Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Metody pro výpočet sémantické podobnosti slov

Plzeň 2015 Ondřej Pražák

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 5. května 2015

Ondřej Pražák

Abstract

This thesis is focused on word-to-word semantic similarity measures. They are tested on English language. Its goal is to give the reader a general overview of existing word similarity measures, and to point to their main advantages and disadvantages. I also tried to develop some extensions of existing methods to get better results. Both main categories WordNet based and corpus based - distributional algorithms representatives are presented. Algorithms are evaluated on common word-to-word similarity datasets.

Abstrakt

Tato práce je zaměřena na metody výpočtu sémantické podobnosti slov. Algoritmy jsou testované v anglickém jazyce. Jejími cíli je poskytnout čtenáři ucelený pohled na současné metody v této oblasti a porovnat je. Také zde jsou představeny modifikace současných algoritmů za účelem vylepšení výsledků. Jsou zde představeny dvě základní kategorie těchto algoritmů: algoritmy založené na ontologii a algoritmy založené na distribuční hypotéze. Výsledky jednotlivých algoritmů budou vyhodnoceny na standardních datových kolekcích.

Poděkování

Děkuji Ing. Miloslavu Konopíkovi, Ph.D. za vedení mé bakalářské práce, za cenné rady a čas, který mi věnoval.

Obsah

1 Úvod

1.1 Formulace problému

Automatické určování sémantické podobnosti slov má mnohé využití v úlohách zpracování přirozeného jazyka, například zjišťování významu slov (Word sense disambiguation) nebo získávání informací (information retrieval). Tyto metody mohou být rozděleny do dvou základních kategorií.

Metody založené na znalostech Tyto metody určují sémantickou podobnost slov na základě jejich polohy v ontologii. Nejčastěji se pro tyto metriky používá WordNet. Tyto metody jsou detailně popsané dále v kapitolách 2 a 3.

Metody založené na korpusu Tyto metody se učí z velkého množství textu a sémantickou podobnost slov určují na základě výskytu v podobných kontextech. Lze je ještě dále rozdělit na metody, které uvažují kontext v celém dokumentu, a metody, které uvažují jen několik pozic v textu si blízkých slov. Detailnější popis těchto metod nejdete v kapitole 4.

U slov se určuje buď sémantická podobnost (similarity) nebo souvislost - do jaké míry spolu slova souvisí (používá se anglický termín relatedness). Abych uvedl nějaký ilustrační příklad, například slova automobil a silnice spolu celkem úzce souvisí, ale rozhodně si nejsou podobná. Tento rozdíl se často nebere v potaz a algoritmy určují jak podobnost, tak souvislost.

1.2 Metody předzpracování textu

Jako téměř každý problém z oblasti zpracování přirozeného jazyka i algoritmy pro určení sémantické podobnosti slov potřebují určité předzpracování textu. V této kapitole tedy stručně vysvětlím základní metody předzpracování textu.

1.2.1 Tokenizace

Tokenizace odstraní interpunkci a rozdělí text na informační celky. Podle potřeby se může jednat o věty, slova nebo entity (jednoslovné či víceslovné). [MRS08]

1.2.2 Stop-words

Odstranění slov, které pro daný problém nejsou důležité. V případě sémantické podobnosti se jedná o slova, která nenesou žádnou sémantickou informaci (například předložky a spojky). [MRS08]

1.2.3 Stemming

Určení „kořene“ slova. Stemmer nepracuje s kontextem, pouze ze slova odstraní předpony, přípony a koncovky. Výsledkem je kořen slova, který často ani není smysluplným slovem. [MRS08]

1.2.4 Lemmatizace

Určení základního tvaru slova (pro slovesa infinitiv, ohebné slovní druhy první pád, jednotné číslo atd.). Často používá plnou morfologickou analýzu, výsledek závisí i na kontextu slova. [MRS08]

1.2.5 POS tagging

Určení slovních druhů ve větě. Algoritmy pro POS tagging většinou používají strojové učení s učitelem, kde je programu nejprve poskytnuta dostatečně velká množina lidmi označeného textu.

1.3 Information Retrieval

1.3.1 Dokumenty ve vektorovém prostoru

Z množiny dokumentů je vytvořena matice (term-document matrix), která na pozici (i,j) obsahuje metriku relevance slova (viz dále) i k dokumentu j. S jednotlivými sloupci respektive řádky této matice potom můžeme pracovat jako s klasickými vektory. Například počítat podobnost dokumentů jako kosinovou vzdálenost mezi sloupcovými vektory této matice. V information refrieval se tento model používá hlavně ke zjištění relevance dokumentu k dotazu. Dotaz se převede na vektor (podobným způsobem jako jednotlivé dokumenty) a relevance dokumentu k dotazu se potom spočte jako kosinová vzdálenost vektoru dokumentu a vektoru dotazu. Více v [Liu07].

Metriky relevance slova k dokumentu

Četnost slova v dokumentu (Term Frequency) Udává, kolikrát se slovo vyskytuje v daném dokumentu. Někdy se používá spíše logaritmická četnost slova:

$$tf_(log) = 1 + \log tf \tag{1.1}$$

Inverzní četnost v dokumentech (Inverse Document Frequency) Četnost slova v dokumentech (document frequency - df) udává počet dokumentů, v nichž se vyskytuje dané slovo. Inverzní četnost klesá s rostoucí df.

$$idf(t) = \frac{N}{df(t)} \tag{1.2}$$

kde N je celkový počet dokumentů

Četnost slova v kolekci dokumentů (Collection Frequency) Četnost slova v kolekci dokumentů je součet tf přes všechny dokumenty.

tf-idf Kombinuje četnost slova v dokumentu a inverzní četnost slova v dokumentech. Jednu z možných *tf-idf* uvádí vzorec 1.3. [Liu07]

$$w(t,d) = 1 + \log tf(t,d) \times logidf(t) \tag{1.3}$$

2 Metody využívající WordNet

2.1 WordNet

WordNet [Pri10] je lexikální databáze inspirovaná teoriemi o lidské slovní paměti. Podstatná jména, přídavná jména a slovesa jsou spojována do synsetů (množin slov mající stejný nebo velmi podobný význam). Samozřejmě mnoho slov má více významů (homonyma) a takové slovo tedy patří do více synsetů. Každý synset vyjadřuje určitý koncept (význam). Synsety jsou spojovány sémantickými a lexikálními vazbami, které z nich tvoří sémantickou síť.

2.1.1 Struktura WordNetu

Základní relací Wordnetu je synonymum. Jak už bylo řešeno dříve, slova (respektive jejich jednotlivé významy) jsou sdružována do synsetů, do množiny slov s podobným významem. Jednotlivé synsety, kterých má anglický Wordnet asi 117 000, jsou spojovány dalšími relacemi. Každý synset navíc obsahuje glos, což je krátký popis významu slov v tomto synsetu. Slovo, které má více významů, se vyskytuje ve více synsetech, dvojice tvar slova a jeho význam je ve WordNetu unikátní.

2.1.2 Relace mezi synsety

Nejběžnější relací WordNetu je hyperonymum/hyponymum (relace vyjadřující nadřazenost a podřazenost). Hyperonymum je obecnější termín. Například zvíře je hyperonymem psa a pes je hyponymem zvířete. Jedná se o tranzitivní relaci, tedy když například ryba je zvíře a zvíře je živá bytost, tak i ryba je živá bytost. Kořenovým hyperonymem všech podstatných jmen je uzel entity. Toto jsou relace důležité pro sémantickou podobnost slov.

WordNet obsahuje spoustu dalších relací, například antonyma (opačné významy), meronyma (části celku) atd. Více informací viz [Fel05].

2.2 Základní metriky podobnosti slov

Výpočet sémantické podobnosti slov pomocí WordNetu je, zjednodušeně řečeno, založen na předpokladu, že podobná slova leží blízko sebe v hierarchii hyponym/hyperonym. Existuje celá řada principiálně podobných metod pro výpočet sémantické podobnosti, pro které budeme potřebovat některé pojmy. V první řadě je potřeba si ujasnit, že WordNet v hierarchii neobsahuje slova, ale synsety. Metriky založené na Wordnetu tedy nepočítají podobnost jednotlivých slov, ale počítají podobnost jejich významů (senses). Pokud tedy chceme počítat podobnost mezi slovy, tak musíme buď mít slova označená významy (sense tagging), nebo alespoň slovním druhem(part-of-speech tagging). Tyto taggery potřebují ke své činnosti kontext slova a lidmi označená data, ze kterých se učí. Pro výpočet sémantické podobnosti slov se většinou nepoužívají a sémantická podobnost mezi slovy se definuje jako největší podobnost mezi synsety, do kterých tato slova patří, jak uvádí rovnice 2.1.

$$sim(w1, w2) = \max_{s1 \in sense(w1), s2 \in sense(w2)} sim(s1, s2) \tag{2.1}$$

Pro definici základních metrik podobnosti slov definujeme hloubku uzlu synsetu, a to jako vzdálenost od kořene. Jelikož uzel může mít ve Wordnetu více rodičů, definujeme hloubku jako minimální vzdálenost od kořene. Další důležitou veličinou je délka cesty z S_1 do S_2. Posledním důležitým pojmem je nejbližší společný předek (lowest common subsummer, zkráceně LCS). Je to nejbližší (ve stromu nejníže postavený) předek obou synsetů, ve smyslu relací hyperonym.

2.2.1 Metrika délky cesty

Nejjednodušší metrikou podobnosti dvou synsetů je převrácená hodnota délky cesty mezi těmito synsety, tedy minimální počet uzlů mezi těmito synsety ve vztahu hyperonymum, hyponymum nebo instance.

$$sim_{Path}(s_1, S_2) = \frac{1}{pathlen(S_1, S_2)} \tag{2.2}$$

Algoritmus má pouze jeden singulární případ, a to když jsou slova synonyma a délka cesty je tedy nulová. V takovém případě ale stačí vrátit podobnost 1.

2.2.2 Wu-Palmerův algoritmus

WU-Palmerův algoritmus kombinuje délku cesty mezi synsety s hloubkou nejbližšího společného předka. Hloubka společného předka zde nese informaci o obecnosti vztahu. Když například budou mít 2 synsety nejbližšího společného předka kořen entity, jsou si určitě méně podobné než ty, které budou mít společného předka konkrétnějšího (například 2 primáti), i pokud bude délka cesty stejná.

$$sim_{W+P}(S1, S2) = \frac{2 \times depth(LCS)}{2 \times depth(LCS) + pathlen(S1, S2)} \tag{2.3}$$

Algoritmus nemá žádné singulární případy.

2.3 Metriky využívající míru informace

Sofistikovanější metriky pro výpočet podobnosti slov prostřednictvím WordNetu využívají míru informace konceptu (synsetu nebo významu slova).[1] Míra informace konceptu se spočítá jako:

$$IC(Syn) = -\log(P(Syn)) \tag{2.4}$$

Kde P(C) je pravděpodobnost, že náhodné slovo patří do tohoto konceptu, nebo je jeho hyponymem. V důsledku toho mají obecnější termíny menší míru informace a konkrétnější termíny naopak větší míru informace. Míra informace tedy udává, do jaké míry je koncept specifický. Pro určení pravděpodobnosti, že náhodné slovo patří do určitého konceptu, se používá textový korpus. Z textu je nejprve potřeba spočítat výskyty jednotlivých slov, respektive jejich významů, pokud jsou známé. Pro každý koncept spočítáme počet slov z korpusu, které do tohoto konceptu patří. Pokud je v textu určen význam slov (sense tagging), s každým slovem (respektive jeho významem) se zvýší počet výskytů pouze pro daný význam, zvýší se tedy počet u příslušného synsetu a všech jeho předků. Pokud není v korpusu určen význam slov, zvyšuje se četnost všem možným významům slova a jejich předkům. Počet výskytů se ale pak dělí počtem významů daného slova, aby nebyla mnohovýznamová slova zvýhodněna. Pravděpodobnost, že náhodné slovo patří do

[1] Anglický termín information content, zkráceně IC.

daného konceptu, se odhaduje jako počet slov patřících do tohoto konceptu dělený celkovým počtem slov v korpusu, jak uvádí vzorec 2.5.

$$P(Syn) = \frac{\sum_{w_i \in Syn \cup hyp(syn)} c(w_i)}{\sum_{w_i \in W} c(w_i)} \tag{2.5}$$

W slovník

hyp(Syn) množina všech hyponym synsetu Syn

c(w_i) četnost slova w_i v korpusu

Míru informace lze použít pouze pro podstatná jména a slovesa, protože ostatní slovní druhy nejsou ve WordNetu uspořádány hierarchicky. Jelikož jsou jejich hierarchie oddělené, nelze navíc těmito metodami počítat podobnost mezi podstatným jménem a slovesem. Navíc lze použít pouze relace typu hyponym-hyperonym opět z toho důvodu, že jediné mají hierarchickou strukturu. Z toho důvodu jsou tyto algoritmy schopné zjišťovat výhradně podobnost slov, nikoliv jejich souvislost. Více detailů v [Ped10].

2.3.1 Resnikův algoritmus

Nejjednodušším algoritmem založeným na míře informace je Resnikův algoritmus. Resnik definoval podobnost mezi koncepty jako míru informace jejich nejbližšího společného předka.

$$sim_{Resnik} = IC(LCS) \tag{2.6}$$

V případě, že míru informace slov nevyskytujících se v korpusu nastavíme na nulu, algoritmus nemá žádné singulární případy.

2.3.2 Linův algoritmus

Lin definoval podobnost dvou konceptů jako dvojnásobek míry informace nejbližšího společného předka ku součtu měr informace testovaných konceptů.

$$sim_{Lin}(C1, C2) = \frac{2 \times IC(LCS)}{IC(C1) + IC(C2)} \tag{2.7}$$

Problém může nastat, když mají oba testované synsety míru informace rovnou nule. Podobnost vyjde nekonečno a jediná možnost, jak tento problém řešit, je vrátit nulovou podobnost pro nedostatek informací.

2.3.3 Jing-Conrathův algoritmus

Vzdálenost mezi slovy je rovna součtu měr informace testovaných konceptů mínus dvojnásobek míry informace jejich společného předka. Podobnost dvou významů slova je potom převrácená hodnota této vzdálenosti.

$$jncDistance = IC(C_1) + IC(C_2) - 2 \times IC(LCS) \tag{2.8}$$

$$sim_{jnc}(C_1, C_2) = \frac{1}{jncDistance} \tag{2.9}$$

Algoritmus má problémy, když jncDistance vyjde 0 a podobnost tedy vychází kladné nekonečno. To se může stát, pokud ani jeden z uzlů nebyl v corpusu a míra informace obou je tedy nulová. V takovém případě by měl program vrátit jednoduše podobnost 0, protože nemá dost informací na to, aby podobnost spočítal. Případně může míru informace uzlů, které se v korpusu nevyskytují, nastavit na nějakou velmi malou hodnotu (menší než 1 / počtem slov korpusu).

Druhou možností je, že součet měr informace testovaných slov bude roven míře informace jejich nejbližšího společného předka. V tomto případě se jedná o největší možnou podobnost ve smyslu, jak tento algoritmus podobnost interpretuje. V tomto případě je nejsnazší nastavit maximální možnou podobnost a v případě, že je číslo větší (což nekonečno vždy je), vrátit tuto hodnotu.

2.4 Leskův algoritmus

Naprosto odlišný přístup k podobnosti slov přes WordNet zvolil Lesk. Definoval ji jako překrytí slov (word overlap) mezi jejich slovníkovými definicemi

(glosy).

2.4.1 Překrytí slov

Překrytí slov (word overlap) dvou textů je přímo úměrné počtu společný slov těchto textů. Pro výpočet překrytí slov existuje několik algoritmů.

Jednoduché překrytí (simple word overlap) Jedná se pouze o počet společných slov ve dvou textech; případně normalizovaný délkou těchto textů.

Překrytí s inverzní četností (IDF overlap) Stejné jako jednoduché překrytí, ale společná slova váží jejich inverzní četností v dokumentech.

Frázové překrytí (phrasal overlap) Frázové překrytí zvýhodňuje společné víceslovné fráze tak, že sčítá přes všechny společné fráze čtverce jejich délek.

$$overlap_{phrase} = \sum_{i=1}^{n} \sum_{m} i^2 \tag{2.10}$$

kde i je počet společných slov ve frázi a m je počet takto dlouhých společných frází.[PA08]

Leskův algoritmus používá frázové překrytí. Banerjee a Pedersen[Ban02] upravili původní Leskův algoritmus pro WordNet tak, že překrytí maximalizuje přes hyperonyma, hyponyma a další vazby daných slov.

2.5 Shrnutí metod využívajících WordNet

Metody využívající WordNet mají spoustu nedostatků. Hlavním nedostatkem je to, že nedokáží příliš dobře vystihnout podobnost mezi slovy jiných slovních druhů než podstatných jmen a sloves, protože pouze podstatná jména a slovesa mají ve WordNetu rozsáhlou hierarchickou strukturu.

Další nevýhodou je, že WordNet má relativně malé pokrytí, zvláště málo obsahuje instancí. A také se dnes už příliš nevyvíjí.

Výhodou WordNetu je to, že když informaci o daném konceptu obsahuje, velmi dobře vystihuje sémantickou podobnost.

Další důležitý fakt, který se dá chápat jako výhoda i nevýhoda (v závislosti na řešeném problému), je ten, že metody využívající WordNet vystihují sémantickou podobnost slov a nikoliv jejich souvislost. [2]

[2]To samozřejmě záleží na použitých relacích, například metrika délky cesty může používat relace meronym - část celku, a pak bude do jisté míry vystihovat i souvislost slov, ale stejně jen ve velmi omezené míře.

3 Dynamické ontologie

Nedostatek informací ve WordNetu a jeho statičnost vede k zamyšlení, jestli by nebylo lepší tyto metody implementovat nad některou z dynamicky se rozvíjejících automaticky generovaných ontologií. Informace uložené v těchto ontologiích sice nejsou tak přesné jako u ručně vytvářených ontologií (WordNet), ale informací obsahují mnohem více. Asi nejznámnější automaticky generovanou ontologií je ontologie DBPedie, která je vytvářená z infoboxů u jednotlivých článků na Wikipedii. Další zajímavou ontologií je pak YAGO, které kombinuje dynamicky se rozvíjející informace stažené z Wikipedie a GeoNames s lépe strukturovanými daty extrahovanými z WordNetu.

3.1 Ontologie

Ontologie je formální deklarativní popis určité problematiky. Ontologie je složena z definic pojmů (glosář) a vztahů mezi těmito pojmy. Ontologie obsahuje 4 základní typy entit:

1. **jedinec (instance)** je určitý objekt z reálného světa. Může být konkrétní i abstraktní.
2. **třída** je množina jedinců s určitými společnými rysy.
3. **atribut** je vlastností určitého jedince.
4. **vazba** je symetrické nebo nesymetrické spojení dvou jedinců.

3.1.1 RDF/OWL - Jazyk pro popis ontologií

RDF (Resource Description Framework) slouží k popisu internetových zdrojů. K definici jednotlivých vlastností používá trojici podmět, vlastnost a hodnota[1]. K identifikaci zdrojů RDF používá URI (Uniform Resource Identifier). Jedná se o W3C standard.

[1] V angličtině se používají termíny object, predicate a subject (podmět, přísudek a předmět). Trojce totiž odpovídá těmto větným členům ve větě v přirozeném jazyce.

N-Triples Formát pro uložení RDF dat. Každá trojce je uložena na jednom řádku. Podmět, vlastnost a hodnota jsou odděleny mezerou.

OWL (Web Ontology Language) je nadstavbou nad RDF a slouží k definici ontologií.

Pro více informací o RDF a OWL viz [W3S14] nebo [W3C14].

3.2 DBPedie

DBPedie extrahuje data z Wikipedie a mapuje je na ontologii vytvořenou z kategorií Wikipedie. Tato ontologie obsahuje:

- 685 tříd (DBpedia 3.9: 529)
- 1 079 objektových vlastností (relací mezi instancemi) (DBpedia 3.9: 927)
- 1 600 prostých vlastností (číslo, text) (DBpedia 3.9: 1 290)
- 116 speciálních vlastností (DBpedia 3.9: 116)

Na tuto ontologii DBPedie mapuje 4,22 milionu instancí. [JL15]

DBpedie sice obsahuje mnoho instancí, ale klasifikace do tříd není příliš rozsáhlá (685 tříd, což není mnoho). Její použitelnost pro výpočet sémantické podobnosti slov je proto diskutabilní.

3.3 YAGO

Ontologie YAGO vznikla spojením ručně vytvářeného WordNetu, který má preciznější strukturu, a dat extrahovaných z Wikipedie a GEONemes, kterých je podstatně více. Obsahuje více než 10 milionů entit klasifikovaných do asi

350 000 tříd a více než 120 milionů fakt o těchto entitách, která mezi nimi definují relace(YAGO2). Klasifikace je velmi rozsáhlá, proto by tato ontologie mohla být zajímavá pro určování podobnosti slov pomocí metod podobných těm, jaké se používají u WordNetu.

Aktuální verzi YAGO2 lze získat ve dvou formátech a to turtle a tsv (tab separand values – jednoduchý formát podobný N-Triples. Objekt, predikát a subjekt jsou odděleny tabulátorem, fakta koncem řádku).

YAGO 2 se skládá z následujících souborů:

- **Schema** – obsahuje informace o všech relacích, které v YAGO existují.
- **YAGO types** – obsahuje seznam instancí a jejich zařazení do kategorií ontologie.
- **YAGO transitive types** – obsahuje zařazení instancí do všech kategorií, tedy explicitně uvádí i nadřazené kategorie.
- **YAGO taxonomy** – obsahuje celou hierarchii tříd - strukturu ontologie.
- **YAGO Simple taxonomy** - řidší klasifikací do tříd, ta pro naše účely ale není zajímavá, protože pro určování podobnosti potřebujeme právě co nejpestřejší klasifikaci do tříd.
- **YAGO facts** – obsahuje všechny relace mezi instancemi, mohly by být zajímavé.
- **YAGO labels** – obsahuje ke všemu textové popisy, velmi důležitá část.
- **YAGO literal facts** – obsahuje data o instancích, která nejsou referencí na další instance, ale jsou definována pouze hodnotou. Pro sémantickou podobnost nejspíš nezajímavé.
- **YAGO WordNetIDs** - mapování uzlů z ontologie YAGO na synsety WordNetu.

Dále už YAGO obsahuje pouze geografické informace, tedy třídy, geografické entity a relace mezi nimi. [SKW07]

4 Distribuční sémantika

Druhou skupinou metod pro výpočet podobnosti slov jsou metody distribuční. Ty jsou založené na distribuční hypotéze, která říká, že dvě slova jsou si podobná, pokud se vyskytují v podobných kontextech. Každé slovo se převede na mnohorozměrný vektor, který popisuje kontext daného slova. Podobnost dvou slov se poté vypočte jako podobnost jejich vektorů. K tomu je možné použít různé metriky podobnosti vektorů (např. korelace, eukleidovskou vzdálenost), nejčastěji se však používá kosinová vzdálenost.

kosinová vzdálenost dvou vektorů se spočítá jako kosinus úhlu mezi těmito vektory, jak uvádí vzorec 4.1.

$$cos(A, B) = \frac{\sum_{i=0}^{N} A_i \times B_i}{\sqrt{\sum_{i=0}^{N} A_i^2} \times \sqrt{\sum_{i=0}^{N} B_i^2}} \tag{4.1}$$

4.1 Základní metody

Základní metody používají matici slov v dokumentech, stejnou, jaká se používá v IR, se stejnými metrikami relevance slova k dokumentu. Podobnost slov počítají jako vzdálenost mezi jejími řádkovými vektory, které reprezentují jednotlivá slova. Taková matice je ale velmi rozsáhlá a většina obsažených informací není pro reprezentaci významu důležitá. Proto se často používají metody pro redukci dimenze této matice (například SVD, viz dále).

Další možností je použít matici kontextů slov, která místo příslušnosti slova k dokumentu udává, jak často se slovo objevuje v kontextu jiného slova. Jako kontext se bere nějaké okolí, například 10 nejbližších slov. Pro hodnoty jednotlivých položek v matici se používá například PPMI.

4.1.1 PMI a PPMI

Vzájemná bodová informace (pointwise mutual information) PMI udává, jestli (a jak výrazně) se 2 slova vyskytují blízko sebe více, než by se vyskytovala, kdyby byla nezávislá.

$$PMI(w1, w2) = \frac{P(w1, w2)}{P(w1) \times P(w2)} \tag{4.2}$$

Kladná PMI (positive PMI, PPMI) pak pouze nahradí záporné hodnoty nulou.

$$PPMI(w1, w2) = \begin{cases} PMI(w1, w2) & \text{pokud PMI(w1,w2)} \geq 0 \\ 0 & \text{pokud PMI(w1,w2)} < 0 \end{cases} \tag{4.3}$$

4.2 LSA

LSA (Latent semantic analysis) je asi neznámější algoritmus pro sémantickou podobnost slov založený na množině dokumentů. LSA nejprve sestaví matici výskytů slov v dokumentech, pro vážení jednotlivých položek se většinou používá tf-idf. Na tuto matici se potom použije rozklad na singulární čísla (Singular Value Decomposition - SVD) [Liu07]. Tento rozklad rozloží matici A na 3 matice:

$$A(m \times n) = U(m \times r)\Sigma(r \times r)V(n \times r)^T \tag{4.4}$$

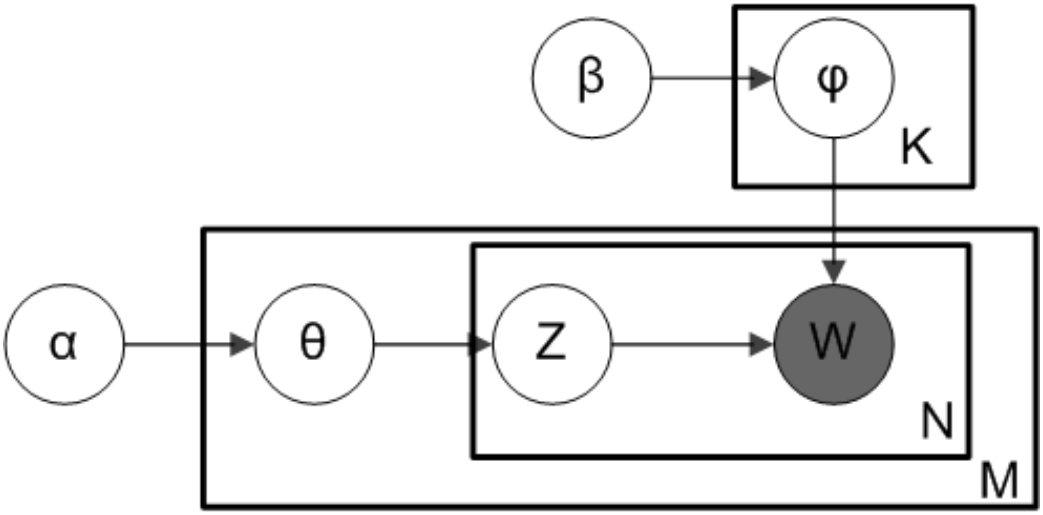
kde U a V jsou ortogonální matice vlastních vektorů matice $A \times A^T$, respektive $A^T \times A$ a Σ je diagonální matice, která má na diagonále singulární čísla (kladná odmocnina z vlastního čísla) těchto dvou matic seřazená sestupně. Dimenze matice A se potom sníží na hodnotu k odebráním $n - k$ posledních řádků a sloupců z matice Σ a stejného počtu sloupců z matic U a V. Odebráním posledních sloupců se ztratí nejmenší možná informace, protože vypouštíme nejnižší hodnoty Σ, které udávají váhu odpovídajících složek vektorů.

$$A_k(m \times n) = U_k(m \times k)\Sigma_k(k \times k)V_k(n \times k)^T \tag{4.5}$$

Řádky matice U jsou redukované k-rozměrné vektory reprezentující jednotlivá slova. Sémantickou podobnost dvou slov vypočítáme jako kosínovou vzdálenost jejich vektorů (případně můžeme použít jinou metriku pro výpočet podobnosti 2 vektorů). Řádky matice V jsou vektory reprezentující jednotlivé dokumenty, které se využívají v IR, pro výpočet sémantické podobnosti slov ale nejsou zajímavé. [Liu07]

4.3 LDA

LDA hledá skrytá témata v dokumentech. K tomu využívá Dirichletovo rozdělení pravděpodobnosti. Modeluje dokument jako rozdělení témat a téma jako rozdělení slov.



Obrázek 4.1: Grafické znázornění LDA

1. Pro každý dokument $D_M \in D$ vyber multinomiální rozdělení témat $\Theta_m \sim Dirichlet(\alpha)$
2. Pro každé téma vyber multinomiální rozdělení slov $\Phi \sim Dirichlet(\beta)$
3. Pro každou pozici $w_{i,j}$ v korpusu:

(a) vyber téma $z_{i,j}$ z pravděpodobnostního rozdělení Θ_i

(b) vyber slovo $w_{i,j}$ z pravděpodobnostního rozdělení $\Phi_{z_i,j}$

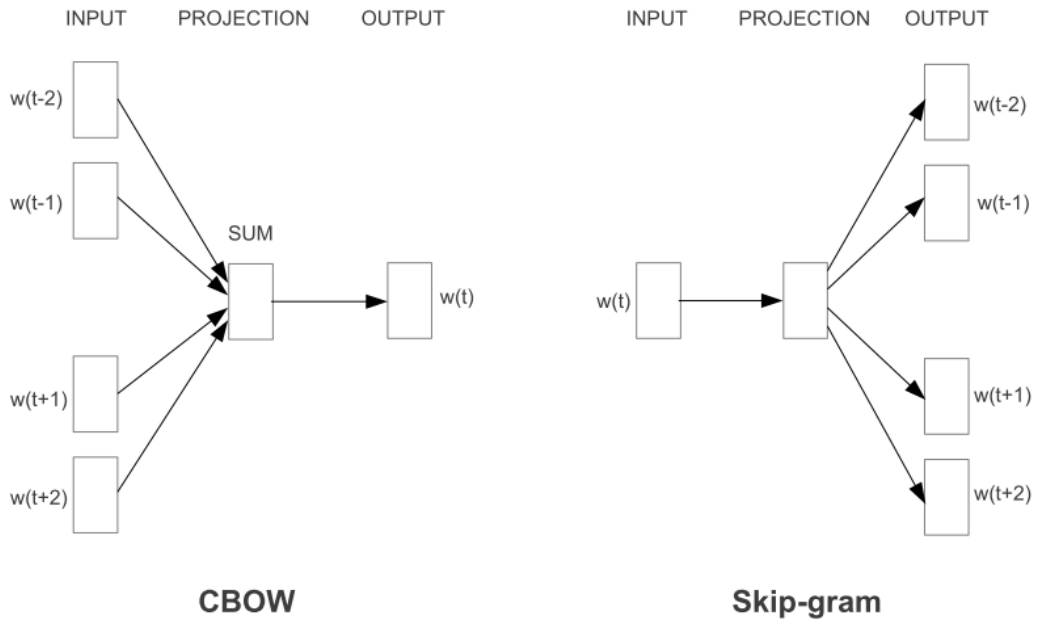
Význam slova lze pomocí LDA reprezentovat jako vektor pravděpodobností příslušnosti slova k jednotlivým tématům.

$$\omega(w) = \{P(w|z) : 1 \leq z \leq K\} \tag{4.6}$$

Více detailů o LDA viz [Wik15] nebo [Ble+03]

4.4 Word2Vec

Word2Vec používá pro převod slova na vektor neuronovou síť s logaritmicky lineárními modely. Jedná se o statistický jazykový model. Vstupem neuronové sítě je skupina n - 1 vektorů velikosti slovníku, kde každý vektor obsahuje právě jednu hodnotu 1 a zbytek jeho položek je roven 0 (model 1 k N). Jedná se o continuous bag-of-words (CBOW) model a skip-gram model. CBOW v každé iteraci určuje aktuální slovo na základě kontextu určitého předem nastaveného rozsahu, zatímco skip-gram naopak určuje kontext na základě aktuálního slova. Hlavní výhodou oproti dřívějším distribučním algoritmům je mnohonásobně vyšší rychlost trénování. Díky tomu lze vytvořit (v „ruzumném“ čase) model s použitím mnohonásobně větších korpusů a hlavně proto dosahují tyto modely výrazně lepších výsledků než jejich předchůdci. [Mik+13a]



Obrázek 4.2: Architektura modelů CBOW a skip-gram. Obrázek pochází z [Mik+13a]

4.5 GloVe

Model závisí na společných výskytech v celém trénovacím korpusu. Model je trénován na matici společných výskytů slov. Hlavní myšlenka vychází z pozorování, že podíl pravděpodobností společných výskytů slov má vliv na význam slova.

Ve fázi učení GloVe vytváří vektory slov pomocí regrese tak, aby skalární součin těchto vektorů byl roven pravděpodobnosti společného výskytu příslušných slov. Pro odvození viz [PSM14].

5 Vyhodnocení algoritmů

5.1 Testovací data

Pro vyhodnocení úspěšnosti algoritmů pro výpočet sémantické podobnosti slov se používají datasety s ohodnocením podobnosti, jak je ohodnotil člověk (respektive skupina lidí). V této práci byly použité následující datasety:

5.1.1 Rubenstein Goodenough test

Asi nejznámějším datasetem je Rubenstein Goodenough test[RG65]. Tento dataset obsahuje 65 dvojic složených pouze z podstatných jmen ohodnocených podobností. Pro objektivní porovnání algoritmů je příliš malý.

5.1.2 Word Similarity 353

O něco rozsáhlejší dataset WordSimilarity-353[Gab02] obsahuje 353 dvojic slov ohodnocených mírou podobnosti nebo souvislosti (relatedness). Tento dataset obshuje podstatná jména, slovesa a přídavná jména. Pro tento dataset existuje rozdělení na 2 datasety, jeden vyjadřuje míru podobnosti slov, druhý míru souvislosti.

5.1.3 MTurk

Dataset MTurk [Rad10] obsahuje 287 párů podstatných jmen ohodnocených mírou jejich souvislosti. Páry slov hodnotilo 10 lidí na stupnici od 1 do 5.

5.1.4 Rare Words

Tento dataset, vytvořený na Standfordské univerzitě, obsahuje 2034 párů slov, která nejsou používaná příliš často. [LSM13].

5.1.5 MEN dataset

Obsahuje 3000 párů slov, ohodnocených podle míry jejich souvislosti, náhodně vybraných ze slov, které se vyskytují dostatečně často v korpusu. Detaily (použitý korpus, způsob anotace atd.) na [Bru12]

Statistiky datasetů, použitých k vyhodnocení úspěšnosti algoritmů uvádí tabulka 5.1

Dataset	**Počet párů slov**	**S/R** [1]	**reference**
Rubenstein Goodenough	65	S	[RG65]
Wordsim	353	R	[Gab02]
MTruk	287	R	[Rad10]
Rare Words	2034	R	[LSM13]
MEN	3000	R	[Bru12]

Tabulka 5.1: Přehled testovacích datových kolekcí

5.2 Metriky úspěšnosti algoritmů

Pro porovnání úspěšnosti algoritmů sémantické podobnosti se používá korelace s lidmi označenými daty. Nejčastěji se používají 2 základní korelace.

5.2.1 Pearsonova korelace

Pearsonova korelace udává míru lineární závislosti dvou veličin.

$$\rho = \frac{cov(X,Y)}{\sigma(X) \times \sigma(Y)} \tag{5.1}$$

$$cov(X,Y) = E(X \times Y) - E(X) \times E(Y) \tag{5.2}$$

kde $\mathsf{E(X)}$ je střední hodnota veličiny X

$$E(X) = \int_{-\infty}^{+\infty} x \times f(x)dx \tag{5.3}$$

[1] udává, jestli dataset obsahuje hodnoty podobnosti(S) nebo souvislosti (R).

$$\sigma(X) = \sqrt{E(X^2) - E^2(X))} \tag{5.4}$$

Pro výpočet korelace mezi vektory lze použít výběrový korelační koeficient.

$$r_{Pearson}(x, y) = \frac{\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=0}^{n} (X_i - \bar{X})^2} \sqrt{\sum_{i=0}^{n} (Y_i - \bar{Y})^2}} \tag{5.5}$$

5.2.2 Spearmanova korelace

Spearmanova (pořadová) korelace udává míru lineární závislosti mezi pořadími jednotlivých položek vektorů.

$$r_{Spearman} = \frac{\sum_{i=1}^{n} (r_{Xi} - \bar{r_X})(r_{Yi} - \bar{r_Y})}{\sqrt{\sum_{i=0}^{n} (r_{Xi} - \bar{r_X})^2} \sqrt{\sum_{i=0}^{n} (r_{Yi} - \bar{r_Y})^2}} \tag{5.6}$$

r_X respektive r_Y je proměnná udávající pořadí jednotlivých hodnot proměnné X respektive Y.

$\bar{r_X}$ respektive $\bar{r_Y}$ je výběrový průměr r_X respektive r_Y.

$$\bar{r_X} = \bar{r_Y} = \frac{n+1}{2}$$

$$\sum_{i=0}^{n} (r_{Xi} - \bar{r_X})^2 = \sum_{i=0}^{n} (r_{Yi} - \bar{r_Y})^2 = \frac{N^3 - N}{12}$$

Po úpravě dostaneme vzorec 5.7. [DWC04]

$$r_{Spearman} = 1 - \frac{6 \sum_{i=0}^{n} (r_{Xi} - r_{Yi})^2}{n \times (n^2 - 1)} \tag{5.7}$$

Korelace je číslo mezi -1 a 1. Korelace v absolutní hodnotě blízká jedné udává velkou lineární závislost, korelace rovna 0 znamená lineární nezávislost. Nutno zdůraznit, že pouze lineární.

6 Implementace a experimenty

6.1 Analýza datových kolekcí

Detailnější informace o jednotlivých datových kolekcích udává tabulka 6.1.

	RG65	**WS353**	**MTurk**	**RW**	**MEN**
Počet párů slov	65	353	287	2034	3000
podobnost(S)/ souvislost(R)	R	S,R	R	R	R
Počet unikátních slov	48	437	498	2949	751
Počet podstatných jmen	48	430	414	1464	648
Počet sloves	24	157	109	758	313
Ostatních slovních druhů	0	0	27	693	31
Pokrytí WordNetu	48	430	443	2501	682
Pokrytí YAGO	48	429	341	1429	643
Pokrytí WN+YAGO	48	433	443	2501	682
YAGO kategorií	209	2843	1921	6904	3956
YAGO instancí	1	895	9	251	19
Pokrytí distrib. metod	48	419	497	2522	751

Tabulka 6.1: Analýza testovacích datových kolekcí

Počet jednotlivých slovních druhů byl zjišťován pomocí WordNetu. V součtu může být počet všech slovních druhů větší než počet slov, protože slova mohou mít více významů napříč slovními druhy.

Ostatních slovních druhů udává, kolik slov, pro které byl nalezen význam ve WordNetu, nemá ve WordNetu význam jako podstatné jméno nebo sloveso. Tento údaj je důležitý proto, že metriky podobnosti, které využívají WordNet, dokáží určit podobnost pouze u podstatných jmen a sloves.

Pokrytí WordNetu udává počet slov datasetu, která byla ve WordNetu nalezena.

YAGO kategorií udává počet slov, respektive významů mapovaných na kategorii YAGO. Protože jedno slovo může mít více významů, může být tento počet větší než celkový počet slov.

YAGO instancí udává počet slov, respektive významů mapovaných na instance YAGO. Protože jedno slovo může mít více významů, může být tento počet větší než celkový počet slov.

Pokrytí distrib. metod udává počet slov nalezených ve slovníku metod distribuční sémantiky, natrénovaných na korpusu získaném z Wikipedie. Detaily tohoto korpusu viz kapitola 6.3.5.

6.2 Podobnost pomocí ontologií

6.2.1 WordNet

Pro manipulaci s Wordnetem existuje jen pro Javu celá řada knihoven.

Java API for WordNet searching (JAWS) Má asi nejjednodušší API, ale pracuje přímo se soubory Wordnetu na disku a je proto velmi pomalý.

Java WordNet interface (JWI) Má také velmi jednoduché API a navíc umožňuje načíst celý Wordnet do paměti, a proto je velmi rychlý. Pro účely implementace jednoduchých metod sémantické podobnosti je naprosto ideální. [Fin14]

Java WordNet library (JWNL a extJWNL) Všude doporučovaná knihovna, která je velmi rozsáhlá a má spoustu sofistikovanějších metod. Její API je ale velice nepřehledné, velmi stručně komentované a není jasně označené, jaké třídy slouží pro běžné používání a jaké jsou pouze pomocné interní. Navíc vyžaduje předzpracování souborů WordNetu, které se na mém počítači vždy zaseklo. Z těchto důvodů jsem od snahy o využití tohoto API upustil.

Existuje spousta dalších knihoven, kterými jsem se ale už nezabýval. Například JawJaw, JavaTools. Srovnání mnoha Java knihoven podle různých kritérií je zpracováno v [Fin14].

6.2.2 Implementace základních metod

Pro přístup k WordNetu jsem implementoval třídu WordNetUtil, která využívá API JWI a poskytuje všechny funkce potřebné pro základní metody podobnosti slov. Například zjištění hloubky uzlu, nalezení nejbližšího společného předka a určení délky cesty mezi dvěma koncepty. Načtení WordNetu do paměti trvá na mém počítači 15 sekund, ale pak už jsou metody podobnosti velmi rychlé.

6.2.3 Metriky WordNetu využívající korpus

Metriky jako Resnik nebo Lin potřebují k výpočtu podobnosti míru informace konceptu, viz 2.3. K výpočtu míry informace jednotlivých konceptů je potřeba dostatečně velký korpus. Použil jsem News Crawl corpus 2011 zveřejněný na [MNL11]. Z korpusu je potřeba spočítat počet výskytů jednotlivých slov. K tomu jsem použil nástroj JFreq, který počítá právě četnosti slov v dokumentech, obsahuje algoritmy pro stemming a umožňuje definovat stopwords. Pro výpočet míry informace potřebujeme nejprve získat odhad pravděpodobnosti, že náhodně vybrané slovo bude patřit do daného konceptu. Tyto pravděpodobnosti spočítáme podle vzorce 2.5. Míru informace vypočítáme podle rovnice 2.4.

Implementoval jsem ještě druhou verzi, kde byla slova nejprve lemmatizována a byl určen jejich slovní druh (POS tagging). Ačkoliv bylo mnohem více slov z korpusu ve WordNetu nalezeno,[1] výsledky metod pro podobnost slov se téměř nelišily. Pro lemmatizaci a POS tagging jsem použil knihovnu coreNLP. [Man+14]

Další dvě podobné verze míry informace, které jsem implementoval, využívají heuristiku k odhadu významu slova. Jako tuto heuristiku jsem použil překrytí slov (word overlap), konkrétně frázové překrytí (viz 2.4.1) mezi kontextem slova v korpusu a glossem nalezeného synsetu.

[1] Jednoduchým přístupem se stemmingem bylo ve WordNetu nalezeno 19 000 ze 124 000 různých slov. S lemmatizací a POS taggingem 52 000 ze 137 000. Větší celkový počet různých slov je způsoben tím, že stejná slova se mohou vyskytovat vícekrát s různým slovním druhem.

bez POS taggingu Tato verze kromě odhadu významu používá pouze stemming na kontext slova a odpovídající glos ve WordNetu. Překrytí slov je počítáno přes všechny slovní druhy.

s POS tagingem Tato verze používá na kontext slova a glos lemmatizaci a k určení slovního druhu využívá POS tagging. Odhad významu je pak použit pouze na slova daného slovního druhu.

Implementoval jsem následující metody podobnosti přes *WordNet*:

- metrika délky cesty
- Wu-Palmerův algoritmus
- Resnikův algoritmus
- Linův algoritmus
- Jing-Conrathův algoritmus

Jejich výsledky jsou uvedeny dále v tabulkách 6.5 a 6.6 společně s metodami využívajícími YAGO.

6.2.4 Srovnání metod výpočtu míry informace

Provedl jsem test úspěšnosti metod WordNetu s různými přístupy ke získání IC představenými v minulé kapitole. Výsledky ukazují tabulky 6.2 a 6.3. Tabulka 6.4 pak ukazuje statistiky jednotlivých metod pro výpočet míry informace.

dataset		**stemming**	**lemma-POS**	**sense-tag**	**sense-POS**
RG65	PC	0,840	0,838	0,850	0,849
	SC	0,774	0,771	0,786	0,794
WS353-sim	PC	0,577	0,571	0,572	0,615
	SC	0,572	0,571	0,554	0,593
WS353	PC	0,308	0,308	0,314	0,340
	SC	0,323	0,329	0,318	0,337

Tabulka 6.2: Porovnání výsledků Linova algoritmu s různými algoritmy výpočtu IC

dataset		**stemming**	**lemma-POS**	**sense-tag**	**sense-POS**
RG65	PC	0,824	0,826	0,789	0,821
	SC	0,756	0,760	0,761	0,757
WS353-sim	PC	0,567	0,579	0,537	0,588
	SC	0,563	0,565	0,535	0,569
WS353	PC	0,306	0,312	0,310	0,335
	SC	0,323	0,319	0,325	0,329

Tabulka 6.3: Porovnání výsledků Resnikova algoritmu s různými algoritmy výpočtu IC

Algoritmus	**unikátních slov**	**významů na slovo**
Stemming	18 743 / 124 115	2,564
Lemma-POS	52 088 / 137 056	2,077
sense-tag	18 877 / 124 115	1,0
sense-tag-lemma-POS	52 057 / 137 056	1.0

Tabulka 6.4: Statistiky jednotlivých algoritmů výpočtu míry informace

Rozdíly ve výsledcích jednotlivých metod nejsou sice nijak zásadní, ale přesto se jako nejlepší jeví metoda, která využívá lemmatizaci a POS tagging a následně odhad významu slova pomocí frázového překrytí slov.

Nejhůře pak dopadla metoda využívající odhad významu pouze se stemmingem. To je celkem logické, protože určování slovních druhů na základě překrytí kontextu s glosem není příliš dobrý způsob.

6.2.5 YAGO

Všechny metody využívající ontologie, které jsem implementoval přes WordNet, jsem implementoval i s využitím ontologie YAGO. Ze zdrojových souborů YAGO je vytvořen strom, který obsahuje následující informace.

1. **Taxonomie** - Systém hierarchicky organizovaných kategorií YAGO extragovaných ze souboru YAGOTaxonomy.
2. **Textová informace** - Obsahuje všechny možné názvy kategorií a instancí. Informace je extrahovaná ze souboru YAGOLabels.
3. **Data instancí** - Zařazení instancí do hierarchického systému kategorií. Vzniká parsováním souboru YAGOTypes.
4. **Mapování na WordNet** - Mapování jednotlivých uzlů na WordNet využívají metody, které kombinují WordNet a YAGO. Informace jsou extrahovány ze souboru YAGOWordNetIDs.

Uzlem tohoto stromu je buď kategorie extragovaná z WordNetu, kategorie extragovaná z Wikipedie nebo instance (entita), která je vždy listem. Uzly jsou spojené relacemi typů hyperonym, hyponym, hyperonym instance a hyponym instance. Takto vznikne strom, svou organizací podobný WordNetu a umožňující velmi podobný způsob vyhledávání. Pro vyhledávání podle slova byla vytvořena mapa pro vyhledávání uzlů podle jejich textové reprezentace. Byla vytvořena také mapa pro vyhledávání podle ID WordNetu. Po implementaci těchto struktur je práce s YAGO pro výpočet sémantické podobnosti slov velice podobná práci s WordNetem.

Protože YAGO obsahuje pouze podstatná jména, implementoval jsem ještě kombinace YAGO a WordNetu. Tyto algoritmy počítají podobnost přes YAGO. Pouze když slovo v ontologii YAGO nenajdou, počítá se podobnost odpovídající metodou WordNetu.

6.2.6 Srovnání metod využívajících ontologie

Následující tabulky zobrazují výsledky metod využívajících ontologie:

Algoritmus	**PC**	**SC**
WN-JNC	0,574	0,766
WN-Lin	**0,840**	0,774
WN-Path	0,783	**0,785**
WN-Res	0,824	0,756
WN-W+P	0,790	0,777
YAGO-Lin	**0,846**	**0,791**
YAGO-Path	0,772	**0,785**
YAGO-Res	0,819	0,759
YAGO-W+P	0,798	0,779
WN-YAGO-Lin	**0,846**	**0,791**
WN-YAGO-Res	0,819	0,759

Tabulka 6.5: Srovnání metod ontologií na Rubenstein Goodenough testu

Algoritmus	**PC**	**SC**
WN-JNC	0,256	0,296
WN-Lin	0,308	0,323
WN-Path	0,371	0,302
WN-Res	0,306	0,323
WN-W+P	0,314	0,357
YAGO-Lin	0,317	0,326
YAGO-Path	**0,388**	0,331
YAGO-Res	0,318	0,340
YAGO-W+P	0,328	**0,369**
WN-YAGO-Lin	0,303	0,316
WN-YAGO-Res	0,306	0,331

Tabulka 6.6: Srovnání metod ontologií na Word Similarity 353 testu

Výsledky ontologie YAGO nejsou tak dobré, jak bych očekával, jsou velmi podobné výsledkům WordNetu. Důvodem bude nejspíše málo konkrétních instancí v testovacích datasetech, viz statistiky v tab. 6.1. Dalším znepokojivým faktem je, že výsledky na datasetu Word Similarity jsou velmi špatné. Výsledky přesto odpovídají i jiným nezávislým testům těchto algoritmů, například na [CL14]. Tyto výsledky jsou mnohem horší než výsledky na datasetu Rubenstein Goodenough hlavně proto, že metody založené na WordNetu, nebo jiné ontologii, vyjadřují pouze podobnost, ne souvislost. Toto tvrzení dokazují výsledky v tabulce 6.7, kde byl Word Similarity test rozdělený na 2 datasety. Jeden vyjadřuje podobnost slov, druhý jejich souvislost.

dataset		**Lin**	**Res**	**Path**	**W+P**
WS353	PC	0,308	0,306	0,371	0,314
	SC	0,323	0,323	0,302	0,357
WS353-sim	PC	0,577	0,567	0,582	0,574
	SC	0,572	0,563	0,576	0,616
WS353-rel	PC	0,031	0,018	0,074	0,048
	SC	0,026	0,034	0,015	0,036

Tabulka 6.7: Porovnání výsledků ontologií na datasetech podobnosti a souvislosti

Jak je vidět z výsledků, algoritmy WordNetu na datasetu vyjadřujícím souvislost absolutně nekorelují, zatímco na datasetu vyjadřujícím podobnost dosahují podstatně lepších výsledků, než na datasetu společném.

6.3 Distribuční sémantika

Pro porovnání statistické sémantiky s metodami využívajícími WordNet jsem použil knihovny GloVe a Word2Vec. Obě knihovny po natrénování vytvoří slovník, který každému slovu přiřazuje vektor. Jejich použití bylo tedy velmi podobné. Abych mohl použít svoji aplikaci pro vyhodnocení úspěšnosti algoritmů, přepsal jsem si jednoduchou metodu pro výpočet podobnosti vektorů do Javy. Jedná se o klasickou kosinovou vzdálenost.

Dále jsem přidal algoritmy LSA a LDA. Použil jsem jejich implementace z knihovny Semilar [RS13].

6.3.1 Word2Vec

Pro natrénování modelu (vytvoření mapy vektorů slov) jsem použil ukázkový korpus dodávaný spolu s programem, konkrétně model GoogleNewsVectors300d dostupný z [Mik+13b].

6.3.2 GloVe

Pro GloVe jsem použil největší model, který byl dostupný na oficiální stránce GloVe [PSM14]. Konkrétně model natrénovaný na Common Crawl Corpus, obsahující 840 miliard slov.

6.3.3 LSA

Pro LSA jsem použil model natrénovaný na článcích Wikipedie, zveřejněný na stránce projektu Semilar [SR14].

6.3.4 LDA

Model dostupný na stránkách projektu Semilar je velmi malý a nedosahuje dobrých výsledků, proto jsem použil pouze model natrénovaný na společném korpusu, viz dále.

6.3.5 Modely natrénované na společném korpusu

Pro objektivnější srovnání metod jsem si opatřil „vlastní“ korpus stažením 472 349 článků z Wikipedie (oficiální obraz wikipedie vytvořený 7. 3. 2015). Na tomto korpusu jsem natrénoval modely pro Word2Vec, GloVe a LDA pomocí trénovacích nástrojů, které jsou součástí příslušných knihoven. K předzpracování Wikipedie (odstranění metadat) jsem použil skript dostupný na [Mah11], pouze s drobnou úpravou, a to rozdělením výsledného textu na řádky podle článků Wikipedie, aby bylo možné natrénovat i LDA, který je založen na dokumentech. Pro GloVe a oba algoritmy Word2Vec byly vytvořeny 300-rozměrné vektory slov.

dataset		**Word2Vec**	**GloVe**	**LSA**	**LDA**
RG65	PC	**0,772**	**0,771**	0,716	0,423
	SC	0,761	**0,770**	0,761	0,534
WS353	PC	0,653	**0,733**	0,586	0,502
	SC	0,700	**0,738**	0,594	0,564
MTurk	PC	0,692	**0,741**	0,516	0,490
	SC	0,633	**0,692**	0,529	0,449
Rare Words	PC	**0,438**	**0,440**	0,240	0,182
	SC	**0,453**	**0,451**	0,302	0,185
MEN	PC	0,736	**0,806**	0,601	0,511
	SC	0,743	**0,805**	0,603	0,587

Tabulka 6.8: Porovnání výsledků statistických metod s různými modely

dataset		**SG**	**CBOW**	**GloVe**	**LDA**
RG65	PC	**0,699**	0,687	0,556	0,423
	SC	**0,717**	0,706	0,575	0,534
WS353	PC	0,556	**0,600**	0,484	0,502
	SC	**0,598**	**0,601**	0,456	0,564
MTurk	PC	0,704	**0,715**	0,664	0,490
	SC	0,648	**0,661**	0,592	0,449
Rare Words	PC	0,189	**0,277**	0,209	0,182
	SC	0,210	**0,260**	0,196	0,185
MEN	PC	**0,704**	**0,706**	0,607	0,511
	SC	**0,721**	0,711	0,617	0,587

Tabulka 6.9: Porovnání výsledků statistických metod natrénovaných na stejném korpusu

Je zajímavé, že výsledky GloVe jsou podstatně horší než výsledky obou metod Word2Vec při natrénování na stejném korpusu, ač podle [PSM14] vychází GloVe o něco lépe. Důvodem mohou být například pro GloVe nepříznivé vlastnosti použitého korpusu (například příliš malý korpus nebo jeho nesprávné předzpracování), ačkoliv jsem neobjevil, jaké by to mohly být. Zkusil jsem použít jiný, větší korpus (Wikipedie 15GB původní velikost, 5,3GB čistý text). Výsledky obou algoritmů byly o něco lepší, ale Word2Vec byl stále výrazně lepší než GloVe.

6.4 Kombinace více metod

Dále jsem implementoval kombinace různých metod formou jednoduché lineární interpolace. Váhy interpolace byly nastaveny vždy optimální pro konkrétní dataset. Účelem experimentu je zjistit, jestli se algoritmy, které využívají pro výpočet podobnosti různé druhy informace, vzájemně doplňují. Výsledky viz v tabulkách 6.10 až 6.12.

6.5 Výsledky

Dosažené výsledky jednotlivých algoritmů ukazují tabulky 6.10 až 6.12.

	RG65			WS353		
	PC	SC	lambda	PC	SC	lambda
GloVe-LDA	0,532	0,609	0,6	0,587	0,628	0,55
GloVe-LSA	0,740	0,760	0,45	0,656	0,664	0,50
CBOW-LDA	0,663	0,729	0,75	0,634	0,680	0,80
CBOW-LSA	0,762	0,780	0,5	**0,670**	**0,700**	0,70
GloVe-Lin	0,861	0,860	0,55	0,545	0,562	0,75
Lin-LDA	0,850	0,846	0,65	0,544	0,594	0,25
LSA-Lin	**0,873**	**0,866**	0,5	0,624	0,645	0,75
CBOW-Lin	**0,872**	0,848	0,55	0,625	0,665	0,85
CBOW-Lin-WN	0,862	0,841	0,65	0,598	0,646	0,85
Lin-CBOW-LDA	0,856	**0,871**	0,75 0,70	**0,661**	**0,705**	0,2 0,75
CBOW	0,687	0,706	-	0,600	0,601	-
CBOW-random	0,687	0,706	1	0,609	0,650	0,95
Lin	0,846	0,791	-	0,317	0,326	-
Lin-random	0,845	0,799	0,85	0,295	0,324	0,55
GloVe	0,556	0,575	-	0,484	0,456	-
GloVe-Random	0,570	0,592	0,95	0,492	0,491	1
LDA	0,423	0,534	-	0,502	0,564	-
LSA	0,716	0,734	-	0,586	0,594	-

Tabulka 6.10: Výsledky I

Lin Linův algoritmus využívající ontologii YAGO

LSA model natrénovaný na celé Wikipedii

CBOW continous bag-of-words model natrénovaný na společném korpusu

GloVe a LDA modely byly natrénované na společném korpusu z Wikipedie

Ostatní metody jsou pouze kombinacemi výše zmíněných s vahami nastavenými tak, aby dosahovaly co nejvyšší spearmanovy korelace.

Hodnoty lambda byly nastaveny tak, aby výsledný algoritmus dosahoval co nejvyšší spearmanovy korelace. Abych ukázal, že přidaná informace nemá zásadní vliv na výsledky, přidal jsem interpolace s náhodnými algoritmy, které korelaci daného algoritmu příliš nezvyšují.

	WS353-sim			MTurk		
	PC	SC	lambda	PC	SC	lambda
GloVe-LDA	0,650	0,673	0,55	0,680	0,605	0,85
GloVe-LSA	0,703	0,706	0,45	0,685	0,621	0,70
CBOW-LDA	0,712	0,717	0,75	0,717	**0,665**	1
CBOW-LSA	0,740	0,740	0,65	0,716	0,662	0,95
GloVe-Lin	0,674	0,698	0,70	0,679	0,610	0,75
Lin-LDA	0,683	0,652	0,50	0,569	0,517	0,60
LSA-Lin	0,734	0,708	0,60	0,579	0,534	0,75
CBOW-Lin	0,739	**0,747**	0,75	**0,726**	**0,674**	0,80
CBOW-Lin-WN	0,726	0,734	0,75	0,727	**0,675**	0,90
Lin-CBOW-LDA	**0,752**	**0,750**	0,25 0,85	**0,729**	**0,674**	0,20 0,95
CBOW	0,712	0,708	-	0,718	**0,666**	-
CBOW-random	0,706	0,712	0,85	0,701	**0,671**	0,80
Lin	0,551	0,541	-	0,440	0,351	-
Lin-random	0,564	0,553	0,90	0,252	0,255	0,90
GloVe	0,574	0,563	-	0,664	0,592	-
GloVe-Random	0,530	0,559	0,95	0,653	0,606	0,95
LDA	0,560	0,573	-	0,490	0,449	-
LSA	0,633	0,622	-	0,568	0,516	-

Tabulka 6.11: Výsledky II

Metody využívající ontologie dopadly velmi špatně na druhé polovině datasetů. Důvod je ten, že ontologie nevyjadřují souvislost ale pouze podobnost, jak ukazují výsledky v tabulce 6.7.

Dalším zajímavým faktem je to, že výsledky kombinací metod ontologie YAGO jsou na všech datasetech lepší, než výsledky WordNetu a i než kombinace YAGO a WordNetu. [2] Z toho plyne, že u slov, která nejsou v YAGO, ale jsou ve WordNetu (například slovesa), neurčuje WordNet podobnost příliš dobře. Lineární kombinace je totiž udělaná tak, aby když jeden algoritmus nemá slovo ve slovníku, vrátila výsledek druhého algoritmu.

Na datasetech Word Similarity a Rubenstein Goodenough dosahují kombinace ontologických a distribučních metod pěkných výsledků.

[2] V tabulce jsem uvedl pouze jeden příklad na kombinaci s CBOW, ale podobné výsledky vykazovaly i ostatní kombinace ontologií, detailnější tabulky na doprovodném CD.

	Rare Words			MEN		
	PC	SC	lambda	PC	SC	lambda
GloVe-LDA	0,228	0,234	0,60	0,643	0,659	0,75
GloVe-LSA	0,270	0,272	0,50	0,680	0,683	0,60
CBOW-LDA	0,290	0,294	0,90	0,708	0,712	0,95
CBOW-LSA	**0,311**	**0,313**	0,70	**0,721**	**0,724**	0,80
GloVe-Lin	0,159	0,193	0,95	0,613	0,623	0,90
Lin-LDA	0,182	0,191	0	0,518	0,594	0,05
LSA-Lin	0,251	0,237	0,90	0,616	0,617	0,75
CBOW-Lin	0,230	0,260	1	0,712	0,718	0,90
CBOW-Lin-WN	0,215	0,241	1	0,711	0,717	0,90
Lin-CBOW-LDA	0,252	0,286	0 0,85	0,714	0,719	0,1 0,95
CBOW	0,277	0,260	-	0,706	0,710	-
CBOW-random	0,211	0,259	0,95	0,706	0,711	1
Lin	0,021	0,023	-	0,264	0,208	-
Lin-random	0,091	0,081	0,90	0,252	0,249	0,90
GloVe	0,209	0,196	-	0,607	0,617	-
GloVe-Random	0,153	0,158	0,80	0,607	0,617	1
LDA	0,182	0,185	-	0,510	0,587	-
LSA	0,240	0,223	-	0,600	0,603	-

Tabulka 6.12: Výsledky III

Zajímavé je, že výsledky interpolací s náhodným algoritmem jsou někdy výrazně horší, než výsledky původního algoritmu. Protože se to stává i při hodnotě lambda rovné jedné, je jedinou možnou příčinou to, že když algoritmus nemá jedno ze slov ve slovníku, vrací nulu a výsledkem interpolace je pak hodnota získaná z druhého algoritmu, tedy místo nul budou náhodné hodnoty.

7 Závěr

Seznámil jsem se s existujícími metodami pro výpočet sémantické podobnosti slov. Některé z nich jsem popsal v teoretické části práce.

Implementoval jsem základní metody sémantické podobnosti slov využívající WordNet, a to jak metody klasické, tak metody využívající míru informace. Implementoval jsem několik různých variant výpočtu míry informace. Také jsem implementoval podobnost přes ontologii YAGO, která obsahuje mnohem více informací než WordNet. To bohužel žádné výrazné zlepšení oproti WordNetu nepřineslo.

Dále jsem vytvořil univerzální aplikaci pro vyhodnocení úspěšnosti algoritmů pro výpočet sémantické podobnosti slov. Kromě metod využívající ontologie jsem přidal několik statistických metod s využitím existujících knihoven. Pro srovnání jsem přidal metody statistické sémantiky s využitím existujících knihoven.

Také jsem vyzkoušel různé kombinace metod statistické sémantiky s metodami využívajícími ontologie. Na datasetech udávajících podobnost slov (nikoliv jejich souvislost) dosahují tyto metody znatelně lepších výsledků než metody samostatné. Také jsem implementoval kombinace různých statistických metod.

Výsledky jsem vyhodnocoval na standardních datových kolekcích, konkrétně Rubenstein Goodenough, Word Similarity, MTurk, Rare Words a MEN.

Seznam zkratek

API Application Programming Interface

IC Information contents (Míra informace kontextu)

idf Inverse Document Frequency (Inferzní četnost slova ve všech dokumentech)

IR Information Retrieval

LDA Latent Dirichled Allocation

LSA Latent Semantic Analysis

NLP Natural Language Processing (Zpracování přirozeného jazyka)

OWL Web Ontology Language

RDF Rosource Description Framework (Jazyk pro popis webových zdrojů)

SVD Singular Value Decomposition (Rozklad na singulární čísla)

tf Term Frequency (Četnost slova v dokumentu)

URI Uniform Resource Identifier

Seznam tabulek

Bibliografie

[Ban02] Pedersen Banerjee. "An adapted Lesk algorithm for word sense disambiguation using WordNet". In: *University of Minnesota, Duluth, USA* (2002).

[Ble+03] D. M. Blei et al. "Latent Dirichlet Allocation". In: *Journal of Machine Learning Research 3* (2003).

[Bru12] Elia Bruni. *The MEN Test Collection*. [citováno 3.4. 2015]. 2012. URL: `http://clic.cimec.unitn.it/~elia.bruni/MEN.html`.

[CL14] Association for Computational Linguistics. *Wiki of the Association for Computational Linguistics*. [citováno 5.4. 2015]. 2014. URL: `http://www.aclweb.org/aclwiki/index.php?title=Similarity_%28State_of_the_art%29`.

[DWC04] Shirley Dowdy, Stanley Wearden a Daniel Chilko. *Statistics for research*. third. John Wiley & Sons, 2004.

[Fel05] Christiane Fellbaum. "WordNet and wordnets". In: *Encyclopedia of Language and Linguistics*. 2005.

[Fin14] Mark Alan Finlayson. "Java Libraries for Accessing the Princeton Wordnet: Comparison and Evaluation". In: *Proceedings of the 7th Global Wordnet Conference. Tartu, Estonia* (2014).

[Gab02] Evgeniy Gabrilovich. *Learning to Predict the Future using Web Knowledge and Dynamics*. [citováno 1. 3. 2015]. 2002. URL: `http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/`.

[JL15] Max Jakob Anja Jentzsch Dimitris Kontokostas Pablo N. Mendes Sebastian Hellmann Mohamed Morsey Patrick van Kleef Sören Auer Christian Bizer Jens Lehmann Robert Isele. *DBpedia*. [citováno 6.4. 2015]. 2015. URL: `http://dbpedia.org/`.

[Kha09] Andy Khan. *Java Excel API*. [citováno 3.4. 2015]. 2009. URL: `http://jexcelapi.sourceforge.net/`.

[Liu07] Bing Liu. "Information Retrieval and Web Search". In: *Web Data Mining*. 2007.

[LSM13] Minh-Thang Luong, R. Socher a Ch. D. Manning. "Better Word Representations with Recursive Neural Networks for Morphology". In: *CoNLL*. Sofia, Bulgaria, 2013.

[Mah11] Matt Mahoney. *About the Test Data*. [citováno 7.4. 2015]. 2011. URL: `http://mattmahoney.net/dc/textdata.html`.

[Man+14] Christopher D. Manning et al. "The Stanford CoreNLP Natural Language Processing Toolkit". In: *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 2014, s. 55–60. URL: `http://www.aclweb.org/anthology/P/P14/P14-5010`.

[Mik+13a] T. Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: ed. ICLR. 2013.

[Mik+13b] T. Mikolov et al. *Word2Vec*. [citováno 2. 3. 2015]. 2013. URL: `https://code.google.com/p/word2vec/`.

[MNL11] MNLP. *EMNLP 2011 SIXTH WORKSHOP ON STATISTICAL MACHINE TRANSLATION*. [citováno 1. 3. 2015]. 2011. URL: `http://www.statmt.org/wmt11/\-translation-task.html\#download`.

[MRS08] Christopher D. Manning, Prabhakar Raghavan a Hinrich Schütze. *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008. ISBN: 0521865719, 9780521865715.

[PA08] S. Xiajiong P. Achananuparp X. Hu. "The Evaluation of Sentence Similarity Measures". In: *College of Information Science and Technology Drexel University, Philadelphia* (2008).

[Ped10] Ted Pedersen. "Information Content Measures of Semantic Similarity Perform BetterWithout Sense-Tagged Text". In: *Department of Computer Science University of Minnesota, Duluth* (2010).

[Pri10] Princeton. *Princeton University "About WordNet."WordNet.* [citováno 28. 2. 2015]. 2010. URL: `http://wordnet.princeton.edu/`.

[PSM14] J. Pennington, R. Socher a C. D. Manning. *GloVe*. [citováno 2. 3. 2015]. 2014. URL: `http://nlp.stanford.edu/projects/glove/`.

[Rad10] Kira Radinsky. *The WordSimilarity-353 Test Collection*. [citováno 1. 4. 2015]. 2010. URL: `http://tx.technion.ac.il/~kirar/Datasets.html`.

[RG65] Rubenstein a Goodenough. "Contextual correlates of synonymy". In: *Communications of the ACM* (1965).

[RS13] Lintean M. Banjade R. Niraula N. Rus V. a D. Stefanescu. "SEMILAR: The Semantic Similarity Toolkit". In: *51st Annual Meeting of the Association for Computational Linguistics*. Sofia, Bulgaria, 2013.

[SKW07] Fabian M. Suchanek, Gjergji Kasneci a Gerhard Weikum. "Yago: A Core of Semantic Knowledge". In: *16th international World Wide Web conference (WWW 2007)*. Banff, Canada: ACM Press, 2007.

[SR14] Banjade R. Stefanescu D. a V. Rus. "Latent Semantic Analysis Models on Wikipedia and TASA". In: *The 9th Language Resources and Evaluation Conference*. 2014.

[W3C14] W3C. *W3C*. [citováno 7.4. 2015]. 2014. URL: `http://www.w3.org/standards/techs/rdf#w3c_all`.

[W3S14] W3Schools. *W3Schools*. [citováno 7.4. 2015]. 2014. URL: `http://www.w3schools.com/webservices/ws_rdf_intro.asp`.

[Wik15] Wikipedia. *Latent Dirichlet allocation*. [citováno 7.4. 2015]. 2015. URL: `http://en.wikipedia.org/wiki/Latent_Dirichlet_allocation`.

A Uživatelská dokumentace

A.1 Aplikace pro vyhodnocení výsledků

Pro porovnání výsledků jednotlivých algoritmů jsem vytvořil Java aplikaci. Aplikace nemá žádné uživatelské rozhraní, je konfigurovatelná přes konfigurační XML soubor. V konfiguračním souboru je možné nastavit následující parametry:

1. **Testované algoritmy** - Přidáváním elementů AlgorithmClass lze přidávat další testované algoritmy. Zadaná třída musí implementovat rozhraní ISimilarityAlgorithm. Aplikace také umožňuje přidat všechny třídy ze zadaného balíčku, které implementují toto rozhraní. K tomu slouží element AlgorithmsPackage.
2. **Datasety** - Element DatasetFile umožňuje přidávat datasety. Obsahem tohoto elementu je cesta k souboru, povinným atributem je word_delimiter, který udává, jakým znakem jsou v datasetu oddělena slova. Testovací páry musí být odděleny koncem řádku.
3. **Balíček s testovanými algoritmy** - Pomocí elementu AlgorithmsPackage je možné zahrnout do testu všechny třídy implementující rozhraní ISimilarityAlgorithm z uvedeného balíčku.
4. **Interpolace** - Pomocí elementu Interpolation je možné přidat novou lineární interpolaci bez zásahu do zdrojového kódu. Tento element má 3 atributy:
   - **algorithm1**
   - **algorithm2**
   - **lambda**

Výsledky jsou uloženy do složky output jako tabulka ve formátu .xls. K zápisu do Excelového souboru byla použita knihovna Java Excel API [Kha09].

Veškerá nastavení jednotlivých algoritmů (umístění modelů pro statistickou sémantiku a ostatních souborů, které jednotlivé algoritmy potřebují pro běh) jsou uložena v souboru resources/configuration.properties.

A.2 Soubory potřebné pro běh aplikace

Všechny soubory, které aplikace při svém běhu může potřebovat, jsou uloženy ve složce resources. Jedná se hlavně o modely statistických algoritmů a data k jejich natrénování. Tyto soubory jsou však velmi velké (dohromady řádově desítky GB), a proto složka resources na přiloženém CD obsahuje v jednotlivých složkách soubor required-resources.txt s odkazy na stažení potřebných souborů a pro Linux bash skript get-resources.sh, který potřebné soubory stáhne automaticky. Když aplikaci bude nějaký soubor chybět, vyzve uživatele ke stažení konkrétní skupiny souborů, kterou potřebuje.

A.3 Přeložení a spuštění

Pro přeložení aplikace je přibalen Ant build, stačí tedy v příkazové řádce napsat ant compile. Aplikaci je možné přeložit a rovnou spustit spuštěním cíle run. Ant script obsahuje cíle pro spuštění různých podaplikací. Jedná se o následující cíle:

- **parseYAGO** - Z datových souborů YAGO vytvoří ontologii a uloží ji do souboru specifikovaného v konfiguračním souboru.
- **trainIC** - Spočítá míry informace z předpočítaných výskytů jednotlivých slov v korpusu.
- **trainICPOSTagged** - Spočítá míry informace z korpusu. Používá lemmatizaci a POS tagging.
- **trainICSense** - Spočítá míry informace z korpusu. Používá odhad významu slova a stemming.
- **trainICSensePOS** - Spočítá míry informace z korpusu. Používá odhad významu slova, lemmatizaci a POS tagging.
- **createWord2VecModel** - Z binárního souboru modelu Word2Vec vytvoří mapu vektorů a tu serializuje. Serializovaná mapa se načítá mnohem rychleji.
- **createGloVeModel** - Ze souboru modelu GloVe vytvoří mapu vektorů a tu serializuje.

- **coverageTest** - Spustí test pokrytí jednotlivých metod.
- **trainLDA** - Z korpusu vytvoří model pro LDA. Parametry tohoto modelu se nastavují v konfiguračním souboru.

Paměťová náročnost aplikace záleží na testovaných algoritmech. Statistické metody většinou používají velké modely, a proto, když člověk testuje více těchto algoritmů, může být paměťová náročnost aplikace obrovská. Když jsem v aplikaci testoval všechny použitě algoritmy, spotřebovávala kolem 25 GB RAM.

B Obsah doprovodného DVD

Doprovodné DVD obsahuje následující složky:

- **app** - Obsahuje celou aplikaci.
- **doc** - Obsahuje zdrojové soubory tohoto dokumentu.
- **test** - Obsahuje kompletní tabulky generované vytvořenou aplikací, jejichž části byly prezentované v této práci.

Složka aplikace obsahuje následující:

- **bin** - Obsahuje aplikaci v binární podobě, přeloženou Javou 1.7.0.71.
- **doc** - Obsahuje Javadoc programátorskou dokumentaci v angličtině.
- **lib** - Obsahuje veškeré knihovny, které aplikace využívá.
- **outputs** - Složka pro soubory vygenerované aplikací.
- **resources** - Obsahuje ostatní soubory, které aplikace potřebuje k běhu.
- **src** - Obsahuje zdrojové soubory.
- **build.xml** - Ant skript pro přiložení a spuštění aplikace i všech podaplikací.