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

Diplomová práce

Vícejazyčná sémantická podobnost textů

Plzeň 2019 Bc. Michal Tušl

Prohlášení

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

V Plzni dne 14. května 2019

Bc. Michal Tušl

Poděkování

Děkuji Ing. Tomáši Brychcínovi, Ph.D., za vedení mé diplomové práce, za cenné rady a čas, který mi věnoval.

Abstract

This master thesis is focused on unsupervised machine learning methods for cross-lingual semantic textual similarity. For monolingual representation, multiple models were trained on the Wikipedia corpus. For language-independent representation of meaning, monolingual semantic spaces are transformed into a shared space by the linear transformation. We study several linear transformations including *Least Square Transformation*, *Canonical Correlation Analysis* and *Orthogonal Transformation* methods. Including standard word transformation, the thesis also introduces two new approaches, transformation on sentences and transformation of *Paragraph2Vec* models.

Experiments were examined on cross-lingual datasets *SemEval-2017* and *GoranGlavas*. We measure Pearson and Spearman correlation between our methods and human judgements. Presented methods show very promising results.

Abstrakt

Tato práce se zabývá metodami strojového učení bez učitele pro měření sémantické podobnosti textů napříč různými jazyky. Pro monolinguální reprezentaci textu bylo natrénováno několik modelů na korpusu z Wikipedie. Pro vytvoření jazykově nezávislé reprezentace významu jsou monolinguální sémantické prostory transformovány do společného prostoru pomocí lineární transformace. Práce zkoumá lineární transformace za pomocí *metody nejmenších čtverců*, *kanonické korelační analýzy* a *ortogonální transformace*. Kromě standardní transformace na slovech práce představuje dva nové přístupy, a to transformaci na větách a transformaci *Paragraph2Vec* modelu.

Experimenty jsou provedeny na vícejazyčných datasetech *SemEval-2017* a *GoranGlavas* a je měřena Pearsonova a Spearmanova korelace oproti člověku. Zkoumané metody dosahují slibných výsledků na těchto datasetech.

Obsah

1 Úvod

Zpracování přirozeného jazyka – NLP (*Natural Language Processing*) je rozvíjejícím se oborem. Používají ho velké společnosti jako jsou Google nebo Facebook ve svých vyhledávačích, zobrazování příspěvků a dalších funkcích. Lze předpokládat, že se obor bude dále rozšiřovat i do jiných oblastí informačních technologií. Jádrem tohoto zpracování je sémantická analýza, která se zabývá porozumění významu výrazů na různých úrovních jazyka. Do oblasti NLP spadá celá řada úloh, jako: strojový překlad, vyhledávání informací, oprava gramatiky, automatické odpovídání na otázky, sumarizace textů, analýza sentimentu, atd.

Přístupy k řešení těchto úloh se dělí na pravidlové a statistické. Pravidlové přístupy jsou založeny na tom, že jazyk lze popsat sadou lingvistických pravidel. Tento přístup byl nicméně z velké části vytlačen s postupným rozvojem výpočetního výkonu počítačů a internetu, kde lze najít velké množství textu v téměř libovolném jazyce. Díky tomu se začaly rozvíjet statistické přístupy, které překonávají přístupy pravidlové.

Statistické přístupy k řešení NLP úloh se dělí na metody učení s učitelem (*supervised*) a metody učení bez učitele (*unsupervised*). Pro metody učení s učitelem je potřeba mít manuálně označkovaná data od lidí. Tato data je těžké získat z důvodu časové a finanční náročnosti. Metody učení bez učitele nepotřebují žádná označená data, ale zvládají se naučit na dostatečně velkém korpusu dat. Díky tomu se metody učení bez učitele velmi rychle naučí jakýkoliv jazyk. Korpusem, na kterém se tyto metody trénují, může být třeba i Wikipedie, na které byly v této práci trénovány modely pro sémantickou reprezentaci textu.

Tato práce se zabývá metodami učení bez učitele pro sémantickou podobnost textu. Navíc se v této práci metody rozšiřují, aby bylo možné vyjádřit význam textu napříč různými jazyky. Cílem úlohy je tedy určit, jak moc jsou dvě věty v odlišných jazycích významově podobné.

Sémantika libovolného textu je reprezentována vektorem ze sémantického prostoru jazyka. Nejprve je potřeba natrénovat sémantické modely pro každý jazyk zvlášť, k tomu jsou použity metody: *GloVe*, *Word2Vec*, *FastText*, *Paragraph2Vec* a *Skip-thoughts*. Po natrénování těchto metod pro všechny testované jazyky jsou sémantické prostory transformovány do společného prostoru pomocí lineární transformace, které se dělají pomocí transformační matice.

V této práci jsou testovány tři metody pro trénování transformační ma-

tice: metoda nejmenších čtverců, ortogonální transformace a kanonická korelační analýza. Hlavním přínosem této práce jsou pak dva nové způsoby jak trénovat a využít transformační matici.

Testování metod a měření experimentů bylo prováděno na datasetech z konferencí *SemEval* a datasetu *GoranGlavas*. Jedná se o datasety obsahující dvojici vět z anglického a testovacího jazyka. Každé dvojici je přiřazeno číslo určující vzájemnou podobnost obou vět na základě ohodnocení lidmi. Dále se změří korelace mezi hodnocením určeném lidmi a automatickým systémem. Tato korelace určuje, jak moc je automatický systém dobrý.

Metody popsané v této práci byly trénovány na následujících jazycích: angličtina, španělština, italština, arabština, turečtina a chorvatština. Většina těchto metod dosáhla na všech jazycích a dostupných datasetech velmi dobrých výsledků.

Práce je strukturována do několika kapitol. Na začátku práce v kapitole 2 jsou popsány základy, jak získat sémantickou reprezentaci slov. Následuje kapitola 3, kde jsou popsány již konkrétní metody pro získání sémantické reprezentace slov. V kapitole 4 jsou popsány metody, jak vektory slov využít pro získání sémantické reprezentace celých textů, a další metody, jak tuto reprezentaci získat. Kapitola 5 se zabývá porovnáváním významu dvou textů. Metody trénování transformační matice, která transformuje sémantické prostory z jednoho jazyka do druhého, jsou popsány v kapitole 6. Kapitola 7 popisuje způsoby trénování transformační matice. Následují kapitoly 8 a 9, kde jsou popsány trénovací a testovací data, a také provedené experimenty a jejich výsledky.

2 Distribuční sémantika

Distribuční sémantika nabízí poměrně jednoduchý a praktický způsob, jak reprezentovat sémantiku jednotlivých slov v textu. Modely distribuční sémantiky jsou založeny na předpokladu, že význam slova je dán okolím, ve kterém se slovo vyskytlo. Podle (Firth, 1957): „*You shall know a word by the company it keeps*“. Většina metod pro sémantickou reprezentaci je založena na trénování bez učitele, jediné co potřebují, je velké množství textu, ze kterého se lze naučit souvislosti mezi slovy nebo větnými celky. Metody distribuční sémantiky jsou založeny na *distribuční hypotéze*, která říká, že dvě slova vyskytující se ve stejných kontextech by měla být sémanticky podobná. Jinak řečeno, jestliže dvě slova jsou podobně rozmístěna v textu, pak by měla mít stejný význam.

2.1 Sémantické prostory

Význam slov $w \in W$, kde W je množina slov (slovník), je reprezentován jako vektor reálných čísel v mnoharozměrném vektorovém prostoru s dimenzí d, $w \in R^n$. Slova, která se vyskytla ve stejných kontextech, jsou si blízko ve vektorovém prostoru a předpokládá se tedy, že mají podobný význam. Tomuto vektorovému prostoru se říká sémantický prostor (*semantic space*) a vektorům jednotlivých slov sémantický vektor (*semantic vector*).

Většina modelů používá pro výpočet sémantických vektorů jeden z následujících čtyř způsobů: kookurenční matici, model témat, náhodné indexování a neuronovou síť (Brychcín, 2015).

Kookurenční matice počítá výskyty slov v kontextech. Tyto metody tedy obsahují matici $\boldsymbol{M}$ s rozměry $|W| \times |D|$, kde $|W|$ je počet slov a $|D|$ je počet kontextů, v kterých se slovo může vyskytnout (např. počet dokumentů). Do této matice se ukládá počet výskytů slova w v kontextu c, obvykle s nějakou váhou. Tato váha se může vypočíst například pomocí TF–IDF (*term frequency – inverse document frequency*) nebo PMI (*pointwise mutual information*) (Church – Hanks, 1990). Nevýhodou těchto modelů je příliš velká dimenze vektorů, které jsou velmi často řídké (většinu hodnot tvoří nuly). Proto se často používá některý z algoritmů pro redukci dimenzionality, například singulární rozklad, viz kapitola 6.1. Mezi modely s kookurenční maticí patří:

- HAL – *Hyperspace Analogue to Language* (Lund – Burgess, 1996),

- LSA – latentní sémantická analýza (*Latent Semantic Analysis*) (Landauer aj., 1998),
- ESA – explicitní sémantická analýza (*Explicit Semantic Analysis*) (Gabrilovich – Markovitch, 2007),
- GloVe – *Global Vectors* (Pennington aj., 2014).

Metody využívající model témat jsou většinou založeny na *Bag-of-Words* hypotéze, podle níž odhadují skrytá (neznámá) témata z textu. Význam textu je obvykle reprezentován jako vektor témat, ale lze jej použít i pro reprezentaci slov. Mezi tyto metody patří LSA, PLSA – pravděpodobnostní latentní sémantická analýza (*Probabilistic Latent Semantic Analysis*) (Hofmann, 1999) a LDA – latentní dirichletova alokace (*Latent Dirichlet Allocation*) (Blei aj., 2003).

Modely založené na náhodném indexování jsou založeny na distribuční hypotéze a používají lokální kontext pro reprezentaci významu slov. Výhodou je, že se zde předem určí dimenze vektorů (v řádu stovek až tisíců). Na začátku je do vektorů náhodně uloženo několik málo hodnot -1 a $+1$, metoda tak předpokládá, že vektory jsou na sebe kolmé. Při výskytu slova v kontextu se pak tento vektor pro daný kontext přičte k vektoru slova. Na tomto principu funguje metoda RI (*Random Indexing*) (Sahlgren, 2005).

Nejnovějším druhem metod jsou modely, které pro trénování vektorů využívají neuronových sítí. Tyto metody se od sebe často velmi liší. Zároveň lze říct, že v poslední době právě metody s neuronovými sítěmi dosahují nejlepších výsledků. Mezi tyto modely patří metoda *Word2Vec* se *Skip-gram*, i CBOW (*Continuous Bag-of-Word*) modelem (Mikolov aj., 2013a), viz kapitola 3.2.

Podobnost slov se dá měřit podle vzdálenosti jejich sémantických vektorů. Tuto vzdálenost lze měřit euklidovskou (kapitola 5.1), nebo kosínovou (kapitola 5.2) vzdáleností.

2.2 Typy kontextů v sémantických modelech

Kontexty, které představují jednotlivé dimenze v sémantickém prostoru, pro něž se určuje míra asociace se slovem, se dělí na dva druhy: *globální kontext* a *lokální kontext*. Globální kontexty obvykle používají modely založené na *Bag-of-Words* hypotéze. *Bag-of-Words* hypotéza je založena na porovnávání množin, kde pořadí prvků nehraje roli. Například množina *a,a,b,b,c* a množina *c,a,b,a,b* jsou ekvivalentní. Praktické využití má především u porovnávání celých dokumentů, tedy pokud dva dokumenty obsahují stejná

slova, jsou dokumenty podobné. Nevýhoda je u krátkých částí textu, jako jsou třeba věty. V *Bag-of-Words* hypotéze jsou věty „Kočka je větší než pes.“ a „Pes je větší než kočka.“ totožné, neboť pořadí slov zde nehraje roli. Jeden kontext odpovídá jednomu dokumentu v trénovacích datech. Tímto dokumentem může být věta, odstavec, nebo i větší kus textu (článek Wikipedie). Tyto modely jsou schopny zachytit větší rozsah závislostí mezi slovy. Například dokument o autech bude pravděpodobně obsahovat slova jako „motor“ nebo „řidič“. Tato slova pak budou významově podobná. Mezi metody používající globální kontext patří LSA, PLSA, ESA a LDA.

Modely využívající lokální kontexty nepotřebují mít text rozdělený na dokumenty, ale stačí mít dostatečné množství textu. Tyto modely berou v úvahu i uspořádání slov. Na rozdíl od modelů využívající globální kontexty jsou tyto modely schopné nalézt vhodné substituty pro slova v zadaných kontextech. Například ve větě „Pes je zvíře.“ může být slovo „pes“ nahrazeno za slovo „kočka“ (Brychcín, 2015). Některé modely ještě rozlišují levý a pravý kontext. Levý kontext jsou slova vyskytující se vlevo od slova jehož význam se určuje. Naopak pravý kontext jsou slova napravo od cílového slova.

V současnosti nejúspěšnější sémantické modely používají lokální kontext, jelikož zachovává syntaktické informace o slovech. Mezi tyto metody patří RI, HAL, *Glove*, *Word2Vec* (*Skip-gram* i CBOW) model. U novějších sémantických modelů využívající lokální kontext, jako jsou *GloVe* a *Word2Vec*, lze najít zajímavé slovní analogie[1] (Mikolov aj., 2013a; Linzen, 2016). Tyto slovní analogie jsou lineární závislosti mezi několika slovy, která tyto analogie tvoří. Například u dvojice slov „muž“ a „žena“ lze analogicky v případě slova „král“ doplnit slovo „královna“. U některých novějších modelů využívající lokální kontexty lze slovo „královna“ dopočítat, pokud u nich bude platit:

$$\text{vektor}(muž) - \text{vektor}(žena) + \text{vektor}(král) = \text{vektor}(královna). \quad (2.1)$$

[1]Aplikace s vizualizací několika slovních analogií: `https://lamyiowce.github.io/word2viz/`

3 Sémantická reprezentace slov

V této kapitole jsou popsány sémantické modely pro vytváření slovních vektorů, které byly využity v této diplomové práci. Tyto modely byly vybrány z toho důvodu, že v současnosti patří mezi nejlepší (dosahují nejlepších výsledků). Metody pro sémantickou reprezentaci textu jsou popsány v kapitole 4.

3.1 GloVe

GloVe (*Global Vectors*) (Pennington aj., 2014) je metoda učení bez učitele pro získání vektorové reprezentace slov. Název *Global Vectors* je z toho důvodu, že tato metoda určuje vektory podle statistik výskytu slov v celém korpusu.

Nejprve se definuje matice $\boldsymbol{M}$, do které se ukládají spoluvýskyty jednotlivých slov (kookurenční matice). Matice má tedy rozměry $|W| \times |W|$, kde $|W|$ je velikost slovníku. Zároveň se definuje velikost okénka r (obvykle 3–20). Nyní se prochází celý korpus a pro každé slovo w_i, jemuž odpovídá řádek i v matici $\boldsymbol{M}$, se zvyší hodnota o 1 ve sloupcích j, kde j jsou indexy slov v okénku r nalevo a napravo od slova w_i. $\sum_{k=1}^{|W|} m_{ik}$ je počet, kolikrát se nějaké slovo vyskytlo v kontextu slova w_i.

Po vypočtení kookurenční matice se vytvoří matice pravděpodobností $\boldsymbol{P}$, s rozměry $|W| \times |W|$. Pro každý prvek matice se vypočte hodnota následujícím vzorcem:

$$p_{ij} = P(j|i) = \frac{m_{ij}}{\sum_{k=1}^{|W|} m_{ik}}. \tag{3.1}$$

Hodnota p_{ij} vyjadřuje pravděpodobnost, že slovo w_j se vyskytlo v kontextu slova w_i.

Vzhledem k tomu, že matice $\boldsymbol{P}$ a $\boldsymbol{M}$ jsou velmi velké, ale řídké (nulové hodnoty tvoří 75–95 % matice) (Pennington aj., 2014), tak je žádoucí redukovat dimenzi. Metoda *GloVe* pomocí log-bilineárního regresního modelu odhadne cílové vektory slov s volitelnou dimenzí d. Pro vytvoření vektorů slov potřebuje metoda dvě matice $\boldsymbol{V}$ a $\boldsymbol{C}$, v matici $\boldsymbol{V}$ jsou odhadované vektory slov a v matici $\boldsymbol{C}$ se nachází vektory slov pro slova, která se vyskytla v kontextu jiného slova. Cílem regrese je, aby platila následující rovnice:

$$\boldsymbol{v}_i^\top \boldsymbol{c}_k \approx \log(1 + m_{ik}), \tag{3.2}$$

kde $\boldsymbol{v}_i$ jsou odhadovaný vektor slova w_i a $\boldsymbol{c}_k$ je kontextový vektor slova w_k.

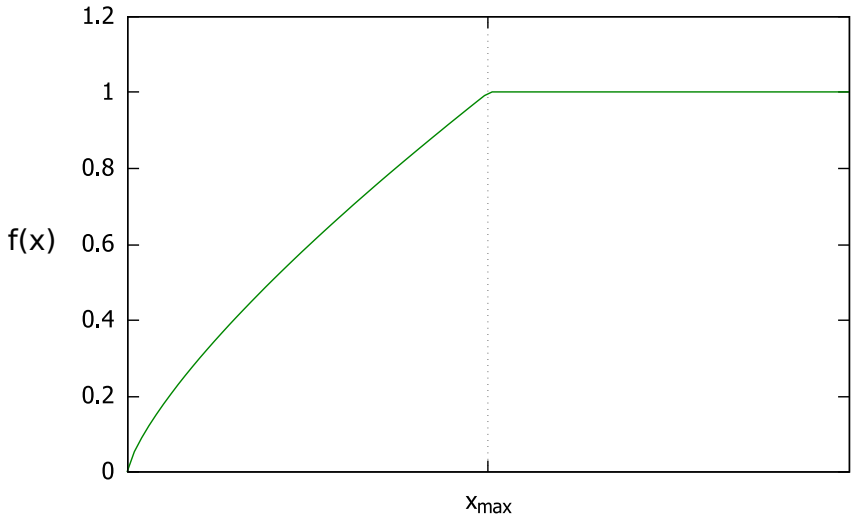
Cenová funkce, kterou model optimalizuje vypadá následovně:

$$J = \sum_{i=1}^{|W|} \sum_{j=1}^{|W|} f(m_{ij})(\boldsymbol{v}_i^\top \boldsymbol{c}_j - \log m_{ij})^2, \tag{3.3}$$

kde $f(m_{ij})$ je váhová funkce (Obrázek 3.1). Tato funkce snižuje vliv často se vyskytujících slov, které nemají příliš velký význam, ale model by jim přiděloval velkou váhu. Funkce zároveň snižuje důležitost vzácných spoluvýskytů slov. Hodnota této funkce pro parametr x se spočte:

$$f(x) = \begin{cases} (\frac{x}{x_{max}})^\alpha & \text{pokud } x < x_{max} \\ 1 & \text{jinak.} \end{cases} \tag{3.4}$$

Hodnota α je nastavena na $0,75$, což se ukázalo jako nejlepší hodnota (Pennington aj., 2014). Hodnota x_{max}, která představuje hranici, pro kterou se snižuje váha spoluvýskytů, je 100 (Pennington aj., 2014).

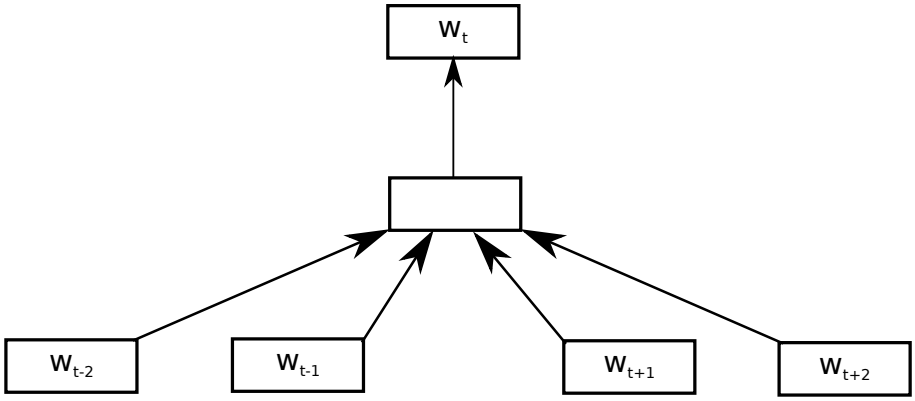


Obrázek 3.1: Váhová funkce $f(x)$.

Po zvoleném počtu iterací, během kterých se optimalizují vektory slov, lze zahodit matice $\boldsymbol{M}, \boldsymbol{P}, \boldsymbol{C}$, neboť už nejsou potřeba. Slovní vektory jsou uloženy v matici $\boldsymbol{V}$.

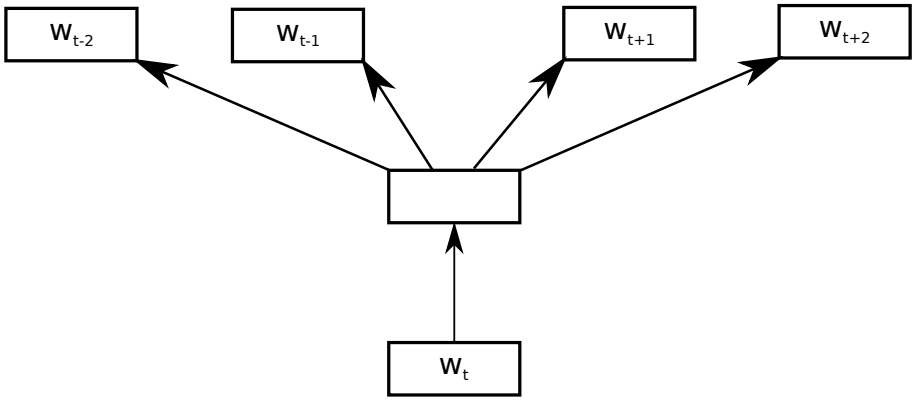
3.2 Word2Vec

Metoda *Word2Vec* (Mikolov aj., 2013a) je založená na jazykových modelech. Jazykové modely předpovídají, které slovo se vyskytne ve větě jako další, za podmínky předešlých slov. Příkladem je například sousloví: „Šel do lesa na ...“, do této věty se s větší pravděpodobností doplní slova „houby“, nebo „dřevo“, než slovo „oběd“. Metoda tedy využívá lokální kontext a trénuje se na neuronových sítích. *Word2Vec* má dvě různé architektury: CBOW (z angl. *Continuous Bag-of-Words*) a *Skip-gram*. CBOW předvídá jedno slovo na základě okolních slov (kontext), viz Obrázek 3.2.



Obrázek 3.2: *Continuous Bag-of-Words* model.

Skip-gram se snaží předpovědět okolní slova ve větě na základě jednoho slova ve větě, viz Obrázek 3.3.



Obrázek 3.3: *Skip-gram* model.

3.2.1 Základní principy Word2Vec

Nejprve je potřeba vytvořit dvě matice $\boldsymbol{V}$ a $\boldsymbol{C}$ o rozměrech $|W| \times d$, kde $|W|$ je velikost slovníku (počet slov, pro které se budou počítat vektory) a d je cílová dimenze těchto vektorů. V matici $\boldsymbol{V}$ jsou vektory slov $\boldsymbol{v}_1, ..., \boldsymbol{v}_n$, které se snažíme předpovídat. V matici $\boldsymbol{C}$ jsou kontextové vektory okolních slov $\boldsymbol{c}_1, ..., \boldsymbol{c}_n$, ze kterých se předpovídá cílové slovo. Obě tyto matice se náhodně inicializují.

V metodě *Skip-gram* se pro každou větu obsahující slova: $w_1, w_2, ..., w_T$, maximalizuje následující funkce:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-r\leq j\leq r, j\neq 0}\log p(w_{t+j}|w_t), \tag{3.5}$$

kde r je velikost okénka, ve kterém se předpovídají slova. Velikost r ovlivňuje přesnost předpovědí, ale zároveň i trénování metody (vyšší r, delší trénování). Obvyklá volba velikosti okénka je 3–20. Pravděpodobnost $p(w_{t+j}|w_t)$ je standardně definovaná jako *softmax* funkce (Mikolov aj., 2013c):

$$p(w_t|w_c) = \frac{\exp(\boldsymbol{v}_{w_t}^{\top}\boldsymbol{c}_{w_c})}{\sum_{w\in W}\exp(\boldsymbol{v}_{w}^{\top}\boldsymbol{c}_{w_c})}, \tag{3.6}$$

kde $\boldsymbol{v}_{w_t}$ jsou cílové slovní vektory. Matice $\boldsymbol{C}$ již není potřeba a lze ji odstranit.

V metodě *Continuous Bag-of-Words* se pouze prohodí kontextová slova a cílové slovo. Metoda tedy pro každou větu obsahující slova $w_1, w_2, ..., w_T$, maximalizuje následující funkci:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-r\leq j\leq r, j\neq 0}\log p(w_t|w_{t+j}), \tag{3.7}$$

kde r je velikost okénka, ve kterém jsou slova, která se berou za kontext slova w_t.

Výpočet *softmax* funkce pro každé slovo je v praxi nepoužitelné, protože časová náročnost roste proporcionálně s velikostí slovníku, který je obvykle velmi velký (10^5–10^6). Proto se používá jedna z následujících aproximací: Hierarchický softmax (z angl. *Hierarchical softmax*) (Morin – Bengio, 2005) a Negativní samplování (z angl. *Negative sampling*) (Mikolov aj., 2013c).

3.2.2 Hierarchický softmax

Hierarchický softmax je jedna z výpočetně efektivních aproximací původní softmax funkce. Největší výhodou této metody je, že není potřeba spočítat softmax pro všechna slova, ale pouze pro něco okolo $\log_2(|W|)$ slov.

Hierarchický softmax reprezentuje výstupní vrstvu neuronové sítě jako binární strom, kde slova jsou listy stromu, a každý uzel představuje relativní pravděpodobnost jeho potomků. Každé slovo w je dosažitelné z kořene stromu, cestou přes vnitřní uzly. Uzly na této cestě se značí: $n(w, j)$, kde j je pořadové číslo uzlu na cestě z kořene stromu $n(w, 1)$ do cílového slova $n(w, L(w))$, kde $L(w)$ je délka cesty. Hierarchický softmax $p(w_t|w_c)$ je tedy definován následovně:

$$p(w_t|w_c) \approx \prod_{j=1}^{L(w_t)-1} \sigma((n(w_t, j+1) = \text{levé ? } 1 : -1) \cdot \boldsymbol{v}_{n(w_t,j)}^{\top} \boldsymbol{c}_{w_c}), \tag{3.8}$$

kde $\sigma(x) = 1/(1 + \exp(-x))$.

Zrychlení a přesnost výpočtu ovlivňuje struktura použitého stromu (Mnih – Hinton, 2009).

3.2.3 Negativní samplování

Negativní samplování nahrazuje výraz $\log p(w_t|w_c)$. Princip této techniky je snížit počet slov, s kterými se slovo w_t porovnává. Podle velikosti slovníku se zvolí počet negativních vzorků k (obvykle 5–20). Pro každé slovo w_t se vyberou náhodná slova $w_1, ..., w_k$ z unigramového rozdělení slov $P_n(w)$. U těchto slov je nepravděpodobné, že se vyskytnou v kontextu slova w_c. Výpočet $\log p(w_t|w_c)$ se tak nahradí následujícím výrazem:

$$\log \sigma(\boldsymbol{v}_{w_t}^{\top} \boldsymbol{c}_{w_c}) + \sum_{w_i \sim P_n(w)} \log \sigma(-\boldsymbol{v}_{w_i}^{\top} \boldsymbol{c}_{w_c}), \tag{3.9}$$

kde $\sigma(\boldsymbol{v}_{w_t}^{\top} \boldsymbol{c}_{w_c})$ se rovná:

$$\sigma(x) = \frac{1}{1 + \exp(-x)}. \tag{3.10}$$

3.3 FastText

Metoda *FastText* (Bojanowski aj., 2017) je založená na stejných principech jako metoda *Word2Vec*, též má dvě architektury pro trénování modelu (*Skip-gram* a CBOW) a pro urychlení výpočtu používá Negativní samplování.

Hlavní přínos této metody je vytváření slovních vektorů pro slova, která nejsou ve slovníku a metoda je při trénování vůbec neviděla. Toho je docíleno tím, že se metoda naučí nejen vektory pro slova, ale i pro n-gramy složené ze znaků jednotlivých slov. Standardně metoda používá n-gramy o 3–6 znacích.

Při uvážení, kolik existuje možných kombinací, by byl slovník n-gramů příliš velký. Proto se používá hashovací funkce, která každému n-gramu přiřadí číslo v rozsahu $\langle 0; |G|)$, kde $|G|$ je parametr, kolik vektorů pro n-gramy se bude používat (obvykle 500 000 – 2 000 000) (Bojanowski aj., 2017).

Při trénování si metoda rozdělí každé slovo na n-gramy, pro které spočte hash, jenž udává index do tabulky s vektory n-gramů. Tyto vektory pak trénuje stejně jako vektory slov. Výstupem metody je matice $\boldsymbol{V}$, obsahující vektory slov a matice $\boldsymbol{G}$, obsahující vektory n-gramů.

Vektor pro slovo nenacházející se ve slovníku, se pak zjistí následujícím způsobem. Nejprve se ze slova vytvoří všechny možné n-gramy o dané velikosti. Pro všechny tyto n-gramy se spočte jejich hash, udávající index do matice $\boldsymbol{G}$. Nyní se sečtou všechny vektory těchto n-gramů a vypočte se průměrný vektor, který reprezentuje význam neznámého slova.

4 Sémantická reprezentace textů

V předchozí kapitole jsou vysvětleny konkrétní metody pro získání vektorové reprezentace slov. V této kapitole jsou popsány metody, jak tyto vektory slov využít pro vyjádření sémantické reprezentace textu (vět, odstavců, dokumentů).

Metody se dělí na dvě skupiny, první jsou metody, kde nezáleží na pořadí slov v textu, pro který se sestavuje vektor. Mezi tyto metody patří *Bag-of-word* (kapitola 4.1) a lineární kombinace (kapitola 4.2). Do druhé skupiny patří metody, které berou v úvahu pozice jednotlivých slov v textu. Sestavování těchto vektorů je již složitější, jedná se tedy o komplexní metody *Paragraph2Vec* (kapitola 4.3) a *Skip-thoughts* (kapitola 4.4).

4.1 Bag-of-words reprezentace

Bag-of-words reprezentace je velmi základní metoda pro vyjádření významu textu, která se používá především v klasifikačních úlohách. V úloze pro sémantickou podobnost textů nedosahuje příliš dobrých výsledků.

Na začátku se z trénovacího korpusu vytvoří slovník, který obsahuje slova vyskytující se v korpusu. Těchto slov je obecně n, tedy i velikost slovníku je n. Dále se vytvoří vektor $\boldsymbol{v}$ s dimenzí n, kde každý prvek vektoru představuje jedno slovo ve slovníku. Vektor je inicializován na hodnoty 0. Nyní se prochází text, pro který se sestavuje vektor. Pokud text obsahuje slovo w_i, tak se změní hodnota vektoru na pozici i na 1. Na konci se tak získá vektor obsahující hodnoty 0 a 1. Těmto vektorům, založených na *Bag-of-words* se též často říká *one-hot* vektor.

Nevýhodou tohoto postupu je velmi vysoká dimenze vektoru, která záleží na slovníku. Obvykle v řádu 10^4–10^6. Na druhou stranu není potřeba udržovat vektory pro jednotlivá slova.

4.2 Lineární kombinace

Význam textu lze vyjádřit i vektorem o menší dimenzi. K tomu jsou potřeba vektory jednotlivých slov, k nimž se lineární kombinací vypočte vektor pro celý text.

Pro text (dokument) obsahující slova $w_1, ..., w_n$ lze spočítat vektor, pokud existuje matice obsahující vektory jednotlivých slov $\boldsymbol{v}_{w_1}, ..., \boldsymbol{v}_{w_n}$ s dimenzí d (Brychcín, 2018). Cílový vektor dokumentu pak bude mít též dimenzi d. Tento vektor $\boldsymbol{v}_D$ lze spočíst následovně:

$$\boldsymbol{v}_D = \frac{1}{n} \cdot \sum_{i=1}^{n} \boldsymbol{v}_{w_i}. \tag{4.1}$$

V této metodě již není nevýhoda příliš velká dimenze vektoru, protože závisí na dimenzi vektorů, které se natrénovaly některou z metod pro sémantickou reprezentaci slov, viz kapitola 3. Nevýhodou je, že vektor nemusí přesně reprezentovat daný dokument. Pokud je v textu například mnoho předložek, zájmen, a dalších nevýznamových slov, může se stát, že tyto slova převáží vektory významových slov v dokumentu. Vektor dokumentu je pak posunutý od skutečného významu. Proto se používá vážení vektorů, kdy každý vektor přispívá k cílovému vektoru s určitou váhou, následně se pak cílový vektor vydělí součtem vah. Vektor dokumentu je tak vyjádřen jako vážený průměr vektorů slov vyskytujících se v dokumentu. Vzorec 4.1 se tedy upraví následovně:

$$\boldsymbol{v}_D = \frac{1}{\sum_i^n \lambda_i} \cdot \sum_{i=1}^{n} \lambda_i \boldsymbol{v}_{w_i}, \tag{4.2}$$

kde λ jsou váhy vektorů.

Váhy λ lze vypočíst několika způsoby, nejpoužívanější je *idf* vážení. Hodnota *idf* vyjadřuje důležitost slova napříč všemi dokumenty. Pokud se slovo vyskytuje ve všech dokumentech, pak není příliš důležité a jeho *idf* hodnota je rovna 0. Naopak jestliže je slovo velmi vzácné a vyskytuje se ve velmi málo dokumentech, je jeho váha vysoká. Hodnota *idf* pro slovo w se vypočte:

$$idf(w) = \log \frac{n}{f_w}, \tag{4.3}$$

kde f_w je počet dokumentů, v kterých se slovo w vyskytlo.

Výpočet *idf* je potřeba udělat dříve než při skládání vektoru pro dokument a zároveň na co největším množství dokumentů. Obvykle se *idf* počítá na korpusu, na kterém se trénovaly vektory pro slova.

Vektory vypočtené váženou lineární kombinací již dosahují lepších výsledků, jelikož nevýznamová slova přispívají do finálního vektoru velmi málo.

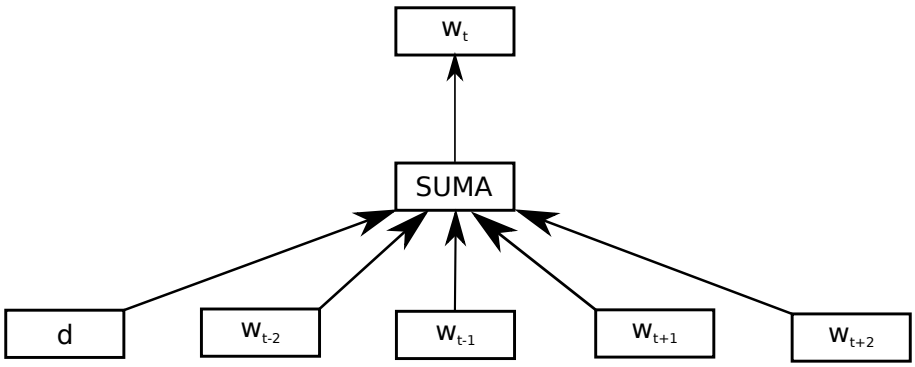
4.3 Paragraph2Vec

Metoda *Paragraph2Vec* (Le – Mikolov, 2014) (někdy označována jako *doc2vec*), je algoritmus učení bez učitele pro získání vektorové reprezentace libovolně dlouhého textu. Může se jednat o věty, odstavce nebo dokonce celé dokumenty. Cílem metody je získat pro různé texty sémantický vektor s určitou dimenzí, podobně jako metoda *Word2Vec*. Tento vektor je trénován pro předvídání slov v dokumentu. Výhodou této metody oproti lineární kombinaci slovních vektorů je, že bere v úvahu pořadí slov v dokumentu. Dokumenty obsahující stejná slova, ale v jiném pořadí, tak budou mít odlišné vektory.

Metoda používá dva druhy vektorů, slovní vektory a vektory dokumentů. Vektory dokumentů jsou unikátní pro každý dokument, zatímco vektory slov jsou stejné ve všech dokumentech. Oba druhy vektorů, slovní i dokumentové, jsou trénovány neuronovou sítí s použitím stochastického gradientního sestupu a zpětnou propagací vah (Rumelhart aj., 1986). Při vytváření vektoru pro nový (dosud neviděný) dokument jsou použity fixní slovní vektory pro trénování dokumentového vektoru. Tento vektor se postupně aktualizuje s každou iterací, do té doby, než dokonverguje k cílovému řešení.

Než bude možné vytvářet vektory pro dosud neviděné dokumenty, je potřeba model nejprve natrénovat. Na začátku se vytvoří dvě matice $\boldsymbol{V}$ a $\boldsymbol{D}$. Matice $\boldsymbol{V}$ obsahuje vektory slov a matice $\boldsymbol{D}$ obsahuje vektory dokumentů. Obě tyto matice jsou na začátku náhodně inicializovány. Velikost matice $\boldsymbol{D}$ je $|D| \times d$, kde $|D|$ je počet dokumentů v trénovacím korpusu a d je velikost cílové dimenze.

Trénování modelu probíhá podobně jako v metodě *Continuous Bag-of-Words* u *Word2Vec*, která je popsána v kapitole 3.2.1. Jediný rozdíl je, že zde se navíc pro predikci slova používá dokumentový vektor, se kterým se počítá stejně jako se slovním vektorem 4.1.



Obrázek 4.1: Model pro učení vektorů dokumentů.

V této metodě se tedy maximalizuje následující cenová funkce:

$$J = \frac{1}{T}\sum_{t=1}^{T}\sum_{-r\leq j\leq r, j\neq 0} \log p(w_t|w_{t+j}; \boldsymbol{d}), \tag{4.4}$$

kde r je velikost okénka a $\boldsymbol{d}$ je vektor pro dokument, ve kterém se vyskytují slova $w_1, w_2, ..., w_T$.

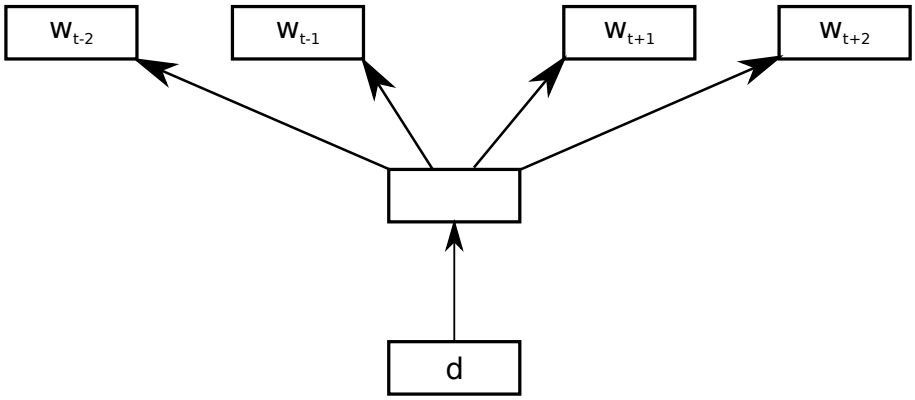
Nyní se sečtou všechny vektory z kontextu spolu s dokumentovým vektorem:

$$\boldsymbol{x} = \boldsymbol{d} + \sum_{-r\leq j\leq r, j\neq 0} \boldsymbol{v}_{w_{t+j}}, \tag{4.5}$$

a výsledný vektor $\boldsymbol{x}$ je pak předán na vstupní vrstvu neuronové sítě, která pomocí stochastického gradientního sestupu a zpětné propagace vah, vypočte aktualizace parametrů modelu. Parametry modelu jsou matice $\boldsymbol{V}$ a $\boldsymbol{D}$ a též softmax váhy ve výstupní vrstvě neuronové sítě.

Výpočet vektoru pro nový dokument, který nebyl obsažen v trénovacích datech se vypočte též pomocí gradientního sestupu. Nejprve se přidá nový vektor do matice $\boldsymbol{D}$. Následně probíhá trénování, stejně jako pro trénovací dokumenty, ale slovní vektory a softmax váhy se již nemění.

Existuje též varianta výpočtu dokumentových vektorů podobná metodě *Skip-gram*, která se nazývá *Distributed Bag-of-Words*. V této verzi je na vstupu pouze vektor dokumentu a metoda se snaží předpovídat náhodná slova z dokumentu, viz Obrázek 4.2. Výhodou tohoto modelu je, že není potřeba ukládat slovní vektory, ale pouze softmax váhy, proto je paměťově úspornější.



Obrázek 4.2: *Distributed Bag-of-Words* model.

Cenová funkce tedy vypadá následovně:

$$J = \frac{1}{T} \sum_{t=1, w \in D}^{T} \log p(\boldsymbol{d}|w), \tag{4.6}$$

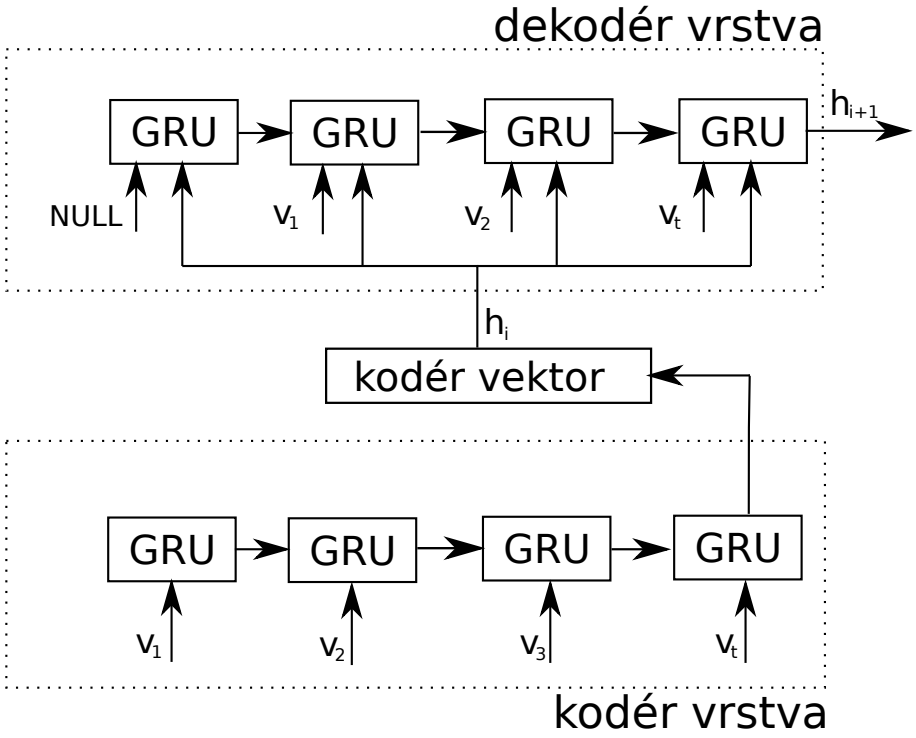
kde T je počet náhodně vybraných slov a w je náhodné slovo z dokumentu D.

Podle (Le – Mikolov, 2014) je dobré kombinovat vektory dokumentů z obou variant, neboť takto lze získat stabilně dobré výsledky pro mnoho různých úloh.

4.4 Skip-thoughts

Metoda *Skip-thoughts* (Kiros aj., 2015) je metoda učení bez učitele pro generování sémantických vektorů pro věty, tyto vektory se v metodě nazývají *Skip-thought vectors*. Metoda vychází z modelu *Skip-gram* (viz kapitola 3.2.1), kde se používalo slovo pro predikci kontextu (okolních slov). V tomto případě se používá celá věta pro predikci okolních vět. Metoda tedy k učení potřebuje souvislý text, nikoliv samostatné věty. V (Kiros aj., 2015) byl použit pro trénování *BookCorpus* dataset, který obsahuje text několika knih. Nicméně, i text Wikipedie obsahuje dostatečně souvislý text pro trénování metody.

Skip-thoughts je založen na modelu kodér-dekodér (z angl. *encoder-decoder*), který se používá v neuronových sítích, například pro strojový překlad. Kodér převádí slova ve větě na vektor a dekodér predikuje okolní věty. Kodér i dekodér jsou dvě rekurentní neuronové sítě (RNN – z angl. *Recurrent Neural Network*) (viz Obrázek 4.3) s GRU (*Gated Reccurent Unit*) (Chung aj., 2014).



Obrázek 4.3: Rekurentní neuronová síť pro zakódování a dekódování větných vektorů.

Každá věta s_i obsahuje slova $w_1^i, ..., w_n^i$. Pro každou větu s_i se přiřadí předchozí a následující věta (s_{i-1}, s_{i+1}). S každým průchodem neuronovou sítí se vytváří nový stav ve skryté vrstvě $\boldsymbol{h}_i^t$, který reprezentuje význam prvních t slov ve větě s_i. Stavy skryté vrstvy po n průchodech pak odpovídají vektoru pro danou větu. Pro zakódování věty se iteruje přes následující rovnice:

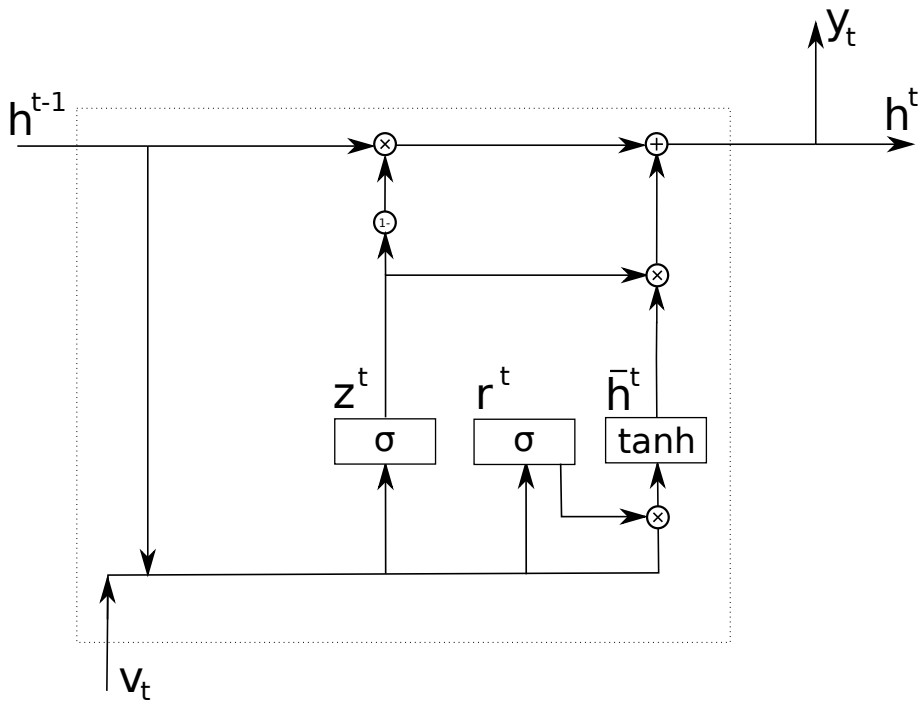
$$\boldsymbol{r}^t = \sigma(\boldsymbol{W}_r \boldsymbol{v}_t + \boldsymbol{U}_r \boldsymbol{h}^{t-1}), \tag{4.7}$$

$$\boldsymbol{z}^t = \sigma(\boldsymbol{W}_z \boldsymbol{v}_t + \boldsymbol{U}_z \boldsymbol{h}^{t-1}), \tag{4.8}$$

$$\bar{h}^t = \tanh(\boldsymbol{W}\boldsymbol{v}_t + \boldsymbol{U}(\boldsymbol{r}^t \cdot \boldsymbol{h}^{t-1})), \tag{4.9}$$

$$\boldsymbol{h}^t = (1 - \boldsymbol{z}^t) \cdot \boldsymbol{h}^{t-1} + \boldsymbol{z}^t \cdot \bar{h}^t, \tag{4.10}$$

kde $\bar{h}^t$ je aktualizace stavu v čase t, $\boldsymbol{z}^t$ je *update gate* a $\boldsymbol{r}^t$ je *reset gate*, viz Obrázek 4.4. Matice $\boldsymbol{W}$ a $\boldsymbol{U}$ jsou váhy neuronové sítě a $\boldsymbol{v}_t$ je vektor na vstupní vrstvě (vektor pro slovo w_t).



Obrázek 4.4: *Gated reccurent unit* pro zakódování vět.

Výpočet v dekodér vrstvě je podobný jako v kodér vrstvě. Jediným rozdílem jsou tři matice $\boldsymbol{C}_z, \boldsymbol{C}_r, \boldsymbol{C}$, viz Obrázek 4.5. Ty jsou použity k regularizaci *reset gate*, *update gate* a výpočtu skrytého stavu. V modelu jsou použity dva dekodéry, jeden pro výpočet následující věty s_{i+1} a jeden pro výpočet předchozí věty s_{i-1}. Každý dekodér používá jiné matice parametrů s výjimkou matice $\boldsymbol{V}$, která obsahuje slovní vektory v. Na vstupu obou je výstup z kodér vrstvy h_i. Výpočet vektoru pro následující větu $\boldsymbol{h}_{i+1}$ se tak vypočte iterací přes následující rovnice:

$$\boldsymbol{r}^t = \sigma(\boldsymbol{W}_r \boldsymbol{v}_{t-1} + \boldsymbol{U}_r \boldsymbol{h}^{t-1} + \boldsymbol{C}_r \boldsymbol{h}_i), \tag{4.11}$$

$$\boldsymbol{z}^t = \sigma(\boldsymbol{W}_z \boldsymbol{v}_{t-1} + \boldsymbol{U}_z \boldsymbol{h}^{t-1} + \boldsymbol{C}_z \boldsymbol{h}_i), \tag{4.12}$$

$$\bar{h}^t = \tanh(\boldsymbol{W} \boldsymbol{v}_{t-1} + \boldsymbol{U}(\boldsymbol{r}^t \cdot \boldsymbol{h}^{t-1}) + \boldsymbol{C} \boldsymbol{h}_i), \tag{4.13}$$

$$\boldsymbol{h}_{i+1}^t = (1 - \boldsymbol{z}^t) \cdot \boldsymbol{h}^{t-1} + \boldsymbol{z}^t \cdot \bar{h}^t. \tag{4.14}$$

Stejným způsobem se vypočte i vektor $\boldsymbol{h}_{i-1}$ pro předcházející větu s_{i-1}. Nyní lze aktualizovat vektor pro slovo w_t^{i+1} za použití předchozích slov ve větě a kodér vektoru. Vektor se přenastaví tak, aby bylo splněno následující:

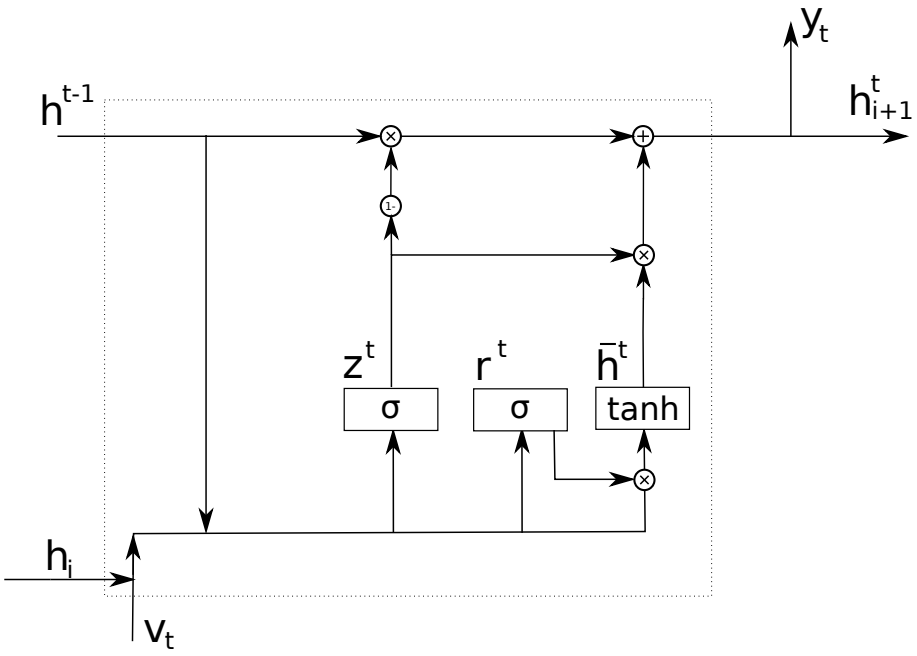
$$P(w_t^{i+1}|w_{<t}^{i+1}, \boldsymbol{h}_i) \propto \exp(\boldsymbol{v}_{w_t^{i+1}} \boldsymbol{h}_{i+1}^t). \tag{4.15}$$

Cílem modelu je optimalizovat funkci pro každou sekvenci vět (s_{i-1}, s_i, s_{i+1}). Tato optimalizační funkce je následující:

$$J = \sum_t \log P(w_t^{i+1}|w_{<t}^{i+1}, \boldsymbol{h}_i) + \sum_t \log P(w_t^{i-1}|w_{<t}^{i-1}, \boldsymbol{h}_i). \tag{4.16}$$

Metoda neustále iteruje přes celý trénovací korpus, aby se co nejlépe optimalizovaly váhy neuronové sítě.

Pro vytvoření vektoru pro dosud neviděnou větu, zůstávají váhy ve všech maticích stejné a pouze se iteruje přes slova ve větě, jejichž vektory jsou vstupními vektory kodér vrstvy.



Obrázek 4.5: *Gated reccurent unit* pro predikování vektorů pro okolní věty.

Metoda *Skip-thoughts* umožňuje pro vytváření vektorů na dosud neviděných datech provést expanzi slovníku. Expanzi slovníku je dobré provádět, pokud v testovacích datech jsou slova, která nebyla v trénovacích datech. Nicméně vzhledem k tomu, že metoda *Skip-thoughts* je navržena pro vytváření dobrých vektorů pro celé věty, a příliš neřeší vytváření vektorů pro samostatná slova, je dobré provádět expanzi slovníku v každém případě. S tím může pomoci například model metody *Word2Vec* (viz kapitola 3.2), která má velmi dobré slovní vektory. Při expanzi slovníku se tak metoda

snaží namapovat vektory z metody *Word2Vec* na vektory v matici $\boldsymbol{V}$, které vznikly při trénování metody.

5 Měření podobnosti textů

Cílem úlohy sémantické podobnosti textů, je vyjádřit podobnost dvou textů (jak moc jsou si podobné). Existuje několik možných způsobů, jak tuto podobnost vyjádřit. Všechna slova a texty jsou vyjádřeny vektorem v mnoharozměrném prostoru. Měří se tedy vzdálenost mezi těmito vektory. Čím vzdálenější jsou vektory, tím méně jsou si texty podobné.

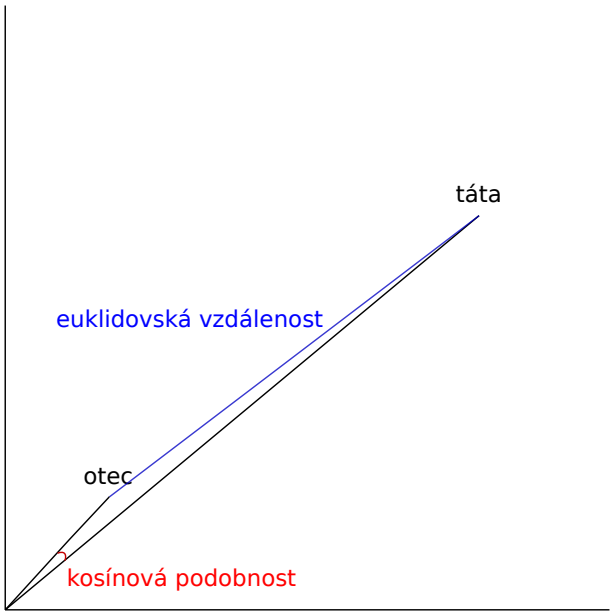
5.1 Euklidovská vzdálenost

Euklidovská vzdálenost je základní metrikou pro měření vzdáleností v mnoharozměrných prostorech. Vzdálenost dvou vektorů $\boldsymbol{x}$ a $\boldsymbol{y}$ se vypočte následovně:

$$s_{eukl}(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\| = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}, \tag{5.1}$$

kde n je dimenze vektoru.

Tato metrika se nicméně v úloze sémantické podobnosti nepoužívá, jelikož podobná slova mají stejný směr vektoru, ale v prostoru mohou být od sebe velmi vzdáleny. Příkladem mohou být dvě slova mající stejný význam jako „táta“ a „otec“, obě slova se vyskytují ve stejných kontextech, ale slovo „otec“ se vyskytuje podstatně méně, než slovo „táta“, pak budou mít tyto slova vysokou Euklidovskou vzdálenost, viz Obrázek 5.1. Z toho důvodu se používá kosínová podobnost.



Obrázek 5.1: Ilustrace rozdílu mezi euklidovskou vzdáleností a kosínovou podobností dvou vektorů ve 2D prostoru.

5.2 Kosínová podobnost

Kosínová podobnost měří podobnost dvou nenulových vektorů. Podobnost vyjadřuje kosinus úhlu mezi těmito vektory. Pokud je úhel mezi vektory 0°, pak je kosínová podobnost rovna hodnotě 1 a slova (texty) mají stejný význam. Naopak jestliže jsou na sebe vektory kolmé (90°), pak je kosínus úhlu roven 0 a mezi slovy (texty) neexistuje žádná spojitost.

Kosínová podobnost dvou vektorů $\boldsymbol{x}$ a $\boldsymbol{y}$ se spočte:

$$s_{cos}(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{\|\boldsymbol{x}\| \cdot \|\boldsymbol{y}\|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \sqrt{\sum_{i=1}^{n} y_i^2}}, \tag{5.2}$$

kde n je dimenze vektoru.

5.3 Hlavní úhly

Metoda hlavních úhlů (*Principal Angles*) je metoda pro měření podobnosti dvou textů. Potřeba jsou k tomu vektory slov. Podle (Mu aj., 2017) je nejvíce informací o slovech ve větách ukryto v malém podprostoru. Z toho vyplývá, že podobné věty, by měly mít podobné podprostory (Brychcín, 2018). Podprostor, který obsahuje nejvíc informací (variace) ve slovních vektorech, lze

získat metodou analýzy hlavních komponent (*Principal Component Analysis – PCA*, kapitola 6.1).

Pro dvě věty x a y, obsahující slova $w_1^x, ..., w_i^x$ a $w_1^y, ..., w_j^y$, se sestaví matice $\boldsymbol{X}$ a $\boldsymbol{Y}$. Matice mají rozměry $d \times k$, kde d je dimenze vektorů slov a l je délka věty. Nyní se do sloupců matice uloží vektory slov. Sloupec i matice $\boldsymbol{X}$ bude obsahovat vektor slova w_i^x. Tyto vektory lze též vážit vynásobením hodnotou λ, stejně jako v metodě lineární kombinace (kapitola 4.2). I zde lze použít *idf* vážení. Po sestavení obou matic se na ně aplikuje singulární rozklad (SVD). Nyní se vezmou matice $\boldsymbol{U}^x$ a $\boldsymbol{U}^y$ vzniklé z rozložení původních matic, a oříznou se sloupce tak, aby zůstalo pouze k prvních sloupců (hlavních komponent). Takto vzniknou redukované matice $\boldsymbol{U}_k^x$ a $\boldsymbol{U}_k^y$. Počet hlavních komponent k se může určit experimentálně, nebo ho lze určit výpočtem, kdy se v matici $\boldsymbol{S}$ sčítají vlastní čísla umístěná na hlavní diagonále a jakmile součet vlastních čísel přesáhne určitou hodnotu (např. $0,99$), tak se hodnota k nastaví na počet vlastních čísel, která přispěla do součtu.

Nyní se obě redukované matice $\boldsymbol{U}_k^{x\top}$ a $\boldsymbol{U}_k^y$ vynásobí a na vzniklou matici se opět aplikuje singulární rozklad. Nyní se určí podobnost vět:

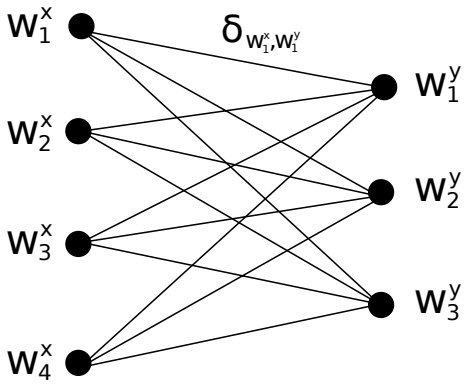
$$s_{pa}(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{r} \sigma_i^2}, \tag{5.3}$$

kde σ jsou hodnoty na hlavní diagonále v matici $\boldsymbol{S}$, která vznikla singulárním rozkladem $\boldsymbol{U}_k^{x\top}\boldsymbol{U}_k^y$.

5.4 Optimální párování

Metoda optimálního párování (*Optimal Matching*) (Glavas aj., 2018) je velmi úspěšná pro měření sémantické podobnosti textů (Brychcín, 2018). Optimální párování je metoda učení bez učitele, která páruje slova ve větách, u kterých se určuje podobnost, podle podobného významu nebo roli ve větě (podmět, přísudek, atd.).

Pro dvě věty x a y, obsahující slova $w_1^x, ..., w_i^x$ a $w_1^y, ..., w_j^y$, se provede zarovnání slov, kde jedno slovo z věty x se zarovná na jedno slovo ve větě y. Každé slovo může být nejvýše v jednom páru. Párování si lze představit jako úplný bipartitní graf M(viz Obrázek 5.2), kde vrcholy na levé straně odpovídají slovům ve větě x a vrcholy na pravé straně grafu odpovídají slovům ve větě y.



Obrázek 5.2: Úplný bipartitní graf sestavený pro metodu optimálního párování.

Každé slovo na jedné straně je spojeno hranou se všemi slovy na opačné straně grafu. Tyto hrany jsou ohodnoceny vahou $\delta_{w_k^x, w_l^y}$. Váha hrany se vypočte jako kosínová podobnost vektorů slov, mezi kterými vede hrana:

$$\delta_{w_k^x, w_l^y} = \cos(\boldsymbol{v}_{w_k^x}, \boldsymbol{v}_{w_l^y}). \tag{5.4}$$

Nyní je cílem nalézt optimální párování, s co největším součtem vah v grafu. Nejlepší párování v bipartitním grafu řeší tzv. Maďarská metoda (*Hungarian method*) (Kuhn – Yaw, 1955). Po nalezení nejlepších párů získáme graf $\hat{M}$ s hranami (w^x, w^y). Dále se již vypočte podobnost obou vět:

$$s_{om} = \sum_{(w^x, w^y) \in \hat{M}} \delta_{w^x, w^y}. \tag{5.5}$$

Podobně jako u předchozích metod, i zde je možné přidat další váhu λ, vyjadřující jak moc je slovo, z kterého vede hrana, důležité pro větu. Při přidání těchto vah je potřeba nejprve pro každou větu vypočíst její párovací skóre:

$$\delta_x = \frac{1}{\sum_{w^x \in x} \lambda_{w^x}} \sum_{(w^x, w^y) \in \hat{M}} \lambda_{w^x} \delta_{w^x, w^y}. \tag{5.6}$$

Párovací skóre pro větu y se vypočte obdobně, pouze se bude počítat s váhami λ_{w^y}. Po vypočtení skóre pro obě věty se určí podobnost těchto vět aritmetickým průměrem jejich skóre:

$$s_{om} = \frac{\delta_x + \delta_y}{2}. \tag{5.7}$$

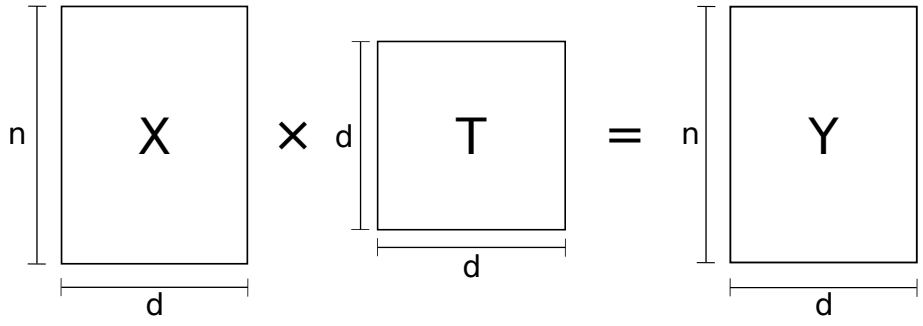
6 Transformace sémantických prostorů

Lineární transformace vektorových prostorů je způsob, jak jeden sémantický prostor transformovat do jiného sémantického prostoru (Brychcín, 2018). Lineární transformace mohou být použity pro provedení afinních transformací, jako je posunutí, otáčení, zrcadlení, zkosení, a další transformace. Toho lze využít pro získání jednotné sémantické reprezentace slov a vět napříč různými jazyky.

Předpokládejme dva sémantické prostory $\boldsymbol{X}$ a $\boldsymbol{Y}$, kde $\boldsymbol{X}$ je sémantický prostor zdrojového jazyka, obsahující vektory $x_1, ..., x_n$, který budeme chtít transformovat do cílového sémantického prostoru $\boldsymbol{Y}$, s vektory $y_1, ..., y_n$. Matice sémantických prostorů mají rozměry $n \times d$, kde n je počet vektorů a d je dimenze těchto vektorů. Cílem je najít transformační matici $\boldsymbol{T}$ s rozměry $d \times d$ takovou, že platí:

$$\boldsymbol{X} \times \boldsymbol{T} \approx \boldsymbol{Y}, \tag{6.1}$$

viz Obrázek 6.1.



Obrázek 6.1: Hledání transformační matice.

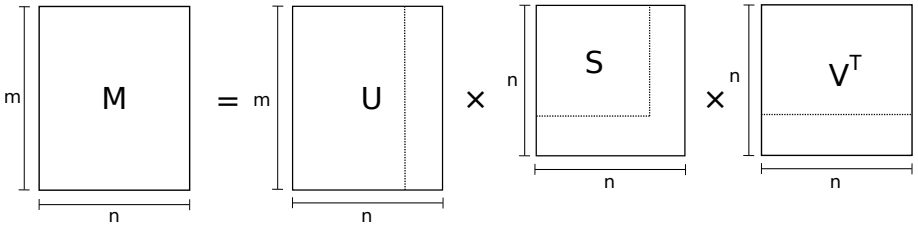
Všechny dále popsané transformace využívají do nějaké míry metodu SVD (singulární rozklad, z angl. *Singular Value Decomposition*), která je popsána v následující kapitole.

6.1 Singulární rozklad

Singulární rozklad (Wall aj., 2003) je rozklad matice na tři matice $\boldsymbol{U}, \boldsymbol{S}, \boldsymbol{V}$, viz Obrázek 6.2. Rozklad matice $\boldsymbol{M}$, o rozměrech $m \times n$, lze provést přes

singulární rozklad pokud je matice složena z reálných nebo případně komplexních čísel. Zároveň platí, že vynásobením rozložených matic vznikne zpět původní matice $\boldsymbol{M}$:

$$\boldsymbol{M} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\top} \tag{6.2}$$



Obrázek 6.2: SVD rozklad matice na U, S, V^T.

Matice $\boldsymbol{U}$ je unitární matice s rozměry $m \times n$, kde každý sloupec matice odpovídá jednomu vlastnímu vektoru matice, která vznikne vynásobením původní matice $\boldsymbol{M}$ transponovanou maticí $\boldsymbol{M}^{\top}$.

Matice $\boldsymbol{S}$ je diagonální matice $n \times n$, kde na hlavní diagonále jsou nezáporná reálná čísla a zbytek matice obsahuje samé nuly. Na diagonále jsou umístěny vlastní čísla matice $\boldsymbol{M}$ seřazené podle velikosti od nejvyšší hodnoty po nejmenší, tedy platí že:

$$\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n \geq 0. \tag{6.3}$$

Matice $\boldsymbol{V}^T$ je transponovaná unitární matice $\boldsymbol{V}$ o rozměrech $n \times n$, kde každý sloupec je vlastní vektor matice vzniklé vynásobením matic $\boldsymbol{M}^T$ a $\boldsymbol{M}$.

Singulární rozklad se používá k celé řadě úloh. Například výpočet pseudoinverzní matice, k řešení metody nejmenších čtverců, aproximace matic s nižší dimenzí, určování řádu matice, atd. Výpočet pseudoinverzní matice se používá k transformaci sémantických prostorů (viz kapitola 6.2). Pseudoinverzní matice k matici $\boldsymbol{M}$ se vypočte singulárním rozkladem této matice $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$. Nyní se spočte pseudoinverzní matice k matici $\boldsymbol{S}$, což se vypočte nahrazením hodnot na diagonále jejich převrácenými hodnotami. Pseudoinverzní matici $\boldsymbol{M}^{-1}$ tak získáme:

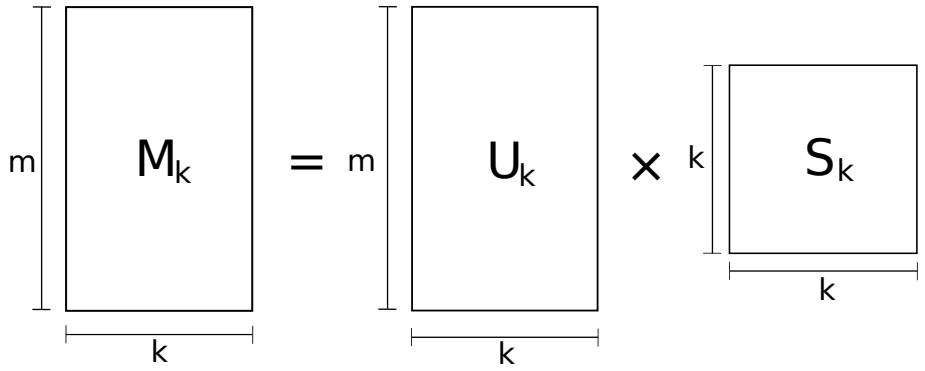
$$\boldsymbol{M}^{-1} = \boldsymbol{V}\boldsymbol{S}^{-1}\boldsymbol{U}^{\top}. \tag{6.4}$$

Dalším využitím singulárního rozkladu u sémantických modelů je redukce dimenzionality metodou PCA (analýza hlavních komponent, z angl. *Principal Component Analysis*). Matici lze redukovat počet řádků nebo sloupců

na cílový počet. Nejprve se provede rozklad matice (viz Obrázek 6.2), poté se matice redukují na požadovanou dimenzi k a oříznou se. Vzniknou tak matice $\boldsymbol{U}_k$ s rozměry $m \times k$, $\boldsymbol{S}_k$ s rozměry $k \times k$ a matice $\boldsymbol{V}_k$ s rozměry $n \times k$. Nyní pro redukci dimenzionality řádků matice se vynásobí matice $\boldsymbol{U}_k$ a $\boldsymbol{S}_k$:

$$\boldsymbol{M}_k = \boldsymbol{U}_k \times \boldsymbol{S}_k. \tag{6.5}$$

Získá se tak aproximace původní matice $\boldsymbol{M}_k$ s rozměry $m \times k$, viz Obrázek 6.3, která obsahuje hodnoty nesoucí nejvíce informací z původní matice.

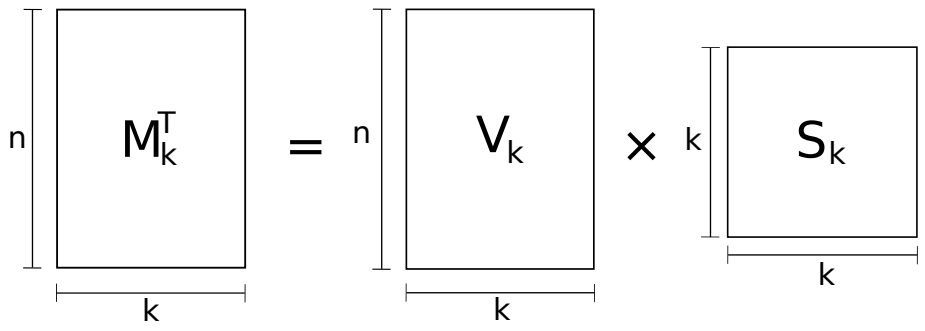


Obrázek 6.3: Redukce řádků matice na dimenzi k.

Opačným přístupem je redukce počtu řádků. Při redukci dimenzionality vektorů ve sloupcích matice se provede násobení matic $\boldsymbol{V}_k$ a $\boldsymbol{S}_k$:

$$\boldsymbol{M}_k^\top = \boldsymbol{V}_k \times \boldsymbol{S}_k. \tag{6.6}$$

Matice M_k^T se poté transponuje, aby vektory byly opět ve sloupcích matice, která bude mít rozměry $k \times n$ (viz Obrázek 6.4).



Obrázek 6.4: Redukce sloupců matice na dimenzi k.

6.2 Transformace metodou nejmenších čtverců

Transformační matice $\boldsymbol{T}$ může být získána řešením optimalizačního problému:

$$\hat{\boldsymbol{T}} = \arg\min_{\boldsymbol{T}} \sum_{i=1}^{n} \|\boldsymbol{T}\boldsymbol{x}_i - \boldsymbol{y}_i\|^2, \tag{6.7}$$

pomocí stochastického gradientního sestupu (Mikolov aj., 2013b).

V rovnici 6.7 si lze všimnout, že se minimalizuje součet rozdílů čtverců euklidovské vzdálenosti mezi vektory dvou vektorových prostorů. Řeší se tedy metoda nejmenších čtverců, která má kromě řešení gradientním sestupem i analytické řešení. Analytické řešení spočívá v získání *Moore-Penrose* pseudoinverzní matice k matici $\boldsymbol{X}$ (Brychcín, 2018). Pseudoinverzní matice může být spočtena pomocí metody SVD (viz kapitola 6.1). Transformační matici tedy získáme následovně:

$$\boldsymbol{T} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{Y}. \tag{6.8}$$

6.3 Ortogonální transformace

Transformační matice řešená metodou nejmenších čtverců má nevhodně zvolenou optimalizační funkci, neboť podobnost slov se měří kosínovou vzdáleností a nikoliv euklidovskou vzdáleností. Podle (Xing aj., 2015) by měla být transformační matice ortogonální, protože zachová velikosti úhlů mezi vektory v sémantickém prostoru, a tedy kosínová vzdálenost zůstane stejná. Matice $\boldsymbol{T}$ je ortogonální, pokud všechny řádky a sloupce matice jsou ortonormální (součin dvou libovolných řádků nebo sloupců je roven 0), pak platí: $\boldsymbol{T}\boldsymbol{T}^\top = \boldsymbol{I}$, kde $\boldsymbol{I}$ je jednotková matice.

Stejně jako transformace metodou nejmenších čtverců, i tento optimalizační problém lze vyřešit gradientním sestupem. Nicméně existuje analytické řešení s použitím singulárního rozkladu, které najde optimální ortogonální transformační matici (Brychcín, 2018).

Optimální ortogonální transformační matice se získá:

$$\boldsymbol{T} = \boldsymbol{V}\boldsymbol{U}^\top, \tag{6.9}$$

kde $\boldsymbol{V}$ a $\boldsymbol{U}^\top$ jsou matice získané singulárním rozkladem součinu matic $\boldsymbol{Y}^\top$ a $\boldsymbol{X}$:

$$\boldsymbol{Y}^\top \boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\top. \tag{6.10}$$

6.4 Kanonická korelační analýza

Kanonická korelační analýza je způsob měření lineární závislosti mezi dvěmi veličinami (vektory). Pro matice $\boldsymbol{X}$ a $\boldsymbol{Y}$ kanonická korelační analýza nalezne matice $\boldsymbol{C}^x$ a $\boldsymbol{C}^y$ takové, že mezi lineární kombinací $(\boldsymbol{C}^x\boldsymbol{X})$ a $(\boldsymbol{C}^y\boldsymbol{Y})$ je maximální korelace. Zároveň jsou takto oba prostory promítnuty do jednoho společného prostoru (Brychcín, 2018).

Postup pro získání matic $\boldsymbol{C}^x$ a $\boldsymbol{C}^y$ je následující. Nejprve se nalezne pár vektorů $(\boldsymbol{c}_1^x, \boldsymbol{c}_1^y)$, jejichž projekce do prostoru $\boldsymbol{X}$ a $\boldsymbol{Y}$ $(\boldsymbol{X}\boldsymbol{c}_1^x, \boldsymbol{Y}\boldsymbol{c}_1^y)$ má nejvyšší hodnotu Pearsonova korelačního koeficientu. Po získání těchto vektorů se hledá další nejlepší pár vektorů $(\boldsymbol{c}_2^x, \boldsymbol{c}_2^y)$, ale pouze v ortogonálním podprostoru k prvně nalezeným vektorům $(\boldsymbol{c}_1^x, \boldsymbol{c}_1^y)$. Takto se hledá d párů vektorů, kde d je dimenze vektorů v maticích $\boldsymbol{X}$ a $\boldsymbol{Y}$. Obecně platí, že k-tý pár vektorů se získá:

$$(\boldsymbol{c}_k^x, \boldsymbol{c}_k^y) = \arg\max_{\boldsymbol{c}^x, \boldsymbol{c}^y} corr(\boldsymbol{X}\boldsymbol{c}^x, \boldsymbol{Y}\boldsymbol{c}^y), \tag{6.11}$$

kde platí:

$$(\boldsymbol{X}\boldsymbol{c}^x) \cdot (\boldsymbol{X}\boldsymbol{c}_i^x) = 0 \wedge (\boldsymbol{Y}\boldsymbol{c}^y) \cdot (\boldsymbol{Y}\boldsymbol{c}_i^y) = 0, \tag{6.12}$$

pro každé $1 \leq i < k; i \in N$.

Na konci algoritmu jsou tedy dvě matice $\boldsymbol{C}^x$ a $\boldsymbol{C}^y$ s rozměry $d \times d$. Pro získání transformační matice se vynásobí matice $\boldsymbol{C}^x$ s inverzní maticí k matici $\boldsymbol{C}^y$:

$$\boldsymbol{T} = \boldsymbol{C}^x(\boldsymbol{C}^y)^{-1}. \tag{6.13}$$

Kromě tohoto algoritmu existuje též analytické řešení za použití singulárního rozkladu. Nejprve se vytvoří matice $\boldsymbol{O}$, která vznikne vynásobením transponované matice $\boldsymbol{U}_X$, z rozkladu matice $\boldsymbol{X}$, s maticí $\boldsymbol{U}_Y$, z rozkladu matice $\boldsymbol{Y}$:

$$\boldsymbol{O} = \boldsymbol{U}_X^\top \boldsymbol{U}_Y. \tag{6.14}$$

Poté se vytvoří matice $\boldsymbol{A}$ a $\boldsymbol{B}$:

$$\boldsymbol{A} = \boldsymbol{V}_X \boldsymbol{S}_X^{-1} \boldsymbol{U}_O, \tag{6.15}$$

$$\boldsymbol{B} = \boldsymbol{V}_Y \boldsymbol{S}_Y^{-1} \boldsymbol{V}_O, \tag{6.16}$$

kde matice $\boldsymbol{U}_O$ a $\boldsymbol{V}_O$ vznikly singulárním rozkladem matice $\boldsymbol{O}$. Nyní se již vytvoří transformační matice $\boldsymbol{T}$:

$$\boldsymbol{T} = \boldsymbol{A}\boldsymbol{B}^{-1}. \tag{6.17}$$

7 Navržené metody pro sémantickou podobnost textů

Hlavním cílem práce je vyzkoušení několika způsobů, jak jeden sémantický prostor promítnout do jiného. K tomu slouží transformace popsané v kapitole 6, které umí transformovat vektory z jednoho sémantického prostoru do druhého. V dosud publikovaných článcích (Brychcín, 2018; Xing aj., 2015; Artetxe aj., 2016) byly transformace trénovány na slovních vektorech, které pak byly transformovány do sémantického prostoru jiného jazyka. Tento způsob je v této kapitole též popsán. Kromě tohoto přístupu jsou v této práci popsány nové, dosud nepublikované metody, jejichž myšlenka je transformovat nikoliv vektory slov, ale rovnou vektory celých vět.

Celkem jsou tak v této kapitole popsány tři metody, jak transformovat sémantické prostory.

7.1 Transformace na slovech

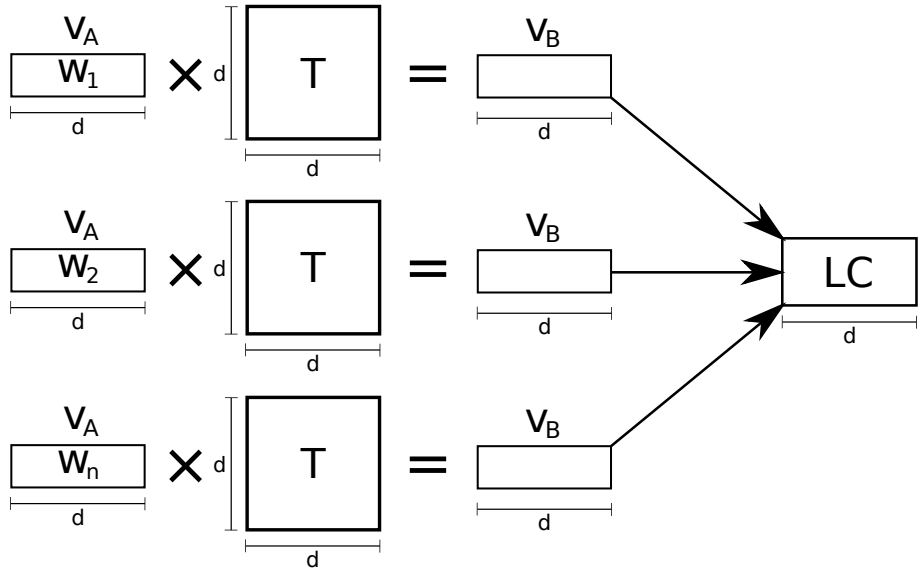
Pro transformaci slovních vektorů do sémantického prostoru cílového jazyka, je potřeba mít zarovnaná slova ze zdrojového jazyka na odpovídající slova v cílovém jazyce. K tomu bylo použity soubory s 50 000 slovy přeloženými do testovaných jazyků. Slova ze souboru v původním jazyce jsou značena $x_1, ..., x_{50000}$ a slova přeložená do cílového jazyka jsou značena $y_1, ..., y_{50000}$.

Pro určení transformační matice $\boldsymbol{T}$, se sestaví dvě matice $\boldsymbol{X}$ a $\boldsymbol{Y}$. Matice $\boldsymbol{X}$ bude obsahovat vektory ze sémantického prostoru zdrojového jazyka A a matice $\boldsymbol{Y}$ vektory sémantického prostoru cílového jazyka B. Nyní se prochází soubor s přeloženými slovy. Pokud sémantický prostor A obsahuje vektor slova x_i a zároveň sémantický prostor B obsahuje vektor pro slovo y_i, tak se vektor slova x_i vloží do matice $\boldsymbol{X}$ a vektor slova y_i se vloží do matice $\boldsymbol{Y}$.

Takto vzniknou dvě matice, které mají zarovnané vektory slov se stejným významem na stejný řádek matice. Nyní se hledá transformační matice jednou z metod, které byly popsány v kapitole 6. Pro promítnutí vektoru $\boldsymbol{v}^A$ ze zdrojového jazyka na vektor do cílového jazyka $\boldsymbol{v}^B$ se vektor $\boldsymbol{v}^A$ vynásobí transformační maticí $\boldsymbol{T}$:

$$\boldsymbol{v}^A \boldsymbol{T} = \boldsymbol{v}^B. \tag{7.1}$$

Sestavení vektoru pro větu se poté udělá váženou lineární kombinací transformovaných vektorů. Výsledný vektor lze poté porovnat s vektorem pro větu z cílového jazyka. Schéma transformace a následného porovnání vět je zobrazeno na obrázku 7.1.



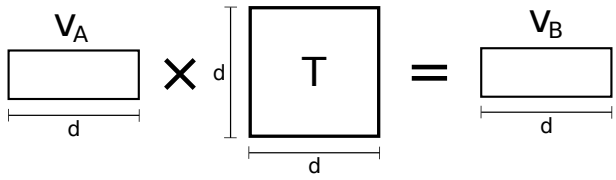
Obrázek 7.1: Transformace slov ze zdrojového do cílového jazyka a jejich lineární kombinace pro porovnání vět mezi těmito jazyky.

7.2 Transformace na větách

Stejně jako funguje transformace na jednotlivých slovech, lze předpokládat, že stejně by mohla fungovat transformace na vektorech celých vět. Vektory pro celé věty jsou získány metodami *Paragraph2Vec*, *Skip-thoughts* nebo lineární kombinací slovních vektorů (viz kapitoly 4.3, 4.4 a 4.2). K trénování transformační matice na větách je potřeba paralelní korpus, například takový, který se používá pro strojový překlad. Tento korpus by měl obsahovat větu ve zdrojovém jazyce a její ekvivalent v cílovém jazyce. Takto by bylo vhodné mít několik tisíc párů vět, aby se transformační matice dobře natrénovala.

Nyní se pro každou větu z korpusu vytvoří vektor, který je trénován na stejném jazyce jako daná věta. Tyto vektory vytvoří matice $\boldsymbol{X}$ a $\boldsymbol{Y}$, kde každý řádek je vektor jedné věty. Nyní se stejně jako pro transformace slov hledá transformační matice $\boldsymbol{T}$.

Po vypočtení transformační matice lze transformovat vektory vět, vytvořené modelem pro zdrojový jazyk, do sémantického prostoru cílového jazyka (rovnice 7.1), viz Obrázek 7.2.



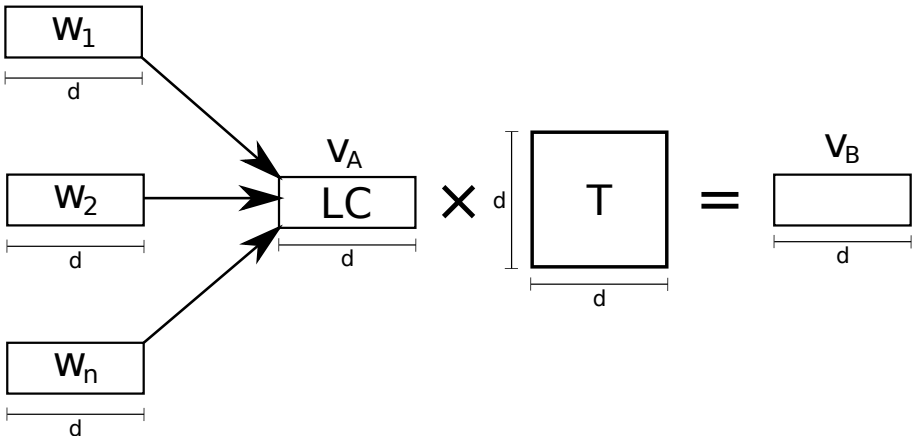
Obrázek 7.2: Transformace vektoru ze zdrojového do cílového jazyka.

V případě transformace vět, jejichž vektory jsou získány lineární kombinací slovních vektorů, se prohodí postup transformací na slovech. Nejprve se tedy ze slovních vektorů lineární kombinací sestaví vektory vět, které jsou pak transformovány do cílového prostoru.

Pro trénování transformační matice $\boldsymbol{T}$ se i zde vytvoří matice $\boldsymbol{X}$ pro vektory ze zdrojového jazyka a matice $\boldsymbol{Y}$ pro vektory z cílového jazyka. Nyní se prochází paralelní korpus a pro každou větu s obsahující slova $w_1, ..., w_n$ se spočte její vektor pomocí vážené lineární kombinace (rovnice 4.2) slovních vektrů získaných z jednoho ze sémantických modelů pro reprezentaci slov (kapitola 3).

Tyto vektory pro věty jsou uloženy do matic $\boldsymbol{X}$ a $\boldsymbol{Y}$, tak aby byly vektory odpovídajících vět zarovnány na stejném řádku. Dále, když už je známá matice $\boldsymbol{X}$ i $\boldsymbol{Y}$, lze spočíst transformační matici.

Po spočtení transformační matice, lze převést vektor pro větu ze zdrojového jazyka do cílového. Nejprve je ale potřeba spočíst vektor pro tuto větu. K tomu se použijí slovní vektory zdrojového jazyka a jejich lineární kombinace, viz rovnice 4.2. Po spočtení vektoru pro tuto větu ji lze vynásobit transformační maticí, tím se získá projekce vektoru do sémantického prostoru cílového jazyka. Princip tohoto postupu lze vidět na obrázku 7.3.



Obrázek 7.3: Transformace lineární kombinace slov ze zdrojového do cílového jazyka.

Celkem jsou v této kapitole popsány tři způsoby transformace na větách:

- Transformace lineární kombinace slovních vektorů.
- Transformace vět z *Paragraph2Vec* modelu.
- Transformace vět ze *Skip-thoughts* modelu.

Transformace na větách je nový funkční způsob, jak porovnávat význam vět mezi jazyky. Jediná nevýhoda tohoto přístupu je, že potřebuje trénovací data podobná těm testovacím, nebo velký paralelní korpus, který bude obsahovat různorodé věty.

7.3 Transformace Paragraph2Vec modelu

Dalším novým způsobem transformace vektorů mezi jazyky je transformace na slovech, které jsou uloženy v *Paragraph2Vec* modelu[1]. Předpoklad je, že pokud funguje transformace na slovech a jejich následná lineární kombinace, pak by mohla fungovat transformace na slovech a sestavení vektoru pro větu v *Paragraph2Vec* modelu.

Pro uskutečnění této transformace je potřeba rozložit natrénované modely ze zdrojového a cílového jazyka, aby bylo možné natrénovat na těchto modelech transformační matici, která vektory zdrojového jazyka transformuje do cílového.

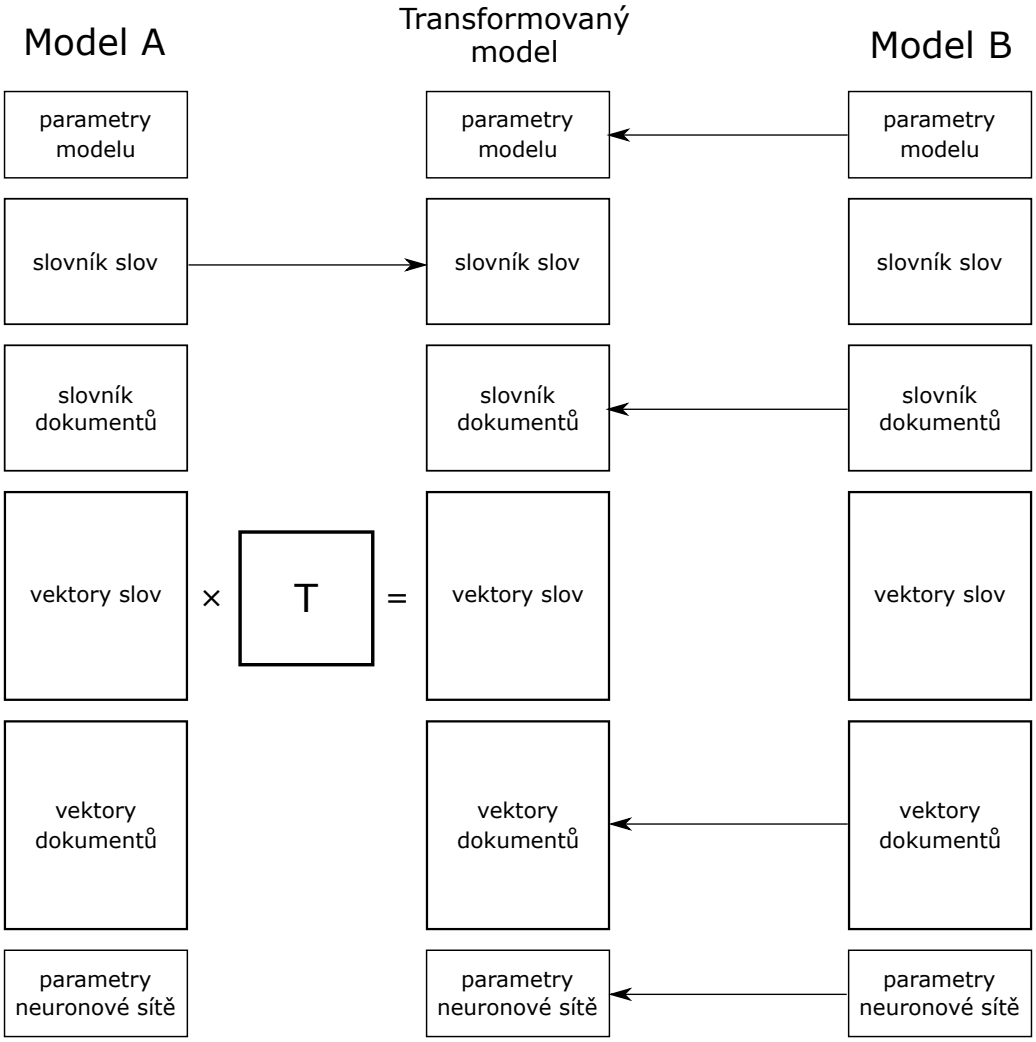
Nejprve se natrénují monolinguální modely pro testované jazyky. Každý model obsahuje následující části: parametry modelu, slovník slov, která se

[1]Implementace dostupná zde: github.com/hiyijian/doc2vec.

použila pro trénování modelu, slovník dokumentů, vektory slov, vektory dokumentů, parametry neuronové sítě. Pro natrénování transformační matice se použijí vektory slov z modelů pro zdrojový a cílový jazyk. Tyto vektory se zarovnají do matic $\boldsymbol{X}$ a $\boldsymbol{Y}$, podle souboru se zarovnanými slovy, stejně jako u transformace na slovech (kapitola 7.1).

Po natrénování transformační matice $\boldsymbol{T}$ se začne skládat model pro zdrojový jazyk, který bude generovat vektory pro věty, jenž budou porovnatelné s vektory získané z modelu cílového jazyka. Z modelu cílového jazyka se použijí následující části: parametry modelu, slovník dokumentů, vektory dokumentů, a parametry neuronové sítě. Z modelu zdrojového jazyka se použije slovník slov a vektory těchto slov, které se transformují do sémantického prostoru cílového jazyka. Každý vektor pro slovo se vynásobí transformační maticí, takto vznikne vektor v sémantickém prostoru cílového jazyka, který se uloží do transformovaného modelu. Složení modelu je zobrazeno na obrázku 7.4. Transformovaný model se pak použije pro vygenerování vektorů pro věty ze zdrojového jazyka, tyto věty jsou pak porovnatelné s vektory z *Paragraph2Vec* modelu cílového jazyka.

Tato metoda nedosahuje tak dobrých výsledků, jak bylo očekáváno, ale pořád dokáže do jisté míry porovnávat věty mezi jazyky.



Obrázek 7.4: Transformace *Paragraph2Vec* modelu do cílového jazyka.

8 Data a evaluace

Metody pro sémantickou reprezentaci textů potřebují k trénování velké množství dat – korpus. Tento korpus je potřeba nejprve připravit do zpracovatelné podoby. Po natrénování modelů je potřeba tyto modely porovnat s testovacími datasety, k číselnému určení úspěšnosti modelů se pak používají korelace. V této kapitole jsou tedy popsány trénovací data, jejich předzpracování a následná evaluace.

8.1 Trénovací data

Pro trénování metod byly použity články z Wikipedie. Ta dělá každý měsíc sjednocení všech článků, které se zde vyskytují, do jednoho souboru v XML formátu, který je pak možné stáhnout z webové stránky[1].

Vzhledem k jazyku testovacích dat se v této práci použily Wikipedie v následujících jazycích: angličtina, španělština, italština, arabština, turečtina a chorvatština. Pro všechny tyto jazyky se použil dump Wikipedie z 1. července 2018. Velikost těchto korpusů, včetně počtu článků, slov a počtu unikátních slov jsou uvedeny v tabulce 8.1. Unikátní slova jsou ta slova, pro které se určoval sémantický vektor, zároveň se brala v úvahu pouze slova, která se v celém korpusu vyskytla minimálně pětkrát.

Jazyk Wikipedie	Velikost (v MB)	Počet článků	Počet slov	Počet unikátních slov
angličtina	12 514	5 801 010	1 987 001 150	4 297 657
španělština	3 207	1 392 788	506 715 486	1 565 579
arabština	1 039	587 184	-	545 051
turečtina	423	314 950	50 179 080	395 934
italština	2 664	1 518 136	405 434 441	994 979
chorvatština	280	190 432	40 821 479	318 759

Tabulka 8.1: Statistiky o velikosti Wikipedie v různých jazycích, které byly použity jako trénovací korpus pro sémantické modely.

Jak lze z tabulky 8.1 vidět, některé korpusy jsou opravdu velké, ale jiné jsou příliš malé na to, aby se metody dobře natrénovaly, hlavně pro chorvatštinu. Proto se pro trénování metod pro chorvatštinu použil, kromě Wikipedie, i korpus složený z textů na webových stránkách[2] (Šnajder aj., 2013).

[1] Wikipedie ke stažení: dumps.wikimedia.org

[2] Korpus dostupný zde takelab.fer.hr

Takto složený korpus je již mnohem větší (7,1 GB) a obsahuje 1 226 940 839 slov. Což už je dostatečné množství pro natrénování metod.

8.2 Předzpracování dat

Tyto stažené korpusy bylo nyní potřeba předzpracovat, což znamená převést je do podoby potřebné pro trénování, kde jeden řádek v souboru bude jeden dokument a slova budou očištěna od přebytečných značek, které Wikipedie používá pro formátování textu.

Pro předzpracování XML souborů, obsahující články Wikipedie se použila již hotová implementace *Wikipedia Extractor*[3], která z XML souboru odstraní všechny přebytečné značky a zanechá pouze oddělené články, označené jejich id hodnotou, a čistý text.

Dále se provedla tokenizaci textu, při které se odstranily všechna interpunkční znaménka, závorky, pomlčky, a další znaky, které se používají pro strukturování a lepší čitelnost textu. U sémantických modelů by tyto znaky zbytečně vytvářely další slova ve slovníku.

Po tokenizaci se též odstranila všechna stop slova pro daný jazyk. Stop slova jsou ta slova, které se vyskytují v textu velmi často a nenesou žádnou důležitou informaci, jako například zájmena, předložky, spojky, částice a citoslovce. Kromě toho jsou stop slovem i pomocná slovesa, v angličtině je to například pomocné sloveso „*be*“.

Po předzpracování textu se tak získá pro každý jazyk jeden soubor, kde každý řádek je jeden dokument, obsahující slova oddělená mezerou. Velikost těchto souborů je díky předzpracování menší, hlavně se zmenší velikost slovníku, což zlepší reprezentaci slov. Změny mezi původním a předzpracovaným korpusem jsou uvedeny v tabulce 8.2.

[3] `http://medialab.di.unipi.it/wiki/Wikipedia_Extractor`

Jazyk Wikipedie	Velikost (v MB)	Počet slov	Počet unikátních slov
angličtina	8 498	1 107 555 852	2 234 517
španělština	2 177	259 171 133	529 918
arabština	938	-	542 719
turečtina	402	44 089 494	364 610
italština	1 957	229 550 041	921 012
chorvatština	5 876	712 635 346	1 114 897

Tabulka 8.2: Statistiky o velikosti korpusu po předzpracování textu a odstranění stop slov.

8.3 Testovací data

K otestování modelů a evaluaci řešení sémantické podobnosti textů se použily datasety z konferencí *SemEval*. Pro monolinguální testování podobnosti vět byly použity datasety ze *SemEval-2015*[4], *SemEval-2016*[5] a *SemEval-2017*[6]. Pro vícejazyčné testování podobnosti vět pak byly použity datasety ze *SemEval-2017* a *GoranGlavas* dataset.

Všechny datasety obsahují páry vět a číselné hodnocení jejich podobnosti na stupnici $\langle 0; 5 \rangle$. Čím větší číslo, tím víc jsou si věty významově podobné, hodnota 0 znamená, že mezi větami není žádná spojitost. Toto číselné hodnocení určili lidé, při evaluaci se tak měří závislost mezi hodnocením udělené lidmi a automatickým hodnocením, které určil systém.

Testovací data ze *SemEval-2015* obsahují monolinguální datasety pro anglický a španělský jazyk. Přehled datasetů ze *SemEval-2015* je uveden v tabulce 8.3.

[4] `http://alt.qcri.org/semeval2015/task2/`
[5] `http://alt.qcri.org/semeval2016/task1/`
[6] `http://alt.qcri.org/semeval2017/task1/`

Název datasetu	Jazyk datasetu	Počet párů vět
answers-forums	en-en	375
answers-students	en-en	750
belief	en-en	375
headlines	en-en	750
images	en-en	750
newswire	es-es	500
wikipedia	es-es	251

Tabulka 8.3: Přehled datasetů z konference *SemEval-2015*.

Z konference *SemEval-2016* byly použity monolinguální datasety pro anglický jazyk. Tyto datasety jsou uvedeny v tabulce 8.4.

Název datasetu	Jazyk datasetu	Počet párů vět
answer-answer	en-en	254
headlines	en-en	249
plagiarism	en-en	230
postediting	en-en	244
question-question	en-en	209

Tabulka 8.4: Přehled datasetů z konference *SemEval-2016*.

Ze *SemEval-2017* byly použity všechny datasety, jednojazyčné i vícejazyčné. Datasety obsahují anglický, španělský, arabský, a turecký jazyk. Celkem je 7 datasetů, které jsou uvedeny v tabulce 8.5.

Název datasetu	Jazyk datasetu	Počet párů vět
track1	ar-ar	250
track2	ar-en	250
track3	es-es	250
track4a	es-en	250
track4b	es-en	250
track5	en-en	250
track6	tr-en	500

Tabulka 8.5: Přehled datasetů z konference *SemEval-2017*.

Posledním datasetem použitým pro evaluaci vícejazyčné sémantické podobnosti textů byl dataset *GoranGlavas* (Glavas aj., 2018). Tento dataset obsahuje datasety z konference *SemEval-2012*, ale jedna věta, z každého páru je přeložená do španělského, italského, a chorvatského jazyka. Při evaluaci datasetu se tak porovnávají přeložené věty spolu s anglickými větami.

Název datasetu	**Jazyk datasetu**	**Počet párů vět**
MSRvid	en-en/es/it/hr	750
multisource-16	en-en/es/it/hr	294
News16	en-en/es/it/hr	301
OnWN	en-en/es/it/hr	750

Tabulka 8.6: Přehled *GoranGlavas* datasetu.

8.4 Evaluace

Naměřená data je potřeba nějakým způsobem evaluovat. Vyhodnocení probíhá změřením korelace mezi dvěma veličinami reálných čísel (hodnoty z datasetu a automaticky určené podobnosti). Korelace hodnotí, jak moc jsou dvě veličiny na sobě závislé. Míru korelace vyjadřuje korelační koeficient v rozmezí hodnot $\langle -1; 1 \rangle$. Čím blíže je korelační koeficient k hodnotě -1 nebo $+1$, tím víc jsou veličiny na sobě závislé a v tomto případě i podobné (lepší). V hodnotě $+1$ jsou veličiny na sobě maximálně závislé a tedy identické. V hodnotě -1 je mezi veličinami nepřímá závislost (antikorelace), pokud se například hodnoty jedné veličiny zvětší, pak se v té druhé zmenší. Pro hodnotu korelačního koeficientu 0 není nalezena závislost mezi veličinami, předpokládá se tedy, že hodnoty veličin nemají mezi sebou žádnou spojitost, tzn. veličiny nejsou podobné (špatné).

8.4.1 Pearsonova korelace

Pearsonova korelace (Chen – Popovich, 2012) se používá pro měření lineární závislosti mezi dvěma veličinami X, Y obsahujících n prvků $\{x_1,...,x_n\}$ a $\{y_1,...,y_n\}$. Výsledek korelace se značí r a je dán vzorcem:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}, \tag{8.1}$$

kde $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ a $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$.

Veličiny jsou maximálně lineárně závislé, pokud pro všechna x_i, y_i platí:

$$y_i = k \ \cdot x_i, \tag{8.2}$$

kde k je libovolné reálné číslo.

8.4.2 Spearmanova korelace

Spearmanova korelace je neparametrický test pořadové korelace, který se používá k měření asociace mezi dvěma veličinami X, Y (Chen – Popovich, 2012). Jednotlivé prvky x_i a y_i obou veličin se uspořádají vzestupně podle hodnot. Následně se přiřadí pořadová čísla p_i a q_i. Hodnota korelace je pak dána vzorcem:

$$\rho = 1 - \frac{6 \sum_i (p_i - q_i)^2}{n(n^2 - 1)} \tag{8.3}$$

kde n je počet hodnot ve veličinách.

Spearmanovu korelaci lze též spočítat jako Pearsonovu korelaci pořadových čísel p_i a q_i.

9 Experimenty

V této kapitole jsou uvedeny výsledky všech metod uvedených v předchozích kapitolách. Tyto metody byly otestovány na datasetech popsaných v kapitole 8.3. Výsledky jsou rozděleny do několika podkapitol, jelikož experimentů bylo provedeno velké množství. Nejdříve jsou popsané implementace, které byly použity pro natrénování sémantických modelů a nastavení jejich parametrů. Následují výsledky na monolinguálních datasetech a poté následují výsledky vícejazyčné, rozdělené podle způsobu trénování transformační matice. Na závěr této kapitoly je pak celkové porovnání metod, a shrnutí výsledků z experimentů.

V této kapitole jsou též používány následující zkratky: **PC** – Pearsonova korelace, **SC** – Spearmanova korelace, **LST** – transformace metodou nejmenších čtverců, **CCA** – kanonická korelační analýza a **ORT** – ortogonální transformace.

9.1 Parametry metod

V rámci práce bylo potřeba natrénovat několik sémantických modelů, k tomu byly použity již hotové implementace.

Pro metodu *GloVe* byla použita implementace ze *Stanford University*[1] (Pennington aj., 2014). Tato implementace je napsána v jazyce ANSI C. Ke stažení je zde i skript pro spuštění implementace a trénování vektorů.

Word2Vec modely jsou trénovány knihovnou *Gensim*[2] (Řehůřek – Sojka, 2010), což je knihovna pro Python, která obsahuje implementace několika metod pro sémantickou reprezentaci textu.

Pro metodu *Skip-thoughts* byla použita původní implementace[3] (Kiros aj., 2015), ta je napsána v jazyce *Python* s využitím knihoven *Theano* a *Gensim*.

Jak již bylo zmíněno v kapitole 7.3 pro trénování *Paragraph2Vec* modelů byla použita implementace[4], která je napsána v jazyce C++. Tato metoda má lépe uložený výsledný model a mnohem snadněji se s ním dělají další operace než u *Gensim* implementace. Také je model podstatně menší, jelikož *Gensim* si do modelu ukládá celé dokumenty, na kterých byl trénován,

[1] nlp.stan-ford.edu/projects/glove/
[2] radimrehurek.com/gensim/models/word2vec.html
[3] github.com/ryankiros/skip-thoughts
[4] github.com/hiyijian/doc2vec

zatímco tato implementace používá hashovací funkci pro uložení dokumentů.

Všechny výše uvedené modely byly trénovány s následujícími parametry. Dimenze výstupních vektorů byla nastavena na velikost 300. Okénko *window*, které slouží jako kontext, bylo nastaveno na 5. *Word2Vec* byl trénován s architekturou CBOW, zatímco implementace *FastText*[5] (Bojanowski aj., 2017; Joulin aj., 2016) využívala k trénování architekturu *Skip-gram*. U *FastText* implementace se ještě nastavuje velikost *bucket* (počet uložených ngramů), ta byla nastavena na hodnotu 500 000. Zbytek parametrů zůstal nastavený na původních hodnotách.

V rámci experimentů byly všechny vektory centrovány a normalizovány pro lepší numerickou stabilitu transformací (Brychcín aj., 2018). Pro získání transformační matice bylo použito 50 000 párů slov.

9.2 Monolinguální výsledky

Natrénované modely byly nejprve otestovány na monolinguálních datasetech, aby se ověřila jejich funkčnost. Monolinguálně byla testována angličtina na datasetech *SemEval-2015* (Tabulka 9.1), *SemEval-2016* (Tabulka 9.2), *GoranGlavas* (Tabulka 9.3) a *SemEval-2017* (Tabulka 9.5). Dalším jazykem testovaným monolinguálně byla španělština, která byla testována na datasetu *SemEval-2015* (Tabulka 9.4) a *SemEval-2017* (Tabulka 9.5). Dataset *SemEval-2017*, též obsahoval monolinguální dataset pro arabštinu, která na tomto datasetu byla též testována.

Vektory z metod pro sémantickou reprezentaci slov byly sečteny váženou lineární kombinací. Před tímto sečtením se všechny vektory normalizovaly na jednotkové vektory a celý sémantický prostor se vycentroval.

Monolinguálně měly všechny metody velmi dobré výsledky, trochu horší byla metoda *Skip-thoughts*, která potřebuje velmi mnoho času pro natrénování dobrých výsledků. Zde byla tato metoda trénována jeden týden na všech jazycích. Pokud by trénování probíhalo déle, pravděpodobně by dosáhla lepších výsledků. Naopak nejlepší výsledky měli metody *Word2Vec* a *FastText*, ale rozdíl mezi těmito metodami a zbývajícími je velmi malý.

[5] github.com/face-bookresearch/fastText

Model	answer-forums	answer-students	belief	headlines	images	průměr
GloVe	0,609	0,570	0,549	0,632	0,752	0,633
Word2Vec	0,596	0,595	**0,638**	0,626	**0,777**	**0,654**
FastText	**0,622**	0,519	0,461	**0,652**	0,763	0,619
Paragraph2Vec	0,549	**0,604**	0,583	0,522	0,663	0,589
Skip-thoughts	0,272	0,445	0,491	0,454	0,587	0,467

Tabulka 9.1: Výsledky Pearsonovy korelace na anglických datasetech z konference *SemEval-2015*.

Model	answer-answer	headlines	plagiarism	postediting	question-question	průměr
GloVe	0,269	0,598	0,688	0,650	0,612	0,558
Word2Vec	0,308	**0,607**	0,699	**0,725**	0,593	**0,583**
FastText	0,154	0,593	**0,719**	0,624	**0,659**	0,541
Paragraph2Vec	0,436	0,487	0,658	0,700	0,484	0,552
Skip-thoughts	**0,464**	0,444	0,650	0,719	0,237	0,508

Tabulka 9.2: Výsledky Pearsonovy korelace na anglických datasetech z konference *SemEval-2016*.

Model	MSRvid	multisource	News	onWN	průměr
GloVe	0,779	0,690	0,730	0,574	0,686
Word2Vec	**0,796**	**0,709**	**0,791**	0,603	**0,714**
FastText	0,788	0,647	0,724	**0,608**	0,695
Paragraph2Vec	0,376	0,559	0,786	0,495	0,503
Skip-thoughts	0,438	0,538	0,622	0,553	0,520

Tabulka 9.3: Výsledky Pearsonovy korelace na anglických datasetech *GoranGlavas*.

	newswire		wikipedia	
Model	**PC**	**SC**	**PC**	**SC**
GloVe	0,448	0,417	0,560	0,465
Word2Vec	**0,465**	0,419	0,494	0,416
FastText	0,462	**0,474**	**0,611**	**0,513**
Paragraph2Vec	0,417	0,400	0,418	0,390
Skip-thoughts	0,243	0,265	0,402	0,413

Tabulka 9.4: Výsledky Pearsonovy a Spearmanovy korelace na španělských datasetech z konference *SemEval-2015*.

	ar-ar		es-es		en-en		průměr	
Model	**PC**	**SC**	**PC**	**SC**	**PC**	**SC**	**PC**	**SC**
GloVe	0,552	0,571	**0,717**	**0,723**	**0,723**	0,743	0,664	0,679
Word2Vec	0,596	0,598	0,707	0,719	0,722	**0,744**	**0,675**	0,687
FastText	**0,638**	**0,637**	0,686	0,707	0,698	0,727	0,674	**0,690**
Paragraph2Vec	0,574	0,548	0,714	0,703	0,622	0,659	0,637	0,637
SkipThoughts	0,414	0,416	0,617	0,651	0,585	0,592	0,539	0,553

Tabulka 9.5: Výsledky Pearsonovy a Spearmanovy korelace na monolingu-álních datasetech z konference *SemEval-2017*.

9.3 Výsledky transformací na slovech

Po natrénování modelů se testovaly transformace na slovech. Tento způsob transformací byl již několikrát testován a tak jsou dostupné výsledky pro tento způsob. Vícejazyčné transformace jsou testovány na datasetech *SemEval-2017* (Tabulka 9.6) a *GoranGlavas* (Tabulka 9.7).

Uvedené výsledky jsou naměřené na vektorech, které vznikly váženou lineární kombinací slovních vektorů.

V tabulce 9.6 jsou uvedeny výsledky transformací z arabského, španělského a tureckého jazyka do anglického jazyka. Nejlepších výsledků zde dosáhla metoda *Word2Vec*, a to na všech transformacích. Stejně jako v (Brychcín, 2018), i zde měla nejlepší výsledky ortogonální transformace.

Model	**Trans.**	**ar-en**	**es-en**	**tr-en**	**průměr**
GloVe	LST	0,091	0,254	0,121	0,155
	CCA	0,123	0,277	0,137	0,179
	ORT	0,180	**0,341**	0,109	0,268
Word2Vec	LST	0,341	0,267	0,154	0,254
	CCA	0,348	0,307	0,176	0,277
	ORT	**0,354**	0,335	**0,191**	**0,293**
FastText	LST	0,157	0,100	0,064	0,107
	CCA	0,150	0,181	0,027	0,119
	ORT	0,125	0,203	0,050	0,126

Tabulka 9.6: Výsledky transformací na slovech na vícejazyčných datasetech z konference *SemEval-2017*. Uvedené hodnoty jsou Pearsonovy korelace.

V tabulce 9.7 jsou uvedeny výsledky transformací ze španělštiny, italštiny a chorvatštiny do angličtiny. Stejně jako na dataset *SemEval-2017*, i zde dosáhla nejlepších výsledků metoda *Word2Vec*. Z transformací pak dosáhly nejlepších výsledků ortogonální transformace a kanonická korelační analýza. Mezi těmito dvěma transformacemi byl pouze malý rozdíl ve výsledcích.

		es-en				it-en				hr-en				průměr
Model	**Trans.**	MSRvid	multi	News	onWN	MSRvid	multi	News	onWN	MSRvid	multi	News	onWN	
GloVe	LST	0,525	0,565	0,533	0,316	0,506	0,564	0,541	0,269	0,431	0,510	0,445	0,220	0,452
	CCA	0,523	0,575	0,532	0,316	0,518	0,578	0,535	0,275	0,412	0,516	0,410	0,225	0,451
	ORT	0,569	**0,604**	0,584	0,346	0,572	0,617	0,625	0,324	0,481	0,559	0,517	0,272	0,506
Word2Vec	LST	0,581	0,596	0,637	0,351	0,634	0,587	0,659	0,395	0,573	0,542	0,629	0,373	0,546
	CCA	0,587	0,601	0,652	0,365	0,654	**0,620**	**0,698**	0,424	0,572	0,583	**0,662**	**0,392**	**0,568**
	ORT	0,588	0,593	**0,655**	0,361	0,660	0,611	0,670	0,402	**0,595**	**0,616**	0,661	0,381	0,566
FastText	LST	0,636	0,438	0,478	0,364	0,614	0,429	0,498	0,339	0,536	0,339	0,461	0,274	0,451
	CCA	**0,650**	0,487	0,541	0,426	0,676	0,479	0,613	0,438	0,567	0,396	0,528	0,322	0,510
	ORT	**0,650**	0,512	0,550	**0,440**	**0,679**	0,477	0,634	**0,465**	0,577	0,378	0,533	0,336	0,519

Tabulka 9.7: Výsledky transformací na slovech na vícejazyčném datasetu *GoranGlavas*. Uvedené hodnoty jsou Pearsonovy korelace.

9.4 Výsledky transformací na větách

Pro transformaci na větách bylo nejprve potřeba získat paralelní korpus pro testované jazyky. Nejprve se jako paralelní korpus pro trénování transformační matice použil korpus z Wikipedie[6]. Ten obsahuje vždy první větu ze článků, který se nachází ve zdrojovém i cílovém jazyce. Tyto počáteční věty definují, o čem daný článek je, a předpokládá se, že obě věty si budou v obou jazycích významově podobné.

Paralelní korpus z Wikipedie byl zvolen z toho důvodu, že obsahuje věty, které budou sémantické modely znát a vytvoří pro ně dobré vektory. Nicméně po otestování na datasetech (kapitola 8.3) dosáhla metoda poměrně špatných výsledků. Pro ověření, jestli je možné provádět transformaci na větách, se přeložil jeden z datasetů pro anglický a španělský jazyk, aby každá věta měla svůj ekvivalent v druhém jazyce. Na těchto větách se pak opět trénovala transformační matice, ta byla tedy trénována na testovacích datech. Po natrénování se znovu otestovala metoda na datasetu. Nyní dosáhla velmi dobrých výsledků, které posloužily jako teoretický maximální limit, kterého lze s danými transformacemi na datasetu dosáhnout. Po prozkoumání vět z datasetu, které se syntakticky velmi lišily od vět, které je možné nalézt na Wikipedii, byl učiněn závěr, že důvod špatných výsledků metody jsou špatné trénovací věty pro transformační matici.

Vytvořil se tedy nový paralelní korpus, který bude lépe reprezentovat testovací data. Nejprve se seskupily všechny anglické věty z datasetů z konference *SemEval-2015*[7] a *SemEval-2016*[8]. Tyto věty se pak pomocí Google překladače přeložily do dalších jazyků, které se testovaly na vícejazyčnou podobnost textů. Získalo se tak 8 373 párů vět, na kterých bylo možné trénovat

[6] `http://opus.nlpl.eu/Wikipedia-v1.0.php`

[7] `http://alt.qcri.org/semeval2015/task2/`

[8] `http://alt.qcri.org/semeval2016/task1/`

transformační matici.

Po vytvoření tohoto paralelního korpusu a natrénování transformační matice na těchto datech dosáhla již metoda dobrých výsledků. Dobré výsledky má metoda proto, že tento korpus je syntaxí vět velmi podobný datasetu, na kterých a metody pro vícejazyčnou sémantickou podobnost textů testovaly, ačkoliv tyto věty nejsou stejné.

Metody *Doc2Vec* a *Skip-thoughts* dosahovaly podstatně horších výsledků, než tranformace lineární kombinace vektorů. To bylo způsobeno pravděpodobně malým množstvím trénovacích vět pro transformační matici. Pro získání více vět se přeložily i zbývající datasety, na kterých byly metody testovány. Po přeložení se tak navýšil počet párů vět na 22 434, k trénování transformační matice, ale nejde použít všechny. Nejprve se musí z těchto vět odstranit věty, na kterých se bude transformace testovat. Na tomto korpusu paralelních vět se pak přetrénovaly transformační matice pro modely *Doc2Vec* a *Skip-thoughts*, větší korpus následně pomohl k dosažení lepších výsledků.

V tabulce 9.8 jsou uvedeny výsledky transformací na větách na datasetu *SemEval-2017*. V tabulce 9.9 jsou uvedeny teoretické maximální limity na tomto datasetu. Tyto limity byly získány trénováním transformační matice na testovacím datasetu.

Na transformacích z arabštiny do angličtiny dosáhla nejlepších výsledků metoda *Word2Vec*. U transformací z turečtiny do angličtiny pak měla nejlepší výsledky metoda *FastText*. Zatímco na transformacích ze španělštiny do angličtiny dosáhla nejlepších výsledků metoda *Doc2Vec*. Zajímavé jsou také výsledky jednotlivých metod transformace, narozdíl od transformací na slovech, nejde jednoznačně určit, která z transformací je nejlepší. Všechny transformace jsou víceméně rovnocenné.

Model	Trans.	ar-en	es-en	tr-en	průměr
GloVe	LST	0,336	0,376	0,185	0,299
	CCA	0,312	0,371	0,158	0,280
	ORT	0,318	0,364	0,179	0,287
Word2Vec	LST	0,435	0,349	0,224	0,336
	CCA	**0,440**	0,386	0,237	**0,354**
	ORT	0,434	0,358	0,249	0,347
FastText	LST	0,266	0,306	0,252	0,275
	CCA	0,260	0,312	**0,274**	0,282
	ORT	0,251	0,302	0,248	0,267
Doc2Vec	LST	0,285	0,386	0,093	0,255
	CCA	0,244	**0,403**	0,095	0,247
	ORT	0,257	**0,403**	0,080	0,247
Skip-thoughts	LST	0,292	0,123	0,170	0,195
	CCA	0,204	0,170	0,125	0,166
	ORT	0,210	0,177	0,047	0,145

Tabulka 9.8: Výsledky transformací na větách na vícejazyčných datasetech z konference *SemEval-2017*. Uvedené hodnoty jsou Pearsonovy korelace.

Model	Trans.	ar-en	es-en	tr-en
GloVe	LST	0,499	0,554	0,393
	ORT	0,442	0,526	0,384
Word2Vec	LST	0,524	0,478	0,478
	ORT	0,520	0,442	0,487
FastText	LST	0,317	0,546	0,403
	ORT	0,461	0,459	0,395
Doc2Vec	LST	0,382	0,509	0,237
	ORT	0,407	0,524	0,262
Skip-thoughts	LST	0,326	0,194	0,287
	ORT	0,292	0,336	0,256

Tabulka 9.9: Teoretické maximální limity transformací na větách na vícejazyčných datasetech z konference *SemEval-2017*. Uvedené hodnoty jsou Pearsonovy korelace.

V tabulce 9.10 jsou výsledky pro dataset *GoranGlavas*. Na tomto datasetu dosáhla nejlepších výsledků metoda *Word2Vec*, která měla průměrné

výsledky lepší než ostatní metody. Velmi dobrých výsledků dosáhly též metody *GloVe* a *FastText* na transformacích ze španělštiny a italštiny, bohužel obě metody měly horší výsledky na transformacích z chorvatštiny do angličtiny. Špatných výsledků pak dosáhly metody *Doc2Vec* a *Skip-thoughts*. Zde není zcela zřejmé proč dosáhly o tolik horších výsledků než ostatní metody. Zajímavé je, že metoda *Doc2Vec* měla dobré výsledky na datasetu *SemEval-2017*, viz Tabulka 9.8.

		es-en				**it-en**				**hr-en**				**průměr**
Model	**Trans.**	MSRvid	multi	News	onWN	MSRvid	multi	News	onWN	MSRvid	multi	News	onWN	
GloVe	LST	0,601	0,560	0,586	0,381	0,596	0,545	0,635	0,348	0,457	0,456	0,559	0,281	0,500
	CCA	0,587	0,564	0,567	0,365	0,575	0,565	0,582	0,324	0,414	0,428	0,499	0,266	0,478
	ORT	0,574	0,578	0,559	0,347	0,597	**0,571**	0,594	0,315	0,472	0,463	0,547	0,276	0,491
Word2Vec	LST	0,613	**0,582**	0,589	0,392	0,654	0,523	0,640	0,417	0,558	0,465	0,602	**0,380**	0,535
	CCA	0,616	0,581	0,611	0,395	**0,668**	0,500	**0,663**	0,404	0,548	0,451	0,600	0,376	0,534
	ORT	0,610	0,571	**0,642**	0,387	0,666	0,562	0,659	0,412	0,573	**0,502**	**0,615**	0,368	**0,547**
FastText	LST	0,600	0,336	0,492	0,340	0,619	0,362	0,565	0,362	0,534	0,242	0,441	0,248	0,428
	CCA	0,593	0,366	0,509	0,377	0,626	0,393	0,601	0,367	0,527	0,282	0,446	0,256	0,445
	ORT	**0,627**	0,454	0,520	**0,421**	0,650	0,400	0,595	**0,437**	**0,574**	0,297	0,490	0,302	0,481
Doc2Vec	LST	0,070	0,119	0,440	0,154	0,177	0,070	0,413	0,105	0,146	0,050	0,325	0,127	0,183
	CCA	0,087	0,161	0,400	0,142	0,168	0,085	0,382	0,099	0,152	0,065	0,315	0,114	0,205
	ORT	0,098	0,160	0,454	0,181	0,191	0,102	0,435	0,116	0,162	0,059	0,368	0,137	0,125
Skip-thought	LST	0,274	0,233	0,275	0,278	0,124	0,173	0,287	0,237	0,086	0,046	0,250	0,214	0,206
	CCA	0,167	0,207	0,290	0,205	0,067	0,148	0,278	0,195	0,110	0,006	0,254	0,147	0,173
	ORT	0,202	0,249	0,309	0,227	0,111	0,186	0,310	0,193	0,105	0,053	0,226	0,158	0,194

Tabulka 9.10: Výsledky transformací na větách na vícejazyčném datasetu *GoranGlavas*. Uvedené hodnoty jsou Pearsonovy korelace.

9.5 Výsledky transformací Paragraph2Vec modelu

V tabulkách 9.11 a 9.12 jsou uvedeny výsledky transformovaného *Paragraph2-Vec* modelu, viz kapitola 7.3. Tato metoda dosáhla dobrých výsledků pouze na několika datasetech a jednoznačně nejlepší transformací byla transformace metodou nejmenších čtverců, která u ostatních přístupů dosahovala nejhorších výsledků.

Trans.	**ar-en**	**es-en**	**tr-en**	**průměr**
LST	0,199	0,305	0,040	0,181
CCA	0,160	0,271	0,020	0,150
ORT	0,167	0,280	0,018	0,155

Tabulka 9.11: Výsledky transformace *Paragraph2Vec* modelu na vícejazyčných datasetech z konference *SemEval-2017*. Uvedené hodnoty jsou Pearsonovy korelace.

	es-en				**it-en**				**hr-en**				**průměr**
Trans.	MSRvid	multi	News	onWN	MSRvid	multi	News	onWN	MSRvid	multi	News	onWN	
LST	0,224	0,255	0,521	0,232	0,204	0,237	0,540	0,170	0,246	0,174	0,497	0,250	0,296
CCA	0,097	0,164	0,427	0,186	0,097	0,130	0,366	0,131	0,196	0,057	0,384	0,166	0,200
ORT	0,105	0,165	0,417	0,181	0,099	0,128	0,359	0,120	0,191	0,039	0,366	0,136	0,192

Tabulka 9.12: Výsledky transformace *Paragraph2Vec* modelu na vícejazyčném datasetu *GoranGlavas*. Uvedené hodnoty jsou Pearsonovy korelace.

V tabulce 9.13 jsou uvedeny výsledky transformovaných modelů na monolinguálních datasetech. Tyto výsledky slouží jako ověření, že transformace byla provedena dobře, a model si zachoval schopnost generovat dobré vektory pro věty. Pro srovnání jsou v prvním sloupci uvedeny výsledky původního natrénovaného modelu před provedením transformace. Z tabulky je vidět, že transformované modely dosáhly lepších výsledků na téměř všech datasetech, kromě španělského datasetu *SemEval-2017*.

	Dataset			
Trans.	es-newswire	es-wikipedia	es-es SemEval17	ar-ar SemEval17
žádná	0,417	0,418	**0,714**	0,574
LST	**0,437**	**0,464**	0,640	0,551
CCA	0,390	0,426	0,613	**0,577**
ORT	0,403	0,413	0,615	0,573

Tabulka 9.13: Výsledky transformovaných *Paragraph2Vec* modelů na monolinguálních datasetech. Uvedené hodnoty jsou Pearsonovy korelace.

9.6 Shrnutí výsledků a porovnání

V následujících tabulkách 9.14 a 9.15 jsou srovnány nejlepší výsledky dosažené s danou metodou transformace a způsobu transformace. Na datasetu *SemEval-2017* dosáhla nejlepších výsledků transformace na větách. Zde je zajímavé, že pro každý způsob transformace dosáhla nejlepších výsledků jiná transformační metoda. Pro transformace na větách dosáhla nejlepších hodnot kanonická korelační analýza. Pro transformaci na slovech měla nejlepší výsledky ortogonální transformace. Pro transformaci *Paragraph2Vec* modelu pak byla nejlepší transformace metodou nejmenších čtverců.

	Způsob transformace					
	na slovech		na větách		Paragraph2Vec	
Trans.	**PC**	**SC**	**PC**	**SC**	**PC**	**SC**
LST	0,254	0,272	0,336	0,352	0,181	0,164
CCA	0,277	0,295	0,354	0,372	0,150	0,147
ORT	0,293	0,308	0,347	0,360	0,155	0,168

Tabulka 9.14: Srovnání dosažených průměrných výsledků na vícejazyčných datasetech z konference *SemEval-2017*.

Na datasetu *GoranGlavas* dosáhla nejlepších výsledků transformace na slovech, která zde byla o trochu lepší než transformace na větách. Transformace na slovech dosáhla nejlepších výsledků s metodou kanonické korelační analýzy, zatímco transformace na větách s metodou ortogonální transformace. Transformace *Paragraph2Vec* modelu měla opět nejlepší výsledky s transformací metodou nejmenších čtverců.

	Způsob transformace					
	na slovech		na větách		Paragraph2Vec	
Trans.	**PC**	**SC**	**PC**	**SC**	**PC**	**SC**
LST	0,546	0,546	0,535	0,518	0,296	0,322
CCA	0,568	0,559	0,534	0,531	0,200	0,228
ORT	0,566	0,556	0,547	0,532	0,192	0,222

Tabulka 9.15: Srovnání dosažených průměrných výsledků na vícejazyčných datasetu *GoranGlavas*.

10 Závěr

V této práci jsou popsány metody učení bez učitele pro vyjádření sémantické podobnosti textů ve vícejazyčném prostředí. Sémantické modely pro reprezentaci slov a textů byly trénovány na korpusu Wikipedie, která obsahuje velké množství souvislého textu ve všech jazycích. Celkem byly testovány následující modely: *Glove*, *Word2Vec*, *FastText*, *Paragraph2Vec*, *Skip-thoughts*. Všechny tyto modely jsou poměrně nové a fungují velmi dobře na monolinguálních datasetech. Cílem této práce tedy bylo otestovat a navrhnout způsoby, jak tyto modely převést do vícejazyčného prostředí.

Pro porovnávání významu dvou textů v odlišných jazycích bylo potřeba nejprve natrénovat transformační matici. Ta se v dostupných pracích trénovala vždy na vektorech slov. Přínosem této práce jsou dva nové způsoby, jak tuto transformační matici natrénovat. První způsob je transformace na větách. Místo trénování na vektorech slov, se trénuje rovnou na vektorech vět, které pak po vynásobení transformační maticí vyjadřují význam věty v cílovém jazyce. K tomu, aby bylo možné tuto transformační matici natrénovat, je potřeba paralelní korpus. Pro potřeby této práce se tak vytvořily paralelní korpusy pro všechny testované jazyky. Druhým způsobem je pak transformace *Paragraph2Vec* modelu, kde se transformují vektory slov z jednoho jazyka a vloží se spolu se slovníkem do modelu cílového jazyka.

Testování metod a měření experimentů bylo prováděno na datasetech z konferencí *SemEval* a datasetu *GoranGlavas*. Celkově byly metody trénovány na porovnávání těchto jazyků: angličtina, španělština, italština, arabština, turečtina a chorvatšina. Většina těchto metod dosáhla na všech jazycích a dostupných datasetech velmi dobrých výsledků. Nejlepší byla metoda *Word2Vec* se způsobem transformací na větách. Pro dataset *SemEval-2017* bylo nejlepší trénovat transformační matici ortogonální transformací nebo kanonickou korelační analýzou, neboť oba přístupy jsou srovnatelné. Pro dataset *GoranGlavas* byla nejlepší ortogonální transformace.

Metody by se daly jistě ještě vylepšit, ať už lepším předzpracováním textu nebo vylepšením trénovacích metod. Všechny metody mají mnoho parametrů, které většinou zůstaly nastavené na původních hodnotách. Zajímavým experimentem by také bylo zjistit, při jak velkém počtu párů vět se nejlépe natrénuje transformační matice na větách.

Literatura

Artetxe, M., Labaka, G., Agirre, E. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, s. 2289–2294, 2016. Dostupné z: `http://aclweb.org/anthology/D/D16/D16-1250.pdf`.

Blei, D. M., Ng, A. Y., Jordan, M. I. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* March 2003, 3, s. 993–1022. ISSN 1532-4435. Dostupné z: `http://dl.acm.org/citation.cfm?id=944919.944937`.

Bojanowski, P., Grave, E., Joulin, A., Mikolov, T. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*. 2017, 5, s. 135–146. Dostupné z: `http://aclweb.org/anthology/Q17-1010`.

Brychcín, T. *Distributional Semantics in Language Modeling*. PhD thesis, University of West Bohemia, 2015.

Brychcín, T. Linear Transformations for Cross-lingual Semantic Textual Similarity. *CoRR*. 2018, abs/1807.04172. Dostupné z: `http://arxiv.org/abs/1807.04172`.

Brychcín, T., Taylor, S. E., Svoboda, L. Cross-lingual Word Analogies using Linear Transformations between Semantic Spaces. *CoRR*. 2018, abs/1807.04175. Dostupné z: `http://arxiv.org/abs/1807.04175`.

Chen, P., Popovich, P. Correlation, parametric and nonparametric measures. In *Correlation, parametric and nonparametric measures*. Sage, 2012.

Chung, J., Gülçehre, Ç., Cho, K., Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR*. 2014, abs/1412.3555. Dostupné z: `http://arxiv.org/abs/1412.3555`.

Church, K. W., Hanks, P. Word Association Norms, Mutual Information, and Lexicography. *Comput. Linguist.* March 1990, 16, 1, s. 22–29. ISSN 0891-2017. Dostupné z: `http://dl.acm.org/citation.cfm?id=89086.89095`.

Firth, J. R. A Synopsis of Linguistic Theory, 1930-1955. *Studies in Linguistic Analysis*. 1957, s. 1–32.

Gabrilovich, E., Markovitch, S. Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis. In *Proceedings of the 20th*

International Joint Conference on Artifical Intelligence, IJCAI'07, s. 1606–1611, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

Glavas, G., Franco-Salvador, M., Ponzetto, S. P., Rosso, P. A Resource-Light Method for Cross-Lingual Semantic Textual Similarity. *CoRR*. 2018, abs/1801.06436. Dostupné z: `http://arxiv.org/abs/1801.06436`.

Hofmann, T. Probabilistic Latent Semantic Analysis. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, UAI'99, s. 289–296, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. Dostupné z: `http://dl.acm.org/citation.cfm?id=2073796.2073829`. ISBN 1-55860-614-9.

Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., Mikolov, T. FastText.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*. 2016.

Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R. S., Torralba, A., Urtasun, R., Fidler, S. Skip-Thought Vectors. *CoRR*. 2015, abs/1506.06726. Dostupné z: `http://arxiv.org/abs/1506.06726`.

Kuhn, H. W., Yaw, B. The Hungarian method for the assignment problem. *Naval Res. Logist. Quart*. 1955, s. 83–97.

Landauer, T. K., Foltz, P. W., Laham, D. An Introduction to Latent Semantic Analysis. *Discourse Processes*. 1998, 25, s. 259–284. Dostupné z: `http://lsa.colorado.edu/papers/dp1.LSAintro.pdf`.

Le, Q., Mikolov, T. Distributed Representations of Sentences and Documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, s. II–1188–II–1196. JMLR.org, 2014. Dostupné z: `http://dl.acm.org/citation.cfm?id=3044805.3045025`.

Linzen, T. Issues in evaluating semantic spaces using word analogies. *CoRR*. 2016, abs/1606.07736. Dostupné z: `http://arxiv.org/abs/1606.07736`.

Lund, K., Burgess, C. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*. 1996, 28, 2, s. 203–208. ISSN 0743-3808, 1532-5970. doi: 10.3758/BF03204766. Dostupné z: `http://link.springer.com/article/10.3758/BF03204766`.

Mikolov, T., Chen, K., Corrado, G., Dean, J. Efficient Estimation of Word Representations in Vector Space. *CoRR*. 2013a, abs/1301.3781.

Mikolov, T., Le, Q. V., Sutskever, I. Exploiting Similarities among Languages for Machine Translation. *CoRR*. 2013b, abs/1309.4168. Dostupné z: `http://dblp.uni-trier.de/db/journals/corr/corr1309.html#MikolovLS13`.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., Dean, J. Distributed Representations of Words and Phrases and their Compositionality. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K. Q. (Ed.) *Advances in Neural Information Processing Systems 26*. Curran Associates, Inc., 2013c. s. 3111–3119. Dostupné z: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Mnih, A., Hinton, G. E. A Scalable Hierarchical Distributed Language Model. In Koller, D., Schuurmans, D., Bengio, Y., Bottou, L. (Ed.) *Advances in Neural Information Processing Systems 21*. Curran Associates, Inc., 2009. s. 1081–1088. Dostupné z: http://papers.nips.cc/paper/3583-a-scalable-hierarchical-distributed-language-model.pdf.

Morin, F., Bengio, Y. Hierarchical Probabilistic Neural Network Language Model. In Cowell, R. G., Ghahramani, Z. (Ed.) *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, s. 246–252. Society for Artificial Intelligence and Statistics, 2005. Dostupné z: http://www.iro.umontreal.ca/~lisa/pointeurs/hierarchical-nnlm-aistats05.pdf.

Mu, J., Bhat, S., Viswanath, P. Representing Sentences as Low-Rank Subspaces. *CoRR*. 2017, abs/1704.05358. Dostupné z: http://arxiv.org/abs/1704.05358.

Pennington, J., Socher, R., Manning, C. D. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, s. 1532–1543, 2014. Dostupné z: http://www.aclweb.org/anthology/D14-1162.

Řehůřek, R., Sojka, P. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, s. 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.

Rumelhart, D. E., Hinton, G. E., Williams, R. J. Learning representations by back-propagating errors. *Nature*. October 1986, 323, s. 533–. Dostupné z: http://dx.doi.org/10.1038/323533a0.

Sahlgren, M. An introduction to random indexing. In *In Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE 2005*, 2005.

Šnajder, J., Padó, S., Agić, Ž. Building and Evaluating a Distributional Memory for Croatian. In *51st Annual Meeting of the Association for Computational Linguistics*, s. 784–789, 2013.

Wall, M. E., Rechtsteiner, A., Rocha, L. M. *Singular Value Decomposition and Principal Component Analysis*, s. 91–109. Springer US, Boston, MA, 2003. doi: 10.1007/0-306-47815-3_5. Dostupné z: http://dx.doi.org/10.1007/0-306-47815-3_5. ISBN 978-0-306-47815-4.

Xing, C., Wang, D., Liu, C., Lin, Y. Normalized Word Embedding and Orthogonal Transform for Bilingual Word Translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, s. 1006–1011, Denver, Colorado, May–June 2015. Association for Computational Linguistics. doi: 10.3115/v1/N15-1104. Dostupné z: https://www.aclweb.org/anthology/N15-1104.