje shluky jeden od druhého, jestli existují slova, která jsou běžná pro jeden shluk, ale ne pro ostatní. BERTopic to řeší pomocí algoritmu c-TF-IDF. [21] Hlavní myšlenka c-TF-IDF je přiřadit ke každému shluku právě jedno téma. TF-IDF je numerická statistika, která byla popsána již dříve v rámci LSA. BERTopic využívá modifikaci c-TF-IDF, která se snaží o vyhodnocení důležitosti jednotlivých slov k tématu. Aby to bylo možné, všechny dokumenty ve shluku jsou zřetězeny a přistupuje se k nim jako k jednomu dokumentu. [19]

Ladění reprezentace

Pomocí c-TF-IDF je získána přesná reprezentace témat v podobě seznamu slov, která popisují soubory dokumentů. Vzhledem k tomu, že se NLP stále vyvíjí, tak lze využívat různých nových technologií pro další vyhlazování modelu. Na c-TF-IDF lze pohlížet jako na potenciální témata, která obsahují klíčová slova a reprezentativní dokumenty. Tyto dokumenty jsou velkou výhodou při dolaďování modelu, protože umožňují provádět výpočty na menších souborech dokumentů. Techniky jako KeyBERT jsou již součástí BERTopic pro jednoduché využívání a zkoumání. [21]



Obrázek č. 13: Modularita algoritmu BERTopic [21]

5 Implementace algoritmů

Pro implementaci algoritmů modelování témat je nutné nejprve vytvořit prostředí, ve kterém je možné aplikovat již existující balíčky. V této práci je zásadně využíván programovací jazyk Python. Hlavní knihovna, pomocí níž jsou vytvářeny modely, je v tomto případě Gensim. Gensim se běžně využívá pro algoritmy modelování témat, v tomto případě LSA, LDA, a NMF.

Algoritmy top2vec a BERTopic mají svoje vlastní knihovny.

Před samotnou činností je nezbytné zajistit všechny potřebné nástroje. V rámci Pythonu se různé knihovny a moduly volají pomocí funkce **import**.

```
import re
import numpy as np
import pandas as pd
import gensim
import gensim.corpora as corpora
from gensim.utils import simple_preprocess
from nltk.corpus import stopwords
from gensim.models import CoherenceModel
import spacy
import pyLDAvis
import pyLDAvis.gensim_models
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
```

Kód č. 1: Import potřebných knihoven a modulů [Vlastní zpracování]

Může se stát, že některé balíčky nelze importovat, protože nejsou součástí základní verze Pythonu, která je nainstalována. V tomto případě je využíván **pip**, nástroj pro stažení požadujících modulů.

```
pip install spacy
pip install pyLDAvis
```

Kód č. 2: Ukázka nástroje pip [Vlastní zpracování]

Po získání potřebných modulů je započata jedna z nejdůležitějších úloh NLP, předzpracování textu.

5.1 Předzpracování textu

V rámci NLP je nezbytné provádět několik kroků navíc oproti obyčejným metodám strojového učení, protože počítač sám o sobě nerozumí psanému textu. Tudíž největším problémem je, jak docílit, aby počítač textu porozuměl. Naštěstí existuje knihovna spaCy, která obsahuje již před trénované pipeline. Stačí je pouze zavolat.

```
nlp=spacy.load('en_core_web_sm',disable=['parser', 'ner'])
```

Kód č. 3: Zavolání předem natrénovanou pipeline [Vlastní zpracování]

Proměnná nlp tedy obsahuje pipeline trénovaný pro anglický jazyk, přičemž atribut disable vypíná komponenty, které nejsou v tomto případě potřebné.

Běžný text často obsahuje spoustu slov, která nejsou využitelná k identifikaci skrytých vzorů, v tomto případě konkrétních témat. Existence těchto slov v datech může způsobit nežádoucí následky, proto je lepší se jich před zpracováním dat zbavit.

```
nltk.download('stopwords')
stop_words = stopwords.words('english')
stop_words.extend(['from', 'subject', 're', 'edu', 'use','for','ax' ])
def remove_stopwords(texts):
    """
    Funkce filtruje stop slova z textu

    Parametry
    ---------
    texts: zpracovávaná data
    """
    return [[word for word in simple_preprocess(str(doc)) if word not in
stop_words] for doc in texts]
```

Kód č. 4: Definice stop-slov + funkce jejich odstranění [Vlastní zpracování]

Knihovna nltk obsahuje běžně vyskytující se stop-slova, která nejsou užitečná pro NLP, a lze je stáhnout. Poté je vytvořena proměnná stop_words, kde je specifikováno, s jakým jazykem se právě pracuje. Tuto proměnnou lze také rozšířit o konkrétní slova pomocí příkazu extend, což jsou většinou slova, která nejsou užitečná v konkrétním korpusu dat. Definovaná funkce remove_stopwords využívá funkce simple_preprocess z knihovny Gensim k postupnému filtrování stop-slov z dat.

Pro následující práci je již potřeba konkrétní korpus dat. V rámci knihovny sklearn existuje několik datasetů, které lze jednoduše zavolat.

```
from sklearn.datasets import fetch_20newsgroups
newsgroups_train = fetch_20newsgroups(subset='all')
data_train = newsgroups_train.data
```

Kód č. 5: Stažení využívaného datasetu [Vlastní zpracování]

Získaná data jsou uložena do proměnné data_train v použitelné formě.

Nyní jsou k dispozic všechny nástroje pro následující krok předzpracování, **tokenizaci**.

```
def gen_words(texts):
    """
    Úprava korpusu
    Parametry
    ---------
    texts:  korpus
    Return
    ------
    final: seznam, který obsahuje slova s malými písmeny a bez uvozovek
    """
    final = []
    for text in texts:
        new = gensim.utils.simple_preprocess(text, deacc=True)
        final.append(new)
    return final
zprac_data = gen_words(data_train)
```

Kód č. 6: Funkce tokenizace [Vlastní zpracování]

Definovaná funkce gen_words využívá funkce simple_preprocess a cyklus for k tokenizaci slov v používaném datasetu. Upravený korpus je uložen do proměnné zprac_data.

Díky tomu, že byla provedena tokenizace, lze vyhledávat a utvářet v dokumentech **n-gramy**, konkrétně bigramy a trigramy.

```
bigram = gensim.models.Phrases(zprac_data, min_count=5, threshold=100)
trigram = gensim.models.Phrases(bigram[zprac_data], threshold=100)
bigram_mod = gensim.models.phrases.Phraser(bigram)
trigram_mod = gensim.models.phrases.Phraser(trigram)
def create_bigrams(texts):
    """
    Funkce vytváří bigramy

    Parametry
```

```
    ---------
    texts: zpracovávaná data
    """
    return [bigram_mod[doc] for doc in texts]
def create_trigrams(texts):
    """
    Funkce vytváří trigramy

    Parametry
    ---------
    texts: zpracovávaná data
    """
    return [trigram_mod[bigram_mod[doc]] for doc in texts]
```

Kód č. 7: Funkce pro vytvoření bigramů a trigramů [Vlastní zpracování]

Knihovna Gensim obsahuje model Phrases, který automaticky vyhledává běžné fráze neboli výrazy složené z více slov, tedy n-gramy. Pro vytvoření bigramu je tento model aplikován na zprac_data, přičemž min_count = 5 indikuje, že model bude ignorovat všechny bigramy, které se v datech vyskytují méně než pětkrát. Threshold pak reprezentuje práh skóre fráze. Trigramy jsou vytvořeny opětovným využitím proměnné bigram. Phraser je využit k další optimalizaci, více v dokumentaci Gensim. [29]

V této fázi jsou k dispozici tokenizovaná data a lze vyhledávat běžně využívané fráze napříč dokumenty v korpusu. Následujícím krokem předzpracování je **lemmatizace**.

```
def lemmatization(text_data, allowed_postags=["NOUN", "ADJ", "VERB"]):
    """
    Lemmatizace korpusu

    Parametry
    ---------
    text_data: korpus dokumentů
    allowed_postags: seznam povolených slovních druhů

    Return
    ------
    documents: lemmatizovaný seznam povolených slov
    """
    documents = []
    for sent in text_data:
        doc = nlp(" ".join(sent))
        documents.append([token.lemma_ for token in doc if token.pos_ in
```

```
allowed_postags])
    return documents
data_wo_stopwords = remove_stopwords(zprac_data)
data_bigrams = create_bigrams(data_wo_stopwords)
data_lemmatized = lemmatization(data_bigrams, allowed_postags=[ 'NOUN', 'ADJ',
'VERB'])
```

Kód č. 8: Funkce lemmatizace [Vlastní zpracování]

Funkce lemmatization využívá již definované pipeline nlp, lingvistické charakteristiky knihovny spaCy token.lemma_ a token.pos při lemmatizaci příslušných dat. [30]

Současně jsou připraveny všechny potřebné nástroje ke konečné úpravě textových dat. Pomocí funkce remove_stopwords do proměnné data_wo_stopwords je uložen tokenizovaný dataset bez stop-slov. Následně je využita funkce create_bigrams pro vyhledání existujících bigramů v datasetu bez stop-slov. Nakonec jsou do proměnné data_lemmatized uložena již lematizovaná data, složená pouze z podstatných jmen, přídavných jmen a sloves.

Ještě předtím, než jsou spuštěny konkrétní metody modelování témat, je nezbytné vytvořit slovník, který mapuje vztahy mezi slovy a jejich identitami vyjádřenými v celých číslech.

```
id2word = corpora.Dictionary(data_lemmatized)
corpus = []
for text in data_lemmatized:
    new = id2word.doc2bow(text)
    corpus.append(new)
```

Kód č. 9: Slovník id2word + vektorizovaný corpus [Vlastní zpracování]

Do proměnné id2word jsou uložena lematizovaná data ve formě corpora.Dictionary (slovníku), což je modul v Gensim, který vytváří právě požadované mapování slov a jejich číselných identit. Poté pomocí funkce doc2bow konvertujeme slovníky do bag-of-words formátu čili seznamu dvojic obsahujících identitu konkrétního tokenu(slova) a jejich počet výskytu v korpusu dokumentů. Nyní lze spustit modelování témat.

5.2 Latent Semantic Analysis

Modul, který je zde využíván, je součástí knihovny Gensim.

```
lsi_model = gensim.models.lsimodel.LsiModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=20,
                                            chunksize=1000)
lsi_model.print_topics()
```

Kód č. 10: Model LSI [Vlastní zpracování]

Atributy modelu corpus a id2word jsou již definovány jako součást předzpracování textu, num_topics je vybraný počet témat a chunksize je počet dokumentů, které model analyzuje najednou. Příkaz lsi_model.print_topics() má následující výstup.

```
[(0,
  '0.264*"file" + 0.218*"say" + 0.186*"get" + 0.178*"go" + 0.172*"image" +
0.151*"people" + 0.146*"know" + 0.145*"make" + 0.145*"see" + 0.135*"use"'),
 (1,
  '-0.397*"file" + 0.351*"say" + -0.294*"image" + 0.253*"go" + 0.179*"peopl
e" + 0.170*"think" + 0.163*"know" + 0.136*"come" + -0.131*"program" + -0.12
6*"format"'),
 (2,
  '-0.376*"image" + -0.338*"file" + -0.211*"say" + 0.196*"com" + -0.193*"jp
eg" + 0.181*"system" + -0.159*"go" + 0.145*"use" + -0.135*"color" + 0.132*"
server"'),
.
.
.
(17,
  '0.211*"widget" + -0.207*"say" + 0.174*"value" + -0.160*"openwindow" + 0.
154*"application" + 0.151*"converter" + 0.142*"return" + 0.141*"resource" +
-0.134*"version" + 0.132*"dos_do"'),
 (18,
  '0.258*"launch" + -0.235*"entry" + 0.192*"bit" + 0.189*"space" + 0.167*"s
atellite" + -0.166*"planet" + -0.155*"com" + 0.140*"output" + 0.134*"key" +
0.120*"openwindow"'),
 (19,
  '-0.278*"entry" + -0.270*"launch" + 0.221*"output" + -0.191*"space" + -0.
178*"satellite" + 0.148*"open" + 0.142*"adl" + 0.141*"printf" + 0.131*"outp
ut_oname" + 0.124*"war"')]
```

Kód č. 11: Output modelu LSI [Vlastní zpracování]

Ukáže 10 slov, která jsou nejvíce zastoupena v jednotlivých tématech, přičemž do závorky nakonec příkazu lze specifikovat počet zobrazených témat. Ovšem při výběru počtu témat nelze dopředu vědět, zda je vybrané množství optimální. Model lze evaluovat pomocí koherence, což je číselná hodnota, která reprezentuje soudržnost textu.

```
coherence_model_lsi = CoherenceModel(model=lsi_model, texts=data_lemmatized,
dictionary=id2word, coherence='c_v')
coherence_lsi = coherence_model_lsi.get_coherence()
coherence_lsi
```

Kód č. 12: Výpočet koherence modelu LSI [Vlastní zpracování]

Pomocí příkazu CoherenceModel knihovny Gensim lze spočítat koherenci daného modelu, přičemž parametr coherence rozhoduje o metodě výpočtu. Zde bylo zvoleno c_v, viz blíže [29]. Vzhledem k tomu, že lze vyhodnotit model, tak je pomocí opětovného modelování vybrán optimální počet témat podle nejvyšší koherence.

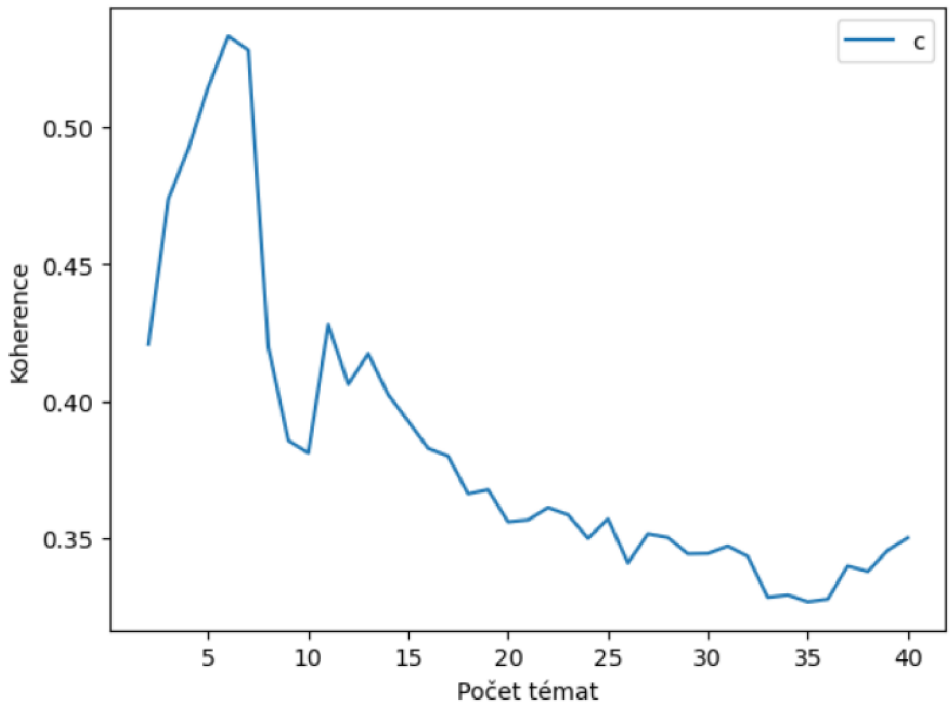
```
def topics_by_coherence(id2word, corpus, data_clean, stop, start, step):
    """
    Spočítá hodnotu koherence modelu LSA s různými počty vybraných témat
    Parametry
    ---------
    id2word : Gensim dictionary
    corpus : Gensim corpus
    data_clean : Seznam předzpracovaných dat
    start: Minimální počet témat
    stop : Maximální počet témat
    step : Velikost kroku
    Return
    ------
    model_list : Seznam LSA modelů
    coherence_values : Koherence jednotlivých modelů na základě počtu témat
    """
    coherence_values = []
    model_list = []
    for k in range(start, stop, step):
        model = gensim.models.lsimodel.LsiModel(corpus=corpus,
                                                id2word=id2word,
                                                num_topics=k,
                                                chunksize=1000)
        model_list.append(model)
        coherencemodel = CoherenceModel(model=model, texts=data_clean,
dictionary=id2word, coherence='c_v')
        coherence_values.append(coherencemodel.get_coherence())
    return model_list, coherence_values
```

Kód č. 13: Funkce seznamu koherencí [Vlastní zpracování]

Funkce topics_by_coherence vytvoří seznam hodnot koherencí pro jednotlivé modely podle počtu témat. Následovně lze tuto funkci přeměnit na spojnicový graf, který zřetelně ukáže optimální počet témat.

```
def plot_coherence(id2word, corpus, data_clean, start, stop, step):
    """
    Funkce vykreslí spojnicový graf, který zobrazuje velikost soudržnosti textu
podle počtu vybraných témat
    Parametry
    ---------
    id2word : Gensim dictionary
    corpus : Gensim corpus
    data_clean : Seznam předzpracovaných dat
    start: Minimální počet témat
    stop : Maximální počet témat
    step : Velikost kroku
    """
    model_list, coherence_values = topics_by_coherence(id2word, corpus, data_clean,
                                                       stop, start, step)
    x = range(start, stop)
    plt.plot(x, coherence_values)
    plt.xlabel("Počet témat")
    plt.ylabel("Koherence")
    plt.legend(("coherence_values"), loc='best')
    plt.show()
plot_coherence(id2word, corpus, data_lemmatized, 2, 41, 1)
```

Kód č. 14: Funkce grafu koherencí [Vlastní zpracování]



Obrázek č. 14: Graf koherencí modelů LSI podle počtu témat [Vlastní zpracování]

Z obrázku je patrné, že model s nejvyšší koherencí, je model s šesti tématy.

```
lsi_model = gensim.models.lsimodel.LsiModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=6,
                                            chunksize=1000)
lsi_model.print_topics()
```

Kód č. 15: Model LSI s optimálním počtem témat [Vlastní zpracování]

Příkaz lsi_model.print_topics() má následující výstup:

```
[(0,
  '0.265*"file" + 0.220*"say" + 0.185*"get" + 0.178*"go" + 0.172*"image" +
0.151*"people" + 0.146*"know" + 0.145*"make" + 0.145*"see" + 0.135*"use"'),
 (1,
  '-0.400*"file" + 0.352*"say" + -0.295*"image" + 0.256*"go" + 0.174*"peopl
e" + 0.169*"think" + 0.165*"know" + 0.139*"come" + -0.128*"program" + -0.12
7*"format"'),
 (2,
  '-0.373*"file" + -0.340*"image" + -0.198*"jpeg" + 0.184*"system" + -0.176
*"say" + 0.159*"server" + 0.147*"com" + -0.141*"go" + 0.139*"use" + 0.137*"
dos_do"'),
 (3,
  '0.479*"file" + -0.414*"image" + 0.331*"entry" + -0.156*"color" + -0.129*
"jpeg" + 0.113*"output" + -0.112*"display" + -0.108*"format" + -0.100*"dos_
do" + -0.100*"version"'),
 (4,
  '-0.422*"dos_do" + -0.229*"window" + 0.216*"internet" + 0.197*"privacy" +
-0.179*"entry" + -0.172*"do" + 0.148*"anonymous" + -0.133*"support" + 0.131
*"image" + 0.125*"mail"'),
 (5,
  '0.375*"wire" + -0.291*"dos_do" + 0.185*"circuit" + 0.184*"wiring" + -0.1
75*"file" + 0.155*"ground" + 0.154*"outlet" + 0.145*"neutral" + 0.137*"use"
+ -0.132*"internet"')]
```

Kód č. 16: Output kódu č. 15 [Vlastní zpracování]

Zda má náš model koherenci zhruba podle grafu, potom lze opět jednoduše ověřit pomocí coherence_lsi.

```
0.5331136840602807
```

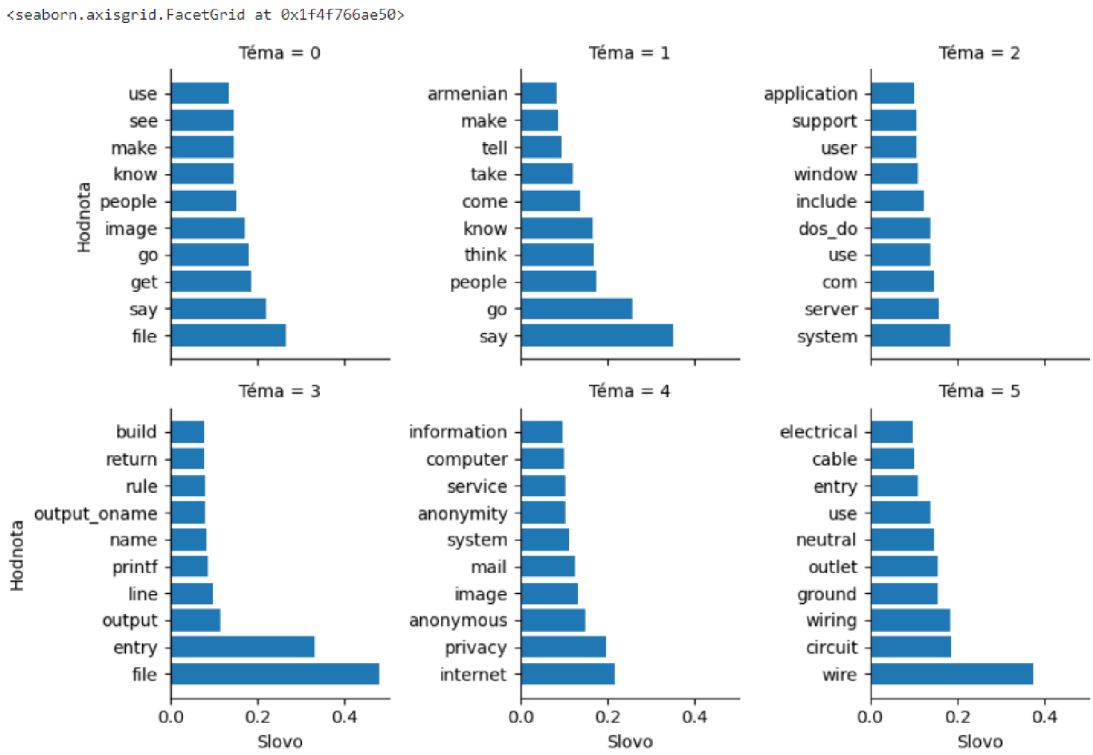
Kód č. 17: Output koherence optimálního modelu LSI [Vlastní zpracování]

Vzhledem k tomu, že modely Gensim vrací také hodnoty zastoupení určitých slov v dokumentech, tak by bylo dobré si toto nějak vizualizovat. V tomto případě je vytvořen Pandas DataFrame, který bude obsahovat sloupce Téma, Slovo a Hodnota, atributy získané z našeho modelu LSI. Dataframe je vizualizován pomocí knihovny seaborn jako sloupcové grafy.

```
num_w = 10
topic_w = pd.DataFrame({})
for i, topic in enumerate(lsi_model.get_topics()):
    top_feature_ids = topic.argsort()[-num_w:][::-1]
    feature_values = topic[top_feature_ids]
```

```
    words = [id2word[id] for id in top_feature_ids]
    topic_df = pd.DataFrame({'Hodnota': feature_values, 'Slovo': words, 'Téma': i})
    topic_w = pd.concat([topic_w, topic_df], ignore_index=True)
graf = sns.FacetGrid(topic_w, col="Téma", col_wrap=3, sharey=False)
graf.map(plt.barh, "Slovo", "Hodnota")
```

Kód č. 18: Sloupcový graf zastoupení slov v tématech. [Vlastní zpracování]



Obrázek č. 15: Sloupcový graf zastoupení slov v tématech. [Vlastní zpracování]

5.3 Latent Dirichlet Allocation

Pro model LDA je opět využita knihovna Gensim.

```
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=20,
                                            chunksize=1000,
                                            )
lda_model.print_topics(20)
```

Kód č. 19: Model LDA [Vlastní zpracování]

Atributy pro LDA model jsou identické jako u modelu LSI. Modely LDA a LSI využívají stejných atributů, avšak jejich výsledky jsou rozdílné.

```
[(0,
  '0.015*"get" + 0.012*"font" + 0.012*"line" + 0.010*"price" + 0.010*"machi
ne" + 0.009*"problem" + 0.009*"run" + 0.009*"mhz" + 0.008*"need" + 0.008*"d
o"'),
 (1,
  '0.026*"team" + 0.023*"player" + 0.019*"year" + 0.016*"win" + 0.014*"good
" + 0.013*"play" + 0.012*"goal" + 0.011*"game" + 0.009*"think" + 0.009*"say
"'),
```

```
 (2,
  '0.023*"space" + 0.009*"earth" + 0.009*"launch" + 0.008*"orbit" + 0.008*"
planet" + 0.007*"year" + 0.007*"mission" + 0.007*"system" + 0.006*"project"
+ 0.006*"cost"'),
.
.
.
(17,
  '0.055*"game" + 0.022*"team" + 0.020*"go" + 0.020*"line" + 0.019*"play" +
0.017*"fan" + 0.014*"get" + 0.014*"year" + 0.013*"season" + 0.011*"hockey"'
),
 (18,
  '0.027*"fire" + 0.012*"koresh" + 0.011*"compound" + 0.011*"write" + 0.010
*"child" + 0.009*"tank" + 0.008*"start" + 0.008*"say" + 0.008*"line" + 0.00
7*"kill"'),
 (19,
  '0.014*"drug" + 0.011*"study" + 0.010*"cause" + 0.009*"disease" + 0.009*"
patient" + 0.009*"medical" + 0.008*"treatment" + 0.008*"doctor" + 0.007*"fo
od" + 0.007*"health"')]
```

Kód č. 20: Output kódu č. 19 [Vlastní zpracování]

Důvodem rozdílnosti je zcela jiný přístup k řešení problémů u obou algoritmů. LSI se soustředí na získání jednotlivých témat pomocí dekompozice SVD a LDA je pravděpodobnostní model, který se zabývá šancí výskytu jednotlivých slov v dokumentech. Při počítání koherence v modelu LDA stačí pouze pozměnit model atribut ve funkce CoherenceModel.

```
coherence_model_lda = CoherenceModel(model=lda_model, texts=data_lemmatized,
dictionary=id2word, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
coherence_lda
```

Kód č. 21: Koherence modelu LDA [Vlastní zpracování]

To znamená, že lze obdobně provádět funkce topics_by_coherence_LDA a plot_coherence_LDA, kde jsou zaměněny relevantní části modelů.

```
def topics_by_coherence_LDA(id2word, corpus, data_clean, stop, start, step):
    """
    Spočítá hodnotu koherence modelu LDA s různými počty vybraných témat

    Parametry
    ---------
    id2word : Gensim dictionary
    corpus : Gensim corpus
    data_clean : Seznam předzpracovaných dat
    start: Minimální počet témat
    stop : Maximální počet témat
    step : Velikost kroku

    Return
```

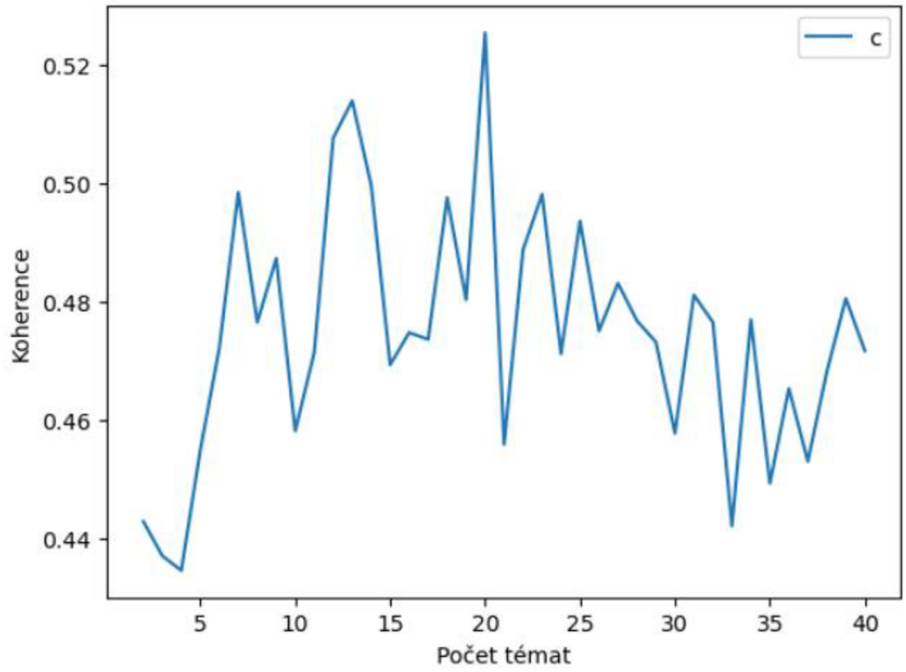
```
    ------
    model_list : Seznam LSA modelů
    coherence_values : Koherence jednotlivých modelů na základě počtu témat
    """
    coherence_values = []
    model_list = []
    for k in range(start, stop, step):
        model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                                id2word=id2word,
                                                num_topics=k,
                                                chunksize=1000)
        model_list.append(model)
        coherencemodel      =      CoherenceModel(model=model,      texts=data_clean,
dictionary=id2word, coherence='c_v')
        coherence_values.append(coherencemodel.get_coherence())
    return model_list, coherence_values

def plot_coherence_LDA(id2word, corpus, data_clean, start, stop, step):
    """
    Funkce vykreslí spojnicový graf, který zobrazuje velikost soudržnosti textu
podle počtu vybraných témat

    Parametry
    ---------
    id2word : Gensim dictionary
    corpus : Gensim corpus
    data_clean : Seznam předzpracovaných dat
    start: Minimální počet témat
    stop : Maximální počet témat
    step : Velikost kroku

    """
    model_list,    coherence_values    =    topics_by_coherence_LDA(id2word,    corpus,
data_clean,
                                                            stop, start, step)
    x = range(start, stop)
    plt.plot(x, coherence_values)
    plt.xlabel("Počet témat")
    plt.ylabel("Koherence")
    plt.legend(("coherence_values"), loc='best')
    plt.show()
```

Kód č. 22: Funkce seznamu koherencí + graf seznamu [Vlastní zpracování]

Spojnicový graf koherencí modelů je získán následovně.

```
plot_coherence_LDA(id2word, corpus, data_lemmatized, 2, 41, 1)
```

Kód č. 23: Zvolání funkce vykreslující spojnicový graf [Vlastní zpracování]



Obrázek č. 16: Graf koherencí modelů LDA podle počtu témat [Vlastní zpracování]

Z obrázku je patrné, že optimální počet témat pro tento model je 19, ukázka modelu v případě 20 témat viz výše. Koherence modelu v tomto případě vypadá následovně.

```
0.5270570613545533
```

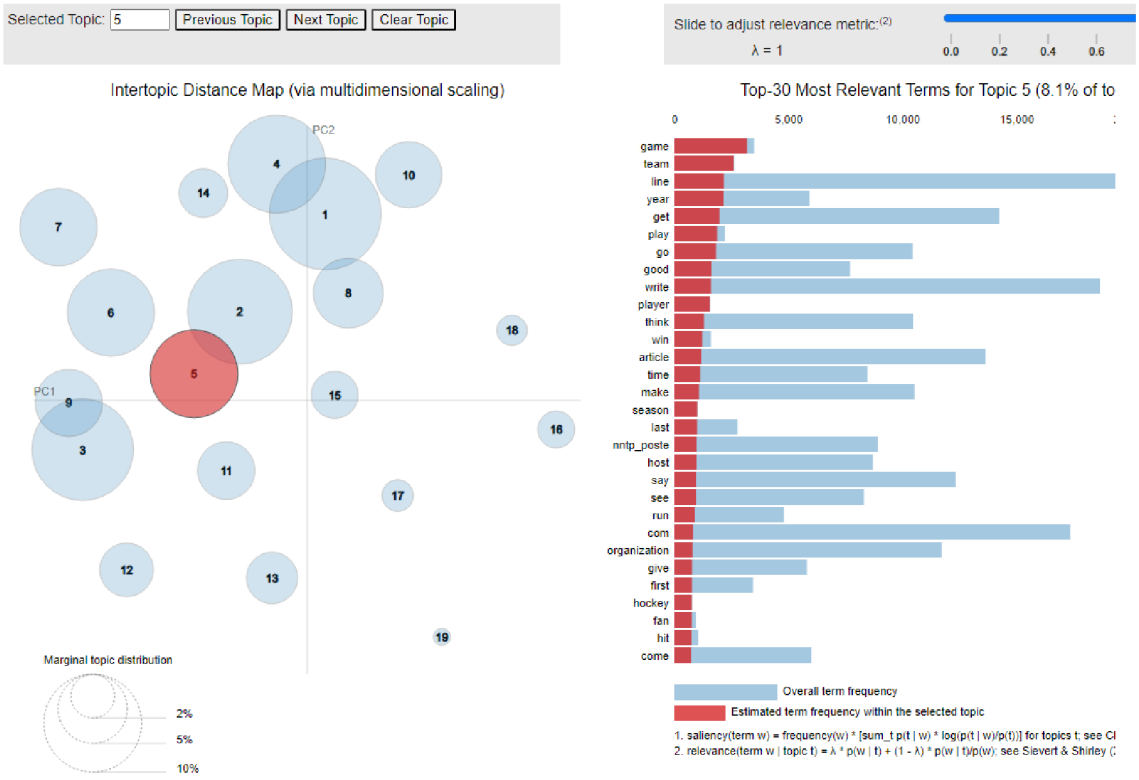
Kód č. 24: Output koherence optimálního modelu [Vlastní zpracování]

Opět je vhodné model určitým způsobem vizualizovat. Pro LDA existuje možnost využít knihovnu pyLDAvis, která umožňuje zobrazit distanční mapu jednotlivých témat s počtem nejvíce zastoupených slov v každém z nich.

```
pyLDAvis.enable_notebook()
vis_lda = pyLDAvis.gensim_models.prepare(lda_model, corpus, id2word, mds="mmds",
R=30)
vis_lda
```

Kód č. 25: Grafická reprezentace modelu LDA [Vlastní zpracování]

Atributy pyLDAvis jsou vybraný model; slovník je ve tvaru doc2vec corpus a id2word; mds znamená vícerozměrné škálování (multidimensional scaling, metric multidimensional scaling pro mmds) a R je počet nejvíce zastoupených slov.



Obrázek č. 17: Výstup pyLDAvis [Vlastní zpracování]

5.4 Non Negative Matrix Factorization

Vzhledem k tomu že NMF je také součástí Gensim jako LSI a LDA, tak v kódu nejsou velké rozdíly.

```
nmf = gensim.models.nmf.Nmf(corpus=corpus,
                            id2word=id2word,
                            num_topics=20,
                            chunksize=1000)

nmf.print_topics(20)
```

Kód č. 25: Model NMF [Vlastní zpracování]

Ovšem ve výstupu samozřejmě rozdíly existují, protože NMF využívá faktorizace nezáporných matic.

```
[(0,
  '0.035*"armenian" + 0.019*"people" + 0.009*"turkish" + 0.008*"turk" + 0.0
08*"government" + 0.006*"russian" + 0.006*"soldier" + 0.005*"kill" + 0.005*
"state" + 0.005*"village"'),
 (1,
  '0.049*"image" + 0.049*"file" + 0.018*"color" + 0.018*"jpeg" + 0.016*"for
mat" + 0.015*"program" + 0.014*"version" + 0.013*"display" + 0.012*"gif" +
0.009*"quality"'),
 (2,
  '0.051*"file" + 0.037*"entry" + 0.014*"make" + 0.010*"rule" + 0.010*"prog
ram" + 0.009*"name" + 0.009*"section" + 0.008*"follow" + 0.008*"need" + 0.0
07*"source"'),
.
.
```

```
.
(17,
  '0.015*"widget" + 0.014*"application" + 0.013*"window" + 0.013*"server" +
0.012*"available" + 0.011*"use" + 0.010*"resource" + 0.009*"set" + 0.009*"g
et" + 0.008*"include"'),
 (18,
  '0.120*"com" + 0.022*"write" + 0.014*"article" + 0.014*"get" + 0.012*"win
dow" + 0.012*"good" + 0.010*"do" + 0.009*"driver" + 0.008*"run" + 0.007*"ti
me"'),
 (19,
  '0.018*"key" + 0.015*"privacy" + 0.015*"government" + 0.014*"encryption"
+ 0.013*"system" + 0.012*"information" + 0.009*"security" + 0.009*"right" +
0.009*"internet" + 0.008*"technology"')]
```

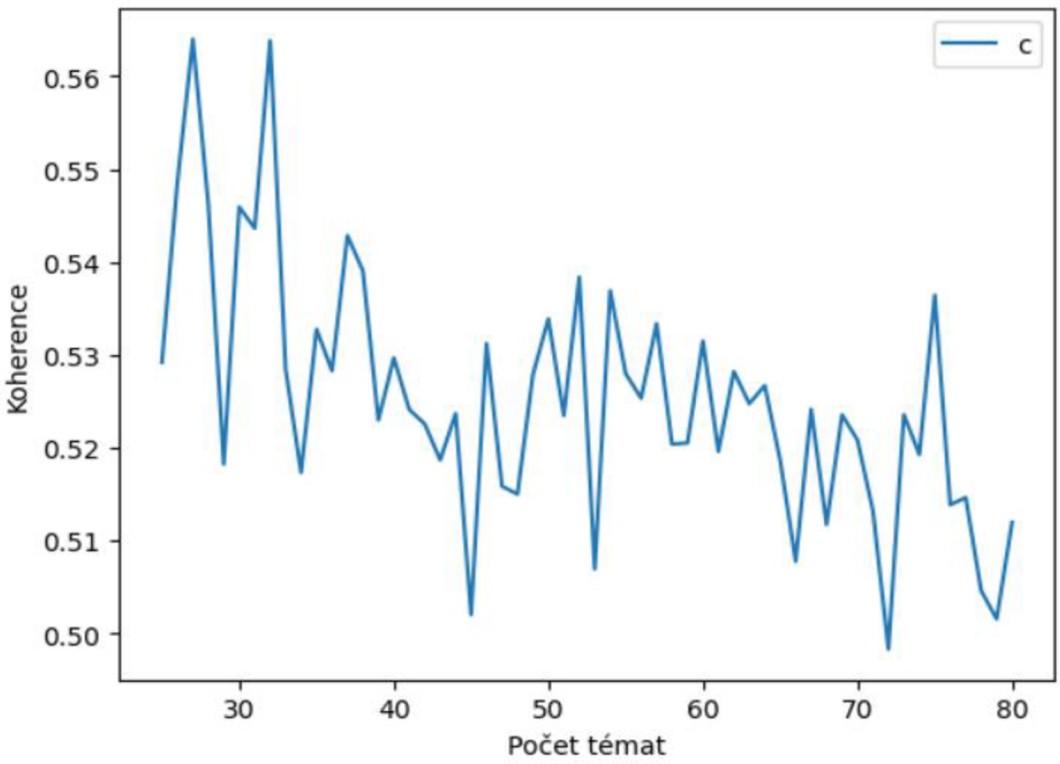
Kód č. 26: Output kódu č. 25 [Vlastní zpracování]

Stejně jako v obou případech LSI a LDA i zde lze získat optimální množství volených témat opakovaným tvořením modelů a výběrem toho nejlepšího z nich na základě koherence.

Na rozdíl od LSI a LDA, kde bylo vytvořeno pouze čtyřicet modelů, při použití NMF jich bylo vytvořeno osmdesát, protože nebylo jasné při prvních čtyřiceti, jaké je optimální množství témat.

```
plot_coherence_NMF(id2word, corpus, data_lemmatized, 25, 81, 1)
```

Kód č. 27: Zvolání funkce vykreslující spojnicový graf [Vlastní zpracování]



Obrázek č. 18: Graf koherencí modelů NMF podle počtu témat [Vlastní zpracování]

Z grafu lze zpozorovat, že modely s 27 tématy a 31 tématy mají identickou koherenci. V tomto případě byl zvolen model s 27 tématy.

```
nmf = gensim.models.nmf.Nmf(corpus=corpus,
                            id2word=id2word,
                            num_topics=27,
                            chunksize=1000)
nmf.print_topics(27)
```

Kód č. 28: Model NMF s optimálním počtem témat [Vlastní zpracování]

```
[(0,
  '0.051*"bit" + 0.018*"scsi" + 0.015*"color" + 0.015*"card" + 0.014*"pc" +
0.012*"hardware" + 0.009*"mode" + 0.009*"build" + 0.009*"cpu" + 0.009*"memo
ry"'),
 (1,
  '0.020*"graphic" + 0.015*"system" + 0.015*"mail" + 0.013*"available" + 0.
012*"send" + 0.011*"pub" + 0.010*"include" + 0.010*"datum" + 0.009*"package
" + 0.008*"software"'),
 (2,
  '0.016*"time" + 0.012*"take" + 0.011*"book" + 0.011*"come" + 0.011*"man"
+ 0.010*"many" + 0.009*"problem" + 0.008*"give" + 0.008*"day" + 0.006*"find
"'),
 (3,
  '0.026*"space" + 0.018*"launch" + 0.013*"satellite" + 0.011*"planet" + 0.
011*"mission" + 0.011*"earth" + 0.009*"probe" + 0.008*"system" + 0.008*"orb
it" + 0.007*"solar"'),
.
.
.
(24,
  '0.024*"internet" + 0.022*"privacy" + 0.021*"anonymous" + 0.016*"post" +
0.016*"user" + 0.015*"service" + 0.014*"email" + 0.013*"anonymity" + 0.012*
"network" + 0.011*"information"'),
 (25,
  '0.083*"file" + 0.033*"entry" + 0.018*"program" + 0.014*"output" + 0.010*
"section" + 0.009*"name" + 0.009*"build" + 0.009*"rule" + 0.008*"info" + 0.
008*"printf"'),
 (26,
  '0.019*"know" + 0.011*"group" + 0.010*"state" + 0.007*"issue" + 0.007*"ne
w" + 0.007*"adl" + 0.006*"year" + 0.006*"report" + 0.005*"public" + 0.005*"
decision"')]
```

Kód č. 29: Output kódu č. 28

Koherence NMF při 27 tématech vypadá následovně.

```
0.5693709110364752
```

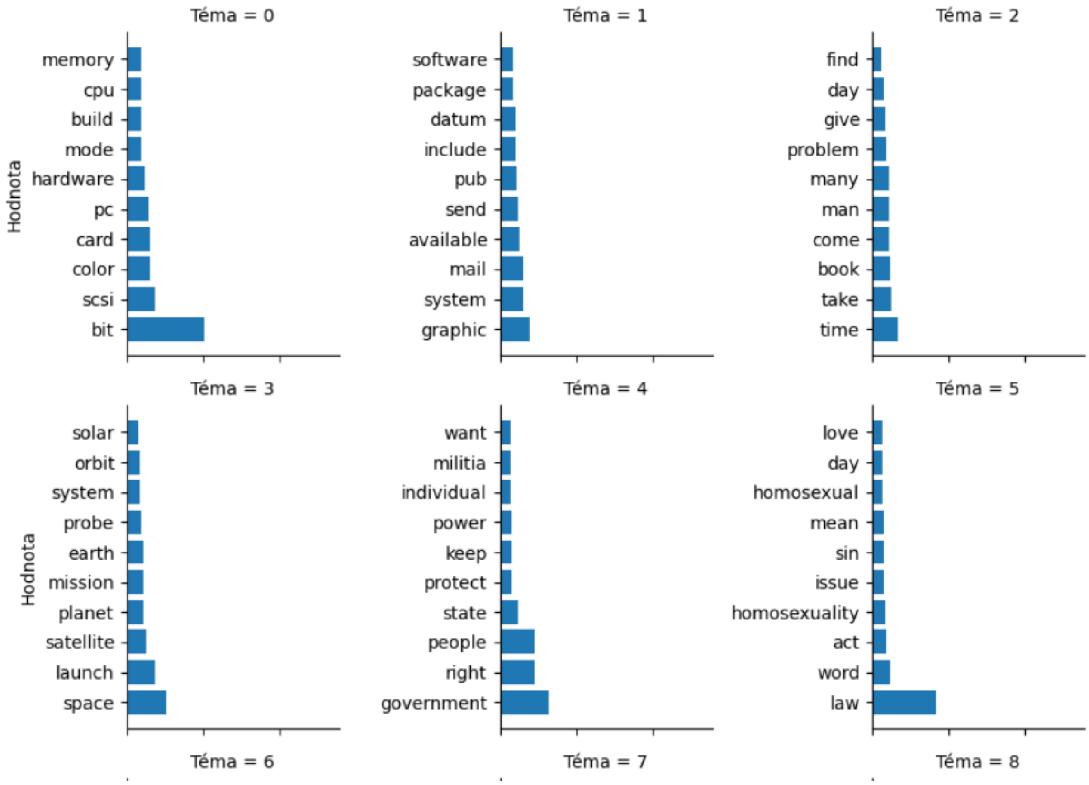
Kód č. 30: Output koherence optimálního modelu NMF [Vlastní zpracování]

Pro vizualizaci byly opět využity knihovny Pandas a Seaborn.

```
num_w = 10
topic_w_NMF = pd.DataFrame({})
for i, topic in enumerate(nmf.get_topics()):
    top_feature_ids = topic.argsort()[-num_w:][::-1]
    feature_values = topic[top_feature_ids]
```

```
    words = [id2word[id] for id in top_feature_ids]
    topic_df = pd.DataFrame({'Hodnota': feature_values, 'Slovo': words, 'Téma': i})
    topic_w_NMF = pd.concat([topic_w_NMF, topic_df], ignore_index=True)
g_NMF = sns.FacetGrid(topic_w_NMF, col="Téma", col_wrap=3, sharey=False)
g_NMF.map(plt.barh, "Slovo", "Hodnota")
```

Kód č. 31: Sloupcový graf zastoupení slov v tématech [Vlastní zpracování]



Obrázek č. 19: Sloupcový graf zastoupení slov v tématech [Vlastní zpracování]

5.5 Top2Vec

Předtím než je implementován model top2vec, je nutné se ujistit, že je k dispozici vše, co je třeba.

```
pip install top2vec
from top2vec import Top2Vec
```

Kód č. 32: Pip install + import Top2Vec [Vlastní zpracování]

Protože top2vec nevyžaduje manuální předzpracování dat, stačí načíst pouze příslušná data, případně s drobnými úpravami.

```
from sklearn.datasets import fetch_20newsgroups
data_emb = fetch_20newsgroups(subset='all', remove=('headers', 'footers',
'quotes'))
data_train_2 = data_emb.data
```

Kód č. 33: Import datasetu bez úprav [Vlastní zpracování]

Následovně je natrénován model.

```
model = Top2Vec(documents=data_train_2, speed="learn", workers=8)
```

Kód č. 34: Model Top2Vec [Vlastní zpracování]

Zde proměnná documents představuje naše data, proměnná speed rychlost učení a workers počet paralelních vláken při výpočtu.

Po zavolání na model není nic nového zjištěno, proto je důležité využívat celé řady funkcí, které jsou součástí top2vec. Ideální by bylo vědět, do kolika témat byla naše data vůbec zařazena.

```
model.get_num_topics()
```

```
110
```

Kód č. 35: Počet nalezených témat + output [Vlastní zpracování]

Tento příkaz vrátí počet vytvořených témat modelem. Lze zpozorovat, že 110 je celkem velké množství témat. Bylo by dobré vědět, jaké je zastoupení dokumentů v jednotlivých tématech. Pomocí příkazu get_topic_sizes lze tuto informaci získat.

```
topic_sizes, topic_nums = model.get_topic_sizes()
topic_sizes
```

```
array([1432, 1195,  846,  742,  669,  541,  454,  401,  370,  343,  340,
        333,  304,  295,  283,  276,  275,  270,  269,  261,  248,  241,
        230,  209,  203,  193,  192,  187,  185,  173,  173,  160,  159,
        159,  153,  151,  150,  148,  146,  141,  129,  126,  123,  118,
        116,  114,  114,  112,  110,  110,  110,  109,  108,  107,  106,
        104,  104,  101,   99,   99,   96,   96,   93,   91,   91,   90,
         84,   82,   79,   78,   77,   76,   76,   73,   70,   69,   69,
         69,   67,   66,   63,   63,   61,   60,   60,   60,   59,   58,
         57,   57,   56,   56,   54,   53,   53,   53,   53,   52,   52,
         50,   47,   46,   45,   44,   42,   39,   38,   38,   34,   32],
      dtype=int64)
```

Kód č. 36: Zastoupení dokumentů v tématech + output [Vlastní zpracování]

Je patrné, že velké množství témat obsahuje srovnatelně malé množství dokumentů. Proto jsou zredukována pomocí funkce hierarchical_topic_reduction, která umožní snížit počet témat a zároveň dokumenty, které jsou si nejvíc podobné, umožní řadit k sobě pod jedno téma.

```
model.hierarchical_topic_reduction(num_topics=18)
```

```
[[100, 56, 54, 61, 102, 0],
 [47, 70, 10, 62, 92, 16, 26, 107, 41, 90, 2],
 [75, 98, 1],
 [106, 51, 46, 20, 49, 15, 55, 39, 30, 35, 72, 109, 18],
 [37, 82, 77, 13, 57, 34, 31, 67, 101, 21],
 [83, 22, 69, 84, 32, 33, 71, 89, 99, 108, 12],
 [105, 103, 79, 36, 25, 91, 76, 59, 52, 68, 94, 38],
```

```
 [80, 9, 64, 11],
 [65, 19, 53, 58, 24],
 [27, 44, 29],
 [74, 87, 40, 50, 78, 7],
 [3],
 [45, 42, 66, 73, 14],
 [104, 28, 86, 93, 96, 8],
 [81, 60, 43, 48, 6],
 [95, 4],
 [88, 23, 97, 63, 85, 17],
 [5]]
```

Kód č. 37: Hierarchická redukce témat + output [Vlastní zpracování]

Z výstupu příkazu lze zjistit, jaká témata dohromady tvoří témata nová. Po opakování get_topic_sizes je získána informace pro upravený model.

```
topic_sizes, topic_nums = model.get_topic_sizes(18)
topic_sizes
```

```
array([1887, 1766, 1491, 1245, 1199, 1049,  996,  966,  905,  880,  867,
        863,  849,  848,  837,  768,  750,  680], dtype=int64)
```

Kód č. 38: Počet dokumentů redukovaného modelu + output [Vlastní zpracování]

Po zmenšení počtu témat nejmenší z nich obsahuje 680 dokumentů. Po upravení modelu je potřeba znát zastoupení slov v jednotlivých tématech. Lze je získat pomocí get_topics.

```
topic_words, word_scores, topic_nums = model.get_topics(18)
topic_words[0]
```

```
array(['psalms', 'psalm', 'jehovah', 'thee', 'theism', 'elohim',
       'corinthians', 'apostles', 'commandments', 'moses', 'isaiah',
       'baptism', 'sinful', 'unto', 'miracles', 'deity', 'mcconkie',
       'sinner', 'resurrection', 'prophets', 'revelation', 'thy', 'judas',
       'boswell', 'doctrines', 'scripture', 'salvation', 'scriptures',
       'luke', 'sins', 'romans', 'dogma', 'believers', 'thou',
       'disciples', 'christ', 'testament', 'gospel', 'allah', 'messiah',
       'obedience', 'cor', 'sabbath', 'contradict', 'teachings',
       'prophecy', 'traditions', 'qur', 'verse', 'theists'], dtype='<U15')
```

Kód č. 39: Zastoupení slov jednotlivých témat + output [Vlastní zpracování]

Zde proměnná topic_words reprezentuje padesát nejvíce zastoupených slov v tématu seřazených sestupně na základě sémantiky, word_scores reprezentuje kosinusovou vzdálenost/podobnost každého slova a topic_nums unikátní index každého tématu v podobě čísla. Při zvolání topic_words se specifikací na první téma lze usoudit, že téma se převážně týká náboženství a lze ho tak i pojmenovat.

Jednotlivá témata lze zobrazit také jako wordcloud.

```
model.generate_topic_wordcloud(0)
```

Kód č. 40: Wordcloud prvního tématu [Vlastní zpracování]

Obrázek č. 20: Wordcloud prvního tématu Top2Vec [Vlastní zpracování]

V rámci knihovny top2vec lze také hledat různá témata na základě klíčových slov. Například jsou hledána témata, která se týkají vesmíru, je specifikováno tedy klíčové slovo *„space“*.

```
topic_words,word_scores,topic_scores,topic_nums=model.search_topics(keywords=["space"],
num_topics = 3)
for topic in topic_nums:
    model.generate_topic_wordcloud(topic)
```

Kód č. 41: Hledání témat „space“ + output wordcloudů [Vlastní zpracování]

Obrázek č. 21: Wordcloud *space* Top2Vec [Vlastní zpracování]

Pomocí funkce search_topic lze v originálním modelu vyhledávat témata podle konkrétních slov. Z výsledných wordcloudů je patrné, že se opravdu jedná o vesmír. Zároveň je vidět, že se jedná o témata 63, 23 a 17, která byla po hierarchální redukci sloučena do stejného tématu. Kromě vyhledávání témat top2vec lze vyhledávat také konkrétní dokumenty, například na základě toho, do jakého tématu patří, nebo lze také využívat klíčová slova.

```
Documents,document_scores,document_ids=model.search_documents_by_keywords(keywords=["cr
ime", "police"], num_docs=5)
for doc, score, doc_id in zip(documents, document_scores, document_ids):
    print(f"Document: {doc_id}, Score: {score}")
    print("-----------")
    print(doc)
    print("-----------")
    print()
```

```
Document: 1155, Score: 0.4934
-----------
 >drug dealers, spies, terrorists, and organized crime figures (assuming
 >enough probable cause to convince a judge) who need to be watched, not
 >law-abiding citizens.
-----------
Document: 8553, Score: 0.4798
-----------
   "A handful of anti-gun zealots are telling the public that their
right to self-defense is of less importance than the interests of
Handgun Control, Inc.  This action comes as local, state and federal law
enforcement officials continue their assault on the Branch Davidian
compound--an assault which has already resulted in the death of one
two year old child at the hands of federal agents.  This has highlighted
the need for citizens to be able to defend themselves and their children
against the excesses of their own government."

  "Any suggestion by opponents that this bill will increase crime is a
distortion of the facts, at best.  The aggressive outreach by officials
in central Florida to train and arm women has led to a dramatic drop in
the level of assault and rape in that area.  Of course, this program is
a rare gem, as many law enforcement officials apparently believe that an
unarmed citizenry will be easier to control, and thus favor tighter
restrictions."

   "The vote today is a tribute to the good sense of the public at large
who are putting their lives on the line every day as they go about their
lawful affairs.  The entire country knows how vulnerable the average
```

```
citizen is, both to attacks from criminals and from armed assault by our
own police.  Texas lawmakers who voted for this concealed handgun bill have
shown total understanding for those innocent, law-abiding citizens on the
front lines, and the families of those who have fallen."

   "I urge the House of Representatives to pay attention to the needs
of their constituents, and not be stampeded by ill-conceived arguments
from ideological fanatics."

Ain't propaganda fun?
-----------

Document: 1962, Score: 0.446
-----------
Here are a few ideas:

1) a free library card so they can look up the FBI
   Uniform Crime Report which shows how good HCI is
   at lying through their teeth,

2) a free RTD Transit Pass which will allow anti-gunners
   to tour South Central Los Angeles and convince
   people living there that they don't need guns to protect
   themselves because the police will do it for them
   (don't lose the pass, you'll need it to get out),

3) a free bus ride to Vermont, which has almost no gun
   control and, curiously enough, almost no crime either,

4) a free calculator, since anti-gunners have heretofore
   been unable to figure out what a small percentage of
   the guns owned in America are used to commit violent crime.
-------------------------------------------------------------
Lee Gaucher    NRA                  |  My opinions.
gaucher@sam.cchem.berkeley.edu      |  No one else's.
-----------
```

Kód č. 42: Vyhledávání dokumentů „crime“ + output [Vlastní zpracování]

Kde document_id je unikátní celočíselná identifikace a score je podobnost dokumentu k danému tématu. Je možné také vyhledávat slova podobná určitým klíčovým slovům.

```
words, word_scores = model.similar_words(keywords=["medicine"], keywords_neg=[],
num_words=20)
for word, score in zip(words, word_scores):
    print(f"{word} {score}")
```

```
medical 0.6233
treatment 0.4993
doctors 0.4895
patients 0.4829
physicians 0.4642
clinical 0.4589
study 0.4564
nutrition 0.4554
patient 0.4473
med 0.444
health 0.433
disease 0.429
england 0.4285
studies 0.4256
therapy 0.4133
candida 0.4061
doctor 0.4036
cure 0.396
treatments 0.3901
physician 0.3888
```

Kód č. 43: Vyhledávání slov podobných „medicine“ + output [Vlastní zpracování]

Pomocí těchto různých metod je top2vec schopný analyzovat velké korpusy textových dat a vyhledávat vztahy a spojitosti mezi dokumenty a slovy.

5.6 BERTopic

Nejdříve je stažen a importován BERTopic.

```
pip install bertopic
from bertopic import BERTopic
from bertopic.representation import KeyBERTInspired
from sklearn.feature_extraction.text import CountVectorizer
```

Kód č. 44: Pip + import BERTopic [Vlastní zpracování]

Protože BERTopic je velice modulární model, tak lze pro jednotlivé kroky algoritmu vybírat různé metody pro tokenizaci, shlukování, zjišťování vah slov apod. [19] Kromě toho je možné využívat reprezentační modely pro přesnější identifikaci témat. Tyto modely nejsou využívány automaticky, proto je potřeba specifikovat, o jaký model se jedná. V tomto případě to je model KeyBERTInspired.

```
representation_model = KeyBERTInspired()
vectorizer_model = CountVectorizer(stop_words="english")
topic_model =
BERTopic(representation_model=representation_model,vectorizer_model=vectorizer_model,
min_topic_size = 100)
topics, probs = topic_model.fit_transform(data_train_2)
```

Kód č. 45: Model BERTopic [Vlastní zpracování]

Pro tokenizaci je využit CountVectorizer knihovny sklearn, který má již definovaná stop-slova. Atribut modelu min_topic_size umožňuje minimalizovat počet témat ještě předtím, než je budou redukována. Poté jsou pomocí funkce fit_transform získána jednotlivá témata a jejich pravděpodobnosti ve formě proměnných topics a probs.

Informace o jednotlivých tématech jsou získána pomocí get_topics_info.

```
topic_model.get_topic_info()
```

Kód č. 46: Základní informace nalezených témat [Vlastní zpracování]

	Topic	Count	Name	Representation	Representative_Docs
0	-1	2290	-1_president_stephanopoulos_general_government	[president, stephanopoulos, general, governmen...	[THE WHITE HOUSE\n\n Office o...
1	0	6057	0_hardware_graphics_monitor_software	[hardware, graphics, monitor, software, interf...	[HI,\n that might look like a dull request, bu...
2	1	1833	1_bikes_motorcycle_bike_tire	[bikes, motorcycle, bike, tire, tires, honda, ...	[: I am in the market for a bike and have rece...
3	2	1828	2_scoring_stats_nhl_hockey	[scoring, stats, nhl, hockey, puck, rangers, p...	[The regular season of the 1992-93 Davis Table...
4	3	1417	3_atheism_atheist_christianity_christians	[atheism, atheist, christianity, christians, a...	[It troubles me that there have been so many p...
5	4	927	4_medicine_physicians_patients_clinical	[medicine, physicians, patients, clinical, med...	[The DEA and other organizations would have th...
6	5	747	5_nasa_spacecraft_astronomy_saturn	[nasa, spacecraft, astronomy, saturn, shuttle,...	[Archive-name: space/references\nLast-modified...
7	6	686	6_encryption_scicrypt_nsa_chip	[encryption, scicrypt, nsa, chip, security, cl...	[\n\nHere is an article I found today in comp....
8	7	603	7_palestinians_gaza_palestinian_israeli	[palestinians, gaza, palestinian, israeli, isr...	[THE WHITE HOUSE\n\n Office...
9	8	532	8_hi___	[hi, , , , , , , ,]	[ites:, Hi,, Hi,]
10	9	406	9_fires_survivors_fired_investigation	[fires, survivors, fired, investigation, fbi, ...	[\input amstex\n\documentstyle{amsppt}\n\pagew...
11	10	399	10_read_say_sarcasm_says	[read, say, sarcasm, says, funny, posting, let...	[...His account that is.\n\nMany important iss...
12	11	333	11_firearm_firearms_handgun_guns	[firearm, firearms, handgun, guns, handguns, n...	[: > Last year the US suffered almost 10,000 w...
13	12	321	12_mail_mailing_newsletter_email	[mail, mailing, newsletter, email, addresses, ...	[Archive-name: space/net\nLast-modified: $Date...
14	13	240	13_homosexuality_homosexuals_homosexual_hetero...	[homosexuality, homosexuals, homosexual, heter...	[# #From the Santa Rosa (Cal.) Press-Democrat,...
15	14	227	14_armenians_armenian_turks_azerbaijanis	[armenians, armenian, turks, azerbaijanis, arm...	[Accounts of Anti-Armenian Human Right Violati...

Obrázek č. 22: Tabulka získaných témat BERTopic [Vlastní zpracování]

Téma označené číslem -1 obsahuje anomálie, tedy odlehlé dokumenty, které lze ve většině případů ignorovat.

Pro zjištění zastoupení konkrétního tématu lze využít funkce get_topic(n), kde n je celé číslo, které reprezentuje dané téma.

```
topic_model.get_topic(1)
```

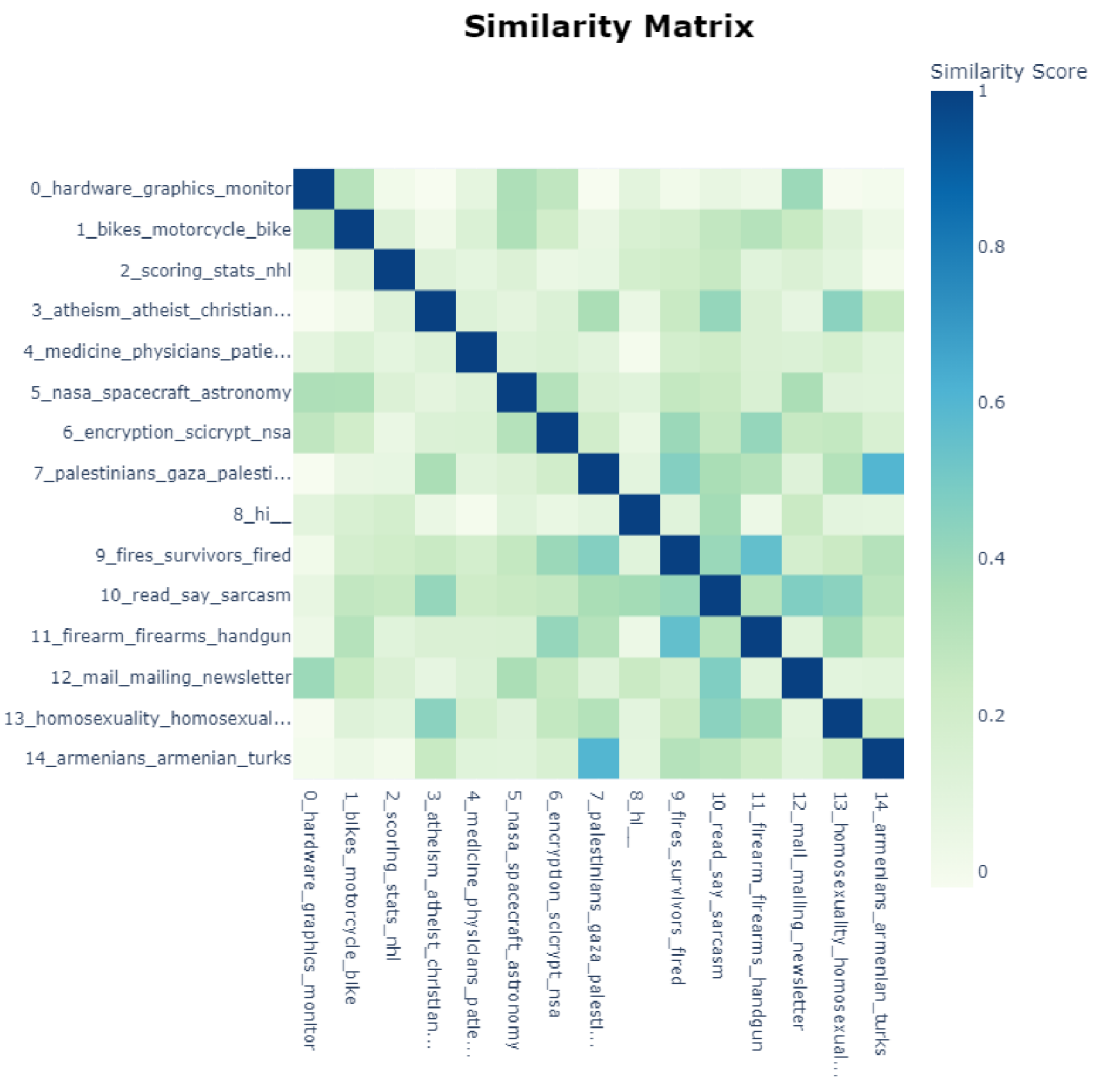
```
[('bikes', 0.5058784),
 ('motorcycle', 0.48880607),
 ('bike', 0.48377943),
 ('tire', 0.45843703),
 ('tires', 0.43571818),
 ('honda', 0.40717676),
 ('riding', 0.39953446),
 ('wheel', 0.39833397),
 ('brake', 0.3646013),
 ('brakes', 0.33162743)]
```

Kód č. 47: Slovní zastoupení tématu + output [Vlastní zpracování]

BERTopic umožňuje slučování různých témat, převážně na základě vlastního uvážení. Pro správné míchání témat, je nutné znát jejich vzájemnou podobnost. Ta je získána pomocí matice podobnosti, kterou lze zavolat příkazem visualize_heatmap.

```
topic_model.visualize_heatmap()
```

Kód č. 48: Vizualizace matice podobnosti témat [Vlastní zpracování]



Obrázek č. 23: Matice podobnosti BERTopic [Vlastní zpracování]

Nyní lze slučovat témata s velkou podobností pomocí funkce merge_topics.

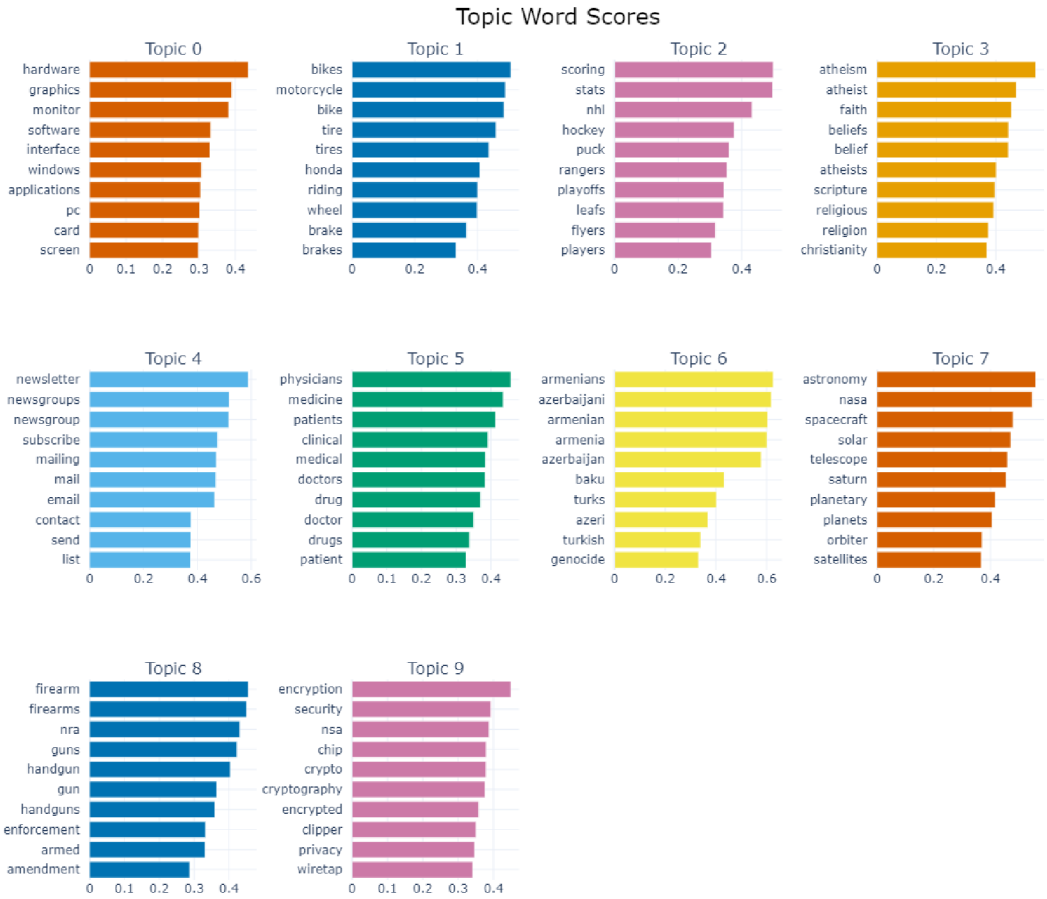
```
topics_to_merge = [[14, 7], [3,13],[12, 10], [9, 11]]
topic_model.merge_topics(data_train_2, topics_to_merge)
```

Kód č. 49: Slučování podobných témat [Vlastní zpracování]

Výsledky je možné i vizualizovat, například sloupcovým grafem.

```
topic_model.visualize_barchart(width = 280, height = 330, top_n_topics = 15,
n_words = 10)
```

Kód č. 50: Vizualizace sloupcových grafů [Vlastní zpracování]



Obrázek č. 24: Sloupcový graf témat BERTOPIC [Vlastní zpracování]

Vzhledem k tomu, že BERTopic je vysoce modulární knihovna, každý uživatel si může přizpůsobit modely podle svých požadavků. O různých možnostech je možné se dočíst v dokumentaci autora, viz [21].

6 Výsledky

V rámci testování těchto konkrétních pěti metod modelování témat byl využit volně dostupný dataset 20newsgroups [32], který se skládá zhruba z 18.000 dokumentů ve formě novinových článků. Výsledky extrakce témat z datasetu jsou reprezentovány v následujících tabulkách, kde je ukázáno 10 klíčových slov u každého tématu.

6.1 Výsledky – LSA

Latent Semantic Analysis		
	Téma	Klíčová slova
1	Operační systém	system, server, com, use, dos_do, include, window, user, support, application
2	Programování	file, entry, output, lint, printf, name, output_name, rule, return, build
3	Cybersecurity	internet, privacy, anonymous, image, mail, system, anonymity, service, computer, information
4	Electrický obvod	wire, circuit, wiring, ground, outlet, neutral, use, entry, cable, electrical
5	N/A	file, say, get, go, image, people, know, make, see, use
6	N/A	say, go, people, think, know, come, take, tell, make, armenian

Tabulka č. 1: Výsledky metody LSA. [Vlastní zpracování]

Při zjišťování optimálního počtu témat pro metodu LSA bylo zjištěno, že největší hodnotu koherence model obsahuje při zvolení šesti témat a při navyšování hodnota koherence rapidně klesá, viz implementace LSA. Šest témat v tomto případě je málo a u dvou z nich není zjevně známo, o jaké téma se jedná. LSA je obdoba distribuční sémantiky. Snaží se analyzovat vztahy mezi dokumenty a slovy, která dokumenty obsahují. Protože toho docílí pomocí singulární dekompozice, tedy soustředí se na zmenšování dimenzí dat, tak je zpravidla rychlejší než většina ostatních metod, ale není až tak přesná.

Nepřesnost LSA lze lehce demonstrovat pomocí výsledků, které byly získány pomocí metody LDA.

6.2 Výsledky - LDA

Latent Dirichlet Allocation		
	Téma	Klíčová slova
1	Náboženství/Pochyby o náboženství	believe, word, mean, question, argument, religion, church, belief, exist, evidence
2	Vesmír	space, earth, orbit, planet, launch, mission, moon, satelite, solar, sun
3	Nukleární hazard	cell, radar, radar_detector, converter, void, deadly, fatal, tissue, dangerous_mistake, led
4	Sport/Hokej	game, team, play, player, win, season, fan, hit, hockey, last
5	Hardware	window, card, do, disk, driver, pc, application, machine, drive, bit
6	Prodej software	price, sell, sale, cost, design, offer, model, space, pay, software
7	Grafický program	file, image, graphic, software, format, tool, library, 67roblém67y, faq, package
8	Cybersecurity	key, encription, security, privacy, clipper, government, secure, protect, chip, technology
9	Elektrický obvod/Náboženství/Chemie	ground, battery, wiring, homosexuality, sin, neutral, theory, chemical, toronto_zoology, gordon_bank
10	Motorová vozidla	car, bike, drive, engine, ride, speed, road, front, mile, auto
11	Noviny	adress, email, post, reply, internet, info, news, message, network, fax
12	Americké volby	state, government, right, law, report, vote, member, president, national, 67roblé_state
13	Zákon o držení zbraní	gun, drug, firearm, crime, weapon, license, criminal, handgun, criminal, gun_control
14	Zbraně	outlet, stratus, motto, captain, portal, packet, pistol, shotgun, explosive, rifle
15	Přestupky/Kampus	purdue, ticked, cop, gatech, technology, university, organization, write, com, article
16	Požár	people, say, come, child, kill, man, live, believe, fire, death
17	Sebevraždy	go, také, day, year, work, start, man, problem, tell, people
18	Válka	armenian, greek, turk, turkish, russian, genocide, argic, nazi, war, soldier
19	N/A	wave, apartment, bank, libertatian, online_communication, prb_access, click, inspection, lean, click

Tabulka č. 2: Výsledky metody LDA. [Vlastní zpracování]

V případě metody LDA bylo zjištěno, že optimální počet témat je podle koherence 19. Lze zpozorovat, že oproti LSA, LDA dokázalo identifikovat zhruba třikrát větší množství témat. Problém zde vzniká v tom, že se několik témat, jako třeba 13 a 14, překrývají, nebo téma 9 obsahuje klíčová slova z více témat. Taktéž téma 19 není snadno identifikovatelné z daných klíčových slov. LDA vyhledává skrytá témata na základě pravděpodobnosti s uvážením, že nad nimi existuje Dirichletovo rozdělení, proto je z větší části přesnější než LSA, ale trvá déle.

6.3 Výsledky - NMF

Non-negative Matric Factorization		
	Téma	Klíčová slova
1	Náboženství	say, make, think, elohim, name, speak, come, son, tell, let
2	Pochyby o náboženství	believe, atheist, exist, argument, say, belief, true, question, religion, mean
3	Homosexualita a náboženství	law, word, act, homosexuality, issue, sin, mean, homosexual, day, love
4	Vesmír	space, launch, satelite, planet, mission, earth, probe, system, orbit, solar
5	Sport/Hokej	make, game, play, team, year, season, go, first, nhl, goal
6	Hardware	bit, scsi, color, card, pc, hardware, mode, build, cpu, memory
7	Firmware	drive, system, controller, scsi, feature, hard_disk, head, bio, support, slave
8	Prodej software	graphic, system, mail, available, send, pub, include, datum, package, software
9	Programování	file, entry, program, output, section, name, build, info, rule, printf
10	Grafika	image, jpeg, color, format, gif, bit, file, display, quality, version
11	Software	window, use, do, application, dos_do, system, com, run, work, software
12	Update	get, file, available, version, program, include, window, server, widget, display
13	Cybersecurity	key, use, bit, chip, encryption, encrypt, security, program, block, datum,
14	Anonymita na internetu	internet, privacy, anonymous, post, user, service, email, anonymity, network, information
15	Noviny	write, line, article, organization, include, com, apr, host, nntp_poste, entry
16	Noviny 2	com, line, mail, organization, host, write, nntp_poste, article, message, sent
17	Americké volby	think, work, ms_myer, make, president, go, job, program, talk, option
18	Hledání bytu	go, say, come, know, tell, get, apartment, start, people, take
19	Veřejné stížnosti	know, group, state, issue, new, adl, year, report, public, decision
20	Zákon o držení zbraní	government, right, people, state, protect, keep, power, individual, militia, want
21	Střílení ve školách	child, gun, study, drug, car, year, research, patient, school, high
22	Sebevraždy	time, take, book, come, man, many, problem, give, day, find
23	Prosba o pomoc	people, think, get, thing, take, do, want, question, know, ask
24	Válka	people, write, article, gun, kill, many, jewish, fact, claim, state
25	Masakry	armenian, turkish, turk, russian, soldier, village, book, massacre, page, road
26	N/A	good, get, year, line, write, know, article, player, team, nntp_poste
27	N/A	see, make, look, know, openwindow, use, start, line, thing, put

Tabulka č. 3: Výsledky metody NMF. [Vlastní zpracování]

Na rozdíl od LSA a LDA, optimální počet témat pro model NMF vyšel 27. Většina témat je obsáhlá a snadno rozpoznatelná, přičemž ale metoda NMF nedokázala identifikovat téma Elektrický obvod, který se objevil v LSA a z části v LDA. Zároveň také existuje velké množství překrývajících se témat a dvě z nich nelze jednoduše identifikovat. Lze podotknout, že první tři témata by mohla spadat pod jedno téma Náboženství, které bylo již identifikováno metodou LDA. Mohlo by se říci, že metoda NMF podrobněji rozdělila témata, která byla nalezena pomocí LDA. NMF také postrádá téma Motorová vozidla, které se vůbec neobjevilo.

6.4 Výsledky - Top2Vec

Top2Vec		
	Téma	Klíčová slova
1	Náboženství	commandments, theism, corinthians, psalm, boswell, jehovah, moses, apostles, sinful, sinner
2	Vesmír	flyby, orbit, lunar, titan, jupiter, mariner, astronaut, gamma, planets, spacecraft
3	Hokej	winnipeg, recchi, scorer, overtime, gilmour, calgary, defenseman, sabres, hawks, nhl
4	Baseball	inning, rbi, pitcher, hitter, batting, sox, braves, mvp, runner, giants
5	Hardware	dram, powerpc, vram, slots, motherboard, quadra, cache, scsi, accelerator, adaptor
6	Prodej hardware	obo, cassette, cds, shipping, retail, warranty, sale, deck, stereo, speakers
7	Firmware	adaptec, irq, bios, controller, seagate, scsi, floppy, drive, port, config
8	Tiskárny	deskjet, cartridge, ink, dpi, printer, mbytes, laser, monochrome, truetype, upgrades
9	Programování	ndet_loop, args, libxmu, xmu, lib, defaults, exec, callback, colormap, printf
10	Grafika	xloadimage, jpg, jfif, gfx, jpeg, bmp, pallete, raster, formats, colormap
11	Cybersecurity	cryptosystem, decrypt, encrypt, clipper, phones, keys, algorithm, secure, trusted, anonymity
12	Kryptografie	siggraph, subscription, listserv, acm, sco, cryptology, iocc, bitnet, cryptanalysis, cornell
13	Elektrický obvod	grounding, volts, amp, circuits, conductor, voltage, outlet, wiring, watts, electricity
14	Motorová vozidla	brakes, throttle, pedal, tires, bikes, wheels, honda, torque, mustang, passenger
15	Zákon o držení zbraní	homicides, batf, handguns, assault, jury, firearm, senate, convicted, warrants, constitution
16	Masakry	syria, lebanese, palestinians, gaza, occupation, israelis, refugees, massacres, attrocities, civilians
17	Medicína	vitamin, chronic, placebo, dose, liver, nutrition, diagnosis, pacients, physicians, disease
18	N/A	leaf, humor, photography, cramer, lyuda, vitamin, ncsl, judas, inning, motorcycles

Tabulka č. 4: Výsledky metody Top2Vec.[Vlastní zpracování]

Top2vec automaticky generuje všechna možná témata, která najde, což může být více než sto témat. Součástí top2vec je však hierarchická redukce témat, kde je možné si zvolit finální požadovaný počet témat. V tomto případě bylo zvoleno osmnáct témat po hlubším prozkoumávání našich dat pomocí metody top2vec, která umožňuje vyhledávat témata pomocí klíčových slov. Využitá klíčová slova byla získána převážně z poznatků minulých metod. Lze vidět, že témata identifikovaná pomocí předchozích metod nechybí. Navíc také přibylo téma Medicína, Sport se rozdělil na hokej a baseball, přibylo téma Kryptografie a z nějakého důvodu top2vec vymezila téma Tiskárny. Chybí však téma, které se vztahuje k novinám.

6.5 Výsledky - *BERTopic*

BERTopic		
	Téma	Klíčová slova
1	Náboženství	atheism, atheist, faith, beliefs, belief, atheists, scripture, religious, religion, christianity
2	Vesmír	astronomy, nasa, spacecraft, solar, telescope, saturn, planetary, planets, orbiter, satellites
3	Hokej	scoring, stats, nhl, hockey, puck, rangers, playoffs, leafs, flyers, players
4	Počítače	hardware, graphics, monitor, software, interface, windows, applications, pc, card, screen
5	Cybersecurity	encryption, security, nsa, chip, crypto, cryptography, encrypted, clipper, privacy, wiretap
6	Motorová vozidla	bikes, motorcycle, bike, tire, tires, honda, riding, wheel, brake, brakes
7	Noviny	newsletter, newsgroups, newsgroup, subscribe, mailing, mail, email, contact, send, list
8	Zákon o držení zbraní	firearm, firearms, nra, guns, handgun, gun, handguns, enforcement, armed, amendmend
9	Masakry	armenians, azerbaijani, armenian, armenia, azerbaijan, baku, turks, azeri, turkish, genocide
10	Medicína	physicians, medicine, patients, clinical, medical, doctors, drug, doctor, drugs, patient

Tabulka č. 5: Výsledky metody BERTopic. [Vlastní zpracování]

Metoda BERTopic podobně jako top2vec identifikuje všechna možná témata. BERTopic taktéž umožňuje po modelování jednotlivá témata slučovat, většinou na základě podobnosti, kterou lze také zobrazit ve formě matice podobnosti. Po sloučení všech nejblíže podobných témat jich zbylo 10, přičemž tato témata vystihují téměř všechna t, která byla nalezena pomocí předchozích metod. Bohužel chybí téma Elektrický obvod.

6.6 Výhody a nevýhody jednotlivých metod

Jednotlivá témata byla pojmenována na základě jejich klíčových slov. Každý algoritmus je jedinečný a zjevně spoléhá na různé předpoklady. Rozdíly mezi nimi vystihuje následující tabulka výhod a nevýhod.

Metoda	Výhody	Nevýhody
LSA	• Lehce pochopitelné a využitelné • Snižuje dimensionalitu dat, což umožňuje práci s velkým množstvím dat • Vyhledává skryté vztahy mezi dokumenty a slovy, což umožňuje přesnější porozumění textu	• Spoléhá na to, že existuje lineární vztah mezi slovy a dokumentem • Nedokáže rozpoznat dvojznačnost nebo kontext • Potřeba velkého množství dat • Předzpracování dat nezbytné
LDA	• Vyhledá smysluplná témata v případě správného zadání hyperparametrů • Funguje s minimálním vstupem • Generuje menší množství témat, než metody využívající vnořování slov • Jeden dokument může obsahovat více témat • Lehce využitelné	• Může dojít k překrývání témat • Optimální počet témat lze určit pouze heuristicky • Výsledky nejsou deterministické • Využívá pouze frekvenci výskytu slov, počítá s tím, že témata jsou nezávislá • Předzpracování dat nezbytné
NMF	• Lehká implementace • Jeden dokument může obsahovat více témat • Využívá TF-IDF pro spočítání vah jednotlivých slov v jednotlivých dokumentech	• Často nachází nesmyslná témata • Optimální počet témat lze určit pouze heuristicky • Předzpracování dat nezbytné
Top2Vec	• Automaticky najde počet témat • Vytváří společně vnořená slova, dokumenty a vektory témat • Umožňuje vyhledávat určitá témata a dokumenty • Nepotřebuje předzpracování dat	• Vnořováním vzniká velké množství témat, jejichž prohlídka trvá dlouho • Vzniká mnoho anomálií • Každý dokument je přiřazen pouze jednomu tématu • Neobsahuje metriky evaluace
BERTopic	• Vysoká modularita • Nepotřebuje předzpracování dat • Automaticky najde počet témat • Umožňuje vyhledávat určitá témata a dokumenty • Větší podpora pro modely vnořování	• Vnořováním vzniká velké množství témat, jejichž prohlídka trvá dlouho • Vzniká mnoho anomálií • Každý dokument je přiřazen pouze jednomu tématu • Neobsahuje metriky evaluace

Tabulka č. 6: Výhody a nevýhody použitých metod. [Vlastní zpracování]

Na základě těchto vlastnosti lze usoudit, že LSA se spíše soustředí na zmenšení dimenzionality dat a není úplně vhodné pro modelování témat, kde je důležité rozpoznat kontext. LDA a NMF jsou obě validní metody pro modelování témat, avšak mají problémy s překrýváním témat a vyhledáváním témat nesmyslných. Metody top2vec a BERTopic jsou si zdánlivě velice podobné, avšak při implementaci různých nástrojů při úpravě u obou modelů bylo mnohem snazší dostat se k požadovanému výsledku pomocí metody BERTopic, převážné kvůli vysoké modularitě. Lze podotknout, že tento výsledek je velice situační. Může se stát, že metoda BERTopic nebude ideální v jiném případě. Avšak v tomto případě byla metoda BERTopic z hlediska výsledků nejlepší z vybraných.

7 Závěr

Modelování témat je technika strojového učení, která je schopná číst soubory dokumentů, vyhledávat v nich slovní a frázové vzory a automaticky je shlukovat do slovních skupin a podobných frází, které nejlépe charakterizují dané dokumenty. V dnešní době se na internetu objevují miliony příspěvků od různých osob po celém světě, převážně na sociálních sítích. Analyzování dat, která jsou složena z příspěvků na sociálních sítích, z emailů, chatů, dotazníků apod., není jednoduchá úloha. Prakticky není možné ji splnit pouze lidskými silami. Pomocí modelů analýzy témat jsou např. velké podniky schopny předat jednoduché úlohy strojům, nemusí tak platit zaměstnance za zkoumání dat kousek po kousku. Ohromné množství času lze využít pro důležitější úlohy, jestliže se stroj bude zabývat zákaznickým průzkumem za vás. Modelování témat není využitelné pouze pro podnikatelské účely, ale lze ho prakticky využít pro vědecký výzkum, zdravotnictví nebo monitorování médií.

Cílem této práce bylo seznámit čtenáře se zpracováním přirozeného jazyka, konkrétně s částí, která se zabývá extrakcí informací z textu, tedy metodami modelování témat. Práce obsahuje podrobně popsanou teorii, o kterou se vybrané metody opírají, přičemž lze z této teorie lehce poznat, že existuje značné množství různých přístupů, jak vytvářet modely témat. Další část práce se zabývá implementací těchto metod, jejíž součástí bylo předzpracování textu pro tři z nich. Vše bylo provedeno pomocí programovacího jazyka Python. Text byl předzpracován pomocí knihoven Gensim a spaCy, přičemž implementace metod LSA, LDA a NMF byla provedena také prostřednictvím knihovny Gensim. Při vytváření různých modelů záleží na cílovem počtu témat, u výše tří zmíněných metod záleží na výběru uživatele. Bohužel vybrat optimální počet témat lze pouze heuristickým přístupem, a to tak, že využijeme metriky koherence, které popisují soudržnost textu, a spočítáme ji pro desítky modelů, abychom zjistili, který má nejvyšší hodnotu koherence. Metody Top2Vec a BERTopic nemají tento problém, protože jsou schopny nalézt počet témat samy. Jenomže nalezený počet témat bývá velice značný, proto je důležité konkrétní témata hluboce prohledat, případně sloučit. Naštěstí obě metody obsahují nástroje pro prohledávání jednotlivých témat, dokumentů a dokonce i slov. Při implementaci Top2vec a BERTopic lze také využít hierarchickou redukci témat, což nám nalezená témata na základě podobnosti sloučí automaticky. Ovšem v tomto případě je volba finálního počtu témat opět na uživateli.

Těchto pět metod bylo v rámci práce aplikováno na dataset zhruba o 18.000 dokumentech. U každé z nich byly pomocí nalezených klíčových slov přiřazeny názvy nalezených témat. Práce pomocí těchto výsledků diskutuje o rozdílech mezi jednotlivými metodami. Součástí zpracování výsledků je i tabulka, ve které jsou porovnány vybrané metody, jejich výhody a nevýhody z pohledu uživatele.

Na závěr lze podotknout, že v rámci modelování témat existuje mnohem více metod než pět vybraných a dosažený výsledek je velmi závislý na vybraném datasetu. V případě dalšího testování doporučuji využívat co nejmodernějších metod, protože oblast umělé inteligence se neustále vyvíjí.

8 Seznam použité literatury

[1] Maynard, Diana, et al. Natural Language Processing for the Semantic Web. Morgan & Claypool.

[2] Natural Language Processing with Python, dokumentace na webu. (https://www.nltk.org/book/)

[3] Jiang, D., Zhang, C., & Song, Y. (2023). Probabilistic topic models: Foundation and application (1st ed.). Springer.

[4] Indurkhya, N., & Damerau, F. J. (2010). Handbook of Natural Language Processing. CRC Press.

[5] What is Natural Language Processing? | IBM. (n.d.). (https://www.ibm.com/topics/natural-language-processing)

[6] Lutkevich, B., & Burns, E. (2023, January 20). natural language processing (NLP). Enterprise AI. (https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP)

[7] Wikipedia přispěvatelé. (2024, January 19). Natural language processing. Wikipedia. (https://en.wikipedia.org/wiki/Natural_language_processing)

[8] Pazienza, M. T. (2006). Information Extraction: a multidisciplinary approach to an emerging information technology: A Multidisciplinary Approach to an Emerging Information Technology. Springer.

[9] GeeksforGeeks. (2022, January 10). Difference between Information Retrieval and Information Extraction. (https://www.geeksforgeeks.org/difference-between-information-retrieval-and-information-extraction/)

[10] Sarawagi, Sunita. (2008). Information Extraction. Foundations and Trends in Databases.

[11] Baby, Anusuya. (2023). Exploring the Power of Topic Modeling Techniques in Analyzing Customer Reviews: A Comparative Analysis.

[12] Kherwa, P., & Bansal, P. (2018). Topic Modeling: A Comprehensive review. ICST Transactions on Scalable Information Systems, 0(0), 159623. (https://doi.org/10.4108/eai.13-7-2018.159623)

[13] Topic Modeling: An Introduction. (2019, September 26). MonkeyLearn Blog. (https://monkeylearn.com/blog/introduction-to-topic-modeling)

[14] Blei, David & Ng, Andrew & Jordan, Michael. (2002). Latent Dirichlet Allocation. The Journal of Machine Learning Research. 3. 601-608.

[15] Evangelopoulos, N. (2013). Latent semantic analysis. WIREs Cognitive Science, 4(6), 683–692. (https://doi.org/10.1002/wcs.1254)

[16] Lee, Daniel & Seung, Hyunjune. (2001). Algorithms for Non-negative Matrix Factorization. Adv. Neural Inform. Process. Syst.. 13.

[17] Van Otten, N. (2023, December 22). Non-Negative matrix factorization explained & practical how to guide in Python. Spot Intelligence. (https://spotintelligence.com/2023/09/08/non-negative-matrix-factorization)

[18] Angelov, Dimo. (2020). Top2Vec: Distributed Representations of Topics.
[19] Grootendorst, Maarten. (2022). BERTopic: Neural topic modeling with a class-based TF-IDF procedure.
[20] Devlin, Jacob & Chang, Ming-Wei & Lee, Kenton & Toutanova, Kristina. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.
[21] Grootendorst, M. P. (n.d.). The algorithm - BERTopic. (https://maartengr.github.io/BERTopic/algorithm/algorithm.html)
[22] Reimers, Nils & Gurevych, Iryna. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. 3973-3983. 10.18653/v1/D19-1410.
[23] Sklearn.feature_extraction.text.CountVectorizer. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.htm
[24] Turing, A. (1937). On Computable Numbers, with an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, s2-42(1), 230–265. (https://doi.org/10.1112/plms/s2-42.1.230)
[25] Grishman, R. & Sundheim, B. (1996). Message Understanding Conference-6: a brief history. Association for Computational Linguistics. (https://doi.org/10.3115/992628.992709)
[26] Prezentace UHK. Martina Husáková. Dolování z textu 2023/2024
[27] https://www.datacamp.com/tutorial/what-is-topic-modeling
[28] https://www.youtube.com/watch?v=M1duqgg8-IM&t=2s
[29] Rehurek, R., & Sojka, P. (2011). Gensim–python framework for vector space modelling. NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic, 3(2).
[30] Honnibal, M., & Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.
[31] BotPenguin. (2023). Information extraction: Process & Applications | BotPenguin. https://botpenguin.com/glossary/information-extraction
[32] The 20 newsgroups text dataset — scikit-learn 0.19.2 documentation. (n.d.). https://scikit-learn.org/0.19/datasets/twenty_newsgroups.html

Seznam obrázků

Seznam tabulek

Seznam kódů

Přílohy

[1] https://unihk-my.sharepoint.com/:f:/g/personal/kanskji1_uhk_cz/Es2qUDzyo61Bsr1do7FHmwAB9ut0pCYYTVNURWfKV0375w?e=1JMv2s

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání diplomové práce

Autor:	**Bc. Jiří Kánský**
Studium:	I2200508
Studijní program:	N0688A140019 Datová věda
Studijní obor:	Datová věda
Název diplomové práce:	**Metody modelování témat**
Název diplomové práce AJ:	Topic modeling methods

Cíl, metody, literatura, předpoklady:

Cílem práce je implementovat různé metody modelování témat v rámci extrakce informací z textu. Práce bude obsahovat praktické použití několika různých algoritmů pro modelování témat, které závěrem porovná a vyhodnotí, zda jsou algoritmy vhodné pro vybraná data.

Osnova:

1. Úvod
2. Zpracování přirozeného jazyka
3. Extrakce informací
4. Přístupy k modelování témat
5. Implementace jednotlivých algoritmů
6. Testování dat
7. Závěry a doporučení

1. Natural language processing and computational linguistics: a practical guide to text analysis with Python, Gensim, spaCy, and Keras / Bhargav Srinivasa-Desikan Birmingham: Packt, 2018 . iv, 295 stran . ISBN 978-1-78883-853-5 (brožováno).
2. Practical natural language processing: a comprehensive guide to building real-world NLP systems / Sowmya Vajjala, Bodhisattwa Majumder, Anuj Gupta, and Harshit Surana Beijing ; Boston ; Farnham ; Sebastopol ; Tokyo : O'Reilly, 2020 . xxvii, 424 stran . ISBN 978-1-492-05405-4
3. Python natural language processing : explore NLP with machine learning and deep learning techniques / Jalaj Thanaki Birmingham : Packt, 2017 . x, 464 stran . ISBN 978-1-78712-142-3.
4. Probabilistic topic models: Foundation and application /Jiang, D., Zhang, C., & Song, Y: Springer Singapore, 2023 . i, 149 stran. ISBN 978-981-99-2431-8.

Zadávající pracoviště:	Katedra informačních technologií, Fakulta informatiky a managementu
Vedoucí práce:	Ing. Martina Husáková, Ph.D.
Datum zadání závěrečné práce:	15.10.2021