VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SHLUKOVÁNÍ SLOV PODLE VÝZNAMU

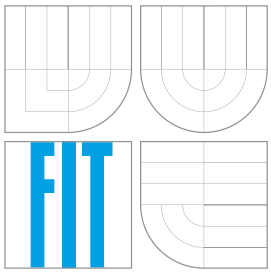
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETR HALJUK

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SHLUKOVÁNÍ SLOV PODLE VÝZNAMU

WORD SENSE CLUSTERING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETR HALJUK

VEDOUCÍ PRÁCE
SUPERVISOR

doc. RNDr. PAVEL SMRŽ, Ph.D.

BRNO 2015

Abstrakt

Tato bakalářská práce se zabývá sémantickou podobností slov. Popisuje návrh a implementaci systému, který vyhledává nejpodobnější slova a určuje sémantickou podobnost vět. Systém využívá model Word2Vec z knihovny GenSim. Vztahy mezi slovy se model učí analýzou korpusu CommonCrawl.

Abstract

This Bachelor's thesis deals with the semantic similarity of words. It describes the design and the implementation of a system, which searches for the most similar words and measures the semantic similarity of words. The system uses the Word2Vec model from GenSim library. It learns the relations among words from CommonCrawl corpus.

Klíčová slova

zpracování přirozeného jazyka, sémantická podobnost, GenSim, Word2Vec

Keywords

natural language processing, semantic similarity, GenSim, Word2Vec

Citace

Petr Haljuk: Shlukování slov podle významu, bakalářská práce, Brno, FIT VUT v Brně, 2015

Shlukování slov podle významu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. RNDr. Pavla Smrže, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Petr Haljuk
20. května 2015

Poděkování

Chtěl bych poděkovat svému vedoucímu doc. RNDr. Pavlu Smržovi, Ph.D. za odborné vedení mé práce.

Obsah

1 Úvod **3**

2 Shlukování slov podle významu **4**
2.1 Předzpracování vstupních textů . . . 4
2.1.1 Tokenizace . . . 4
2.1.2 Stemming . . . 4
2.1.3 Lemmatizace . . . 5
2.2 Distribuční sémantický model . . . 5
2.2.1 Modely založené na počtu výskytů (Count models) . . . 5
2.2.2 Modely založené na odhadu kontextu (Context-predicting models) . . . 6
2.2.3 Podobnost vektorů . . . 6
2.3 Vektory vět/dokumentů . . . 6
2.3.1 Bag-of-words . . . 6
2.3.2 TF-IDF . . . 6
2.3.3 Využití vektorů slov . . . 7
2.4 Word2Vec . . . 7
2.4.1 Continuous Bag-of-words Model . . . 8
2.4.2 Skip-gram Model . . . 9
2.4.3 Hierarchical Softmax . . . 9
2.4.4 Negative Sampling . . . 10
2.4.5 Podhodnocení častých slov . . . 10
2.4.6 Parametry ovlivňující výsledný model . . . 10
2.4.7 Další vlastnosti vektorů . . . 11

3 Návrh **12**
3.1 Požadavky . . . 12
3.2 GenSim . . . 12
3.3 Tokenizace . . . 12
3.4 Stemming . . . 12
3.5 Podobnost vět . . . 13
3.6 Korpus CommonCrawl . . . 13
3.7 Princip fungování . . . 13
3.8 Vyhodnocování . . . 14

4 Implementace **15**
4.1 Požadavky . . . 15
4.2 Architektura . . . 16
4.3 Zpracování korpusu . . . 16

4.4 Porter stemmer . . . 17
4.5 Trénování modelu . . . 17
4.6 Vyhodnocování podobností . . . 18
4.6.1 Porovnávání vět . . . 20
4.6.2 Vyhodnocení WordSim353 . . . 21
4.6.3 Vyhodnocení SemEval 2014 . . . 21

5 Vyhodnocení **23**
5.1 Způsob vyhodnocování . . . 23
5.1.1 Datová sada SemEval 2014 . . . 23
5.1.2 Testované modely . . . 23
5.2 Dosažené výsledky . . . 24
5.2.1 WordSim353 . . . 24
5.2.2 Vyhledávání podobných slov . . . 24
5.2.3 Porovnávání vět . . . 28
5.3 Shrnutí . . . 31

6 Závěr **33**
6.1 Dosažené výsledky . . . 33
6.2 Přínos práce . . . 33
6.3 Možnosti rozšíření . . . 33

A Obsah DVD **36**

Kapitola 1

Úvod

Zpracování přirozeného jazyka (NLP - Natural language processing) je obor z oblasti umělé inteligence a počítačové lingvistiky. Zabývá se interakcí mezi počítačem a člověkem pomocí přirozeného jazyka. Jednou z úloh, kterou se NLP zabývá, je i shlukování slov podle významu. Cílem je vytvořit model schopný vyhodnotit, jak významově blízké si jsou dvě slova případně dokumenty nebo jiné větší celky. Tato metrika se nazývá sémantická podobnost. Vhodným modelem pro popis sémantické podobnosti je vektorový prostor, kde jsou slova sémanticky podobná (například synonyma nebo antonyma) umístěna blíže k sobě než slova významově rozdílná. V současné době je jednou z nejperspektivnějších metod zjišťování sémantických podobností slov statistické strojové učení na rozsáhlých textových datech obsahující více než miliardu slov. Sémantická podobnost se uplatňuje v široké škále aplikací z oblasti NLP jako je vyhledávání relevantních dokumentů, strojové překlady, automatické třídění dokumentů nebo nalezení dokumentů pojednávajícím o stejném nebo podobném tématu.

Cílem práce bylo seznámit se s metodami určování sémantické podobnosti slov a navrhnout a implementovat systém, který pro zadané slovo najde další sémanticky podobná slova. Práce se dále zaměřuje na nástroj Word2Vec. Popisuje principy, na kterých funguje, a prakticky řeší vytváření vektorového modelu nad korpusem CommonCrawl. Testován byl vliv různých algoritmů trénování modelu na úspěšnost modelu, která je v závěru práce vyhodnocena na datových sadách WordSim353 a SemEval 2014 Task 10.

Kapitola 2

Shlukování slov podle významu

Shlukování slov podle významu spadá do oblasti umělé inteligence. Počítač se analýzou rozsáhlých textových dat, tzv. korpusu, učí vztahy mezi jednotlivými slovy. Takové učení probíhá například z čistého textu, kde se vztahy mezi slovy určují pomocí okolních slov. V tomto případě není potřeba, aby byl korpus nějakým způsobem anotován. Jedná se tedy o učení bez učitele. Jinou možností je učení ze stromů odpovídajících větnému rozboru.

2.1 Předzpracování vstupních textů

Před samotnou analýzou vzájemných vztahů mezi slovy je nutné korpus, který chceme analyzovat, předzpracovat. V první fázi je potřeba text rozdělit na lexikální jednotky neboli tokeny. Takto vzniklé tokeny je dále možné ještě upravovat, aby se například omezil počet různých tvarů stejných slov v modelu. Stejný proces je potřeba aplikovat i na vstupní texty například při vyhodnocování sémantické podobnosti vět.

2.1.1 Tokenizace

Tokenizace je rozdělení nestrukturovaného textu na jednotlivá slova nebo fráze. Při tomto procesu jsou z textu odstraněny závorky, tečky, čárky a další znaménka. Ve většině případů jsou v této fázi vypuštěny číslovky, spojky, předložky a další slova, která nenesou sémantický význam daného textu. Není vždy nejvhodnější rozdělovat text pouze na slova, někdy se hodí považovat například názvy složené z více slov za jeden token. Slova "Canada"a "Air"mají samostatně jiný význam než spojeni "Canada Air", kde se jedná o leteckou společnost.

2.1.2 Stemming

Slova se v korpusech objevují v různých tvarech. Pro výpočet jejich podobnosti je vhodné tyto tvary upravit na jeden, se kterým se poté počítá. Jednou z možností je stemming. Výsledkem stemmování slova je kmen[1] slova. Algoritmus, který tuto operaci provádí, se nazývá stemmer. Stemmery jsou založeny na odřezávání přípon, může se tedy stát, že dvě nesouvisející slova budou mít stejný kmen. [16]

[1] Část slova, která se v různých tvarech slova nemění

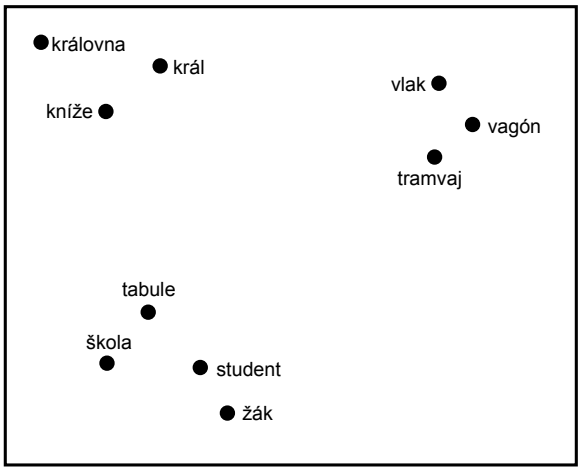
2.1.3 Lemmatizace

Lemmatizace je proces, kdy je ze slova určen jeho základní tvar neboli lemma. Narozdíl od stemmingu nelze tuto úlohu řešit algoritmicky. Proto se používá databáze slov a jejich lemmat, ve které lemmatizér najde odpovídající záznam pro dané slovo. Ani lemmatizace nemusí být jednoznačná a někdy závisí na kontextu. Článek [15] uvádí jako příklad nejednoznačného určení základního tvaru slova větu *"Jeden z nejhodnotnějších zdrojů o maďarských tancích"*, kde můžeme jako základní tvar slova *"tancích"* určit *"tanec"* nebo *"tank"*. Součástí výstupu lemmatizéru je často i slovní druh, pád a číslo slova, ze kterého bylo lemma získáno.

Častým přístupem je využití tzv. hybridního řešení, kdy je stemmer doplněn o tabulku lemmat. Tato tabulka většinou obsahuje pouze menší počet vybraných slov (např.: nepravidelná slovesa). Pokud není vyhodnocované slovo nalezeno v tabulce, použije se stemming na základě ořezávání koncovek.

2.2 Distribuční sémantický model

Distribuční sémantický model určuje podobnost slov pomocí slov vyskytujících se v jejich okolí. Tomuto okolí se říká kontext. Může to být okno několika okolních slov, případně i celé dokumenty. Tento model vychází z distribuční hypotézy, která říká, že slova, která jsou významově podobná, se nacházejí v podobném kontextu. [11] Z toho vyplývá, že jako slova podobná jsou určeny jak synonyma, tak i antonyma, protože slova s opačným významem se s velkou pravděpodobností budou vyskytovat ve stejném kontextu. V těchto modelech se přiděluje každému slovu ve slovníku vektor v rámci mnohadimenzionálním vektorového prostoru. Podobnost slov se pak určuje jako podobnost jejich vektorů. Tyto vektory se získávají analýzou korpusu.



Obrázek 2.1: Příklad rozmístění slov ve dvourozměrném prostoru

2.2.1 Modely založené na počtu výskytů (Count models)

Modely založené na počtu výskytů využívají matice spoluvýskytů, kde řádky jsou jednotlivá slova a sloupce kontexty. Tato matice obsahuje počty výskytů slova v daném kontextu (tyto hodnoty jsou často normalizované, aby se vyrovnaly rozdíly mezi dlouhými a krátkými

dokumenty). Kontext nemusí být pouze dokument nebo okolní slova, ale také například stejný druh gramatické vazby se stejným slovem. Vektory slov jsou tedy celé řádky této matice. Jelikož by při zpracování velkých dat byly vektory nepřiměřeně velké (výpočty by byly příliš náročné na paměť i čas), používá se matematická metoda SVD[2] na redukci rozměrů vektorů při zachování jejich podobnosti. Tato metoda je vhodná pro menší korpusy, nikoliv pro korpusy, které obsahují miliardy slov. [6]

2.2.2 Modely založené na odhadu kontextu (Context-predicting models)

Modely založené na odhadu kontextu nepočítají počty výskytů slov v kontextovém okně, ale snaží se určit nejpravděpodobnější slovo v daném kontextu (nebo kontext podle slova). Využívá se učení neuronových sítí. Zjednodušeně řečeno se pro každé kontextové okno v korpusu snaží určit slovo podle kontextu, a pokud neuspěje, provede úpravu vektoru. Podrobnější popis je uveden v kapitole 2.4. [4]

2.2.3 Podobnost vektorů

Protože slova máme reprezentována jako vektory a chceme určit jejich podobnost, je potřeba určit podobnost dvou vektorů. K tomu se používá kosinová vzdálenost. Ta nabývá hodnot v intervalu od 0 (zcela rozdílná slova) do 1 (stejná slova).

Mějme dva vektory slov A a B:

$$similarity = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}} \tag{2.1}$$

2.3 Vektory vět/dokumentů

Pomocí vektorů lze popisovat nejen slova, ale i celé věty nebo dokumenty. K určení těchto vektorů lze využít již natrénované vektory slov.

2.3.1 Bag-of-words

Jde o model reprezentace slov v textu. Jedná se o neuspořádanou multimnožinu slov, kde se u každého slova počítá četnost výskytů v textu. Tento model tedy ztrácí informaci o pořadí slov, a věty "Alice vidí Boba"a "Bob vidí Alici"jsou tedy považovány za ekvivalentní, ačkoli nemusí znamenat stejnou skutečnost. [13]

2.3.2 TF-IDF

Slova ve větě nebo dokumentu nemusí mít pro určování významu věty stejnou důležitost. Obecným slovům, jako sloveso "be"nebo člen "the", které se v textu vyskytují častěji, je tedy vhodné přikládat menší váhu. Jednou ze základních metod vážení slov v rámci textu je *Term Frequency - Inverse document frequency*. Tato metoda snižuje váhu slov, které se vyskytují ve velkém množství dokumentů v korpusu a naopak zvýhodňuje slova méně častá.

[2] Singulární rozklad (singular value decomposition)

Výpočet se skládá ze dvou složek. První složkou je *Term Frequency*, která popisuje, jak často se dané slovo vyskytuje v daném dokumentu. Existuje více variant této hodnoty. Nejjednodušší variantou je počet výskytů slova v dokumentu. Další možností je normalizovaní hodnoty pomocí logaritmu. Tento přístup omezuje důležitost slova, které se ve váženém dokumentu vyskytuje výrazně častěji než jiná.

$$wtf_{t,d} = 1 + \log(tf_{t,d}) \tag{2.2}$$

Kde $tf_{t,d}$ je počet výskytů vyhodnocovaného slova t_i v dokumentu d_j. Další možností je zohlednit pouze fakt, že se slovo v dokumentu vyskytuje.

$$btf_{t,d} = \begin{cases} 1 & \text{pokud } tf_{t,d} > 0 \\ 0 & \text{jindy} \end{cases} \tag{2.3}$$

Inverse document frequency reprezentuje v kolika dokumentech z celého korpusu se slovo vyskytuje. Reprezentuje tedy důležitost slova. Pokud se slovo vyskytuje ve velkém množství dokumentů, je považováno za méně důležité.

$$idf_i = \log \frac{|D|}{df_t} \tag{2.4}$$

Kde $|D|$ je počet dokumentů v korpusu a df_t udává kolik dokumentů obsahuje slovo t_i. Váhu daného slova poté spočítáme jakou součin obou složek

$$tfidf = tf \times idf \tag{2.5}$$

Po vypočtení vah pro všechna slova ve vstupním dokumentu se váhy ve většině případů znormalizují.

$$normalized\_tfidf = \frac{w}{\sqrt{w_1^2 + w_2^2 + \cdots + w_M^2}} \tag{2.6}$$

Kde w je váha, kterou normalizujeme, a w_n jsou jednotlivé vypočtené váhy v dokumentu. [7]

2.3.3 Využití vektorů slov

K reprezentaci větších celků textu můžeme využít vektory slov. Větu nebo obecně text zpracujeme jako BoW a pro každé slovo vezmeme jeho vektor. Tyto vektory poté sečteme nebo zprůměrujeme. Abychom omezili šum příliš obecných slov, je potřeba vektorům určit jejich váhu, například pomocí TF-IDF.

2.4 Word2Vec

Word2Vec [3] je nástroj pro automatické učení vztahů mezi slovy z textu. Nejedná se o jeden algoritmus, ale spíše o menší sadu algoritmů a architektur modelů pro trénování. Jde o strojové učení bez učitele využívající neuronovou síť (NNLM[3]). Oproti jiným řešením, Word2Vec nevyužívá hlubokou neuronovou síť. Použitá síť obsahuje pouze vstupní, skrytou a výstupní vrstvu. Vstupní a výstupní vrstva obsahuje vektory slov, které se trénují.

[3] Neural network language models

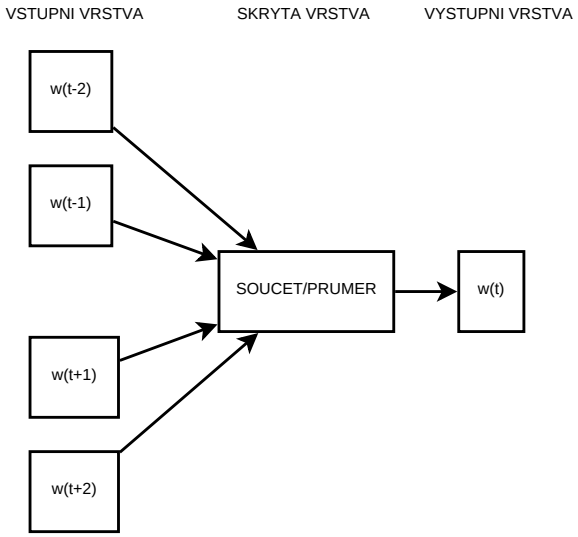
Jako výstupní vektory modelu se používají vektory ze vstupní vrstvy. Vstupní vrstva je matice o velikosti $V \times N$, výstupní $N \times V$ kde V je počet slov ve slovníku a N velikost vektoru. Vektory v obou maticích nejsou identické. Skrytá vrstva má velikost N. Všechny váhy jsou na začátku inicializovány na náhodné hodnoty a následně díky zpětné propagaci upravovány při trénování. Většina implementací umožňuje zvolit jednu ze dvou architektur: Continuous Bag-of-words a Skip-gram. Jde o klasifikátor do n tříd, kde n je velikost slovníku. V ideálním případě se pomocí Softmax regrese určí rozložení pravděpodobnosti, že slovo patří k dané třídě (slovu). Pro jednoduchost uvažujme jako kontext jedno slovo. Algoritmus se pomocí Stochastic Gradient Descent[4] snaží minimalizovat rozdíl výstupního rozložení a cílového rozložení pravděpodobnosti. Cílové rozložení je určeno z kontextu slova a nabývá hodnoty jedna pouze v aktuálním slově, všude jinde je nulové.

Protože Softmax regrese je pro rozsáhlé slovníky výpočetně náročná, využívá Word2Vec jeho aproximace, které jsou výpočetně méně náročné. Podrobnější popis je v kapitole 2.4.3 a 2.4.4. Tyto optimalizace vycházejí z analýzy četnosti slov v korpusu. Toto je jeden z důvodů, proč Word2vec vyžaduje dva průchody korpusem: [12]

1. Tvorba slovníku
2. Trénování modelu

2.4.1 Continuous Bag-of-words Model

Základní myšlenkou Continuous Bag-of-words (CBOW) je určit nejpravděpodobnější aktuální slovo podle kontextu (několik okolních slov). Pokud určené slovo neodpovídá skutečnému aktuálnímu slovu, je spočítána chyba a upraveny jednotlivé váhy neuronové sítě. [8]

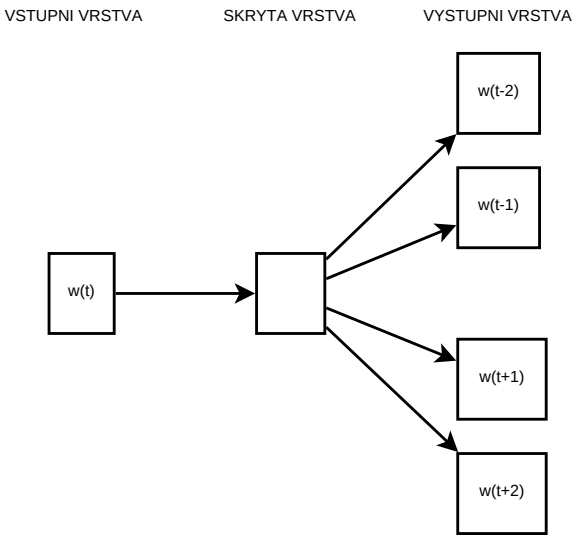


Obrázek 2.2: Schéma architektury CBOW

[4] Algoritmus pro nalezení lokálního minima funkce pomocí gradientu

2.4.2 Skip-gram Model

Tento model je opakem CBOW. Na základě aktuálního slova se určuje nejpravděpodobnější kontext. Z toho vyplývá, že tento model je výpočetně náročnější. V tomto modelu není stále stejná velikost okna. Parametrem je maximální velikost okna C. Pro každé slovo se velikost okna náhodně vybírá z intervalu $< 1; C >$. [8, 9]



Obrázek 2.3: Schéma architektury Skip-gram

2.4.3 Hierarchical Softmax

Pokud bychom počítali pravděpodobnost společného výskytu dvou slov normální softmax funkcí, bylo by trénování velmi pomalé:

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^T v_{w_I}\right)}{\sum_{j=1}^{V} \exp\left({v'_{w'_j}}^T v_{w_I}\right)} \tag{2.7}$$

Kde v_w je vektor slova w ve vstupní vrstvě a v'_w ve výstupní vrstvě. Jak je vidět ve jmenovateli, je nutné při každém výpočtu proiterovat přes všechny slova ve slovníku. Aby se tomu předešlo, Word2Vec využívá *Hierarchical softmax*. Jde o efektivní způsob, jak spočítat softmax pomocí reprezentace slov binárním stromem. Word2Vec využívá Huffmanův binární strom, ve kterém je nejčastějším slovům přidělen nejkratší kód. Každé slovo ze slovníku musí být list stromu a ke každému listu vede unikátní cesta z kořene. Tato cesta slouží k odhadnutí pravděpodobnosti slova reprezentovaného listem. Každý vnitřní uzel má určené pravděpodobnosti pro průchod do levého nebo pravého podstromu. Součin těchto pravděpodobností po cestě k listu je pak výsledná pravděpodobnost. Pravděpodobnost pro uzel n spočítáme takto:

$$p(n, left) = \sigma({v'_n}^T \dot{h}) \tag{2.8}$$

$$p(n, right) = 1 - p(n, left) \tag{2.9}$$

Kde h je výstupní hodnota skryté vrstvy. Využití tohoto algoritmu mění uspořádání výstupní vrstvy neuronové sítě, protože již neobsahuje vektor pro každé slovo ze slovníku, ale pro každý uzel ve stromě. [12]

2.4.4 Negative Sampling

Negative sampling je jiný způsob jak určit pravděpodobnost bez nutnosti procházet všechna slova ve slovníku. Na základě rozložení pravděpodobnosti založeného na frekvenci výskytů slova v korpusu je vybráno k dalších slov se kterými se počítá. Tato hodnota bývá 5-20 pro menší datové sady, pro velké 2-5.

Ve Word2Vec se rozložení pravděpodobnosti pro náhodně vybraná slova určuje jako:

$$p_n(w) = \frac{U(w)^{3/4}}{Z} \tag{2.10}$$

$$Z = \sum_{i=1}^{i} U(w_i)^{3/4} \tag{2.11}$$

Kde $U(w)$ je počet výskytů slova. [9, 5]

2.4.5 Podhodnocení častých slov

Ve velkých korpusech se některá slova mohou vyskytovat výrazně častěji než jiná. Taková slova (v angličtině například "a", "the", "in",...) mají často nižší informační hodnotu než ta méně častá a vyskytují se ve velkém množství různých kontextů. Například spoluvýskyt slov "the"a "France"nebude pro model příliš užitečný a do výsledných vektorů vnáší spíše šum.

Aby se vyrovnal rozdíl mezi častými a málo častými slovy, vypočítá Word2Vec pro každé slovo ve slovníku pravděpodobnost, že slovo bude z trénování vyřazeno.

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \tag{2.12}$$

Kde $f(w_i)$ je počet výskytů daného slova a t je zvolený práh. Typická hodnota prahu je kolem 10^{-4}. [9]

2.4.6 Parametry ovlivňující výsledný model

Na kvalitu výsledných vektorů mají vliv hlavně následující parametry:

Architektura Skip-gram (pomalejší trénování, lepší pro málo častá slova) nebo CBOW (rychlejší trénování)

Trénovací algoritmus Hierarchical Softmax (lepší pro méně častá slova) nebo Negative Sampling (lepší pro častá slova a vektory s méně rozměry)

Počet rozměrů vektoru většinou 50-1000, častá hodnota je 300

Velikost kontextové okna obvyklá hodnota pro Skip-gram je okolo 10, pro CBOW okolo 5

Počet iterací nad korpusem tato hodnota je závislá na velikosti korpusu

2.4.7 Další vlastnosti vektorů

S natrénovanými vektory pomocí Word2Vec lze nejen určovat podobnost slov nebo vyhledávat nejpodobnější slova, ale i vyhledávat slova na základě sémantických vztahů jiných slov. Uvažujme dvě dvojice slov $a : b$, $c : d$, kde d je neznámé slovo. Výsledkem by mělo být slovo d, které má ke slovu c podobný vztah jako slovo a ke slovu b. Toho lze dosáhnout pomocí běžné vektorové aritmetiky.

$$y = x_b - x_a + x_c \tag{2.13}$$

Kde x_a, x_b a x_c jsou vektory slov a y je vektor hledaného slova d. Je velice nepravděpodobné, že vektor y bude odpovídat přesně nějakému slovu ze slovníku, proto se jako výsledek použijte slovo s nejpodobnějším vektorem.

Díky této vlastnosti je možné popisovat vztahy mezi slovy pomocí vektoru. Následující vektory tak budou stejné, nebo velmi podobné. [10]

$$v_{queen} - v_{king} = v_{woman} - v_{man} \tag{2.14}$$

Kapitola 3

Návrh

3.1 Požadavky

Cílem práce by měl být systém, který pro dané slovo nalezne slova sémanticky podobná. Systém by dále měl být schopen určovat sémantickou podobnost dvou slov, případně celých vět. Vztahy mezi slovy by se měly učit z velkých korpusů, proto bude pro učení použit model založený na odhadu kontextu Word2Vec. Architektura systému by měla umožňovat učení z libovolného korpusu nebo umožnit vyhodnocování z již natrénovaných dat. Celý systém bude implementován v jazyce Python. Systém by měl umožňovat jak trénování modelu, tak jeho uložení do souboru pro pozdější vyhodnocování.

3.2 GenSim

Hlavním stavebním kamenem celého systému bude knihovna GenSim [2]. Jde o open-source knihovnu pro jazyk Python s nástroji pro modelování ve vektorovém prostoru. Obsahuje i implementaci Word2Vec, TF-IDF, LSA a dalších. Díky optimalizacím vytvořených pomocí Cython[1] poskytuje i dostatečný výkon.

3.3 Tokenizace

Tokenizaci korpusu systém neprovádí, očekává korpus již předzpracovaný. V implementaci se nachází třídy pro práci s korpusem *CommonCrawl*, jehož předzpracovaná verze je k dispozici na FIT VUTBR. Díky architektuře systému je možné pracovat i s jakýmkoli jiným korpusem i s nepředzpracovaným. Pro získání jednotlivých tokenů by v tomto případě bylo možné použít tokenizaci implementovanou v knihovně GenSim. Ta je využita i pro tokenizaci vstupních vět při porovnávání jejich sémantické podobnosti.

3.4 Stemming

Pro omezení počtu slov ve slovníku bude použit stemming. Stemmer by měl umět nejen najít kmen pro dané slovo, ale i vést statistiku, ze kterých slov byl kořen určen. Tato vlastnost je důležitá pro vyhledávání podobných slov. Jako vstup i výsledek totiž očekáváme celé slovo, nikoli pouze kmen. Systém pomocí této statistiky bude schopen určit nejčastější

[1] Optimalizační překladač pro jazyk Python

slovo odpovídající nalezenému kmenu a to použít jako výstup. Předpokládaným jazykem korpusu je angličtina, proto jako stemmovací algoritmus bude použit *Porter Stemmer*.

3.5 Podobnost vět

Vektor věty je určen váženým průměrem vektorů jednotlivých slov ve větě. Před samotným výpočtem je potřeba odstranit z věty slova, která nejsou ve slovníku. Dále je definován tzv. stoplist, který určuje slova, která budou vyřazena i přesto, že se nachází ve slovníku. Tato slova jsou určena ručně, ve většině případů, podle počtu výskytů slova v korpusu. Váhy vektorů slov jsou vypočteny pomocí TF-IDF. Tyto informace si systém ukládá do speciálního souboru při trénování na korpusu.

3.6 Korpus CommonCrawl

Jako korpus, na kterém bude model trénován, byl zvolen korpus CommonCrawl[2]. Jedná se o volně přístupný korpus složený z webových stránek. Podle [14] obsahoval korpus v listopadu 2014 135TB dat z 1,95 miliard stránek. Tento korpus je dostupný na serverech FIT VUTBR. Data jsou v rámci korpusu k dispozici jako neupravené webové stránky (včetně HTML kódu atd.), stránky doplněné o metadata nebo jako pouhá textová data.

Systém bude pracovat s již předzpracovanou verzí CommonCrawl, která byla vytvořena na FIT VUTBR. Předzpracovaný korpus může vypadat například takto:

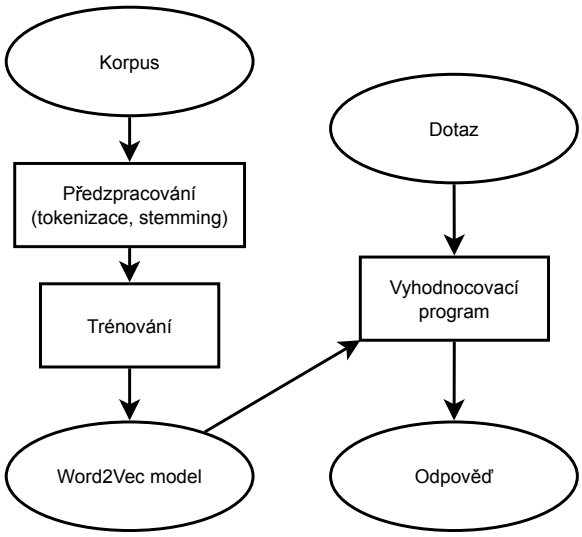
```
<doc id="id dokumentu" uri="http://webova-adresa.cz" title="">
  <p>
    <s>
      toto
      je
      věta
      <g/>
      !
    </s>
  </p>
</doc>
```

Tento příklad ukazuje jeden dokument s jedním odstavcem obsahující jednu větu. `<p>` tedy značí odstavec, `<s>` větu a `<g/>` označuje, že další znak je interpunkční znaménko, závorka a podobně. Takovýchto dokumentů se v korpusu nachází velké množství.

3.7 Princip fungování

Aby bylo možné vyhodnocovat sémantickou podobnost slov, je potřeba nejdříve natrénovat model Word2Vec (analyzovat korpus). Model s výslednými vektory pro každé slovo se poté uloží jako soubor. Při trénování může volitelně probíhat i stemming, který vede k omezení počtu slov ve slovníku. Pokud chceme využít vyhodnocování s váženým průměrem vektorů, je nutné nad korpusem ještě vytvořit statistiky frekvencí výskytů slov v jednotlivých dokumentech. Pro vyhodnocování je poté nutné načíst vektory z modelu a tyto statistiky.

[2] http://commoncrawl.org



Obrázek 3.1: Obecné schéma fungování výsledného systému

3.8 Vyhodnocování

Vyhodnocování úspěšnosti podobnosti vět je realizováno pomocí testovací sady ze SemEval 2014. Na testování sémantické podobnosti dvou slov bude využita sada WordSim353. Na testování skriptu hledající n nejpodobnějších slov nebyla nalezena odpovídající testovací sada, proto pro každý testovací příklad bude ručně vyhodnoceno, zda jsou slova na výstupu relevantní.

SemEval je soutěž skládající se z více úkolů z oboru sémantické analýzy. Jedním z úkolů bylo v ročníku 2014 vyhodnocování sémantické podobnosti dvou vět. Podobnost každého páru vět je oznámkována známkou v rozsahu 0 až 5 kde:

0 obě věty jsou úplně o něčem jiném

1 dvojice vět nejsou schodné, ale mají stejné téma

2 věty nejsou shodné, ale shodují se v pár detailech

3 dvojice vět je zhruba shodná, ale nějaká důležitá informace chybí nebo se v ní věty liší

4 dvojice vět je téměř shodná, ale liší se v nepodstatných detailech

5 dvojice vět je zcela shodná, popisují stejnou věc

Součástí testovací sady jsou i očekávané výsledky tzv. gold standard. Tyto hodnoty jsou průměrem více známek sesbíraných od lidí pro každou dvojici vět. Díky těmto hodnotám lze vyhodnotit korelace mezi gold standard a výsledky testovaného systému.

Data v testovací sadě WordSim353 obsahují vždy dvojici slov a očekávanou podobnost vyjádřenou známkou 0 až 10, kde 0 značí nejmenší podobnost. Tato data byly získány obdobným způsobem jako gold standard u sady SemEval 2014.

Kapitola 4

Implementace

Systém se skládá z následujících skriptů:

train_cc.py trénování Word2vec modelu na korpusu CommonCrawl

top_similar.py hledání sémanticky nejpodobnějších slov

sentences_similarity.py určování sémantické podobnosti dvou vět

semeval.py určí výsledky sémantické podobnosti dvou vět z testovací sady SemEval 2014

stemmer.py implementace Porterova Stemmeru

sr.py obsahuje třídu se společnou funkcionalitou při určování semantické podobnosti, využívají ji ostatní skripty

CC_corpus.py třída umožňující načítání korpusu CommonCrawl do Word2Vec

explanation.py vysvětluje, proč bylo dané slovo určeno jako sémanticky podobné

coocurence.py vytvoří matici spoluvýskytů z korpusu, kterou lze použít pro zlepšení výsledků skriptu top_similar.py

semeval.sh vyhodnocení porovnání vět z testovací sady SemEval 2014

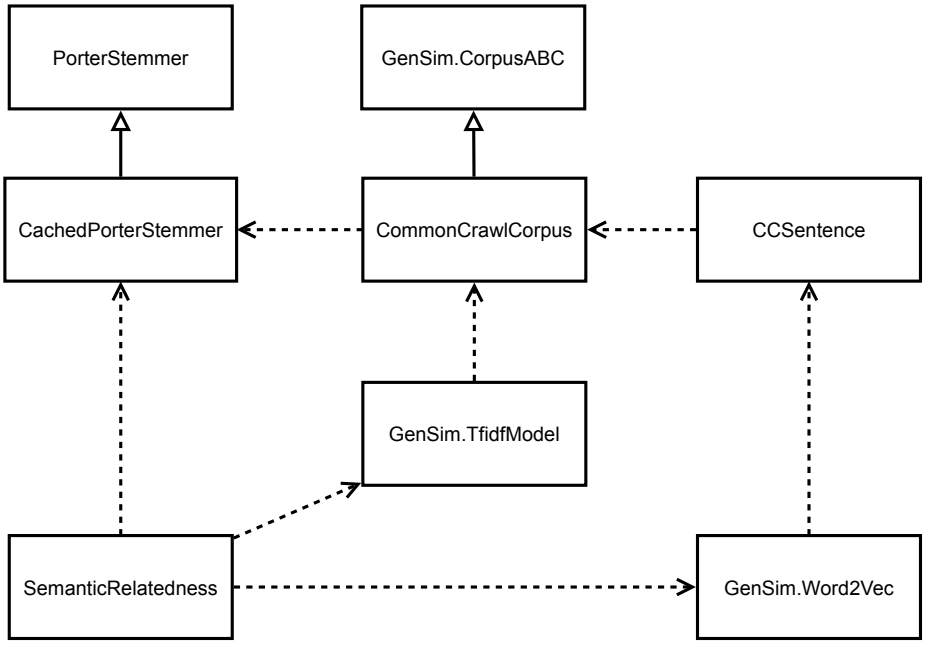
4.1 Požadavky

Aby bylo skripty možné spustit, je nutné mít nainstalovaný Python 2.7 nebo vyšší a následující knihovny:

- GenSim 0.10.3 nebo vyšší
- NumPy
- SciPy

4.2 Architektura

Hlavní třídou je *SemanticRelatedness*, kterou využívají všechny skripty pro vyhodnocování. Umožňuje nalezení nejvíce podobných slov ve Word2Vec modelu, a to jak na modelu na který byl aplikován stemming, tak i na model bez stemmingu. V případě, že má třída k dispozici i statistiku stemmeru, tak vybere nejčastější slovo, které k výslednému kmenu vedlo. Třída dále umožňuje vyhodnocení sémantické podobnosti dvou vět, které přijímá v nepředzpracované podobě. Automaticky je provedena tokenizace, stemming, vážení jednotlivých slov a je vytvořen vektor pro každou větu. Podobnost je poté kosinová vzdálenost těchto vektorů. Stemming a vážení lze vypnout. Pro vážení třída vyžaduje již sestavený slovník, který lze získat metodou *buildDictionary* třídy *CommonCrawlCorpus*. Z tohoto slovníku je pomocí třídy *TfidfModel* určena váha pro každé slovo ve větě.



Obrázek 4.1: Třídy v systému a jejich vztahy

4.3 Zpracování korpusu

Zpřístupnění korpusu pro trénování zajišťuje třída *CommonCrawlCorpus*. Tato třída implementuje rozhraní *CorpusABC* z knihovny *GenSim*. Díky tomu lze s tímto korpusem pracovat ve všech třídách *GenSim*. Pomocí metody *get_docs()* vrací jednotlivé dokumenty z korpusu formou seznamu odstavců, který obsahuje seznam vět. Věta se skládá ze seznamu slov. Jedná se tedy o více vnořených seznamů do sebe. Třída *CCsentence* je adaptér na třídu *CommonCrawlCorpus*, který umožňuje třídě *Word2Vec* iterovat korpusem po jednotlivých větách. Toto řešení bylo zvoleno proto, že *Word2Vec* iteruje vstupním textem po větách a *TfidfModel* iteruje po jednotlivých dokumentech. Třída umožňuje volitelně číst i soubory komprimované pomocí *gzip*.

Při načítání jednotlivých tokenů jsou vyřazeny čísla, protože se mohou vyskytovat v jakémkoli kontextu a jejich trénování tak nemá smysl. Zároveň jsou ořezány číslice ze začátku

a konce slov. Například token *28year* je tak trénován jako *year*. Takovýchto slov se vzhledem k tomu, že korpus pochází z webových stránek, vyskytuje mnoho. Dále jsou ignorovány slova *rd*, *st* a *th*, protože se jedná o řadové číslovky v angličtině, od kterých byly odseknuty číslice. Pro přílišnou obecnost jsou také vypouštěna slova obsahující pouze jeden znak.

4.4 Porter stemmer

Jako algoritmus pro stemming byl zvolen *Porter Stemmer*. Implementace se nachází ve třídě *PorterStemmer*. Tato třída obsahuje implementaci, která určuje kořen pro každé slovo znovu. Třída *CachedPorterStemmer* si vytváří slovník, do kterého si ukládá již určené kmeny. Pokud už byl pro vstupní slovo kmen určen, najde ho v tabulce. Zároveň si vede statistiku, kolikrát bylo jaké slovo stemmováno. Tuto statistiku lze uložit a později použít při hledání podobných slov na stemmovaném modelu. Implementace je doplněna tabulkou, která udává, jak se má jaké slovo stemmovat. Pokud není slovo v tabulce nalezeno, je použit původní stemmovací algoritmus. Ve výsledném systému je tabulka využita na správné stemmování anglických nepravidelných sloves.

Příklad tabulky pro nepravidelná slovesa:

```
do;did;done
say;said;said
make;made;made
```

První sloupec je výsledný kmen. Další sloupce jsou slova, která budou na tento kmen převedena.

4.5 Trénování modelu

K trénování se využívá třída *Word2Vec* z knihovny *GenSim*. Veškeré parametry popsané v kapitole 2.4 se nastavují v konstruktoru. Defaultně je velikost vektoru 100, velikost okna 5, architektura skipgram, hierarchical softmax, podhodnocení častých slov je vypnuto. Data jsou požadována ve formě objektu, který má implementovanou metodu *__iter__* a pro každé čtení vrací jednu větu ve formě seznamu slov. V tomto systému tomu odpovídá třída *CCSentence*. Příklad trénování modelu v GenSim:

```
# corpus_files je seznam souborů, na kterých
# bude probíhat trénování
corpus = CommonCrawlCorpus(corpus_files)
sentences = CCSentence(corpus)

# workers je počet vláken použitých při trénování
w2vmodel = Word2Vec(workers=4)

# vytvoříme slovník
w2vmodel.build_vocab(sentences)

# samotné trénování
w2vmodel.train(sentences)

# uložíme model
```

```
w2vmodel.save("model.w2v")
```

Tento kód uloží celý model včetně vstupní a výstupní vrstvy neuronové sítě. Pokud již dále chceme model používat pouze k vyhodnocování, umožňuje GenSim uložit pouze vektory slov buď v binárním nebo textovém formátu. Oba formáty jsou kompatibilní s implementací Word2Vec v jazyce C.

```
#uložení modelu v textovém formátu
w2vmodel.save_word2vec_format('model.txt', binary=False)

#uložení modelu v binárním formátu
w2vmodel.save_word2vec_format('model.bin', binary=True)
```

Na trénování modelu je v systému určen skript *train_cc.py*, který umožňuje volit parametry přímo z příkazové řádky. Zároveň umožňuje použít stemming, uložit statistiku stemmeru a statistiku o frekvencích výskytů slov v rámci korpusu. Syntaxe volání skriptu je následující:

```
./train_cc.py adresar_korpusu -ar skipgram|cbow
-al hierarchical|negative -s velikost_vektoru -ws velikost_okna
[-sub prah] [--nodict] [-st [--stemmstat soubor]]
-o soubor_modelu
```

Pokud je zadán *-sub*, použije se podhodnocení častých výrazů s daným prahem. Parametr *--nodict* způsobí, že se nevytvoří slovník s frekvencemi výskytů slov. Parametr *-st* zapne stemmer, pokud požadujeme i vytvořit statistiku stemmeru, použijeme parametr *--stemmstat*.

Pokud je model uložen ve formátu GenSim, je možné pokračovat v trénování modelu. Tímto způsobem lze řešit situaci, kdy je model trénován nad korpusem, jehož části se nachází na různých serverech atd. Při pokračování trénování se do slovníku již nepřidávají další slova, a není možné měnit parametry trénování (architektura, algoritmus atd.). Stačí tedy jeden průchod korpusem. Příklad s knihovnou GenSim:

```
#načtení modelu
w2vmodel = Word2Vec.load("model.w2v")

#trénování na nových větách
w2vmodel.train(new_sentences)
```

I v tomto případě lze využít skript *train_cc.py*:

```
./train_cc.py adresar_korpusu -m soubor_modelu [--nodict]
[-st [--stemmstat soubor]] -o soubor_modelu
```

4.6 Vyhodnocování podobností

Podobnost slov nebo i vět je počítána jako kosinová vzdálenost jejich vektorů. K těmto výpočtům je využita knihovna *NumPy* pro Python. Knihovna *GenSim* přímo obsahuje implementaci pro nalezení n nejpodobnějších slov. Následující příklad demonstruje vyhledávání nejpodobnějších slov jak pro jedno slovo, tak pro více slov. Knihovna *GenSim* umožňuje

zadat slova, jejíchž významu by výsledky měly odpovídat, tak i slova, jimž by výsledky měly odpovídat nejméně. Tato funkcionalita je implementována tak, že se hledají nejpodobnější slova pro vektor, který vznikne váženým průměrem všech slov. Slova, kterým se má výsledek nejvíce podobat mají váhu 1, v opačném případě mají váhu -1

```
# vyhledá 10 slov s nejvyšší kosinovou vzdáleností jejich vektorů
w2vmodel.most_similar('queen')

# vyhledá 10 slov nejvíce odpovídajících dvojici slov women a
    ↪ king
# a nejméně odpovídající slovu man
print w2vmodel.most_similar(positive=['woman','king'], negative=[
    ↪ 'man'])
```

Třída *SemanticRelatedness* rozšiřuje možnosti této metody o zavedení stemmingu, kde je nutné pro vstup nejdříve určit kmeny a poté pro kmeny na výstupu opět určit nejpravděpodobnější slova. Parametr *min_count* umožňuje ignorovat slova, která se v korpusu vyskytla méněkrát než je tento parametr. To dovoluje odfiltrovat z výsledku málo častá slova, která se v daném kontextu vyskytla jen s minimální četností a nejsou tolik relevantní. Následující kód zobrazuje zjednodušenou ukázku hledání nejpodobnějších slov se stemmingem a filtrováním málo častých slov pro jedno vstupní slovo.

```
# předpokládejme, že proměnná word obsahuje vstupní slovo
# inicializace stemmeru včetně tabulky nepravidelných
# anglických sloves a načtení Word2Vec modelu
stemmer = PorterStemmer(dictionary="stemmer.csv")
model = Word2Vec.load("model.w2v")

#určení kmene zadaného slova
stem = stemmer.getStem(word)

# získání kosinové vzdálenosti vektoru zadaného slova
# a vektorů ostatních slov ve slovníku
stems_similarity = model.most_similar(stem, topn=None)

# vytvoření setříděného seznamu indexů slov podle
# kosinové vzdálenosti
similar_stems = argsort(stems_similarity)[::-1]

# nalezení 10 nejpodobnějších slov
similar_words = []
i = 0
for similar_stem_index in similar_stems:
  # které slovo odpovídá indexu
  a = self.model.index2word[similar_stem_index]

  # pokud se slovo vyskytuje méně často než prah
  # nebo se jedná o vstupní slovo, je z vyhodnocování vyřazeno
  if (a==word or model.vocab[a].count<min_count):
    continue
```

```
  # určení nejčastějšího slova pro daný kmen
  word = stemmer.getMostFrequentWord(a)

  similar_words.append((word, stems_similarity[similar_stems]))
  i=i+1
  if i>=10:
    break

print similar_words
```

V rámci systému tuto funkcionalitu implementuje skript *top_similar.py*. Skript přijímá dotazy ze standardních vstupu, vždy jeden dotaz na jeden řádek. Více slov je odděleno mezerou. Negativní slova jsou uvozena pomlčkou. Skript dále umožňuje filtrovat výstupní slova podle slovních druhů. K tomu potřebuje soubor obsahující přiřazení slovních druhů k jednotlivým slovům. Tento soubor lze vytvořit například pomocí nástroje TreeTagger[1] nebo využít již otagovanou verzi korpusu CommonCrawl dostupnou na FIT VUTBR. V případě využití stemmeru se systém pokusí dohledat slovo odpovídající kmenu tak, aby se jednalo o požadovaný slovní druh.

```
./top_similar.py model [-f format_modelu]
 [-s statistika_stemmeru][--showstems]
 [-m minimalni_pocet_vyskytu] [-c matice_spoluvyskytu]
 [-p slovni_druhy]
```

Volitelně je možné pro vyhodnocování použít matici spoluvýskytů k odfiltrování slov, která lze určit pouhým spoluvýskytem. Díky tomu se na výstupu omezí slova, která nejsou sémanticky podobná, ale mohou se vyskytovat třeba v rámci víceslovného označení vedle sebe. Matici spoluvýskytů je možné vytvořit skriptem *coocurence.py*. Jedná se o jednoduchou implementaci, kdy jsou počítány spoluvýskyty jednotlivých slov v daných kontextech a počet výskytů je poté vážen pomocí TF-IDF, aby se omezil vliv příliš častých slov. Kontext může být věta nebo dokument. Pro omezení velikosti matice, lze zadat práh minimálního počtu výskytů slov, které se do matice zahrnou. Skript si sám nevytváří slovník, ale využívá slovník vygenerovaný skriptem *train_cc.py*. Pro lepší pochopení, proč bylo určeno dané slovo jako podobné lze použít skript *explanation.py*.

```
./coocurrence.py adresar_korpusu slovník [-m minimalni_pocet]
[-t veta|dokument]

./explanation.py adresar_korpusu vstupni_slovo vystupni_slova
```

4.6.1 Porovnávání vět

Jak již bylo zmíněno, k určení vektoru věty je využit vážený průměr vektorů jednotlivých slov. To opět zajišťuje třída *SemanticRelatedness*. Ze slovníku vytvořeného nad korpusem se vytvoří TFIDF model, který slouží pro získání vah slov. Následující ukázka demonstruje určení vektoru věty.

[1] http://www.cis.uni-muenchen.de/ schmid/tools/TreeTagger/

```
dictionary = Dictionary.load("slovnik.dict")

#vytvoření BoW z věty
bow = list(utils.tokenize(sentence, deacc=False, to_lower=True))
bow = stemmer.getStems(bow)

#odfiltrovaní slov, které mají pouze jeden znak
#a slov, které nejsou ve slovníku
bow_filter = [word for word in bow if word in self.model and len(
    ↪ word)>1)]

#vypočítání Tf-idf vah
tfidf = TfidfModel(dictionary=dictionary)
tfidf_sentence = tfidf[dictionary.doc2bow(bow_filter)]

#určíme výsledný vektor věty
word_vectors = [self.model[self.dictionary[word[0]]] for word in
    ↪ tfidf_sentence]
weights = [word[1] for word in tfidf_sentence]
vector = numpy.average(array(word_vectors), axis=0, weights=
    ↪ weights)
```

Použití vážení je volitelné. Demonstrace je k dispozici ve skriptu *sentences_similarity.py*.

```
./sentences_similarity.py model [-d soubor_slovniku]
[-f format_modelu] [-st] [--stoplist slova]
```

Pokud není zadán slovník, je váha všech slov ve větě 1. Jednotlivá slova pro parametr *--stoplist* jsou odděleny středníkem.

4.6.2 Vyhodnocení WordSim353

Vyhodnocení úspěšnosti na této datové sadě je realizováno pomocí Pearsonova korelačního koeficientu nebo Spearmanova korelačního koeficientu. Jako vstup zpracovává soubor obsahující na řádku dvě slova a očekávanou podobnost. Tyto hodnoty jsou odděleny tabulátorem. Výsledkem je korelační koeficient mezi očekávanými výsledky a kosinovou vzdáleností daných dvojic slov.

```
./wordsim.py model -i soubor_wordsim.tab [-f format_modelu] [-st]
[-l logovaci_soubor]
```

Logovací soubor obsahuje vstupní slova, očekávané výsledky, vypočtené výsledky a jejich rozdíl. Díky tomu lze určit, v jakých případech systém vykazuje nejvyšší chybu.

4.6.3 Vyhodnocení SemEval 2014

Sada SemEval 2014 Task 10 se vyhodnocuje pomocí Pearsonovi korelace. Tato sada obsahuje i svůj vyhodnocovací skript napsaný v jazyce Perl. Pro porovnávání vět v této datové sadě se používá postup popsaný v kapitole 4.6.1. Vyhodnocení výsledku pro daný model zajišťuje

skript *semeval.sh*. Tento skript interně volá *semeval.py*, který pomocí modelu vypočítá podobnosti vět a *correlationnoconfidence.pl*, který vyhodnotí Pearsonovu korelaci.

```
./semeval.sh model [-d soubor_slovniku] [-f format_modelu]
[-st] [-l logovaci_soubor]
```

Logovácí soubor má stejný formát jako v kapitole 4.6.2.

Kapitola 5

Vyhodnocení

5.1 Způsob vyhodnocování

V této kapitole jsou popsány výsledky na obou datových sadách s různými parametry trénování modelu. Pro datovou sadu SemEval 2014 je provedeno porovnání s výsledky účastníku této soutěže. Pro WordSim353 je porovnání provedeno s výsledky uvedenými na webu Association for Computational Linguistics [1]. Dále je vyhodnocena relevance výstupu při hledání 10 nejpodobnějších slov.

5.1.1 Datová sada SemEval 2014

Testovací data ze SemEval 2014 Task 10 obsahují přibližně 3750 dvojic vět. Ty jsou rozděleny podle původu do následujících skupin.

Headlines každá věta je novinový titulek

Deft-News část článků z projektu DEFT DARPA

Deft-Forum části příspěvků z diskuzního fóra projektu DEFT DARPA

OnWN definice z databází OntoNotes a WordNet

Tweet-News jedna z vět je novinový titulek a druhá je reakce na serveru Twitter

Images věty popisující obsah obrázku

5.1.2 Testované modely

V rámci testování bylo natrénováno 8 modelů na části korpusu *CommonCrawl* obsahující přibližně 1,6 miliardy slov. Pro obě architektury (CBOW a Skip-gram) byly testovány oba algoritmy (Hierarchical Softmax a Negative Sampling). Každá kombinace architektury a algoritmu byla trénována ve variantách se stemmingem a bez stemmingu. Všechny modely byly trénované na stejné části korpusu. Všechny modely využívaly podhodnocení častých výrazů s prahem 10^{-4}. Velikost vektoru byla zvolena 300. V případě algoritmu Negative Sampling byl počet slov k zvolen 10. Kontextové okno bylo pro CBOW zvoleno 5, pro Skip-gram 10.

5.2 Dosažené výsledky

Při trénování modelů bylo zjištěno, že trénování bez stemmingu je průměrně o 28% rychlejší než trénování modelu se stemmingem. Stemming snížil počet slov ve slovníku o přibližně 20%. Průměrná doba trénování byla 20 hodin a 11 minut.

5.2.1 WordSim353

Tabulka 5.1 popisuje korelace s očekávaným výstupem pro jednotlivé testované modely. Nejlepší dosažená korelace jiných systémů je podle [1] 0,81 u Spearmanova korelačního koeficientu. Dosažený výsledek 0,725 byl v době psaní práce čtvrtý nejlepší. Nejlepší dosažený Pearsonův korelační koeficient je 0,674, ale tato hodnota není známá u všech zveřejněných výsledků. Z tabulky vyplývá, že nejlepších výsledků dosahuje architektura CBOW s algoritmem Negative Sampling. Z tabulky lze také vyčíst pozitivní vliv stemmingu na výsledky, které vylepší o zhruba 10%.

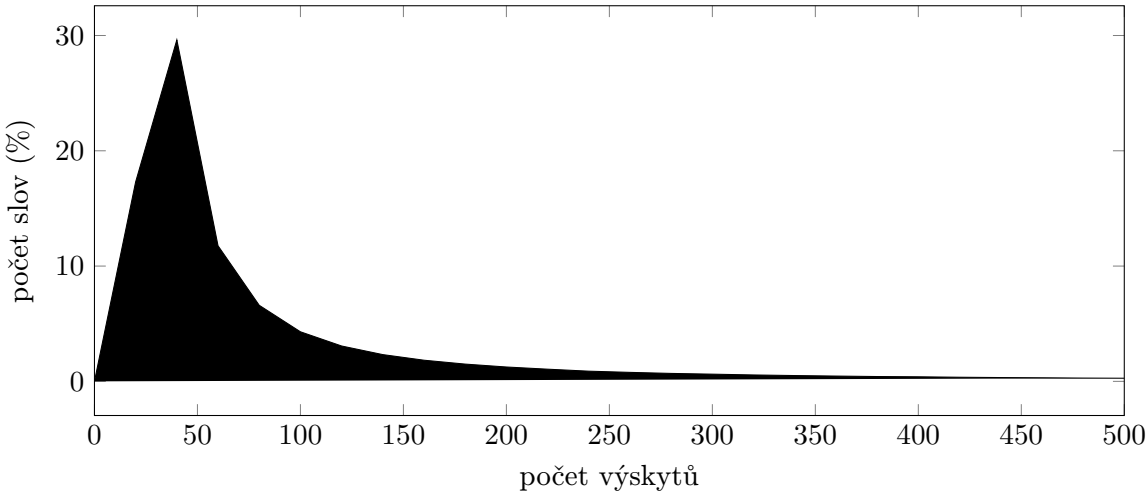
Architektura	**CBOW**				**Skip-Gram**			
Algoritmus	**NS**		**HS**		**NS**		**HS**	
Stemming	**Ano**	**Ne**	**Ano**	**Ne**	**Ano**	**Ne**	**Ano**	**Ne**
Pearson	0,687	0,585	0,641	0,589	0,653	0,510	0,600	0,526
Spearman	0,725	0,610	0,691	0,609	0,653	0,558	0,611	0,533

Tabulka 5.1: Korelace výsledků všech modelů s gold standard

NS v tabulce značí Negative Sampling a *HS* značí Hierarchical Softmax.

5.2.2 Vyhledávání podobných slov

Z provedených experimentů vyplynulo, že zásadním parametrem je prahová hodnota minimální frekvence výskytu slov v korpusu, mezi kterými hledáme výsledky. V korpusu se nachází velké množství slov s velmi nízkou frekvencí výskytu.



Obrázek 5.1: Histogram počtu výskytů slov

Z histogramu 5.1 vyplývá, že velká část slov se v korpusu vyskytuje méně než 100x. Při celkovém počtu slov v korpusu (přibližně 1,6 miliardy) se tedy tato slova vyskytují

velmi řídce. Dle měření se stokrát a méně vyskytuje 70% slov ve slovníku. Tisíckrát a méně se vyskytuje 91% slov ze slovníku. Velká část těchto slov jsou překlepy, neznámé zkratky, přezdívky z internetových diskusí apod. Tato slova jsou ve většině případů nerelevantní v jakémkoli výstupu.

Slovo	**Kos. vzdálenost**
nimera	0,66427
iridia	0.62728
aleppa	0.61980
medb	0.60519
kianra	0.60270
venommennonn	0.60078
tokelove	0.59723
excluders	0.59179
tearling	0.58518
kolache	0.57811

Tabulka 5.2: Nejpodobnější slova pro 'queen' bez filtrování málo častých slov

Slovo	**Kos. vzdálenost**
king	0,52455
princess	0,44393
elizabeth	0,38678
monarch	0,38339
jubilee	0,37956
majesty	0,37844
borough	0,36700
marie	0,36602
victoria	0,35185
scot	0,34601

Tabulka 5.3: Nejpodobnější slova pro 'queen' s odfiltrovanými slovy s počtem výskytů pod 10000

Z tabulek 5.2 a 5.3 vyplývá, že vyřazení málo častých slov značně zlepšuje výsledky. V levé tabulce není ani jedno ze slov relevantní. U druhé tabulky je relevantních 80% slov. Nejvhodnějším řešením je odfiltrování těchto slov přímo během trénování. Tato slova se poté ani nezapočítávají do kontextu.

Ve všech případech byl využit stemming a pro výsledná slova byla použita nejčastěji stemmovaná slova. V tabulce 5.3 a dalších jsou slova filtrována podle slovních druhů. Pokud systém dovede určit slovní druh vstupního slova, použije jako výstup pouze slova se stejným slovním druhem. Dále jsou z výstupu odstraněna slova, které lze získat pomocí spoluvýskytů v rámci dokumentu. Taková slova často nemají podobný význam, ale pouze například tvoří slovní spojení s daným slovem. Tabulka 5.4 demonstruje výstup bez tohoto filtrování. Odfiltrována byla slova *bee (včela)* a *hive (úl)*. Tato slova se na výstupu vyskytla hlavně díky pojmu *queen bee (včelí matka)* související s chovem včel.

Slovo	**Kos. vzdálenost**
king	0,52455
princess	0,44393
elizabeth	0,38678
monarch	0,38339
jubilee	0,37956
majesty	0,37844
bee	0,37590
borough	0,36700
marie	0,36602
hive	0,36516

Tabulka 5.4: Nejpodobnější slova pro 'queen' bez odstranění slov získaných spoluvýskytem

Slovo	Kos. vzdálenost
workstation	0.63753
device	0,58010
server	0,54951
system	0,52185
interface	0,49460
pda	0,47688
macintosh	0,47418
desktop	0,46155
modem	0,44870
server	0,42854

Tabulka 5.5: Nejpodobnější slova pro 'computer' s filtrováním málo častých slov a stemmingem, architektura skip-gram, algoritmus hierarchical softmax

Slovo	Kos. vzdálenost
workstation	0,64500
device	0,58250
laptop	0,57835
server	0,52774
supercomputer	0,52425
interface	0,52357
macintosh	0,52052
system	0,51906
pda	0,49596
desktop	0,46150

Tabulka 5.6: Nejpodobnější slova pro 'computer' s filtrováním málo častých slov a stemmingem, architektura skip-gram, algoritmus negative sampling

Tabulka 5.3 obsahuje výstupy, u kterých je třeba diskutovat jejich relevanci. Slovo *Elizabeth* souvisí s britskou královnou Alžbětou případně Alžbětou II. Slovo *jubilee* také souvisí s královnou Alžbětou II, konkrétně s výročím šedesáti let její vlády v roce 2012. Tento výsledek ukazuje, že sémantická podobnost slov se v čase mění a reaguje na aktuální události. Slova *jubilee* a *queen* by byla před rokem 2012 považována za nepodobná a do budoucna bude vnímání jejich podobnosti pravděpodobně znovu klesat. Slovo *borough (městská část)* se ve výstupu vyskytuje kvůli obvodu Queens města New York. *Marie* se na výstupu objevuje z více důvodů. Hlavně ale kvůli spojitosti s britskou královnou Marií z Tecku, universitou nesoucí její jméno (Queen Mary University of London) a skotské královně Marii Stuartovně. Podobně slovo *Victoria* souvisí s britskou královnou Viktorií. Slovo *scot (skotský)* se ve výsledcích vyskytuje hlavně v souvislosti s britskou královnou, popřípadě Marií Stuartovnou. Tato souvislost je ale poměrně slabá a toto slovo lze považovat za nerelevantní výsledek.

Dále byl testován vliv architektury modelu a algoritmu na výsledná slova. Všechny modely poskytly pro slovo *computer (počítač)* výsledky, které lze považovat za relevantní. Filtrování podle slovních druhů odstranilo přídavná jména *remote (vzdálený)*, *portable (přenosný)* a *readable (čitelný)*. V tomto případě nejlepších výsledků dosáhly obě architektury s algoritmem Negative Sampling.

Dále bylo vybráno 6 slov nacházející se ve slovníku, pro která byla nalezena nejpodobnější slova a ručně zhodnocena relevance výstupu. Byly vybrány různé slovní druhy.

Slova v tabulce 5.9 jsou ke vstupnímu slovu *elephant (slon)* ve většině případů relevantní. Ve většině případů se jedná o africká zvířata. Slovo *Ivory* znamená slonovinu. Méně relevatní je *whale (velryba)*, kde je nejčastější společné kontextové slovo *water (voda)*. Pro slovo *anime* nebyla analýzou objevena žádná vazba ke slovu *elephant*. Tato vazba pravděpodobně pochází z faktu, že slon je poměrně častou postavou dětských animovaných seriálů. Tabulka 5.10 zobrazuje nejpodobnější slova pro sloveso *borrow (půjčit si)*. Všechna slova odpovídají slovesům, která popisují činnost, kdy se s něčím manipuluje, něco se splácí nebo kupuje. Nejvíce relevantní sloveso *lend (půjčit někomu)* ale chybí.

Tabulka 5.11 obsahuje nejpodobnější slova ke slovu *Hobbit* (národ z knihy Pán prstenů). Většina výsledků jsou postavy a místa z knihy Pán Prstenů. Lze je tedy považovat za relevantní. Slovo *trilogy* se ve výstupu vyskytuje z důvodu, že filmová adaptace knihy

Slovo	Kos. vzdálenost
workstation	0,58529
device	0,53652
processor	0,42543
microprocessor	0,42382
interface	0,41766
modem	0,40223
laptop	0,40153
network	0,38547
desktop	0,38251
modem	0,38132

Tabulka 5.7: Nejpodobnější slova pro 'computer' s filtrováním málo častých slov a stemmingem, architektura cbow, algoritmus hierarchical softmax

Slovo	Kos. vzdálenost
workstation	0,60528
device	0,56335
server	0,52313
pc	0,51824
processor	0,48775
laptop	0,48572
desktop	0,47557
microprocessor	0,45760
network	0,44757
modem	0,44444

Tabulka 5.8: Nejpodobnější slova pro 'computer' s filtrováním málo častých slov a stemmingem, architektura cbow, algoritmus negative sampling

Slovo	Kos. vzdálenost
rhino	0,45108
ivory	0,44101
zoo	0,40226
whale	0,37214
donkey	0,36167
lion	0,35948
anime	0,35722
monkey	0,34803
antelope	0,34625
alligator	0,33446

Tabulka 5.9: Nejpodobnější slova pro 'elephant' (slon), které se v korpusu vyskytlo 71719x

Slovo	Kos. vzdálenost
repay	0,54085
owe	0,39371
bail	0,34746
invest	0,34202
buy	0,33585
pay	0,32576
swap	0,29895
rent	0,29832
sell	0,29355
deposit	0,29026

Tabulka 5.10: Nejpodobnější slova pro 'borrow' (půjčit si), které se v korpusu vyskytlo 126192x

Hobbit je trilogie. S trilogií souvisí i podobnost se slovem *Batman*, neboť filmy režiséra Christophera Nolana, ve kterých tato postava vystupuje, jsou také trilogie. V korpusu se několikrát vyskytuje i označení jako *Hobbit trilogy* a *Batman trilogy*. Přesto lze slovo *Batman* považovat za nerelevantní ke slovu *Hobbit*. Také pro přídavné jméno *good (dobrý)* byly nalezeny povětšinou odpovídající výsledky. Ve většině případů jde o různě stupňované slovo *dobrý*, případně o opaky. Slova *hard (těžký)* a *done (dodělaný)* nejsou relevantní a ve výsledcích se objevili z důvodu, že se vyskytují v okolí podobných slov.

I v případě tabulek 5.13 a 5.14 lze označit výsledky za relevantní. Pro vstupní slovo *singer (zpěvák)* jsou výstupní slova hráči na různé hudební nástroje nebo jiné profese spojené s hudbou. Slovo *Beyonce* je umělecké jméno americké zpěvačky a *Grammy* je název hudebního ocenění. Pro slovo *flower (květina)* jsou výsledky relevantní. Za zmínku stojí slovo *butterfly (motýl)*, které má velký počet spoluvýskytů se slovem *flower* a často se vyskytuje v kontextu slov souvisejících s přírodou.

Výsledky v tabulkách 5.15 a 5.16 ukazují možnost práce s homonymy. V prvním případě je slovo *Apple* výrobce elektroniky, v druhém případě se jedná o ovoce. Bez bližší specifikace

Slovo	Kos. vzdálenost
frodo	0,54000
bilbo	0,53429
aragorn	0,47493
mordor	0,47312
trilogy	0,44890
shire	0,43489
sauron	0,41689
lord	0,37666
rohan	0,33569
batman	0,32839

Tabulka 5.11: Nejpodobnější slova pro 'hobbit', které se v korpusu vyskytlo 38899x

Slovo	Kos. vzdálenost
bad	0,69553
decent	0,66512
nice	0,65459
terrible	0,55241
excellent	0,54778
lousy	0,54053
fantastic	0,51488
done	0,51432
terrific	0,51156
hard	0,50840

Tabulka 5.12: Nejpodobnější slova pro 'good' (dobrý), které se v korpusu vyskytlo 5680134x

Slovo	Kos. vzdálenost
vocalist	0,70780
songwriter	0,62642
guitarist	0,56130
musician	0,53230
vocal	0,48890
pianist	0,48195
drummer	0,47404
rapper	0,47236
beyonce	0,47171
grammy	0,47002

Tabulka 5.13: Nejpodobnější slova pro 'singer' (zpěvák), které se v korpusu vyskytlo 197873x

Slovo	Kos. vzdálenost
vase	0.49617
orchid	0.48436
blossom	0.46357
poppy	0.44404
lieu	0.42777
butterfly	0.42205
herb	0.41053
sunflower	0.41036
lavender	0.40180
wildflower	0.39973

Tabulka 5.14: Nejpodobnější slova pro 'flower' (květina), které se v korpusu vyskytlo 262543x

byly určeny jako nejpodobnější slova odpovídající výrobci elektroniky. Vetšina slov jsou výrobky této firmy, případně firmy zabývající se stejnou oblastí trhu. V druhém případě bylo přidáno na vstup slovo *fruit (ovoce)*. Výsledky jsou poté částečně relevantní tomuto smyslu slova. Poslední dvě slova jsou pro daný dotaz nerelevantní, neboť odpovídají jinému smyslu slova *Apple*.

Slova lze také rozdělit do požadovaného počtu shluků významově podobných slov bez nutnosti zadávat slova, ke kterým chceme ostatní přiřazovat. Tabulka 5.17 představuje jeden výsledný shluk z celkových 200 shluků vytvořenchý metodou k-means z přibližně 8000 nejčastějších slov ze slovníku. Tyto shluky částečně odpovídají výsledkům vyhledávání podobných slov.

5.2.3 Porovnávání vět

Pro každou část datové sady SemEval 2014 Task 10 byl spočítán Pearsonův korelační koeficient pro každý natrénovaný model. Každý model byl vyhodnocen dvakrát. V jednom případě se vektor věty počítal jako průměr vektorů slov, v druhém případě bylo přidáno

Slovo	Kos. vzdálenost
macbook	0,51675
itunes	0,49341
samsung	0,48223
siri	0,46279
ipod	0,45824
microsoft	0,45447
ios	0,45324
macintosh	0,43607
3gs	0,43255
blackberry	0,41941

Tabulka 5.15: Nejpodobnější slova pro 'apple' (jablko)

Slovo	Kos. vzdálenost
blueberry	0,67092
orchard	0,65268
pineapple	0,65181
vegetable	0,64064
veggie	0,63448
grapes	0,63113
pear	0,62331
juice	0,61913
blackberry	0,60646
itunes	0,60625

Tabulka 5.16: Nejpodobnější slova pro dvojici 'apple' (jablko) a 'fruit' (ovoce)

slova ve shluku	
kingdom	prince
princess	sir
crown	king
empire	duke
palace	kings
earl	queen
royal	dynasty
dynasty	castle
imperial	

Tabulka 5.17: Shluk nejvíce odpovídající sémanticky podobným slovům pro slovo 'queen'

Tf-idf vážení. Tabulky 5.18 a 5.19 zobrazují výslednou korelaci s očekáváným výstupem pro každou sadu a každý model. Tučně zvýrazněné hodnoty jsou nejvyšší dosažené.

Dosažené výsledky jsou oproti výsledkům soutěže SemEval 2014 uvedenými v tabulce 5.20 nadprůměrné. Model s nejvyšší průměrnou korelací je CBOW s Hierarchical softmax, bez použití stemmingu a vyhodnocovaný bez vážení. Vážení se pozitivně projevilo pouze u zhruba poloviny testovací sady.

Stemming má v této úloze na výsledky vyhodnocování minimální vliv (maximálně v řádech jednotek procent). Lze konstatovat, že je to vhodný prostředek pro omezení počtu slov ve slovníku.

Architektura	CBOW				Skip-Gram			
Algoritmus	NS		HS		NS		HS	
Stemming	Ano	Ne	Ano	Ne	Ano	Ne	Ano	Ne
headlines	**0,678**	0,671	0,661	0,653	0,667	0,666	0,651	0,649
deft-news	0,689	0,688	**0,709**	0,707	0,661	0,666	0,670	0,683
deft-forum	0,441	0,405	0,464	0,436	0,407	0,409	0,421	0,432
tweet-news	0,686	**0,704**	0,677	0,703	0,657	0,687	0,670	0,674
OnWN	0,672	0,676	0,741	0,732	0,627	0,618	0,637	0,653
images	0,742	0,740	0,775	0,764	0,697	0,674	0,717	0,716

Tabulka 5.18: Korelace výsledků všech modelů za použití neváženého průměru

Architektura	CBOW				Skip-Gram			
Algoritmus	NS		HS		NS		HS	
Stemming	Ano	Ne	Ano	Ne	Ano	Ne	Ano	Ne
headlines	0,650	0,656	0,627	0,622	0,655	0,476	0,627	0,628
deft-news	0,665	0,693	0,659	0,678	0,632	0,670	0,644	0,669
deft-forum	0,457	0,473	0,445	0,450	0,450	**0,476**	0,437	0,453
tweet-news	0,609	0,664	0,570	0,625	0,590	0,646	0,547	0,601
OnWN	0,745	**0,799**	0,768	0,781	0,734	0,785	0,738	0,772
images	0,776	0,664	0,769	0,758	**0,781**	0,763	0,764	0,754

Tabulka 5.19: Korelace výsledků všech modelů za použití váženého průměru

	Nejlepší výsledek	Průměrný výsledek
headlines	0,7837	0,60436
deft-news	0,7850	0,63651
deft-forum	0,5305	0,36073
tweet-news	0,7921	0,61626
OnWN	0,8745	0,71629
images	0,8343	0,69424

Tabulka 5.20: Výsledky účastníku SemEval 2014 Task 10

Vážení slov pomocí Tf-idf se nejvíce pozitivně projevilo u sady *OnWN*. Ve všech případech můžeme u této sady sledovat zlepšení. V 75% případů se jedná o více než 10% nárůst. Naopak u sady *Tweet-news* vážení výsledky zhoršilo. Tabulky 5.21 a 5.22 popisují dvojice vět, u kterých došlo zavedením vážení k největší změně. Tento jev je dán povahou dat v jednotlivých datových sadách. U definic z WordNet (sada OnWN) se jedná o krátké a strohé definice. Tf-idf zde přiřadí vyšší váhy slovům, která mají pro význam věty největší hodnotu. U zpráv z twitteru nebo novinových titulků je situace opačná.

```
# váhy pro vstup: "2012: The Year of Extreme #Weather": #
    ↪ ClimateChange #Drought #NationalClimateAssessment #
    ↪ GlobalChangeResearch
[('year', 0.1534), ('the', 0.0152), ('drought', 0.4488),
('of', 0.0249), ('weather', 0.2813), ('climatechange', 0.7647),
```

```
('extreme', 0.3318)]

# váhy pro vstup: IHT Rendezvous: 2012: The Year of Extreme
    ↪ Weather
[('rendezvous', 0.5409), ('year', 0.1482), ('the', 0.0147),
('of', 0.0241), ('weather', 0.2719), ('iht', 0.7126),
('extreme', 0.3207)]
```

Pro tyto vstupy by intuitivně byla očekávaná slova s nejvyšší váhou *year*, *extreme*, *weather* a *drought*. Vážení sice správně zvolilo nízké váhy pro předložky, ale zároveň přiřadilo vysoké váhy slovům, která se vyskytují v malém počtu dokumentů v korpusu a zároveň nemají pro vstup relevantní významovou hodnotu. V tomto případě jde o slova *rendezvous*, *iht* nebo hashtag *climatechange*. Některá slova ze vstupu ve výčtu vah chybí z důvodu, že se nenachází ve slovníku.

```
# váhy pro vstup: the cause or instigation of something
[('or', 0.0734), ('the', 0.0207), ('instigation', 0.9047),
('of', 0.0339), ('something', 0.2485), ('cause', 0.3355)]

# váhy pro vstup: the end or completion of something
[('or', 0.1154), ('completion', 0.8200), ('the', 0.0325),
('end', 0.3972), ('of', 0.0532), ('something', 0.3905)]
```

Vážení v tomto případě přiřadilo váhy intuitivně. Obecnější slova mají nižší váhy než slova, která mají ve vstupu nejvyšší informační hodnotu jako *instigation* a *completion*. Stejná situace jako u výše uvedených dvojic vstupů je i u ostatních vstupů uvedených v tabulkách 5.21 a 5.22.

Věta 1	**Věta 2**	**Neváž. výsl.**	**Váž výsl.**	**Gold stand.**
the cause or instigation of something	the end or completion of something.	3,89	1,41	0,00
the state of being employed	the state of being excommunicated.	3,97	1,52	0,00
the state of being retained	the state of being angry.	4,09	1,81	0,00
the state of being retained	the state of being suspected.	4,11	1,89	1,00

Tabulka 5.21: Dvojice vět ze sady OnWN, kde se nejvíce projevilo vážení slov

5.3 Shrnutí

Podle testování na datové sadě WordSim353 je na určování podobnosti dvou slov nejvhodnější model CBOW s algoritmem Negative Sampling. Pro úlohy porovnávání podobnosti slov a hledání nejpodobnějších slov je vhodné využít stemming, který zmenší počet slov ve slovníku, odstraní duplicitu ve výsledcích a zlepší přesnost výsledků. Při vyhledávání nejpodobnějších slov je vhodné odfiltrovat málo vyskytující se slova. Empiricky byla určena

Věta 1	Věta 2	Neváž. výsl.	Váž výsl.	Gold stand.
"2012: The Year of Extreme #Weather": #ClimateChange #Drought #NationalClimateAssessment #GlobalChangeResearch	IHT Rendezvous: 2012: The Year of Extreme Weather	3,88	2,07	4,00
'Is there a worse way to live?' (cc:) #JIDF	'Is there a worse way to live than this?'	4,24	2,48	4,20
U.S. steps up involvement in Mallpp#pppj##p #pl###p###p	U.S. steps up involvement in Mali	4,09	1,81	4,20
Interesting article on the CNN homepage. "Mickleson has a point on taxes."	Mickelson has a point on taxes	3,31	2,33	4,40

Tabulka 5.22: Dvojice vět ze sady OnWN, kde se nejvíce projevilo vážení slov

vhodná prahová hodnota jako 10000 výskytů, po které zbude pro vyhodnocení přibližně 10% nejčastějších slov v korpusu. Vliv použitého modelu pro tuto úlohu je minimální. Vhodné je využít CBOW kvůli kratší době trénování modelu.

Při porovnávání sémantické podobnosti vět se jako nejuniverzálnější model jeví CBOW s využitím algoritmu Hierarchical Softmax. V této úloze vede stemming ve většině případů k nepatrně horším výsledkům. Využití váženého průměru záleží na typu porovnávaných dat. Lepších výsledků pomocí vážení lze dosáhnout u jednoduchých vět, jako jsou například definice z výkladových slovníků. Pro komplexnější texty se vážení nehodí a může způsobovat i zhoršení výsledků. Podle zveřejněných výsledků na webu SemEval 2014 by tento systém skončil někde kolem 20. místa z celkového počtu 38 účastníků.

Kapitola 6

Závěr

6.1 Dosažené výsledky

S použitím implementace Word2Vec v knihovně GenSim byl v jazyce Python vytvořen systém umožňující vytvořit nad korpusem CommonCrawl vektorový model pro vyhodnocování sémantické podobnosti slov. Systém je navržen tak, aby byla možná jeho úprava na trénování nad jiným korpusem pouze vytvořením nové třídy, která bude umět číst formát daného korpusu. V systému jsou skripty umožnující testování výsledného modelu na testovací sadě WordSim353. Dále je k dispozici skript na vyhodnocování podobnosti vět včetně jeho vyhodnocení na testovací sadě ze souteže SemEval 2014. Je také možné vyhledávat sémanticky nejpodobnější slova pro jedno nebo více vstupních slov. Součástí systému je i vlastní implementace Porter Stemmeru, který celý systém může volitelně využívat. V rámci práce byl otestován vliv parametrů trénování modelu na výsledky jednotlivých úloh.

6.2 Přínos práce

Práce přináší srovnání výsledků modelů natrénovaných s různými parametry (architektura, algoritmus,...) a zkoumá pro jaké úlohy jsou jednotlivé modely vhodné. Implementuje adaptér pro trénování modelu Word2Vec na korpusu CommonCrawl. Zároveň využívá natrénované modely k určování sémantické podobnosti vět.

6.3 Možnosti rozšíření

Modulární struktura systému umožňuje jeho další vylepšení. Nejpotřebnější je zřejmě zavedení dalších způsobů, jak určit vektor pro věty nebo dokumenty, například podle syntaktického stromu. Také je možné přidat podporu pro trénování na jiných korpusech nebo zavést podporu pro tokeny složené z více slov.

Literatura

[1] ACL Wiki - WordSimilarity-353 Test Collection (State of the art [online]). [cit. 2015-05-07] Dostupné z: http://www.aclweb.org/aclwiki/index.php?title=WordSimilarity-353_Test_Collection_(State_of_the_art).

[2] GenSim - Topic modelling for humans. Dostupné z: https://radimrehurek.com/gensim/.

[3] Word2Vec - Tool for computing continuous distributed representations of words. Dostupné z: https://code.google.com/p/word2vec/.

[4] Baroni, M.; Dinu, G.; Kruszewski, G.: Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) [online]*, Association for Computational Linguistics, 2014, s. 238–247, [cit. 2015-04-22] Dostupné z: http://aclweb.org/anthology/P14-1023.

[5] Goldberg, Y.; Levy, O.: word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method [online]. 2014, [cit. 2015-04-22] Dostupné z: http://arxiv.org/abs/1402.3722.

[6] Landauer, T. K.; Dumais, S.: Latent semantic analysis [online]. ročník 3, č. 11, 2008: str. 4356, revision #91420 [cit. 2015-04-22] Dostupné z: http://www.scholarpedia.org/article/Latent_semantic_analysis.

[7] Manning, C. D.; Raghavan, P.; Schütze, H.: *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008, ISBN 0521865719, 9780521865715.

[8] Mikolov, T.; Chen, K.; Corrado, G.; aj.: Efficient Estimation of Word Representations in Vector Space. 2013, [cit. 2015-04-22] Dostupné z: http://arxiv.org/abs/1301.3781.

[9] Mikolov, T.; Sutskever, I.; Chen, K.; aj.: Distributed Representations of Words and Phrases and their Compositionality [online]. 2013, [cit. 2015-04-22] Dostupné z: http://arxiv.org/abs/1310.4546.

[10] Mikolov, T.; tau Yih, W.; Zweig, G.: Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2013)*, Association for Computational Linguistics, 2013, [cit. 2015-04-22] Dostupné z: http://research.microsoft.com/apps/pubs/default.aspx?id=189726.

[11] Piits, L.: DISTRIBUTIONAL HYPOTHESIS: WORDS FOR 'HUMAN BEING' AND THEIR ESTONIAN COLLOCATES. *TRAMES: A Journal of the Humanities & Social Sciences*, ročník 17, č. 2, 2013: s. 141 – 158, ISSN 14060922.

[12] Rong, X.: word2vec Parameter Learning Explained [online]. 2014, [cit. 2015-04-22] Dostupné z: http://arxiv.org/abs/1411.2738.

[13] WIKIPEDIA: Bag-of-words model [online]. 2015, [cit. 2015-04-22] Dostupné z: http://en.wikipedia.org/wiki/Bag-of-words_model.

[14] WIKIPEDIA: Common Crawl [online]. 2015, [cit. 2015-05-02] Dostupné z: http://en.wikipedia.org/wiki/Common_Crawl.

[15] WIKIPEDIA: Lemmatisation [online]. 2015, [cit. 2015-04-22] Dostupné z: http://en.wikipedia.org/wiki/Lemmatisation.

[16] WIKIPEDIA: Stemming [online]. 2015, [cit. 2015-04-22] Dostupné z: http://cs.wikipedia.org/wiki/Stemming.

Příloha A

Obsah DVD

- Natrénovaný model včetně slovníku a statistiky stemmeru
- Skripty pro trénování modelu Word2Vec
- Skript pro vyhodnocení datové sady WordSim353
- Skript pro vyhodnocení datové sady SemEval 2014 Task 10
- Skript pro vyhledávání nejpodobnějších slov
- Technická zpráva v PDF
- Zdrojový text technické zprávy v LaTeX
- Návod na spouštění skriptů
- Plakát v PDF