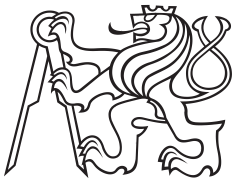
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Využití Text miningu pro automatické generování názvu rodiče

Petr Kostka

Vedoucí práce: Ing. Jiří Šebek
Obor: Software
Studijní program: Otevřená informatika
Květen 2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kostka** Jméno: **Petr** Osobní číslo: **466178**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Software**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Využití Text miningu pro automatické generování názvu rodiče

Název bakalářské práce anglicky:

Pokyny pro vypracování:

Parent naming je u adaptivní struktury[2] často využíván, je třeba zlepšit tento proces za použití metod z text miningu.
Cíle práce:
1) Prostudujte, které možnosti z oboru Text miningu [1] jsou vhodné k tomuto řešení
2) Zjistěte míru použití vašeho řesení a jeho efektivnost
3) Implementujte do frameworku [2] vaše řešení a otestujte ho.
Testy proveďte pomocí jak uživatelských testů tak pomocí heuristik.
4) Vyhodnoťte funkcionalitu výsledného řešení, výhody a limita oproti stávajícímu řešení.

Seznam doporučené literatury:

[1] Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos santos, Caglar Gulcehre, Bing Xiang: Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond: 19 Feb 2016
[2] Šebek, J. - Richta, K.: Usage of Aspect-Oriented programming in Adaptive Application Structure. New Trends in Databases and Information Systems: ADBIS 2016 Short Papers and Workshops, BigDap, DCSA, DC, Prague, Czech Republic, August 28-31, 2016, Proceedings

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Jiří Šebek, kabinet výuky informatiky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **14.02.2019** Termín odevzdání bakalářské práce: **24.05.2019**

Platnost zadání bakalářské práce: **20.09.2020**

Ing. Jiří Šebek podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studenta

Poděkování

Poděkování patří především mému vedoucímu bakalářské práce panu Ing. Jiřímu Šebkovi za jeho cenné rady, připomínky a trpělivost.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 23. května 2019

Abstrakt

Tato bakalářská práce se zabývá využitím Text Miningu pro generování společného sjednocujícího jména pro řadu slov a návrhem na jejich následnou kategorizaci a dělení. Hlavním cílem je vymyslet algoritmus pro podání přesných výsledků této funkcionality, protože na trhu takový chybí. Dále jej implementovat a umožnit jeho snadné použití.

Pro řešení této práce byl nezbytný framework JWNL na bázi lexikologické databáze Wordnet. Finální výsledek umožňuje vytváření rodičů pomocí dvou odlišných metod využívající znalosti Text Miningu. Jejich odlišnosti s danými výhodami a nevýhodami byly porovnány a vyhodnoceny na testovacích sadách, které potvrdily dobré výsledky s rychlou odezvou.

Klíčová slova: Text mining, Parent naming, Kategorizace textu, Adaptivní aplikace, Kontext, Text

Vedoucí práce: Ing. Jiří Šebek

Abstract

This bachelor thesis is focused on the usage of Text Mining for generating a common unifying name for the group of words and their subsequent categorization and division. The main goal is to devise algorithm for creating accurate results of this functionality, because that one is missing in the market. Next task is to implement it and make it easy to use.

The JWNL framework based on the lexicological database Wordnet was necessary to solve this work. The final result allows the creation of parents using two different methods using Text Mining. Their differences with the given advantages and disadvantages were compared and evaluated on test kits which confirmed good results with rapid response.

Keywords: Text mining, Parent naming, Text Categorization, Adaptive appliactiom, Context, Text

Title translation: Usage of Text mining in automatic parent name generation

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

1.1 Motivace

Čas věnovaný ovládání mobilních zařízení se každým rokem zvyšuje [21] a objevují se snahy ho uživateli alespoň zkrátit usnadněním komunikace se zařízením. V posledních letech se rozmáhá tzv. adaptivní rozhraní, které se uživateli na jakémkoli přístroji automaticky přizpůsobí dle kontextových informací v jeho okolí ve snaze usnadnit mu svou ovladatelnost. Tento přístup je však stále na svém počátku a má ještě několik problémů, které je potřeba vyřešit, aby toto řešení mohlo být naplno využito v praxi.

Jedním z těchto problémů je chování prvků v meta-modelu [15], a to konkrétně tzv. parent-naming - neboli pojmenování několika elementů jedním charakterizujícím souhrnným slovem. Jak tedy co nejlépe automatizovaně vybrat to nejlepší slovo? Abych přispěl k řešení, rozhodl jsem se danou problematiku zkoumat za pomoci Text Miningu. V textu je skryto mnoho dat a pro budoucnost se jedná o jeden z velmi perspektivních a žádaných řešení k získávání informací. Díky kontextovým informacím z daných slov tedy můžeme získat potřebné údaje ke správnému pojmenování. Přímo souvisejícím problémem může býti seskupování prvků pod rodiče, které by se mohlo řešit také za pomoci kontextu slov.

1.2 Cíle práce

Tato bakalářská práce má za cíl nalézt řešení pro pojmenování rodiče prvků a jejich rozdělování do různých tříd na základě kontextu slov. K tomuto využiji největší lexikologickou databázi Wordnet, která je implementována frameworkem JWNL pro Javu. Vytvořím na základě tohoto frameworku program, který bude získávat veškeré informace ze zkoumaných slov (například v menu aplikace), která využije k hledání společného charakterizujícího jména

jejich rodiče. Vedlejším úkolem bude pokusit se najít řešení ke kategorizaci slov do jednotlivých skupin podle podobnosti.

Dále otestuji moje řešení na různých testovacích sadách a pokusím se nalézt výhody mého řešení, kdy je vhodné jej použít a naopak kdy bude vhodný jiný přístup. Jinými slovy budu hledat limity daného řešení pro obecné použití. Finálním krokem bude implementace mého řešení do Mischenkovy Java EE aplikace [14], které bude demonstrovat jednoduchost implementace pro praktické využití.

Kapitola 2

Rešerše

2.1 Uživatelské rozhraní

Anglickým termínem User interface neboli v českém jazyce uživatelským rozhraním značíme jistý sled událostí poskytovaný zařízením lidem [10]. Pomocí tohoto obousměrného komunikačního kanálu můžeme efektivně zadávat operace se zpětnou vazbou od stroje. Pro uživatelovo pohodlí a rychlost práce zůstává právě uživatelské rozhraní klíčovou částí vývoje aplikací po mnoho let a nachází zde také mnoho aspektů, které zlepšit, ať se už jedná o rychlost odezvy, uživatelskou přívětivost či intuitivnost používání a s tím spojené efektivní využívání systému.

Pro tuto práci se budeme zabývat dvěma kategoriemi, které jsou odlišné v přístupu k těmto cílovým problémům [15]:

- **Statické uživatelské rozhraní**:

Rozhraní zůstává stejné a neměnné i po uživatelově zásahu. Výhodou tohoto přístupu je, že po chvíli se uživatel seznámí s daným rozhraním a naučí se ho, což mu dodá potřebné znalosti k využívání v dalších podobně situovaných částech systému. Pokud také dodržujeme stejné konvence v jiných aplikacích, učící časová křivka je podstatně menší. Nevýhodami může být stále se opakující schéma, které ubírá atraktivnosti rozhraní, ale hlavně žádné statické rozhraní nebude dokonale vyhovovat každému jednotlivému uživateli.

- **Adaptivní uživatelské rozhraní**:

Jak již název napovídá, toto rozhraní se adaptuje uživatelovým potřebám. Jak uvedli Norcio a Stanley [16]: *Rozhraní by se mělo přizpůsobit uživateli a ne uživatel systému.* Adaptivní UI mohou být klasifikována do několika sfér

závisejících na míře zapojení a úrovně uživatelových schopností. Jedním z nich je programovatelné adaptivní UI, které dává možnost naprogramovat nebo nastavit interakci a chování systému. Dalším je dynamicky adaptivní UI, jež se automaticky mění dle uživatelova chování.

Výhodou jednoznačně je přizpůsobení se vlastní potřebě jednotlivce, tento přístup nabízí tedy možnost, aby si uživatel přizpůsobil rozhraní a ne designer. Po určité době používání klesá strávený čas nad orientací se v systému. Na druhou stranu vývoj takové aplikace bude složitý na implementaci a zároveň finančně nákladnější oproti vývoji statického rozhraní.

2.2 Kontext

O nejlepší definici kontextu mezi sebou vědci a autoři výzkumných prací vedou spory již hezkou řádku let. Tento často používaný avšak až příliš obecný pojem používáme při každodenních situacích. Lidé denně úspěšně interagují navzájem mezi sebou a okolím díky mnoha věcem, ať už se jedná o bohatou slovní zásobu a její význam nebo všeobecný tok událostí v běžném každodenním životě, který zpracujeme a použijeme informace z daného okamžiku - kontext.

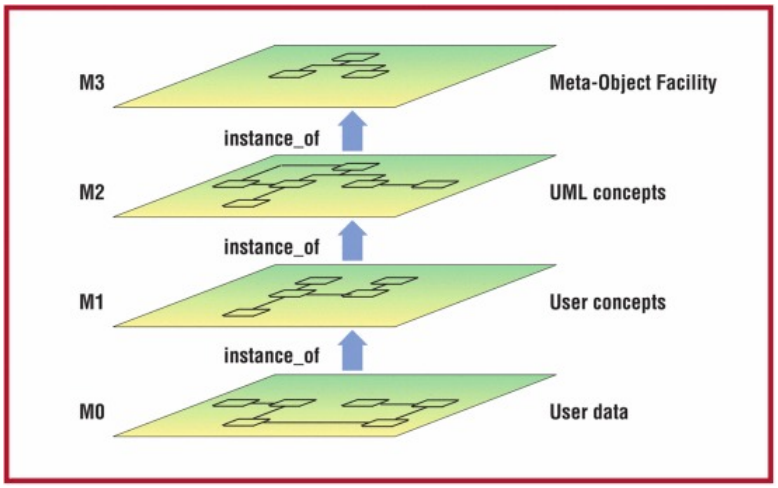
Přesněji tento termín zadefinoval Anind K. Dey [8]: *Kontext je jakákoliv informace, která může být využita k charakterizaci situace nějaké entity. Tato entita je člověk, místo nebo objekt, který je zvážen relevantním k interakci mezi uživatelem a aplikací včetně samostatného uživatele a aplikace.*

Avšak v interakci člověka s počítačem nejsou zařízení plně schopna využít tohoto kontextu, přičemž zlepšení tohoto nedostatku by mohlo výrazně posunout vývoj aplikací kupředu. V souvislosti s počítači a kontextem byl vytvořen nový pojem context-aware computing neboli počítačově kontextové povědomí [8]. Tento termín se poprvé objevil již v roce 1994, kdy ho Schilit a Theimer definovali jako [19]: *Software, který se přizpůsobí vzhledem k místu použití, skupině blízkých lidí a objektů, tak jako změnám u těchto objektů v čase.* Jinými slovy využíváme kontextové informace a kontext jako takový v zařízení, které tyto informace získá a následně je použije ke zpětné odezvě uživateli. Kontext se může získávat pomocí senzorů - od kamery, mikrofonu až po text a jiná data. Tato odezva může například sloužit jako lepší adaptace uživateli od změny designu až po různé customizace.

2.3 Metamodel a jeho související problémy

Pro metamodel existuje plno definicí, ale zdá se, že na jednotné se odborníci neshodnou. Abychom si mohli přiblížit, o co se jedná, položíme definici pro

model [5]. Model je zjednodušená verze reality a zachycuje entity a vztah mezi nimi. Metamodel [5], [4] je jeho další abstrakcí, ale jedná se o tzv. model modelu, který má malý sémantický obsah. Tyto abstrakce nám umožní lépe zachytit tzv. OMGs metamodelling infrastruktura[8] na obrázku 2.1.



Obrázek 2.1: OMG's metamoddeling infrastructure [5]

Jak je vidno, skládá se ze 4 vrstev, kde každá (až na M3) je modelem předchozí. Nejníže postavená je M0, která vlastní uživatelská data, která instancujeme jako model M1. M2 je model modelu označovaný jako již zmíněný metamodel. M3 se často označuje jako meta-meta model nebo jako Meta-Object Facility.

Metamodel zadefinoval ve své bakalářské práci Mischenko [14] trochu zjednodušeně jako: *Stromová struktura, která bude generována frameworkem ze které bude vývojář schopný zobrazit uživatelské rozhraní.* Z výsledků jeho práce lze nalézt až osm problémů při práci s metamodelem, které jsou následující [14]:

- chování meta-modelu po přidání nových prvků, nebo změně logického smyslu již existujících
- frekvence aktualizace metamodelu
- podle čeho generovat meta-model
- **seskupování prvků do rodiče**
- příliš mnoho prvků se stejným rodičem
- příliš hluboký strom
- **pojmenování rodiče**:
- vytvoření meta-modelu pro nové uživatele

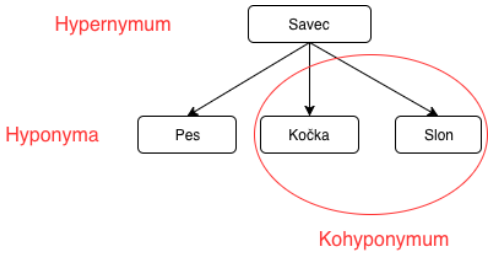
Tato bakalářská práce se zabývá právě řešením pojmenováním rodiče a poodkryje možná řešení seskupování prvků do rodiče.

2.4 Lexikologie

Lexikologie je lingvistická (=jazykovědní) disciplína zabývající se slovní zásobou a vztahy mezi nimi. Právě tyto významové vztahy lexikálních jednotek si uvedeme a vysvětlíme pro pozdější využití. U každého pojmu přidám definici a jednoduchý příklad pro porozumění [17], [18].

Termín	Definice	Příklad
Synonymum	Často označovaná jako slova souznačná, mají tedy podobný, až stejný, zaměnitelný význam.	teplo - horko
Antonymum	Často označována jako slova protikladná, mají tedy opačný význam, protiklad k synonymu.	teplý - studený
Hyperonymum	Často označována jako slova nadřazená, mají tedy obecnější význam než slova ze stejného významového okruhu. Vztah se označuje jako HAS-A.	savec je hyperonymem psovi, kočce a slonovi
Hyponymum	Často označována jako slova podřazená, mají tedy konkrétnější význam než slova ze stejného významového okruhu, protiklad k Hyperonymu. Vztah se označuje jako IS-A.	pes je hyponymem savce
Kohyponymum	Často označována jako slova souřadná, jsou tedy na stejné úrovni stejného významového okruhu.	pes je kohyponymem kočky a naopak
Meronymum	Označuje součást většího celku. Vztah se označuje jako Part-of.	prst je meronymem k ruce
Holonymum	Označuje název většího celku, protiklad k Meronymu. Vztah se označuje jako Has-Part.	ruka je holonymem k prstu
Polysémie	Často označovaná jako mnohoznačnost, se objevuje u slov, které mají 2 a více významů.	oko - zrakový orgán, díra v punčoše, past

Obrázek 2.2: Lexikologické pojmy.



Obrázek 2.3: Grafická demonstrace pojmů.

2.5 Text mining

Text mining [20], [11] by se dal souhrnně označit jako strojový proces získávání užitečných dat z často objemného, nestrukturovaného textu či textových databází, aby se jejich následnou analýzou dospělo k objevení

závislých vztahů mezi těmito získanými informacemi pro jejich účelné využití. Na tento problém se dá však nahlížet třemi směry [11]:

- Text mining jako pouhá extrakce informací:

Tento přístup může být používán, jestliže v daném textu hledáme konkrétní informace dle frází. Příkladem může být že chceme z textu získat jména osob a k nim jejich záliby, které se v textu objevují.

- Text mining jako Text Data Mining:

Podobně jako čistý data mining je využito strojové učení a statistické údaje k nalezení schémat a užitečných závislostí. Text mining se ale od Data miningu liší tím, že nemá strukturovaná data - text, je tedy potřeba je nejdřív převést, aby mohly být využity stejné postupy pro extrakci dat.

- Text mining jako Knowledge Discovery process (KDD):

Hearst tento proces obecně shrnul jako [9]: *Extrakce dosud neobjevených informací ve velkých kolekcí textu. Jde na to tedy nahlížet a aplikovat jako kombinaci předešlých dvou směrů.*

Vedou se odhady [11], že právě až 85% informací se nachází ve formě textu, které se ovšem dle klasických programovacích technik velmi obtížně sbírají. Jak tedy co nejlépe text reprezentovat, klasifikovat a hledat nové vztahy? Existují tři obory jak se k textu postavit [11]:

- Information Retrieval(IR)

Asi nejstarší z přístupů využívající se u vyhledávacích systémů stojí na bázi otázek a odpovědí. Dle klíčového slova se vám vrátí výsledek a to celý dokument. Často používané například při prohledávání právnických dokumentů, novin nebo webových stránek.

- Natural Language Processing(NLP)

Zpracování přirozeného jazyka je technika, která slouží lepšímu porozumění textu stroji. Základními kroky[**?**] tohoto přístupu je tedy převést nestrukturovaný text na strukturovaný. Hlavním pomocným kamenem je lexikologická databáze (lexicon) 2.6.

- Information Extraction(IE)

Tento přístup slouží k zjištění informací dle jednotlivých slov, frází či úryvků, o kterých víme, že se v textu budou pravděpodobně vyskytovat. Jedná se také o směr text miningu, jakým je jako celek chápán.

Abychom ale mohli text začít zpracovávat, je třeba ho upravit do formy, která bude vhodnější pro následnou analýzu. Tomuto procesu se říká text preprocessing [11] - tedy takové předzpracování textu. Daný text si lze představit jako posloupnost slov, která získáme pomocí tzv. tokenizačního procesu, tedy odstraníme veškeré mezery, tabulátory apod. mezi slovy a vytvoříme tzv. slovník textu. Ne však veškerá slova mají potřebnou informační hodnotu. Předložky, spojky a jiné znaky nemají žádnou kontextovou hodnotu, můžeme je tedy odstranit - tomuto procesu se také často říká filtrování. Neméně důležitou částí je převést slova do nějakého základního tvaru. Slova se často vyskytují s různými příponami, předponami a koncovkami, přičemž slovní význam je stejný. Hledáme tedy tzv. stem - v českém ekvivalentu kořene, logicky se tedy proces nazývá stemming.

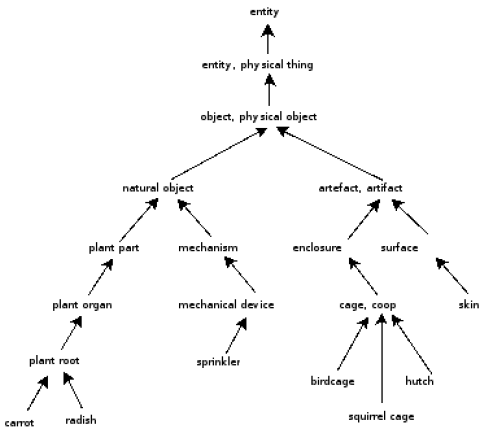
Po samostatném preprocessingu, nám zbývá najít vztah mezi slovy. Pro tuto bakalářskou práci budu využívat NLP techniku, kde pomocí lexiconu budu hledat tzv. sémantické vztahy [12]. Jedná se o vazby mezi významem slov nebo vět. Mezi nejčastější vazby patří Is-A (hyperonymum) nebo Part-Of (meronymum) viz definice 2.4.

2.6 WordNet

WordNet je největší lexikální databáze anglického jazyka. Tato databáze se často označuje jako tzv. lexicon. Lexicon [12] lze definovat jako komplexní výčet slov organizovaných jako databáze, mohou obsahovat lingvistické a sémantické informace. Lingvistické informace obsahují vědní disciplínu zvanou morfologie(tvarosloví), která se zabývá tvarováním slov, kde manipulujeme např. s předponami, příponami, koncovkami slov, kde základní stavební jednotkou slova je morfém. Sémantické informace obsahují synonymické vztahy, Is-a vztahy (hyperonymum) a různé konverze typu numerický znak na slovo.

Wordnet je oficiálně definován jako [22]: *Lexikální databáze anglického jazyka, která seskupuje podstatná jména, přídavná jména, slovesa, příslovce do synonymních vazeb zvaných synsets, které jsou mezi sebou propojeny sémantickými a lexikálními vztahy.* Druhy vztahů [22] jsou následující:

- Hyperonymní vztah: Vztahuje slova a fráze do Is-a hierarchie, která nám umožňuje získávat ze slova hyperonyma, hyponyma a kopyhonyma a také určovat relevantní sémantický vztah mezi slovy vzhledem k míře významové podobnosti.
- Meronymní vztah: Vztahuje slova do Part-of nebo Has-part hierarchie, která nám umožňuje získávat ze slov meronyma a holonyma.



Obrázek 2.4: Is-a vztahový strom.

- Troponymní vztah:Vztahuje pouze slovesa do vazeb, kde je seskupuje do podobně významových skupin.
- Antonymní vztah: Vztahuje pouze přídavná jména do sémantických vazeb jako antonyma, podobná přídavná jména lze najít jako “indirect antonyms” ve frameworcích, ale obvykleji jako synonyma.

2.7 JWNL

Java WordNet Library je dle vývojářů [23] API (Application Programming Interface) pro využívání služeb lexikální databáze Wordnet. Mimo datového přístupu umožňuje vytváření vztahových vazeb a morfologického zpracování dat. Nicméně morfologické programové rozhraní framework obsahuje, datové údaje, různá morfologická data si uživatel musí dodat sám.

Framework lze jednoduše naimplementovat pomocí mavenu nebo gradlu. Dále je ovšem třeba vložit Properties.xml soubor, kde je potřeba uvést cesta ke složce obsahující data z WordNetu, následně je ještě tento inicializační soubor potřeba implementovat do kódu. Dokumentace však není příliš dobře zpracovaná. Parametry, návratové datové typy uvádí pečlivě [23], avšak u jednotlivých metod je vysvětlení trochu chaotické ve spleti různých metod a datových typů, zde tedy uvádím nejdůležitější a nejpoužívanější.

Datový typ	Vysvětlení
IndexWord	Základní objekt reprezentující slovo nikoliv jako String, ale již součástí WordNet knihovny s veškerými dostupnými daty o tomto slovu a všech jeho možných interpretací a významů daného slovního druhu.
Word	Základní objekt reprezentující slovo podobně jako IndexWord se všemi dostupnými daty o slovu pro jeden slovní druh, ale je již specifičtější pro pouze jeden sémantický význam
Synset	Objekt reprezentující kolekci slov, které mají stejný čči podobný význam.
POS	POS reprezentuje objekt jako slovní druh slova.
PointerUtils	Objekt sloužící ke generování různých operací na Synset. Příkladem může být generování hypernym, antonym apod.
PointerTargetNode	Objekt sloužící jako uchování jednoho cílového uzlu stromu vygenerovaným pomocí PointerUtils a pro další práci s ním.
AsymmetricRelationship	Objekt reprezentující stav počatečního a cílového uzlu stromu/vztahu, kde nám hlavně umožňuje hledat divergentní bod ve vztahu.
Dictionary	Objekt sloužící jako abstraktní reprezentaci WordNet knihovny umožňující především volání metody getIndexWord().

Tabulka 2.1: Významné datové typy JWNL.

Název metody	Vysvětlení
initialize()	Metoda pro počáteční import WordNet databáze do frameworku.
Dictionary.getInstance()	Získání instance Dictionary.
getIndexWord()	Získání instance IndexWord, je třeba zadat String slova a slovní druh.
getSynset()	Získání Synsetu z objektu Word.
getWord()	Získání objektu Word ze Synsetu.
getDirectHypernyms()	Získání Hyperonyma ze slova anebo ze vztahu slov.
getDirectHyponyms()	Získání Hyponyma ze slova anebo ze vztahu slov.
getLemma()	Získaní Stringu, který reprezentuje název slova.
getGloss()	Získaní Stringu, který reprezentuje stručný popis významu slova.
findRelationships()	Získání objektu reprezentující vztah dvou slov jako stromovou datovou strukturu. Vztah může být např hyperonomický.
getRelativeTargetDepth()	Vrací vzdálenost dvou slov ve stromové struktuře.
getSense()	Vrací kolekci Synsetů pro dané slovo.

Tabulka 2.2: Významné metody JWNL.

Kapitola 3

Analýza

Využití technik Text miningu je jednou z klíčových oblastí, které použiji pro návrh tohoto frameworku. Práce se bude především zaobírat vytvořením jména souhrnně popisujícího vstupní slova. Vedlejším tématem je kategorizace slov pomocí technologií podobných jako u výběru rodiče. Framework má za cíl být lehce importovatelný s co nejmenší možnou měrou práce při zprovozňování a s co nejkratším nutným kódem pro zavolání potřebných metod. Jelikož rodiče obvykle chceme co nejméně obecného ale zároveň popisujícího všechna slova, není třeba nijak vstupovat do implementovaného kódu ke změně míry obecnosti. Následující analýza se tedy rozdělí do výše zmíněných skupin, tedy Parent Namingu a Kategorizaci slov.

3.1 Parent Naming

Prvním a hlavním úkolem tohoto frameworku je ze vstupních dat vybrat slovo, které souhrnně označuje tato slovní data a mělo by se jednat o název nejrelevantnější, nejpřesnější a co nejméně obecný. Pro řešení tohoto problému jsem se vydal dvěma cestami. První řešení závisí na na tzv. Relations, tedy vztazích, které navzájem mezi sebou slova chovají, a budu zkoumat jejich vzájemnou relativní hloubku. Druhé řešení bude také částečně záviset na stavbě Relations, ale hlavně na tzv. Sémantické příbuznosti, tedy jak jsou si slova navzájem podobná významem. Abychom ale mohli tato řešení použít, musíme vyřešit následující problémy:

- **Počet slov**:

Logicky by se dalo usuzovat, že čím víc dat (slov), tím lepší výsledky a lepší souhrnný název. Ne vždy to tak musí platit, jelikož velká skupina slov nemusí mít jednotný, podobný význam a s přibývajícím počtem slov může souhrnné slovo ztrácet na konkrétnosti a stávat se více obecným. Jak píše Mischenko

[14] ve své bakalářské práci, maximálně 7 prvků je vhodných do Meta modelu, tak dle jeho rady je toto číslo relevantní a počet slov by neměl přesáhnout tento počet, aby byl název rodiče co nejkonkrétnější.

- **Relevantnost slov**:

Na předchozí problém navážeme s tím, že použití tohoto frameworku postrádá smysl, jestliže do vstupu programu vložíme spolu nesouvisející slova. Příkladem mohou být slova dog a wall, která spolu evidentně nesouvisí a jako výsledek dostaneme whole, unit. Pro tato slova pravděpodobně nic lepšího vymyslet nejde, ale pokud budou často vyskytovat takové výrazy, tak všechny rodiče ponesou název whole, entity apod. Další problém může nastat, když mezi spolu souvisejícími slovy nalezneme jedno, které absolutně významově vybočuje. Je lepší přístup, kdy bereme v úvahu všechny, i odlišné, významy a dle nich vybíráme souhrnný název? Tento přístup může způsobit, že výsledek se stane, jak už bylo zmíněno, daleko obecnějším a méně přesným kvůli jednomu vybočujícímu slovu. Druhý způsob je do určité míry odchylku ignorovat. Pokud se tedy bude slovo vychylovat průměru sémantické podobnosti, nebude bráno v potaz ve výběru rodiče. Každá z mých metod parent namingu 3.1.1 a 3.1.2 bude tuto záležitost řešit jinak.

- **Slovo není v základním tvaru**:

Program přijímá slova jako String, abych tedy mohl pracovat s možnostmi frameworku JWNL, potřebuji String převést na IndexWord. K tomu je však potřeba slovní druh a základní tvar slova. K získání základního tvaru slova použijeme morfologickou techniku stemming, která je implementovaná v JWNL a je k dispozici pro základní slovní druhy.

- **Slovní druh**:

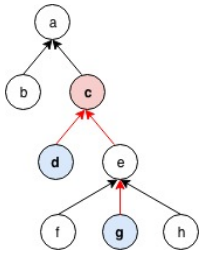
Framework JWNL umožňuje práci pouze se základními slovními druhy - podstatná jména, přídavná jména, slovesa a příslovce. To je ostatně i práce preprocessingu z Text Miningu, kde chceme jiné slovní druhy kvůli nedostatku kontextových informací vyloučit. Následujícím problémem je zjistit ze stringu, o jaký slovní druh se jedná, jelikož slova mohou mít homonymní formu a mít význam pro oba slovní druhy, jednoduše si zjistím velikost Synsetu pro každý slovní druh a ten nejrelevantnější vyberu.

Po vyřešení těchto problémů máme slova braná jako objekty v JWNL a můžeme s nimi pracovat. Pro oba nadcházející způsoby se ovšem budu řídit hierarchickou vazbou, kterou WordNet umožňuje - hyperonomickou. Díky této vazbě mohu získat vztah mezi dvěma slovy, který je reprezentován jako strom. Tato vazba je dostupná pouze pro podstatná jména, to by ale neměl být problém, když chceme souhrnně pojmenovat objekty, většinou se

vyskytují jako podstatná jména. JWNL nepodporuje konverzi slov dle jejich významů. WordNet ovšem vydal databázovou tabulku [2], která obsahuje sloveso a k němu podstatné jméno, které je významově nejbližší danému slovesu. Přídavná jména a příslovce bohužel nelze takto konvertovat. Způsob práce s těmito vztahy se však bude u následujících řešení lišit. Pro první způsob tedy budu pro Parent naming brát podstatná jména a slovesa, zbytek nebudu brát v potaz. Ve druhé metodě budu využívat veškeré slovní druhy a zde bude také rozebráno do hloubky, jak tato data využívám pro výběr rodiče.

3.1.1 Relativní hloubka hyperonymických vztahů

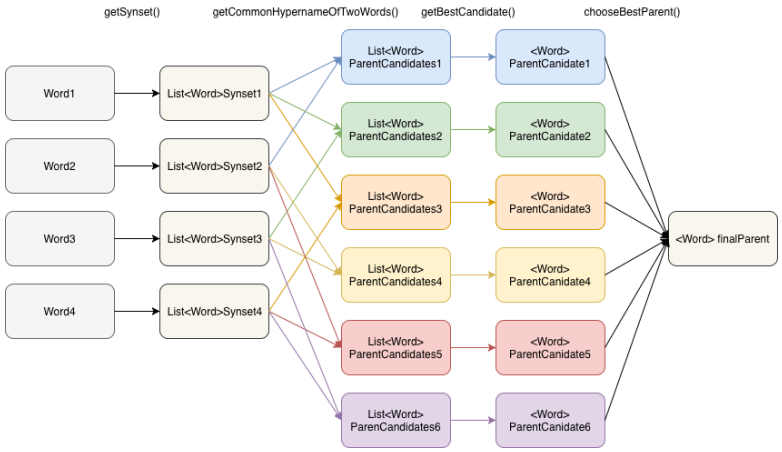
Toto řešení se bude zabývat analýzou hloubky vztahů generovaných přes hyperonomický vztah. Tento vztah popisuje hierarchii podstatných jmen spojených jako stromovou strukturu, ve které můžeme mezi dvěma uzly, reprezentovaných jako právě hyperonomický vztah, hledat souvislost. V tomto stromu mohu vzít společného rodiče jako nejkratší, označovanou jako shortest ancestral path [3]. Je logické, aby tento vztah měl co nejkratší cestu, jelikož chceme, aby souhrnný název byl nejkonkrétnější, čili nejblíže oběma uzlům a v hierarchii nad nimi. V grafu 3.1 je tato cesta červeně vyznačená mezi body d a g se společným parentem v bodě c s celkovou délkou hran rovno 3.



Obrázek 3.1: Shortest ancestral path

Tento přístup vyhledá nejlepšího společného rodiče pro 2 slova. Jak ale vyhledávat rodiče pro 3 a více slov? Nabízí se řešení, že jednoduše stromovým prohledáváním do šířky BFS nalezneme v tomto stromě společný uzel s nejmenším počtem hran a tento uzel bude výsledkem - parentem. V normální stromové struktuře by toto řešení bylo nejlepším a nejpřesnějším. Jenomže i přes obrovskou databázi slov a jejich vztahů ve WordNetu je skoro nemožné zachytit každý vzájemný vztah mezi slovy. Proto např. mezi slovy car a tire je nejbližším parentem slovo object. To by znamenalo, že i pro slova car, bus, bike a tire by dle této metody byl výsledkem také object. Jak tedy docílit, aby pro více slov nebylo vyhledávání příliš obecné?

Na diagramu 3.2 lze vidět proces popisující průběh této metody. Ze slov braných jako objekt IndexWord získám list všech slov daného slova s jeho významy - Synsets, ze kterých udělám jejich vzájemné kombinace po dvou a pro každou tuto dvojici získám jejich seznam potenciálně vhodných kandidátů



Obrázek 3.2: Průběh metody hledání rodiče.

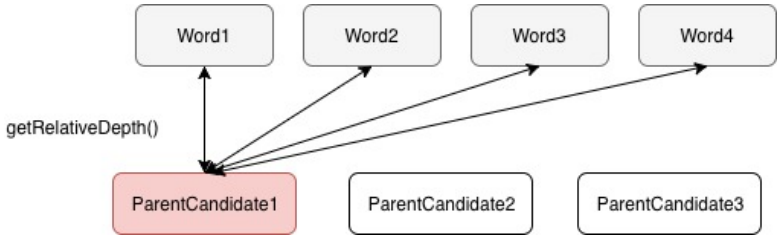
na post rodiče pomocí výše zmíněné metody shortest ancestral path. Z každé této sady chceme vybrat jednoho nejlepšího a nejrelevantnějšího kandidáta. To lze udělat právě pomocí relativní hloubky mezi dvěma uzly, tedy vzdálenosti hran mezi těmito uzly. U každého kandidáta si vyhledám hloubku vztahu ze slov, logicky chci, aby tato vzdálenost byla co možná nejmenší, hledám tedy její minimum a pro danou sadu mám rodiče. Zbývá tedy vyřešit, který z těchto finálních kandidátů nejlépe odpovídá pro souhrnný popis všech původních slov. Je jasné, že chceme nejpřesnější výraz a tedy relativní hloubka mezi uzly musí být nejmenší. Mezi čím ale budeme porovnávat? Zde uvádím seznam možností využívající relativní hloubky uzlu, které se zdají být k této problematice relevantní:

- **Kombinací původních slov a jednoho z parentu**:

Pro zvolené parent slovo zjistím jejich relativní hloubku mezi všemi původními slovy 3.3 a vzorcem 4 tak, že se vypočítá celková hodnota vzdálenosti definující podobnost souhrnného slova s původními. Toto číslo reprezentuje pomyslné relativní vzdálenosti ve stromové struktuře hyperonym. Na první pohled lze postřehnout, že nehraje roli četnost shodnosti rodičů, závisí čistě na skóre mezi rodičem a původními slovy. Největší výhodou tohoto přístupu je skutečnost, že porovnáváme výsledek přímo s počátečním zdrojem - tedy hledáme nejblíže popisující variantu. Další výhodou je, že pro takový přístup existuje spousta vzorců na obdobné vztahy a hledání nejmenší vzdálenosti. Nevýhodou může být fakt, že některé vztahy nemusí být registrované ve WordNet databázi, čili tento nedostatek může významně znehodnotit celkové hodnoty podobnosti u slov, které si jsou v některých aspektech podobné.

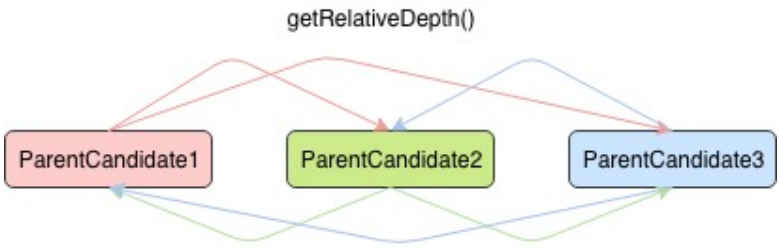
- **Mezi vzájemnými kombinacemi rodičů**:

Další možností je porovnávat relativní hloubky mezi vzájemnými kombinacemi parentů místo původních slov. Jelikož ale rodiče mohou být totožní, mohou



Obrázek 3.3: Zobrazení kombinací rodiče a původními slovy.

nastat případy, kdy se věc zkomplikuje. Ano, vícenásobné zastoupení slova může znamenat významnou shodu, a tedy nejvíce shrnujícího rodiče, jenomže takové výrazy jsou zpravidla nejvíce obecné, a tedy nicneříkající. Tím se dostáváme k nevýhodě tohoto přístupu, jelikož pro to abychom vybrali nejpřímějšího, ale všech slov určujícího rodiče bychom museli nějakou konstantou odebírat nevhodné rodiče, kteří jsou moc obecní. Problém s touto konstantou by byla její poměrně těžká nastavitelnost. Aby vždy fungovala nejlépe, musela by se hodnota přizpůsobovat a měnit dle situace. To by vyžadovalo mnoho hodin času strávených nad testováním a následným dokazováním. V případě, že bychom vymazali duplicitní rodiče, pak bychom mohli tento nežádoucí efekt minimalizovat. Konstanty by ale byly stále nutné pro správnou funkčnost a vybírání vhodného rodiče a společně s dalšími vzorci by tato metoda také fungovala. Navíc po redukci duplicit by nemusel být dostatek slov na vzájemné porovnávání mezi sebou.



Obrázek 3.4: Zobrazení kombinací mezi kandidáty.

Proto pro jednodušší a přímější implementaci s vhodnějším obsahem a podloženou teorií bude více vhodná první uvedená metoda, kterou detailně zpracuji v návrhové části 4.1.

3.1.2 Sémantické příbuznosti slov

Narozdíl od předchozí metody, kde jsme se rozhodovali dle relativní hloubky ve stromové struktuře, primární metodou jak rozeznávat nejvhodnějšího rodiče je jejich sémantická podobnost. Ze samotných slov se sémantických informací moc zjistit nedá, proto WordNet disponuje tzv. Glossem, neboli popisem všech významů slov a to krátkým shrnutím ve větě. Tato metoda se tedy bude

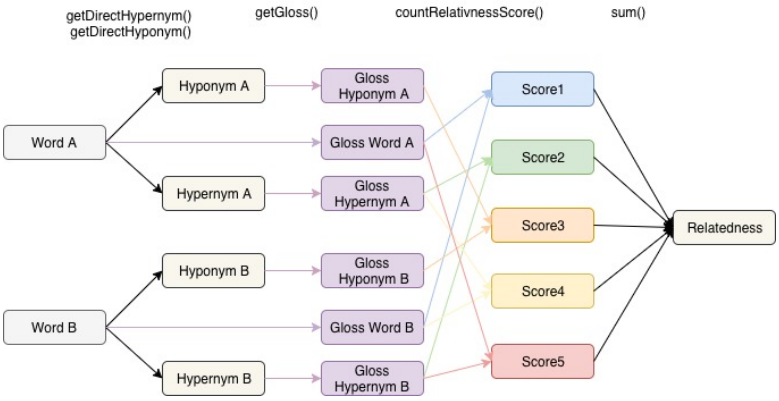
lišit oproti předchozí pouze v místě vybírání nejvhodnějšího rodiče. Proces dle obrázku 3.2 bude totožný až do bodu chooseBestParent, kde k výběru použijeme níže popsanou metodu sémantické podobnosti.

- Výpočet sémantické podobnosti dvou slov

V mnoha publikacích [13][zdroje na naky vypocty podobnosti] se podobnost dvou slov zjistí dle všech společných slov z Glossu. U hodně podobných slov tato metoda může být přesná, kde mají jedno, dvě slova v popisu společná, ale u lehce sémanticky vzdálenějších slov pravděpodobně nenajde nic. Tento problém přetrvává i u podobných slov, kde v Glossu nemusí být žádná společná slova. Tento problém je tedy způsoben nedostatkem dat ke zpracování relativnosti.

Studie [7] však popisují nový způsob, kde se nebude využívat jen dané slovo, ale také hyperonomické vztahy. Místo tedy samostatného Glossu z jednoho slova budeme využívat původní slovo, jeho hyperonymum a jeho hyponymum. Článek se dále zaměřuje na možné kombinace Glossů ze slov, které použít pro výpočet relativnosti. Na obrázku 3.5 lze vidět proces a postup výpočtu Kombinace pro výpočet podobnosti dvou slov A a B vypadá následovně ve dvojicích, lze si všimnout, že vzorec je reflexivní, tedy nezáleží na pořadí slov:

$$relatedness(A,B) = score(gloss(A),gloss(B)) + score(hype(A),hype(B)) + score(hypo(A), hypo(B)) + score(hype(A),gloss(B)) + score(gloss(A),hype(B))$$



Obrázek 3.5: Průběh metody hledání rodiče.

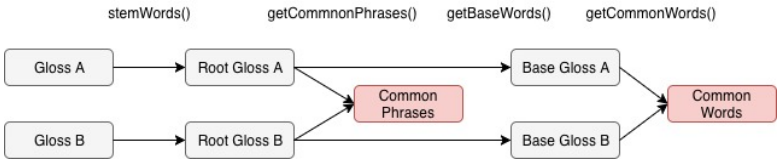
- Slovní druhy

V metodě 3.1.1 mohu využívat pouze slovesa a podstatná jména, ale Gloss obsahují veškeré slovní druhy, které nesou nějaký kontext, tedy podstatná jména, přídavná jména, slovesa a příslovce. Jelikož v této metodě budu také využívat hyperonomických vazeb, která sdružují pouze podstatná jména,

budu muset pro výpočet analogie vynechat tyto vazby a použít Gloss pouze daného slova a ne jejich hyperonym a hyponym, jestli se jedná o přídavné jméno nebo příslovce. Slovesa díky morfologické tabulce jsem schopen převést na podstatné jméno za účelem zvýšení objemu relevantních dat. S ostatními slovními druhy lze také s tímto přístupem pracovat, ale pouze pokud budou alespoň dvě podstatná jména nebo slovesa u vstupních slov, aby šlo vytvořit z těchto dvou slov kandidáty na parenta. Poté díky Glossu všech slov lze vybrat sémanticky nejbližší souhrnné slovo.

- Zpracování Glossu

Při porovnávání slov z jednotlivých Glossů se musí myslet na to, že slova opět nejsou v základním tvaru, proto je díky morfologické technice stemming převede slovo do základního stavu. Zároveň vyřadíme veškerá slova nenesoucí žádný kontext. Kupříkladu kdy oba Glossy obsahují spojku a předložku, nijak nespecifikuje společnou příbuzenskou vazbu, ba naopak může slova, které nejsou sémanticky relevantní označit jako podobná. Tato bezkontextová slova však tvoří slovní fráze, tedy seskupení slov či slovní obrat, který se pojí jako více slov za sebou. To dle [7] má mnohem větší váhu při porovnávání podobnosti při shodě nežli u samotných slov. Fráze bude mít hodnotu rovnou kvadrátu počtu slov. Tedy prvně zkontrolujeme případnou shodu vazeb a poté odstraníme veškeré slovní druhy nepodporované frameworkem JWNL, tedy slov nenesoucí žádný kontext.



Obrázek 3.6: Průběh zpracování Glossu

3.2 Text Categorization

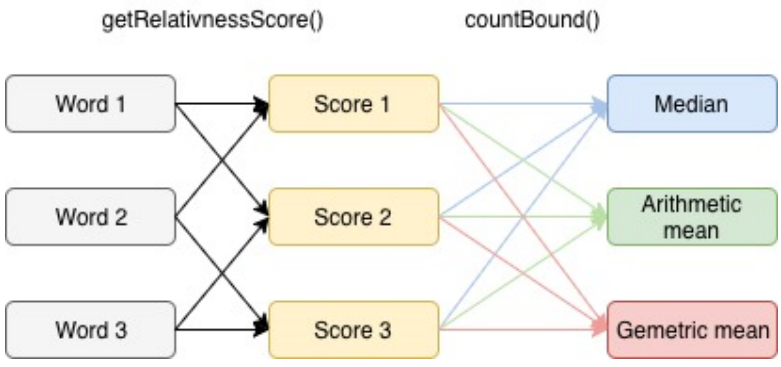
Dalším problémem, který však budu řešit jen okrajově je seskupování slov na základě podobnosti. Opět jako u Parent Namingu 3.1 lze tuto problematiku řešit dvěma způsoby a to přes hloubku vzájemných slov v hyperonomické vazbě 3.2.1 a přes sémantickou podobnost . Cílem tedy je rozdělit slova ze vstupu do dvou a více skupin. Záviset bude mimo jiné na počtu slov, jelikož dle Mischenka [14] je maximální počet slov v jedné kategorii rovný sedmi. Hlavní problém ovšem bude nalézt hranici, kdy jsou si slova významově podobná a kdy ne. V následujících technikách uvedu možné postupy.

3.2.1 Kategorizace dle hloubky hyperonomických vazeb

Tato metoda rozdělování čistě navazuje a využívá znalostí, které jsme rozebrali v 3.1.1. Slova ze vstupu budu také dávat do hyperonomického vztahu a přes relativní hloubku porovnávat jejich vzájemné kombinace a přes shortest ancestral path vybírat a sdružovat k sobě ta nejbližší slova. Vzdálenost ve stromě označující jejich příbuznosti bude určovat konstanta, která však bude pravděpodobně nestálá z důvodu uspořádání vztahů ve Wordnetu, kde některá slova, i když jsou ve stromu vzdálená, tak mají z lidského vnímání o dost blíž k sobě nežli slova, která mají od sebe minimální vzdálenost. Proto se asi díky tomuto handicapu nikdy nedočkáme perfektních výsledků, nicméně relativně dobrých lze dosáhnout.

3.2.2 Kategorizace dle sémantické příbuznosti

Tento způsob bude využívat výpočet sémantické podobnosti slov, jaký jsem použil v 3.1.2, tedy mezi slovy budu porovnávat vzájemnou podobnost. Problém je, že výstup této podobnosti je počet slov a frází, které mají porovnávaná slova v Glossu společná. Při jakém počtu tedy označit analogická slova? V kapitole 4 je vysvětlen vzorec, který používám na výběr nejvíce relativního slova z ostatních. Problematika je velice triviální - ve smyslu nejvyšší skóre znamená nejvíce souhrnné slovo. Možným způsobem jak získat hraniční číslo je udělat vzájemné kombinace slov po dvojicích a z nich vypočítat jejich sémantickou hodnotu podobnosti. Zde lze využít výpočet této hodnoty pomocí Glossu ze slova, jeho hyperonym, a hyponym detailněji popsané v 3.1.2. S těmito hodnotami se bude poté především pracovat, a to tak, že jednou z možností by bylo udělat aritmetický průměr, geometrický průměr nebo medián. Pro výběr nejlepší možnosti by bylo třeba veškeré způsoby řádně otestovat a zjistit parametry, které výběr nejvíce ovlivňují.

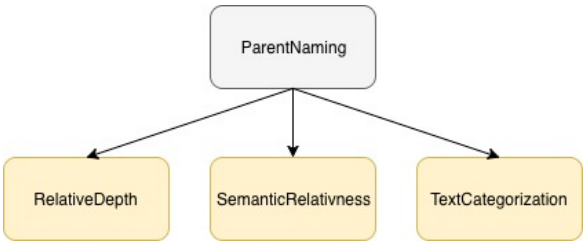


Obrázek 3.7: Možný průběh při kategorizaci.

Kapitola 4

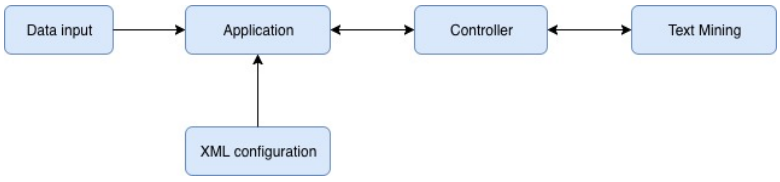
Návrh

Dle rozboru analýzy se framework bude dělit na dvě rozdílné části Parent Namingu, které budou fungovat odděleně, proto i každou část navrhnu odděleně i přes to, že oba přístupy budou využívat některé stejné komponenty. Ve třetí části poskytnu základní vysvětlení ohledně Text categorization, která není zcela dokončená, jelikož nebyla tématem této práce a bude sloužit spíše jako inspirace pro budoucí práce.



Obrázek 4.1: Rozdělení frameworku

Datový vstup se přijme v Aplikaci a tato data pošle do Controlleru, který drží instance veškerých implementovaných tříd z Text Miningu a umožňuje volání metod z modelu Text Mining, kde jsou rozdělené na níže uvedené části frameworku. Tento přístup je známý pod MVC - model, view, controller. V tomto případě je ochuzen o View, jelikož tento framework nemá vlastní uživatelské rozhraní, avšak při správné implementaci do jiného programu využívající UI bude splněn princip tohoto schématu.



Obrázek 4.2: Architektura frameworku.

Aplikace používá konfigurační soubor, který načte složku obsahující WordNet a vytvoří FileManager, který nám dovoluje pozdější import a využití

funkcionalit WordNetu pro framework JWNL. Součástí tohoto souboru [6] je i definice pro tzv. Morfologický procesor, který pomáhá transformovat slova do jejich základního tvaru. Na obrázku [4.3] vidíme část této definice, a to konkrétně chování převodu pro dané přípony za daných slovních druhů.

Příklad 4.1: JSON formát morfologické části.

```
<param value="net.didion.jwnl.dictionary.morph.
DetachSuffixesOperation">
    <param name="noun" value="|s=|ses=s|xes=x|
    zes=z|ches=ch|shes=sh|men=man|ies=y|"/>
    <param name="verb" value="|s=|ies=y|es=e|
    es=|ed=e|ed=|ing=e|ing=|"/>
    <param name="adjective" value="|er=|est=|er=e|est=e|"/>
    <param name="operations">
        <param value="net.didion.jwnl.dictionary
        .morph.LookupIndexWordOperation"/>
        <param value="net.didion.jwnl.dictionary
        .morph.LookupExceptionsOperation"/>
    </param>
</param>
```

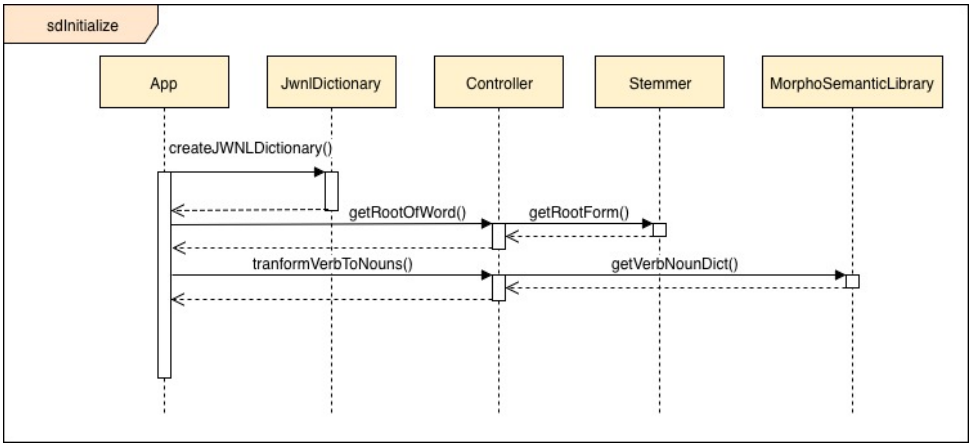
Z analýzy taktéž vyplynulo, že některým vzorcem budeme u obou metod pojmenování rodičů potřebovat vzorec, který nám umožní číselně vyjádřit hodnoty výsledků slovních kombinací. Pro oba případy využiji metodu součtu čtverců, která se u velmi podobné problematiky používá v [3].

$$d_j = \sum_{i=1}^{n} dist(A_j, A_i)$$

Postup od inicializace po převod Stringu na IndexWord je pro veškeré návrhy 4.1, 4.2 a 4.3 totožný a je zobrazen na sekvenčním diagramu 4.3. Datovým vstupem do tohoto frameworku bude jednotný List<String>, tedy List slov reprezentovaných jako String, který bude, jak již bylo zmíněno, třeba převést na IndexWord. Nejdříve však z každého slova získám jeho kořen provoláním Controlleru na Stemmer. Finální krok bude převedení sloves provoláním Controlleru na MorphoSemanticLibrary, kde převede na významově nejpodobnější podstatné jméno.

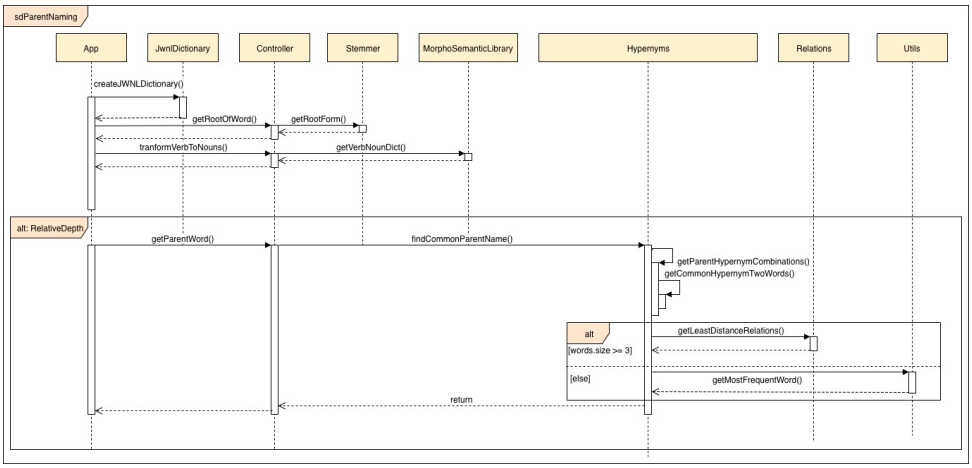
4.1 Návrh části relativní hloubky

Tato část se bude zabývat návrhem analýzy z 3.1.1, kde je cílem najít nejvhodnějšího rodiče za pomoci relativních hloubek v hyperonomickém stromě. V sekvenčním diagramu 4.4 lze vidět detailní průchod aplikace, kde Controller zavolá třídu Hypernyms, která nalezne rodiče všem dvojicím slov, z těch vybere jedno nejrelevantnější. Ze všech kandidátů je však třeba



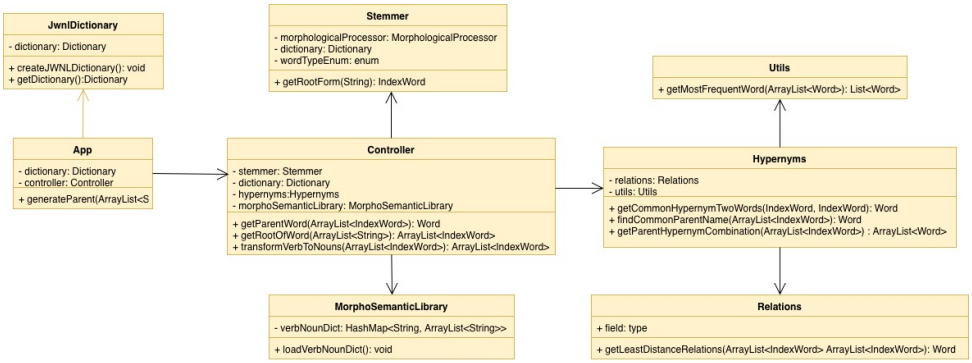
Obrázek 4.3: Sekvenční diagram převodu Stringu na IndexWord.

vybrat významově nejbližšího rodiče vzhledem k původním slovům. Je-li počet původních slov menší než 3, potom se aplikuje jednoduchý princip nejčastějších rodičů v pozici kandidátů, tedy stejně jako u předchozího výběru. V opačném případě přechází seznam kandidátů do třídy Relations, kde dle metody součtu čtverců se vybere nejmenší celková hodnota reprezentující nejbližší vazebnou kombinaci ve stromě.



Obrázek 4.4: Sekvenční diagram dle návrhu relativní hloubky.

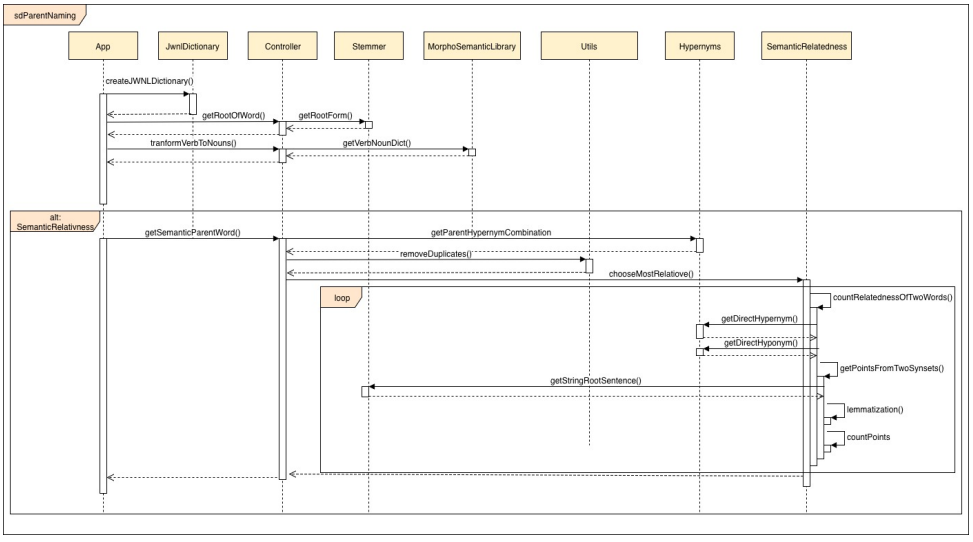
Pro detailnější náhled lze vidět na obrázku 4.5 strukturu programu a obsah tříd v class diagramu.



Obrázek 4.5: Class diagram dle návrhu relativní hloubky.

4.2 Návrh části sémantické podobnosti

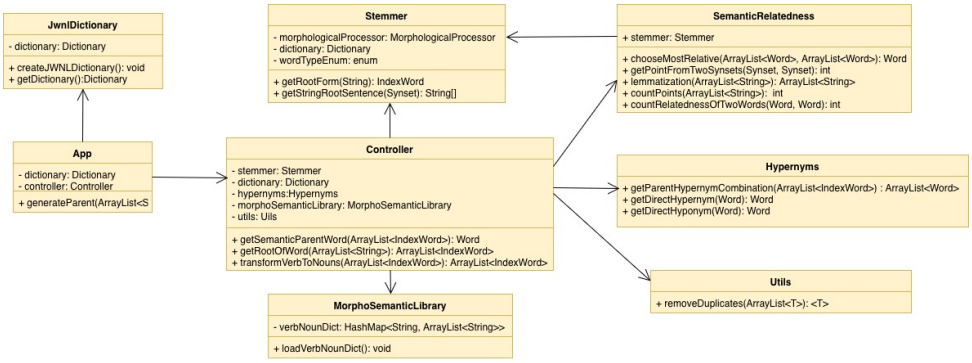
Návrh této části se zakládá na rešerši 3.1.2, kde jsou podrobně rozebrány techniky pro tento způsob. Controller zde opět slouží k provolání metod Text Miningu, na obrázku 4.6 je zobrazen sekvenční diagram popisující průběh. Prvně jsou tedy vybrány ve třídě Hypernyms kombinace rodiče všech vstupních slov a následně jsou odstraněny všechny duplicity ve třídě Utils. Důležitou částí se stává SemanticRelatedness, kde je třeba rozebrat jak původní slova, tak kombinace rodičů přes jejich významovou - sémantickou rovinu. Cílem je najít takového rodiče z kandidátů, aby jeho sémantická hodnota po porovnání s ostatními byla maximalizována.



Obrázek 4.6: Sekvenční diagram dle návrhu sémantické podobnosti.

Porovnání dvou slov a výpočet jejich podobnosti se tedy bude získávat z popisu jednotlivých slov, který dostanu jako String, který rozparsuji na

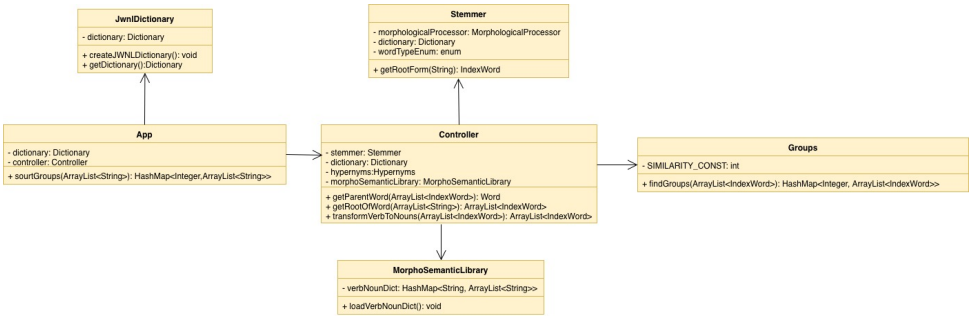
jednotlivá slova za pomoci mezer a následně za použití Stemmeru se získá kořen těchto slov a uloží se do datové struktury například pole Stringů. Dále slova lemmatizuji, odstraním veškeré nevyžádané znaky a bezkontextové slovní druhy. Zbývá už jen spočítat fráze a slova, které mají dvě slova společná. Tento proces bude zopakován pro každou kombinaci rodič - potomek a tyto hodnoty se budou přes součet nejmenších čtverců ukládat a vždy největší hodnota bude sloužit jako nejlepší kandidát a hranice pro ostatní. Na Class diagramu 4.7 jsou zobrazeny detaily a propojení jednotlivých tříd pro lepší orientaci.



Obrázek 4.7: Class diagram dle návrhu sémantické podobnosti.

4.3 Návrh části kategorizace textu

Tento návrh je spíše orientační pro budoucí práci jak již bylo uvedeno v analýze 3.2. Rozhodl jsem se navrhnout základ řešení 3.2.1, které je založeno na hloubce relativních vztahů. Kategorizují se tedy slova dle hloubky v hyperonomickém stromě, které dle konstanty SIMILARITY_CONST můžeme upravovat. Třída Groups bude klíčem k prohledávání vzájemných vztahů mezi slovy a jejich následným rozřazováním do různých tříd dle výše zmíněné konstanty. Class diagram 4.8 ukazuje propojení jednotlivých tříd pro celkovou funkčnost tohoto návrhu.



Obrázek 4.8: Class diagram dle návrhu sémantické podobnosti.

Kapitola 5

Implementace

Tato aplikace je napsaná v objektově orientovaném jazyce Java ve verzi 8 ve vývojovém prostředí IntelliJ IDEA jako Maven aplikace. Framework využívá knihovnu JWNL ke zprovoznění dat z databáze WordNet.

5.1 Celková struktura

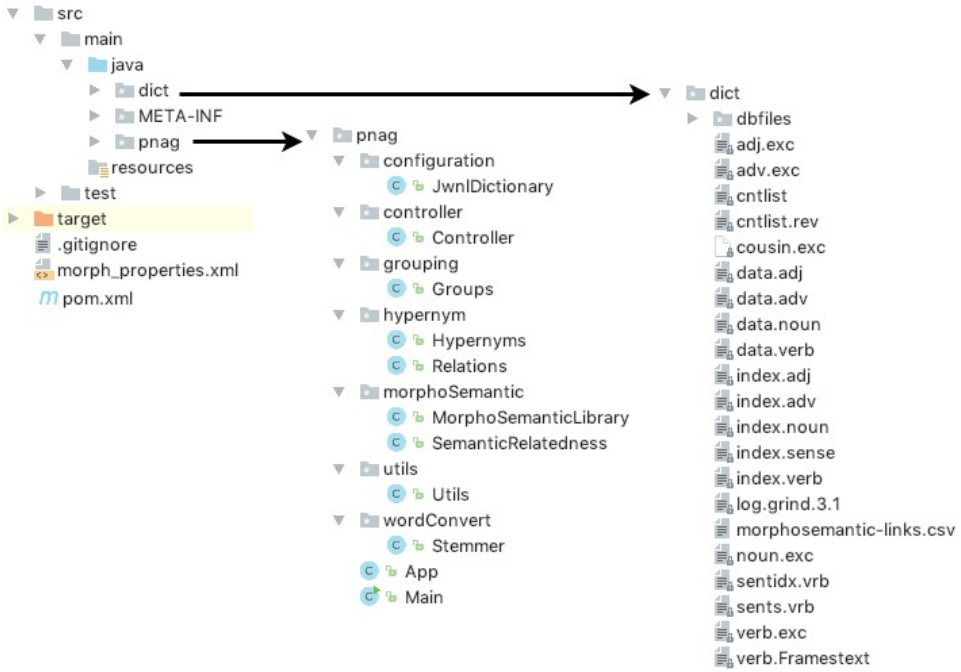
Jako každý Maven projekt i tento má klasickou strukturu rozdělenou do package a Java classes implementovaných dle návrhu. Pro celkové rozdělení programu jsou zde čtyři části, které jsou důležité pro základní pochopení programu.

- pom.xml - POM znamená Project Object Model a zastupuje každý Maven projekt, kde lze nastavit vše týkající se projektu. Zde se právě vytváří závislost na frameworku JWNL.

Příklad 5.1: Závislot na JWNL frameworku.

```
<dependency>
    <groupId>net.sf.jwordnet</groupId>
    <artifactId>jwnl</artifactId>
    <version>1.4_rc3</version>
</dependency>
```

- morph_properties.xml - Konfigurační soubor nutný pro JWNL primárně kvůli načtení dat WordNetu a sekundárně kvůli Morfologickému procesoru k lemmatizaci slov. Zadáváme zde cestu ke složce dict ke správné inicializaci frameworku. Není zde nutné měnit cestu ke složce z důvodu relativní cesty, která je u hodnoty zvolena.



Obrázek 5.1: Rozdělení a struktura programu

Příklad 5.2: Ukázka zadání cesty pro načtení souboru.

```
<dependency>
    <param name="file_manager" value="net.didion
    .jwnl.dictionary.file_manager.FileManagerImpl">
    <param name="file_type" value="net.didion
    .jwnl.princeton.file
    .PrincetonRandomAccessDictionaryFile"/>
    <param name="dictionary_path"
    value="src/main/java/dict"/>
    </param>
</dependency>
```

- dict - Složka obsahující veškerá data a údaje z WordNetu. Soubor morphosemantic-links.csv je dodatečně stáhlý soubor pro konverzi slovních druhů.

- pnag - Je složka obsahující tento program rozdělený do packagů a tříd dle zpracovaného návrhu.

5.2 Struktura jednotlivých částí

Tato část se zaměřuje na bližší prozkoumání jednotlivých tříd společně s náhledem na důležité a zajímavé části kódu společně s vysvětlením, které poskytnu v následujících fázích.

5.2.1 Inicializace

Pro inicializaci frameworku JWNL vkládáme jako argument konfigurační soubor pro načtení dat z WordNetu nacházející se ve třídě JwnlDictionary jak je vidět v ukázce 5.2.1.

Příklad 5.3: Načtení konfiguračního souboru.

```
JWNL.initialize(new FileInputStream("morph_properties.xml"));
this.dictionary = Dictionary.getInstance();
```

5.2.2 Konverze slov

Po zavolání metody na vytvoření souhrnného názvu slov se zpracovává v Controlleru, kde tedy dochází k provolání mezi jednotlivými třídami a k ovládání a uchovávání objektů tříd Text Miningu. Před samostatným výběrem rodiče je třeba převést String na IndexWord o což se stará třída Stemmer, kdy pro tento String je nejprve nalezen jeho kořen. Přes enum podporovaných slovních druhů tedy vyhledám kořen Stringu a naleznu pro něj odpovídající slovní druh a IndexWord jak je vidět v ukázce 5.2.2.

Příklad 5.4: Získání kořene slova.

```
private enum wordTypeEnum{
        noun,
        verb,
        adverb,
        adjective

    }

public IndexWord getRootForm(String word) {
    IndexWord finalWord;

    for(wordTypeEnum pos : wordTypeEnum.values()){
        finalWord = morphologicalProcessor.lookupBaseForm
        (POS.getPOSForLabel(pos.toString()), word);
        if (finalWord != null){
            return finalWord;
```

```
        }
    }
    return null;
}
```

5.2.3 Hledání rodiče

O tuto problematiku se u obou metod najití souhrnného slova stará třída Hypernyms, která slouží především pro výběr rodiče dvou slov. Pro demonstraci práce s hyperonymy jsem vybral ukázku 5.2.3 pro nejjednodušší případ, kdy hledáme přímého rodiče jednoho slova, tedy v hyperonomickém stromě jdeme o jeden uzel výš. Na podobné bázi funguje řešení pro dvě slova, která nalezne pomocí shortest ancestral path ve stromě 3.1. Poté, co vygenerujeme veškeré kandidáty na rodiče analyzují se mezi nimi vztahy ve třídě Relations. Výběr finálního rodiče ze dvou metod řeším přes relativní vztahy a vyberu z posledních dvou kandidátů vždy méně obecné slovo.

Příklad 5.5: Získání přímého rodiče.

```
public Word getDirectHypernym(Word word){
    PointerUtils pointerUtils = PointerUtils.getInstance();

    PointerTargetNodeList hypernyms = pointerUtils
        .getDirectHypernyms(word.getSynset());
    if (hypernyms.isEmpty()){
        return null;
    }
    PointerTargetNode pointerTargetNode = ((PointerTargetNode)
        hypernyms.get(0));

    return pointerTargetNode.getSynset().getWord(0);
}
```

Pro druhou metodu slouží především třída SemanticRelatedness kde z vybraných kandidátů porovnávám oproti původním slovům jejich významovou podobnost z jejich Glossu. Významnou složkou funkcionality je proces lematizace 3.6, kde z tohoto Glossu odstraňuji bezkontextová slova nemající význam, která by výsledné hodnoty vzorce mohla znevážit.

5.2.4 Kategorizace

Implementace kategorizace se nachází ve třídě Groups, kde využívám hyperonomické vztahy, které pro všechny dvojice slov vytvářím pomocí JWNL třídy RelationshipFinder v ukázce 5.2.4. Následně je procházím a řadím podle vzájemné relativní hloubky.

Příklad 5.6: Hyperonomický vztah mezi slovy.

```
RelationshipList list = RelationshipFinder
    .getInstance().findRelationships(candidate.getSynset(),
        origin.getSynset(), PointerType.HYPERNYM);
```

Samostatnou kapitolou se stává třída Utils, která slouží jako pomocná pro ostatní třídy a můžeme zde nalézt například funkci pro vyhledání nejčastějšího slova implementovaného pomocí streamu a lambda výrazů viz 5.2.4.

Příklad 5.7: Lambda výraz.

```
Integer max = wordFrequency.values().stream()
    .max(Comparator.naturalOrder()).get();
return wordFrequency.entrySet().stream()
    .filter(k -> k.getValue().equals(max))
    .map(Map.Entry:: getKey).collect(Collectors.toList());
```

5.3 Použití a implementace v jiném frameworku

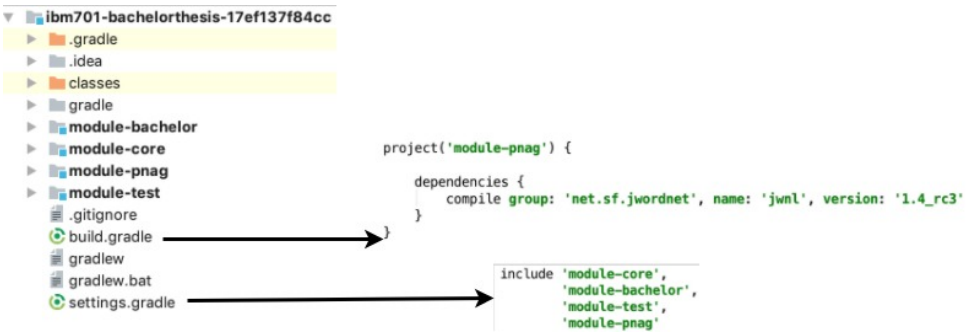
Framework byl navržen pro co nejjednodušší použití, a tak pro spuštění stačí importovat třídu App a z její vytvořené instance zavolat nabízené metody pro vyhledání společného sjednocujícího jména a nebo kategorizace slov a předat ArrayList<String> daných slov jako parametr viz 5.3.

Příklad 5.8: Použití vyhledání rodiče.

```
App app = new App();
ArrayList<String> words = new ArrayList<>();
words.add("apple");
words.add("cherry");
words.add("ham");
words.add("salami");
words.add("chicken");
words.add("pig");
words.add("rabbit");

String parent = app.generateParent(words);
```

Pro implementaci do Mischenkovy Java EE aplikace [14] je nutné využít gradle wrapper pro správu a rozdělení do modulů. Právě můj framework budu implementovat jako jeden z modulů. V hlavním adresáři se vytvoří složka např. “module-pnag”, která bude obsahovat celý framework. Pro začlenění mého modulu do aplikace je nutné v souboru settings.gradle přidat tento modul. Následně v build.gradle vložit novou definici pro tento modul s danými závislostmi. Tento postup je zachycen na obrázku 5.2.



Obrázek 5.2: Struktura vložení do Mischenkovy aplikace.

Kapitola 6

Testování

Testování je dnes již samozřejmostí při vývoji a před oficiálním vydáním produktu, kdy nám pomůže odhalit chyby a nedokonalosti programu. Účelem tohoto testování není hledat chyby v programu, ale ověřit, jak tento framework funguje za různých okolností a podmínek. Cílem je zjistit, která z metod Parent namingu je v jaké situaci lepší. Proto vezmu několik testovacích sad, které se rozdělí dle náročnosti a počtu testovaných slov. Náročností je míněna podobnost zadaných slov, kde bude hrát roli poměr slov podobných a cizích. Jelikož testování na data-setech bude zkoumáno ze subjektivního hlediska, tak tedy v další části budu za pomoci uživatelského testování představovat problematiku a výsledky frameworku, které budou objektivně posouzeny.

Dle Mischenka [14] je maximální počet slov vhodný do metamodelu roven sedmi, to bude hranice i pro testování, minimální počet slov budou tři, jelikož pro jedno slovo nebo dvě budeme dostávat stejné výsledky dle 4. Stěžejním bodem budou také slovní druhy, na které se také zaměřím v data-setech.

6.1 Testování na datasetech

V následujících tabulkách jsou uvedeny vstupní slova, subjektivní pohled náročnosti, výpis potencionálních rodičů shodných u obou metod a výsledný vybraný rodič od každého způsobu. Bude zde také výpis kombinací s výstupem výsledku vzorce. U metody s relativní hloubkou uvedu u každého kandidáta číslo symbolizující vzájemnou vzdálenost v hyperonomickém stromě, hledáme tedy co možná nejmenší číslo. Naopak u sémantického způsobu hledáme u každého kandidáta co největší hodnotu, která symbolizuje největší shodu slov. Pro přehlednost u výsledků červeně označím nejlepší skóre a napíšu výsledného rodiče. Zajímat nás tedy bude především porovnání obou metod, v jakých případech mají výhodu a kde mají slabiny i místa pro případná vylepšení. Mimo jiné účelem je i prozkoumat rozsah a různorodost slov.

- 3 slova

V případě vstupu tří slov se obě metody chovají velmi podobně, jelikož na výběr je většinou k dispozici pouze jeden relevantní rodič nebo dva velice podobní, mezi sebou o úroveň od sebe, v hyperonomickém stromě. Výsledky také hovoří jasně v prvním případě, kdy je prakticky správná jenom možnost carnivore (= masožravec), ve druhém jsou rodiče velmi podobní, to lze postřehnout i na ohodnocení. Lepší odpovědí by bylo pravděpodobně arthropod (=členovec), jelikož pavouk nespadá do hmyzu, ale tyto drobné niance lze přejít.

Origin Words	Parent Candidates	Depth Relations	Semantic relativnes
bite	entity	30000	3
dog	carnivore	9	150
cat			

Tabulka 6.1: První dataset tří slov s výsledky.

Origin Words	Parent Candidates	Depth Relations	Semantic relativnes
bug	insect	6	52
ant	arthropod	17	33
spider			

Tabulka 6.2: Druhý dataset tří slov s výsledky.

- 4 slova

Nyní lze spatřit podobný průběh jako u předchozích tabulek. Nyní u obou případů máme dva relevantní nálezy ze tří a taktéž u obou je nalezen ten vhodnější a všem slovům vybraný rodič velmi dobře odpovídá a zastupuje. Původní slova jsem se snažil volit nejednoznačně s drobnými odchylkami a přídavnými jmény tak, abych otestoval trochu složitější vztahy.

Origin Words	Parent Candidates	Depth Relations	Semantic relativnes
tea	beverage	18	41
coffee	food	34	15
sugar	entity	4000	2
mug			

Tabulka 6.3: První dataset čtyř slov s výsledky.

Origin Words	Parent Candidates	Depth Relations	Semantic relativnes
run	act	114	2
fast	entity	40000	4
quick	activity	78	6
jogging			

Tabulka 6.4: Druhý dataset čtyř slov s výsledky.

- 5 slov

Pro detailnější analýzu začíná být vhodná až tato varianta, kde v první tabulce jsou relativně podobné případy slov, a to konkrétně končetin a vnitřních orgánů. Jako vhodní kandidáti se jeví tyto zmíněné skupiny plus obecnější termíny. U ohodnocení se projevují rozličnosti obou metod. Jak je vidno metoda hloubky vztahů se zaměřuje především na co nejkonkrétnější názvy a obecné upozaďuje, tak metoda sémantické podobnosti má velice podobné hodnoty u všech kandidátů. Preferovaný rodič je sice shodný s první metodou, ale obecnější pojem má tendence být blíž. Zde by se jednoznačně hodil právě obecnější pojem body part (=část těla), a ne vybraný limb(=končetina). To je však způsobeno převažujícím počtem právě zmíněných názvů končetin, kde právě program upřednostní většinu. V druhé tabulce je možné vidět rozdílný

Origin Words	Parent Candidates	Depth Relations	Semantic relativnes
leg	limb	3	35
arm	external body part	35	24
hand	body part	66	29
stomach	internal organ	14	27
kidney			

Tabulka 6.5: První dataset pěti slov s výsledky.

výběr související s výše vysvětleným rozdílem u výběru vzhledem k obecnosti. První metoda vybrala conveyance (=transport), což je vcelku trefný výsledek pro původní slova, leč výběr je řízen především tak, že většina slov je blízko v hyperonomickém stromě blízko u sebe. Druhá metoda zvolila obecnější pojem object, který více popisuje slova jako celek.

Origin Words	Parent Candidates	Depth Relations	Semantic relativnes
car	conveyance	50	6
bus	physical entity	256	1
driver	instrumentailiy	64	3
wheel	object	193	9
seat			

Tabulka 6.6: Druhý dataset pěti slov s výsledky.

- 6 slov

U více slov jsem se zaměřil na otestování různých slovních druhů a jejich chování na rozdílné kandidáty. V první tabulce vidíme, že ostatní slovní druhy opravdu nejsou kontextově bohaté, tedy hlavní dopad mají podstatná jména a slovesa, která určila kandidáty, na něž mají dopad ostatní slovní druhy, ovšem pouze v sémantické metodě. V první tabulce u obou metod se vybral s jednoznačným skóre, nejlepší možný, i když až moc odborný rodič - placental (= určitá skupina savců). V této tabulce jsme se dočkali

Origin Words	Parent Candidates	Depth Relations	Semantic relativnes
elephant	placental	61	140
big	entity	5 000	14
giraffe	organism	156	18
land	abstraction	367	14
tall			
heavy			

Tabulka 6.7: První dataset šesti slov s výsledky.

opět různých výsledků u poměrně vzdáleně souvislých slov. Opět se projevila tendence u první metody brát spíše specifický objekt oproti obecnému u metody druhé. Avšak oba rodiče byli vybráni vzhledem k podmínkám dobře a sumarizují daná slova.

Origin Words	Parent Candidates	Depth Relations	Semantic relativnes
wash	artifact	184	9
dish	material	175	38
dishwasher	abstraction	418	4
plate	instrumentality	130	3
water	physical entity	403	5
clean	entity	60 000	12

Tabulka 6.8: Druhý dataset šesti slov s výsledky.

- 7 slov

Maximální počet slov ukazuje odlišnosti obou metod hledání rodiče, a to u obou tabulek. V té první jsem se snažil o využití jiných slovních druhů za účelem zvýšení přesnosti druhé metody a zde je právě vidět hlavní přednost oproti té první. Metoda relativní hloubky dbala jen na podstatná jména a jednoznačně zvolila cleaning implements(=úklidové nástroje). Metoda sémantických vztahů vzala kontext i z ostatních slov a vybrala obecnější souhrnný název condition(= stav). V druhé tabulce jsem se pokoušel nasimulovat různé

Origin Words	Parent Candidates	Depth Relations	Semantic relativnes
mess	condition	49	139
disorder	entity	50 000	5
dirty	cleaning implement	4	45
filthy			
mop			
broom			
grime			

Tabulka 6.9: První dataset sedmi slov s výsledky.

skupiny s hodně vzdáleným společným významem. Jak je vidno ze vstupních slov, jsou zde slova, která lze zařadit do několika konkrétních kategorií. Metoda relativních hloubek poměrně jednoznačně vybrala slovo placental (=

určitá skupina savců), přičemž toto označení sedí pouze na dvě ze sedmi slov. Tento rodič může být v hyperonomickém stromě nejblíže ke všem ostatním slovům, avšak z celkového hlediska je tento rodič nevhodně zvolený. U druhého způsobu vidíme lepší výsledek meat (=maso), který shrnuje pět ze sedmi prvků. Opět vidíme tendenci zlepšení v oblasti širšího spektra u této metody za přítomnosti většího počtu datových údajů. Pomocí deseti datových setů

Origin Words	Parent Candidates	Depth Relations	Semantic relativnes
apple	matter	412	24
cherry	food	240	54
ham	whole	352	20
salami	physical entity	519	2
chicken	meat	175	199
pig	placental	51	75
rabbit			

Tabulka 6.10: Druhý dataset sedmi slov s výsledky.

jsem ověřil funkčnost tohoto frameworku a využitelnost i pro ne zcela jasné případy. Obě metody ukázaly své přednosti, je ovšem zřejmé z výsledků, že větší potenciál má metoda sémantické příbuznosti slov, jelikož se osvědčuje při složitějších situacích, kde dokáže využít veškeré slovní druhy a jejich kontext. Naopak první metoda je vhodná pro triviálnější případy, kde velmi přesně vyhodnotí správného rodiče. Je také méně výpočetně náročná, jelikož procházíme jenom daná slova a jejich rodiče oproti sémantické metodě, kde procházíme všech slov hyponyma a hyperonyma.

Krátce také vyzkouším na dvou testovacích sadách Text categorization, které ač bylo vedlejším produktem této bakalářské práce tak v tabulce 6.11 rozdělilo slova dle významu bezchybně. Ve vedlejší tabulce už vidíme chybné rozdělení, to je však pravděpodobně způsobeno tím, že daná zvířata mají jistou souvislost s končetinami. Je zde však vidět potenciál pro budoucí práce s dobře navrženým základem.

1. skupina	2. skupina
dog	pasta
cat	bread
wolf	cookie

1. skupina	2. skupina
dog	hand
cat	
wolf	
arm	
leg	

Tabulka 6.11: Textová kategorizace.

6.2 Uživatelské testování

Toto testování bude sloužit jako reference pro toto řešení a také jako možné zdroje pro zlepšení a náměty budoucích prací. Koncept odpovídá

dotazníku [1],kde položím otázky ohledně využití Text miningu, využitelnosti parent namingu, kategorizace textu a ohledně ohodnocení funkcionality tohoto frameworku. Zpočátku se vysvětlí daná problematika, aby byl respondent v obraze. Na každou otázku ohledně projektu lze odpovědět čísly v rozmezí 1 (=určitě ne) až 5 (=určitě ano). Z těchto dat se získají statistické údaje, které se analyzují.

6.2.1 Dotazník

Uživatelům se po vysvětlené problematice položí osm otázek, na které se dá odpovědět pomocí výše zmíněného bodování v rozmezí 1 (=určitě ne) až 5 (=určitě ano). Další otázka je dobrovolná, kde byli požádáni o uvedení pěti slova jejich souhrnného slova, která později porovnám se svou generovanou hodnotou. Tato dobrovolná otázka slouží k nezaujatému porovnáni a analýzou mezi výsledkem tohoto programu a lidského uvažování. Příklady v osmi otázkách jsou brány z kapitoly 6, které se snaží volit spíše obtížná slova pro uživatele, abych otestoval tento framework při obtížných situacích a mohl získat objektivní výsledky z odpovědí uživatelů. Dotazníku se zúčastnilo celkem deset lidí, přičemž šest z nich vyplnilo dobrovolný úkol s vytvořením vlastního datasetu a výběrem vhodného rodiče. Seznam otázek:

- Je vhodným souhrnným slovem carnivore pro slova bite, dog, cat?
- Je vhodným souhrnným slovem activity pro slova run, fast, quick, jogging?
- Je vhodným souhrnným slovem limb pro slova leg, arm, hand, stomach, kidney?
- Je vhodným souhrnným slovem material nebo instrumentality pro slova wash, dish, dishwasher, plate, water, clean?
- Jsou slova dobře rozdělená do těchto 2 skupin - první dog, cat, wolf; druhá pasta, bread, cookie?
- Jsou slova dobře rozdělená do těchto 2 skupin - první dog, cat, wolf, leg, arm; druhá hand?
- Je Parent naming podle Vás užitečná vědní disciplína a má smysl do budoucna v tomto vývoji pokračovat?
- Je Text categorization podle Vás užitečná vědní disciplína a má smysl do budoucna v tomto vývoji pokračovat?

Odpovědi jsem převedl do tabulky 6.12. U každé otázky jsem uvedl aritmetický průměr a medián z výsledných odpovědí. Volitelný úkol od uživatelů lze vidět v tabulce 6.13, kde jsem uvedl jejich pět vybraných slov a souhrnné slovo jak od uživatelů, tak z výstupu tohoto frameworku.

	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	Aritmetický průměr	Medián
Q1	4	4	4	5	3	4	3	4	3	3	3,7	4
Q2	5	5	4	5	4	4	5	5	5	4	4,6	5
Q3	3	2	2	3	3	3	2	2	3	2	2,5	2,5
Q4	4	4	3	3	3	3	3	3	3	3	3,2	3
Q5	5	5	5	5	5	5	5	5	5	5	5	5
Q6	4	3	2	3	2	2	2	2	2	2	2,4	2
Q7	5	5	5	4	3	5	4	4	4	4	4,3	4
Q8	5	5	5	5	5	5	5	4	5	4	4,8	5

Tabulka 6.12: Výsledné odpovědi na otázky.

6.2.2 Vyhodnocení

Aritmetický průměr všech odpovědí je roven 3,8125 a celkový medián je roven 4, kdy 5 je možné maximum. Směrodatná odchylka, která nám říká "jak moc se od sebe hodnoty liší", je rovna 1,092. Tyto čísla lze brát jako úspěch, i díky skutečnosti, že jsem do otázek dával spíše nejasná a náročná slova pro výběr rodiče, abych zjistil reakce na výsledky při limitním použití frameworku. Za zmínku také stojí, že lidé vidí budoucnost a přínos této práce, kde Kategorizace textu měla o trochu vyšší úspěch než Parent naming.
Nyní se zaměřím na analýzu tabulky 6.13, kde porovnám jejich souhrnné slovo s tím, co vygenerovala tato aplikace. Na první pohled je vidět, že dvě hodnoty u sad D1 a D2 jsou naprosto totožné a u D6 jsou prakticky stejné kde uživatel vybral vehicle(=vozidlo) oproti motor vehicle(=motorové vozidlo) z frameworku. U datasetu číslo pět framework vybral správně food(jídlo), ale daná slova by e dala lépe shrnout jak je uvdeno jako sweet(=sladkost). Poslední dva datasety D2 a D3 jsou spíše testem abstraktního myšlení. Výsledek woody plant(=dřevěná rostlina) je velmi dobře uvedený název, kdy uživatel myslel obecnější pojem garden(=zahrada). Poslední parent artifact je bohužel velmi vzdálený od chtěné school(=škola).
Uživatelské testování ověřilo dobré výsledky tohoto frameworku, kdy uživatel

						User parent	My parent
D1	salad	steak	apple	cheese	bread	food	food
D2	beer	wine	juice	tea	coffee	beverage	beverage
D3	flower	bush	tree	grass	leaf	garden	woody plant
D4	notebook	desk	teacher	test	class	school	artifact
D5	candy	chocolate	lollipop	cake	pie	sweet	food
D6	car	truck	bus	bike	plane	vehicle	motor vehicle

Tabulka 6.13: Porovnání rodičů.

potvrdil, že vybraná slova se hodí jako souhrnný název. Dále na jejich vlastní dataset a rodiče vybralo stejného ve třech ze šesti případů totožného rodiče a v dalších dvou vybralo jiného avšak správného rodiče. Navíc, uživatelé potvrdili zájem o tuto problematiku a myslí si, že je to perspektivní obor do budoucích let.

Kapitola 7

Instalace

Tato práce byla vyvíjena na operačním systému mac OS, ale měla by fungovat i na ostatních platformách podporující Java 8.

Kroky isntalace:

- Vložte CD do mechaniky.
- Rozbalte komprimovaný PNAG.zip soubor
- V IntelliJ IDEA nebo v jiném IDE naimportujte framework jako projekt
- V případě implementace do jiného frameworku postupujte dle kapitoly 5.3
- Hotovo

Kapitola 8

Závěr

Navrhnout framework pro generování souhrnného názvu ze vstupních slov bylo hlavním cílem této bakalářské práce. Za pomoci technologií z oboru Text Miningu a frameworku JWNL jsem implementoval řešení, které danou funkcionalitu zprostředkovává. Z rešerše technologií, kde jsem se do hloubky seznámil s danou problematikou, a následné analýzy se jevilo jako vhodné řešení využít dvou různých metodik.

Toto rozdělení umožnilo zkoumat podobnosti slov z jiných úhlů za účelem porovnání obou řešení a výběru lepšího z nich. V první metodě se analyzují vztahy a jejich hloubka vytvořené WordNetem ve stromových strukturách dat. Ta druhá se zaobírá porovnáváním slovních významů, kde se využily pokročilejší techniky Text Miningu založené na vědeckých publikacích.

Obě metody již při analýze naznačily a ve výsledné implementaci při testování potvrdily své přednosti pro různé způsoby využití, kdy přispívají nekonvenčním řešením, které prohlubuje znalosti této perspektivní oblasti. Ostatní již existující řešení se touto problematikou zabývají jen okrajově a tento framework nabízí oproti nim funkční a ucelené metody pro vstup více než dvou slov různých slovních druhů. Limitaci současného řešení vidím u WordNetu, kde, ač přes obrovskou komplexnost této databáze, stále chybí mnoho dat, vztahů a popisujících údajů.

Vedlejším úkolem bylo najít způsob pro dělení slov podle jejich podobnosti a roztřídit je do jednotlivých skupin. Výsledkem je analýza a návrh dvou způsobů založených na návrhu parent namingu. Řešení využívající analýzu hloubky vztahů jsem v základní verzi implementoval, ale je myšleno spíše jako inspirace pro budoucí práce. Výsledné testování s uživateli potvrdilo funkčnost této aplikace, kdy uživatelé se většinou ztotožnili s výsledkem, a také ukázalo její velký potenciál i mimo technické obory.

8.1 Budoucí práce

Pro následné rozšíření a vylepšení se jako vhodné jeví především pokračovat ve vývoji kategorizace textu. Tato oblast má velký potenciál v nepřeberné škále softwarového využití, kterou potvrdilo i uživatelské testování. Klíčem bude využití mnou sepsané analýzy a návrhu společně s vyřešením vzorce a metody pro správné řazení do skupin. Důležitý krok bude nalezení konstanty, která umožní zvolit si míru podobnosti slov, což zredukuje značné výchylky a nepřesnosti mého řešení.
Jako další možnost se nabízí pokračovat v parent namingu a vylepšit moje řešení, ať už rozšířit ho o nový způsob, nebo spojit mé dvě odlišné verze a dosáhnout tak maximalizace výhod obou metod. Zároveň by šlo přidat i jiný vstup než pouze samotná slova, například související textový kontext k těmto slovům, aby bylo více relevantních dat pro následnou analýzu.

Příloha A

Seznam zkratek

Zkratka	Význam
PNAG	Parent Name Automatic Generation
JWNL	Java WordNet Library
KDD	Knowledge Discovery process
IE	Information Extraction
NLP	Natural Language Processing
IR	Information Retrieval
SD	Sekvenční diagram
UML	Unified Modeling Language
MVC	Model View Controller
JSON	JavaScript Object Notation
Q	Question
U	User
D	Dataset

Příloha B

Příklady

Příloha C

Literatura

[1] *Methods. usability.gov*, 2010. [Online; Dostupné 10-5-2019] https://www.usability.gov/how-to-and-tools/methods/index.html.

[2] *Standoff files*, 2010. [Online; Dostupné 10-5-2019] https://wordnet.princeton.edu/download/standoff-files.

[3] K. W. Alina Ene, *Cos 226 programming assignment wordnet*, 2006. [Online; Dostupné 15-5-2019] http://www.cs.princeton.edu/courses/archive/spring07/cos226/assignments/wordnet.html.

[4] C. Atkinson and T. Kühne, *The role of metamodeling in mda*, in Proc. UML 2002 Workshop on Software Model Engineering, 2002, pp. 67–70. Dostupné z: http://homepages.ecs.vuw.ac.nz/~tk/publications/papers/role-meta.pdf.

[5] C. Atkinson and T. Kuhne, *Model-driven development: a metamodeling foundation*, IEEE Software, 20 (2003), pp. 36–41. Dostupné z: http://ieeexplore.ieee.org/document/1231149/.

[6] T. Augustine, *Using java wordnet library*, 2015. [Online; Dostupné 13-5-2019] https://stackoverflow.com/questions/29329297/using-jwnl-java-wordnet-library-concerning-file-properties-xml.

[7] S. Banerjee and T. Pedersen, *Extended gloss overlaps as a measure of semantic relatedness*, in Ijcai, vol. 3, 2003, pp. 805–810. Dostupné z: https://pdfs.semanticscholar.org/63f2/2bd89534ba75c4c6d00d95fa50d07e6b97af.pdf.

[8] A. K. Dey, *Understanding and using context*, Personal Ubiquitous Comput., 5 (2001), pp. 4–7. Dostupné z: http://link.springer.com/10.1007/s007790170019.

[9] M. A. Hearst, *Untangling text data mining*, in Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics, ACL '99, Stroudsburg, PA, USA, 1999,

Association for Computational Linguistics, pp. 3–10. Dostupné z: http://portal.acm.org/citation.cfm?doid=1034678.1034679.

[10] D. M. Hilbert and D. F. Redmiles, *Extracting usability information from user interface events*, ACM Comput. Surv., 32 (2000), pp. 384–421. Dostupné z: http://portal.acm.org/citation.cfm?doid=371578.371593.

[11] A. Hotho, A. Nürnberger, and G. Paass, *A brief survey of text mining.*, in Ldv Forum, vol. 20, Citeseer, 2005, pp. 19–62. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.447.4161&rep=rep1&type=pdf.

[12] P. Kroha, *Text Mining*, 2015.

[13] M. Lesk, *Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone*, in Proceedings of the 5th annual international conference on Systems documentation, Citeseer, 1986, pp. 24–26.

[14] N. Mishchenko, *Aspektově orientovanỳ vỳvoj adaptivní struktury aplikace pro java ee aplikace*, B.S. thesis, České vysoké učení technické v Praze. Vypočetní a informační centrum., 2017.

[15] J. Mitchell and B. Shneiderman, *Dynamic versus static menus: An exploratory comparison*, SIGCHI Bull., 20 (1989), pp. 33–37. Dostupné z: http://portal.acm.org/citation.cfm?doid=67243.67247.

[16] A. Norcio and J. Stanley, *Adaptive human-computer interfaces*, tech. rep., NAVAL RESEARCH LAB WASHINGTON DC, 1988. Dostupné z: https://apps.dtic.mil/dtic/tr/fulltext/u2/a200930.pdf.

[17] E. Novodomská, *Chorvatsko-česká mezijazyková homonymie a paronymie*, D.S. thesis, Masarykova univerzita v Brně. Filozofická fakulta., note = Dostupné z: https://is.muni.cz/th/quvda/diplomova_prace.txt, 2017.

[18] R. Richardson and A. F. Smeaton, *Using wordnet in a knowledge-based approach to information retrieval*, (1995). Dostupné z: https://www.computing.dcu.ie/wpapers/1995/0395.pdf.

[19] B. Schilit, *Theimer,(1994) m. disseminating active map information to mobile hosts*, IEEE Network, 8.

[20] A.-H. Tan et al., *Text mining: The state of the art and the challenges*, in Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases, vol. 8, sn, 1999, pp. 65–70. Dostupné z: http://www.ntu.edu.sg/home/asahtan/papers/tm_pakdd99.pdf.

[21] S. Temple, *Mobile vs desktop traffic in 2019.* https://www.stonetemple.com/mobile-vs-desktop-usage-study/, 2019. [Online; Dostupné 18-5-2019].

[22] P. University, *About wordnet*, 2010. [Online; Dostupné 15-5-2019] https://wordnet.princeton.edu/.

[23] S. University, *Jwnl v1.1*. [Online; Dostupné 8-5-2019] http://web.stanford.edu/class/cs276a/projects/docs/jwnl/javadoc/overview-summary.html.

Příloha D

Obsah CD

Na přiloženém CD naleznete tyto soubory:

- bakalarska_prace - složka obsahující bakalářskou práci v pdf formátu, ale také všechny použité obrázky, excel tabulky, xml diagramy a zdrojové tex soubory
- PNAG.zip - komprimovaný soubor obsahuje implementaci frameworku