Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

FIIT-5208-8363

Bc. Matúš Pikuliak

Budovanie ontológií z webových zdrojov

Diplomová práca

Študijný program: Informačné systémy
Študijný odbor: 9.2.6 Informačné systémy
Miesto vypracovania: Ústav informatiky, informačných systémov a softvérového inžinierstva, FIIT STU v Bratislave
Vedúci práce: Ing. Marián Šimko, PhD.

Máj 2016

Anotácia

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný program: Informačné systémy
Autor: Bc. Matúš Pikuliak
Diplomová práca: Budovanie ontológií z webových zdrojov
Vedúci práce: Ing. Marián Šimko, PhD.
Máj 2016

Spracovanie veľkých korpusu dát prístupných na webe vo forme textu v prirodzenom jazyku je predmetom intenzívneho výskumu. Problémom je množstvo a charakter dát, ktoré nie sú štruktúrované a sú ťažko spracovateľné strojmi. Jedným z prístupov k tomuto problému je aj budovanie ontológií, akýchsi konceptuálnych máp, nad týmito dátami v snahe o ich explicitné opísanie a strojové pochopenie. Takéto ontológie, budované automaticky, majú veľký potenciál pri riešení viacerých problémov spracovania prirodzeného jazyka.

V tejto práci analyzujeme súčasný stav v oblasti budovania ontológií zo zdrojov v prirodzenom jazyku. Predmetom nášho záujmu je predovšetkým objavovanie sémantických vzťahov medzi lexikálnymi jednotkami. Pre naše potreby sme sa rozhodli používať metódu modelovania prirodzeného jazyka nazývanú vektory latentných čŕt.

Navrhli a overili sme metódu, ktorá v tomto modely automaticky objavuje nové páry lexikálnych jednotiek s požadovaným sémantickým vzťahom. Vzťah pritom definujeme len pomocou malej množiny ukážkových párov. Naša metóda vychádza z predpokladu, že páry s určitým sémantickým vzťahom zanechávajú vo vektorovom priestore určité vzory. Objavovanie nových vzťahov je teda transformované na úlohu objavovania týchto vzorov.

Annotation

Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES
Degree Course: Informatics
Author: Bc. Matúš Pikuliak
Master thesis: Ontology learning from the Web
Supervisor: Ing. Marián Šimko, PhD.
May 2016

Processing of data written in natural language accesible on the Web is the subject of intensive research. The quantity and nature of this unstructured data make it difficult to process for machines. One of the approaches to this problem is building so called ontologies, which are conceptual maps capturing the meaning of text in explicit knowledge structure. This makes the data easier to grasp for machines. Automatically built ontologies have great potential for many of the tasks of the natural language processing.

In this work we are analysing current state of research in the field of ontology learning and population from natural language sources. The main subject of our research is extraction of semantic relations between lexical units. We are using natural language modelling technique called word embeddings to work with text corpora.

We have proposed and evaluated method capable of automatic discovery of new pairs of lexical units with desired semantic relation. Relation in our method is defined by just a handful of exemplary pairs. Our method is based on assumption that certain semantic relations are leaving certain patterns in the vector space we use. Relationship extraction task is therefore transformed to extraction of these patterns.

POĎAKOVANIE

Chcem sa poďakovať vedúcemu práce Mariánovi Šimkovi za odbornú pomoc a konzultácie, ktoré mi poskytol pri vypracovaní diplomovej práce. Takisto sa chcem za podporu poďakovať svojej rodine a priateľom

ČESTNÉ PREHLÁSENIE

Čestne prehlasujem, že záverečnú prácu som vypracoval samostatne s použitím uvedenej literatúry a na základe svojich vedomostí a znalostí.

.....

Bc. Matúš Pikuliak

Obsah

1 Úvod **1**

2 Ontológie **3**
2.1 Ontológie a semantický Web 5
2.2 Budovanie ontológií 6
2.3 Vrstvy budovania ontológie 6
2.3.1 Termy 7
2.3.2 Koncepty 7
2.3.3 Vzťahy 8
2.3.4 Axiómy 8
2.4 Metódy budovania ontológií z textu 9
2.4.1 Štatistické metódy 9
2.4.2 Syntaktické metódy 10
2.4.3 Logické metódy 10
2.5 Súčasné trendy a budúcnosť 10

3 Objavovanie vzťahov z textu **12**
3.1 Rozdelenie vzťahov 12
3.1.1 Taxonomické vzťahy 12
3.1.2 Netaxonomické vzťahy 13
3.2 Prístupy k objavovaniu vzťahov 13
3.2.1 Syntaktické metódy objavovanie vzťahov 14
3.2.2 Štatistické metódy objavovanie vzťahov 15
3.3 Modely vektorových priestorov 15
3.3.1 Matica dokument-term 16
3.3.2 Matica pár-vzor 17
3.3.3 Matica term-kontext 17

4 Vektory latentných čŕt **19**
4.1 Vlastnosti natrénovaných vektorov 22
4.2 Techniky strojového učenia vektorov latentných čŕt 23
4.3 Objavovanie vzťahov pomocou latentných čŕt 24

4.3.1 Reprezentácie vzťahov 27
4.4 Kritika 28

5 Metóda objavovania inštancií vzťahov **30**
5.1 Opis krokov metódy 32
5.1.1 Vytvorenie modelu jazyka 32
5.1.2 Transformácia párov 34
5.1.3 Získanie kandidátov 35
5.1.4 Ohodnotenie kandidátov 36
5.2 Vyhodnocovanie výkonnosti metódy 40
5.3 Aplikácia metódy v kontexte znalostných štrukúr 41
5.4 Implementácia 42

6 Experimenty **44**
6.1 Opis použitých datasetov 44
6.1.1 Vektory latentných čŕt 44
6.1.2 Ukážkové páry 44
6.2 Výskumné otázky 46
6.3 Experiment s generovaním kandidátov 47
6.4 Experiment s hodnotením kandidátov 50
6.5 Meranie presnosti metódy 55
6.5.1 Porovnanie použitých algoritmov hodnotenia 58
6.6 Vplyv veľkosti ukážkovej množiny 59
6.6.1 Pokrytie množiny 60
6.6.2 Medián pozícií 63
6.7 Výsledky metódy pre zvyšné množiny 65
6.8 Vyhodnotenie experimentov 65

7 Záver **67**

Literatúra **69**

A Množiny ukážkových vzťahov **76**

B Technická dokumentácia **82**

C Elektronické médium **97**

Zoznam obrázkov

1 Jednoduchá ontológia 3
2 Príklad hierarchie konceptov. 13
3 Premietnutie inštancií vzťahu do priestoru 25
4 Diagram aktivít opisujúci našu metódu 33
5 Vzťah veľkosti okolia a pokrytia množiny 48
6 Vzťah veľkosti okolia a množstva vygenerovaných kandidátov 49
7 Vizualizácia hodnotení kandidátov 54
8 Vzťah medzi počtom výsledkov a počtom relevantných kandidátov. . . 59
9 Vplyv veľkosti ukážkovej množiny na jej pokrytie 62
10 Vplyv veľkosti množiny na presnosť hodnotenia 64
11 Diagram tried 83

Zoznam tabuliek

1 Podobné slová vo vektorom priestore 22
2 Rozličné spôsoby reprezentácie vzťahov 28
3 Výsledky z experimentu s generovaním jedincov. 49
4 Výsledky z experimentu s hodnotením jedincov. 52
5 Vyhodnotenie výsledkov metódy pre naše množiny 56
6 Výsledky metódy vyhodnotené metrikou NDCG 56
7 Miera podobnosti medzi výsledkami algoritmov hodnotenia. 58
8 Priemerný počet opakovaní slova vo výsledkoch 60
9 Výsledky metódy vyhodnotené metrikou NDCG pre zvyšné množiny. . 65

1 Úvod

Najbežnejší spôsob, akým si ľudia vymieňajú svoje myšlienky či názory, je prirodzený jazyk v ústnej alebo textovej podobe. Pre ľudí prirodzená a triviálna úloha porozumieť textu v známom jazyku je pre stroje veľkou výzvou a uvažuje sa, či ťažkosti spojené s takýmto spracovaním budeme vôbec schopní prekonať. Plné porozumenie jazyka zrejme vyžaduje okrem rozsiahlych syntaktických či sémantických znalostí aj schopnosť abstraktného uvažovania a poznávania sveta. Umelá inteligencia schopná chápať prirodzený jazyk ľudí by sa radila medzi tzv. silné umelé inteligencie, ktorých zostrojenie je zatiaľ v nedohľadne.

Aj napriek tomu však v oblasti spracovania prirodzeného jazyka prebieha intenzívny výskum v rozličných oblastiach súvisiacich so snahou o porozumenie významu hovoreného či písaného slova. Medzi takto skúmané problémy patrí napríklad strojový preklad, sumarizácia textu, odpovedanie na dopyty, rozpoznávanie témy a mnohé iné, pri riešení ktorých sa používa spektrum štatistických, sémantických i logických metód. Medzi známe výzvy patrí aj tzv. Turingov test, ktorý vyžaduje od počítača, aby dokázal viesť konverzáciu s ľudským používateľom bez toho, aby ten zistil že komunikuje so strojom.

Jednou z oblastí takéhoto spracovania textu je objavovanie sémantických vzťahov medzi entitami jazyka. Na úrovni slov sem môžu patriť vzťahy ako synonymita, antonymita a podobne. Taktiež však môžeme hľadať faktografické vzťahy, ako napríklad vzťah medzi krajinou a jej hlavným mestom. Takéto vzťahy potom môžeme využiť pri ďalších úlohách spracovania prirodzeného jazyka. Zautomatizovanie, alebo aspoň urýchlenie objavovania nových inštancií faktografických vzťahov môže využiť napríklad znalostné inžinierstvo.

Na objavovanie vzťahov sa používa viacero metód, ktoré používajú poznatky zo štatistiky, lingvistiky, umelej inteligencie, distribučnej sémantiky a mnohých ďalších vedeckých oblastí. Moderným a perspektívnym trendom pri úlohách spracovania prirodzeného jazyka je využitie tzv. vektorov latentných čŕt. Vektory latentných čŕt sú v podstate výsledkom techniky modelovania prirodzeného jazyka. Slová z textového korpusu sú premietnuté do latentného mnohorozmerného priestoru pomocou neurónových sietí. Rozmery tohto priestoru nemajú samé osebe žiadnu interpretáciu, v priestore

sa však odrážajú sémantické súvislosti medzi slovami. Sémanticky podobné slová sú napríklad premietnuté blízko pri sebe.

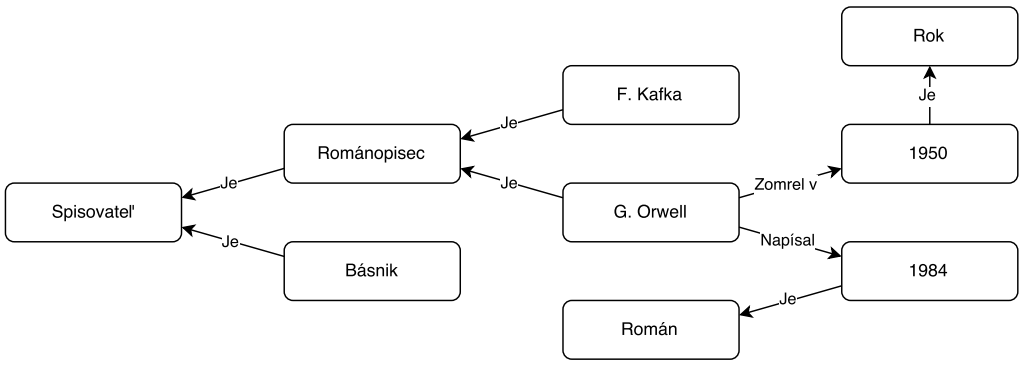
Taktiež bolo dokázané, že sémantické vzťahy medzi lexikálnymi jednotkami zanechávajú v tomto priestoru istú stopu. Páry slov s podobným sémantickým vzťahom bývajú do tohto priestoru premietnuté podobným spôsobom, tj. vzájomné poloha vektorov zložiek takýchto párov býva podobná. V tejto práci navrhujeme metódu, založenú práve na tomto koncepte. Úlohou ktorú sme si vytýčili je zobrať začať s malou množinou ukážkových párov s určitým sémantickým vzťahom, napr. **{Paríž-Francúzsko, Rím-Taliansko, Moskva-Rusko}** a rozšíriť takúto množinu o nové páry s tým istým vzťahom. Naša metóda sa snaží rozpoznať špecifický vzor, ktorý takéto páry zanechávajú v našom vektorovom priestore a skúsiť takýto vzor nájsť aj pri iných párach.

Výhodou našej metódy je jednoduchý spôsob špecifikácie vzťahu. Stačí nam len textový korpus a niekoľko – v experimentoch sme používali 25 – ukážkových párov. Väčšina metód extrakcie vzťahov vyžaduje anotovaný korpus či prepojenie na zložité znalostné štruktúry, ktoré robia prácu pomerne náročnou. Jednoduchosť použitia metódy je podľa nás jednou z najväčších predností našej práce.

Predtým sa ale v Kapitole 2 venujeme ontológiám a ich budovaniu. V Kapitole 3 sa zameriavame na vzťahy a ich objavovanie. V Kapitole 4 opisujeme štatistický model jazyka zložený z tzv. vektorov latentných čŕt a opisujeme metódy, ako pomocou tohto modelu pracovať so sémantikou vzťahou. V Kapitole 5 navrhujeme vlastnú metódu objavovania nových inštancií vzťahu, zadaného malou ukážkovou množinou párov. V Kapitole 6 opisujeme experimenty, ktoré sme vykonali pri overovaní našej metódy. Kapitola 7 je záver, v ktorom zhodnocujeme našu prácu.

2 Ontológie

Slovo ontológia má pôvod v gréckom slove ontologia, ktoré znamená náuka o bytí, jestvovaní. Ontológia je v klasickom poňatí filozofická disciplína zaoberajúca sa existenciou či súcnom a klasifikáciou konceptov. V kontexte počítačovej vedy je to však *štruktúra*, ktorá zachytáva špecifikáciu konceptov a vzťahov medzi nimi [Gruber, 1995]. Inými slovami ontológia je akási sieť konceptov, ktorá sa vytvára zo znalostných dát podľa explicitne určenej formy [Wong et al., 2012]. Príklad jednoduchej ontológie pozostávajúcej z pár konceptov je znázornený na Obrázku 1.



Obr. 1: *Jednoduchá ontológia, zachytávajúca niekoľko konceptov (krabičky) a vzťahy medzi nimi (šípky).*

Koncept je dôležitý pojem v znalostnom inžinierstve a venuje sa mu aj samostatná časť tejto kapitoly. Ide v podstate o pojem, či akúsi platónsku ideu, o ktorej možno uvažovať. Konceptom sú napríklad konkrétne veci (**Beethoven**), ich generalizácie (**hudobník**), abstraktné pojmy (**platónska idea**), alebo čokoľvek iné, čo dokážeme nejako opísať, či aspoň odlíšiť. Treba si tu uvedomiť aj vzťah medzi konceptom a jeho názvom, kde viacero konceptov môže zdielať rovnaký názov, aj keď ide o úplne iné myšlienkové obrazy, napr. **1984** môže odkazovať na rok v gregoriánskom kalendári, ale aj na knihu od Georga Orwella.

V tejto práci budú slová, ktoré odkazujú na ontologické koncepty zvýraznené tučným písmom (napr. **krieda**). Všeobecný vzťah medzi konceptami budeme značiť pomlčkou

(napr. **krieda-jura**). Konkrétny vzťah medzi konceptami budeme označovať tučným písmom a kurzívou (napr. **jura *nasleduje* kriedu**).

Ontológie však môžu obsahovať aj axiómy či ohraničenia týkajce sa ich častí. Vo svojej podstate sú ontológie znalostné štruktúry, ktoré sa snažia zachytiť svet ľudských ideí a abstrakcií do symbolickej formy prístupnej strojom. Text napísaný v prirodzenom jazyku sa taktiež snaží o zachytenie nejakej idey. Proces vytvárania ontológií z textu sa teda dá chápať ako pokus o preklad myšlienky z jednej formy do druhej.

V závislosti od bázy dát môžeme vytvoriť ontológie, ktoré obsahujú úplne odlišné koncepty a vzťahy medzi nimi. Ako zdroj dát možno použiť rozlične štruktúrované dáta z rozličných zdrojov, napr. aj prirodzeným jazykom písané texty z Webu. Ontológie ako formálne a explicitne špecifikované štruktúry môžu napomáhať pri komunikácii medzi strojmi, či medzi strojmi a človekom, keďže okrem syntaxe zachytávajú aj sémantiku svojich dát [Hazman et al., 2011].

Ľudia prirodzene dokážu premýšlať nad konceptami na rozličnej úrovni abstrakcie. Ontológie sa v určitej miere snažia o formalizovanie ľudských myšlienok do podoby symbolov, ktoré sú potom prístupné aj počítačom. Počítače sú stroje zostrojené na manipuláciu so symbolmi a preto dokážu veľmi efektívne ukladať ontológie, prenášať ich či inak s nimi pracovať. Formálne a explicitne definované ontológie sú navyše ľahko rozšíriteľné a prenositeľné medzi strojmi [Cimiano, 2006]

Proces vytvárania ontológii z bázy dát sa nazýva aj budovanie ontológií. Ide teda v podstate o vyhľadanie konceptov a ich vzťahov z ľubovolných dát. Budovanie ontológií môže prebiehať manuálne, poloautomaticky aj automaticky. Zdroje dát môžu byť neštruktúrované, tj. prirodzený jazyk, semištruktúrované aj štruktúrované. Bližšie si proces budovania ontológií opíšeme v samostatnej podkapitole.

Ontológie, ako aj iné reprezentácie znalostí, sa v praxi používajú pri širokom spektre úloh. Medzi tieto patrí napríklad komunikácie medzi agentami, integrácia informácii, objavovanie webových služieb a ich kompozícia, opis obsahu, vyhľadávanie obsahu, obrázkov a hlavne mnohé úlohy z oblasti spracovania prirodzeného jazyka.

Ontológie sa v súčastnosti používajú ako pomôcka pri úlohach z mnohých doménových oblastí, ktoré požadujú znalostné inžinierstvo. Medzi niektoré z nich patria aj bioinformatika, priemyselná výroba, priemyselná bezpečnosť, právo, ochrana životného prostredia, správa prírodných katstrôf, e-Government, e-Commerce, e-Learning či turizmus.

2.1 Ontológie a semantický Web

Dôležitým zdrojom pre objavovanie znalostí a budovanie ontológií sa stáva čoraz viac aj Web a jeho obrovský korpus dát v prirodzenom jazyku postupne vytvorený ľudmi po celom svete. Obsahuje miliardy stránok venovaných najrozličnejším témam v rôznych jazykoch a formátoch. Ako taký obsahuje veľkú časť ľudského poznania, ale automaticky získať toto poznanie pomocou strojov predstavuje veľkú výzvu pre znalostné inžinierstvo. Pre výskum je v súčastnosti zaujímavá najmä online encyklopédia *Wikipedia*. Ako najväčšia encyklopédia obsahuje množstvo článkov, ktoré sú napísané objektívným štýlom, ktorých pravdivosť je overovaná používateľmi a ktoré sú dôsledne kategorizované. Preto v posledných rokoch stúpa jej použitie pre výskumné účely [Wong et al., 2012].

S budovaním ontológií súvisí aj semantický Web, s ktorého konceptom prišli Tim Berners-Lee et al. Berners-Lee et al. [2001]. Vo svojom článku ho navrhol ako rozšírenie súčasného Webu o štruktúrované prvky, ktoré pomôžu strojom porozumieť významu obsahu stránok. Od vydania tohto populárneho článku prebieha výskum, ktorý sa snaží naplniť túto víziu. Na porozumenie významu týchto štruktúrovaných prvkov slúžia najčastejšie práve ontológie. Ich schopnosť spracuvávať dáta a narábať s nimi je pre ozajstné fungovanie semantického Webu veľmi dôležitá [Maedche, 2002].

Rozvoj semantického Webu je však brzdený malou ochotou používateľov internetu a tvorcou obsahu pracne ručne štruktúrovať informácie, ktoré umiestňujú na Web. V súčasnom výskume je preto badateľná snaha o budovanie ontológií a teda aj semantického Webu spracovaním textu v prirodzenom jazyku [Gómez-Pérez et al., 2003, Cimiano et al., 2005, Wu and Weld, 2007]

Pri výskume ontológií a semantického Webu bolo vytvorených a štandardizovaných viacero ontologických jazykov určených pre Web. Medzi najvýznamnejšie patrí jazyk OWL (Web Ontology Language) vytvorený konzorciom *W3C*. Tento jazyk poskytuje formalizovaný semantický opis dát, založený na objektovom rozširení jazyk XML, ktoré sa nazýva RDF (Resource Description Framework). Toto rozšírenie je postavené ako špecifikácia metadátového modelu [Ding et al., 2007]. Tieto štandardy sa uplatnili pri viacerých vedeckých aj komerčných projektoch.

2.2 Budovanie ontológií

Budovanie ontológií je proces, pri ktorom sa vytvárajú ontológie zo znalostí, ktoré sú objavované v dátach. Ručné budovanie ontológií je pri veľkom korpuse dát prirodzene namáhavá a zdĺhavá práca a môžu tak vzniknút len malé ontológie. Toto platí aj napriek tomu, že v posledných desaťročiach bolo vytvorených viacero nástrojov a metód ako rýchlo budovať ontológie manuálne, či dokonca poloautomaticky. Tento jav sa nazýva aj *knowledge acquisition bottleneck*, teda akési úzke hrdlo pri budovaní ontológií, ktoré fatálne spomaľuje ich vytváranie [Wong et al., 2012].

Odpoveďou na tento problém sa stala snaha o zautomatizovanie procesu budovania ontológie, či aspoň obmedzenie ľudského vstupu na prijateľné minimum. S týmto súvisí aj odklon výskumníkov od štruktúrovaných dát a štandardov, akými sú napríklad jazyky XML či vyššie spomínaný OWL. Do popredia naopak vystupujú semištruktúrované, či úplne neštruktúrované dáta, ktorých najväčším zdrojom je samozrejme Web. Ten poskytuje istú formu štruktúry, keďže dokumenty v ňom sú pospájané odkazmi. Zároveň mnohé stránky, ako napríklad *Wikipedia*, ponúkajú na spracovanie aspoň nejaké štruktúrované údaje ako doplnok ku textu v prirodzenom jazyku.

Úplná automatizácia je však veľmi náročná a komplexná úloha, pri ktorej sa využívajú znalosti z oblastí ako počítačové učenie, objavovanie znalostí, spracovanie prirodzeného jazyka, získaveni informácií, umelá inteligencia, správa databáz a iné. Aj napriek intenzívnemu výskumu je ľudský vstup stále potrebnou a žiadanou súčasťou väčšiny algoritmov budujúcich ontológie [Zhou, 2007]. S neustálym napredovaním výpočtových prostriedkov však automatické získavanie jednotlivých častí ontológií stále napreduje.

2.3 Vrstvy budovania ontológie

Metódy používané pri budovaní ontológií sa dajú rozdeliť do niekoľkých vrstiev, ktoré na seba nadväzujú. Tieto vrstvy tvoria štruktúru, ktorá sa v literatúre opisuje ako *layer cake* [Buitelaar et al., 2005]. V tomto pomyselnom vrstvenom koláči vrchné vrstvy môžu dobre fungovať, len ak sú kvalitne preskúmané a vybudované aj spodné vrstvy ontológie. Zároveň platí, že čím vyššie sa vrstva nachádza, tým komplexnejší a abstraktnejší je jej účel.

Každá vrstva má svoj vstup a svoj výstup, pričom horné vrstvy majú za vstup okrem iného spravidla aj výstup vrstiev pod nimi. Bežne sa proces budovania ontológí delí do

štyroch až ôsmych vrstiev [Buitelaar et al., 2005, Wong, 2009, Cimiano and Völker, 2005], v tejto práci pomenovanej podľa výstupov. V krátkosti si predstavíme štyri vrstvy, ktoré ďalej rozoberieme podrobnejšie. Postupnosť pôjde od najnižších, najkonkrétnejších ku tým vyšším:

1. **Termy** - Úroveň lexikálnych jednotiek. Ide o slová či n-tice slov. Niekedy sa v tomto kroku spájajú synonymá.
2. **Koncepty** - Z úrovne lexikálnej sa dostávame do sémantickej. Koncepty sú tvorené z významu termov.
3. **Vzťahy** - Jedná sa o vzťahy, ktoré sa vyskytujú medzi konceptami.
4. **Axiómy** - Axiómy su akési pevné pravidlá či poznatky, ktoré sú pravdivé vo vytváranej ontológii.

2.3.1 Termy

Termy sú z teórie dolovania znalostí akési kľučové slová získané z korpusu dát. Tieto reprezentatívne vyjadrujú obsah dokumentu. S *extrakciou termov* v súvislosti s prirodzeným jazykom súvisí aj *predspracovanie textov* do podoby, z ktorej sa dajú termy ľahko získať. Takéto predspracovanie zahŕňa vymazanie často používaných či neplnovýznamových slov, ale aj štatistické overovanie výskytu termov [Paralič et al., 2010]. Extrakcia termov, ktorá sa však používa aj v iných oblastiach výskumu, je tak prvou z vrstiev budovania ontológií.

2.3.2 Koncepty

Ako bolo spomenuté v úvode tejto kapitoly, koncepty sú akési pojmy, či idey, o ktorých ľudia premýšľajú. Zjednodušene je to čokoľvek, fiktívne či reálne, abstraktné alebo konkrétne, o čom vieme niečo povedať. Pri vytváraní konceptov sa dajú identifikovať dva hlavné problémy: *hľadanie konceptov* a *pomenovanie konceptov*. Pri spracovaní prirodzeného jazyka sa spravidla koncepty hľadajú pomocou množiny termov. Termy sú slová, či korene slov, o ktorých sa domnievame že nesú význam dôležitý pre obsah dokumentu. Pri vytváraní konceptov je však úlohou odhaliť ktoré termy naozaj odkazujú na koncept a keď, tak na ktorý.

Náročnou úlohou je aj problém vysporiadania sa so synonymami, keď viacero rozličných termov odkazuje na jeden koncept, či homonymami, keď jeden term môže podľa kontextu odkazovať na viacero konceptov. Problém synoným sa niekedy uvádza ako samostatna vrstva [Cimiano, 2006]. Tieto problémy odpadajú, pokiaľ ako zdroj dát nepoužívame prirodzený jazyk, ale štruktúrovaná databázu znalostí, kde vieme povedať, ktoré dáta tvoria koncepty, napr. databáza miest a podobne. Tu spadá napríklad úloha rozlíšiť medzi konceptami s rovnakým názvom, ale iným významom (už spomenutá kniha/rok **1984**), ale aj úloha zistiť, že rovnaký termín sa týka jedného konceptu (**Kráľ Slnko/Ľudovít XIV**).

2.3.3 Vzťahy

Medzi konceptami sa dajú hľadať isté vzťahy, alebo relácie. V súčasných ontológiách sa spravidla používajú len vzťahy medzi dvoma konceptami. Problematike objavovanie vzťahov sa venuje v našej práci Kapitola 3.

Objavovanie vzťahov však nie je jediný problém, ktorý sa v tejto vrstve rieši. Ďalšími úlohami sú klasifikácia a pomenovanie vzťahov. Pri klasifikovaní vzťahov sa snažíme zistiť, ktoré vzťahy majú rovnaký sématický zmysel.Inými slovami, ktoré dva vzťahy majú z hľadiska konceptov ktoré prepájajú rovnaký význam [Sánchez and Moreno, 2008]. Chceme teda napríklad určiť, že vzťahy prepájajúce krajiny s ich hlavným mestom patria do spoločnej triedy.

V tomto poňatí sa vzťahy približujú konceptom, keďže získavajú okrem štrukturálnej funkcie v ontológii aj semantický zmysel. Nad týmto zmyslom pracuje potom aj ďalší problém tejto vrstvy, *pomenovanie vzťahov*. Ide tu o hľadanie priliehavého názvu určitej triede vzťahov.

2.3.4 Axiómy

Poslednou,najvyššie položenou vrstvou, je vrstva axióm. Táto vrstva tiež patrí medzi najmenej preskúmané. Axiómy sú podobne ako v logike vety alebo výrazy, ktoré sú po interpretovaní zakaždým pravdivé. Rozširujú teda ontológie o logickú vrstvu, ktorá zlepšuje pochopenie dát a vzťahov v nich. Môžu pomôcť pri interpretácií dát, ako aj pri hľadaní nových relácií medzi už existujúcimi konceptami.

Axiómy sa môžu do ontológie vkladať už hotové [Cimiano, 2006]. Nehľadajú sa teda v dátach, ktoré tvoria znalostnú bázu ontológie, ale pochádzajú z iného zdroja. Často bývajú manuálne vytvorené. Takéto axiómy vyjadrujú základné logické úsudky, ktoré možno nad axiómov urobiť. Medzi takéto patrí napríklad tranzitivita taxonomických relácií a iné jednoduché a všeobecné pravidlá.

Okrem toho poznáme aj automatické hľadanie nových axióm v zostrojenej ontológii. Tieto ontológie sú menej všeobecné ako tie ručne vkladané a spravidla aj sa dotýkajú menšej problémovej oblasti v závislosti od toho, nad akými dátami boli vytvorené. Pri hľadaní týchto dát sa väčšinou používajú znalosti z logiky. Pri pridávaní nových dát do ontológie je tu potrebné axiómy preverovať a upravovať tak, aby zodpovedali novým skutočnostiam.

2.4 Metódy budovania ontológií z textu

Pri budovaní jednotlivých vrstiev ontológií a riešení úloh, ktoré sme uviedli v predchádzajúcej podkapitole, sa využívajú rozličné metódy a prístupy, ktoré sa dajú deliť podľa viacero kritérií. Spomínali sme už delenie budovania ontológií podľa miery do akej sú dáta štruktúrované, a podľa toho nakoľko do procesu zasahuje človek. V tejto časti práce sa budeme venovať predovšetkým metódam, ktoré *automaticky* budujú ontológie z neštruktúrovaných dát, teda z textu napísaného v prirodzenom jazyku.

Takéto metódy sa dajú rozdeliť do troch základných kategórií podľa spôsobu prístupu k textu ako zdroju dát. V praxi sa často používajú kombinácie týchto prístupov, ktoré ponúkajú to najlepšie z jednotlivých prístupov

2.4.1 Štatistické metódy

Štatistické metódy sa opierajú hlavne o výskyt slov, resp. termov, v dokumente. Tieto metódy sú spravidla prebrané z oblastí blízkych dolovaniu znalostí z textu [Paralič et al., 2010, Weiss et al., 2010]. Základnou myšlienkou za týmto prístupom je tzv. hypotéza štatistickej sémentiky [Furnas et al., 1983]. Tá vraví, že štatistické vzory výskytu slov v texte predznamenávajú význam tohto textu. Takto sa dá napríklad predpokladať vzťah medzi slovami, ktoré sa v textoch vyskytujú blízko pri sebe, alebo také, ktoré sa objavujú v podobnom kontexte [Ruiz-Casado et al., 2005, Harris, 1954].

Štatistické metódy sa používajú hlavne pri nižších vrstvách budovania ontológií, ako napr. extrakcia termov či budovanie hierarchie konceptov. Toto je spôsobené problematickým hľadaním sémantickej informácie v čisto štatistickom spracovaní textu. Týmto metódam sa v súvislosti s hľadaním vzťahov budeme venovať aj v ďalších kapitolách tejto práce.

2.4.2 Syntaktické metódy

Syntaktické metódy, niekedy nazývané aj štrukturálne alebo lingvistické, pristupujú k vete ako ku lingvistickej štruktúre. Takéto metódy hľadajú znalosti v texte analyzovaním vetnej štruktúry či obsahu [Suchanek et al., 2006]. Chápu teda text ako postupnosť viet, kde každá má svoj význam, ktorý sa dá rospoznať.

Jedným z lingvistických prístupov je napr. rekurízvne rozdelenie vety podľa pravidiel gramatiky daného jazyka na menšie časti. Výsledkom takéhoto rozdelenia je strom, kde listy predstavujú slová vety. Štruktúra stromu zachytáva vzťah medzi týmito listami a tým aj význam vety. Lingvistické metódy sa používajú prakticky vo všetkých vrstvách budovania ontológie.

2.4.3 Logické metódy

Logické metódy su najmenej bežné a využívajú sa hlavne pri budovaní vyšších vrstiev ontológií, ako hľadanie netaxonomických vzťahov alebo axióm. Medzi tieto metódy zaraďujeme tie, ktoré využívajú techniky automatického logického odvodzovania pre hľadanie nových faktov, ktoré sa dajú využiť pre potreby budovania ontológií.

2.5 Súčasné trendy a budúcnosť

Podľa prieskumov [Wong et al., 2012, Zhou, 2007, Buitelaar et al., 2005] v oblasti budovania ontológií sú spodné vrstvy tohto procesu pomerne dobre spracované a vyriešené. Problémom zostávajú horné vrstvy ontológií, kde sa aktuálne práce zaoberajú najmä hľadaním netaxonomických a komplexnejších vzťahov medzi konceptami. Vrstva axióm zostáva stále len veľmi slabo preskúmaná. Toto je spôsobené tým, že podľa modelu vrstiev je potrebné najprv uspokojivo zostrojiť spodné vrstvy, prv než sa pustíme do tých

vyšších. Takýto stav pritom môžeme pozorovať už niekoľko rokov. Autori spomenutých prieskumov však veria, že pokrok v oblasti budovania ontológií neustal, len sa spomalil.

Moderným trendom, ktorý významne ovplyvnil smerovanie oblasti je aj zameranie sa na obrovský korpus dát, ktorým je Web. Ohromný objem dát na Webe v sebe nesie prísľub možnosti lepšej automatizácie procesu budovania ontológií. Projekty, ktoré sa snažia o využitie Webu postupne vytláčajú tie, ktorých úlohou bolo len vybudovať menšie ontológie z omnoho obmedzenejšieho koprusu dát týkajúcich sa len malej domény.

Z používania Webu ako zdroju dát profitujú štatistické aj syntaktické metódy. V prípade štatistických metód je zrejmé, že čím väčší korpus dát majú k dispozícii, tým viac výsledkov s vyššou presnostou dokážu dodať. Syntaktické metódy sa zasa opierajú hlavne o dáta, ktoré na Web umiestňujú početné komunity používateľov jednotlivých stránok, a ktoré obsahujú množstvo štruktúrovaných informácií. Medzi často používané zdroje týchto dát patrí napr. Wikipedia a jej systém kategórií a štruktúrovaných infoboxov.

Použitie Webu však prináša aj problémy. Pri budovaní ontológií sa treba vysporiadať so stále rastúcim objemom dát a tomu aj prispôsobovať techniky ukladania a manipulácie s dátami. Nezanedbateľným problémom je aj zašumenie dát, ktoré prináša kolaboratívna povaha Webu. Treba sa teda vysporiadať s nespisovným textom, nárečiami, slangom a podobne. Začína sa vynárať aj otázka hodnovernosti zdrojov, ktorá sa niekedy zvykne riešiť vyrátavaním pravdepodobnosti pravdivosti danej znalosti z ontológie.

S rastúcim počtom ontológií sa do popredia zrejme dostane aj otázka ich prepájania a zlučovania, pri ktorom treba odhaľovať potencionálne prekryvy viacerých ontológií. S touto otázkou súvisí aj problém s formalizovaným a štandardizovaným spôsobom ukladania dát.

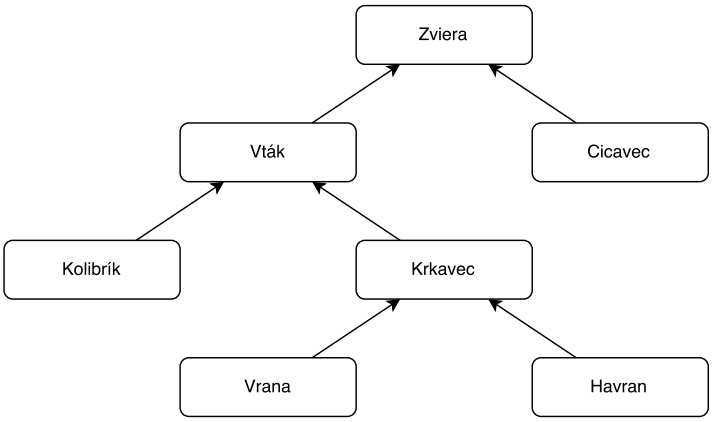
3 Objavovanie vzťahov z textu

Objavovanie vzťahov je jedna z úloh, ktoré sa riešia pri procese populácie ontológií. Jej cieľom je nájsť vzťahy medzi existujúcmi konceptami a tieto vzťahy dostatočne podrobne opísať. V tejto kapitole sa budeme venovať hlavne objavovaniu vzťahov z neštruktúrovaných dát napísaných v prirodzenom jazyku. Najprv opíšeme rozdelenie vzťahov podľa ich typu. Potom si podrobnejšie rozoberieme prístupy k objavovaniu vzťahov a podrobne si opíšeme niektoré metódy.

3.1 Rozdelenie vzťahov

3.1.1 Taxonomické vzťahy

Taxonomické vzťahy sú vzťahy, ktoré rozdeľujú koncepty do tried a podtried v zmysle generalizácie a konkretizácie. Základným vzťahom je teda vzťah **A** ***je*** **B**, kde **A** je konkrétna entita a **B** je jej generalizácia, napr. **kolibrík** ***je*** **vták**, **vták** ***je*** **zviera**. Takéto relácie vytvárajú akýsi pomyselný strom, v ktorom sú entity usporiadané. Tento strom sa bežne nazýva aj hierarchia konceptov [Buitelaar et al., 2005] a jeho jednoduchý príklad je znázornený na Obrázku 2. V lingvistike sa koncepty v tejto relácii nazývajú hyperonymum (**vták**) a hyponymá (**kolibrík**, **sýkorka**, atď.). Okrem toho môžeme v hierarchiách uvažovať aj ďalšie vzťahy, napríklad kohyponýmia, čo je vzťah medzi konceptami na rovnakej úrovni, v našom prípade teda napríklad vzťah **kolibrík-sýkorka**.



Obr. 2: *Príklad hierarchie konceptov.*

Takéto vzťahy reflektujú ľudskú tendenciu klasifikovať koncepty pre potreby logického uvažovania. Známy je sylogizmus: "Všetci ľudia sú smrteľní, Sokrates je človek. A preto je Sokrates smrteľný". Taxonomické vzťahy teda bývajú spravidla tranzitívne. Tj. platí že ak **A** ***je*** **B** a **B** ***je*** **C**, tak **A** ***je*** **C**.

3.1.2 Netaxonomické vzťahy

Netaxonomické vzťahy sú všetky vzťahy medzi konceptami, okrem tých taxonomických. Oproti taxonomickým môžu byť omnoho komplexnejšie, ťažšie uchopiteľné a horšie definovateľné. Medzi dvoma konceptami môže napríklad existovať aj viacero vzťahov. Takisto viacero vzťahov môže mať veľmi podobný význam, aj keď práve tento malý rozdiel môže mať pre interpretáciu dát veľký význam [Kavalec et al., 2004].

3.2 Prístupy k objavovaniu vzťahov

Metódy objavovanie vzťahov sa dajú rozdeliť do kategórií, podobne ako všetky metódy použité pri budovaní ontológií. Základný koncept tohto rozdelenia sme rozoberali v Kapitole 2. V tejto podkapitole sa budeme venovať metódam objavovania všeobecných, hlavne netaxonomických vzťahov v kontexte tohto rozdelenia. Objavovanie taxonomických vzťahov je jednoduchšia a menej všeobecná úloha, ktorá bola preskúmaná o niečo lepšie, ako oblasť netaxonomických vzťahov.

3.2.1 Syntaktické metódy objavovanie vzťahov

Medzi základné metódy syntaktického objavovania vzťahov patria metódy založené na tzv. syntaktických závislostiach. Pod týmito závislosťami si môžeme predstaviť napríklad závislosť medzi predmetmi vety a slovesom, ktoré sa ku nim vzťahuje. Z vety "Vtáky lovia hmyz" môžeme pomocou syntaktických metód usúdiť, že koncepty **vták** a **hmyz** spája vzťah **X** ***loví*** **Y**. V tomto prípade koncepty spája sloveso, možu to však byť aj iné vetné konštrukcie, napr. spojenie "vtáky, ako napríklad kolibrík, sýkorka", naznačuje vzťah medzi jednotlivými konceptami, ktoré obsahuje [Schutz and Buitelaar, 2005, Sánchez and Moreno, 2008].

Iné systémy sa zameriavajú na všeobecnejšie vzťahy ako **X** ***je časťou*** **Y**, **X** ***spôsobuje*** **Y** a podobne [Berland and Charniak, 1999, Yamada and Baldwin, 2004, Poesio and Almuhareb, 2005]. Takéto systémy majú spravidla vynikajúcu presnosť, teda odhalené vzťahy sú skoro vždy správne identifikované. Ich závislosť na presných formách im však prekáža pri odhalovaní viet naformátovaných inak, ako to určuje zadaný vzor. Tieto vzory, podľa ktorých sa vzťahy hľadajú, navyše väčšinou zadáva používateľ systému a má podobu nejakého regulárneho výrazu, ktorý opisuje hľadané časti vety.

Hľadať a zostavovať vzory pre každý možný vzťah je časovo prakticky nesplniteľná úloha. Náročné je aj spísať všetky možné vzory alebo slovesá pre daný vzťah. Existujú však aj prístupy, ktoré dokážu toto hľadanie vzorov automatizovať na základe dostupných informácii [Ruiz-Casado et al., 2005, Suchanek et al., 2006]. Očividnou výhodou takýchto prístupov je zmenšenie podielu práce na vyhľávaní vzťahov, ktorý musí vykonať človek.

Pre syntaktické metódy je typické, že sa v nich dá pomerne ľahko riešiť problém pomenovania a klasifikácie vzťahu, ktorý väčšinou priamo súvisí s použitým slovesom, či inou vetnou konštrukciou, ktorá prepája koncepty. Zaujímavým rozšírením týchto metód je využitie webových vyhľadávačov na vyhodnocovanie správnosti fráz [De Lima and Pedersen, 1999, Etzioni et al., 2005]. Fráza *Kolibrík je vták* má väčšiu šancu objaviť sa vo výsledkoch vyhladávanie ako fráza *Vták je kolibrík*, aj keď obe môžu mať vo svojom kontexte zmysel. Webové prehliadače majú dve črty, ktoré sú pre objavovanie znalostí veľmi dôležité:

1. Pracujú s celým Webom, ktorý obsahuje veľké množstvo textu v prirodzenom jazyku.

2. Tieto dáta priebežne spracúvajú a sprístupňujú komukoľvek. Pri práci s vyhľadávačmi teda odpadá výskumníkom nutnosť pracovať s velkým objemom dát.

3.2.2 Štatistické metódy objavovanie vzťahov

Štatistika sa pri objavovaní znalostí používa aj pri syntaktických metódach opisovaných v predchádzajúcich odstavcoch. Pod pojmom štatistické metódy však rozumieme také metódy, ktoré pri objavovaní vzťahov vychádzajú z tzv. distribučnej hypotézy [Harris, 1954]. Tá hovorí, že slová, ktoré sa vyskytujú v texte pri sebe, majú medzi sebou pravdepodobne nejaký semantický vzťah. Tento prístup má výhodu v tom, že na získavanie znalostí netreba okrem štatistiky ďalšie znalosti o syntaxe a sémantike jazyka.

Očividným problémom štatistického prístupu je, že len na základe informácie o spoločnom výskyte dvoch slov nevieme povedať, akú sémantickú povahu má ich vzťah. Inými slovami, vieme povedať že dva koncepty spolu súvisia, ale nevieme ako. Toto je problémom hlavne z hľadiska vyšších vrstiev ontológií, ktoré očakávajú, že koncepty budú prepájať kvalitne vybudované vzťahy.

Ďalším problémom je podmienka práce s veľkým objemom dát, ktoré treba na získanie štatisticky významných výsledkov. Opäť sa tu ako odpoveď ponúka použitie Webu, ktorý sa stal základom veľkej časti výskumu venovaného štatistickým metódam.

Gentner [1983] preskúmal a definoval dve odlišné miery podobnosti, atribučnú podobnosť a relačnú podobnosť„ o ktorých môžeme uvažovať pri štatistickom spracovaní textu. Atribučná podobnosť je podobnosť medzi dvoma slovami (**a, b**), a hovorí o tom, nakoľko sa tieto slová zhodujú. Relačná podobnosť je podobnosť medzi dvojicou vzťahov (**a-b, c-d**). Vysoká relačná podobnosť znamená, že vzťah a ku b je podobný ako vzťah c ku d. Vysokú relačnú podobnosť môžu mať napríklad páry (**muž-kráľ, žena-kráľovná**).

3.3 Modely vektorových priestorov

Pri štatistických metódach spracovania prirodzeného jazyka sa objavil problém, ako štatisticky modelovať jazyk. Modely, ktoré vychádzajú z distribučnej hypotézy a ktoré modelujú vzťahy medzi entitami v jazyku pomocou vektorov sa nazývaju modely vektorových priestorov (v anglofónnej literatúre Vector Space Model), skrátene z anglického

termínu VSM. Základnou myšlienkou je reprezentovať vybranú entitu (dokument, term, frázu) ako bod vo vektorovom priestore.

Zložky vektorov sa rátajú sledovaním štatistického výskytu sledovanej entity. To, aké konkrétne aspekty jej výskytu sa sledujú závisí od typu zostavovaného modelu. Môžeme sledovať napríklad výskyt termu v dokumente, výskyt termu v jeho okolí (napr. vo vete) či štatisticky významný spoluvýskyt dvoch termov pri sebe. Takéto modely sa dajú použiť pri riešení mnohých úloh spracovania prirodzeného jazyka, napríklad pri sledovaní vzťahu medzi termom a dokumentom [Manning et al., 2008], hľadaní atribučnej [Lin and Pantel, 2001] či relačnej podobnosti termov [Nakov and Hearst, 2008]. Podobnosť medzi entitami v týchto modeloch sa spravidla ráta práve na úrovni vektorov.

Ďalej si v tejto kapitole rozoberieme niektoré typy takýchto modelov:

3.3.1 Matica dokument-term

Azda najrozšírenejším modelom je matica dokument-term, ktorá sa používa pri analyzovaní obsahu dokumentov. Riadky tejto matice predstavujú dokumenty a stĺpce termy, ktoré sa v týchto dokumentoch vyskytujú. Matica obsahuje frekvencie výskytu termu v danom dokumente. Z takejto matice dokážeme vydolovať znalosti týkajúce sa vzťahov medzi termami a dokumentami. V doméne objavovania vzťahov môžeme napríklad konštatovať, že termy, ktoré sa vyskytujú v podobných dokumentoch, budú mať medzi sebou zrejme nejaký sémantický vzťah [Deerwester et al., 1990].

Takéto objavovanie vzťahov sa opiera predovšetkým o atribučnú podobnosť. Jedným z možných využití je pomocou zhlukovania termov objavovať množiny termov, o ktorých predpokladáme že majú podobnú triedu, napr. mená miest [Vyas and Pantel, 2009]. Ďalším možným využitím je klasifikácia slov do rozličných kategórií. Turney et al. [2003] napríklad klasifikovali slová podľa toho, či sú nositeľmi pozitívnej alebo negatívnej konotácie.

Spracuvávané dokumenty však môžu byť pomerne rozsiahle. To oslabuje vzťahy medzi slovami, ktoré sa v nich nachádzajú. Môžeme prirodzene predpokladať, že vzťah medzi slovami, ktoré sa vyskytujú spolu v jednom odseku alebo dokonca v jednej vete, je silnejší ako vzťah medzi slovami, ktoré sa vyskytujú spolu v článkoch či knihách. Preto sa teda popri celých dokumentov používajú aj kratšie úseky textu ako odseky, vety a po-

dobne. Súhrnne sa takéto okolie slova nezávisle na jeho rozsahu nazýva kontext [Turney et al., 2010].

3.3.2 Matica pár-vzor

Iný prístup ku hľadaniu vzťahov, tentokrát založený na relačnej podobnosti, ponúka matica pár-vzor. V riadkoch tejto matice sú dvojice slov a v stĺpcoch sú vzory, ktoré tieto páry spájajú. Pod vzorom si môžeme predstaviť napríklad frázu **X *spôsobuje* Y**, kde **X** a **Y** sú terminály, za ktoré sa dosadzujú slová, ktoré tvoria pár z daného riadku. Hodnoty v takýchto maticiach sa rátajú z frekvencie výskytov daných vzorov v korpuse dát. Hľadanie takýchto vzorov môže prebiehať štatisticky, na základe výskytu slov a ich prepojení vo vete, alebo syntakticky, rozborom vetnej štruktúry. Takáto matica sa využíva napríklad pri pomerne známej metóde Latent Relational Analysis (LRA) [Turney, 2006], kde sa porovnávaním vzťahov medzi dvojicami slov hľadá podobnosť týchto vzťahov.

3.3.3 Matica term-kontext

Posledným prístupom ku modelovaniu priestoru je matica term-kontext. Hlavnou ideou tejto matice je sledovať v akých kontextoch sa vyskytuje určitý term, alebo naopak, aké termy sa vyskytujú v určitých kontextoch. Kontextami v tomto prípade chápeme logické okolia termu v texte [Turney et al., 2010]. Takéto okolia môžu byť napríklad:

- N-gram. Usporiadaná N-tica termov, ktoré sa vyskytujú pred alebo za skúmaným termom.
- Bag-of-words. Podobná myšlienka ako N-gramy, ale skupinu slov z okolia termu neberie ako usporiadanú N-ticu, ale ako neusporiadanú množinu.
- Syntaktické okolie. Oproti predošlým prístupom sa nepozerá na fyzickú susednosť, tj. termy ktoré idú bezprostredne za alebo pred termov o vete, ale na syntaktickú susednosť. Takáto susednosť sa dá odhaliť zostrojením syntaktického stromu vety.

Problémom tohto prístu je však dimenzionala dát, ktoré sa snažíme modelovať. Predstavme si maticu term-kontext, ktorej riadky sú slová z korpusu a stĺpce sú N-gramy (v tomto prípade predstavujúce kontext), v ktorých sa dané slovo vyskytuje. Hodnotou jedného prvku takejto matice je množstvo výskytov slova v danom N-grame vrámci

korpusu, ktorý sa snažíme spracovať, prípadne nejaká odvodená metrika. Často sa používa napr. metrika PMI (Point-wise mutual information) [Church and Hanks, 1990], ktorá sa ráta z podielu spoluvýskytu slov.

Celková veľkosť takejto matice je V^N, kde V je počet termov v slovníku a N je veľkosť N-gramu, ktorý používame. Pri exponencionálnom náraste veľkosti je zrejmé, že takýto prístup môže fungovať len pre menšie slovníky a malé N-gramy. Tento problém, nazývaný aj *kliatba dimenzionality*, sa môže riešiť napríklad zmenšením objemu informácií, ktorý v matici uchovávame. Medzi takéto riešenia patrí napríklad použitie bag-of-words namiesto N-gramov. Pritom však samozrejme strácame údaje o relatívnych pozíciách slov. Ide v takýchto prípadoch teda zrejme o kompromis medzi veľkosťou výslednej matice a množstvom sémantickej informácie, ktoré uchovávame.

Existujú viaceré prístupy k tomu, ako zmenšovať dimenzionalitu takýchto rozsiahlych matíc. Pomerne populárny je algoritmus SVG (Singular Value Decomposition) [Golub and Reinsch, 1970], ktorý sa snaží o vhodne faktorizovať maticu tak, aby vznikla matica s menším počtom rozmerov, ale so zachovanou väčšinou informácii. Podobným spôsobom sa snaží redukovať počet dimenzií v matici aj systém GloVe [Pennington et al., 2014], ktorý získal pozornosť odbornej verejnosti, keďže mal veĺmi dobré výsledky.

Tieto prístupy sú založené na vyrátaní početnosti výskytov a následnej úprave takto vzniknutých vektorov. Naproti tomu v sa v posledných rokoch ako zaujímavý ukázal prístup tzv. distribučných sémantických modelov [Miller and Charles, 1991]. Vektory takýchto modelov sú priamo vytvárané tak, aby predikovali v akých kontextoch sa daný term vyskytuje. Tým, že sémanticky podobné termy sa vyskytujú v podobných kontextoch, táto pravdepodobnosť vedie k tomu, že v tomto modely budú mať podobnú reprezentáciu [Baroni et al., 2014]. My budeme v našej práci používať práve takýto distribučný model. Bližšie si ho ale predstavíme v nasledujúcej kapitole.

4 Vektory latentných čŕt

Problém dimenzionality matice term-kontex, ktorý sme rozoberali v predchádzajúcej kapitole riešili napríklad aj Bengio et al. [2003]. Nosnou myšlienkou ich práce je využiť neurónové siete na vygenerovanie pomerne krátkych číselných vektorov, ktoré budú opisovať jednotlivé lingvistické jednotky – termy. Zložky týchto vektorov majú pritom latentný význam. Vstupnými údajmi, na základe ktorých sa tieto vektory generujú, sú kontexty slov z dátového korpusu, napr. všetky N-gramy v texte a pod. Ďalej uvádzame stručný všeobecný opis algoritmu, ktorý trénuje neurónovú sieť a ktorého výsledkom sú vektory latentných čŕt pre termy z korpusu. Podrobnosti o architektúre použitej neurónovej siete a o procese strojového učenia tu neuvádzame, keďže sa môžu medzi rôznymi riešeniami výrazne líšiť.

1. Je vytvorená neurónová sieť G.
2. Pre každý term zo slovníka je vytvorený náhodný vektor v_{term}. Dĺžka týchto vektorov je voliteľná.
3. Z dátového korpusu sa vytiahnu všetky prítomné dvojice term-kontext. Napríklad to môžu byť všetky N-gramy s piatimi slovami, kde prvé slovo je term a ostatné tvoria kontext. Ak si teda zoberieme N-gram *"Paríž je hlavné mesto Francúzska"*, kde *"Paríž"* je term a *"je hlavné mesto Francúzska"* je kontext.
4. Každej takejto dvojici term-kontext sa vyráta pravdepodobnosť:

   $$P(term|kontext) \tag{1}$$

   , ktorá hovorí, aká je pravdepodobnosť že sa v danom kontexte vyskytuje práve daný term. Napríklad pravedpodobnosť

   $$P(term|"je\ hlavné\ mesto\ Francúzska") \tag{2}$$

   bude zrejme najvyššia pre term *"Paríž"*.
5. Trénuje sa zároveň neurónovú sieť G a vektory v tak, aby sme maximalizovali podobnosť výsledkov výrazov $G(v_{term}, v_{kontext_1}, ..., v_{kontext_N})$ a $P(term|kontext)$.

Inými slovami trénujeme neurónovu sieť a vektory termov tak, aby modelovali ditribučnú pravdepodobnosť jazyka. Táto podobnosť je v tomto prípade zachytená vo vyrátaných pravdepodobnostiach. Pre náš príklad teda maximalizujeme mieru podobnosti medzi výrazmi:

$$G(v_{Paríž}, v_{je}, v_{hlavné}, v_{mesto}, v_{Francúzska}) \tag{3}$$

$$P(\text{"}Paríž\text{"} \mid \text{"}je\ hlavné\ mesto\ Francúzska\text{"}) \tag{4}$$

6. Keď dosiahneme dostatočnú podobnosť pre všetky výrazy generované z korpusu, algoritmus ukončíme.

Výsledná neurónová sieť tak dokáže s určitou presnosťou vyrátať pravdepodobnosť výskytu termu v kontexte. Keďže dvojicu term-kontext vieme zostrojiť z ľubovoľného N-gramu, môžeme pomocou tejto siete vyrátať aj pravdepodobnosť jednotlivých N-gramov. Takáto pravdepodobnosť v tomto prípade vlastne značí akúsi validitu N-gramu, teda to, či takýto N-gram môžeme očakávať v textovom korpuse.

Zaujímavou vlastnosťou je aj to, že takto dokážeme overiť N-gram, ktorý sa v našom textovom korpuse vôbec nevyskytol. Ak by sme používali nejakú formu matice term-kontext, takýto N-gram by mal nulovú pravdepodobnosť výskytu. Je však zrejmé, že aj pri rozsiahlych korpusoch v nich nebudeme mať prítomne všetky možné N-gramy daného jazyka, keďže ich počet je veľmi veľký aj pre malé hodnoty N.

Tento spôsob využitia neurónových sietí však tento nedostatok dokáže odpružiť, keďže na termy sa pozerá len ako na vektory. Stačí, ak máme neurónovú sieť natrénovanú na sémanticky podobných vektoroch. Môžeme pritom prirodzene očakávať, že ak do neurónovej siete na vstup vložíme podobné vektory, dostaneme podobné výsledky. Podobnosť vektorov, ktoré vstupujú do neurónovej siete, je v priamej úmere s podobnosťou výstupu tejto siete.

$$sim(W_1, W_2) \cong sim(G(W_1), G(W_2)), \tag{5}$$

kde sim je miera podobnosti, W je množina vektorov zaznamenávajúca dvojicu term-kontext a G je neurónová sieť. Treba však ešte dokázať, že úplne nový N-gram, ktorý sme v našom korpuse nemali má naozaj podobné vektory, ako jemu sémanticky

blízke N-gramy. Pri N-gramoch *"Paríž je hlavné mesto Francúzska"* a *"Rím je hlavné mesto Talianska"* tak očakávame, že termy na rovnakých pozíciach budú mať podobné vektory. pre termy *"je hlavné mesto"* sú vektory prirodzene rovnaké.

Sú si však podobné dvojice vektorov termov **Paríž-Rím**, resp. **Francúzsko-Taliansko**? Obe dvojice môžeme považovať za sémanticky podobné. Môžeme tiež predpokladať, že termy z dvojích sa objavujú v podobných kontextoch:

$$\forall c \in C : \ P(\text{"}Paríž\text{"} \, |c) \approx P(\text{"}Rím\text{"} \, |c), \tag{6}$$

kde C je množina kontextov z jazyka, ktorý používame a P je pravdepodobnosť toho, že term sa nachádza v danom kontexte. Keďže táto pravdepodobnosť sa používa aj pri trénovaní neurónovej siete, táto sémantická podobnosť sa prenáša aj do výsledných vektorov:

$$\forall c \in C : \ G(a, c) \approx G(b, c) \implies v_a \approx v_b \tag{7}$$

, kde a a b sú termy z jazyka.

Z Harrisovej distribučnej hypotézy predpokladáme, že termy v podobných kontextoch sú sémanticky príbuzné. Tu prepokladáme, že termy v podobných kontextoch majú podobné vektory. Logicky teda môžeme odvodiť, že medzi týmito dvoma dôsledkami platí ekvivalencia a sémanticky príbuzné termy budú mať podobné vektory. Toto je veľmi dôležitý fakt, keďže toto bolo naším cieľom od začiatku. Podarilo sa nám teda zachytiť sémenatickú informáciu medzi termami aj bez vytvárania matice term-kontext. To, že sa táto sémantická informácia zachováva bolo overené viacerými autormi pri rozličných pokusoch a výsledky dokonca predčili predošlé metódy [Bengio et al., 2003, Schwenk, 2007, Mikolov et al., 2013a].

Všeobecne sa pre ako miera podobnosti medzi termami používa bežná kosínusová vzdialenosť. V tabuľke 1 uvádzame ukážku takýchto vzdialeností pre jeden natrénovaný priestor vektorov latentných čŕt. Pomocou nástroja *word2vec* [Mikolov et al., 2013a] sme pre tri slová našli ich päť najbližších susedov. Týchto susedov uvádzame aj z kosínusovou vzdialenosťou. Môžeme si všimnúť, že v tomto zozname sa nachádzajú aj funkčne podobné slová: **france-spain**, ale aj doménovo podobné slová: **crime-prosecution**.

Tabuľka 1: *Podobné slová vo vektorom priestore pre jeden dataset vektorov latentných čŕt.*

france		money		crime	
spain	0.69417	wages	0.65146	murder	0.64402
belgium	0.67582	taxes	0.62951	crimes	0.62500
italy	0.67484	profits	0.61899	criminal	0.60171
netherlands	0.66298	demand	0.61851	homicide	0.55996
germany	0.64375	seignorage	0.60932	prosecution	0.54173

4.1 Vlastnosti natrénovaných vektorov

Priradenie vektora každému termu zo slovníka v algoritme opísanom v predchádzajúcej podkapitole je vlastne len jeho vedlajším efektom. Tento vektor daný term premieta do nového n-rozmerného priestoru. Jednotlivé zložky takéhoto vektora pritom nemajú samé osebe žiadny význam, nedokážeme z nich získať žiadnu informáciu. Ich význam je implicitný, skrytý, a nazývajú sa teda latentné. Tieto vektory potom nazývame **vektormi latentných čŕt** (v anglofónnej literatúre sa najčastejšie používa označenie *word embeddings*).

Ani jeden vektor sám osebe nám žiadnu informáciu sprostredkovať nedokáže. Zmysel začnú mať, až keď ich začneme porovnávať medzi sebou. Toto je dôsledkom toho, ako sa s hodnotami zložiek vektorov pracuje počas procesu strojového učenia. Jediný spôsob, ako z týchto vektorov môžeme získať nejakú informáciu, je ich vzájomným porovnávaním, ktorým odhaľujeme ich sémantickú podobnosť.

Pri rozličných pokusoch nad priestorom vektorov latentných čŕt sa často testuje aj vplyv dĺžky na výsledok [Mikolov et al., 2013a, Levy et al., 2015a]. Dĺžka vektorov je parameter systému, ktorý sa da ľubovolne meniť. Čím sú vektory dlhšie, tým presnejšie sa dajú natrénovať a tým lepšie dokážu zachytiť sémenatickú informáciu. Od istej hranice však väčšie vektory prinášajú len nepatrné zlepšenie. Pri návrhu tu tak treba zvážiť, či drobné zlepšenie stojí za dlhší čas výpočtu takýchto vektorov. Dlhšie samozrejme budú trvať aj operácie, ktoré sa nad takýmito vektormi budú vykonávať.

Schopnosť zachytiť sémantickú podobnosť medzi termami viedlo k ich využitiu pre rozličné úlohy spracovania prirodzeného jazyka. Medzi úlohy, na ktoré boli aplikované, patrí napríklad zhlukovanie slov, rozpoznávanie entít [Turian et al., 2010], rozpoznávanie vetných členov [Luong et al., 2013], preklad slov a fráz [Zou et al., 2013], pomenovanie obrázkov [Frome et al., 2013], zisťovanie sentimentu[Socher et al., 2013] a iné. Pre našu

prácu je dôležité, že sa používajú aj pri objavovaní vzťahov a analógií medzi slovami. Týmto metódam sa budeme podrobnejšie venovať ďalej v tejto kapitole.

4.2 Techniky strojového učenia vektorov latentných čŕt

Architektúra neurónovej siete, ktorá vektory latentných čŕt vytvára, ovplyvňuje kvalitu výsledku, tj. ako dobre vektory zachytávajú sémantickú informáciu. Pôvodná implementácia od Bengio et al. [2003] bola pomerne pomalá a nepresná. Aj keď sa postupom času výsledky v tejto oblasti zlepšovali [Collobert and Weston, 2008, Mnih and Hinton, 2009, Turian et al., 2010], neprekonali konkurečné prístupy.

Výrazný pokrok do tejto oblasti priniesli Mikolov et al. [2013a]. Vo svojej práci navrhli architektúru, ktorá dokáže veľmi rýchlo spracovať aj veľké korpusy dát. Navyše dokázali, že prekonáva predchádzajúce riešenia aj v zachytávaní sémantickej informácie. Prekonávali aj iné prístupy k modelovaniu štatistického priestoru jazyka, ako napríklad state-of-the-art algoritmus Latent Semantic Analysis (LSA) [Zhila et al., 2013] a pod. Vo svojej práci ďalej tvrdia, že najlepšie výsledky prináša tzv. rekurentný neurónový model, ktorí použili aj iní autori [Socher et al., 2013].

Okrem architektúry sú dôležitý aj spôsob výberu n-tice termov, na ktorých sa realizuje algoritmus strojového učenia v neurónovej sieti. Podobne ako pri maticiach term-kontext, ktoré sme analyzovali v podkapitole 3.3.3, aj tu môžeme okrem N-gramov použiť aj bag-of-words prístup. Levy and Goldberg [2014] upravili pôvodnú implementáciu Mikolova a v N-gramoch namiesto fyzickej susednosti slov (ako idú slová za sebou v texte) použili syntaktickú susednosť (ako susedia v syntaktickom strome vety).

Pri tomto prístupe sa najprv z vety automaticky zostrojí jej syntaktický strom. Susedia v takomto strome nemusia za sebou nutne nasledovať aj v texte. Levy et al. tvrdia, že syntaktická blízkosť má väčšiu sémantickú hodnotu ako tá fyzická, a výsledne vektory sú preto presnejšie. Ukázali tiež, že tento prístup oproti predošlým viac zvýrazňuje funkčnú podobnosť na úkor doménovej. Tento prístup síce priniesol výrazne štruktúrovaný priestor vektorov, no autorom sa nepodarilo dokázať prínosnosť tohto riešenia pre niektorú z úloh spracovania prirodzeného jazyka.

Aj ku zostrojeniu funkcie pravdepodobnosti, ktorá sa používa pri trénovaní neurónovej siete, poznáme viac prístupov [Mikolov et al., 2013a]. Prvý, ktorý sme prezentovali v opise algoritmu, sa nazýva *Continuous bag-of-words* model a rátame v ňom pravde-

podobnosť $P(term|kontext)$, teda ako pravdepodobné sú jednotlivé termy pre daný kontext. Druhý prístup sa nazýva *Continuous skip-gram* model a pravdepodobnosť sa v ňom ráta ako $P(kontext|term)$. V tomto prípade teda máme daný term a rátame pravdepodobnosť, s akou sa pri ňom vyskytujú jeho kontexty. Rátame tu pritom každú pozíciu kontextu zvlášť.

4.3 Objavovanie vzťahov pomocou latentných čŕt

Pri práci s vektormi latentných čŕt sa empiricky preukázalo, že takéto vektory veľmi dobre zachovávajú sémantické a štrukturálne vzťahy medzi slovami. Mikolov et al. [2013b] ukázali, že pomocou týchto vektorov dokážu identifikovať aj zložitejšie sémenatické vzťahy medzi termami, nielen jednoduchý vzťah podobnosti. Tvrdia, že vektory latentných čŕt zachytávajú aj relačnú podobnosť medzi dvojicami slov.

Napríklad páry slov **kráľ-kráľovná** a **muž-žena**, majú rovnaký sémantický vzťah založený na opačnosti pohlaví. Mikolov et al. objavili, že takýto vzťah sa premietol aj do priestoru vektorov. Tvrdia, že pokiaľ majú medzi sebou páry slov rovnaký sémantický vzťah, vektory týchto slov budú mať medzi sebou rovnaký geometrický vzťah.

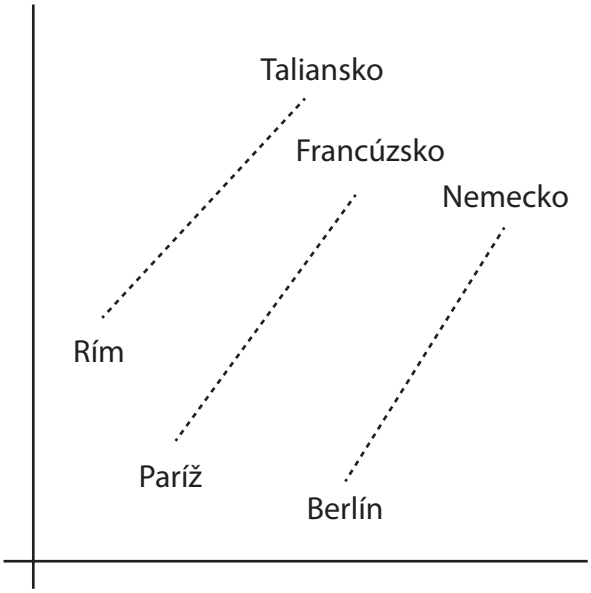
Ak geometrický vzťah vyjadríme pomocou rozdielu dvoch vektorov $v_1 - v_2$, pre náš príklad platí nasledujúci vzorec:

$$v_{kráľ} - v_{kráľovná} \approx v_{muž} - v_{žena}, \tag{8}$$

kde v_X je vektor slova X. Vo všeobecnosti ak máme dva páry termov s podobným sémantickým vzťahom **a-b** a **c-d**, očakávame že platí:

$$v_a - v_b \approx v_c - v_d, \tag{9}$$

Zjednodušený nákres toho, ako môže vizualizácia takejto podobnosti vyzerať v priestore ilustrujeme na obrázku 3. Ak dokážeme takto jednoducho zachytiť vzťahy v priestore, môžeme túto vlastnosť použiť pre objavovanie nových vzťahov, alebo analógií. Prave na analógiách overili Mikolov et al. svoje riešenie.



Obr. 3: *Do zjednodušeného dvojrozmerného priestoru sme premietli termy jazyka. Páry s rovnakým sémantickým vzťahom vytvárajú v priestore podobné vzory.*

Úlohou, ktorú si stanovil bolo nájsť analógie na základe ukážkového vzťahu. V jeho úlohe mal daný vzťah **Paríž-Francúzsko**, hladal analogický vzťah pre ďalšie dané slovo, napr. **Rím**. V tomto prípade očakával, že ako výsledok nájde slovo **Taliansko**. Pre tieto potreby upravil pôvodný vzorec tak, aby neznámu premennú dal na jednu stranu: [Mikolov et al., 2013b]

$$v_d + v_a - v_b \approx v_c \tag{10}$$

Ak poznáme vektory všetkých slov na ľavej strane (v našom prípade vektory slov Paríž, Francúzsko a Rím), dokážeme ľavú stranu vyčísliť do nového vektora. Tento výsledok potom porovnáme so všetkými vektormi z nášho slovníka. Ten, ktorý je najbližšie je výsledkom nášho hľadania. Očakávame teda, že v našom prípade bude najbližšie vektor slova Taliansko. Mikolov s touto metódou dosiahol úspešnosť približne 40%. To znamená, že pre 40% trojíc slov dokázal úspešne nájsť správne slovo, ktoré dopĺňalo analógiu. Ako dataset používal dáta z úlohy 2 SemEval 2012 Jurgens et al. [2012], ako aj vlastný vytvorený dataset.

Ďalším poznatkom z tejto práce je, že percento úspešnosti hľadania takýchto analógií

sa výrazne líši medzi rozličnými vzťahmi a pohybuje sa v rozmedzí 20%-60%. Toto je zrejme spôsobené rozličnými povahami vzťahov, nad ktorými sa analógie hľadali. Autori však neskúmajú, aké sú vlasntosti vzťahov, ktoré vedú k takto rozdielnym výsledkom. Dokázali však, že okrem sémantických vzťahov dokážu zachytiť aj tie syntaktické, napr. vzťah medzi prídavným menom v základnom tvare a v superlatíve (**good-best**) a podobne.

Na túto prácu nadviazali Levy et al. [2014], ktorí navrhli novú formulu hľadania analógií, ktorá nebola založená na aditivite vektorov, ale na ich násobení: [Levy et al., 2014]

$$arg\ max_{b' \in V} \frac{\cos(b', b) \cos(b', a')}{\cos(b', a) + \varepsilon} \tag{11}$$

kde *a-a'* je vzorový vzťah a *b-b'* je vzťah, ktorý hľadáme. Poznáme pritom term b, ku ktorému chceme nájsť *b'*. Ak teda máme vzťah **muž-žena**, a chceme takýto vzťah premietnuť aj na term **kráľ**, očákávame že najvyššiu hodnotu bude mať term **kráľovná**. V tomto prípade teda budú platiť rovnosti:

$$a = v_{muž},\ a' = v_{žena},\ b = v_{kráľ}\ a\ b' = v_{kráľovná} \tag{12}$$

Tento vzorec hovorí, že pre tento term *b'* by mal výraz naberať maximálnu hodnotu zo všetkých slov z množiny V. Funkcia cos je kosínusová vzdialenosť dvoch vektorov. Zanedbateľne malá konštanta ε zabraňuje deleniu nulou v prípade, že by vektory *a* a *b'* boli totožné a ich kosínusová vzdialenosť by teda bola 0.

Pre porovnanie pôvodný vzťah, ktorý navrhol Mikolov, sa dá v tejto notácii kosínusových podobností zapísať aj takto: [Levy et al., 2014]

$$arg\ max_{b' \in V}(\cos(b', b) - \cos(b', a') + \cos(b', a')) \tag{13}$$

Ďalej argumentujú, že takéto hľadanie vzťahov v priestore pomocou vektorových operácii nie je výlučne doménou modelov založených na latentných črtách. Tvrdia, že sa rovnako takéto metódy dajú použiť aj na modeloch založených na maticiach *term-kontext*. Tieto modely nazýva explicitné, keďže informácia o vzťahu medzi termom a kontextom je v nich explicitne uložená.

Ich nová metóda hľadania analógií výrazne vylepšila úspešnosť objavovania správ-

nych analógií pre všetky testovacie prípady - v niektorých prípadoch až o 20%. Pri pôvodnom vzorci od Mikolova et al. boli vektory latentných čŕt výrazne lepšie. S týmto novým prístupom sa však explicitné modely ukázali ako porovnateľne úspešné a v niektorých prípadoch dokonca vektory latentných črt predbehli.

Fu et al. [2014] využili vektory latentných čŕt na objavovanie taxonomických vzťahov. Vychádzajú z práce Mikolova et al. v tom, že predpokladajú, že rovnaké sémantické vzťahy, v tomto prípade **hyperonym-hyponym**, tvoria vo vektorových modeloch rovnaké vzory. Správne však predpokladali, že tento taxonomický vzťah môže vytvárať viacero vzorov, v závislosti od jeho bližšieho sémantického významu. Tento predpoklad potvrdili tak, že zhlukovali známe taxonomické vzťahy a odhalili, že vzťahy v rozličných oblastiach tvoria rozličné vzory. Vzťah **vták-kolibrík** zo zvieracej ríše je iný taxonomický vzťah ako vzťah **robotník-stolár** z triedy ľudských zamestnaní, aj napriek tomu že v oboch prípadoch ide o vzťah **hyperonym-hyponym**.

Fu et al. sa pomocou vektorou latentných čŕt pokúšali z vopred vybraných termov zostrojiť hierarchický strom. Používali pritom ten jav, že sémanticky podobné slová sa nachdázajú pri sebe a vytvárajú tak akési zhluky. Každý vzťah porovnal s vzorovými zhlukmi a ak nastal významný prekryv, usudzovali, že tento vzťah má rovnaký sémantický význam ako tie vzorové. Výsledné F-skóre prekonalo predchádzajúce riešenia, niektoré z nich tiež založené na distribučnej hypotéze. Tento výsledok boli schopní ešte zlepšiť využitím manuálne zostrojených ontológií.

4.3.1 Reprezentácie vzťahov

Doteraz sme v práci uvažovali o vzťahu na úrovni vektorov ako o rozdiele dvoch jeho členov. Toto však nie je jediný spôsob, ako môžeme algebraicky reprezentovať vzťah jeho členov. V literatúre môžeme naraziť aspoň na tri rôzne spôsoby spracovania vzťahu dvoch termov. Vymenované spolu so vzorcom na ich získanie a pamäťovou náročnosťou výslednej reprezentácie sú uvedené v tabuľke 2. Metóda rozdielu je zrejme najintuitívnejšia a používa sa aj pri iných problémoch, kde sa objavujú vzťahy pomocou vektorov. Zreťazenie použili vo svojej práci napríklad Baroni et al. [2012]. Výhradou voči tomuto prístupu býva, že nezachytáva explicitne vzťah medzi dvojicou vektorov. Prístup s maticovým súčinom využili vo svojej práci Fu et al. [2014]. Vzorec z tabuľky

Tabuľka 2: *Rozličné spôsoby reprezentácie vzťahov. r vo vzorcoch je výsledná reprezentácia. N je v dĺžka pôvodných vektorov latentných čŕt.*

Názov reprezentácie	**Vzorec**	**Pamäťová náročnosť**
Rozdiel vektorov z páru	$r = v_1 - v_2$	$O(N)$
Zreťazenie vektorov z páru	$r = v_1 \| v_2$	$O(2N)$
Maticový súčin	$v_1 * r = v_2$	$O(N^2)$

však nemá pre neznámu maticu r len jedno riešenie. Preto treba na natrénovanie tejto reprezentácie viacero párov, pomocou ktorých sa dá matica r aproximovať.

4.4 Kritika

Aj keď vo všeobecnosti odborná obec považuje vektory latentných čŕt za perspektívny prístup k hľadaniu vzťahov, objavili sa aj kritické hlasy, ktoré spochybňujú skvelé výsledky, ktoré sa pomocou tohto prístupu dosiahli.

Levy et al. [2015b] vo svojej práci tvrdia, že modely jazyka vychádzajúce z modelu term-kontext, vrátane vektorov latentných čŕt, vskutočnosti nedokážu veľmi dobre zachytiť niektoré sémantické vzťahy medzi slovami. Autori sa konkrétne venujú lexikálnemu záveru (ak **A** je lexikálny záver **B**, môžeme **B** v texte kedykoľvek nahradiť za **A**). Tento poznatok overujú na viacerých metódach reprezentácie slov i objavovania vzťahov. Vo svojej práci však napokon uvádzajú len veľmi málo nameraných výsledkov.

Levy et al. [2015a] sa zamerali na porovnanie rozličných prístupov reprezentácie slov. Pracovali s explicitnými maticami term-kontext, ich SVD faktorizáciou a s vektormi latentných čŕt. Nad každým z týchto prístupov testovali viacero nastavaní a skúmali ich vplyv na výsledky v úlohach hľadania podobnosti medzi slovami a objavovania analógií. Namerali, že drobné zmeny v nastaveniach algoritmov môžu priniesť výrazné zlepšenie. Odporúčajú preto pred porovnávaním rozličných metód riešenia úlohy radšej venovať čas optimalizácii parametrov.

Vo svojej práci [Levy et al., 2015a] ďalej tvrdia, že zle vyladené konkurenčné prístupy v iných článkoch nadhodnotili výkon metód založených na vektoroch latentných čŕt. Pri ich porovnávani napríklad dokázali vyladiť SVD prístup tak, že predčil v úlohe hodnotenia podobností medzi slovami vektory latentných čŕt. Pri úlohe hľadania analógií však konštatovali, že vektory latentných čŕt prinášajú najlepšie výsledky. To zrejme

znamená, že dokážu relačnú podobnosť zachovávať lepšie ako atribučnú. Toto nám dáva nádej, že vektory latentných čŕt sú najlepšia voľba pri objavovaní nových vzťahov čo je zámerom našej práce. Levy et al. ďalej konštatujú, že sú na tom najlepšie aj z hľadiska pamäťovej a výpočtovej zložitosti.

5 Metóda objavovania inštancií vzťahov

V našej práci navrhujeme novú metódu objavovania inštancií sémantických vzťahov medzi termami v texte písanom v prirodzenom jazyku. Pod pojmom inštancie vzťahu rozumieme *pár* termov, ktoré medzi sebou majú daný sémantický vzťah. Ďalej v tejto práci budem slovo *pár* naberať práve tento význam.

V našej metóde využívame štatistický model jazyka vytvorený z vektorov latentných čŕt tak, ako ho navrhli Bengio et al. [2003]. Ako sme spomenuli v predošlej kapitole, takýto model bol použitý na viaceré úlohy týkajúce sa práce so sémantickými vzťahmi medzi termami. Spomenuli sme určovanie podobností medzi termami [Mikolov et al., 2013a], hľadania analógií [Levy et al., 2014], zostrojovanie taxonomického stromu [Fu et al., 2014].

Ako úlohu, ktorú budeme riešiť v našej práci, sme si vytýčili objavovanie inštancií sémantických vzťahov medzi slovami na základe množiny ukážkových párov s takýmto vzťahom. Napríklad pre úkažkovú množinu **Francúzsko-Paríž, Taliansko-Rím, Nemecko-Berlín** chceme nájsť nové páry s rovnakým sémantickým vzťahom, čo môžu byť **Španielsko-Madrid, Rusko-Moskva** a iné. V tomto prípade ide samozrejme o vzťah medzi krajinami a ich hlavnými mestami. Akúkoľvek špecifikáciu sémantického vzťahu však v našej metóde nebudeme musieť uvádzať, vystačíme si iba s ukážkovými pármi. Veríme, že takýto intuitívny jednoduchý prístup k definícii vzťahov je jednou z dôležitých predností našej metódy. Na základe spomenutých úloh a ich výsledkov usudzujeme, že sme z modelu vektorov latentných čŕt naozaj schopní zachytiť dostatočne veľa informácie o sémantických vzťahoch.

Naša metóda má dva vstupy:

- *Textový korpus dát.* Z tohto korpusu, napísanom v prirodzenom jazyku, modelujeme daný jazyk pomocou neurónových sietí tak, ako sme opisovali v predošlej kapitole. Keďže využitie vektorov latentných čŕt radíme medzi štatistické prístupy spracovania prirodzeného jazyka, platí, že čím väčší je tento korpus, tým presnejší bude aj model. V našom prípade teda väčší korpus značí presnejšie zachytenie sémantických vzťahov medzi termami. Ako termy môžeme použiť čokoľvek, v našej práci budeme vo všeobecnosti za termy považovať všetky slová z jazyka.

• *Ukážkové páry termov s hľadaným vzťahom.* Druhým vstupom je množina párov termov, ktoré reprezentujú hľadaný vzťah. Termy, ktoré tvoria tento pár, musia byť zastúpené aj v textovom korpuse. Môžeme prirodzene predpokladať, že čím viac sa dané termy vyskytujú v textovom korpuse, tým lepšie bude v modeli zachytená sématická informácia o vzťahoch. Ak hľadáme napríklad vzťahy medzi geografickými entitami, nemôžeme očakávať, že tieto budú štatisticky významne zachytené v textovom korpuse, ktorý sa aspoň čiastočne nezaoberá geografiou. Na vstupe pritom očakávame množinu obsahujúcu rádovo desiatky párov. Teda takú veľkosť, akú dokáže pohodlne pripraviť aj človek.

Problémom takto zadefinovaného vstupu je, že jeden term môže mať viacero sémantických významov; inými slovami, že v prirodzenom jazyku existujú homonymá. To, že nejaký term má v jazyku viacero významov, sa zrejme odrazí aj na tom, ako je v ňom štatisticky zastúpený. Napríklad pokiaľ hľadáme sémanticky podobné slová, tak pre term **koruna** tak môžeme označiť aj slovo založené na funkčnej podobnosti s menou koruna, napríklad **dolár**, ale aj slovo založené na tématickej podobnosti s kráľovským klenotom, napríklad **kráľ**. Pre potreby tejto metódy by teda bolo najlepšie, keby sme namiesto slov používali entity, ktoré zachytávajú sémanticky význam daného slova či skupiny slov. Takúto funkciu majú napríklad ontologické koncepty a ich inštancie. Ich hľadanie v termoch textu však nie je triviálna úloha.

Výstupom našej metódy je usporiadaný zoznam párov, medzi ktorými očakávame rovnaký sémantický vzťah ako v ukážkových párach. Čím vyššie sa pár v tomto zozname nachádza, tým väčšiu by mal mať šancu že je naozaj relevantný pre náš hľadaný vzťah. Úlohou je teda nájsť potencionálne relevantné páry a ohodnotiť ich tak, aby tie viac relevantné mali vyššie skóre. Pre posúdenie výsledkov metódy môžeme použiť metriky známe z oblasti objavovania znalostí, ktoré hodnotia správnosť zoradenia, ako napríklad *NDCG* alebo *Precision@K*.

Ďalšie obmedzenie ktoré si kladieme je počet výsledkov, ktoré z takejto metódy berieme do úvahy. Pokiaľ chceme uvažovať o reálnom využití našej metódy v znalostnom inžinierstvo, musíme rátať s tým, že ľudskí používatelia nebudú ochotní prezerať tisícky výsledkov. Naším cieľom je teda predložiť používateľovi zoznam s dĺžkou približne 100 párov, ktorý bude obsahovať nové páry, ktoré naša metóda objaví.

Okrem metódy sú výstupom tejto práce aj nové poznatky o vektoroch latentných čŕt z hľadiska uchovávania sémantickej informácie medzi termami. Tieto poznatky sú výsledkom našej potreby bližšie preskúmať štruktúru a vlastnosti použitého modela jazyka. Výskum tejto oblasti je zatiaľ pomerne slabý a povrchný. Bližšie sme sa preto museli pozrieť na viacero aspektov zachytávania sémantickej informácie, ktoré doteraz neboli preskúmané. Potreba týchto poznatkov pramení aj z faktu, že tak, ako sme si úlohu postavili my, nebola zatiaľ podľa naších znalostí v kontexte vektorov latentných čŕt riešená.

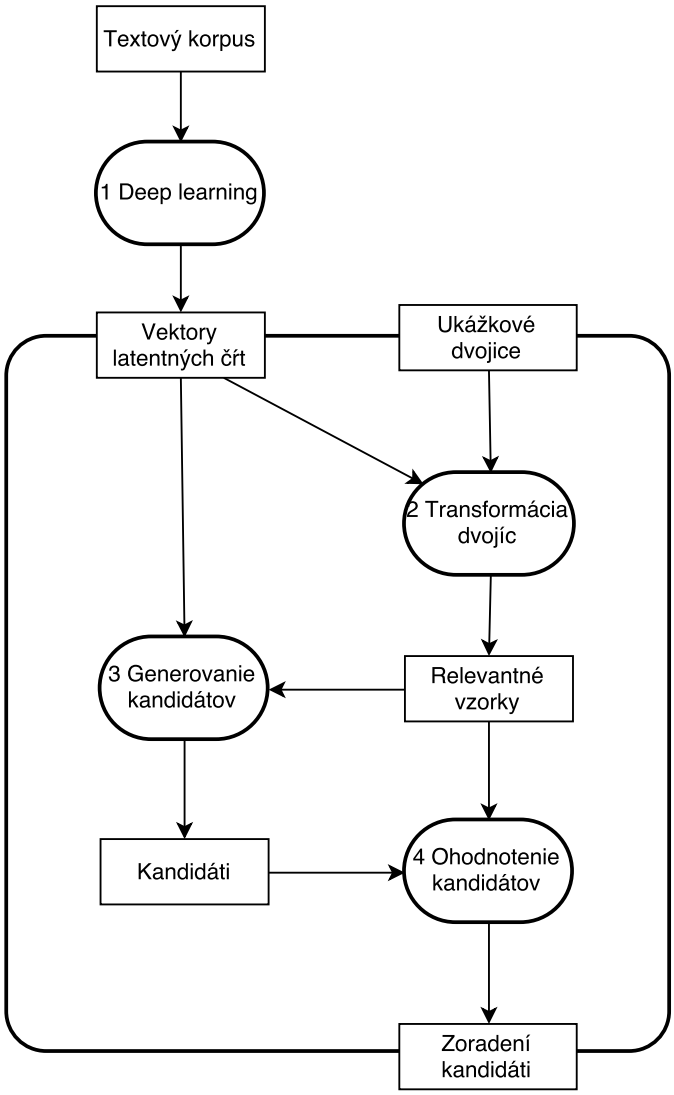
5.1 Opis krokov metódy

V tejto podkapitole opíšeme jednotlivé kroky našej metódy objavovania nových párov. Spôsob výberu a hodnotenia, ktorý sme navrhli, je podľa našich znalostí nový a originálny. Najprv si heslovite našu metódu priblížime vymenovnaním jej krokov a potom všetky tieto kroky podrobne rozoberieme. Náčrt metódy je znázornený aj na diagrame aktivít na Obrázku 4.

1. Vytvoríme model prirodzeného jazyka z textového korpusu pomocou techniky vektorov latentných čŕt.
2. Ukážkové páry transformujeme na vzorky. Znamená to že nájdeme vektory jednotiek tohto páru a vyrátame ich rozdiel.
3. V okolí týchto párov nájdeme množinu kandidátov. Očakávame, že medzi týmito kandidátmi sme schopní nájsť nové relevantné páry.
4. Ohodnotíme kandidáty podľa vybranej metriky podobnosti so vstupnou množinou. Kandidáty podľa tohto hodnotenia potom zoradíme. Výsledkom metódy je práve toto zoradenie.

5.1.1 Vytvorenie modelu jazyka

V našej metóde využívame ako model jazyka priestor vektorov latentných čŕt. Algoritmus vytvárania týchto vektorov sme opísali v Kapitole 4. Neurónové siete v ňom na základe vstupného textového korpusu dát priradia každému termu z korpusu vektor dĺžky n tak,



Obr. 4: *Diagram aktivít popisujúci našu metódu. Proces trénovania vektorov sme vyčlenili mimo samotnú metódu, keďže v skutočnosti ho stačí urobiť len raz prípadne je možné použiť aj existujúci dataset. Diagram znázorňuje proces rátania našej metódy a medzivýsledky, ktoré pri tomto rátaní vznikajú. Jednotlivé aktivity sú očíslované a toto číslovanie zodpovedá aj krokom našej metódy.*

že v priestore zostávajú zachované niektoré sémantické vzťahy. Levy and Goldberg [2014] ukázali, že rozličné nastavenia toho algoritmu strojového učenia môžu výrazne ovplyvniť výsledný priestor. Niektoré nastavenia preferujú doménovú podobnosť medzi termami, iné zasa funkčnú. Experimentovanie s týmito nastaveniami nám môže povedať, ktorý prístup je najlepší pri odhaľovaní sémantickej podobnosti vzťahov.

Z istého pohľadu môžeme tento krok postaviť ešte mimo samotnú našu metódu. Prakticky nepotrebujeme rátať vektory latentných čŕt zakaždým, keď sa naša metóda bude spúšťať. V skutočnosti ich stačí vyrátať len raz alebo dokonca stačí pracovať s hotovými predpripravenými vektormi. Myslíme však že tento aspekt našej metódy je veľmi dôležitý a preto ho tu uvádzame ako samostatný krok.

Pri testovaní takýchto nastavení sa môžeme pozrieť na vplyv viacerých faktorov na výsledky celej metódy. Všetky z nich sme podrobne rozobrali v predošlej kapitole:

- *Dĺžka vektorov.* Dĺžka vektorov je nastaviteľný parameter systémov generujúcich vektory latentných čŕt. Môže nadobúdať ľubovoľnú hodnotu, najčastejšie sa používajú rádovo stovky. Bolo dokázané [Mikolov et al., 2013b], že dlhšie vektory dokážu lepšie zachycovať sémantickú informáciu. S dĺžkou sa však samozrejme zvyšuje výpočtová zložitosť vytvárania i práce s takýmito vektormi.

- *Architektúra neurónovej siete.* Mikolov et al. [2013a] preskúmali aj rozličné architektúry neurónovej siete. Podľa tejto práce má najlepšie výsledky model *continuous skip-gram*.

- *Definícia kontextu termu.* Ďalším aspektom algoritmu, ktorý môže výrazne ovplyvniť výsledné vektory, je spôsob, akým sa určuje kontext termu. Ide teda o postup, akým určujeme n-tice termov, ktoré považujeme za okolie.

5.1.2 Transformácia párov

V tomto novovytvorenom priestore potom vyhľadáme vektory, ktoré prislúchajú termom tvoriacim ukážkové páry. Nutnou podmienkou je, aby sa tieto termy vyskytovali v textovom korpuse, z ktorého boli vytvárané vektory latentných čŕt. Pokiaľ sa niektoré termy v tomto korpuse nenachádzajú, prislúchajúci vektor sa pochopiteľne nevytvorí. V našich experimentoch pracujeme s termami na úrovni slov a ďalej v práci budeme aj používať termín *slovo*. Jednoduchým rozšírením môže byť aj využívanie nielen slov, ale

aj celých fráz. Takto dokážeme napr. pracovať s termami ako *Los Angeles*, s ktorými sa inak pracuje ako s dvoma slovami.

Po vyhľadaní vytvoríme *vektor páru* vyrátaný ako rozdiel vektorov jeho zložiek. Ďalej budeme pod týmto pojmom vždy rozumieť takýto vektorový rozdiel. Črty tohto vektoru budú taktiež latentné. Ide teda o spôsob, ktorý je v literatúre pomerne rozšírený a ktorého výhody boli rozoberané v predchádzajúcej kapitole.

5.1.3 Získanie kandidátov

Okrem známych párov daných hľadaným sémantickým vzťahom nemáme žiadne informácie o tom, aké by mohli byť ďalšie potencionálne páry. Jednoduchým prístupom by mohlo byť skúsiť ohodnotiť všetky možné páry slov z nášho slovníka. Pre binárne vzťahy je však takýchto párov $O(V^2)$, kde V je veľkosť slovníka. Toto je pre bežne používané korpusy s slovníkmi s miliónmi termov zrejme príliš vysoké číslo.

Potrebujeme preto túto množinu všetkých možných párov „orezať“ tak, aby sme vybrali len perspektívne páry. Cenou za akékoľvek orezanie je prirodzene to, že existuje šanca že vylúčime aj páry slov, ktoré majú medzi sebou hľadaný vzťah. Pri orezávaní vychádzame z predpokladu, že slová z jednej triedy vzťahu majú medzi sebou funkčnú podobnosť. Pri vzťahu krajiny a jej hlavnému mesta teda logicky predpokladáme, že na prvej pozícii bude slovo so sémantickým významom krajiny a na druhej pozícii bude slovo so sémantickým významom mesta. V priestore vektorov latentných čŕt sa takáto sémantická podobnosť odráža tak, že sa slová z jednej takejto triedy zhlukujú a sú v priestore blízko pri sebe [Mikolov et al., 2013b].

Znamená to, že pri sémanticky rovnakých párach neberieme do úvahy len vzťah medzi jeho dvoma zložkami, teda napríklad ich rozdiel $v_1 - v_2$, ale aj umiestnenie týchto zložiek v priestore. Podobnosť dvoch vzťahov $sim(r(v_1, v_2), r(v_3, v_4))$ je podmienená podobnosťou oboch ich zložiek $sim(v_1, v_3)$ a $sim(v_2, v_4)$. V tomto prípade funkcia r predstavuje vzťah medzi dvoma vektormi a funkcia sim ráta podobnosť dvoch entít. Podobný predpoklad využívajú vo svojej práci aj Fu et al. [2014]. Ten na základe umiestnenia zložiek párov zhlukoval páry do samostatných skupín, kde každá skupina mala jeden sémantický vzťah.

Pri našom orezávaní pre každý pár z ukážkovej množiny nájdeme jeho okolie. Zjednotenie týchto okolí pre všetky ukážkové páry tvorí výsledný zoznam kandidátov. Okolie

páru pritom definujeme ako karteziánsky súčin slov, ktoré sa v priestore nachádzajú v blízkosti zložiek páru. Pri páre *Paríž-Francúzsko* sa teda jedná o súčin okolia slova *Paríž* a okolia slova *Francúzsko*. Ak slovník všetkých slov označíme písmenom V, vstupnú množinu párov označíme písmenom S a našu novú množinu kandidátov označíme písmenom C, tak C zostrojíme podľa vzorca:

$$C = \bigcup_{\langle a,b\rangle \in S} \langle x, y\rangle : \forall x \in V - a, \forall y \in V - b, |x - a| \leq \alpha, |y - b| \leq \alpha, \tag{14}$$

kde α je nastaviteľná hranica, pomocou ktorej môžeme upravovať požadovanú veľkosť okolia slov. Pri implementácii sme používali ako ohraničenie prvých n najpodobnejších slov. Výpočtová náročnosť tohto prístupu je $O(|V| * |S|)$. Pokiaľ je náš predpoklad správny a slová z nových párov by mali byť podobné tým v ukážkovej množine, táto množina by mala obsahovať aj relevantné páry.

Okrem zmenšenia veľkosti množiny overovaných párov na prijateľné číslo má definovanie tohto okolia ešte jednu funkciu. Pri klasifikačných algoritmoch je niekedy potrebné použiť množinu neoznačených (tzv. *unlabeled*) vzoriek. Tieto neoznačené vzorky by mali byť z oblasti, kde očakávame aj pozítivne vzorky. Náš prístup tento predpoklad, na rozdiel od napríklad náhodného výberu párov, spĺňa.

Ako ďalšie rozšírenie našej metódy do budúcnoti navrhujeme vytvoriť na základe vstupu akýsi filter, ktorým zmenšíme dĺžku slovníka. Navrhujeme dva prístupy, ako takýto filter zostrojiť. Morfologický filter by do priestoru premietal len tie slová, ktoré sa morfologicky podobajú na slová zo vstupnej množiny. Ak sú vo vstupnej množine podstatné slová v singulári, tak len takéto sa budú používať. Druhý filter je sémantický. Ten je náročnejší a vyžaduje ďalšiu sémantickú informáciu o vstupe i o slovách v slovníku. Ak napríklad máme na vstupe geografické entity, tak do priestoru by sme radi premietli len tieto. Tento prístup si vyžaduje ďalšiu prácu so vstupnou množinou a môže do metódy zbytočne vnášať nepresnosti.

5.1.4 Ohodnotenie kandidátov

Napokon chceme na základe podobnosti s ukážkovými pármi ohodnotiť naše kandidáty. Podobnosť pritom máme na mysli výlučne v zmysle podobností vektorov týchto pá-

rov. Za účelom tohto ohodnocovania sme navrhli viacero metrík, ktoré chceme overiť pri experimentovaní s našou metódou. Dve metriky sú založené na jednoduchom rátaní podobnosti vektorov. Posledná používa menej zvyčajnú variantu binárneho klasifikovania nazývanú PU učenie alebo po anglicky *Positive-Unlabeled Learning*. V nasledujúcich odsekoch si tieto metriky opíšeme:

• *Priemerná podobnosť*. Prvou metrikou ktorú si opíšeme je priemerná podobnosť ku ukážkovým párom. Všeobecný vzorec pre túto metriku je:

$$score(x) = \sum_{s \in S} weight(s) \times sim(x, s), \tag{15}$$

kde x hodnotený pár, S je naša vstupná množina, sim je metrika podobnosti medzi dvoma vektormi a $weight$ je váha priradená každému páru z množiny S. Ako vektorovú metriku podobnosti uvažujeme dve bežne používané miery: kosínusovú podobnosť a euklidovskú podobnosť.

Váha $weight$ slúži na ohodnotenie relevantnosti každého páru zo vstupnej množiny. Hlavnou myšlienkou za týmto váhovaním je zmenšiť vplyv párov zo vstupnej množiny, ktoré sú menej podobné zvyšku tejto triedy. Toto sa môže stať napríklad pri zle zadanom vstupe používateľa, keď zadá nie až tak typický pár, prípadne keď urobí preklep a podobne. Snažíme sa teda znížiť vplyv takýchto outlierov na výsledné skóre. Bez tejto penalizácie má každý ukážkový pár váhu w rovnú $|S|^{-1}$. Penalizáciu rátame ako priemernú podobnosť jedného páru ku ostatným párom z množiny:

$$weight(s) = \sum_{u \in S \setminus s} \frac{sim(s, u)}{|S| - 1} \tag{16}$$

kde s je pár z ukážkovej množiny S, pre ktorý váhu rátame. Túto myšlienku môžeme ďalej rozviesť a na tieto váhy použiť normalizáciu funkciou *softmax*.

• *Maximálna podobnosť*. Tento prístup je podobne jednoduchý ako predošlý. Namiesto priemeru však rátame maximálnu podobnosť hodnoteného kandidáta s pármi v ukážkovej množine. Ráciom za touto metrikou je fakt, že niekedy sémantické vzťahy tvoria v priestore viacero menších zhlukov, nie len jeden veľký [Fu et al.,

2014]. Pri použití maximálnej podobnosti teda stačí, aby sa kandidát podobal na jedného svojho suseda zo svojho zhluku. Nemusí sa nutne podobať na každý zadaný pár. Túto mieru môžeme vzorcom vyjadriť ako:

$$score(x) = arg\max_{s \in S} sim(S, x) \tag{17}$$

kde význam symbolov zostáva rovnaký ako v predošlých vzorcoch.

• *PU učenie.* Každý pár slov buď má alebo nemá hľadaný vzťah. Určenie, či ho má, je vlastne problém binárnej klasifikácie. Pri bežnej binárnej klasifikácii máme v trénovacej množine prítomné pozitívne aj negatívne vzorky. V našom prípade však negatívne vzorky nemáme k dispozícii. V takomto prípade sa hovorí o tzv. jednotriedovej klasifikácii. Aj keď je to variant binárnej nemôžeme spravidla použiť klasické metódy klasifikácie. Existuje však viacero algoritmov špecializovaných práve na jednotriedovú klasifikáciu [Irigoien et al., 2014].

  Obmenou tohto prístupu je PU (*Positive-Unlabeled*) učenie, v ktorom je okrem pozitívnych vzoriek vstupom pri trénovaní algoritmu aj množina vzoriek, o ktorých nevieme povedať do akej triedy patria. Môžu sa v nej nachádzať pozitívne aj negatívne vzorky. Informácia z tejto množiny v algoritmoch pomáha lepšie pochopiť priestor, v ktorom sa pozitívne vzorky vyskytujú. Tento prístup bol tiež aplikovaný pri viacerých úlohách pracujúcich s prirodzeným jazykom [Li and Liu, 2003]. PU učenie sa tiež zvykne implementovať obmenou algoritmu *SVM*. V našom prípade môžeme ako množinu neoznačených dát rovno použiť kandidáty z predošlého kroku.

  Otázkou je aj to, čo bude pri našej klasifikácii tvoriť vzorky. V predošlej kapitole sme uviedli viacero spôsobov, ako reprezentovať vzťahy. Tieto spôsoby sú uvedené v Tabuľke 2 na strane 28. Ako atribúty vzoriek pri klasifikácii môžeme použiť priamo zložky týchto reprezentácií. Tieto zložky vznikli priamo z latentných čŕt jednoduchými vektorovými operáciami a preto sú samé osebe latentné, zmysel nadobúdajú len ako jeden celok. Klasické klasifikačné algority sa spoliehajú na objavovanie závislostí medzi atribútmi. Problémom môže byť fakt, že nechávame na algoritmus, aby sa pokúsil zistiť z pomerne malého množstva dát, že sa snažíme nájsť podobnosť medzi vektormi. Netreba zabúdať, že predpokladéme len

niekoľkod desiatok pozitívnych vzoriek. Atribúty vzorky P v klasifikácii vyzerajú nasledovne:

$$P = \langle v_1, v_2, ..., v_j \rangle, \tag{18}$$

kde j je dĺžka vektorov v našom priestore. v sú zložky vektora tohto páru.

Namiesto toho navrhujeme menej intuitívny model klasifikácie, kde atribútmi vzoriek sú miery podobnosti s ukážkovými pármi S. Najprv množinu S orežeme o určitý počet párov. Z týchto párov sa stanú pozitívne vzorky pri klasifikovaní. Orezanú množinu nazvime S'. Každý atribút vzorky p zodpovedá podobnosti klasifikovaného páru s jedným členom množiny S'. Napríklad j-ty atribút vzorky zodpovedá podobnosti páru s j-tym členom množiny S'. Táto podobnosť môže byť pritom nadefinovaná ľubovoľne, ako v predošlých algoritmoch aj tu uvažujeme najmä o klasickej euklidovskej a kosínusovej podobnosti. Celá vzorka P vytvorená z páru p teda vyzerá nasledovne:

$$p = \langle sim(p, s_1), sim(p, s_2), ..., sim(p, s_{|S'|}) \rangle, \tag{19}$$

kde s sú páry z množiny S' a sim je ľubovoľná funkcia podobnosti.

Od takto navrhnutej klasifikácie si sľubujeme väčšiu kontrolu nad tým, ako klasifikátory pracujú so sémantickou informáciou. Toto dosiahneme vďaka tomu, že dokážeme skrz funkciu podobnosti usmerňovať podobu vzoriek. Tým sme abstrahovali spôsob reprezentácie párov a hľadania sémantickej podobnosti medzi nimi od klasifikačného algoritmu. Môžeme zmeniť spôsob, akým porovnávame páry vektorov, bez toho, aby sme museli modifikovať klasifikátor. Keďže nič okrem tejto funkcie nemusíme pri zmene reprezentácie meniť, môžeme priamo porovnávať výsledky odlišných prístupov bez toho, aby sme museli dáta ďalej upravovať a zohľadňovať ďalšie okolnosti, ktoré môžu z takého upravovania plynúť. Takisto sme čiastočne vyriešili problém početnosti atribútov, ktorú sme zmenšili na počet prvkov vo vstupnej množine.

Výsledkom všetkých troch metód je, že každému kandidátu je priradené reálne číslo, ktoré by malo korelovať s pravdepodobnosťou, že pár je inštanciou relevantného

sémantického vzťahu. Napokon nam teda stačí len páry podľa tohto hodnotenia zoradiť a vybrať tie najlepšie. Tie sa stávajú výstupom našej metódy.

5.2 Vyhodnocovanie výkonnosti metódy

Pri vyhodnocovaní si musíme uvedomiť niektoré okolnosti vyplývajúce z povahy objavovaných znalostí, v našom prípade sémantických vzťahov. Sémantické vzťahy môžu mať rozličnú úroveň špecifickosti. Napríklad vzťah medzi krajinou a hlavným mestom je špecifikácia vzťahu medzi krajinou a akýmkoľvek mestom. Tento vzťah zasa môže byť špecifikáciou vzťahu medzi krajinou a akoukoľvek geografickou entitou. Aj pre človeka môže byť v niektorých prípadoch náročne určiť, aká miera špefickosti je definovaná vstupnou množinou ukážkových párov. Pri zostrojovaní tejto množiny preto treba dbať na to, aby vybrané páry adekvátne reprezentovali hľadaný vzťah a jeho špecifickosť. Napríklad môžeme chcieť objavovať páry vzťahov medzi krajinami a mestami v nich. Ak by vstupná množina obsahovala veľký podiel hlavných miest, zrejme by lepšie zachytávala špecifickejší vzťah medzi krajinou a jej hlavný mestom, ako ten, ktorý naozaj chceme objavovať.

Naša metóda v konečnom dôsledku rieši daný problém klasifikáciou. Potom ako sa z metódy vráti zoradený zoznam kandidátov, používateľovi sa predloží len najlepších n jedincov z tohto zoznamu. Tým ich vlastne naša metóda klasifikuje na tie, ktoré považuje za relevantné a tie ktoré za relevantné nepovažuje. Pri klasifikačných problémoch poznáme viaceré metriky, ktoré sa používajú pri vyhodnocovaní jej úspešnosti. Tradične sa používa presnosť a pokrytie (*precision* a *recall* v anglofónnej literatúre). V prípade objavovania nových párov je pre náš z týchto dvoch dôležitejšia presnosť. Nezáleží nám teda na tom, aby sme odhalili všetky páry daného vzťahu, ktoré sa v prirodzenom jazyku nachádzajú. Viac nás zaujíma, aby tie čo najviac nami nájdených párov malo hľadaný sémantický vzťah.

Pokrytie ako metrika nie je pre nás až tak dôležitá hlavne preto, že pri niektorých vzťahoch nevieme dopredu povedať, koľko relevantných párov sa v jazyku vôbec vyskytuje. Nevieme napríklad jednoducho vyčísliť, kolko antonymických párov obsahuje ten-ktorý prirodzený jazyk a už vôbec nevieme odhadnúť, koľko z týchto párov sa aj naozaj vyskytlo v pôvodnom textovom korpuse v takej miere, aby sa tento vzťah odrazil aj v štatistickom modeli tvorenom vektormi latentných čŕt.

Táto argumentácia platí všeobecne pre veľa metód získavania znalostí. Často nedokážeme odhaliť všetky relevantné znalosti, ktoré by sme chceli získať, no dokážeme ohodnotiť tie, ktoré reálne získame. Pokrytie sa dá merať na testovacích dát rozličnými technikami krížovej validácie, v našom prípade však len ťažko dokážeme posúdiť, či sú testovacie sady dostatočne reprezentatívne pre daný vzťah v jazyku.

Na druhej strane presnosť je metrika, ktorú dokážeme veľmi jednoducho manuálne vyhodnocovať. Je aj metrikou, na ktorú kladieme najväčší dôraz. Taktiež môžeme používať metriky súvisiace s presnosťou ale zohľadňujúce aj poradie prvkov. Medzi tieto patrí napríklad *Precision@K* alebo *NDCG*.

Keďže pokrytie nie je našou prioritou, neprekážejú nám až tak veľmi niektoré obmedzenia, ktoré sme v priebehu metódy vykonali a ktoré môžu odrezať aj relevantné páry. Ide tu najmä o proces výberu kandidátov. Zlepšenie pokrytia naopak predpokladáme, ak novonájdené páry zaradíme do vstupnej množiny a celú metódu spustíme odznova. Myslíme si, že tieto nové páry môžu v niektorých prípadoch do klasifikácie priniesť novú sémantickú informáciu, ale aj rozšíriť prehľadávané okolie. Celú metódu takto opakovať s novými znalosťami môžeme dovtedy, dokiaľ nám prináša nové výsledky.

5.3 Aplikácia metódy v kontexte znalostných štrukúr

Naša metóda patrí medzi metódy objavovania znalostí. Dokáže na základe používateľom nadefinovaného vstupu odhaliť nové vzťahy medzi slovami. Myslíme si, že takáto metóda môže byť aplikovaná pri úlohách znalostného inžinierstva. Aj keď jej výsledky nebudú dostatočne dobré pre automatické objavovanie, stále sa môže používať ako pomôcka pre znalostných inžinierov. Môže im výrazne urýchliť objavovanie nových inštancií vzťahov medzi konceptami tým, že zúži množinu možných párov, ktorú musia manuálne prehľadať. Inými slovami môže im odporúčiť páry, medzi ktorými sa hľadaný vzťah môže nachádzať.

Znalostné štruktúry, ako napríklad ontológie, však môžeme použiť aj ako zdroj ďalšej sémantickej informácie, ktorú použijeme počas metódy. Treba si však uvedomiť, že v znalostných štruktúrach spravidla prechádzame z úrovne termov na úroveň konceptov. Tie treba rozlišovať už pri spracovaní textového datasetu, čo nie je triviálna úloha. Je potrebné tu riešiť také úlohy ako rozpoznávanie vlastných pomenovaní, odhaľovanie

synoným či rozlišovanie homoným. Dodatočnú sémantickú informáciu z týchto štruktúr môžeme použiť na viacerých miestach v metóde:

- *Ako ukážkovú množinu párov.* Definované inštancie vzťahov z jednej triedy môžeme použiť ako vstupnú ukážkovú množinu párov. Vstup teda nebude tvorený manuálne, ale automaticky extrahovaný z už existujúcich štruktúr.

- *Pri definovaní prehľadávaného okolia.* Pri opise druhého kroku metódy sme vyslovili predpoklad, že rovnaké sémantické vzťahy sa vytvárajú medzi termami z určitých tried. Napríklad vzťah krajiny a jej hlavného mesta je vzťah, kde zložky inštancií patria buď do množiny krajín alebo do množiny miest. Namiesto hľadania týchto množín vo vektorovom okolí ich môžeme priamo extrahovať zo znalostných štruktúr, kde môžu byť uložené ako triedy konceptov.

5.4 Implementácia

Metódu sme implementovali v rámci našej vytvorenej knižnice v jazyku *Python*. Táto knižnica je dostupná aj na GitHube[1]. Technická dokumentácia tejto knižnice je uvedené v technickej dokumentácii v Prílohe B. Pri implementovaní tejto knižnice sme sa snažili naplniť tieto požiadavky:

- Možnosť spustiť našu navrhnutú metódu vo viacerých variantách.
- Jednoduché rozhranie k tejto metóde v príkazovom riadku.
- Schopnosť pracovať s binárnym formátom vektorov latentných čŕt, ktoré používajú bežné knižnice.
- Možnosť rátania rozličných metrík ukážkových množín, ktoré si predstavíme v Kapitole 6.
- Vyhodnocovanie súborov s výsledkami.

Táto knižnica nedokáže vyrátať vektory latentných čŕt z textového korpusu, na tento účel sa dá použiť napríklad Python knižnica *gensim*[2]. Túto knižnicu používame aj

[1]Dostupná na: https://github.com/matus-pikuliak/word-embeddings
[2]Dostupná na: https://radimrehurek.com/gensim/

na prácu s datasetom vektorov latentných čŕt. Ďalšou dôležitou použitou knižnicou je *SVM Perf*[3], pomocou ktorej rátame PU učenie.

V rámci knižnice sme vytvorili aj spustiteľný skript, pomocou ktorého môže používateľ jednoducho z prostredia príkazového riadku spustiť našu metódu nad vstupnou množinou ukážkových párov. Táto množina je pritom definovaná v textovom súbore špecifického formátu. Na každém riadku takéhoto súboru je jeden pár, čo je vlastne dvojica slov oddelených medzerou. Príklad takéhoto súboru je:

```
Atény Grécko
Paríž Francúzsko
Berlín Nemecko
```

Výstupom je podobný súbor, pred každým párom sa ale ešte nachádza znak '?' oddelený od páru tabulátorom. Tento znak sa používa pri manuálnom vyhodnotení ako príznak správnosti. Keď používateľ prechádza tento výstupný súbor, môže tento znak nahradiť znakom symbolizujúcim, či je daný pár správny, čiastočne správny alebo nesprávny:

- `r` - Správny pár
- `w` - Neprávny pár
- `p` - Čiastočne správny pár

Ak sa takto vyhodnotia všetký páry vo výstupnom súbore, naša knižnica ďalej ponúka možnosti ako tento súbor spracovať a získať z neho vyhodnocovacie metriky. Takýto ohodnotený súbor môže vyzerať napríklad takto:

```
r Caracas Venezuela
w Paríž Nemecko
r Madrid Španielsko
```

[3]Dostupná na: http://www.cs.cornell.edu/people/tj/svm_light/svm_perf.html

6 Experimenty

V tejto kapitole opisujeme a vyhodnocujeme experimenty, ktoré sme vykonali v rámci našej práce. Najpr opíšeme datasety, s ktorými sme experimenty vykonávali. Uviedeme niekoľko výskumných otázok, ktoré sme si položili a opíšeme experimenty, ktorými sme sa ich snažili zodpovedať. Výsledkom našich experimentov je overenie správnosti našej metódy, ale aj získanie nových znalostí o vzoroch, ktoré sémantické vzťahy zanechávajú v priestore vektorov latentných čŕt.

6.1 Opis použitých datasetov

6.1.1 Vektory latentných čŕt

Pri našich experimentoch sme použili už hotový priestor vektorov latentných čŕt. Použili sme dataset uverejnený na stránke knižnice word2vec[4]. Tieto vektory sú natrénované podľa algoritmu rozoberaného v Kapitole 4. Vektory v ňom majú dĺžku 300. Obsahuje presne 3 milióny termov. Tieto termy môžu byť buď slová alebo frázy, pričom obsahuju aj diakritiku a interpunkciu. Boli natrénované na textovom datasete získaného zo žurnalistického agregátora *Google News*. Termy sme orezali z 3.000.000 na 100.000 najpoužívanejších anglických slov. Účelom tohto orezania bolo zmenšiť výpočtovú zložitosť, pričom sme stratili len slová, ktoré sú zriedkavejšie a aj ich vektory sú teda zrejme menej dobre natrénované.

Pri vlastnom vytváraní vektorového priestoru by sme mali väčšiu kontrolu nad výsledkom. Mohli by sme experimentovať napríklad s ich dĺžkou, architektúrou neurónovej siete a podobne. Taktiež by sme mohli použiť rozmanitejšie zdroje textových dát, aj napriek tomu že spravodajstvo má pomerne široký záber tém. Nevýhodou je ale časová i pamäťová náročnosť takéhoto vytvárania.

6.1.2 Ukážkové páry

Pre vytvorenie trénovacích množín ukážkových párov sme sa inšpirovali existujúcimi datasetmi použitými na podobné úlohy. Prvý dataset je z Mikolov et al. [2013b], kde ho použili pri objavovaní analógií v priestore vektorov latentných čŕt. Tento dataset je veľmi

[4]Dostupné na: https://code.google.com/archive/p/word2vec/

kvalitný, každá trieda v ňom je veľmi jednoznačná a konkrétna. Jedna trieda vzťahov máva približne 35 inštancií.

Z neho sme pre naše experimentovanie vybrali a upravili 4 množiny párov s určitým sémantickým vzťahom. Tieto sme upravili na zhodnú veľkosť 25 párov, ktoré sme uložili do súboru. Formát takýchto súborov je opísaný v prílohe B, sekcii 2. Každá množina má svoj názov, ktorý v tejto práci budeme uvádzať osobitým písmom, napr. `Capitals`. Všetky spomenuté množiny sú uvedené aj v Prílohe A. Tieto množiny sú:

1. `Capitals` - Vzťah medzi krajinou a jej hlavným mestom (napr. **France-Paris**).
2. `Cities` - Vzťah medzi štátom USA a mestom, ktoré sa v tomto štáte nachádza (napr. **Florida-Tampa**).
3. `Currency` - Vzťah medzi krajinou a jej menou (napr. **Russia-ruble**)
4. `Gender` - Vzťah medzi mužským a ženským tvarom slova (napr. **king-queen**)

Tieto množiny spĺňajú naše požiadavky. V prvom rade sa v troch prípadoch jedná o faktografický sémantický vzťah. Práve o takéto vzťahy máme záujem z hľadiska potencionálneho použitia našej metódy v kontexte znalostných štruktúr. Taktiež sú tieto dvojice jasne objektívne definované, napr. každá krajina má jasne definované akú menu používa. A napokon si myslíme, že tieto vzťahy (azda s výnimkou `Gender`) sa mohli aj dostatočne často manifestovať v textovom korpuse. V dátach zo žurnalistických zdrojov budú zrejme často spomínané geografické entity a vzťahy medzi nimi. Niektoré iné druhy vzťahov sa takto explicitne v texte prejaviť nemusia a mohli by sme pri nich narážať na problém tzv. *background knowledge*.

Okrem týchto štyroch základných množín sme skúsili použiť našu metódu aj na iné typy vzťahov. Z datasetu od Mikolova sme prebrali niektoré gramatické vzťahy. Taktiež sme vybrali niekoľko vzťahov z datasetu z Jurgens et al. [2012], ktorý bol použitý pri jednej z evaluačných úloh série *SemEval*. Pôvodne obsahuje 79 tried, každá s približne 40 pármi. My sme z tohto datasetu vybrali 2 vzťahy. Zo všetkých sme opäť vytvorili množiny s 25 pármi. Tieto zvyšné množiny sme netestovali tak podrobne ako vyššie spomenuté 4, vykonali sme nad nimi len základné experimenty, ktoré sú spomenuté v podkapitole 6.7. Patria sem nasledovné množiny, ktorých plné znenie je taktiež v prílohe:

1. `Past Tense` - Vzťah medzi slovesným podstatným menom a týmto slovesom ohnutým do jednoduchého minulého času (napr. **jumping-jumped**).

2. `Comparative` - Vzťah medzi prídavným menom a jeho komparatívom (napr. **high-higher**).

3. `Nationality` - Vzťah medzi krajinou a prídavným menom vzťahujúcim sa na jej obyvateľov (napr. **Russia-Russian**)

4. `Knowledge` - Vzťah medzi vedným odborom a predmetom jeho skúmania (napr. **mathematics-numbers**)

5. `Part` - Vzťah medzi množinou častí a celkom, ktorý tvoria (napr. **forest-trees**)

6.2 Výskumné otázky

Pred začiatkom experimentovania sme si položili viacero výskumných otázok, ktoré sme chceli zodpovedať. Súvisia s overením jednotlivých krokov metódy aj výkonu metódy ako celku. Tieto otázky sú:

1. Je náš spôsob generovania kandidátov správny? Touto otázkou sa pýtame, či dokážeme pomocou nášho spôsobu vygenerovať takú množinu kandidátov, ktorá bude obsahovať aj relevantné páry. Je pre nás zaujímavé aj zistiť, akú časť relevantných párov takto dokážeme získať do množiny kandidátov.

2. Je náš spôsob zoraďovania kandidátov správny? Touto otázkou sa pýtame, či naše metriky použité pri zoraďovaní naozaj dokážu kandidáty zoradiť tak, aby sa tie relevnantné objavili na popredných pozíciách. Našou snahou je mať relevantné páry čo najvyššie v zoradenom zozname.

3. Asi najpodstatnejšou otázkou je ako dobre si naša metóda počína pri hľadaní nových relevantných párov. Pokiaľ dokážeme správne generovať kandidáty a potom ich správne zoraďovať, čo je predmetom prvých dvoch otázok, mal by byť aj celkový výkon metódy pomerne dobrý.

4. Poslednou otázkou je, či pri použití viacerých ukážkových párov dokážeme naozaj presnejšie definovať daný sémantický vzťah. Aby mala naša metóda zmysel, musí

platiť, že model vzťahu získaný z jedného jedinca je menej presný ako model získaný z viacerých.

6.3 Experiment s generovaním kandidátov

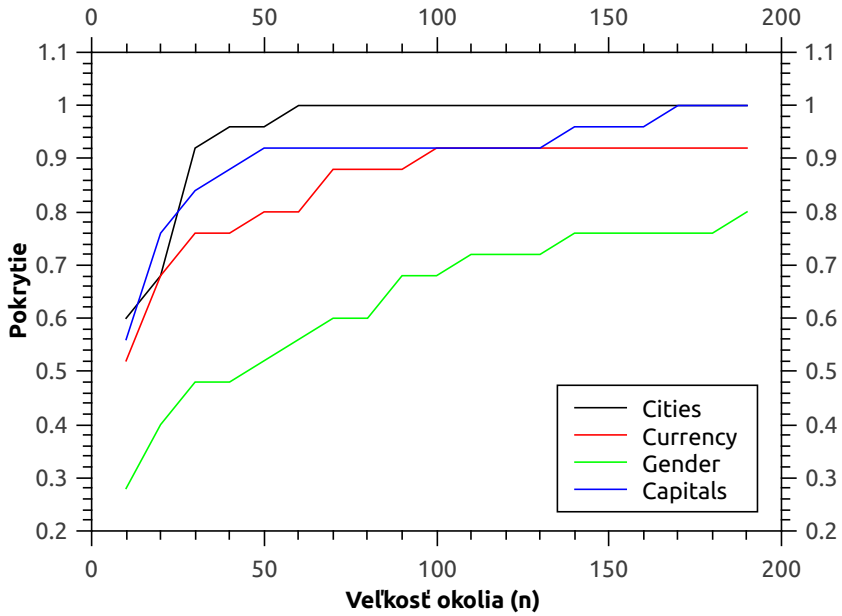
Tento experiment sme navrhli v snahe zodpovedať našu prvú výskumnú otázku. Pokiaľ chceme aby naša metóda prisudzovala relevantným kandidátom vysoké skóre, musia sa najprv v zozname kandidátov nejaké relevantné kandidáty nachádzať.

Pre overenie tohto kroku metódy sme navrhli jednoduchý experiment. Zoberieme jeden ukážkový pár z množiny. Ostatné páry okrem tohto jedného použijeme na generovanie kandidátov v súlade s návrhom našej metódy. Pokiaľ sa vynechaný pár nachádza v zozname kandidátov, berieme to ako úspech. Toto opakujeme pre každý jeden pár z pôvodnej množiny a na konci vyrátame pomer úspešných párov. Čím vyšší je tento pomer, tým viac relevantných kandidátov očakávame pre danú množinu. Tento pomer sme ako metriku nazvali *pokrytie množiny*. Spôsob rátania tejto metriky je zachytený aj nasledujúcim pseudokódom:

```
# Premenna 'set' predstavuje ukazkovu mnozinu
# Jej metoda pairs vracia zoznam parov
def set_recall(set)
  # Vyberie len pary obsiahnute v kandidatoch
  included_pairs =
  set.select do |pair|
    candidates = (set - [pair]).generate_candidates
    candidates.include?(pair)
  end
  # Vydeli ich pocet celkovym poctom parov
  return included_pairs.size / set.size
end
```

Ako okolie každého vektoru, ktoré sa použije pri karteziánskom súčine používame n jemu najpodobnejších vektorov. Dokopy sa teda z jedného páru vygeneruje n^2 nových kandidátov. Vyskúšali sme správanie tohto kroku metódy pre rozličné veľkosti n. Vý-

sledky z tohto experimentu sme zaznamenali pre naše 4 množiny ukázkových párov na Obrázku 5.



Obr. 5: *Vzťah medzi veľkosťou okolia použitého pri generovaní kandidátov a pokrytím množiny pre naše 4 základné ukážkové množiny.*

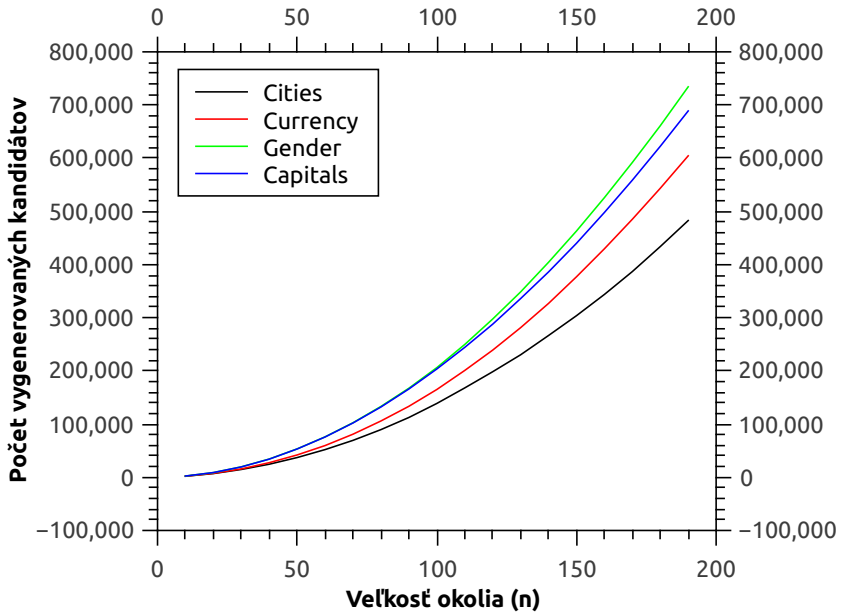
Z grafov vidíme, že aj pri relatívne malých okoliach sme vedeli dosiahnuť pomerne vysoké pokrytie. Myslíme si teda, že náš spôsob generovania kandidátov vhodný a v množine kandidátov sa zvyknú nachádzať aj relevantné páry. Pri dvoch množinách sme v rámci nášho experimentu dokonca dosiahli 100% úspech. Celkovo najhorší výsledok mala množina `Gender`. Usudzujeme, že sémantika vzťahu v tejto skupine je omnoho menej doménovo závislá. Pri prehľadávaní priestoru môžeme čakať že mestá či krajiny budú vytvárať akési zhluky, na tomto predpoklade je založený tento krok našej metódy. Nečakáme však že takýto zhluk budú tvoriť mužské tvary slov, keďže sú natoľko doménovo odlišné.

Na grafe na Obrázku 6 vidíme že rast počtu kandidátov zhruba kopíruje predpokladanú kvadratickú funkciu. Postupne sa však medzi množinami objavujú rozdiely. Tento jav je spôsobený tým, že kandidáty vygenerované z jednotlivých párov sa navzájom prekrývajú. Môže sa tak stať že jeden kandidát sa nachádza v susedstve viacerých párov. Vo

Tabuľka 3: *Výsledky z experimentu s generovaním jedincov pre veľkosť okolia 100. Zistili sme počet generovaných kandidátov pre danú množinu, ako aj pokrytie množiny.*

	Capitals	Currency	Cities	Gender
Pokrytie množiny	0,92	0,92	1	0,68
Počet kandidátov	203.617	165.054	138.814	206.205

výslednom zozname ho však samozrejme uvádzame len raz. Rozdiely medzi množinami sú zrejme spôsobené tým, ako blízko sa pri sebe nachádzajú jednotlivé ukážkové páry. Ak sú veľmi blízko, zvýši sa počet prekryvov.



Obr. 6: *Vzťah medzi veľkosťou okolia použitého pri generovaní kandidátov a množstvom vygenerovaných kandidátov.*

Pre naše ďalšie experimenty sme sa rozhodli používať okolie veľkosti 100. Túto hranicu sme zvolili preto, že pokrytie v tom bode už bolo väčšinou cez 90%, ale počet kandidátov ešte nebol priveľmi vysoký a teda ďalšia práca s týmito kandidátmi bude rýchlejšia. Hodnoty metrík pri nastavení veľkosti okolia na 100 sme zaznačili do Tabuľky 3.

6.4 Experiment s hodnotením kandidátov

Druhým experimentom sme chceli nájsť odpoveď na našu druhú výskumnú otázku. Zaujímalo nás, či dokážeme kandidáty zoradiť tak, aby sa relevantné páry dostali na samý začiatok zoradenia. Najpriamočiarejším overením by bolo ručne vyhodnocovať zoradenia vzniknuté z naších ukážkových množín. Tento postup, aj keď presný, by bol veľmi časovo náročný, hlavne ak by sme chceli overiť viacero možných variácií našich hodnotiacich algotirmov. Navrhli sme preto jednoduchší plne automatizovaný experiment s nasledujúcimi krokmi:

1. Z množiny vygenerujeme kandidáty
2. Ukážkovú množinu rozdelíme v pomere 80%-20% na dve množiny, ktoré sme nazvali „trénovacia“ a „testovacia“.
3. Trénovaciu množinu použijeme na ohodnotenie aj množiny kandidátov (čo sú unlabeled vzorky), aj testovacej množiny (čo sú pozitívne vzorky) pomocou vybraného algoritmu.
4. Ohodnotené páry zoradíme a zistíme, na akých pozíciách sa v tomto zoradení vyskytujú páry z testovacej množiny. Výstupom sú práve tieto pozície.

Tento proces môžeme zachytiť aj pomocou pseudokódu:

```
def positions(seed_set)
  candidates = seed_set.generate_candidates
  # Splits set to training and testing sets
  training_set, testing_set = set.training_testing_sets
  evaluated = candidates + testing_set
  evaluated.each do |pair|
    # Ohodnotenie paru pomocou nasho algoritmu
    # hodnotenia
    pair.rate(training_set)
  end
  evaluated.sort_by(&:rating)
  # Najdeme pozicie na akych sa v celkovom hodnotenom
  # zozname nachadzaju testovacie vzorky
  positions = testing_set.map do |testing_pair|
    evaluated.find_position(testing_pair)
  end
  return positions
end
```

Pri našom experimentovaní sme ho nechali prejsť pre každú množinu a algoritmus 20 krát. Výsledok tohto experimentu bol vektor pozícií, z ktorého sme vyrátali medián. Túto metriku budeme ďalej označovať ako *medián pozícií*. Hovorí nám o tom, na akých pozíciách môžeme čakať relevantné páry a môžeme vďaka nej porovnávať jednotlivé algoritmy. Čím menší majú tento medián, tým vyššie skóre dávajú relevantným vzorkám z testovacej množiny oproti ostatným kandidátom.

Pri našom experimente sme použili nasledovné algoritmy hodnotenia kandidátov. Všetky spôsoby vychádzajú priamo z návrhu. Ďalej budeme tieto algoritmy označovať ich skráteným názvom, ktorý uvádzame na začiatku opisu:

- Avg - Hodnotenie páru je priemerná podobnosť vektoru tohto páru ku vektorom párov z ukážkovej množiny. Vyskúšali sme viacero variánt tohto algoritmu. Podľa toho, či poutívame euklidovskú alebo kosínusovú vzdialenosť tento algirmut

Tabuľka 4: *Výsledky z experimentu s hodnotením jedincov. V tabuľke sú zaznačené mediány pozícií pre rozličné algoritmy hodnotenia a rozličné ukážkové množiny.*

	Cities	Currency	Capitals	Gender
PU_{cos}	31	130	10	2236
PU_{euc}	**7**	**48**	**6**	947
Max_{euc}	18	1762	15	889
Avg_{cos}	82	339	9	857
Avg_{euc}	10	163	8	654
$Avg_{euc\_softmax}$	8	71	15	**358**

označujema ako Avg_{euc} alebo Avg_{cos}. Tento prístup k dolným indexom sa dá aplikovať aj na zvyšné dva algoritmy hodnotenia, kde sme tiež aplikovali obe miery podobnosti. Ak používame aj softmaxovú normalizáciu, značíme to tiež v dolnom indexe, napr.: $Avg_{euc\_softmax}$.

- Max - Hodnotenie páru je jeho maximálna podobnosť ku párom z ukážkovej množiny.

- PU - Hodnotenie je vyrátané pomocou klasifikátora netrénovaného v súlade s návrhom uvedenom v podkapitole 5.1.4. Ako algoritmus PU učenia sme použili variantu populárneho algoritmu SVM. Táto spočíva v použití *ROC-Area* stratovej funkcii pri trénovaní modelu. Ako implementáciu tohto algoritmu sme použili knižnicu *SVM Perf*[5] [Joachims, 2006]. Trade-off sme nastavili na 0.01, ako štrukturálny učebný algoritmus sme použili tzv. *dual l-slack*.

Výsledky tohto experimentu vyrátané sú v Tabuľke 4. Zďaleka najhorší výsledok dosiahla množina `Gender`. Dôvody budú zrejme rovnaké ako pri predošlom experimente. Páry v tejto množine síce majú rovnaký vzťah, no medzi sebou až tak veľmi nesúvisia. Dôvodom môže byť aj to, že sme narazili na obdobu problému tzv. *background knowledge*. Mestá a meny krajín sa v texte spravodajstva zrejme budú uvádzať pomerne často. Explicitne v texte sa však vzťah medzi mužským a ženským tvarom slova (napr. **host-hostess**) až tak často vyskytovať nebude.

Ak si odmyslíme túto jednu najhoršiu množinu, môžeme vidieť, že najlepšie výsledky pre každý vstup má algoritmus PU_{euc}. S týmto prístupom sa podľa nás dá ďalej pracovať

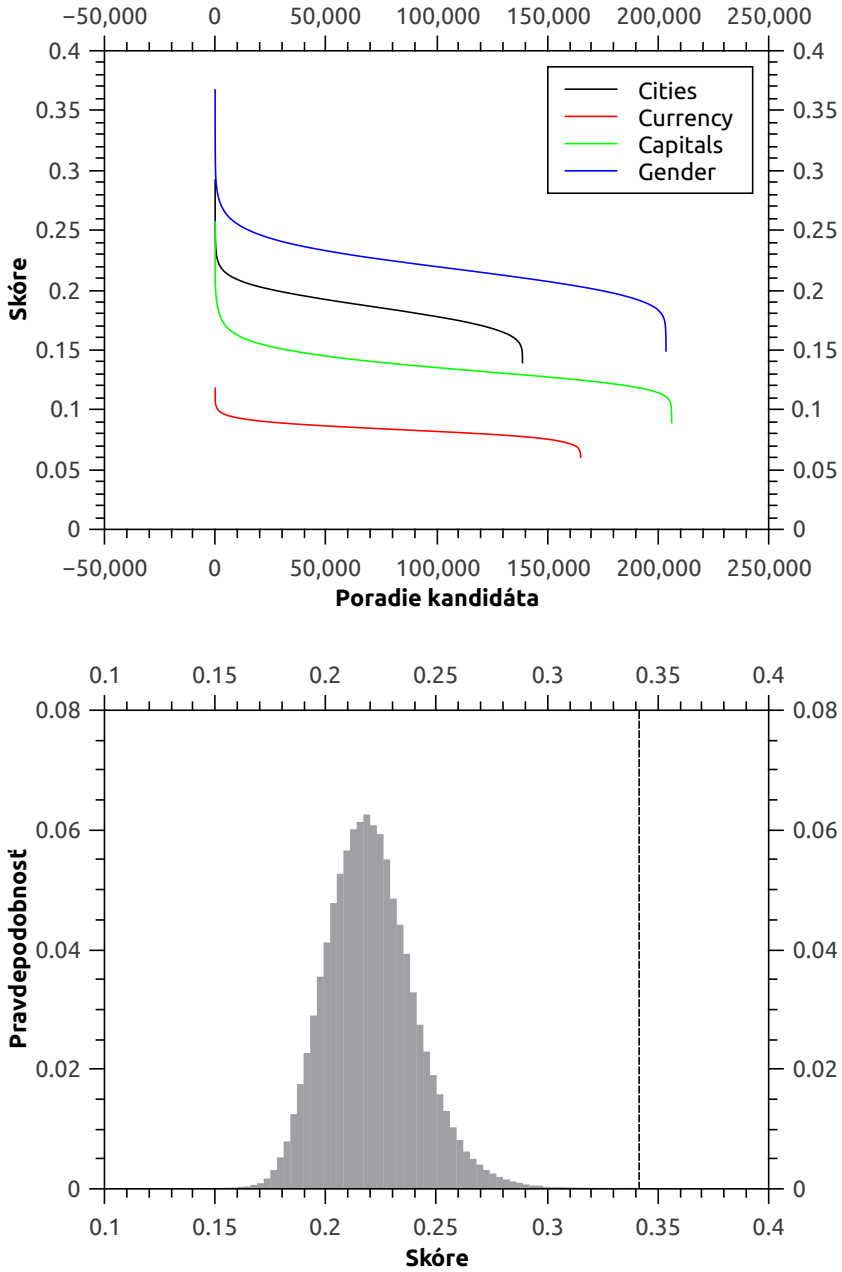
[5]Dostupná na: https://www.cs.cornell.edu/people/tj/svm_light/svm_perf.html

a odladením rozličných parametrov možno dostať aj lepšie výsledky. Jeho najlepšia hodnota pozícia 6 pre páry z množiny `Capitals`. Môžeme teda očakávať že pozitívne vzorku sa budú nachádzať aj v najlepšej desiatke nášho zoradenia. Tieto výsledky nám dávajú nádej, že dokážeme naozaj správne zoraďovať generované kandidáty. Pre 3 zo 4 množín sme boli schopní umiestniť medián to prvej 100, ktorú chcem manuálne vyhodnotiť. Tieto výsledky však považujeme za pomerne nepresné a skutočný výkon našej metódy overíme až v nasledujúcom experimente.

Ostatné algoritmy sú pomerne vyrovnané a vo všeobecnost dosahujú pre rozličné množiny pomerne nevyrovnané prístupy. Rozdiely však nie sú veľmi veľké a len zriedkavo presiahnu rozdiel jedného rádu. Tento experiment nám teda nepreukázal, ktoré z týchto prístupov sú naozaj dobré alebo naozaj zlé. Najhoršie sa nám z tejto tabuľky javí algoritmus Max, ktorý skončil zakaždým v najhoršej trojke. Toto riešenie je zrejme najviac náchylné na chyby, keďže stačí náhodná podobnosť s jedným so vzorom a kandidát môže dostať veľmi vysoké hodnotenie.

Pri priamom porovnaní kosínusových a euklidovských variánt algoritmov PU a Avg môžeme vidieť, že zakaždým zvíťazí tá euklidovská. To nám naznačuje, že tá sa lepšie hodí pre našu úlohu. K analýze, prečo je to tak sa ešte dostaneme pri zhodnotení našich experimentov v podkapitole 6.8 na strane 66. Použitie funkcie normalizácie softmax pri rátaní priemeru neprinieslo jednoznačný výsledok. Niekedy sa medián zlepšil, niekedy zasa zhoršil. Výsledky pre jednotlivé množiny sú pomerne vyrovnané s výnimkou `Currency`. Pre túto anomáliu nemáme vysvetlenie, ani pri hodnotení výkonu metódy v nasledujúcej podkapitole sme tento zvláštny jav nepozorovali.

Na grafoch na Obrázku 7 sme znázornili, aké rozloženie majú hodnotenia kandidátov pre algoritmus Avg_{euc}. Na grafe hore sme zaznačili ako so stúpajúcim miestom v poradí klesá hodnotenie. Môžeme vidieť, že pre všetky znázornené množiny má táto krivka podobný tvar. Líši sa však vo výške, v ktorej sa nachádza. Toto môže poukazovať na rozličnú hustotu priestoru v ktorom sa ukážkové páry nachádzajú. Takisto to môže znamenať že vnútrotriedová podobnosť jednotlivých množín je rozličná, čo taktiež ovplyvňuje výsledky. Vo všetkých prípadoch však na začiatku krivky vidíme akýsi chvostík, ktorý napovedá že existuje niekoľko párov, ktoré majú naozaj nadpriemerné hodnotenie.



Obr. 7: *Hore: Vývin skóre v zoradenej množine kandidátov. Znázornené je skóre vyrátane pre kandidáty vygenerované z našich 4 ukážkových množín pomocou PU učenia. Dole: Pravdepodobnosť že kandidát bude mať dané skóre. Vyrátané pre množinu* ***Capitals*** *s* PU_{euc} *algoritmom. Prerušovaná čiara v pravej časti grafu znázorňuje hodnotenie dosiahnuté mediánom pozícií.*

Rozloženie pravdepodobností pre jednu takúto krivku sme zakreslili do grafu dole. Môžeme vidieť, že sa jedná približne o normálne rozdelenie, pričom prerušovaná čiara v pravej polovici grafu znázorňuje akú hodnotu dosiahol medián pozícií.

6.5 Meranie presnosti metódy

Keďže obe predchádzajúce experimenty dopadli veľmi dobre a indikovali že naša metóda môže byť úspešná, ako ďalší experiment sme sa rozhodli overiť jej presnoť pri výsledkoch z vyrátaných z ukážkových množín. Priebeh tohto experimentu bol priamočiary: V súlade s návrhom sme vygenerovali kandidátov a ohodnotili sme ich. Potom sme z usporiadaného zoznamu zobrali prvých 100 najlepších kandidátov a manuálne sme vyhodnotili ich sémantickú správnosť. Každý kandidát sme ohodnotili ako správny, nesprávny, alebo v niektorých prípadoch čiastočne správny. Najlepšie výsledky pre každú množinu sú uvedené v Prílohe A

Presnosť metódy je potom pomer správnych – relevantných – párov. Za čiastočne správne sme vyhodnocovali dva druhy chýb:

- *Morfologická chyba* - Toto sú chyby v prípade keď slová nemali správny morfologický tvar. Môže sa jednať napríklad o plurál na mieste kde očakávame singulár (**shilling-shillings**) a podobne.

- *Underfitting* - Toto sú chyby, ktoré vyplývajú z nedostatočnej špecificity vzťahu tak, ako je opísaný pomocou vzorových párov. Tento fenomén sme rozoberali už pri návrhu našej metódy v podkapitole 5.2. Najčastejšie sem patril prípad, keď sme pri hľadaní hlavných miest objavili iné ako hlavné mestá, ktoré však boli priradené do správnej krajiny.

Okolie vektorov pri generovaní kandidátov sme definovali podobne ako v predošlom experimente ako 100 najpodobnejších slov. Použili sme naše dva algoritmy hodnotenia – priemernú podobnosť a PU učenie. Pri oboch algoritmoch sme použili dve miery podobnosti – kosínusovú i euklidovskú. Tieto prístupy budeme ďalej označovať PU_{cos}, PU_{euc}, Avg_{cos} a PU_{euc}. Nastavenia algoritmov zostali rovnaké ako v predošlom experimente. Výsledky z tohto experimentu sme zaznamenali do Tabuľky 5. Iný pohľad na tieto výsledky ponúkame v Tabuľke 6, kde sú výsledky zredukované metrikou *NDCG* na jedno výsledné číslo.

Tabuľka 5: *Vyhodnotenie prvých 100 výsledkov našej metódy pre rozličné ukážkové množiny a algoritmy hodnotenia. Pre každý algoritmus a množinu máme uvedené tri hodnoty:* R *pre správne odpovede,* P *pre čiastočne správne odpovede a* W *pre nesprávne odpovede.*

	PU_{euc}			Avg_{euc}		
	R	P	W	R	P	W
Cities	53	0	47	63	0	37
Currency	16	17	67	20	11	69
Capitals	24	20	56	23	19	58
Gender	9	8	83	9	6	85
	PU_{cos}			Avg_{cos}		
	R	P	W	R	P	W
Cities	11	0	89	21	0	79
Currency	13	1	86	15	2	83
Capitals	19	8	73	20	10	70
Gender	17	12	71	15	10	75

Tabuľka 6: *Výsledky metódy vyhodnotené metrikou NDCG. Pre túto metriku je nutné zadefinovať relevanciu každého výsledku. My sme pre správne páry použili relevanciu 1, pre nesprávne páry relevanciu 0 a pre čiastočne správne páry relevanciu 0,5.*

	Avg_{euc}	PU_{euc}	Avg_{cos}	PU_{cos}
Cities	**0,69**	0,60	0,25	0,15
Curency	**0,33**	0,27	0,18	0,12
Capitals	0,43	**0,44**	0,34	0,31
Gender	0,21	0,21	0,26	**0,29**

Najlepší výsledok sme dosiahli s algoritmom Avg_{euc} pre množiny `Cities`, ktorú sa nám podarilo rozšíriť o 63 nových párov. Tento algoritmus dosiahol náš najlepší výsledok a v priemere dokázal rozšíriť pôvodné množiny 25 párov až o 37,75 páru. Pre tri množiny `Cities`, `Capitals` a `Currency` sme dosiahli pomerne dobré a vyrovnané výsledky. Ako najlepšie algoritmy pre tieto množiny sa nám javili dva algoritmy používajúce euklidovskú podobnosť. Algoritmus PU_{cos} mal v týchto troch množinách najhoršie výsledky, exceloval však v jednej zostávajúcej.

Množina `Gender` si podľa očakávaní viedla najhoršie a naplnila tak očakávania z minulých experimentov. Pri analýze výsledkov sme si všimli, že väčšinq novoobjavených párov sú buď zámená (napr. **he-she**) alebo sa týkajú rodinných vzťahov (napr. **brother-sister**). Dôvody, prečo dosahuje táto množina také zlé výsledky sme už skúšali nájsť pri predošlých experimentoch. Z výsledkov v tomto experimente usudzujeme, že vzťah medzi mužskou a ženskou formou slov je skrátka priveľmi vágny sémantický vzťah. V skutočnosti sa zrejme jedná o celú triedu vzťahov, kde jedným členom triedy môže byť vzťah medzi mužskou a ženskou formou rodinných príslušníkov. Ďalšími členmi môžu zasa byť vzťahy medzi formami zamestnaní či pozícií. Táto hypotéza zodpovedá aj poznatkom z Fu et al. [2014].

Rozhodne pozoruhodným javom je, že si v tejto množine najlepšie počínal algoritmus PU_{cos}. Ten mal nielenže najviac relevantných párov, ale aj kvalita týchto párov bola nadpriemerná. Oproti zvyšným dvom prístupom našiel aj páry s inou ako rodinnou témou, napr. **bariton-soprano**.

Pri skúmaní rozdielov sme zisťovali aké prekryvy sú v zoznamoch 100 najlepších kandidátov pre jednotlivé algoritmy. Merali sme teda, koľko výsledkov zdieľaju algoritmy hodnotenia. Tieto merania sme zaznačili do Tabuľky 7. Hodnota v bunke značí percento spoločných párov pre prvých 100 výsledkov zo všetkých 4 množín. Maximum bola zhoda na 82,5% pre metirky PU_{euc} a Avg_{euc}. Táto hodnota však neberie do úvahy poradie výsledkov, to sa môže medzi porovnávanými výsledkami líšiť. Naznačuje však, že medzi týmito dvoma prístupmi nie je až taký veľký rozdiel.

Dôležitým poznatkom je, že podobnosť viac zależí od použitej miery podobnosti vektorov ako od použitého algoritmu. Zhody 76,5% a 82,5% medzi PU učením a priemernou podobnosťou pokiaľ používajú rovnakú mieru podobnosti napovedá, že tieto dva prístupy sú si veľmi podobné. Naopak veľké rozdiely medzi výsledkami rovnakého algoritmu pri použití inej miery podobnosti vektorov nám zasa hovoria, že jej výber je

Tabuľka 7: *Miera podobnosti (vyjadrená v percentách) medzi vygenerovanými výsledkami jednotlivých algoritmov. Číslo v bunke znamená, koľko percent párov sa objavilo vo výsledkoch oboch algoritmov hodnotenia.*

	PU_{euc}	Avg_{euc}	PU_{cos}	Avg_{euc}
PU_{euc}	-	**82,5**	35,25	37,25
Avg_{euc}	**82,5**	-	26,5	30,25
PU_{cos}	35,25	26,5	-	**76,5**
Avg_{cos}	37,25	30,25	**76,5**	-

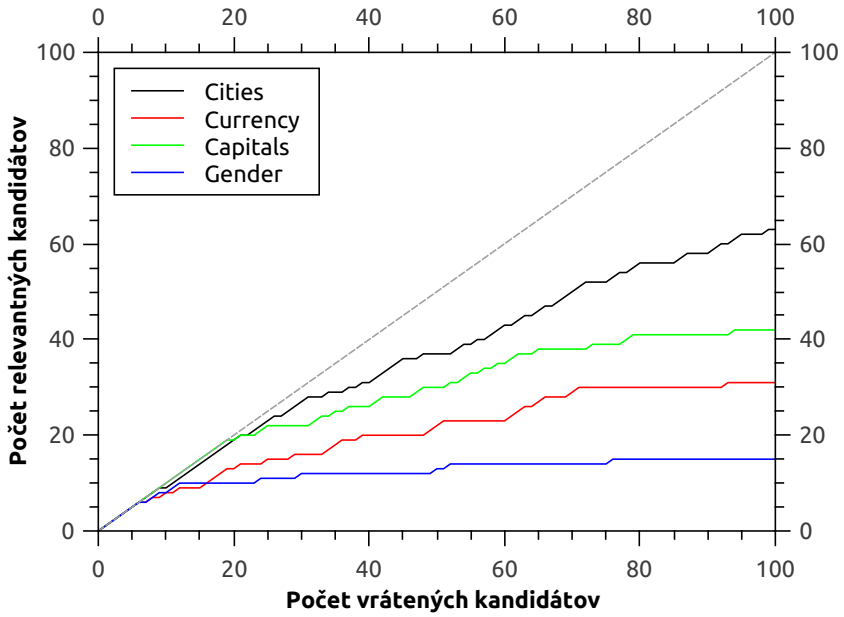
pre výsledok metódy kľučový.

Napokon sme do grafu na Obrázku 8 zaznamenali, ako stúpa počet nových relevantných párov ktoré sme objavili s tým, ako postupne vyhodnocujeme prvých 100 párov s algoritmom Avg_{euc}. Interpretácia bodu $[x, y]$ v grafe je taká, že v pri vyhodnotení prvých x kandidátov sme získali y relevantných párov. Za relevantné páry v tomto prípade považujeme tie, ktoré sme vyhodnotili ako správne alebo čiastočne správne. Na tomto grafe vidíme, že aj pre množinu `Gender` sme spočiatku dosahovali sľubné výsledky. Toto boli práve páry týkajúce sa rodinných vzťahov, ktoré sme ako jediné boli schopní odhaliť. Ďalej sme však už nové páry neobjavovali.

6.5.1 Porovnanie použitých algoritmov hodnotenia

Rozdiely vo výsledkoch medzi PU učením a priemernou podobnosťou sa na prvý pohľad veľmi nelíšia. V Tabuľke 7 môžeme vidieť že tieto výstup týchto dvoch metód sa zhoduje v priemere na okolo 80%. Aj úspešnosť týchto dvoch prístupov je približne rovnaká. Výhodou PU učenia je však to, že sa pri ňom ráta o polovicu menej podobností medzi vektormi. Polovica ukážkovej množiny je totižto použitá ako pozitívne vzorky a teda s týmito pármi sa kandidáty neporovnávajú. Je zaujímavé, že aj keď máme v podstate o polovicu menej informácií, výsledky sú natoľko podobné.

Nevýhodou PU učenia je naopak pamäťová náročnosť. Okrem samotných vektorov latentných treba pre natrénovanie klasifikátora vytvoriť trénovaciu množinu, ktorá má v našom prípade veľkosť $O(|C| \times |S|)$, kde C je množina kandidátov a S je ukážková množina. Samplovaním množiny kandidátov by sa však táto nevýhoda dala zrejme výrazne potlačiť. Rátanie priemernej podobnosti si vystačí len so samotnými vektormi latentných čŕt.



Obr. 8: *Vzťah medzi počtom k najlepších výsledkov a počtom relevantných kandidátov, ktorých sme dokázali nájsť. Použitý bol algoritmus Avg_{euc}.*

Pri porovnávaní mier podobnosti presvedčivo nad kosínusovou zvíťazila euklidovská podobnosť, ktorá dosiahla omnoho lepšie výsledky. Zaujímavý je fakt, že rozdiely medzi týmito podobnosťami vo výsledkoch neboli len kvantitatívne, ale aj kvalitatívne. Jedným z najvýraznejších rozdielov bol fakt, že pri kosínusovej podobnosti sme zaznamenali v prvej stovke omnoho viac opakovaní jedného slova. Rekordérom je ukrajinská mena **hryvnia**, ktorá bola pri algoritme PU_{cos} priradená až 41 krajinám zo 100. Priemerný počet, koľkokrát sa slovo opakovalo vo výsledkoch je pre jednotlivé algoritmy a ukážkové množiny zaznačený v Tabuľke 8.

6.6 Vplyv veľkosti ukážkovej množiny

Doteraz sme sa pokúsili overiť jednotlivé kroky našej metódy a potvrdili sme taktiež jej úspešnosť. Otázkou zostáva, či naozaj platí že väčšia množina párov dokáže definovať žiadaný sémantický vzťah lepšie, ako napríklad jeden ukážkový pár. Toto bolo jedno z

Tabuľka 8: *Priemerný počet opakovaní jedného slova v prvých 100 výsledkoch pre 4 použité algoritmy.*

	PU_{euc}	Avg_{euc}	PU_{cos}	Avg_{cos}
Cities	1.75	1.65	3.51	2.98
Currency	2.11	1.94	2.86	2.63
Capitals	1.47	1.50	2.17	2.11
Gender	2.56	2.60	2.35	2.53

východisiek našej práce. Pokiaľ by sme dokázali lepšie vzťahy odhaľovať s použitím iba jedného vzoru, celá naša metóda je bezcenná.

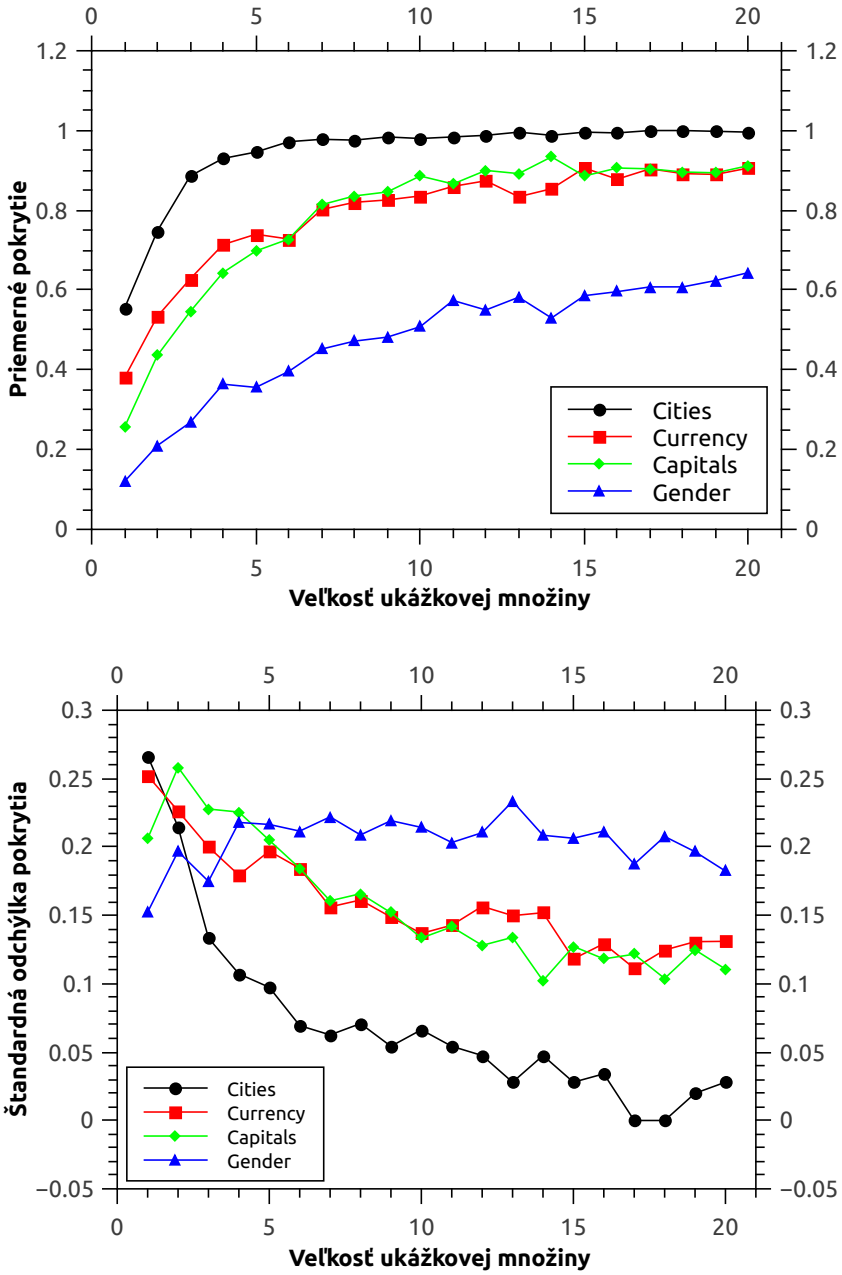
Na vyhodnotenie výsledkov pre rozličné veľkosti vstupnej množiny sme sa rozhodli použiť metriky z prvých dvoch experimentov – *pokrytie množiny* a *medián pozícií* – upravené pre naše aktuálne potreby. Tieto metriky nám umožnia zautomatizovať proces vyhodnocovania, keďže pri nich nemusíme pracne manuálne hodnotiť novonájdené páry. Pri tomto experimente sa pokúsime vyrátať tieto metriky pre množiny rozličných veľkostí, vytvorených výberom náhodných n párov z našich pôvodných ukážkových množín.

6.6.1 Pokrytie množiny

Začneme s mierou pokrytia, ktorá nám hovorí, koľko relevantných párov môžeme v zozname kandidátov očakávať. Pôvodnú množinu s 25 pármi si rozdelíme na trénovaciu množinu veľkosti n a testovaciu množinu o veľkosti 5. Zvyšné páry zahodíme. Veľkosť n je práve tá veličina, ktorej vplyv sledujeme a naberá hodnoty z intervalu $\langle 1, 20 \rangle$. Z trénovacej množiny si v súlade s návrhom metódy necháme vygenerovať kandidáty. Pokrytie je potom pomer takých párov z testovacej množiny, ktoré sa v množine kandidátov nachádzaju voči celkovej veľkosti testovacej množiny. Tento postup zachytáva nasledovný pseudokód:

```
# Parameter 'set' predstavuje ukazkovu mnozinu
# Parameter 'n' je skumana velicina
def set_recall_for_n_pairs(set, n)
  testing_set = set.sample(5)
  set -= testing_set
  training_set = set.sample(n)
  candidates = training_set.generate_candidates
  included_pairs =
  testing_set.select do |pair|
    candidates.include?(pair)
  end
  return included_pairs.size / testing_set.size
end
```

Výsledok pre jednu veľkosť veličiny n sme vyrátali ako priemer zo 100 opakovaní tohto experimentu. Páry do jednotlivých množín sme pritom vyberali náhodne. Výsledky tohto experimentu sme zaznačili do grafov pre jednotlivé ukážkové mmnožiny na Obrázku 9. Môžeme vidieť, že s rastúcou veľkosťou veličiny n rastie aj pokrytie. Toto je očakávaný výsledok, keďže čím viac máme párov, tým viac kandidátov sa vygeneruje. Otázkou bolo skôr ako rýchlo tento rast prebieha.



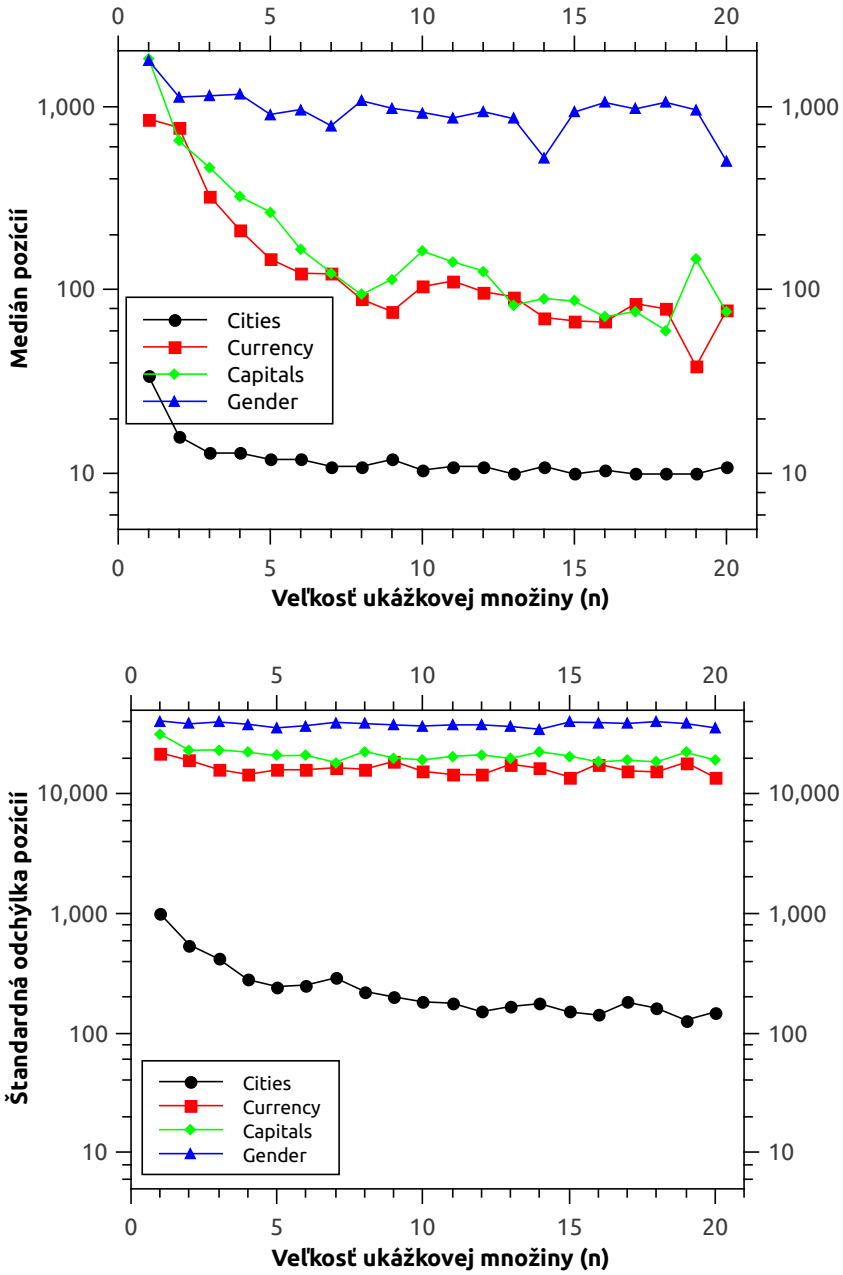
Obr. 9: *Hore: Vplyv veľkosti ukážkovej množiny na jej pokrytie. Hodnota pre jednotlivé veľkosti je vyrátaná ako priemer zo 100 opakovaní experimentu s algoritmom hodnotenia* Avg_{euc}*. Dole: Štandardná odchýlka z tých istých dát.*

Môžeme si taktiež všimnúť, že pokrytie rozličných množin stúpalo rozličnou rýchlosťou. Myslíme si, že tie množiny ktoré stúpajú rýchlo zrejme vo vektorovom priestore vytvárajú kompaktnejší zhluk kde všetky páry sú blízko pri sebe. Iné množiny sú viac roztrúsené, napr. pri množine `Capitals` zrejme geograficky príbuzné krajiny vytvárajú svoje osobité zhluky (napr. zhluk škandinávskych krajín, východo-ázijských krajín, atď.). Množina `Cities` jasne dominuje zatiaľ čo množina `Gender` je zasa najslabšia. Zvyšné dve množiny sa zasa držia blízko pri sebe. Tento trend môžeme vidieť pri priemere aj štandardnej odchýlke.

6.6.2 Medián pozícií

Dôležitejšou metrikou na zhodnotenie je medián pozícií. Ten sa ráta z hodnotenia kandidátov, čo je tou dôležitejšou časťou našej metódy. Aj tu sme mierne upravili spôsob rátania metriky oproti experimentu opísanému v podkapitole 6.4. Tentokrát sa mení len spôsob vytvárania trénovacej množiny. Tá sa v pôvodnom opísanom algoritme vyberie ako 80% z celej množiny. V tomto algoritme tento výber ešte ďalej okrešeme na n náhodných párov, kde n je opäť z intervalu $\langle 1, 20 \rangle$. Inak sa algoritmus rátania nemení. Takýto experiment sme opakovali pre každé n 100 krát a výsledok sme vyrátali ako medián vrátených pozícií. Tým, že množina kandidátov je zakaždým rovnaká, môžeme priamo porovnávať ako sa menia pozície testovacích vzoriek.

Na grafoch na Obrázku 10 môžeme vidieť, že pre 3 množiny s lepšími výsledkami sa medián pozícií výrazne zlepšuje so stúpajúcim počtom ukážkových párov. Znamená to teda, že čím viac párov máme, tým lepšie dokážeme ohodnotiť kandidáty. Tento pokles, aj keď v omnoho menšej miere sme zaznamenali aj pre ostatnú množinu `Gender`. Tým sme ukázali, že náš prepoklad je správny a použitie väčšieho počtu ukážkových párov je dobrým prístupom k definovaniu sémantického vzťahu. Ukázalo sa však aj to, že v zle definovanej množine to veľmi nepomôže. Opäť môžme na týchto grafoch vidieť výrazne rozdiely medzi jednotlivými množinami, ktoré sa preukázali v prakticky každom našom experimente.



Obr. 10: *Hore: Vplyv veľkosti ukážkovej množiny na medián pozícií. Hodnota pre jednotlivé veľkosti je vyrátaná ako medián zo 100 opakovaní experimentu s algoritmom hodnotenia* Avg_{euc}. *Dole: Štandardná odchýlka vyrátaná z tých istých dát. Na grafoch je použitá logaritmická škála.*

Tabuľka 9: *Výsledky metódy vyhodnotené metrikou NDCG pre zvyšné množiny. Pre túto metriku je nutné zadefinovať relevanciu každého výsledku. My sme pre správne páry použili relevanciu 1, pre nesprávne páry relevanciu 0 a pre čiastočne správne páry relevanciu 0,5.*

	Avg_{euc}	Avg_{cos}
Comparative	0,39	0,52
Knowledge	0,00	0,02
Nationality	0,60	0,46
Part	0,00	0,10
Past Tense	0,87	0,80

6.7 Výsledky metódy pre zvyšné množiny

Do Tabuľky 9 sme zaznačili aj výsledky pre naše zvyšné množiny, ktoré sme opísali v podkapitole 6.1.2 na strane 6.1.2. Pre tieto množiny sme rátali len pomocou algoritmov Avg_{cos} a Avg_{euc}. Výsledky týchto metrík sme zredukovali iba na NDCG skóre. Môžeme vidieť, že pre 2 množiny – `Knowledge` a `Part` – sme dosiahli veľmi slabé výsledky. Pri množine `Knowledge` je to podľa nás spomenuté subjektívnou formuláciou párov. V rrípade množiny `Past` zasa zrejme ide o príliš všeobecný vzťah. My síce vnímame sémantickú podobnosť medzi dvojicami ako sú napr. **forest-trees** a **body-cells**, táto podobnosť sa podľa nás však len veľmi slabo premieta aj do textového korpusu. Tieto dvojice sú priveľmi doménovo rozdielne na to, aby sa vety v ktorých sa vyskytujú podobali.

Naopak zvyšné množiny, založené na gramatických vzťahoch, dosahovali vynikajúce výsledky. Zrejme sa takéto dvojice veľmi dobre otláčajú do vektorového priestoru. Môžeme konštatovať, že presnosť našej metódy je veľmi závislá od vstupnej množiny a nedosahuje rovnomernú úroveň. Taktiež si môžeme všimnúť, že sa v týchto množinách nepotvrdila prevaha euklidovskej podobnosti, ked v 3 z 5 prípadoch prevážila naopak tá kosínusová.

6.8 Vyhodnotenie experimentov

Ukázali sme, že nami navrhnutá metóda dokáže úspešne nachádzať nové páry so zadefinovaným sémantickým vzťahom. Úspešnosť našej metódy však kolíše v závislosti od toho, ako veľmi sa otlačil daný sémantický vzťah do priestoru vektorov latentných čŕt. To, čo presne spôsobuje že niektoré vzťah sa otlačí dobre a iný nie je námetom na ďalšie

skúmanie. Otázne je aj to, či rozličné druhy vzťahov nevyžadujú rozličné algoritmy rátania. V našich dátach sa napríklad ako anomália javí množina `Gender`, ktorá dosiahla najlepší výsledok s úplne iným algoritmom ako ostatné množiny.

Pravdepodobne však pôjde o závislosť na pôvodnom textovom korpuse a intenzite s akou sa v ňom daný vzťah odráža. Najlepší výsledok sme dosiahli pre množinu `Cities`, ktorá zachytáva vzťah medzi americkými mestami a štátmi, ktoré v týchto mestách ležia. Dominancia tejto množiny je zrejme spôsobená americkým zvykom uvádzať v texte za názvom mesta automaticky aj štát v ktorom sa nachádza, napr. „Tampa, Florida“.

Prekvapivým výsledkom pre nás bola pomerne veľká podobnosť výsledkov PU učenia a rátania jednoduchej priemernej podobnosti. Dúfali sme, že pomocou strojového učenia dokážeme vo vzoroch vo vektorovom priestore odhaliť nové súvislosti. Problémom tohto prístupu ale môže byť na pomery strojového učenia malá množina pozitívnych vzoriek použitá pri trénovaní. Prekvapil nás aj veľký rozdiel medzi euklidovskou a kosínusovou podobnosťou. V prácach, kde sa podobne ako v našej skúmajú vzory, ktoré sémantické vzťahy nechávajú vo vektorovom priestore sa doteraz používala takmer výlučne iba tá kosínusová.

Tento fakt môže pramenit' z toho, že vektory slov sú pri spracovaní korpusu spravidla normalizované. Ak sa teda porovnávajú iba jednoduché normalizované vektory, kosínusová a euklidovská podobnosti sú navzájom izomorfné. Kosínusová podobnosť je však pri normalizovaných vektoroch časovo výhodnejšia. Vektorové rozdiely vyrátané z dvojíc však normalizované nie sú a úvaha o výhodách kosínusovej podobnosti tu tak už neplatí. To, že euklidovská podobnosť v niektorých prípadoch dosahovala lepšie výsledky zrejme znamená, že pri vektoroch vzťahov je dôležitá aj dĺžka, nielen smer takýchto vektorov. Ukázalo sa, že pre niektoré množiny je lepšia podobnosť euklidovská, pre iná zasa kosínusová. Kombinácia oboch týchto prístupov by možno mohla dosiahnuť ešte lepšie výsledky.

Dôležitým aspektom našej práce bolo aj skúmanie vlastností množín párov s rovnakým sémantickým vzťahom. Toto skúmanie sa týkalo najmä prvých dvoch experimentov. Navrhli sme dve metriky, pokrývajúce dva hlavné kroky našej metódy, ktoré nám môžu pomôcť automaticky vyhodnocovať potenciál množín párov. Takéto hodnotenie sa tiež javí ako perspektívny smer ďalšieho skúmania. Na záver sme s úspechom overili náš predpoklad, že s použitím viacerých ukážkových párov dokážeme vylepšovať presnosť hodnotenia a tým aj celkový výkon našej metódy.

7 Záver

Medzi úlohy z oblasti spracovania prirodzeného jazyka patrí aj objavovanie sémantickych vzťahov medzi entitami jazyka. Hľadanie takýchto vzťahov medzi slovami, frázami a podobne v textových korpusoch je netriviálnym problémom. Jedným zo základných prúdov výskumu v tejto oblasti je využitie štatistickej analýzy jazyka. Objavovanie takýchto sémantických znalostí je prínosné hlavne pre znalostné inžinierstvo, využiť sa však dajú aj v iných odboroch a úlohách.

V našej práci sme sa venovali najmä modelom prirodzeného jazyka vytvorených pomocou neurónových sietí. Takéto techniky vytvárajú pre termy z jazyka tzv. vektorý latentných čŕt a vychádzajú z Harrisovej distribučnej hypotézy. Opierajú sa teda o predpoklad, že znalosti zakódované v textoch napísaných v prirodzenom jazyku sa dajú odhaliť pomocou štatistiky. V práci sme opísali vlastnosti vektorového modelu, ktorý takto vieme neurónovými sieťami zostrojiť.

Zanalyzovali sme vlastnosti takýchto vektorov a priblížili sme viacero prístupov, ako pomocou nich dokážeme pracovať so sémantickými vzťahmi. Uviedli sme viacero prác, ktoré sa práve takémuto využitiu venujú. Tieto práce sa venovali úlohám ako odhaľovanie analógií či zostrojovanie taxonómie konceptov. Okrem toho sa však vektory latentných čŕt používajú aj pri mnohých iných úlohách spracovania prirodzeného jazyka. Sú teda overeným a veľmi efektívnym prostriedkom ako riešiť problémy v tomto odbore.

Zaujímavé výsledky dosiahnuté pomocou takéhoto modelu nás priviedli k myšlienke použiť ho pri objavovaní nových inštancií sémantických vzťahov medzi termami v textovom korpuse. Na základe vstupnej množiny ukážkových párov termov, ktoré medzi sebou hľadaný vzťah majú, sa snažíme v korpuse objaviť nové inštancie toho istého vzťahu. Zo vstupu **Paríž-Francúzsko, Rím-Taliansko, Berlín-Nemecko** chceme dostať ďalšie páry slov, s rovnakým vzťahom, napr. **Moskva-Rusko**. Takýto jednoduchý spôsob definovania vzťahu je podľa nás veľmi dôležitá výhoda nášho prístupu.

Za účelom takéhoto objavovania inštancií vzťahov sme navrhli vlastnú metódu. Táto metóda je navrhnutá tak, aby dokázala štatisticky spracovať textový korpus a odhaliť v ňom vzory, ktoré charakterizujú určitý sémantický vzťah. Na základe takýchto objavených vzorov následne dokáže objavovať nové vzťahy s hľadaným vzťahom. Pri štatistickom spracovaní jazyka na vektory latentných čŕt vychádzame z práce iných

autorov. Naším hlavným záujmom je objavovanie vzorov v týchto vektoroch a aplikácie vzorov pri objavovaní nových znalostí.

Z návrhu našej metódy vyplynulo niekoľko výskumných otázok. Sériou experimentov sme sa ich snažili zodpovedať, pričom sme preskúmali vlastnosti a výkon našej metódy. Našu metódu sme testovali najmä na niekoľkých sémantických vzťahoch s použitím textového korpusu *Google News*. Overili sme, že naša metóda dokáže nájsť pre rozličné ukážkové množiny desiatky nových dvojíc. Musíme však konštatovať, že výkon našej metódy značne kolíše v závislosti aj od vlastností hľadaného sémantického vzťahu. Naším najlepším výsledkom bolo, keď sme rozšírili pôvodnú množinu 25 párov amerických miest a štátov, v ktorých sa nachádzajú, o 62 nových párov pri spracovaní 100 najlepších výsledkov.

Výsledkom týchto experimentov je aj bližšie pochopenie toho, ako jednotlivé faktory vplývajú na našu schopnosť extrahovať z takýchto priestorov sémantickú informáciu o vzťahoch. Taktiež sme získali základné pochopenie toho, ako sa rozličné triedy vzťahov v takomto priestore správajú.

Existuje viacero možností ďalšieho rozšírenia tejto práce. Extrakcia vzťahov sa napríklad okrem prirodzených jazykov používa aj v bioinformatike pri výskume správania proteínov. Naša metóda by sa možno dala aplikovať aj v tejto oblasti. Určite zaujímavým smerom by mohlo byť dôkladnejšie preskúmanie vlastností vzťahov s cieľom zistiť, čo robí daný vzťah vhodný pre použitie našej metódy. Takéto skúmanie by mohlo prebiehať jednak v rovine matematickej na úrovni vektorov latentných čŕt, ale aj v rovine lingvistickej.

Literatúra

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 23–32. Association for Computational Linguistics, 2012.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL (1)*, pages 238–247, 2014.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.

Matthew Berland and Eugene Charniak. Finding parts in very large corpora. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 57–64. Association for Computational Linguistics, 1999.

Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001.

Paul Buitelaar, Philipp Cimiano, and Bernardo Magnini. *Ontology learning from text: methods, evaluation and applications*, volume 123. IOS press, 2005.

Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990.

Philipp Cimiano. *Ontology learning from text*. Springer, 2006.

Philipp Cimiano and Johanna Völker. Text2onto. In *Natural language processing and information systems*, pages 227–238. Springer, 2005.

Philipp Cimiano, Andreas Hotho, and Steffen Staab. Learning concept hierarchies from text corpora using formal concept analysis. *J. Artif. Intell. Res.(JAIR)*, 24:305–339, 2005.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

Erika F De Lima and Jan O Pedersen. Phrase recognition and expansion for short, precision-biased queries based on a query log. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 145–152. ACM, 1999.

Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *JAsIs*, 41(6):391–407, 1990.

Li Ding, Pranam Kolari, Zhongli Ding, and Sasikanth Avancha. Using ontologies in the semantic web: A survey. In *Ontologies*, pages 79–113. Springer, 2007.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial intelligence*, 165(1):91–134, 2005.

Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems*, pages 2121–2129, 2013.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. Learning semantic hierarchies via word embeddings. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics: Long Papers*, volume 1, 2014.

George W Furnas, Thomas K Landauer, Louis M Gomez, and Susan T Dumais. Human factors and behavioral science: Statistical semantics: Analysis of the potential performance of key-word information systems. *Bell System Technical Journal, The*, 62(6): 1753–1806, 1983.

Dedre Gentner. Structure-mapping: A theoretical framework for analogy. *Cognitive science*, 7(2):155–170, 1983.

Gene H Golub and Christian Reinsch. Singular value decomposition and least squares solutions. *Numerische mathematik*, 14(5):403–420, 1970.

Asunción Gómez-Pérez, David Manzano-Macho, et al. A survey of ontology learning methods and techniques. *OntoWeb Deliverable D*, 1(5), 2003.

Thomas R Gruber. Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, 43(5):907–928, 1995.

Zellig S Harris. Distributional structure. *Word*, 1954.

Maryam Hazman, Samhaa R El-Beltagy, and Ahmed Rafea. A survey of ontology learning approaches. *database*, 7:6, 2011.

Itziar Irigoien, Basilio Sierra, and Concepción Arenas. Towards application of one-class classification methods to medical data. *The Scientific World Journal*, 2014, 2014.

Thorsten Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM, 2006.

David A Jurgens, Peter D Turney, Saif M Mohammad, and Keith J Holyoak. Semeval-2012 task 2: Measuring degrees of relational similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 356–364. Association for Computational Linguistics, 2012.

Martin Kavalec, Alexander Maedche, and Vojtěch Svátek. Discovery of lexical entries for non-taxonomic relations in ontology learning. In *SOFSEM 2004: Theory and Practice of Computer Science*, pages 249–256. Springer, 2004.

Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 302–308, 2014.

Omer Levy, Yoav Goldberg, and Israel Ramat-Gan. Linguistic regularities in sparse and explicit word representations. *CoNLL-2014*, page 171, 2014.

Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015a.

Omer Levy, Steffen Remus, Chris Biemann, Ido Dagan, and Israel Ramat-Gan. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics â AS Human Language Technologies (NAACL HLT 2015), Denver, CO*, 2015b.

Xiaoli Li and Bing Liu. Learning to classify texts using positive and unlabeled data. In *IJCAI*, volume 3, pages 587–592, 2003.

Dekang Lin and Patrick Pantel. Dirt@ sbt@ discovery of inference rules from text. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 323–328. ACM, 2001.

Minh-Thang Luong, Richard Socher, and Christopher D Manning. Better word representations with recursive neural networks for morphology. *CoNLL-2013*, 104, 2013.

Alexander Maedche. *Ontology learning for the semantic web*. Springer Science & Business Media, 2002.

Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751, 2013b.

George A Miller and Walter G Charles. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28, 1991.

Andriy Mnih and Geoffrey E Hinton. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088, 2009.

Preslav Nakov and Marti A Hearst. Solving relational similarity problems using the web as a corpus. In *ACL*, pages 452–460, 2008.

Ján Paralič, Karol Furdík, Gabriel Tutoky, Peter Bednár, Martin Sarnovskỳ, Peter Butka, and František Babič. Dolovanie znalostí z textov. *Equilibria, Košice*, 2010.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

Massimo Poesio and Abdulrahman Almuhareb. Identifying concept attributes using a classifier. In *Proceedings of the ACL-SIGLEX Workshop on Deep Lexical Acquisition*, pages 18–27. Association for Computational Linguistics, 2005.

Maria Ruiz-Casado, Enrique Alfonseca, and Pablo Castells. Automatic extraction of semantic relationships for wordnet by means of pattern learning from wikipedia. In *Natural Language Processing and Information Systems*, pages 67–79. Springer, 2005.

David Sánchez and Antonio Moreno. Learning non-taxonomic relationships from web documents for domain ontology construction. *Data & Knowledge Engineering*, 64(3): 600–623, 2008.

Alexander Schutz and Paul Buitelaar. Relext: A tool for relation extraction from text in ontology extension. In *The Semantic Web–ISWC 2005*, pages 593–606. Springer, 2005.

Holger Schwenk. Continuous space language models. *Computer Speech & Language*, 21(3):492–518, 2007.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer, 2013.

Fabian M Suchanek, Georgiana Ifrim, and Gerhard Weikum. Combining linguistic and statistical analysis to extract relations from web documents. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 712–717. ACM, 2006.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

Peter Turney, Michael L Littman, Jeffrey Bigham, and Victor Shnayder. Combining independent modules to solve multiple-choice synonym and analogy problems. 2003.

Peter D Turney. Similarity of semantic relations. *Computational Linguistics*, 32(3): 379–416, 2006.

Peter D Turney, Patrick Pantel, et al. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188, 2010.

Vishnu Vyas and Patrick Pantel. Semi-automatic entity set refinement. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 290–298. Association for Computational Linguistics, 2009.

Sholom M Weiss, Nitin Indurkhya, Tong Zhang, and Fred Damerau. *Text mining: predictive methods for analyzing unstructured information*. Springer Science & Business Media, 2010.

Wilson Wong, Wei Liu, and Mohammed Bennamoun. Ontology learning from text: A look back and into the future. *ACM Computing Surveys (CSUR)*, 44(4):20, 2012.

Wilson Yiksen Wong. *Learning lightweight ontologies from text across different domains using the web as background knowledge*. University of Western Australia, 2009.

Fei Wu and Daniel S Weld. Autonomously semantifying wikipedia. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 41–50. ACM, 2007.

Ichiro Yamada and Timothy Baldwin. Automatic discovery of telic and agentive roles from corpus data. In *Proceedings of the The 18th Pacific Asia Conference on Language, Information and Computation (PACLIC 18)*. Citeseer, 2004.

Alisa Zhila, Wen-tau Yih, Christopher Meek, Geoffrey Zweig, and Tomas Mikolov. Combining heterogeneous models for measuring relational similarity. In *HLT-NAACL*, pages 1000–1009, 2013.

Lina Zhou. Ontology learning: state of the art and open issues. *Information Technology and Management*, 8(3):241–252, 2007.

Will Y Zou, Richard Socher, Daniel M Cer, and Christopher D Manning. Bilingual word embeddings for phrase-based machine translation. In *EMNLP*, pages 1393–1398, 2013.

A Množiny ukážkových vzťahov

Pri experimentovaní sme používali nasledovné množiny ukážkových vzťahov. Pri prvých štyroch sme pre úplnosť uviedli aj najlepší dosiahnutý výsledok – 100 najlepšie ohodnotených kandidátov. V týchto výsledkoch sú hrubým písmom zvýraznené páry, ktoré sme označili za správne. Kurzívou sú označené páry, ktoré sme označili za čiastočne správne. Pri každom výsledku uvádzame aj algoritmus, ktorým sme k nemu dospeli. Vysvetlenie týchto algoritmov sa nachádza v podkapitole 6.4 na strane 50.

A.1 Cities

A.1.1 Ukážková množina

Chicago Illinois, Houston Texas, Philadelphia Pennsylvania, Phoenix Arizona, Dallas Texas, Jacksonville Florida, Indianapolis Indiana, Austin Texas, Detroit Michigan, Memphis Tennessee, Boston Massachusetts, Seattle Washington, Denver Colorado, Baltimore Maryland, Nashville Tennessee, Louisville Kentucky, Milwaukee Wisconsin, Portland Oregon, Tucson Arizona, Fresno California, Sacramento California, Mesa Arizona, Atlanta Georgia, Omaha Nebraska, Miami Florida

A.1.2 Najlepšie výsledky

Výsledky dosiahnuté s použitím algoritmu Avg_{euc}:
Tampa Florida, **Honolulu Hawaii**, **Cleveland Ohio**, **Winnipeg Manitoba**, **Billings Montana**, **Reno Nevada**, **Orlando Florida**, **Minneapolis Minnesota**, **Wichita Kansas**, Seattle Oregon, **Knoxville Tennessee**, **Charlottesville Virginia**, **Edmonton Alberta**, **Cincinnati Ohio**, **Hartford Connecticut**, **Lansing Michigan**, **Chattanooga Tennessee**, **Peoria Illinois**, **Laramie Wyoming**, **Toronto Ontario**, **Toledo Ohio**, Memphis Arkansas, **Bridgeport Connecticut**, **Harrisburg Pennsylvania**, **Portland Maine**, **Racine Wisconsin**, Memphis Alabama, **Pittsburgh Pennsylvania**, **Richmond Virginia**, **Fairbanks Alaska**, **Scranton Pennsylvania**, Denver Wyoming, Winnipeg Alberta, **Waltham Massachusetts**, Edmonton Manitoba, Memphis Kentucky, **Columbus Ohio**, Wichita Missouri, **Stamford Connecticut**, Winnipeg Saskatchewan, **Cheyenne Wyoming**, **Lexington Kentucky**, **LA California**, **Casper Wyoming**, **Lubbock Texas**, Jack-

sonville Alabama, Denver Utah, **Madison Wisconsin**, Jacksonville Tennessee, Dallas Austin, Philadelphia Delaware, Toronto Ottawa, **Fayetteville Arkansas**, **Savannah Georgia**, Dallas Houston, **Birmingham Alabama**, Memphis Mississippi, **Lowell Massachusetts**, **Montpelier Vermont**, **Champaign Illinois**, Minnesota Wisconsin, **Roanoke Virginia**, **Topeka Kansas**, Waltham Woburn, **Springfield Illinois**, **Tempe Arizona**, Edmonton Saskatchewan, **Herndon Virginia**, **Appleton Wisconsin**, **Alameda California**, **Macon Georgia**, **Ottawa Ontario**, Billings Wyoming, Atlanta Alabama, Cincinnati Kentucky, **Vancouver BC**, **Jonesboro Arkansas**, Chandler Brewer, **Bangor Maine**, **Wilmington Delaware**, Boston Connecticut, Chicago Wisconsin, Atlanta Tennessee, Stamford Greenwich, Sacramento Fresno, **Louisville Tennessee**, **Dayton Ohio**, Oklahoma Kansas, Atlanta Florida, Dallas Denton, **Charleston Carolina**, **Covington Kentucky**, Chicago Michigan, **Gresham Oregon**, **Waterbury Connecticut**, Toledo Michigan, Milwaukee Minnesota, Denver Montana, **Williamsburg Virginia**, Maui Hawaii

A.2 Currency

A.2.1 Ukážková množina

Algeria dinar, Armenia dram, Brazil real, Bulgaria lev, Cambodia riel, Croatia kuna, Denmark krone, Europe euro, Hungary forint, India rupee, Iran rial, Japan yen, Korea won, Lithuania litas, Malaysia ringgit, Mexico peso, Nigeria naira, Poland zloty, Romania leu, Russia ruble, Sweden krona, Thailand baht, Ukraine hryvnia, USA dollar, Vietnam dong

A.2.2 Najlepšie výsledky

Výsledky dosiahnuté s použitím algoritmu Avg_{euc}:
U.S. dollar, *India rupees*, **China Yuan**, *European euro*, **Norway krone**, **Germany euro**, Poland forint, **Turkey lira**, Russia's denar, *Russia rubles*, Russia's dollar, **Angola kwanza**, Osmanli denar, country dollar, Sweden krone, *Kenya shillings*, **Canada dollar**, **Australia dollar**, **Philippines peso**, Hungary zloty, *Bulgaria leva*, USA denar, Romania forint, Benin naira, **Israel shekel**, Ukraine ruble, Romania zloty, Roumania denar, **world dollar**, Singapore ringgit, Hungary leu, Frisians denar, France livres, *Tanzania shillings*,

America dollar, **France euro**, Rumanians denar, Pologne denar, **Italy euro**, Russia's florin, Germany German, Publikum denar, Europe dollar, Brazil kwanza, India dollar, Indonesia ringgit, Hasdrubal denar, Bulgaria Bulgarian, *India Rs*, **Japan Yen**, *Thai baht*, Egypt Egyptian, Romania euro, Germany denar, Colombia Colombian, Ghana naira, Diemen denar, Diemen vei, Women's florin, Germany forint, **Portugal euro**, **Spain euro**, *Brazil reals*, Spain denar, *Uganda shillings*, **Belgium euro**, overseas dollar, Brazil dollar, Belarus hryvnia, *Hungarian forint*, **Moldova leu**, Frisians dollar, Razumov denar, China dollar, Tientsin dollar, U.S.A. denar, Moldova hryvnia, Hungary euro, Russia hryvnia, Switzerland Swiss, Women's denar, international denar, Russia's milliards, Russia Russian, Belgium denar, Japan dollar, global dollar, Osmanli gaun, Norway Norwegian, Romania denar, Osmanli agi, U.S. denar, **Belarus ruble**, Rumanians fuertes, country denar, Roumania mesi, Italy denar, Moscow ruble, Thailand Thai, France groupes,

A.3 Capitals

A.3.1 Ukážková množina

Athens Greece, Baghdad Iraq, Bangkok Thailand, Beijing China, Berlin Germany, Bern Switzerland, Cairo Egypt, Canberra Australia, Hanoi Vietnam, Havana Cuba, Helsinki Finland, Islamabad Pakistan, Kabul Afghanistan, London England, Madrid Spain, Moscow Russia, Oslo Norway, Ottawa Canada, Paris France, Rome Italy, Stockholm Sweden, Tehran Iran, Tokyo Japan, Budapest Hungary, Caracas Venezuela

A.3.2 Najlepšie výsledky

Výsledky dosiahnuté s použitím algoritmu PU_{euc}:
Kiev Ukraine, *Sydney Australia*, **Manila Philippines**, **Warsaw Poland**, **Dublin Ireland**, **London Britain**, **Damascus Syria**, **London UK**, *Shanghai China*, *Milan Italy*, **Seoul Korea**, **Honolulu Hawaii**, *Melbourne Australia*, **Teheran Iran**, **Sofia Bulgaria**, **Belgrade Serbia**, **Khartoum Sudan**, **Delhi India**, **Amsterdam Netherlands**, **Tunis Tunisia**, Moscow Ukraine, Prague Poland, *Munich Germany*, Budapest Romania, Stockholm Norway, *Frankfurt Germany*, *Dusseldorf Germany*, *Brisbane Australia*, Soldaten Deutschlands, **Brussels EU**, **Edinburgh Scotland**, Sceaux Cayrol, *Tokyo Japanese*, Singapore Malaysia, Sceaux Bologne, *Glasgow Scotland*, **Bogota Colombia**, Babylone

Cayrol, **Vienna Austria**, Stockholm Finland, *Moscow Russian*, Malesherbes Cayrol, Turm Deutschlands, Sammlung Deutschlands, Helsinki Sweden, *Cologne Germany*, Paris Belgium, Parijs Cayrol, *Barcelona Spain*, *Manila Philippine*, *Montreal Canada*, Anlage Deutschlands, **Quito Ecuador**, Babylone Bologne, Budapest Poland, Stockholm Denmark, *Hamburg Germany*, *Melun France*, German Germany, Caracas Bolivia, Bulgarian Bulgaria, Malesherbes Bologne, **Riga Latvia**, Polignac Cayrol, Turm Vaterland, Moscow Belarus, Choisy Bologne, Quijote Castilians, **Winnipeg Manitoba**, *Seville Spain*, Montalembert Cayrol, Luis Jorge, Neuburg Vaterland, Conciergerie Bologne, Melbourne Adelaide, Norwegian Norway, Conciergerie Cayrol, Parijs Bologne, *Beijing Chinese*, Budapest Austria, Brazilian Brazil, **Lisbon Portugal**, Soldaten Vaterland, Nikolaevna Russia's, Neuburg Deutschlands, Egyptian Egypt, Choisy Cayrol, Parisiens Bologne, Julio Luis, Deutschen Deutschlands, Miguel Luis, Jorge Luis, Parisiens Cayrol, Nikolaevna Razumov, Chilean Chile, Maslova Razumov, Israeli Israel, Brisbane Adelaide, Austrian Austria, *Zurich Switzerland*,

A.4 Gender

A.4.1 Ukážková množina

boy girl, brother sister, father mother, grandpa grandma, groom bride, he she, husband wife, king queen, man woman, nephew niece, policeman policewoman, prince princess, actor actress, bachelor spinster, count countess, czar czarina, emperor empress, god goddess, heir heiress, hero heroine, host hostess, master mistress, sir madam, wizard witch, waiter waitress

A.4.2 Najlepšie výsledky

Výsledky dosiahnuté s použitím algoritmu PU_{cos}:
himself herself, **son daughter**, *his her*, *boys girls*, *brothers sisters*, **He She**, **grandson granddaughter**, **Uncle Aunt**, father daughter, son mother, brother daughter, nephew daughter, **uncle aunt**, **schoolboy schoolgirl**, brother mother, uncle mother, uncle niece, *sons daughters*, he her, *him her*, brother niece, **Uncle Auntie**, nephew mother, his herself, **boyhood girlhood**, *him she*, **Brother Sister**, himself her, uncle daughter, son niece, man girl, him herself, brothers sister, he herself, brothers mother, *men women*, grandson

daughter, nephew granddaughter, **baritone soprano**, grandson niece, demigod heroine, himself she, *his hers*, father niece, **grandfather grandmother**, **guy gal**, Uncle waitress, sons daughter, himself actress, uncle grandmother, demigod vixen, brother aunt, brothers daughter, **lad lass**, he actress, uncle sister, schoolboys schoolgirl, patriarch actress, nephew sister, Uncle housewife, his actress, he hers, villanous heroine, **Uncle Aunty**, villanous actress, brother girl, **Englishman Englishwoman**, Father Sister, nephew aunt, charioteer heroine, villanous vixen, **dad mother**, genius minx, countryman actress, **Man Woman**, wizard vixen, boyhood actress, himself woman, brothers girl, Grandpa Aunt, Brother Aunt, grandfather mother, *heroes heroines*, father sister, Uncle Grandma, wizard minx, godfather actress, Uncle mother, *actors actresses*, Uncle girl, him woman, nephew girl, brother actress, legend actress, grandfather granddaughter, his woman, *fathers mothers*, kings queen, Jr Marie, brothers actress,

A.5 Comparative

bad worse, big bigger, bright brighter, cheap cheaper, cold colder, cool cooler, deep deeper, easy easier, fast faster, good better, great greater, hard harder, new newer, old older, quick quicker, safe safer, sharp sharper, short shorter, simple simpler, slow slower, small smaller, smart smarter, strong stronger, wide wider, young younger

A.6 Knowledge

mathematics numbers, geology rocks, linguistics languages, zoology animals, entomology insects, dermatology skin, economics markets, history past, dance movements, anatomy body, pharmacology drugs, architecture buildings, medicine bodies, biology life, oceanography oceans, meteorology weather, anthropology people, botany plants, anatomy bodies, criminology crimes, genealogy family, chemistry elements, philosophy ideas, astronomy stars, ornithology birds

A.7 Nationality

Albania Albanian, Argentina Argentinean, Australia Australian, Bulgaria Bulgarian, Cambodia Cambodian, Chile Chilean, Denmark Danish, Egypt Egyptian, England English, France French, Germany German, Greece Greek, Iceland Icelandic, India Indian,

Ireland Irish, Israel Israeli, Italy Italian, Japan Japanese, Mexico Mexican, Peru Peruvian, Poland Polish, Portugal Portuguese, Spain Spanish, Thailand Thai, Ukraine Ukrainian

A.8 Part

flock sheep, album songs, paragraph words, flock birds, team players, band musicians, army soldiers, company employees, bouquet flowers, pride lions, album photos, pack wolfs, class students, senate senators, book pages, forest trees, poem verses, fleet ship, medley melodies, arsenal weapons, forest trees, body cells, gallon ounces, chain links, movie scenes

A.9 Past Tense

dancing danced, decreasing decreased, describing described, feeding fed, flying flew, generating generated, going went, hiding hid, jumping jumped, knowing knew, listening listened, looking looked, moving moved, paying paid, running ran, saying said, screaming screamed, seeing saw, selling sold, shrinking shrank, taking took, thinking thought, vanishing vanished, walking walked, writing wrote

B Technická dokumentácia

V tejto prílohe je uvedená technická dokumentácia ku knižnici v jazyku *Python*, ktorú sme implementovali v rámci našej práce. Táto knižnica je dostupná aj na GitHube[6]. Najprv predstavíme funkcionalitu tejto knižnice a rozoberieme jej architektúru. Rozoberieme formáty súborov, ktoré používa a uvedieme návod na inštaláciu a obsluhu našej knižnice. Opíšeme orgnanizáciu súborov a priečinkov a na záver uvedieme dokumentáciu k triedam a funkciám, ktoré sme implementovali.

Naša knižnica ponúka nasledovnú funkcionalitu, ktorů sme už uviedli v Kapitole 5:

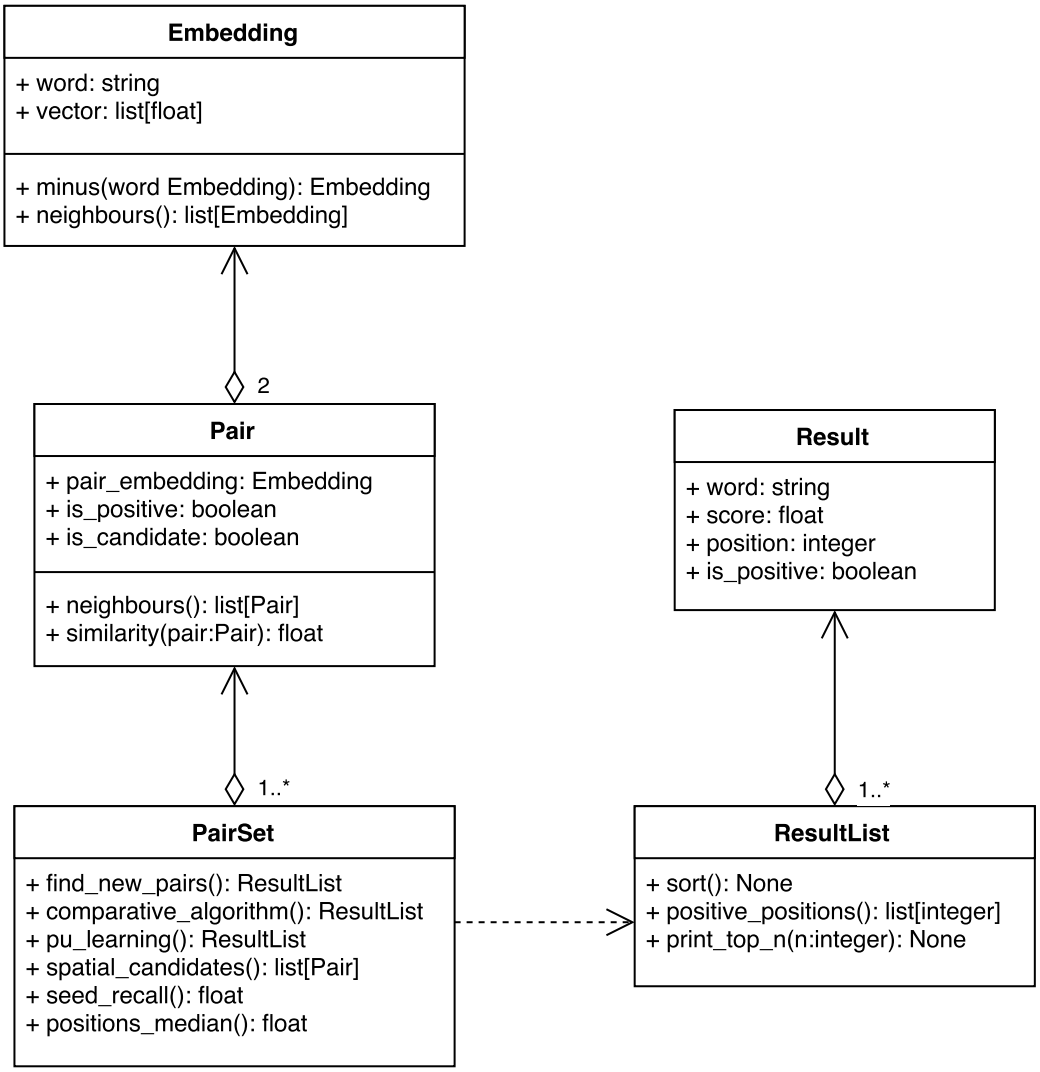
- Možnosť spustiť našu navrhnutú metódu vo viacerých variantách v zmysle jej návrhu z Kapitoly 5.
- Jednoduché rozhranie k tejto metóde v príkazovom riadku.
- Schopnosť pracovať s binárnym formátom vektorov latentných čŕt, ktoré používajú bežné knižnice (napr. *word2vec*[7] alebo *gensim*[8]).
- Možnosť rátania rozličných metrík ukážkových množín, ktoré sme použili pri experimentovaní v Kapitole 6.
- Vyhodnocovanie súborov s výsledkami, z ktorých dokáže vyrátať rozličné miery úspešnosti.

Táto knižnica nedokáže vyrátať vektory latentných čŕt z textového korpusu, na tento účel sa dá použiť napríklad už spomenutá Python knižnica *gensim*. Túto knižnicu používame aj na prácu s datasetom vektorov latentných čŕt. Dokáže z takýchto datasetov vrátiť vektor zodpovedajúci danému termu a taktiež dokáže v tomto datasete rýchlo vyhľadať n najbližších susedov pre takýto vektor. Ďalšou dôležitou použitou knižnicou je *SVM Perf*[9], pomocou ktorej rátame PU učenie.

Základom našej knižnice je 5 hlavných tried, v ktorých je implementovaná naša metóda. Tieto triedy a ich vzťahy sú zaznačené na Obrázku 11. Vstupné a výstupné

[6]Dostupná na: https://github.com/matus-pikuliak/word-embeddings
[7]Dostupná na: https://code.google.com/p/word2vec/
[8]Dostupná na: https://radimrehurek.com/gensim/
[9]Dostupná na: http://www.cs.cornell.edu/people/tj/svm_light/svm_perf.html

súbory tejto metódy sú opísané v príslušnej sekcii B.1 tejto prílohy. Okrem toho naša knižnica obsahuje aj niekoľko *helperov*, čo sú súbory pomocných funkcií a triedu `ResultFile`, ktorá slúži na spracovanie súborov s ohodnotenými výsledkami.



Obr. 11: *Diagram tried*

Tri tieto triedy majú za úlohu modelovať entity s ktorými pracujeme. `Embedding` je vektor latentných čŕt, môže sa jednať o vektor slova alebo vektor páru. `Pair` zodpovedá páru dvoch slov a skladá sa z páru vektorov. `PairSet` je zasa množina párov nad ktorou sa ráta naša metóda alebo metriky takýchto množín. Na začiatku behu metódy tento

dátový model naplníme zo vstupného textového súboru. Výsledkom je jedna inštancia triedy `PairSet`, ktorá obsahuje páry `Pairs` zodpovedajúce tomuto vstupu.

Potom môžeme nad týmto `PairSet` objektom spustiť našu metódu, alebo nechať vyrátať rozličné metriky. Pokiaľ sa pri práci s touto metódou ohodnocujú kandidáti, vytvára sa pre každé takéto hodnotenie objekt triedy `Result`, ktoré sa agregujú v objekte triedy `ResultList`. Nad týmto zoznamom výsledkov potom môžeme volať rozličné funkcie, ktoré ho spracujú.

Výsledkom našej metódy je potom súbor v špecifickom formáte. Výsledky v tomto súbore treba manuálne vyhodnotiť. Tento postup opisujeme v našom Návode na obsluhu v sekcii B.3. Vyhodnotený súbor sa potom dá načítať do objektu triedy `ResultFile`, ktorý nad týmito výsledkami dokáže spraviť určité vyhodnotenie.

B.1 Opis použitých formátov súborov

Predtým, ako si podrobnejšie opíšeme našu knižnicu uvedieme podrobnosti o rozličných formátoch súborov, ktoré v nej používame a v krátkosti predstavíme ich účel.

B.1.1 Vstupný súbor s ukážkovými pármi

Vstupný súbor je bežný textový súbor, ktorý sa pripravuje manuálne. Na každom riadku takéhoto súboru sa nachádza jeden pár termov. Pár termov je dvojica textových reťazcov oddelených medzerou. Pomocou takýchto súborov sa definujú ukážkové množiny použité pri výpočte metódy. Pre ukážku uvádzame tri riadky takéhoto súboru:

```
Athengs Greece
Paris France
Berlin Germany
```

B.1.2 Výstupný súbor s výsledkami

Výstupný súbor je textový súbor, ktorý sa generuje pri behu našej metódy a sú v ňom zaznačené jej výsledky. Na každom riadku je *príznak*, ktorý opisuje či je pár v danom riadku sémantický správny. Za ním nasleduje tabulátor a samotný pár termov, ktorý predstavuje výsledok. Výsledky sú usporiadané od najviac relevantného výsledku po

menej. Na prvom riadku je teda výsledok s najlepším skóre. Ukážka takéhoto súboru po vygenerovaní metódou je:

```
? Caracas Venezuela
? Paris Germany
? Madrid Spain
```

Príznaky správnosti sú po vygenerovaní nastavené na otázniky. Človek tento súbor vyhodnocuje tak, že tieto príznaky prestaví podľa toho, či daný pár spĺňa jeho požiadavky. Po vyhodnotení párov dokáže naša metóda takýto súbor načítať a vyhodnotiť niektoré metriky výstupu. Znak otáznika sa teda nahradí písmenkom:

- `r` - Ak je pár správny.
- `p` - Ak je pár čiastočne správny.
- `w` - Ak je pár nesprávny.

Ak sa teda pozrieme na uvedenú ukážku a vyhodnotíme ju s tým, že hľadáme hlavné mestá, tento súbor by po spracovaní vyzeral následovne:

```
r Caracas Venezuela
w Paris Germany
r Madrid Spain
```

B.1.3 Súbor so slovníkom

V tomto súbore sa nachádzajú zoznam termov, pomocou ktorého môžeme filtrovať slová vyčítané z datasetu vektorov latentných čŕt. V datasete s ktorým sme pracovali sme mali napríklad rovné 3 milióny termov a ich vektorov. Pomocou tohto súboru sme ale definovali 100.000 termov, ktoré sme z takéhoto datasetu naozaj vyčítali do modelu, s ktorým pracuje naša metóda. Na každom riadku súborku slovníka je uvedený jeden term. Riadky začínajúce znakom mriežky „#“ sú považované za komentáre.

B.1.4 SVM súbory

Pri rátaní SVM algoritmu v rámci PU učenia sa vytvárajú rozličné druhy súborov v štandardnom *libsvm* formáte. Informácie o tomto formáte sa nachádzajú na webovom sídle tejto knižnice[10].

B.1.5 Súbor s vektormi latentných čŕt

Ako súbor s uloženými vektormi latentných čŕt používame binárnu podobu dátoveho formátu, ktorú vytvárajú niektoré knižnice na spracovanie textových korpusov. Medzi tieto knižnice patria napríklad *word2vec* alebo *gensim*, ktoré sme už spomínali. Na webových sídlach týchto knižníc sú aj bližšie informácie o tomto formáte.

B.2 Návod na inštaláciu

Naša knižnica je v podstate súbor skriptov a ako taký ho netreba kompilovať či inštalovať. Zdrojový kód stačí nakopírovať do ľubovoľného priečinku. Treba však nainštalovať prerekvizity pre vykonávanie tohto kódu. Pri jednotlivých položkách uvádzame verziu, nad ktorou sme vyvíjali knižnicu my. Vyššia či nižšia verzia môže, ale nemusí, fungovať bezproblémovo. Prerekvizity našej knižnice sú:

- Operačný systém *Ubuntu 14.04*. Je možné použiť aj iné linuxové distribúcie, ich funkčnosť však nemáme overenú.
- Programovací jazyk *Python* 2.7.6.
- Knižnica *gensim* 0.12.4.
- Knižnica *NumPy* 1.11.0, ktorá je zároveň prerekvizitou knižnice *gensim*.
- Knižnica *SciPy* 0.13.3, ktoré sú zároveň prerekvizitou knižnice *gensim*.
- Interpreter jazyku Fortran *gfortran*, ktorý je prerekvizitou týchto knižníc.
- Balíček *BLAS*, ktorý je prerekvizitou týchto knižníc.

[10]Dostupné na: https://www.csie.ntu.edu.tw/ cjlin/libsvm/

Po inštalácií prerekvizít je potrebné taktiež nakonfigurovať knižnicu. Konfigurácia sa dá meniť v súbore `config.py` a v súčasnosti ponúka tri možnosti:

- `svm_folder` - Tu patrí cesta do adresára, kde sa budú vytvárať súbory pri rátaní SVM v rámci algoritmu PU učenia. Upozorňujeme, že tieto súbory môžu dosahovať aj veľkosti rádovo v stovkách MB.
- `word2vec_file` - Tu patrí cesta k súboru s vektormi latentných čŕt.
- `default_output_file` - Tu patrí cesta k súboru, do ktorého sa budú defaultne vypisovať výsledky našej metódy.

B.3 Návod na obsluhu

V tomto návode opíšeme ako používať našu knižnicu pomocou príkazového riadku. Týmto spôsobom dokážeme nad ľubovoľým vstupným súborom v správnom formáte spustiť našu metódu, resp. jej rozličné varianty. Taktiež tu opíšeme ako potom spracovať výstupný súbor. V tomto návode sa nebudeme venovať tomu, ako používať našu knižnicu na získavanie ďalších metrík o vstupnej množine, ako napríklad pokrytie množiny. Ukážky takejto práce sa však okomentované nachádzajú v súbore `README.md`. Formáty súborou už boli opísane v sekcii B.1.

```
./method.py input_file [options]

options:
-o [string] -> Vystupny subor.
               default: hodnota default_output_file z
               konfiguracneho suboru.

-t [int]    -> Kolko najlepsich vysledkov nasa metoda
               zapise do suboru.
               default: 100

-m [1..3]   -> Druh hodnotiaceho algoritmu.
```

```
                default: 1
                1: Priemerna podobnost (Avg)
                2: Maximalna podobnost (Max)
                3: PU ucenie (PU)

-n [int]     -> Velkost okolia vektoru ratana pri
                hladani kandidatov.
                default: 100

-d [1..2]    -> Pouzita miera podobnosti.
                default: 1
                1: Euklidovska podobnost
                2: kosinusova podobnost

-s [1..3]    -> Pouzita normalizacia nad vahami ukaz-
                kovych parov. Toto nastavenie ma vplyv
                iba pri pri -m 1.
                default: 1
                1: Ziadna normalizacia
                2: Standardna normalizacia
                3: Softmax normalizacia
```

Po vygenerovaní výstupného súboru sú všetky príznaky správnosti nastavená na znak otáznika. Ak chceme využívať triedu `ResultFile` na spracovanie tohto súboru, treba tieto príznaky prestaviť podľa toho, či je daný pár správny (príznak `r`), čiastočne správny (príznak `p`) alebo nesprávny (príznak `w`). Nad triedou `ResultFile` sme rozhranie v príkazovom riadku nevytvorili, podrobnosti o nej sú však uvedené v tejto dokumentácii. Ukážka takejto práci je taktiež v súbore `README.md`.

B.4 Organizácia knižnice

lib/ Adresár pre externé knižnice.

- **svm_perf/** Adresár so zdrojovým kódom knižnice *SVM Perf*.
- **__init__.py** Pomocný súbor potrebný pri importovaní súborov.
- **gensim_word2vec.py** Naša mierne upravená verzia súboru word2vec z knižnice *gensim*.

.gitignore Zoznam sůborov, ktoré ignoruje *git*.

config.py Konfiguračný súbor. Jeho položky sme vysvetlili v sekcii B.2.

data_helper.py Súbor s funkciami, ktoré nám pomáhajú pracovať s dátami. Toto sa týka cachovania už vyrátaných výsledkov, tak aby sme nemuseli niektoré výpočty opakovať.

embeddings.py Súbor s hlavnými triedami našej knižnice, ktoré sa používajú na rátanie samotnej metódy. Pri importovaní tohto súboru sa do globálnej premennej `model` načítajú vektory latentných čŕt ako objekt triedy `gensim.Word2Vec`. Toto načítanie môže v závislosti od veľkosti datasetu vektorov trvať aj niekoľko minút.

helper.py Súbor so všeobecnými pomocnými funkciami.

libraries.py Súbor, v ktorom naraz importujeme všetky externé knižnice.

method.py Tento súbor slúži ako rozhranie pre používateľa na pohodlné spúšťanie našej metódy.

README.md README súbor v *Markdown* formáte zobrazovaný aj na *GitHub* stránke tejto knižnice. Obsahuje aj ukážkové zdrojové kódy určené na prácu s našou knižnicou.

result_file.py Súbor s triedou `ResultFile`, ktorá nám pomáha spracovať vyhodnotené výsledky.

svm_helper.py Súbor s metódami, ktoré nám pomáhajú pracovať s SVM súbormi a ich formátmi.

wiki_100k.txt Zoznam 100.000 najpoužívanejších anglických slov. Formát tohto súboru sme už opísali.

B.5 Opis tried a funkcií

V tejto sekcii opíšeme funkcie, triedy a ich metódy, ktoré sú implementované v našej knižnicie. Funkcie začínajú malými písmenami, triedy veľkými. Pokiaľ funkcia náleží triede a je teda jej metódou, nachádza sa pod touto triedou odsadená od okraja. Za názvom metódy alebo funkcie sú uvedené na osobitých riadkoch jej jednotlivé atribúty. V tejto dokumentácii uvádzame len informácie o piatich hlavných triedach a niektoré dôležité funkcie. Neuvádzame tu dokumentáciu pomocných alebo triviálnych metód a funkcií. Všetky tu nepokryté časti sú zdokumentované v kóde samotnom.

load_model()
seed_file=None: string
returns: Word2Vec
Načíta a vráti `Word2Vec` model vytvorený z datasetu vektorov latentných čŕt uvedeného v konfiguračnom súbore. Tento model je ofiltrovaný na základe nášho slovníkoveho súboru. Pokiaľ sa funkcia zavolá s parametrom `seed_file`, slová z tohto súboru sa trvale pridajú do slovníka

embedding_object()
word: string
vector=None: list[float]
returns: Embedding
Pokúsi sa podľa zadaného slova `word` nájsť, či už neexistuje objekt triedy `Embedding` s takýmto slovom a vráti ho, ak áno. Pokiaľ ho nenájde, vytvorí nový takýto objekt. Pokiaľ nie je zadaný parameter `vector`, pokúsi sa daný vektor nájsť v modeli vektorov.

Embedding
word: string
vector: list[float]
Táto trieda predstavuje jeden vektor latentných čŕt priradený buď slovu alebo páru slov. Každý takýto objekt má svoj vektor `vector` a svoj term `word`, ktorý je unikátny.

__sub__()
embedding: Embedding
returns: Embeddings
Vytvorí nový objekt `Embedding`, ktorý je vytvorený z rozdielov vektorov tohto objektu a objektu z parametru `embedding`.

neighbours()
size=100: integer
returns: list[Embedding]
Nájde a vráti najbližších susedov tohto objektu z modelu. Počet je daný parametrom `size`.

Pair
embedding_1: Embedding
embedding_2: Embedding
pair_embedding: Embedding
is_candidate: boolean
is_positive: boolean
Táto trieda predstavuje pár dvoch slov. Práve takéto páry spracovávame a hľadáme. Pár sa skladá z dvoch vektorov, `embedding_1` a `embedding_2`. Z týchto sa vyráta aj ich rozdiel a uloží sa ako `pair_embedding`.

cosine_similarity()
pair: Pair
returns: float
Vyráta kosínusovú vzdialenosť ku zadanému páru `pair`. Tieto vzdialenosti sú ukladané do pamäti, aby sa podobnosť medzi dvoma pármi nerátala viac krát.

euclidean_similarity()
pair: Pair
returns: float
Vyráta Euklidovskú vzdialenosť ku zadanému páru `pair`. Tieto vzdialenosti sú ukladané do pamäti, aby sa podobnosť medzi dvoma pármi nerátala viac krát.

neighbors
size=100: integer
returns: list[Pair]
Vráti zoznam najbližších párov v priestore. Vyráta sa ako karteziánsky súčin susedstiev zložiek páru. Veľkosť `size` určuje veľkosť susedstva zložiek, ktorá sa do tohto súčinu dostane, viď `Embedding.neighbours()`.

svm_transform()
pairs: list[Pair]
distance='euclidean': 'euclidean' alebo 'cosine'
returns: string
Transformuje pár na reťazec do SVM súboru, kde hodnotami je podobnosť páru s pármi zadanými v zozname `pairs`. Typ podobnosti určuje `distance`.
Napr.: `-1 1:0.255 2:0.133 3:0.445`

svm_values()
pairs: list[Pair]
distance='euclidean': 'euclidean' alebo 'cosine'
returns: string
Pre daný pár vráti časť reťazca do SVM súboru, ktorá obsahuje hodnoty a ich indexy. Hodnotami sú podobnosti páru s pármi zadanými v zozname `pairs`. Typ podobnosti určuje `distance`.
Napr.: `1:0.255 2:0.133 3:0.445`

svm_label()
returns: 1 alebo -1

Vráti SVM príznak pre daný pár podľa toho či je považovaný za pozitívnu vzorku (1) alebo neoznačeného kandidáta (-1).

PairSet
pairs: list[Pair]
filename: string
Ide v podstate o agregáciu párov `pairs`, nad ktorou sa vykonáva naša metóda. Pokiaľ bol tento objekt vytvorený z textového súboru, jeho názov sa uloží do atribútu `filename`

create_from_file_() @classmethod
filename: string
returns: PairSet
Spracuje vstupný súbor s pármi tak, že z jednotlivých riadkov vytvorí objekty triedy `Pair` a z nich zostrojí objekt `PairSet`, ktorý je výstupom tejto metódy.

spatial_candidates()
size=100: integer
filter_positives=True: boolean
returns: list[Pair]
Vygeneruje kandidátov pre danú množinu. Spôsob generovania je opísaný v Kapitole 5. Veľkosť okolia jednotlivých vektorov je daná parametrom `size`. Parameter `filter_positives` určuje, či sa do zoznamu kandidátov dostanú také páry, ktoré tvoria tento objekt.

spatial_candidates_words()
size=100: integer
filter_positives=True: boolean
returns: list[string]
Podobná ako predchádzajúca metóda, namiesto objektov typu Pair však vracia len textové reťazce a je teda podstatne rýchlejšia.

pair_weight()
pair: Pair
distance='euclidean': 'euclidean' alebo 'cosine'
returns: float
Vráti váhu daného páru `pair` ako priemer jeho podobností z ostatnými pármi množiny tohto objektu. `distance` určuje typ podobnosti

comparative_algorithm()
seed: PairSet
testing: PairSet
candidates: list[Pair]
method='avg': 'avg' alebo 'max'
weight_type='none': 'none', 'normalized' alebo 'softmax'
distance='euclidean': 'euclidean' alebo 'cosine'
returns: ResultList
Vyráta skóre pre kandidáty podľa algoritmov Avg a Max z návrhu našej metódy. Ako ukážkové páry slúži parameter `seed`, pokiaľ nie je zadaný použijú sa páry z tohto objektu. Ako kandidáty sa použije parameter `canddidates`, pokiaľ nie sú zadané, vyrátajú sa z okolia párov tohto objektu. `testing` sú pozitívne páry, ktoré chceme tiež ohodnotiť. `method` hovorí o tom, či rátame priemernú alebo maximálnu podobnosť. `distance` popisuje použitú podobnosť. `weight_type` hovorí o spôsobe normalizácie podobnosťí pri metóde priemernej podobnosti. Pre každého hodnoteného kandidáta sa vytvori objekt triedy `Result`, ktorý toto ohodnotenie opisuje. Výsledkom tejto metódy je agregácia týchto výsledkov v objekte triedy `ResultList`.

pu_algorithm()
seed: PairSet
testing: PairSet
candidates: list[Pair]
distance='euclidean': 'euclidean' alebo 'cosine'
returns: ResultList
Podobná metóda ako predošlá `PairSet.comparative_algorithm()`. Na-

miesto porovnávania však táto hodnotí páry pomocou PU učenia. Najprv vytvorí potrebné SVM súbory pre danú množinu, nechá na nich natrénovať klasifikátor a oklasifikuje kandidáty. Pri každom behu sa vytvoria 4 súbory s rozličnou funkciou (trénovací, testovací, model a predikcie). Takáto skupinka súborov je rozlíšiteľná pomocou časovej pečiatky v názvoch súborov. Z výsledkov tejto klasifikácie sa opäť vytvorí objekt triedy `ResultList`

seed_recall()
size=100: integer
interesting_pairs: list[Pairs]
returns: float
Vyráta metriku pokrytie množiny. Podrobnosti o tejto metrike sa dajú nájsť v podkapitole 6.3. `size` určuje veľkosť okolia pri generovaní kandidátov, viď. `PairSet.generate_cand`
Parametrom
`interesting_pairs` sa dajú kontrolovať páry, ktoré sa budú hľadať pri rátaní pokrytia. Defaultne sa používajú páry tohto objektu.

positions_tesing()
repeat=10: integer
method='avg': 'avg', 'pu' alebo 'max'
***kwargs returns: float*
Vyráta medián pozícií danej množiny. Podrobnosti o tejto metrike sa dajú nájsť v podkapitole 6.4. `repeat` hovorí koľko krát sa má experiment opakovať. Z agregácie výsledkov sa potom ráta končený medián. `method` hovorí aká metóda hodnotenia sa má použiť. `**kwargs` sú ďalšie parametre poslané do algoritmov hodnotenia. Pre podrobnosti o možnostiach nastavení týchto algoritmov viď. `PairSet.comaprative_algorithm`
a
`PairSet.pu_learning()`

testing_and_training_set()
testing_proportion: float
returns: (PairSet, PairSet)
Rozdelí množinu párov na dve – testovaciu a trénovaciu množinu – v pomere podľa

`testing_proportion`. Vytvoria a vrátia sa dva nové objekty triedy `PairSet`

Result
name: string
is_positive: boolean
score: float
position: integer
Objekt tejto triedy reprezentuje jeden ohodnotený kandidát. `name` je meno tohto kandidáta, v našej metóde je to dvojica termov oddelená pomlčkami. `is_positive` hovorí, či sa jedná o kandidát alebo o testovaciu pozitívnu vzorku. `score` je samotné hodnotenie a `position` je pozícia tohto objektu v zoradenom zozname výsledkov v objekte `ResultList`

ResultList
results_list: list[Result]
Objekty tejto triedy agregujú objekty `Result` a dokážu ich ako celom vyhodnocovať.

append()
result: Result
returns: None
Pridá do zoznamu nový výsledok `result`.

sort()
returns: None
Zoradí výsledky objektu a priradí im pozície začnúc od pozície 1.

positive_positions()
returns: list[integer]
Vráti pozície výsledkov, ktoré boli označené ako výsledky pozitívnych vzoriek.

print_top_n()
n=100: integer returns: list[integer]
Vypíše `n` najlepších výsledkov.

C Elektronické médium

Priložené DVD obsahuje nasledovné súbory a adresáre:

relations/ Adresár obsahuje textové súbory s ukážkovými pármi definujúcimi sémantické vzťahy použité pri experimentoch. Formát týchto súborov bol už opísaný v technickej dokumentácii.

results/ Adresár obsahuje manuálne vyhodnotené výstupné súbory z našich experimentov. Formát týchto súborov bol už opísaný v technickej dokumentácii.

src/ Adresár so zdrojovými kódmi našej knižnice. Obsah tohto adresára bol opísaný v technickej dokumentácii.

obsah.txt Textový súbor s obsahom DVD.

referencie.bib Bibtexový súbor s referenciami tejto práce.

diplomova_praca.pdf PDF verzia tejto práce. V nej sa nachádza aj anotácia a technická dokumentácia.