VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

WORD2VEC MODELY S PŘIDANOU KONTEXTOVOU INFORMACÍ

WORD2VEC MODELS WITH ADDED CONTEXT INFORMATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **MARTIN ŠŮSTEK**
AUTHOR

VEDOUCÍ PRÁCE **Doc. Ing. FRANTIŠEK V. ZBOŘIL, CSc.**
SUPERVISOR

BRNO 2017

Zadání diplomové práce/20124/2016/xsuste07

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2016/2017

Zadání diplomové práce

Řešitel: **Šůstek Martin, Bc.**
Obor: Inteligentní systémy
Téma: **Word2vec modely s přidanou kontextovou informací**
Word2vec Models with Added Context Information
Kategorie: Umělá inteligence

Pokyny:

1. Nastudujte způsob vektorové reprezentace slov pomocí word2vec modelů.
2. Navrhněte možnosti integrace dostupných kontextových informací do architektury word2vec modelu.
3. Naimplementujte vybraný word2vec model s možností přidání kontextové informace.
4. Naučte implementovaný model pomocí dostupných datasetů.
5. Zjistěte vliv kontextové informace na výslednou vektorovou reprezentaci slov.
6. Zhodnoťte dosažené výsledky.

Literatura:

- Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient Estimation of Word Representations in Vector Space, 2013, https://arxiv.org/pdf/1301.3781.pdf
- Rong, X.: word2vec Parameter Learning Explained, 2016, https://arxiv.org/pdf/1411.2738v4.pdf
- Goldberg, Y., Levy, O.: word2vec Explained: Deriving Mikolov et al.s Negative-Sampling Word-Embedding Method, 2014, https://arxiv.org/pdf/1402.3722v1.pdf
- Svoboda, L., Brychcín, T.: New word analogy corpus for exploring embeddings of Czech words, 216, https://arxiv.org/pdf/1608.00789v1.pdf
- Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching Word Vectors with Subword Information, 2016, https://arxiv.org/pdf/1607.04606v1.pdf

Při obhajobě semestrální části projektu je požadováno:

- První dva body zadání.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Zbořil František V., doc. Ing., CSc.**, UITS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 24. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Tato práce se zabývá porozuměním word2vec modelů. Přestože tyto modely vznikly nedávno (2013), staly se velmi populárními. Učením těchto modelů lze obdržet vektorovou reprezentaci slov v N-dimenzionálním prostoru reálných čísel. Pomocí operací nad těmito vektory je možné určit sémantické vazby mezi slovy. Dále se práce snaží o rozšíření představených modelů za účelem jiné reprezentace slov. K tomuto účelu je navrženo využití obrazové informace. Taktéž je diskutována možnost použití konvolučních neuronových sítí ve spojitosti s poskytnutím odlišné kontextové informace.

Abstract

This thesis is concerned with the explanation of the word2vec models. Even though word2vec was introduced recently (2013), many researchers have already tried to extend, understand or at least use the model because it provides surprisingly rich semantic information. This information is encoded in N-dim vector representation and can be recall by performing some operations over the algebra. As an addition, I suggest a model modifications in order to obtain different word representation. To achieve that, I use public picture datasets. This thesis also includes parts dedicated to word2vec extension based on convolution neural network.

Klíčová slova

NLP, LM, AI, CNN, Zpracování přirozeného jazyka, Word2vec, Vnořování slov, Neuronové sítě, Sémantická podobnost, Softmax, Umělá inteligence, Konvoluční neuronové sítě

Keywords

NLP, LM, AI, CNN, Natural Language Processing, Language Modelling, Word2vec, Word Embeddings, Artificial Neural Networks, Semantic Similarity, Softmax, Artificial Inteligence, Convolution Neural Network

Citace

ŠŮSTEK, Martin. *Word2vec modely s přidanou kontextovou informací.* Brno, 2017. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Zbořil František.

Word2vec modely s přidanou kontextovou informací

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana doc. Ing. Františka V. Zbořila, CSc. Další informace mi poskytl Ing. Adam Kolář. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Šůstek
24. května 2017

Poděkování

Chtěl bych poděkovat vedoucímu práce doc. Ing. Františkovi V. Zbořilovi, CSc. a odbornému konzultantovi Ing. Adamovi Kolářovi za vedení a směrování této práce, poskytnutí informací a vstřícný přístup.

Obsah

Kapitola 1

Úvod

Word2vec je skupina modelů, které přitáhly v poslední době velkou pozornost a v návaznosti na ně vzniklo mnoho článků. Přestože byl word2vec představen až v roce 2013, celkový počet citací původního článku [14] již přesahuje číslo dva a půl tisíce[1]. Za autora této myšlenky je považován absolvent doktorského studia na FIT VUT v Brně Ing. Tomáš Mikolov Ph.D. a spolu s dalšími členy Google Brain týmu[2] publikovali zmíněný článek. Word2vec zahrnuje dva modely, *Continuous Bag-of-Words* (CBOW) a *Continuous Skip-gram* (Skip-gram, SG) a spadá do kategorie tzv. *word embedding* (v češtině se lze setkat s pojmem *vnořování slov*). Jedná o reprezentaci slov (obecně termů, protože se může jednat i o sousloví nebo fráze, jak je uvedeno v [15]) ve vektorovém prostoru reálných čísel, typicky o velikosti několika stovek dimenzí. Smyslem této reprezentace je zachování sémantických vazeb mezi slovy. Pro tyto účely se využívá distribuční hypotézy ([2]), která říká, že vyskytují-li se 2 slova ve **stejných kontextech** v dostatečně obsáhlém a obecném textu, pak to vypovídá o jejich vzájemném **vztahu**. Pokud se tedy slovo `kočka` vyskytuje často poblíž slova `pes` nebo kupříkladu existují věty „máme doma kočku“ a „máme doma psa“, bude mezi slovy určitá vazba, kterou je žádoucí zachytit pomocí reprezentujícího vektoru. Toho je implicitně využito ve většině modelů zpracovávajících velké množství souvislého textu, ze kterého se snaží vytěžit nějakou sémantickou informaci. Analyzovaný text se vždy vztahuje k některému přirozenému jazyku, a tak tyto modely spadají do kategorie **zpracování přirozeného jazyka** (NLP).

Tato práce popisuje word2vec modely, činí tak prostřednictvím vysvětlení jednotlivých fází učení v kapitole 2. Operace nad vektory (vektorová algebra) se staly velmi populárními a domnívám se, že právě tato skutečnost způsobila rozmach word2vec modelů. Zmíněné operace jsou přiblíženy v rámci stejné kapitoly, konkrétně v části 2.3.1. Kapitola 3 se věnuje možným úpravám word2vec modelu a její součást 3.3 je zaměřena na integraci jiných kontextových informací do architektur word2vec modelů. Krátká kapitola 4 popisuje použité datové sady, část 5 se pak zabývá implementací zvoleného word2vec modelu a vybraných rozšiřujících úprav. Kapitola 6 se zamýšlí nad použitím konvolučních neuronových sítí ve spojitosti s rozšiřováním kontextu.

[1] https://scholar.google.com/citations?view_op=view_citation&hl=cs&user=oBu8kMMAAAAJ&citation_for_view=oBu8kMMAAAAJ:UeHWp8X0CEIC

[2] https://research.google.com/pubs/BrainTeam.html

Kapitola 2

Proces trénování word2vec modelů

Na procesu trénování bude vysvětleno, jak model funguje, pro tyto účely bylo čerpáno ze zdrojů [2, 6, 14, 15, 18]. *Word2vec Parameter Learning Explained* ([18]) se snaží o nejpodrobnější popis a tvoří základ pro tuto kapitolu, příloha tohoto článku taktéž přibližuje učení neuronových sítí (BP[1] s využitím SGD[2]), jehož znalost se předpokládá. Pro samotný proces trénování (učení neuronových sítí) je zapotřebí korpusu. Korpus ([3]) můžeme neformálně chápat jako soubor textů (článků, knih, internetového obsahu, ...), pro naše účely je důležité, že text je souvislý (po sobě jdoucí slova sdílí kontext). Kvalita korpusu ovlivňuje možnosti modelů, protože model jazyka je vytvořen na základě dodaných textů, pokud trénovací sada reprezentována korpusem nereflektuje například vztah mezi *Prahou* a *Českou Republikou*, pak to nelze očekávat ani od vytvořeného modelu. Nad korpusem je možné provést operace jako sdružování více slov do jednoho termu nebo podvzorkování (subsampling), které zlepšuje kvalitu modelu a urychluje učení ([15]). Z výsledného textu je buď vybrána pouze podmnožina (podle [15] řádově 10^5 až 10^7) termů[3], které jsou obvykle nazývány jako *slovní zásoba* (*vocabulary*).

Učení probíhá tak, že ze souvislého textu postupně vybíráme termy (dále slova – w) a zároveň termy v jejich bezprostředním okolí, které označujeme jako kontext – c, velikost tohoto okolí je proměnlivá, přičemž parametrem modelu je maximální dosah okolí C a následně je pro každé slovo w zvolen aktuální dosah okolí R v rozmezí $< 1; C >$, které říká, že kontext slova w bude tvořen R předcházejícími slovy a R následujícími slovy. Uveďme příklad: `byl jednou jeden bohatý král`. Kontext slova `jeden` činí pro $C = 1$ [`jednou`, `bohatý`][4] a pro $C = 2$ [`byl`, `jednou`, `bohatý`, `král`]. Přestože z hlediska definice trénování modelů se jedná o učení s učitelem, po dokončení trénování požadujeme, aby bylo možné zjistit určité zákonitosti mezi slovy, jak bude zmíněno v části 2.3.1. Z tohoto hlediska můžeme hovořit o učení bez učitele, jelikož tyto zákonitosti nebyly modelu dodány (učení neprobíhalo s ohledem na ně).

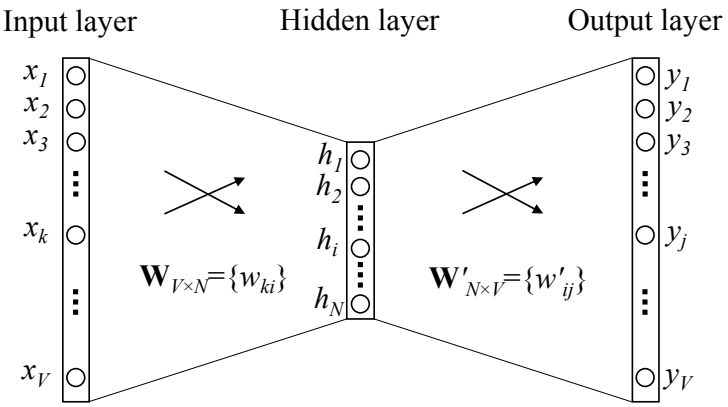
Jak již bylo zmíněno, word2vec zahrnuje 2 modely, **Skip-gram** a **CBOW**, v obou případech lze na model nahlížet jako na neuronovou síť, která má vstup, jednu skrytou vrstvu a výstupní vrstvu, což je znázorněno na Obrázku 2.1. Rozdíl mezi *SG* a *CBOW* je patrný z Obrázku 2.2, kde $w(t)$ značí slovo na pozici t. Pro běžně používané značení

[1] Backpropagation

[2] Stochastic gradient descent

[3] Součástí tohoto procesu může být odstranění číslovek, velkých písmen nebo mohou být aplikovány jiné metody předzpracování

[4] Zápis množiny pomocí hranatých závorek je uveden pro přehlednost



Obrázek 2.1: Jednoduché znázornění word2vec modelu (převzato z [18]).

(`vstupní vektor`, `výstupní vektor`) tvoří u *CBOW* trénovací (datovou) sadu množina (c, w) a u *SG* množina (w, c).

Tento odstavec bude věnován rozdílům mezi Obrázky 2.1 a 2.2 a jejich důkladnějšímu popisu. Skrytá vrstva ve word2vec modelu se někdy nazývá také **projekční**, protože neobsahuje žádnou nelinearitu (výstupy bázové a aktivační funkce jsou totožné). Pro podrobnější vysvětlení procesu učení bude dále předpokládáno, že počet vstupních i výstupních slov je **právě 1**, jelikož v takovém případě probíhá učení obou word2vec modelů shodně[5], rozdíly plynoucí z konkrétní architektury *(CBOW, SG)* budou zmíněny explicitně v části 2.1.3. Dále bude využito značení totožné s Obrázkem 2.1, vstupem sítě je V-dimenzionální binární *jeden z n* (*one-hot*) **vstupní vektor sítě x**, kde $|V|$[6] je velikost *slovní zásoby*. Formálně $x_{w_I} = 1, x_w = 0, \forall w \neq w_I$. Přestože se jedná o neuronové sítě a indexace pomocí pořadí neuronu a vrstvy by byla nasnadě, všechny práce, ze kterých čerpám, využívají popis pomocí matic, který se ukazuje jako velmi vhodný a výhodný. Váhy mezi vstupem a skrytou vrstvou tvoří matici W, kde každý řádek obsahuje vektor vah vztahující se k jednomu vstupnímu slovu (formálně $\mathbf{v}_w, \forall w \in V$), tento vektor má délku N, což značí počet neuronů ve skryté vrstvě. Váhy mezi skrytou a výstupní vrstvou jsou reprezentovány maticí W', která má počet sloupců V a každý z nich je vektorem (formálně $\mathbf{v'}_w, \forall w \in V$). Zmíněné vektory $\mathbf{v}_w$ a $\mathbf{v'}_w$ jsou velmi důležité a často jsou označovány jako **vstupní a výstupní vektory slova** w. Po dokončení fáze trénování právě tyto vektory (nejčastěji vstupní vektory slov $\mathbf{v}_w$) představují *word embedding*.

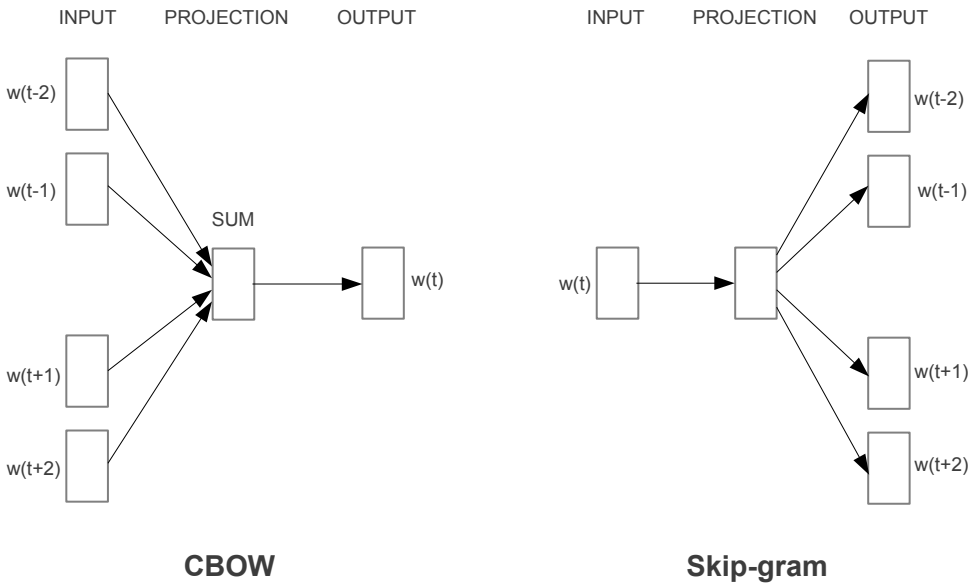
2.1 Jednotlivé fáze trénování

Projekční i výstupní vrstva mají lineární bázové funkce (pro skrytou vrstvu platí vztah 2.1), jelikož vstupní vektor sítě **x** zaobaluje pouze jedno (vstupní) slovo w_I (jeho index je ve vztahu značen jako k^*, váhy ostatních složek vektoru **x** jsou nulové), pak se na výstupu[7] skryté vrstvy objeví právě vektor $\mathbf{v}_w$. Pro bázovou funkci j-tého neuronu výstupní vrstvy (jedna dimenze výstupního vektoru y délky V) platí vztah 2.2 (ten lze chápat pouze jako

[5] Uvažujeme množinu testovacích případů (w_I, w_O)

[6] Někdy bude velikost slovní zásoby značena pouze jako V, jelikož z kontextu vyplývá, jestli se jedná samotnou slovní zásobu či pouze její velikost

[7] Připomínám, že aktivační funkce je identita



Obrázek 2.2: Rozdíl mezi CBOW a Skip-gramem (převzato z [14]).

skalární součin dvou vektorů), kde N je počet neuronů skryté vrstvy a notace odpovídá Obrázku 2.1 (vektor slova $\mathbf{v}_{w_j}$ je v tomto případě označen jako $\mathbf{v}_{wo_j}$, aby nebyly stejně označeny váhy i slova). Výpočtem získáme skóre vyjadřující míru, že se slovo w_j vyskytuje v kontextu slova w_I[8].

$$\begin{aligned} h_i = u_i^{\text{hid}} &= \sum_{k=1}^{V} w_{ki} \cdot x_k = w_{k^*,i} \cdot x_{k^*} = w_{k^*,i} \\ \mathbf{h} = \mathbf{u}^{\text{hid}} &= \mathbf{v}_{w_I} \end{aligned} \tag{2.1}$$

$$u_j^{\text{out}} = \sum_{i=1}^{N} w'_i \cdot h_i = {\mathbf{v}'_{wo_j}}^T \mathbf{v}_{wo_I} \tag{2.2}$$

Aby mělo trénování nějaký smysl, je nutné určit cíl trénování, který je pro Skip-gram[9] definován jako 2.3, kde θ značí parametry modelu (matice W a W'), θ^* hledané ideální parametry a D datovou sadu. Jelikož je nutné vyhodnotit podmíněnou pravděpodobnost[10] slova w_j na základě spočteného skóre u_j a prozatím se v modelu nevyskytuje nelinearita, použitím funkce **soft-max**[11] podle vztahu 2.4[12] vyřešíme oba problémy.

$$\theta^* = \arg\max_{\theta} \prod_{(w,c)\in D} p(c|w;\theta) \tag{2.3}$$

[8] Nebo naopak při použití CBOW
[9] Pro CBOW je pouze prohozen význam w a c
[10] Posterior
[11] Nebo také *softmax*
[12] Nyní již s implicitním zápisem parametru θ

$$y_j = p(w_j|w_I) = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u_{j'})} \tag{2.4}$$

2.1.1 Úprava vah mezi skrytou a výstupní vrstvou

Pro jeden testovací vzorek (w_I, w_O)[13] se snažíme maximalizovat podmíněnou pravděpodobnost (2.5), w_I značí vstupní slovo, w_O výstupní a j^* je index výstupního slova. Naše *ztrátová funkce* (*celková chyba, loss function*) je označena jako E a naším cílem je ji minimalizovat. Abychom se při počítání pravděpodobnosti vyhnuli násobení malých čísel, je vhodné využít logaritmu.

$$\begin{aligned} \max p(w_j|w_I) &= \max y_{j^*} \\ &= \max \log y_{j^*} \\ &= \max \log \frac{\exp(u_{j^*})}{\sum_{j'=1}^{V} \exp(u_{j'})} \\ &= u_{j^*} - \log \sum_{j'=1}^{V} \exp(u_{j'}) := -E \end{aligned} \tag{2.5}$$

Chyba ve vztahu k j-tému skóre se snaží, aby se v našem případě na výstupu objevil binární vektor *1 z n*, kde hodnota 1 přísluší výstupnímu slovu w_O, jak značí t_j (formálně $t_{j^*} = 1, t_j = 0, \forall t_j \neq t_{j^*}$). Ovšem nejsem schopen ověřit, jak lze na základě chyby E ze vztahu 2.5 získat vztah 2.6, druhý v pořadí tak považuji za axiomaticky definovaný bez ohledu na první. [14]

$$\frac{\partial E}{\partial u_j} = y_j - t_j := e_j \tag{2.6}$$

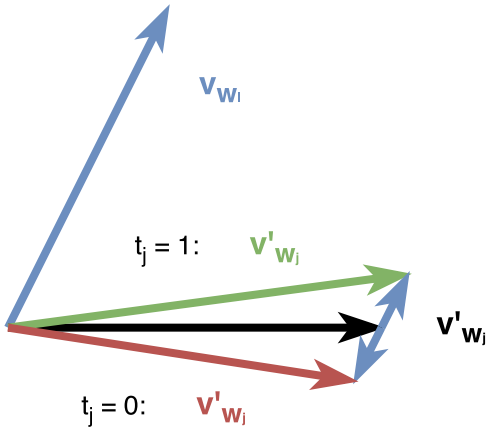
S využitím spočtené chyby lze odvodit gradient vah matice W' (2.7). Za použití stochastického gradientního sestupu (stochastic gradient descent, SGD) dojde k úpravě vah (2.8), jež lze chápat tak, že je ke všem vahám vedoucím od i-tého neuronu vstupní vrstvy připočtena část chyby tohoto neuronu. Daleko vhodnější vyjádření však naznačuje vztah 2.9, kdy chápeme úpravu vah matice W' po sloupcích namísto po řádcích. Upravujeme tak váhy vedoucí k výstupnímu slovu, což je možné ze zápisu interpretovat jako **přičtení části reprezentace vstupního slova $\mathbf{v}_{w_I}$ k reprezentaci výstupního slova $\mathbf{v'}_{w_O}$**. Pro ilustraci poslouží Obrázek 2.3, z něhož je patrné, že pro slovo w_O dojde ke zmenšení úhlu, který svírají vektory $\mathbf{v}_{w_I}$ a $\mathbf{v'}_{w_O}$. Pro ostatní vektory $\mathbf{v}_{w_j}$, kde $w_j \neq w_O$ dojde ke zvětšení úhlu. Velikost posunu závisí přímo úměrně na velikosti chyby e_j.

$$\frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial w'_{ij}} = e_j \cdot h_i \tag{2.7}$$

$$w'_{ij}{}^{(\text{new})} = w'_{ij}{}^{(\text{old})} - \eta \cdot e_j \cdot h_i \tag{2.8}$$

[13] (w, c) nebo (c, w)

[14] Požádal jsem autora článku [18] o podrobnější popis prostřednictvím elektronické pošty, avšak neobdržel jsem žádnou odpověď. Jelikož se model v této variantě nepoužívá (viz 2.2), není tento nedostatek závažný



Obrázek 2.3: Pohyb vektoru $\mathbf{v'}_{w_j}$, černou barvou je znázorněn původní vektor, zelenou respektive červenou barvou nový vektor, který vznikne přičtením respektive odečtením části vstupního vektoru $\mathbf{v}_{w_I}$ slova w_I.

$$\begin{aligned} \mathbf{v'}_{w_j}^{(\text{new})} &= \mathbf{v'}_{w_j}^{(\text{old})} - \eta \cdot e_j \cdot \mathbf{h} \\ &= \mathbf{v'}_{w_j}^{(\text{old})} - \eta \cdot e_j \cdot \mathbf{v}_{w_I} \end{aligned} \tag{2.9}$$

2.1.2 Úprava vah mezi vstupní a skrytou vrstvou

Vypočtená derivace pro každý neuron skryté vrstvy (2.10) může být chápána jako suma všech chyb, ke kterým došlo mezi skrytou a výstupní vrstvou, opět je ale vhodnější představit si tuto operaci pomocí vektorů tak, že se od vstupního vektoru v_{w_I} postupně odečítají (případně přičítají) výstupní vektory slov. Takto lze nahlížet na úpravu vah matice W pouze díky absenci nelinearity ve skryté vrstvě. V kontextu slova a jeho vstupních a výstupních reprezentací lze na základě parciální derivace celkové chyby E podle 2.11 určit, jakým způsobem budou upraveny váhy vstupního vektoru $\mathbf{v}_{w_I}$ (2.12).

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial h_i} = \sum_{j=1}^{V} e_j \cdot w'_{ij} := \text{EH}_i \tag{2.10}$$

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial h_i} \cdot \frac{\partial h_i}{\partial w_{ki}} = \text{EH}_i \cdot x_k \tag{2.11}$$

$$\begin{aligned} \mathbf{v}_{w_j}^{(\text{new})} &= \mathbf{v}_{w_j}^{(\text{old})} - \eta \text{EH}^T \\ &= \mathbf{v}_{w_j}^{(\text{old})} - \eta \sum_{j=1}^{V} e_j \cdot \mathbf{v'}_{w_j} \end{aligned} \tag{2.12}$$

Ve výsledku se jedná o podobný efekt, který je zobrazen obrázkem 2.3. Na rozdíl od prvního případu, kdy bylo pohybováno všemi výstupními vektory $\mathbf{v'}_{w_j}$ ve směru vstupního vektoru $\mathbf{v}_{w_I}$, nyní dochází k pohybu vstupního vektoru $\mathbf{v}_{w_I}$ postupně ve směru všech výstupních vektorů $\mathbf{v'}_{w_j}$. Tyto pohyby lze abstraktně chápat tak, že mezi vektory působí

vzájemné přitažlivé a odpudivé síly[15] ve smyslu zvětšování a zmenšování svíraného úhlu mezi vektory. Protože se slovo w málokdy vyskytuje ve svém vlastním kontextovém okolí c ($w \notin c$), často může nastat, že $\mathbf{v}_w$ a $\mathbf{v'}_w$ svírají úhel blízký 180°.

2.1.3 Rozdíly v učení pro jednotlivé modely

Zde se spokojíme s velmi jednoduchým a stručným vyjádřením o změnách popsaného modelu na základě zvolené architektury *CBOW* nebo *SG* a budeme uvažovat kontext c o kardinalitě větší než 1, dále budeme uvažovat, že $C = |c|$, [18] poskytuje detailnější popis včetně úpravy prezentovaných vztahů.

CBOW

Uvedené vztahy se liší pouze vstupním vektorem sítě $\mathbf{x}$, již nebude představovat binární vektor *1 z n*, ale uvažuje se buď průměr C vektorů vstupních slov z daného kontextu c nebo jejich suma (dělení konstantou nemá vliv na učení při pevně stanové velikosti kontextového okna), jak je vyjádřeno pomocí vztahu 2.13.

$$v_{w_I} = \frac{1}{C} \sum_{p=1}^{C} v_{c_p} \tag{2.13}$$

Je vhodné si v kontextu popsaných přitažlivých a odpudivých sil představit model *CBOW*. V původním případě se jednalo o „magnetickou sílu“ mezi vstupním vektorem jednoho slova a všemi ostatními výstupními slovy. Nyní je vstupní vektor $\mathbf{v}_{w_I}$ kompozicí několika vstupních slov a přitažlivé síly působí mezi tímto středem a všemi výstupními slovy. Při této operaci jsou vstupní vektory celého kontextu vůči sobě zafixovány (jejich vzájemná poloha je nezměněna).

Skip-gram

Skip-gram očekává kontext daného vstupního slova na výstupu své sítě. Dojde tím ke změně výpočtu chyby mezi skrytou a výstupní vrstvou, která musí brát v úvahu maximalizaci podmíněné pravděpodobnosti pro všechny výstupní vektory slov.

Nyní působí „magnetická síla“ mezi vstupním vektorem jednoho slova, ale na rozdíl od obecného popisu dojde k přitažení vstupního vektoru doprostřed[16] několika výstupních slov.

Zatímco pohyb vektorů u *SG* se liší od pohybu vektoru v obecně popsaném modelu[17] pouze tím, že vstupní vektor slova $\mathbf{v}_{w_I}$ je současně přitahován více výstupními vektory, u *CBOW* je změna razantnější, jelikož může v důsledku nevyváženosti[18] dojít k posunu, který nemusí reprezentovat celý kontext a vliv síly působící na výstupní vektory slov nezáleží na příspěvku jednotlivých slov (každé slovo přispívá stejně). V jednom kroku učení tak **může u *SG* dojít k posunu (výstupních) slov z kontextu vůči sobě**[19] na základě

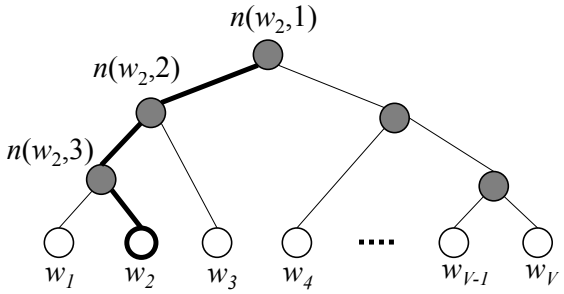
[15] Tento jev připomíná zobecněnou magnetickou (případně elektrickou) sílu, kdy by se některé magnety přitahovaly a některé odpuzuvaly

[16] Ve smyslu symbolického středu, každý vektor $\mathbf{v'}_w$ přispívá na základě své chyby

[17] Ten se někdy označuje jako *bigram*, protože obsahuje dvě slova, jedno vstupní a jedno výstupní

[18] Průměr vstupních vektorů slov $\mathbf{v}_{w_I}$ v kontextu c není od všech slov stejně daleko, ale zohledňují se se stejnou měrou

[19] Protože chyby u jednotlivých slov nebudou stejné



Obrázek 2.4: Znázornění binární stromové struktury určené pro výpočet hierarchického softmaxu, listy (bílá barva) představují slova ze slovní zásoby. V každém vnitřním uzlu (tmavá barva) je obsažen vektor. Cesta vedoucí od kořene stromu ke slovu w_2 je zvýrazněna. Její délka $L(w_2) = 4$. $n(w, j)$ označuje j-tý uzel na cestě od kořene ke slovu w. Pro tyto účely je délkou cesty míněn počet uzlů na cestě. Obrázek byl převzat z [18].

vstupního slova, zatímco u *CBOW* k takovému pohybu dojít nemůže[20]. Zamyšlení ohledně „magnetických sil“ není striktně založeno na matematických základech, ale pouze nabízí alternativní pohled na word2vec modely prostřednictvím paralely mezi těmito modely a magnetickou silou, která může být pro pochopení fungování modelů intuitivnější.

2.2 Optimalizace výpočtu

Při aktualizaci vah mezi vstupní a výstupní maticí je nutné pro výpočet skóre u_{j^*} získat celkový součet všech u_j. Ten je použit jako normalizační koeficient při výpočtu pravděpodobnosti výskytu slova y_j (funkce soft-max, 2.4), jež slouží k výpočtu chyby e_j (2.6). Pro jeden krok učení algoritmu je tak potřeba zjistit rozložení pravděpodobnosti (výstupní vektor $\mathbf{y}$), což je velmi drahá operace, uvážíme-li, že slovní zásoba V obsahuje řádově desítky až stovky tisíc slov. Z tohoto důvodu se téměř vždy využívá buď *hierarchického soft-maxu* (HS, podle [16]) popsaného v části 2.2.1 nebo postupu nazývaného *negative sampling* (NS), který bude prezentován v sekci 2.2.2. Hierarchický softmax neprodukuje výstupní vektory slov $\mathbf{v'}_w$, naproti tomu celková chyba E pro negative sampling není striktně odvozena od posteriorní pravděpodobnosti $p(w_O|w_I)$.

2.2.1 Hierarchický soft-max

Hierarchický softmax je založen na binárním stromu[21], o jehož struktuře vypovídá Obrázek 2.4. Tato reprezentace neobsahuje žádné výstupní vektory slov $\mathbf{v'}_{w_j}$, ale pro výpočet pravděpodobnosti y_j slouží vektory vyskytující se na cestě od kořene stromu ke slovu w_j. Pravděpodobnost, že slovo w je výstupním slovem, je definována podle vztahu 2.14, kde σ značí sigmoidální funkci, lch($node$) je levý syn uzlu $node$, $\mathbf{v'}_{node}$ je vektorová reprezentace ukrytá v uzlu $node$ a $[\![x]\!]$ je speciální funkce podle vztahu 2.15.

$$p(w = w_O) = \prod_{j=1}^{L(w)-1} \sigma\Big([\![n(w, j+1) = \text{lch}(n(w, j))]\!] \cdot \mathbf{v'}_{n(w,j)}{}^T \mathbf{h}\Big) \tag{2.14}$$

[20] Úprava vah všech vstupních vektorů je stejná

[21] Ideálně Huffmanův strom pro maximální zrychlení, ale klidně postačí výškově vyvážený binární strom

$$[\![x]\!] = \begin{cases} +1 & \text{pokud je } x \text{ pravdivé} \\ -1 & \text{jinak} \end{cases} \tag{2.15}$$

Pro určení posteriorní pravděpodobnosti slova w je potřeba projít sekvenci uzlů na cestě od kořene ke slovu w. Pro každý nelistový uzel se spočte skalární součin mezi vektorem daného uzlu $\mathbf{v'}_{node}$ a vstupním vektorem $\mathbf{v}_{w_I}$ ($\mathbf{v}_{w_I} = \mathbf{h}$). Je-li skalární součin 0, pak je pravděpodobnost pro oba syny stejná, kladná hodnota zvyšuje pravděpodobnost levému synovi, záporná pravému. Účelem jednoho kroku trénování je zvýšení pravděpodobnosti $p(w|w_I)$ pomocí úprav vektorů na cestě k němu. Pokračuje-li cesta vpravo, dojde ke snížení skalárního součinu, jinak dojde k jeho zvýšení. Podrobnější popis včetně příkladu je uveden v [18], než přejdeme k samotnému učení, neuškodí připomenout definici sigmoidy a některé její dobré vlastnosti pomocí rovností 2.16.

$$\begin{aligned} \sigma(u) &= \frac{1}{1+e^{-u}} \\ \sigma(u) &= 1 - \sigma(-u) \\ \frac{\partial \sigma(u)}{\partial u} &= \sigma(u)\sigma(-u) \end{aligned} \tag{2.16}$$

Celkovou chybu E jsme již v popsaných modelech stanovili, dosazením podmíněné pravděpodobnosti a za použití zkráceného zápisu (2.17) získáme vztah 2.18.

$$[\![\cdot]\!] := [\![n(w, j+1) = \text{lch}(n(w, j))]\!] \tag{2.17}$$

$$E = -\log p(w = w_O|w_I) = -\sum_{j=1}^{L(w)-1} \log \sigma([\![\cdot]\!]\mathbf{v'}_{n(w,j)}{}^T\mathbf{h}) \tag{2.18}$$

Derivací celkové chyby podle součinu vektorů $\mathbf{v'}_{n(w,j)}$ (zkráceně $\mathbf{v'}_j$) a $\mathbf{h}$ získáme vztah 2.19, kde $t_j = 1$ právě tehdy, když $[\![\cdot]\!] = 1$, jinak $t_j = 0$. Jelikož nás zajímá, jak upravit váhy pro uzly na cestě, získáme parciální derivaci podle vztahu 2.20 a následně za použití *SGD* celkový vztah 2.21 pro úpravu vah.

$$\begin{aligned} \frac{\partial E}{\partial \mathbf{v'}_j{}^T\mathbf{h}} &= -\Big(\frac{1}{\sigma([\![\cdot]\!]\mathbf{v'}_j{}^T\mathbf{h})} \cdot \sigma([\![\cdot]\!]\mathbf{v'}_j{}^T\mathbf{h}) \cdot \sigma(-[\![\cdot]\!]\mathbf{v'}_j{}^T\mathbf{h})\Big)[\![\cdot]\!] \\ &= -\Big(\sigma(-[\![\cdot]\!]\mathbf{v'}_j{}^T\mathbf{h})\Big)[\![\cdot]\!] \\ &= \Big(\sigma([\![\cdot]\!]\mathbf{v'}_j{}^T\mathbf{h}) - 1\Big)[\![\cdot]\!] \\ &= \begin{cases} \sigma(\mathbf{v'}_j{}^T\mathbf{h}) - 1 & ([\![\cdot]\!] = 1) \\ \sigma(\mathbf{v'}_j{}^T\mathbf{h}) & ([\![\cdot]\!] = -1) \end{cases} \\ &= \sigma(\mathbf{v'}_j{}^T\mathbf{h}) - t_j \end{aligned} \tag{2.19}$$

$$\frac{\partial E}{\partial \mathbf{v'}_j} = \frac{\partial E}{\partial \mathbf{v'}_j{}^T\mathbf{h}} \cdot \frac{\partial \mathbf{v'}_j{}^T\mathbf{h}}{\partial \mathbf{v'}_j} = \Big(\sigma(\mathbf{v'}_j{}^T\mathbf{h}) - t_j\Big) \cdot \mathbf{h} \tag{2.20}$$

$$\mathbf{v'}_j{}^{(\text{new})} = \mathbf{v'}_j{}^{(\text{old})} - \eta\Big(\sigma(\mathbf{v'}_j{}^T\mathbf{h}) - t_j\Big) \cdot \mathbf{h} \tag{2.21}$$

Parciální derivace celkové chyby ve skryté vrstvě se získá analogicky (jako suma přes všechny uzly na cestě ke slovu). Výpočet se tak celkově zrychlí, jelikož není nutné počítat skalární

součin a chybu pro všechna slova, ale vzhledem k výšce stromu pouze pro $\log V$ slov, což výrazně urychluje výpočet. Počet výstupních vektorů[22] se díky stromu změní z V na $V-1$.

2.2.2 Negative sampling

Myšlenka dalšího způsobu urychlení výpočtu spočívá v aktualizaci pouze vybraných výstupních vektorů slov $\mathbf{v'}_w$. Pro výpočet celkové chyby E se použijí všechny chyby e_j, které se vztahují ke slovům očekávaným na výstupu[23]. Ze zbylých negativních[24] příkladů (slov) se vybere pouze určitá podmnožina (vzorek) slov, o čemž má vypovídat název metody (negativní vzorkování).

Jak již bylo zmíněno v části 2.2, pro převod skóre u_j na výpočet posteriorní pravděpodobnosti $p(w_O|w_I)$ je nutné vyhodnotit skóre všech slov ze slovní zásoby (jmenovatel softmaxu). Abychom určili tuto „pravděpodobnost“[25], použijeme vztah 2.22.

$$p(w_O|w_I) = \sigma(\mathbf{v'}_{w_O}{}^T \mathbf{v}_{w_I}) \tag{2.22}$$

Takto definovaný model má triviální řešení[26], např. $\forall w_1, w_2 \in V, \mathbf{v'}_{w_1} = \mathbf{v}_{w_2}, |\mathbf{v}_{w_1}| = 10$, kde $|\mathbf{v}|$ značí velikost vektoru $\mathbf{v}$. Tím pádem budou všechny vektory stejné a jejich pravděpodobnost bude prakticky rovna 1. Abychom tomu zabránili, přiblížíme se softmaxu tím, že vybereme právě K jiných náhodných příkladů, přičemž předpokládáme, že všechny tyto příklady jsou negativní (požadujeme, aby jejich posteriorní pravděpodobnost byla rovna nule). Pozitivní příklady v tomto případě působí jako přitažlivá síla, která táhne slova k sobě, zatímco negativní příklady kompenzují přibližování nesouvisejících slov. Rozložení pravděpodobnosti výběru slova w jako negativní vzorek $P_n(w)$ (*noise distribution*) není nijak omezeno, jedná se tak o parametr modelu. Nejlepších výsledků[27] lze podle [15] dosáhnout na základě vztahu 2.23, kde $\text{count}(w)$ značí počet slov w v korpusu, $P(w)$ tak značí pravděpodobnost výskytu slova w. Vztah 2.24 určuje celkovou chybu, kde W_{neg} označuje množinu K náhodně vybraných slov na základě $P_n(w)$.

$$P_n(w) = P(w)^{3/4}, \text{ kde } P(w) = \frac{\text{count}(w)}{\sum_{w' \in V} \text{count}(w')} \tag{2.23}$$

$$E = -\log \sigma(\mathbf{v'}_{w_O}{}^T \mathbf{h}) - \sum_{w_j \in W_{\text{neg}}} \log \sigma(-\mathbf{v'}_{w_j}{}^T \mathbf{h}) \tag{2.24}$$

Přestože takto definovaná chyba není přímo matematicky podložena, pro naše účely je dostačující, jelikož hlavním smyslem je zisk použitelné reprezentace slov (*word embedding*). Pro parciální derivace celkové chyby a úpravu vah W' dostáváme vztahy 2.25, 2.26 a 2.27, kde $t_j = 1$, právě když se jedná o pozitivní vzorek, jinak 0. Tyto nápadně připomínají 2.19, 2.20 a 2.21 vztahující se k hierarchickému softmaxu.

$$\begin{aligned} \frac{\partial E}{\partial \mathbf{v'}_{w_j}{}^T \mathbf{h}} &= \begin{cases} \sigma(\mathbf{v'}_{w_j}{}^T \mathbf{h}) - 1 & \text{pokud } w_j = w_O \\ \sigma(\mathbf{v'}_{w_j}{}^T \mathbf{h}) & \text{pokud } w_j \in W_{\text{neg}} \end{cases} \\ &= \sigma(\mathbf{v'}_{w_j}{}^T \mathbf{h}) - t_j \end{aligned} \tag{2.25}$$

[22] V tomto případě se nevztahují ke slovům ale uzlům

[23] Pro CBOW je výstupní slovo w_O pouze jedno

[24] Snažíme se minimalizovat jejich skóre, aby se tím zvýšilo skóre očekávaných výstupních slov

[25] Nejedná se o pravděpodobnost v pravém slova smyslu, protože součet všech takto získaných „pravděpodobností“ nebude roven 1

[26] Řešením modelu je nalezení parametrů θ, tedy vah W a W', které vyhovují 2.3

[27] Ověřeno empiricky

$$\frac{\partial E}{\partial \mathbf{v'}_{w_j}} = \frac{\partial E}{\partial \mathbf{v'}_{w_j}{}^T\mathbf{h}} \cdot \frac{\partial \mathbf{v'}_{w_j}{}^T\mathbf{h}}{\partial \mathbf{v'}_{w_j}} = \Big(\sigma(\mathbf{v'}_{w_j}{}^T\mathbf{h}) - t_j\Big) \cdot \mathbf{h} \tag{2.26}$$

$$\mathbf{v'}_{w_j}{}^{(\text{new})} = \mathbf{v'}_{w_j}{}^{(\text{old})} - \eta\Big(\sigma(\mathbf{v'}_{w_j}{}^T\mathbf{h}) - t_j\Big) \cdot \mathbf{h} \tag{2.27}$$

Úpravy vah mezi vstupní a skrytou vrstvou probíhají analogicky k prezentovanému obecnému popisu. Přestože je nelinearita v původním modelu bez optimalizace tvořena pomocí softmaxu, při optimalizacích je využito sigmoidální funkce[28] $\sigma(u)$ (2.16).

2.3 Výsledek trénování

Po dokončení procesu trénování[29] můžeme využít váhy W (případně W') a jim příslušející vektory slov pro *word embedding*. Vektory pro každé slovo bývají typicky normalizovány tak, aby odpovídaly jednotkovému vektoru. Vzájemná poloha těchto vektorů uchovává sémantickou informaci na základě zmíněné distribuční hypotézy[30]. Dále (2.3.1) bude demonstrováno, jakým způsobem mohou být tyto vektory interpretovány a v části 2.3.2 bude poukázáno na souvislost s faktorizací PMI matice.

2.3.1 Jazykové zákonitosti (linguistic regularities)

Jak ukazuje Obrázek 2.5, z promítnutí vybraných států a jejich hlavních měst do 2-D prostoru je patrný jejich vztah (vysvětlení poskytuje popisek Obrázku 2.5). Platí tak kupříkladu, že $vec(\texttt{Spain})$[31] - $vec(\texttt{Madrid})$ + $vec(\texttt{Lisbon})$ je velmi blízko $vec(\texttt{Portugal})$. Příkladem jiné operace může být $vec(\texttt{Czech})$ + $vec(\texttt{currency})$, což má blízko k $vec(\texttt{koruna})$ a k $vec(\texttt{Czech crown})$, jak uvádí [15]. Blízkostí je v tomto smyslu typicky[32] míněn úhel, který spolu dva vektory svírají[33]. Nejjednodušším způsobem, jak určit úhel mezi dvěma vektory, je kosinová vzdálenost (2.28). Nejdříve určíme $\mathbf{X} = vec(\texttt{Spain}) - vec(\texttt{Madrid}) + vec(\texttt{Lisbon})$ a následně zjistíme, kterému slovu se $\mathbf{X}$ nejvíce podobá podle vztahu 2.29, přičemž z výsledků jsou odstraněny slova tvořící vektor $\mathbf{X}$. V mnoha případech obdržíme právě očekávané slovo[34] (zde $\texttt{Portugal}$).

$$\text{similarity}(\mathbf{A}, \mathbf{B}) = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} \tag{2.28}$$

$$\begin{aligned} \text{closest}(\mathbf{X}) &= \arg\max_{w \in V} \Big(\text{similarity}(\mathbf{v}_w, \mathbf{X}) \Big) \\ &= \arg\max_{w \in V} \frac{\mathbf{v}_w \cdot \mathbf{X}}{\|\mathbf{v}_w\|\|\mathbf{X}\|} \end{aligned} \tag{2.29}$$

[28] Softmax může být stále použit pro určení rozložení pravděpodobnosti výstupních slov, ale není použit ve fázi učení

[29] Není určený pevný limit iterací přes slovní zásobu, musí dojít k rozhodnutí o ukončení trénování

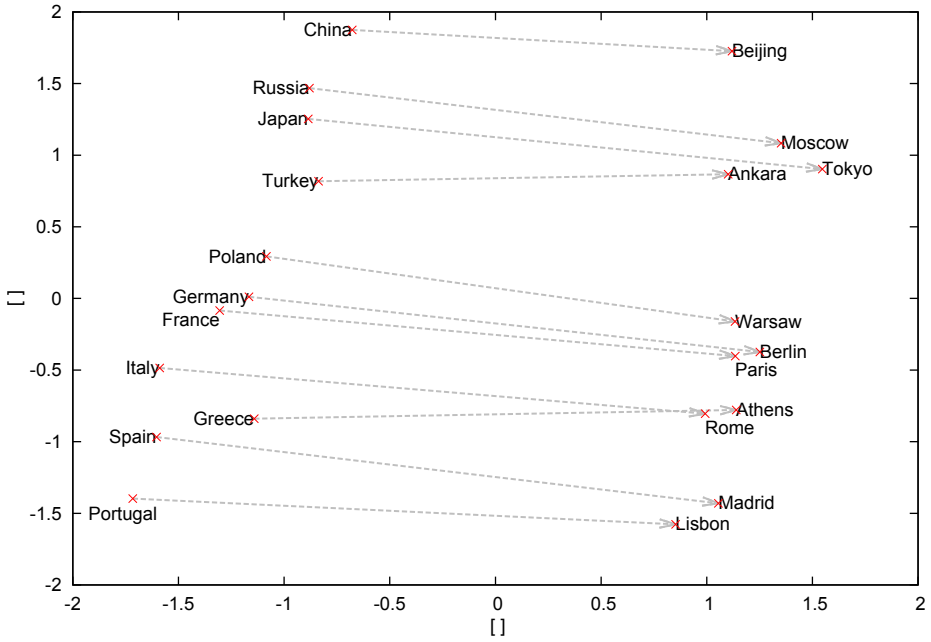
[30] Tedy blízkosti slov, model nemá žádnou informaci o významu slova

[31] Pro lepší čitelnost je zápis $\mathbf{v}_{\text{Spain}}$ nahrazen zápisem $vec(\texttt{Spain})$

[32] Jiné způsoby měření podobnosti lze nalézt v [11]

[33] To odpovídá pohybu vektorů při fázi trénování, kdy se úhel podobných slov zmenšoval, zatímco úhel ostatních slov zvětšoval, zde se však ovlivňovaly vstupní a výstupní vektory slov

[34] Nejedná se tak pouze o podobné slovo, ale ze všech slov **nejpodobnější**, dá se tedy mluvit o jisté umělé inteligenci, která zachycuje vztah některých slov



Obrázek 2.5: Znázornění vztahů mezi státem a jeho hlavním městem. Vypočítané vektory (od státu k jeho hlavnímu městu) mezi sebou navzájem svírají malý úhel. Lze si také povšimnout, že sousední (obecně sémanticky blízké) státy k sobě mají blíž. Reprezentace byla získána promítnutím vektorů reprezentující slova do dvoudimenzionálního prostoru pomocí PCA. Obrázek byl převzat z [15].

2.3.2 Souvislost s PMI

Bodově vzájemná informace (často označována zkratkou *PMI*[35]) se vždy vztahuje ke dvěma náhodným proměnným (x a y)[36] a je definována vztahem 2.30. Dá se chápat jako míra entropie plynoucí ze „sounáležitosti“ x a y.

Entropii lze intuitivně vysvětlit na Huffmanově kódování, to se snaží zakódovat časté události krátkým řetězcem a ojedinělé dlouhým. To je důsledkem snahy o co nejkratší celkový text, ze kterého by se zpětně dalo jednoznačně určit, ke kterým událostem došlo. O míře entropie (neurčitosti) události pak vypovídá neobvyklost události a její velikost je rovna[37] délce slova v Huffmanově kódování.

„Sounáležitostí“ rozumíme míru, která určuje, jak moc lze z přítomnosti[38] události x usuzovat o přítomnosti události y. Představme si, že známe pravděpodobnost události y a označme ji jako $P(y)$, nyní vyberme podmnožinu všech událostí a to tak, že obsahuje pouze případy, ve kterých se objevila událost x. V této podmnožině zjistíme pravděpodobnost výskytu události y, tedy $P(y|x)$. Pokud $P(y|x) > P(y)$, pak výskyt události x do jisté míry vypovídá o události y a dá se mluvit o „sounáležitosti“.

$$\mathrm{pmi}(x;y) \equiv \log \frac{p(x,y)}{p(x)p(y)} = \log \frac{p(x|y)}{p(x)} = \log \frac{p(y|x)}{p(y)} \tag{2.30}$$

[35] Pointwise mutual information
[36] $\mathrm{pmi}(x,y) = \mathrm{pmi}(y,x)$, proto se ve značení používá středník
[37] Bráno nezaokrouhleně jako reálné číslo
[38] Zde uvažujeme pouze přítomnost události, neuvažujeme, co lze vyvozovat z její nepřítomnosti

Největší hodnoty PMI (2.30) dosahují slova, která se vždy vyskytují společně a navíc nejsou v celém korpusu frekventovaná. Příkladem[39] dvojice slov, která se často objevují vedle sebe, ale zřídkakdy odděleně, může být `Puerto Rico` nebo `Hong Kong`, opakem pak dvojice {`of`, `and`}.

Podle zdroje [10] word2vec model (konkrétně SG s využítím negative sampling) implicitně faktorizuje matici PMI (která je posunuta o konstantu) pomocí matic vah W a W'. Vynásobením matic W a W' a přičtením konstanty bychom tak měli obdržet právě PMI matici. V ní má každé políčko hodnotu $\text{pmi}(w_r, w_c)$, kde w_r je slovo v řádku a w_c slovo ve sloupci. Zajímavé hodnoty se mohou objevit na diagonále, jelikož předpokládáme, že většina slov se nevyskytuje ve svém vlastním kontextu a očekáváme zde čísla blížící se $-\infty$, což odpovídá zmíněnému předpokladu o vzdálenosti blízké 180°.

[39] Podle https://en.wikipedia.org/wiki/Pointwise_mutual_information

Kapitola 3

Možné úpravy word2vec

V této kapitole budou navrženy úpravy modelu, jež by mohly mít pozitivní dopad na jeho vlastnosti. Dále budou diskutovány možnosti integrace jiných než textových informací do architektury word2vec (část 3.3).

3.1 Slova opačného významu

Přestože výsledná reprezentace vykazuje velmi dobré vlastnosti, jak bylo zmíněno v části 2.3.1, pro některé úlohy nemusí být ideální. Kosinová vzdálenost se u velmi podobných objektů blíží hodnotě 1, při ortogonalitě vektorů hodnotě 0 a v případě slov, která se spolu nikdy nevyskytují současně, -1.

Model dokáže za použití kosinové vzdálenosti určit i protiklad slova w, nesnaží se však najít antonymum, ale teoreticky slovo, které se vyskytuje zřídkakdy současně se slovem w. Dá se však očekávat, že např. za použití NS[1] nemusí být dostatek protipříkladů, což ale nemusí být na škodu, protože samotná informace není dále nijak použita. Pro přirozené jazyky je typické, že k některým slovům existují antonyma.

3.1.1 Vyhledávání obrázků

Pokud budeme vnímat modely v kontextu předpovědi slova při úloze rozpoznání řeči, pak absence jednoduchého prostředku pro získání antonyma neovlivňuje kvalitu výsledku. Zaměříme-li se však na úlohu vyhledání podobných obrázků na základě kontextu, tato skutečnost může způsobit velké obtíže. Pokud si někdo přeje vyhledat „velké vozidlo“, asi nebude spokojen s výsledkem, který mu zobrazí „malé vozidlo“, přestože ve vytvořeném *word embeddingu* si jsou tyto výsledky podobné na základě distribuční hypotézy. Naopak neradi bychom přišli o informaci o synonymech, jelikož „obrovský automobil“ bychom ve výsledcích rádi viděli. Tento problém se nevztahuje pouze ke slovům opačného významu, ale také na kategorické atributy (červená, žlutá, modrá, . . .), numerické hodnoty a obecně slova, která chápeme jako výběr jedné z několika možností („Půjdeme do kina nebo do divadla?“).

Abychom se pokusili odstranit nedostatek s obrázky, je v části 3.1.2 navrženo potenciální řešení, pomocí něhož bychom požadovali, aby se ve výsledcích vyhledávání nezobrazoval protiklad. K tomuto problému se taktéž vztahuje sekce 3.3, jelikož přidáním jiné kontextové informace může vzniknout odlišná reprezentace, která by tyto vztahy mohla lépe zachycovat.

[1] negative sampling

3.1.2 Zobecňování a antonyma

V tomto případě se nejedná doslova o úpravu modelu, ale o rozšíření interpretace modelu. Jelikož lze v modelu provádět operace na základě jazykových zákonitostí (2.3.1), vystává otázka, zda dokáže model také zobecňovat. Lze předpokládat, že např. pro množinu {`red`, `blue`, `black`, `orange`} existuje způsob, jak obdržet slovo `color` případně `colors`.

Dále se domnívám, že model by mohl do jisté míry zachycovat zápor ve smyslu, v jakém je obecně chápán[2]. K tomuto účelu by mohlo posloužit slovo `or`. Přestože v kontextu slova `or` se nebudou vyskytovat pouze antonyma, předpokládám, že by se mohly zlepšit výsledky ve smyslu úlohy představené v části 3.1.1. Očekávám totiž, že synonyma se nebudou často vyskytovat v přítomnosti slova `or`, naopak kategorická[3] slova budou, což vyhovuje zmíněným účelům.

Pro účely záporu by mohla postačit jednoduchá operace sčítání[4] vektorů a vyhledání nejbližšího $\text{closest}(vec(w)+vec(\texttt{or}))$ slova podle vztahu 2.29. Pokud by $vec(\texttt{or})$ nedostatečně zachycoval tyto zákonitosti, může být „vektor protikladů“ vypočten na základě několika příkladů významově opačných slov.

Úkol získání obecné reprezentace skupiny objektů nemusí být možný pomocí obyčejného sečtení všech vektorů, v takovém případě se dá ještě uvažovat o nějakém způsobu normalizace jako o metodě předzpracování. Jinou možností může být pokus o nalezení shluků různými metodami a jejich zpětná interpretace[5].

3.2 Změna ve fázi učení

Tato část převážně zkoumá vztah mezi vektory $\mathbf{v}_w$ a $\mathbf{v'}_w$ a představuje několik možných úprav včetně myšlenky propojení architektur SG a CBOW. Některé návrhy implicitně předpokládají, že slovo se nemůže vyskytovat ve svém vlastním kontextu, na případné výjimky není brán zřetel, přestože tím může model ztratit na obecnosti.

3.2.1 Explicitní vazba mezi vektory jednoho slova

Přestože je při fázi učení požadavek, aby $\mathbf{v}_w$ svíral s $\mathbf{v'}_w$ úhel blízký 180 stupňům, myšlenka vlivu úhlu mezi těmito vektory na model je přinejmenším zajímavá. Navrhuji tak zkoumat rozdíly při požadavcích na:

protilehlost vektorů
: ke změně oproti prezentovaným řešením dochází pouze u NS, kdy budeme explicitně požadovat, aby se mezi K negativními vzorky pokaždé objevilo slovo w_I[6]

rovnoběžnost vektorů
: požadavek na rovnoběžnost by měl způsobit, že matice W a W' budou velmi podobné, toho bude dosaženo definicí nového kontextu c^{new} pomocí původního slova a kontextu jako $c^{\text{new}} = c \cup \{w\}$[7]

nezávislost vektorů
: vektory se nesmí přímo ovlivňovat, při učení bude pro slovo w_j axiomaticky stanovena chyba ve výstupní vrstvě e_j jako 0 ($\forall w_j, 0 < j < V : e_j = 0$)

[2] Antonymum
[3] Případně je možné zkoumat vztah ke slovu `and`
[4] Případně odečítání
[5] Určení, jestli dávají požadované výsledky
[6] Případně všechna slova na vstupu pro CBOW
[7] Na diagonále matice PMI by se měla objevit entropie odpovídajícího slova podle vztahu 2.30

S využitím těchto znalostí by mohla být případně upravena i fáze učení, aby po určitém počtu iterací[8] došlo k přeuspořádání modelu[9] pomocí:

- normalizace vektorů – převod na jednotkové, případně zmenšení délek[10]
- vstupních a výstupních vektorů stejného slova[11] – k sobě nebo od sebe

Případně po dokončení fáze učení pokus o pohyb všech slov tak, aby jejich vektory svíraly úhel téměř 180°.

3.2.2 Kombinace SG a CBOW

Datová sada pro tuto úlohu klasifikace předpokládá množinu (c, w) případně (w, c), můžeme však navrhnout kombinaci obou, přičemž bychom ve fázi trénování střídavě přikládali (c, w) a (w, c)[12].

Jiný způsob spojení obou přístupů bude znázorněn na příkladu korpusu `byl jednou jeden bohatý král`. Standardně, jak bylo zmíněno v kapitole 2, vytvoříme pro $R = 1$ datovou sadu jako množinu 3.1 respektive jako množinu 3.2. Můžeme však předpokládat, že w nebude slovo, ale bude značit množinu o např. 2 slovech, datovou sadu by pak tvořil zápis 3.3 respektive 3.4.

$$\{(\texttt{jednou}, [\texttt{byl}, \texttt{jeden}]), (\texttt{jeden}, [\texttt{jednou}, \texttt{bohatý}]), (\texttt{bohatý}, [\texttt{jeden}, \texttt{král}])\} \tag{3.1}$$

$$\{([\texttt{byl}, \texttt{jeden}], \texttt{jednou}), ([\texttt{jednou}, \texttt{bohatý}], \texttt{jeden}), ([\texttt{jeden}, \texttt{král}], \texttt{bohatý})\} \tag{3.2}$$

$$\{([\texttt{byl}, \texttt{bohatý}], [\texttt{jednou}, \texttt{jeden}]), ([\texttt{jednou}, \texttt{král}], [\texttt{jeden}, \texttt{bohatý}])\} \tag{3.3}$$

$$\{([\texttt{jednou}, \texttt{jeden}], [\texttt{byl}, \texttt{bohatý}]), ([\texttt{jeden}, \texttt{bohatý}], [\texttt{jednou}, \texttt{král}])\} \tag{3.4}$$

3.3 Integrace dostupných kontextových informací do word2vec

Word2vec modely používají pro své trénování korpus, vyvstává tak otázka, jaký vliv by na model mělo použití jiných dat.

3.3.1 Druh informace

Pro tyto účely předpokládáme dva různé typy rozšiřujících informací:

- získané pomocí obrázků
- získané pomocí dotazů v internetovém vyhledávači

První předpokládá databázi obrázků, ke kterým existuje současně popis objektů na obrázku. Existuje možnost, že součástí anotovaných obrázků bude i informace o poloze a rozloze objektů, ta však pro tyto účely využita nebude. Informace o prováděné činnosti by naopak mohla být využita. Volně dostupnou databázi je např. MS COCO ([12]).

[8] Nebo na základě úhlu mezi vektory všech slov $\mathbf{v}_w$ a $\mathbf{v'}_w$
[9] Prostřednictvím vnějšího zásahu, který by upravil váhy
[10] Zkrácení všech vektorů nebo pouze nejdelších
[11] Hierarchický softmax neposkytuje výstupní reprezentaci slova
[12] Úprava vah by se prováděla opět střídavě pro CBOW nebo SG

Dále lze využít informací získaných při dotazech v internetových vyhledávačích. Databáze by měla obsahovat dvojici (W, url), kde W je množina[13] zadaných slov a url je URL adresa stránky, na kterou daný uživatel po vyhledání klikl. Předpokladem je, že tato databáze obsahuje několik dvojic pro každou url.

3.3.2 Reprezentace datové sady

Jelikož je získaná databáze z internetových vyhledávačů reprezentována jako $\{(W, url)\}$, použijeme celou tuto množinu, kde W budeme interpretovat jako kontext c a url jako slovo w. Je vhodné zmínit, že se bude lišit množina „slov“ (URL adres) od množiny slov, která se budou vyskytovat v kontextu.

Pro obrázky s popisem činnosti na obrázku obdržíme krátkou větu nebo souvětí. Věty lze rozložit na kontext a slovo a naložit s nimi stejně jako v případě korpusu. Obrázky popsané slovy poskytují pouze množinu slov[14], a tak může být náhodně vybráno jedno slovo z množiny a zbytek může být považován za jeho kontext.

3.3.3 Integrace do modelu

Pro fázi učení je nutné specifikovat, jak bude síť tvořena, nabízí se následující možnosti, které budou pod uvedením výčtu vysvětleny:

1. Samostatná síť – neuvažujeme korpus
2. Dvě oddělené sítě – spojení až po dokončení fáze trénování
3. Spojená síť
   (a) Beze změny – nelze použít URL
   (b) Rozšíření vstupu
      - SG – pro URL pouze střídání
      - CBOW – nelze použít URL
   (c) Rozšíření výstupu
      - SG – nelze použít URL
      - CBOW – pro URL pouze střídání
   (d) Rozšíření vstupu i výstupu

Možnost **1.** nepředpokládá existenci korpusu, pro URL je nutné použít vstupní vektorovou reprezentaci slov pro CBOW a výstupní pro SG, nelze tak použít SG v kombinaci s HS[15]. **2.** alternativa přináší spojení obou vektorových reprezentací slov, přičemž považuji za vhodné, aby ke spojení došlo prostřednictvím konkatenace. Bude tak možné využít každou z reprezentací zvlášť, ale také obě dohromady. Nevýhodou je, že se tím zvětší dimenze reprezentujícího vektoru.

Posledním navrženým způsobem je spojení obou sítí do jedné. Rozšířením chápeme prodloužení vstupního případně výstupního vektoru sítě, kde vektor bude rozdělen na dvě části, první bude tvořena odpovídajícími složkami vektoru z korpusu a druhá z přidané informace. To přináší určitou volnost z hlediska učení a nabízí se:

[13] Nebo seznam
[14] Bez určeného pořadí
[15] Hierarchický softmax

- přikládat střídavě korpus a rozšiřující informaci (URL, obrázky)
- přikládat zároveň obě informace

Pokud chceme zachovat původní vstupy i výstupy (**3.**a), pak rozšiřujeme pouze datovou sadu, což jak již bylo zmíněno, není možné pro URL adresy.

Pro URL adresy nepřipadají některé možnosti vůbec v úvahu a současné přikládání obou informací nelze využít v **3.**b ani v **3.**c.

Rozšíření vstupu i výstupu (**3.**d) by nutně vytvořilo dvě reprezentace pro jedno slovo, přičemž by se výsledek opět musel nějak spojit a nevidím v tomto přístupu výhodu oproti **2.** navržené variantě.

Jako nejlepší možnost se tak dle mého názoru jeví využití URL adresy ke střídavému učení nebo jako konkatenaci vektorů z dvou odlišných sítí. Obrázky vykazují značnou volnost a bylo by zajímavé zkusit to, co URL adresy neumožňují – přikládání obou informací zároveň. Ideálně tak, aby pro jedno slovo w byl zároveň vybrán jeho kontext c^{t} z korpusu a nějaký náhodný obrázek obsahující ve svém popisku slovo w. Z popisku by dále byl extrahován kontext (na základě věty nebo slov) c^{p} a kontext slova w by byl tvořen $c = c^{\mathrm{t}} \cup c^{\mathrm{p}}$.

Kapitola 4

Použité datové sady

Jako korpus byla použita předzpracovaná sada prvních 100 milionů znaků z anglické verze otevřené encyklopedie Wikipedia[1]. Tato datová sada je využita ve většině ukázkových příkladů včetně samotného originálu od T. Mikolova[2], zde je pouze doporučeno použít 1 miliardu znaků. Pro získání trénovací sady rozšiřující kontext prostřednictvím obrázků byly uvažovány následující dostupné datové sady:

1. ImageNet [19]
2. MS COCO [12]
   (a) Labels
   (b) Captions
3. Open Images [7]

Aby bylo možné získat ID obrázku a k němu několik kategorií (jeho kontext), musí datová sada tyto závislosti postihovat. **ImageNet** obsahuje pouze jednu kategorii ke každému obrázku, a tak je v tomto případě nepoužitelný.

Microsoft COCO naproti tomu poskytuje 3 různé části, přičemž pro tyto účely mohou být využity 2 – „Labels“ a „Captions“. Labels obsahují pro každý obrázek několik kategorií. Počet různých kategorií je ale velmi malý a pohybuje se okolo čísla 100. Captions obsahují pro každý obrázek textový popisek ve formě jedné či více vět.

Datová sada **Open Images** má podobnou formu jako Labels v MS COCO, ovšem obsahuje více než 6000 různých kategorií. Každý obrázek se váže k několika kategoriím, přičemž pro každou kategorii existuje *jistota*, že obrázek do této kategorie spadá. Použita byla část anotována lidmi, kde je *jistota* rovna 100 % nebo 0 %. Případy s 0 procentní jistotou nebyly žádným způsobem využity.

Nalezení datové sady, která by vyhovovala URL adresám popsaným výše (obsahující vyhledávaný text a odkaz, na který uživatel klikl), není snadné. Takové datové sady nebývají jednoduše veřejně přístupné. Jednou z takových datových sad je „AOL Search Data Collection“, která již není oficiálně přístupná, ale může být stažena z https://archive.org/download/AOL_search_data_leak_2006/AOL_search_data_leak_2006.zip.

[1] http://mattmahoney.net/dc/text8.zip
[2] https://code.google.com/archive/p/word2vec/

Kapitola 5

Implementace

Jelikož existuje mnoho implementací zmíněných word2vec modelů, a to jak CBOW, tak SG, rozhodl jsem se využít dostupné implementace za použití některé z populárních knihoven, z tohoto důvodu byla zvolena knihovna TensorFlow [4].

5.1 TensorFlow

TensorFlow je knihovna založena na data flow přístupu. Požadovanou úlohu si lze představit jako graf, kde uzly reprezentují matematické operace a hrany tensory. Tenzory jsou obecně multidimenzionální datová pole. Konkrétním případem jsou pak vektory či matice. Knihovna je koncipována tak, aby uživatelům usnadnila práci primárně v úlohách spojených se strojovým učením. Jelikož je struktura programu ve formě grafu, TensorFlow dokáže automaticky provést derivace výstupních uzlů podle vstupních na základě specifikace ztrátové funkce E. Tato knihovna je zpřístupněna pomocí programovacího jazyka Python. Využívá se jednoduchosti zápisu v tomto jazyce, přičemž knihovní funkce jsou implementované v C++, čímž je dosaženo zrychlení. Taktéž je možné vytvořit vlastní funkce v jazyce C++, které budou volány prostřednictvím skriptu v jazyce Python. Součástí TensorFlow knihovny jsou již funkční implementace Skip-gramu, které jsou dostupné skrze [2] a https://github.com/tensorflow/models/tree/master/tutorials/embedding, tyto byly použity a dále upraveny.

5.2 Trénování modelu

Jako referenční model byla zvolena architektura Skip-gram (SG) s využitím negativního vzorkování (negative sampling). Model má na vstupu jedno slovo w_I a na výstupu několik slov w_{O_i} (i značí index výstupního slova) pro každý trénovací případ. Podle úpravy vah sítě tak, jak bylo prezentováno v části 2.2.2. Jeden trénovací případ (původně 1 krok učení) pro SG může být rozdělen do k kroků, kde $k = |i|$. S využitím dávkového (batch) pravidla, kdy je úprava vah provedena až po vykonání několika kroků (v tomto případě je velikost dávky k), dostáváme k vztahů (5.1) pro úpravu vah, kde $\mathbf{v}_{w_I}$ respektive $\mathbf{v'}_{w_O}$ je reprezentace aktuálního slova na vstupu respektive na výstupu. Chyba E je vyjádřena ve formě absolutní hodnoty tak, aby byla v souladu s chybou podle vztahu 2.24. Pomocná proměnná $Match$ vyjadřuje shodu vstupního vektoru s výstupním[1], proměnná $Move$ určuje požadavek na změnu míry sdílení vstupního a výstupního vektoru (nabývá hodnoty v rozmezí -1 až 1),

[1] To odpovídá výstupu neuronu y_j, kde j značí index slova w_O

pozitivní hodnoty představují přibližování vektorů, zatímco negativní oddalování. σ značí sigmoidální funkci tak, jak byla zavedena v 2.16. Požadovaná hodnota výstupu D nabývá hodnoty 1 pro výstupní slovo w_O a 0 pro negativní vzorky.

$$
\begin{aligned}
\mathrm{u} &= ({\mathbf{v}'_{w_O}}^T \mathbf{v}_{w_I}) \\
\mathrm{Match} &= \sigma(u) \\
-E &= \log(|(1-D) - Match|) \\
-E &= \log(|(1-D) - \sigma(u)|) = \begin{cases} \log(\sigma(-u)) & \text{pokud } D = 0 \\ \log(\sigma(u)) & \text{pokud } D = 1 \end{cases} \\
\mathrm{Move} &= \frac{\partial - E}{\partial u} = \begin{cases} \frac{1}{(\sigma(-u))} \cdot (\sigma(-u) \cdot \sigma(u)) & \text{pokud } D = 0 \\ \frac{1}{(\sigma(u))} \cdot (\sigma(u) \cdot \sigma(-u)) & \text{pokud } D = 1 \end{cases} \\
\mathrm{Move} &= D - \sigma(u) = D - \mathrm{Match} \\
\frac{\partial - E}{\partial \mathbf{v}_{w_I}} &= \mathrm{Move} \cdot {\mathbf{v}'}_{w_O} \\
\frac{\partial - E}{\partial {\mathbf{v}'}_{w_O}} &= \mathrm{Move} \cdot \mathbf{v}_{w_I}
\end{aligned}
\tag{5.1}
$$

5.2.1 Záměna vstupu a výstupu

Přestože by velikost dávky měla být k, v praxi tato podmínka není důležitá a velikost dávky může být zvolena libovolně vhodně. Je nutné si uvědomit, že tento postup nelze použít pro CBOW, 1 trénovací případ totiž nelze rozložit na více kroků, protože vstupní vektor je kombinací všech slov na vstupu.

Pokud chceme, aby se na vstup sítě přikládala slova (matice W bude použita pro *word embedding*), ale na výstup sítě např. ID obrázků, nelze využít architekturu SG, protože jeden trénovací případ tvoří 1 ID obrázku a několik slov. Pokud však využijeme vztahů 5.1 pro úpravu vah v jednom kroku a zaměníme vstupní a výstupní reprezentaci slova (nahrazení ${\mathbf{v}'}_{w_O} \rightarrow \mathbf{v}_{w_O}$, $\mathbf{v}_{w_I} \rightarrow {\mathbf{v}'}_{w_I}$), docílíme vztahů pro úpravu vah tak, že výstupní vrstva může být reprezentována např. ID obrázků a vstupní vrstva bude obsahovat požadovaná slova. Za předpokladu, že na vstupu i výstupu budou slova a proběhne učení z textového korpusu tak, jak bylo prezentováno, pak budou výsledné trénovací sady až na první a poslední případ ekvivalentní. To lze ukázat následujícím příkladem, kdy je velikost kontextového okna rovna 1, 5.2 pak vyjadřuje původní datovou sadu, 5.3 upravenou (pro větu `byl jednou jeden bohatý král`). Pokud z datových sad odstraníme první a poslední dvojici, datové sady jsou ekvivalentní (liší se pouze v pořadí přikládání).

$$
\begin{aligned}
&\{(\texttt{jednou}, \texttt{byl}), (\texttt{jednou}, \texttt{jeden}), (\texttt{jeden}, \texttt{jednou}), \\
&(\texttt{jeden}, \texttt{bohatý}), (\texttt{bohatý}, \texttt{jeden}), (\texttt{bohatý}, \texttt{král})\}
\end{aligned}
\tag{5.2}
$$

$$
\begin{aligned}
&\{(\texttt{byl}, \texttt{jednou}), (\texttt{jeden}, \texttt{jednou}), (\texttt{jednou}, \texttt{jeden}), \\
&(\texttt{bohatý}, \texttt{jeden}), (\texttt{jeden}, \texttt{bohatý}), (\texttt{král}, \texttt{bohatý})\}
\end{aligned}
\tag{5.3}
$$

5.3 Struktura projektu

Projekt se skládá z několika skriptů, nejpodstatnější složky a soubory budou blíže popsány a jsou zobrazeny pomocí následující stromové struktury. Složka *embeddings* obsahuje výsledné

reprezentace slov, *models* váhy celé sítě tak, aby bylo možné pokračovat ve fázi učení sítě, *projections* vizualizaci (maximálně 500) nejčastějších slov ve 2D prostoru pomocí metody t-SNE [13], *vocabs* soubory se slovní zásobou, kde číslo řádku odpovídá indexu slova ve word embeddingu.

Složka *picture_datasets* obsahuje soubory *cifar-3closest.txt*, *coco-captions.txt*, *coco-labels.txt* a *open-image-labels.txt*, které slouží jako datová sada s rozšířenou kontextovou informací. Dále pak zahrnuje složku *cnn-cifar*, v níž se mimo jiné nachází skript *cifar10_eval.py*, který vytváří datovou sadu související s konvolučními neuronovými sítěmi, ostatní obrazové datové sady jsou vytvořeny pomocí skriptu *preprocess.py* ve složce *create* ze zbývajícího obsahu této složky. Obrazové datové sady jsou tvořeny posloupností slov vztahující se k jednomu obrázku, jednotlivé obrázky jsou odděleny oddělovačem `|||`.

Na hlavní úrovni se vyskytuje textový korpus *text8*, evaluační sada *questions-words.txt*, skript *download*, který stáhne evaluační sadu a korpus, a sada příkazů *install* ve formě skriptu.

Konečně soubory *word2vec_kernels.cc* a *word2vec_ops.cc*, které obsahují implementaci operací potřebných ve fázi trénování v jazyce C++ a jejichž překladem vznikne soubor *word2vec_ops.so*. Tyto operace jsou využity skriptem *train.py*, *eval_embeddings.py* pracuje s vytvořenými vektorovými reprezentacemi slov.

```
DP
├── embeddings
├── models
├── picture_datasets
│   ├── cifar-3closest.txt
│   ├── coco-captions.txt
│   ├── coco-labels.txt
│   ├── open-image-labels.txt
│   ├── cnn-cifar
│   │   └── cifar10_eval.py
│   └── create
│       ├── coco-captions.json
│       ├── dict.csv
│       ├── instances_train2014.json
│       ├── labels.csv
│       └── preprocess.py
├── projections
├── vocabs
├── word2vec_kernels.cc
├── word2vec_ops.cc
├── word2vec_ops.so
├── questions-words.txt
├── eval_embeddings.py
├── train.py
├── text8
├── download
└── install
```

5.4 Instalace

Pro využití funkcí z tohoto projektu je nutné mít nainstalovanou knihovnu TensorFlow, Python 3 a jeho knihovny matplotlib, sklearn a scipy. Projekt byl vyvíjen pod operačním systémem Mac OS a výše zmíněné nástroje by pod tímto operačním systémem mělo být možné nainstalovat prostřednictvím skriptu *install* (namísto spuštění skriptu je doporučeno ručně provést jednotlivé příkazy). Tento skript taktéž zkompiluje zdrojové soubory v C++ a vytvoří z nich soubor *word2vec_ops.so*. Pro operační systémy Windows není zaručeno, že bude možné spustit fázi trénování. Další operační systémy založené na Unixu by měly být podporovány, ovšem instalace některých nástrojů a překlad se může lišit, v tom případě doporučuji postupovat podle návodu uvedeného na https://www.tensorflow.org/install/ respektive https://github.com/tensorflow/models/tree/master/tutorials/embedding pro překlad. Pokud se nevyžaduje žádný externí zásah do aplikace (jiné nastavení), spouštění procesu trénování je zbytečné[2], modely jsou již vytvořeny. Taktéž opětovný běh skriptů vytvářejících datové sady postrádá smysl. Využít však lze skriptu *eval_embeddings.py*, který prostřednictvím velmi jednoduchého grafického uživatelského rozhraní přibližuje naučené reprezentace slov.

5.5 Možnosti nastavení

Skripty *eval_embeddings.py* a *train.py* sdílejí nastavení prostřednictvím globálních proměnných na jejich počátku. Předpokládá se, že uživatel před jejich spuštěním tyto proměnné ručně modifikuje, jedná se o proměnné `EVALUATION_TYPE` (dále `ET`) a `PICTURE_DATASET` (dále `PD`). `PD` se vztahuje k použitým obrazovým datovým sadám, jak bylo prezentováno v 4.

- **ET = 0** – základní typ (SG architektura)
- **ET = 1** – modifikovaný SG podle 5.2.1 (výměna vstupů a výstupů při učení)
- **ET = 2** – pouze obrazová informace, na výstup se přikládají ID obrázku, na vstup slova vztahující se k danému obrázku
  - **PD = 1** – MS COCO Labels
  - **PD = 2** – MS COCO Captions
  - **PD = 3** – Open Images
- **ET = 3** – obrazová informace ve formě textu, tento způsob využívá pouze tradiční textový korpus, který je vytvořen na základě obrazové datové sady, ze které jsou odstraněny oddělovače obrázků `|||`
  - **PD = 1** – MS COCO Labels
  - **PD = 2** – MS COCO Captions
  - **PD = 3** – Open Images
- **ET = 4** – střídavé přikládání obrazové a textové informace (střídání probíhá po epochách, jedna epocha představuje iteraci přes celou datovou sadu)

[2] Některé varianty se zdají být nezajímavé a nejsou natrénovány, viz 5.5

  - **PD = 1** – MS COCO Labels (není dopředu natrénováno)
  - **PD = 2** – MS COCO Captions
  - **PD = 3** – Open Images (není dopředu natrénováno)

- **ET = 5** – obrazová informace ve formě textu společně s původním korpusem (střídání po epochách)
  - **PD = 1** – MS COCO Labels (není dopředu natrénováno)
  - **PD = 2** – MS COCO Captions
  - **PD = 3** – Open Images (není dopředu natrénováno)

- **ET = 6** – základní typ, který je rozšířen o reflexi tak, jak bylo popsáno v části 3.2.1 pod pojmem rovnoběžnost vektorů (datová sada obsahuje navíc (w, w))

- **ET = 7** – pro tvorbu trénovací sady je využita konvoluční neuronová síť (podrobněji bude vysvětleno v části 6)

- **ET = 8** – pro tvorbu trénovací sady je použita konvoluční neuronová síť, zároveň je zaměněn vstup a výstup

Varianty, u kterých se uvažují obrázky z obrazových datových sad vždy využívají úpravu podle 5.2.1, aby se dala použít matice vah W (vstupní vektorová reprezentace slov) pro word embedding.

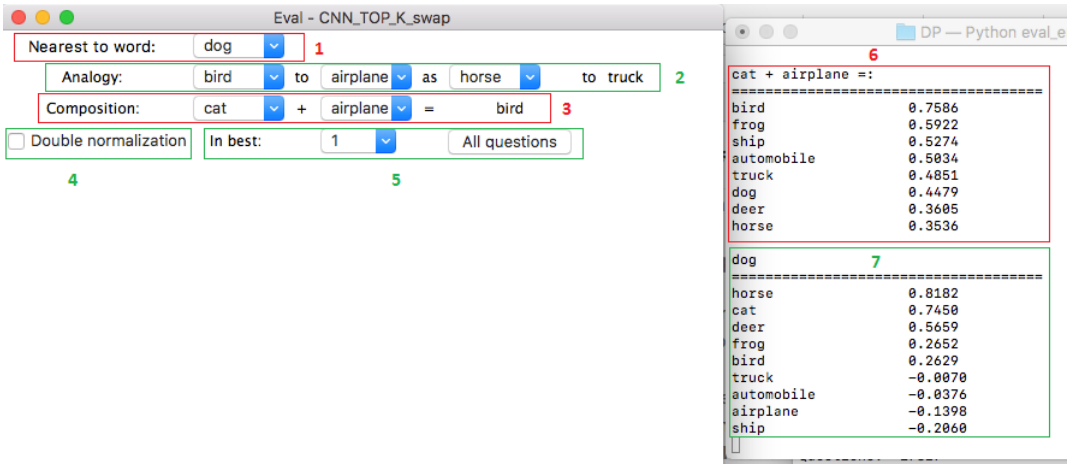
5.6 Skripty pro trénování

Trénování probíhá podle vztahu 5.1, koeficient učení η je na počátku $0,025$ a lineárně se snižuje až k hodnotě $0,0001$ v závislosti na počtu provedených kroků. Do slovní zásoby jsou zahrnuta všechna slova, která se vyskytla alespoň 5x ve zdrojovém korpusu. Na jeden pozitivní případ se vyskytuje 25 negativních případů a reprezentace slov je 200 dimenzionální. Velikost dávky (batch) je 1, velikost kontextového okna je pro každý případ náhodně uniformě zvolena v rozmezí 1 až 5, počet epoch je stanoven na 15. Rozložení negativních vzorků odpovídá vztahu 2.23, *word2vec_ops.cc* slouží jako hlavičkový soubor, *word2vec_kernels.cc* má za úkol zpracovat datové sady a na základě kombinace vstupních parametrů upravovat váhy sítě. Vstupních parametrů je několik a není doporučeno je libovolně měnit, neboť jejich kombinace nemusí dávat smysl. Dokumentace vstupních parametrů se vyskytuje přímo ve zdrojových kódech a proto ji zde nebudu uvádět. Pro každý z `EVALUATION_TYPE` jsou přednastavené. Trénování lze spustit příkazem `python3 train.py`, přičemž právě prostřednictvím *train.py* probíhá koordinace a nastavení parametrů.

5.7 Evaluační skript

Na začátku evaluačního skriptu *eval_embeddings.py* jsou kromě již zmíněných parametrů `EVALUATION_TYPE` a `PICTURE_DATASET` další parametry. `SAVE_VISUALIZATION` říká, zda-li se má vytvářet a ukládat již zmíněná projekce reprezentace vektorů slov do 2D prostoru po ukončení grafického uživatelského rozhraní (opět platí, že jsou již součástí projektu). Počet slov ve slovní zásobě může být velký[3] a vybírat slova z rozbalovacího seznamu [4]

[3] Pro některé typy je počet slov ve slovní zásobě větší než 70 tisíc
[4] Combobox



Obrázek 5.1: Jednoduchý náhled na aplikaci pro evaluaci naučených vektorových reprezentací slov.

tak trvá nepřiměřeně dlouho, z tohoto důvodu je parametr `USE_COMBO` nastaven na hodnotu 1, která říká, že se rozbalovací seznamy použijí pouze, pokud velikost slovní zásoby nepřesáhne velikost 10 tisíc, tuto variantu lze změnit, kdy 0 znamená použití textových polí a 2 použití rozbalovacích seznamů. Obrázek 5.1 znázorňuje pohled na aplikaci, část **1** dává možnost prozkoumat nejbližší slova danému slovu, **2** již představené analogie, **3** kombinace slov[5], **5** provede ohodnocení analogií přes evaluační datovou sadu, která bude popsána v 5.8. Volba `In best` určuje, mezi kolika nejbližšími slovy musí požadovaná odpověď být (toto ohodnocení může být pro velké slovní zásoby časově náročné). Možnost `Double normalization` v části **4** souvisí s analogiemi, které vychází ze vztahu 2.29. Ten očekává vektor $\mathbf{X}$ například pro analogii `Spain to Madrid as Lisbon to?` ve tvaru 5.4, volbou `Double normalization` dojde k použití vztahu 5.5, kde norm() vyjadřuje normalizaci vektoru na jednotkový. Taková úprava způsobí, že při porovnání ještě více zvýhodníme svíraný úhel oproti absolutní velikost vektoru, což má na některá slova velký vliv. Rozdíl v celkové procentuální úspěšnosti testovací sady mezi oběma případy však není pro textový korpus markantní. Změna parametru `PRINT_CORRECT_ANSWERS` na hodnotu 1 způsobí průběžný výpis každé správné odpovědi z celé vyhodnocovací sady při provedení evaluace po stisku tlačítka `All questions`.

V části **6** a **7** dochází k tisku[6] odpovědí. Odpovědi se vyhodnotí po změně hodnoty rozbalovacího seznamu nebo po stisku klávesy *enter* u textových polí.

$$\mathbf{X} = vec(\texttt{Spain}) - vec(\texttt{Madrid}) + vec(\texttt{Lisbon}) \tag{5.4}$$

$$\mathbf{X} = \text{norm}(vec(\texttt{Spain}) - vec(\texttt{Madrid})) + vec(\texttt{Lisbon}) \tag{5.5}$$

5.8 Evaluace

Na rozdíl od klasifikačních či rozpoznávacích úloh neexistuje objektivní možnost, jak určit kvalitu naučených reprezentací slov. Jedním ze způsobů, který se používá, jsou již prezentované analogie, jako evaluační sada analogií je použita ta, která byla prezentována v [14]

[5] Například stát + měna
[6] Do terminálu, ze kterého byl skript spuštěn

(dostupná z http://www.fit.vutbr.cz/~imikolov/rnnlm/word-test.v1.txt). Je složena z několika podkategorií (celkový počet analogií je 19544). Následuje výčet těchto podkategorií vyskytujících se v evaluační sadě.

- Hlavní město – stát
- Město – stát
- Stát – měna
- Rodina (chlapec ku dívce jako bratr ku sestře)
- Přídavné jméno – příslovce (úžasné – úžasně)
- Slova opačná (přijatelný – nepřijatelný)
- Stupně přídavných jmen (dobrý – lepší)
- Slovesa v prostém a průběhovém přítomném čase (go – going)
- Stát – příslušník státu (Česko – Čech)
- Jednotné – množné číslo (banán – banány)
- Koncovka ve 3. osobě čísla jednotného u sloves (decrease – decreases)

Již před provedením evaluace je jasné, že z dostupných obrazových datových sad nebude možné většinu vztahů vůbec odvodit. Pro vyhodnocení se tak používají pouze analogie, jejichž (všechna 4 požadovaná) slova se vyskytují ve slovní zásobě.

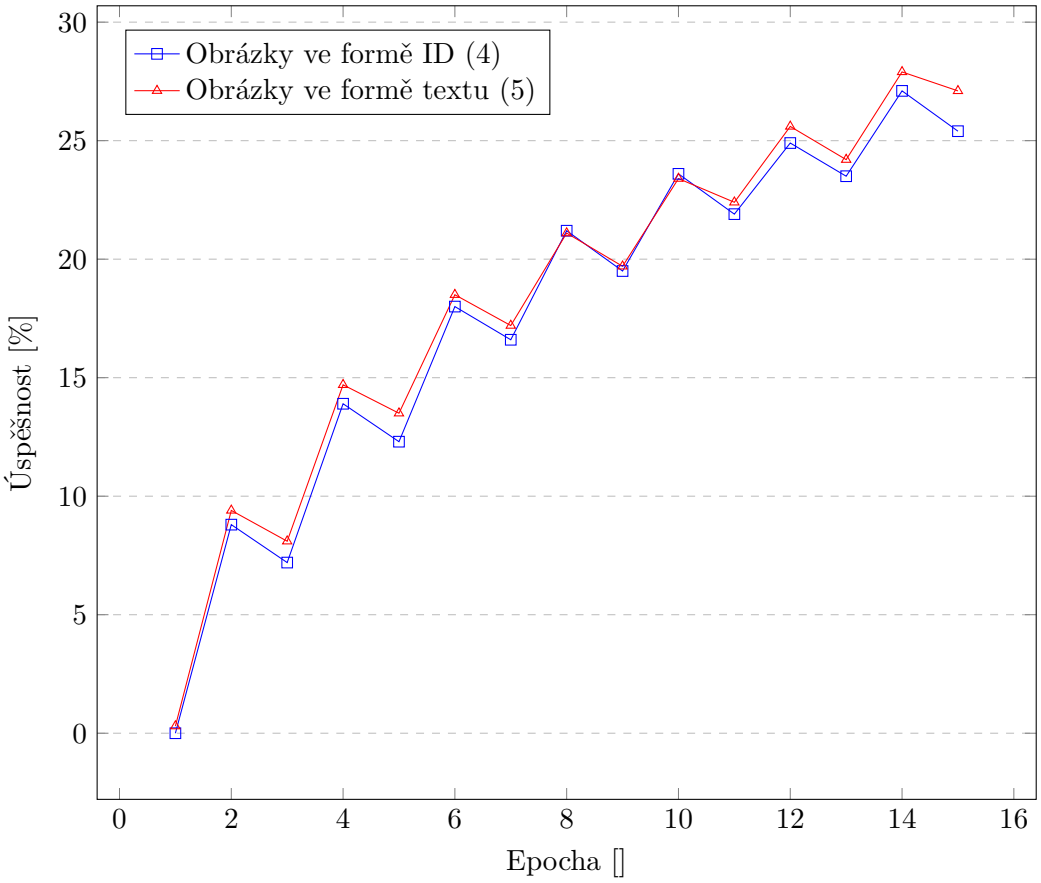
5.8.1 Úspěšnost vyhodnocovací sady

Tabulka 5.1 ukazuje úspěšnost při použití textového korpusu, který garantuje vyhodnocení většiny otázek z evaluační sady. Za zmínku stojí vizualizace průběhu trénování (Obrázek 5.2), kdy se střídavě používají textové a obrazové informace. V lichých epochách je aktivní obrazový a v sudých textový korpus. Reprezentace obrázků pomocí ID ani extrahovaný text nepřispívají ke zlepšení vektorové reprezentace slov, dokonce ji podstatně zhoršují. Je využito popisků (MS COCO Labels), které by měly být pro tyto účely nejvhodnější. Zhoršení si vysvětluji tím, že informace z obrázků poskytují pouze velmi omezenou podmnožinu slov a tím pádem mají negativní dopad na úspěšnost dalších analogií.

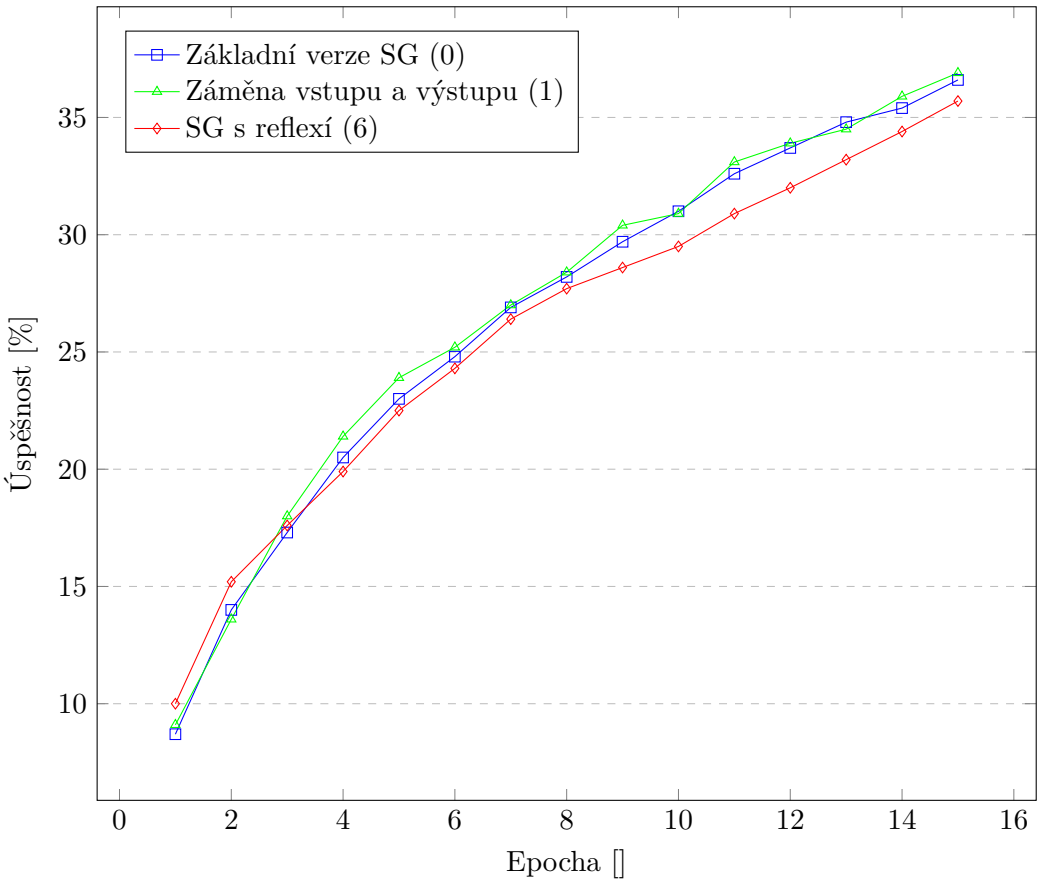
Tabulka 5.1: Úspěšnost architektur pracujících s datovými sadami tvořenými pouze textem nebo textem a obrazovou informací. *Otázek* vyjadřuje počet otázek, které by mohly být správně zodpovězeny (vyskytují se ve slovní zásobě).

	Otázek	EVALUATION TYPE				
		0	**1**	**4**	**5**	**6**
Úspěšnost [%]	17827	36.6	36.9	25.4	27.1	35.7

Z Obrázku 5.3 lze usoudit, že pro čistě textový korpus a učení přes kontextové okno nemá záměna vstupu a výstupu u architektury SG vliv na úspěšnost v průběhu trénování[7],



Obrázek 5.2: Porovnání úspěšnosti při kombinovaném učení z textu a z obrázků, jako obrazová informace je použit dataset MS COCO Captions (varianty, kdy `EVALUATION_TYPE` nabývá hodnot 4 a 5; `PICTURE_DATASET = 2`). Úspěšnost odpovídá procentu správně zodpovězených analogií.



Obrázek 5.3: Porovnání úspěšnosti metod, které pracují nad textovým korpusem (varianty, kdy `EVALUATION_TYPE` nabývá hodnot 0, 1 a 6). Úspěšnost odpovídá procentu správně zodpovězených analogií.

jak bylo předpokládáno v 5.2.1. Přidání reflexe do datové sady taktéž nemá markantní vliv na kvalitu word embeddingu, čehož bude využito později.

Při využití čistě obrazových informací ve formě ID nebo textu je podstatně redukována velikost slovní zásoby, tím pádem se také zmenšuje maximální možný počet otázek, které může model zodpovědět. Pro MS COCO Labels dokonce neexistuje žádná, jelikož slovní zásoba je tvořena pouze 90 slovy. Jak je patrné z Tabulky 5.2, lepších výsledků je dosaženo, pokud se obrazová informace využije ve formě textu. Domnívám se, že je to způsobeno sdílením části informace ve výstupní vrstvě. Uvažujeme-li ID obrázků, pak jeden neuron ve výstupní vrstvě představuje jeden obrázek. Těchto neuronů je velmi mnoho a jejich cílem je naučit se reagovat pouze na tento jeden případ. Zatímco pokud uvažujeme textovou variantu, jeden neuron výstupní vrstvy je ovlivněn mnoha případy a informace je zde sdílena. Navíc, jak bylo ukázáno v části 5.2.1, učení prostřednictvím posunu kontextového okna generuje pro slova a a b případy (a, b) i (b, a).

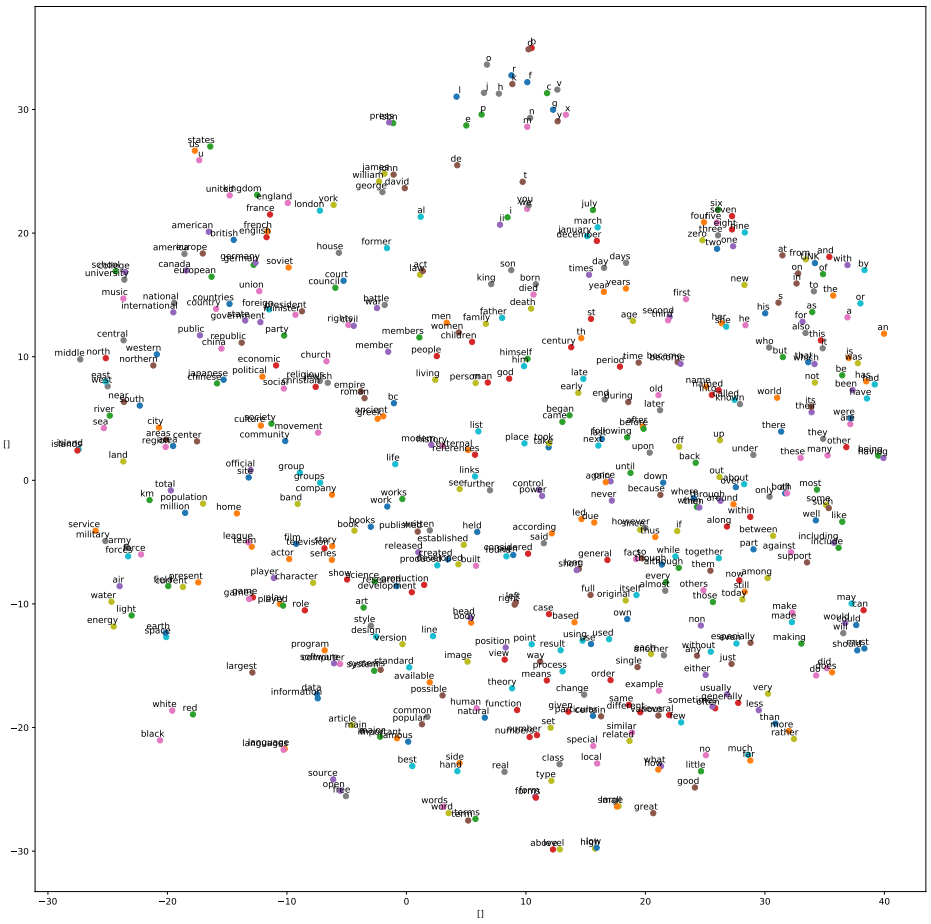
Tabulka 5.2: Úspěšnost architektur pracujících pouze s datovými sadami tvořenými obrázky. `PD` označuje `PICTURE_DATASET` a `ET` je zkrácenina `EVALUATION_TYPE`. *Otázek* vyjadřuje počet otázek, které by mohly být správně zodpovězeny (vyskytují se ve slovní zásobě).

	PD		**Otázek**	**ET** **2**	**3**
Úspěšnost [%]	**1**	MS COCO Labels	0	-	-
	2	MS COCO Captions	898	6.0	14.7
	3	Open Images	90	4.4	8.9

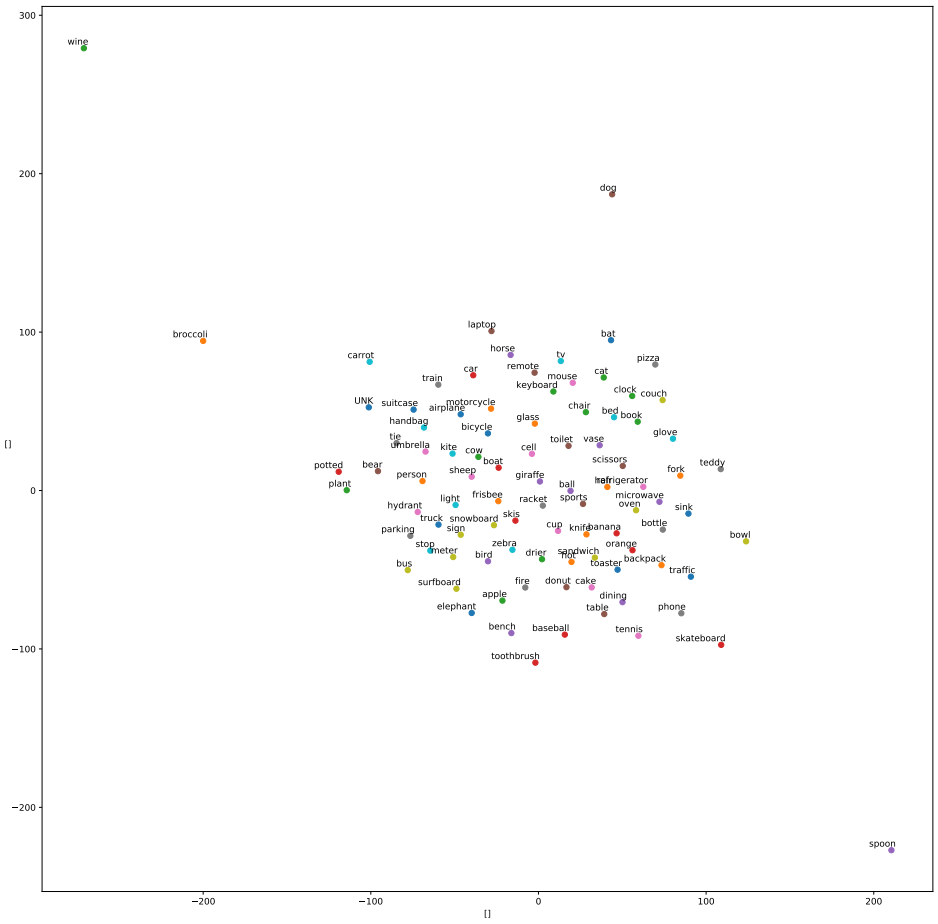
5.9 Výsledná reprezentace slov

Jak bylo zmíněno, pro každou variantu byla vytvořena projekce do 2D prostoru prostřednictvím metody t-SNE. Přestože se jedná o projekci do 2D prostoru, 3. dimenzi tvoří barva, podobné objekty mají stejnou barvu. Je zobrazeno maximálně 500 nejčastějších slov, Obrázek 5.4 ukazuje prostor naučený základní verzí SG pomocí textového korpusu, Obrázek 5.5 MS COCO Labels, Obrázek 5.6 MS COCO Captions a Obrázek 5.7 Open Images. Znázorněny jsou výsledné reprezentace s využitím obrazové informace ve formě ID obrázků (`EVALUATION_TYPE = 2`).

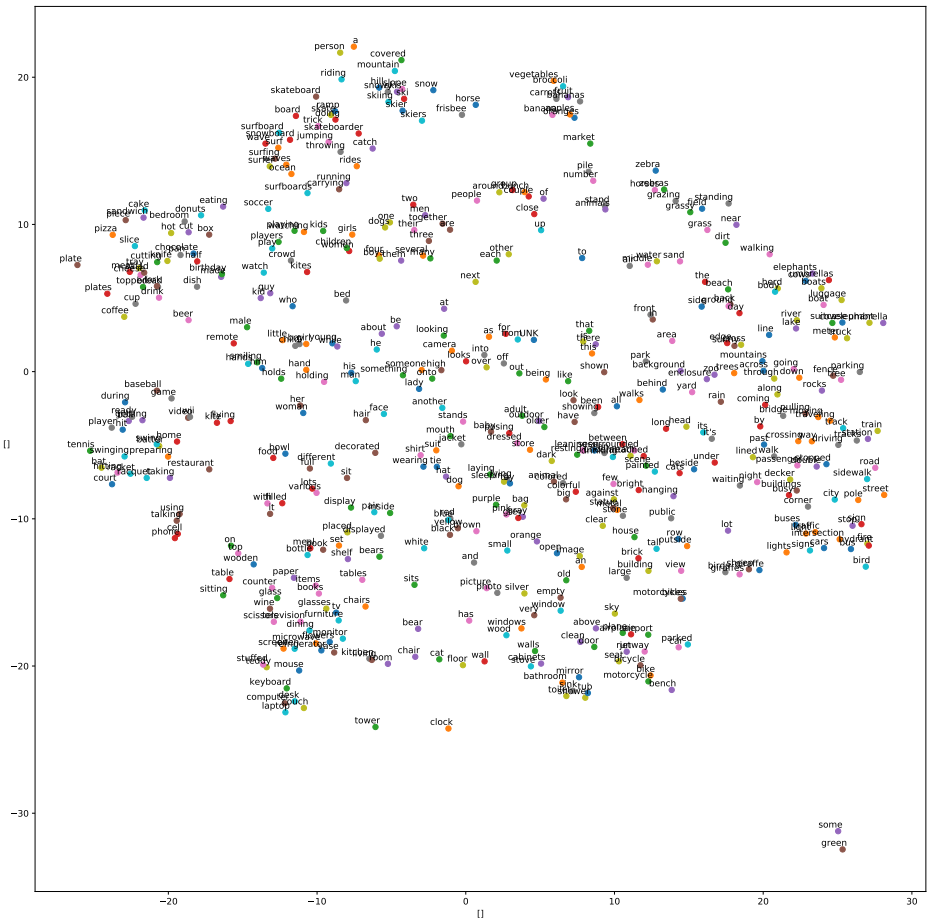
[7] Měřeno podle počtu korektně zodpovězených analogií



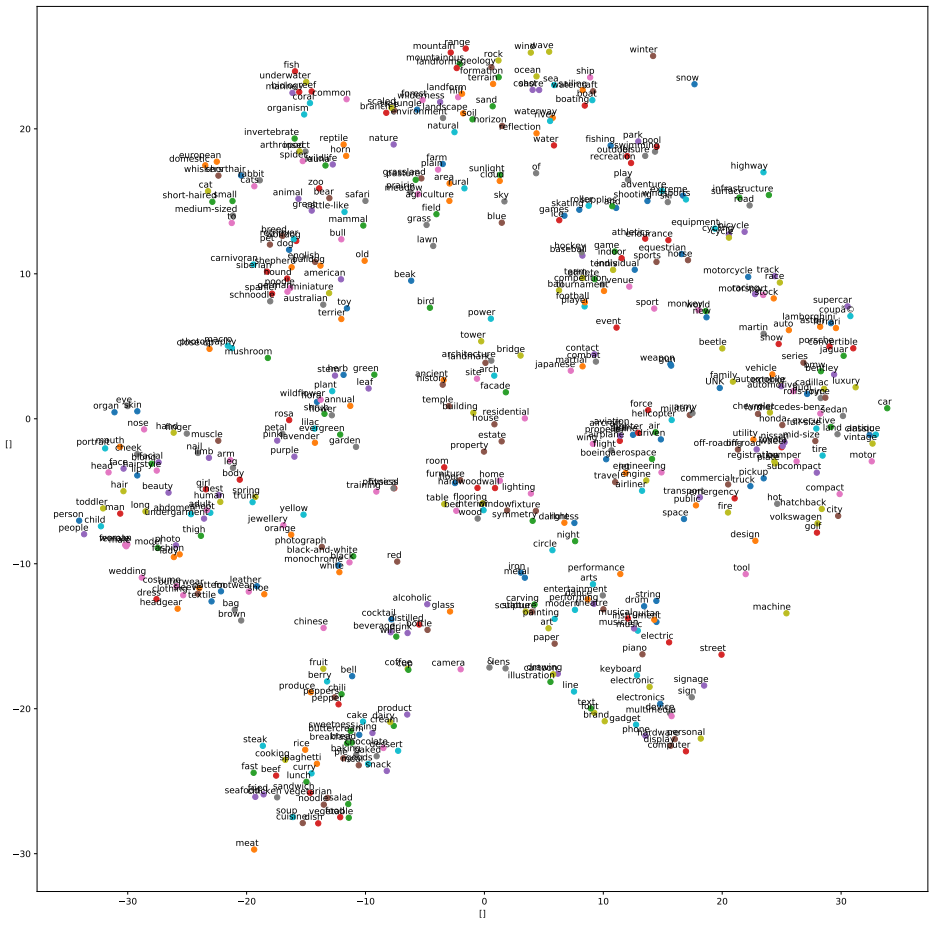
Obrázek 5.4: Projekce vytvořeného word embeddingu pro základní architekturu SG.



Obrázek 5.5: Projekce vytvořeného word embeddingu pro model naučený na základě datové sady MS COCO Labels.



Obrázek 5.6: Projekce vytvořeného word embeddingu pro model naučený na základě datové sady MS COCO Captions.



Obrázek 5.7: Projekce vytvořeného word embeddingu pro model naučený na základě datové sady Open Images.

Kapitola 6

Využití konvolučních neuronových sítí

Přestože lze k učení word2vec modelů využít obrázky tak, jak bylo prezentováno výše, vztah slov bude vždy závislý na dodaných kategoriích[1] z obrázku. Myšlenka této metody spoléhá na společný výskyt souvisejících kategorií v jednom obrázku. To nápaditě připomíná distribuční hypotézu, na které jsou založeny word2vec modely. Primárním účelem word2vec modelů je zpracování přirozeného jazyka (NLP), a tak se předpokládá, že bude existovat vektorová reprezentace pro každé slovo. Tento předpoklad je ale velmi obtížně dosažitelný skrze obrázky, kde se klade důraz na popis **objektů** vyskytujících se v obraze.

Pokud bychom využili nějaký prostředek, který na základě obrázku určí kategorie, které se na něm vyskytují, pak to odpovídá Open Images a MS COCO Labels datovým sadám. Tímto prostředkem bude konvoluční neuronová síť (CNN), stručný popis jedné z možných variant CNN následuje v části 6.1.

6.1 Konvoluční neuronová síť

V poslední době jsou velmi populárním prostředkem pro klasifikaci obrázků právě konvoluční neuronové sítě. Nově navrženými modifikacemi se dosahuje stále lepších výsledků, pro účely této práce však bude popsána pouze původní architektura sítě, jež způsobila rozmach této oblasti, ta byla prezentována v [9]. Pro popis CNN bylo také čerpáno z [1, 5, 17], tyto mohou sloužit pro hlubší pochopení tematiky.

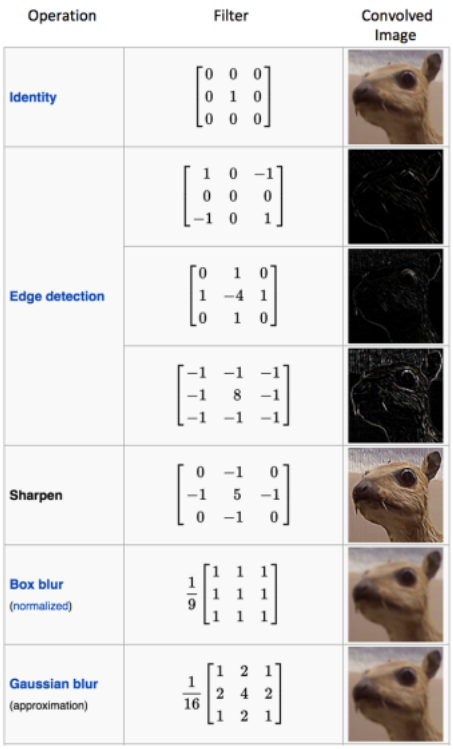
Pro porozumění fungování CNN je nejdříve nutné vysvětlit funkci obrazových filtrů a konvoluci. Jádro obrazového filtru je matice o rozměrech (n, m). To je aplikováno na každý pixel v obraze tak, že dojde k přiložení jeho středu na tento pixel. Překrývající se složky obrazu a jádra filtru se mezi sebou vynásobí a odezvě filtru odpovídá suma vynásobených hodnot[2]. Aplikací na všechny pixely se získá odezva obrazového filtru. Ukázky jader filtrů o rozměrech 3×3 a jejich odezvy jsou zachyceny Obrázkem 6.1.

Jednu z architektur CNN sítě zobrazuje Obrázek 6.2. Na vstup se přikládá obrázek, který může být barevný[3] nebo ve stupních šedi. Provedeme *konvoluci* (convolution) s k obrazovými filtry a získáme k map rysů (feature map). Na tyto rysy je aplikována nelineární

[1] Případně popisku u MS COCO Captions

[2] Tato operace se nazývá konvoluce, z toho plyne název konvoluční neuronová síť

[3] Barevný obrázek se přikládá jako 3 barevné kanály RGB



Obrázek 6.1: Ukázka některých obrazových filtrů a jejich jader (převzato z [1]).

aktivační funkce, která se nazývá ReLU (Rectified linear unit). Je velmi jednoduchá a lze ji vyjádřit pomocí vztahu 6.1 (způsobí odstranění záporných hodnot).

$$\mathrm{ReLU}(x) = \max(x, 0) \tag{6.1}$$

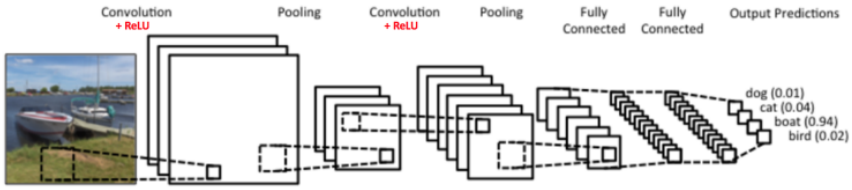
Na každou takto vytvořenou mapu rysů se aplikuje operace nazvaná *pooling*, která provede zmenšení počtu rysů pro určitou velikost okna (uvažujme 2×2). Jednou z variant je *max pooling*, který prozkoumá okno a vybere z něho maximální hodnotu. Pokud tak uvažujeme okno o velikosti 2×2, kdy každý rys z původních map rysů spadá právě do jednoho okna (okno posouváme o 2 pixely vodorovně i svisle), získáme 50 procent původních rysů.

Operace *konvoluce* a *max pooling* se několikrát opakují. Na výstupu poslední vrstvy se objeví vektor rysů, ten je připojen na vstup jedné nebo více plně propojených vrstev. Jejich výstup je přiveden na vrstvu tvořenou softmaxem, která přiřadí každému známému objektu [4] pravděpodobnost, že se na vstupu tento objekt vyskytuje[5].

6.2 Tvorba datové sady

Výsledné pravděpodobnostní rozložení přibližuje Obrázek 6.3. V datových sadách MS COCO Labels nebo Open Images se vyskytují ID obrázků a k těm se váže několik objektů, které se v obraze opravdu vyskytují. Pokud porovnáme odezvu CNN s těmito datovými sadami, pak v některých případech, kdy se na obrázku vyskytuje více objektů, může být ekvivalentní. Mnohem zajímavější jsou situace, kdy se na obrázku vyskytuje pouze jeden objekt

[4] Obecně kategorii ke klasifikaci
[5] Součet pravděpodobností přes všechny objekty musí být 1



Obrázek 6.2: Příklad struktury konvoluční neuronové sítě (převzato z [1]).

(např. leopard). Pokud využijeme faktu, že si síť s určitou pravděpodobností myslí, že se na obrázku nevyskytuje leopard, ale jaguár, pak můžeme tuto skutečnost interpretovat jako názor sítě o podobnosti objektů. Využíváme tak nedokonalosti klasifikace pro účely vytvoření vektorové reprezentace slov.

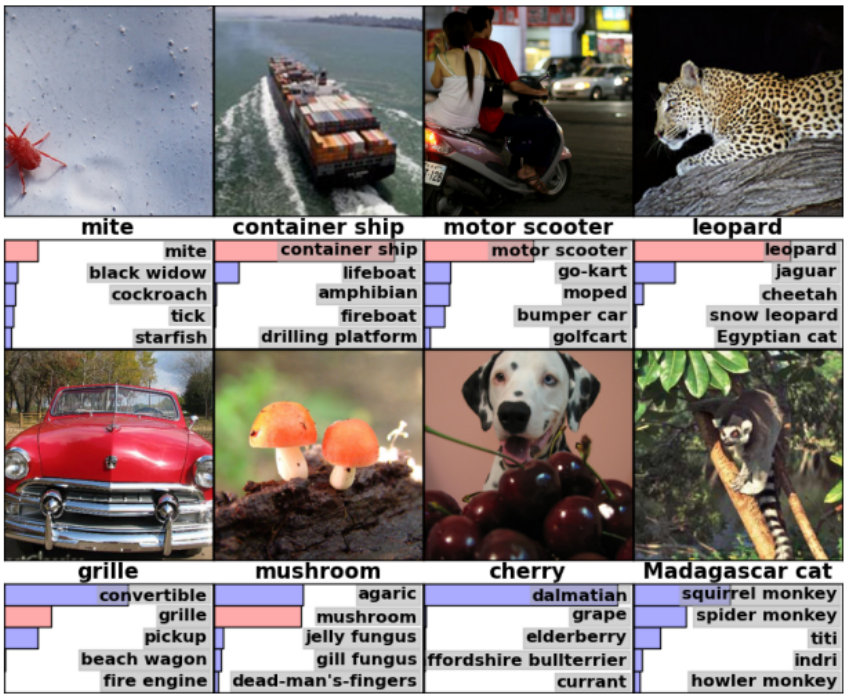
Přestože může tato změna působit nepodstatně, její význam je nezanedbatelný. Budeme-li síti předkládat pouze obrázky, na kterých se vyskytuje právě jeden objekt, pak člověk ve fázi učení síti **nedodává žádnou informaci o vztazích mezi objekty**. Souvislost objektů tak může plynout pouze z jejich podobnosti (či z typického okolí objektu). Naproti tomu při učení word2vec modelů z textového korpusu nebo anotovaných obrázků souvisí podobnost objektů s informací získanou od lidí (ručně anotované obrázky, přirozený jazyk), která je explicitně dodána (přestože se uvádí, že word2vec spadá do metod učení bez učitele).

Informace dodaná této (CNN) síti má pouze formu **„nechť se tyto objekty nazývají objekt *pes*“**. Naučená vektorová reprezentace slov tak není žádným způsobem závislá na člověku a jeho interpretaci. S využitím těchto znalostí se nabízí několik způsobů, jak vytvořit datovou sadu pro trénování word2vec modelů.

1. Použití nejlepších k kategorií
2. Použití pravděpodobnostního rozložení do kategorií
   (a) Zahrnutí správné kategorie
   (b) Ignorování správné kategorie
3. Použití výstupu z mapy rysů

Varianta **1.** vytvoří pro obrázek, který má být zařazen do kategorie lab, k trénovacích případů (lab, cat_i), kde $1 \leq i \leq k$ a cat_i značí kategorii s i-tou největší pravděpodobností. Jak bylo ukázáno v části 5.8.1, výskyt (lab, lab)[6] nemá negativní vliv na učení sítě. Varianta **2.** vytvoří pro obrázek, který má být zařazen do kategorie lab, trénovací sadu (lab, cat) pro všechny kategorie (případně pro všechny vyjma (lab, lab) u **2.**b). Síť bude ale učena s požadovanou hodnotou D, která nebude buď 0 nebo 1, ale bude odpovídat pravděpodobnosti, kterou síť přiřadí kategorii cat (případně bez uvažování správné kategorie na výstupu pro variantu **2.**b). Učení pak bude probíhat podle vztahu 5.1, který byl vyjádřen ve formě požadované hodnoty D právě pro tyto účely. Varianta **3.** využije vstupů do první plně propojené vrstvy, zde se vyskytuje velké množství rysů. Tyto by mohly tvořit výstupní vrstvu word2vec sítě (stejně jako ID obrázků). Učení by mohlo probíhat na podobném principu, který byl prezentován ve variantách **1.** nebo **2.**.

[6] Reflexe



Obrázek 6.3: Příklad odezvy CNN, jejímž cílem je přiřadit každé kategorii pravděpodobnost výskytu v obraze za předpokladu, že v obraze se má vyskytovat právě jedna kategorie (převzato z [17]).

6.3 Trénování

Pro trénování CNN bylo využito vzorového příkladu na konvoluční neuronové sítě z knihovny TensorFlow[7]. Tento skript využívá 2 *konvoluční* a *max pooling* vrstvy. Jeho datovou sadu tvoří *CIFAR-10* představený v [8][8]. Obsahuje 10 kategorií, přičemž každá kategorie zahrnuje 5 tisíc trénovacích a tisíc testovacích obrázků. Rozměry obrázků jsou pouze 32×32 pixelů.

Při trénování je využito prostředních 24×24 pixelů. Aby bylo dosaženo větší robustnosti, v polovině případů dochází k převrácení obrazu podle svislé osy a náhodně se upravuje kontrast a jas. Pro natrénování modelu bylo použito 130 tisíc kroků, kde jeden krok odpovídá dávce o velikosti 128, celkově tak bylo provedeno více než 300 epoch. Kombinovaná chyba na trénovacích a testovacích datech je přibližně 7 %. Ze všech dostupných 60 tisíc obrázků byla vytvořena datová sada pro word2vec podle varianty **1.**, kde $k = 3$. Datová sada je uložena ve formě *lab* $closest_1$ $closest_2$ $closest_3$, kde $closest_i$ značí i-tou nejpravděpodobnější kategorii, jednotlivé případy (obrázky) jsou odděleny oddělovačem `|||`.

Z důvodů malého počtu kategorií je pro trénování word2vec modelu zvolena velikost výsledného embeddingu 5 a počet negativních vzorků taktéž 5. Tuto síť představuje volba `EVALUATION_TYPE = 7` a `EVALUATION_TYPE = 8` (v druhé variantě dochází k záměně vstupu a výstupu). Jelikož nyní neprobíhá učení s využitím kontextového okna, záměna vstupu a výstupu má vliv na vytvořený embedding.

[7] Příklad je dostupný z https://www.tensorflow.org/tutorials/deep_cnn

[8] Její podrobnější popis je dostupný z http://www.cs.toronto.edu/~kriz/cifar.html

6.4 Vytvořená reprezentace slov

Jelikož neexistuje kritérium, které by posoudilo, jak kvalitní výsledná reprezentace je, pokusím se přiblížit tuto reprezentaci pomocí tabulek zobrazujících blízkost slov. Z Tabulky 6.1 lze pozorovat konkrétní vztah mezi určitými objekty, které se v otázkách odlišení živých a neživých objektů shodují. Naopak v některých kategoriích se liší.

Tabulka 6.1: Výčet nejpodobnějších slov na základě úhlu, který mezi sebou svírají vektorové reprezentace pro naučený model s využitím chybovosti klasifikace CNN.

	Nejpodobnější slovo	
Slovo	**EVALUATION_TYPE**	
	7	**8**
dog	cat	horse
cat	dog	frog
deer	horse	dog
horse	deer	dog
frog	bird	cat
bird	airplane	frog
ship	airplane	truck
airplane	ship	ship
automobile	truck	truck
truck	automobile	automobile

Ještě podrobněji je přiblížena varianta `EVALUATION_TYPE = 7` skrze Tabulky 6.2 a 6.3, která pro každé slovo ukazuje podobnost vektorů, kde 1 vyjadřuje stejné vektory a -1 naprosto odlišné. Kromě zmíněných vztahů mezi živými a neživými objekty lze také pozorovat vazbu mezi kočkou a psem, letadlem a ptákem nebo automobilem a nákladním vozidlem.

Tabulka 6.2: Podobnost reprezentací slov představující neživé objekty pro `EVALUATION_TYPE = 7`.

automobile		**truck**		**ship**		**airplane**	
truck	0.9877	automobile	0.9877	airplane	0.7608	ship	0.7608
ship	0.2725	ship	0.3723	truck	0.3723	bird	0.6377
frog	0.1371	airplane	0.2558	automobile	0.2725	truck	0.2558
airplane	0.1104	frog	0.1228	bird	0.1719	horse	0.1667
horse	0.0870	horse	0.0888	dog	0.1397	frog	0.1288
deer	0.0726	deer	0.0653	cat	0.1208	deer	0.1208
cat	-0.0258	cat	-0.0948	horse	0.0552	automobile	0.1104
dog	-0.0359	dog	-0.1054	frog	-0.0210	cat	-0.1357
bird	-0.2750	bird	-0.1957	deer	-0.3449	dog	-0.1537

Tabulka 6.3: Podobnost reprezentací slov představující živé objekty pro `EVALUATION_TYPE = 7`.

dog		**cat**		**deer**		**bird**	
cat	0.9715	dog	0.9715	horse	0.6744	airplane	0.6377
horse	0.4560	frog	0.5029	bird	0.5978	deer	0.5978
frog	0.2944	horse	0.2886	frog	0.3080	frog	0.5888
deer	0.2542	bird	0.2698	dog	0.2542	cat	0.2698
bird	0.1761	deer	0.2455	cat	0.2455	horse	0.2166
ship	0.1397	ship	0.1208	airplane	0.1208	dog	0.1761
automobile	-0.0359	automobile	-0.0258	automobile	0.0726	ship	0.1719
truck	-0.1054	truck	-0.0948	truck	0.0653	truck	-0.1957
airplane	-0.1537	airplane	-0.1357	ship	-0.3449	automobile	-0.2750

Kapitola 7

Závěr

V rámci diplomové práce byly detailně představeny obě architektury (*CBOW* a *SG*) word2vec a jejich proces učení (v kapitole 2). Aby se tyto daly efektivně používat, je nutné zvolit jednu z optimalizačních technik, které pro velké slovní zásoby výrazně zrychlují proces učení. Při představení učení těchto neuronových sítí byl umožněn alternativní pohled na úpravu vah sítě prostřednictvím vektorů. Aby byl proces učení neuronových sítí více intuitivní, byla nabídnuta paralela mezi magnetickými silami a pohybem vektorů. S využitím sémantické informace mezi státem a jeho hlavním městem bylo pomocí příkladů demonstrováno zachování této vazby v naučených modelech. Dále bylo v kapitole 3 navrženo několik možných úprav včetně možnosti integrace jiných informací.

V části 5 byla implementována architektura SG s využitím NS a tato byla rozšířena o některé navržené úpravy zahrnující obrázky. Výsledná reprezentace po aplikaci těchto úprav není srovnatelná s původní architekturou, a to zejména z důvodu absence dostatečně obecné a rozsáhlé obrazové datové sady. V této je přinejmenším velmi obtížné reprezentovat některá slova. Kombinace textové a obrazové informace v jednom modelu nepřináší užitek. Od experimentů s URL adresami bylo upuštěno po špatných zkušenostech s obrazovými datovými sadami.

Při vytváření slovní zásoby z obrazového korpusu došlo k překvapivému zhoršení úspěšnosti na evaluační sadě po vyfiltrování interpunkce. Původně byla některá slova reprezentována až třemi případy („slovo“, „slovo.“, „slovo,“), vyvstává tak otázka, jestli by zahrnutí interpunkce v textovém korpusu nemělo příznivý vliv na vektorovou reprezentaci slov.

Při použití neuronových sítí ke klasifikačním účelům se často stává, že se ani nesnažíme interpretovat váhy vytvořených sítí, ale síť chápeme pouze jako aproximaci určité funkce $y = f(x)$. Naproti tomu ve word2vec modelech záleží po naučení výhradně na **vahách** a výstup sítě $\mathbf{y}$ na základě vstupu sítě $\mathbf{x}$ pro nás není podstatný, přestože by vypovídal o kookurenci vstupního a výstupního slova.

Myšlenku interpretace vah sítě a učení bez učitele se snaží ještě více zvýraznit kapitola 6 zabývající se možností využití konvoluční neuronové sítě. Výsledky získané tímto způsobem dávají prostor pro další rozšíření a ukazují princip reprezentace objektů bez zásahu člověka. Vzniklá reprezentace dokáže zachytit určité vztahy a může být použita pro separaci živých a neživých objektů na základě vektorové reprezentace slov použité datové sady.

Literatura

[1] An Intuitive Explanation of Convolutional Neural Networks.
URL https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

[2] Vector Representations of Words.
URL https://www.tensorflow.org/versions/r0.10/tutorials/word2vec/

[3] Český národní korpus.
URL http://www.korpus.cz

[4] Abadi, M.; Agarwal, A.; Barham, P.; aj.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015, software available from tensorflow.org.
URL http://tensorflow.org/

[5] Deshpande, A.: A Beginner's Guide To Understanding Convolutional Neural Networks.
URL https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/

[6] Goldberg, Y.; Levy, O.: Word2vec Explained. *arXiv preprint arXiv:1402.3722*, 2014.

[7] Krasin, I.; Duerig, T.; Alldrin, N.; aj.: OpenImages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://github.com/openimages*, 2016.

[8] Krizhevsky, A.; Hinton, G.: Learning multiple layers of features from tiny images. 2009.

[9] Krizhevsky, A.; Sutskever, I.; Hinton, G. E.: ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, editace F. Pereira; C. J. C. Burges; L. Bottou; K. Q. Weinberger, Curran Associates, Inc., 2012, s. 1097–1105.
URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[10] Levy, O.; Goldberg, Y.: Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, 2014, s. 2177–2185.

[11] Levy, O.; Goldberg, Y.; Ramat-Gan, I.: Linguistic Regularities in Sparse and Explicit Word Representations. In *CoNLL*, 2014, s. 171–180.

[12] Lin, T.-Y.; Maire, M.; Belongie, S.; aj.: Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, Springer, 2014, s. 740–755.

[13] Maaten, L. v. d.; Hinton, G.: Visualizing data using t-SNE. *Journal of Machine Learning Research*, ročník 9, č. Nov, 2008: s. 2579–2605.

[14] Mikolov, T.; Chen, K.; Corrado, G.; aj.: Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*, 2013.

[15] Mikolov, T.; Sutskever, I.; Chen, K.; aj.: Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 2013: s. 3111–3119.

[16] Morin, F.; Bengio, Y.: Hierarchical Probabilistic Neural Network Language Model. *AISTATS*, ročník 5, 2005: s. 246–252.
URL http://www.iro.umontreal.ca/~lisa/pointeurs/hierarchical-nnlm-aistats05.pdf

[17] Olah, C.: Conv Nets: A Modular Perspective.
URL http://colah.github.io/posts/2014-07-Conv-Nets-Modular/

[18] Rong, X.: Word2vec Parameter Learning Explained. *arXiv preprint arXiv:1411.2738*, 2014.

[19] Russakovsky, O.; Deng, J.; Su, H.; aj.: ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, ročník 115, č. 3, 2015: s. 211–252, doi:10.1007/s11263-015-0816-y.

Přílohy

Příloha A

Seznam příloh

- DVD se zdrojovými soubory, natrénovanými modely, tímto textem ve formátu pdf a šablonou pro Latex