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

Bakalářská práce

Testování podobnosti vět

Plzeň 2017 Jiří Hankovec

Místo této strany bude zadání práce.

Prohlášení

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

V Plzni dne 27. června 2017

Jiří Hankovec

Abstract

The goal of this bachelor thesis is to analyze the problematics of Semantic Textual Similarity (STS) and adapt current state-of-the-art methods for Czech language. STS systems are usually compared with manually annotated data. The data comes from international SemEval competition. This thesis reveals methods and evaluation process of STS and consist of dataset with two sentences rated with a score between 0 and 5. Testing is done on manually created dataset of 1200 pair of sentences. System is measured by Pearson's correlation and compared with human annotators.

Abstrakt

Cílem této práce je prozkoumat problematiku výpočtu sémantické podobnosti dvou českých vět. V této práci jsou představeny některé metody pro výpočet sémantické podobnosti jednotlivých slov i celých vět. Lidmi ohodnocené původní věty jsou převzaty z dat z mezinárodní soutěže SemEval. Testování sémantické podobnosti vět je prováděno na vytvořeném českém korpusu, který se skládá z 1200 dvojic vět bodově ohodnocených od 0 do 5 na základě jejich sémantické podobnosti. Přesnost vypočtené sémantické podobnosti u testovacích dvojic vět je měřena pomocí Pearsonovy korelace.

Poděkování

Tímto bych chtěl poděkovat Ing. Lukáši Svobodovi, za trpělivé vedení mé bakalářské práce, za cenné rady a čas, který mi věnoval.

Obsah

1 Úvod

Žijeme v době plné informačních technologií, výpočetní techniky a snažíme se, aby tato technika byla stále chytřejší, aby dokázala řešit problémy na podobné úrovni jako lidé, aby s lidmi dokázala rozumně komunikovat. Proto se obory zpracování přirozeného jazyka a umělá inteligence stávají stále populárnější.

Porovnávání sémantické podobnosti slov, slovních spojení, vět nebo i celých dokumentů se využívá v mnoha oblastech zpracování přirozeného jazyka. Určení sémantické podobnosti textů se využívá například při vyhledávání informací v textu. Metod pro výpočet sémantické podobnosti existuje mnoho, počínaje metodami, které pracují pouze se znakovými řetězci, až po metody, které využívají vektorovou reprezentaci slov nebo celých dokumentů.

Cílem této práce je testování sémantické podobnosti dvojic českých vět. Jelikož dvojice českých vět ohodnocené jejich podobností nejsou k dispozici, je jedním z cílů této práce také přeložení anglických párů vět a vytvořit tak český korpus párů vět. Dalším cílem této práce je upravit již existující systém, který porovnává podobnost dvou anglických vět tak, aby dokázal vypočítat sémantickou podobnost dvou českých vět. Poté je důležité na výsledném systému provést experimenty, které určí s jakou přesností je systém schopen určit podobnost dvou českých vět.

Struktura práce je koncipována následovně. V kapitole 2 jsou popsané základní principy zpracování přirozeného jazyka. Kapitola 3 se věnuje popisu metod, které se používají pro porovnávání textu a dále jsou zde popsány metody, které se využívají k předzpracování textu. V kapitole 4 jsou uvedeny dva typy korelace, které se používají k vyhodnocení úspěšnosti vypočtených podobností vět, přičemž v této práci byla využita Pearsonova korelace. Kapitola 5 se zabývá úpravou existujícího systému. Kapitola 6 obsahuje popis provedených experimentů a jejich výsledky.

2 Zpracování přirozeného jazyka

Pro automatické zpracování strojově čitelného textu nebo mluveného slova je vyžadována určitá znalost jazykového systému, ve kterém daný text vznikl. Zpracování přirozeného jazyka [5][10] (*NLP – Natural language processing*) je soubor výpočetních technik pro analýzu a reprezentaci přirozeně se vyskytujících textů za účelem zpracování textu na úrovni, na které by ho zpracoval člověk.

Přirozeně se vyskytující text je jakýkoliv text (mluvený nebo psaný), v nějakém existujícím jazyce, kterým lidé mezi sebou vzájemně komunikují. Aby mohl být text strojově zpracován na stejné úrovni, na jaké by ho zpracoval člověk, je potřeba použít metody z oboru umělé inteligence. *NLP*[1] je komplexní problém a spadá pod mnoho oborů (např. lingvistiku, akustiku – mluvené slovo), ale obecně se dá říci, že *NLP* patří do oboru umělé inteligence.

Samotný obor zpracování přirozeného jazyka se dále ještě dělí na dvě odlišná zaměření:

1. Zpracování přirozeného jazyka – zabývá se analýzou jazyka za účelem vytvoření jeho smysluplné reprezentace, ekvivalent k roli čtenáře nebo posluchače.

2. Generování přirozeného jazyka – zabývá se, jak již název vypovídá, generováním přirozeného jazyka z jeho reprezentace, ekvivalent k roli spisovatele nebo mluvčího.

Existuje mnoho typů (vrstev) jazykové analýzy, které lidé aplikují, když používají přirozený jazyk. Předpokládá se, že lidé normálně používají všechny tyto vrstvy analýzy, protože každá vyjadřuje různé typy významu jazyka. Kdežto *NLP* systémy mohou používat pouze některé z těchto vrstev jazykové analýzy.

[1] Natural language processing (Zpracování přirozeného jazyka)

2.1 Typy jazykové analýzy

Dílčí problémy jazykové analýzy lze uspořádat do pomyslných vrstev (rovin) uspořádaných od povrchové po hloubkovou čili významovou. Dalo by se říci, že čím více z následujících typů jazykové analýzy *NLP* systém používá, tím je schopnější a propracovanější.

2.1.1 Fonologie

Tento typ jazykové analýzy se zabývá zkoumáním zvukové stránky přirozeného jazyka, a to hlavně zvukovými rozdíly, které mají v daném jazyce schopnost rozlišovat význam.

Ve fonologické analýze se používají tři typy pravidel:

1. Fonetická pravidla – hlásky ve slovech.
2. Fonemická pravidla – variace ve výslovnosti slov.
3. Prozodická pravidla – kolísání důrazu a intonace ve větě.

NLP systém, který zpracovává mluvený vstup, zanalyzuje zvukové vlny a zakóduje je do digitalizovaného signálu, který je možné porovnat s jazykovým modelem.

2.1.2 Morfologie

Morfologie je v lingvistice věda, která zkoumá ohýbání slov (skloňování, časování) a odvozování slov pomocí předpon a přípon. Obecně se zabývá strukturou slov. Slova se skládají z *morfémů* – nejmenší jednotka jazykového systému, která nese význam.

Slovo *udivený* může být morfologicky analyzováno na tři samostatné *morfémy*: předponu *u*, kořen *div* a příponu *ený*. Význam každého morfému zůstává stejný napříč slovy, proto lidé mohou pro ně neznámé slovo rozdělit do, pro ně známých, *morfémů*. *NLP* systém může získat význam, konkrétních *morfémů*, aby mohl zjistit význam slova.

2.1.3 Lexikální analýza

V této vrstvě analýzy se určuje význam jednotlivých slov. Jeden z několika typů zpracování, který pomáhá určit význam slova, je přiřazení slovu takzvaného *POS*[2] *tagu*. Homonymům (slova souzvučná, která mají více než jeden význam) se při tomto typu zpracování přiřadí nepravděpodobnější *POS tag* podle kontextu.

V lexikální vrstvě mohou být slova, která mají pouze jeden význam nahrazena jejich synonymy. Při lexikální analýze se ve většině případů používá slovník daného jazyka, ve kterém je analýza prováděna. Tyto slovníky mohou být jednoduché (např. pouze slova a jejich *POS tagy*), nebo mohou být i složitější a obsahovat více informací o slovech.

2.1.4 Syntaktická analýza

Syntax (větná skladba) je v lingvistice odvětví zabývající se slovosledem ve větě, správným tvořením větných konstrukcí a vztahy mezi slovy ve větě. Ovšem nezabývá se významem, který daná věta má.

Syntaktická analýza formálně popisuje gramatickou strukturu věty. V obecném případě jde o velmi obtížný problém. Gramaticky správných vět je v přirozeném jazyce nekonečně mnoho a délka jedné věty teoreticky není omezena. Výsledkem syntaktické analýzy je reprezentace věty, která ukazuje strukturální vztahy mezi slovy. Syntaktická analýza textu v českém jazyce může být problematická, protože čeština nemá striktní pravidla pro uspořádání členů ve větě (čeština má volný slovosled).

2.1.5 Sémantická analýza

Sémantika je v lingvistice odvětví, které se zabývá významem výrazů z různých strukturních úrovní jazyka (slov, slovních spojení, frází, vět), dále se také zabývá jejich vztahů ke skutečnosti, kterou označují.

Při sémantickém zpracování se určuje možný význam věty. Význam věty se určuje pomocí významů jednotlivých slov a vztahů mezi nimi. Tento typ zpracování může zahrnovat proces zvaný sémantická desambiguace (zjedno-

[2]Part of Speech (Slovní druh)

značnění) slov. Zjednoznačnění slov umožňuje určit slovům s více významy jediný význam a vybrat ho do sémantické reprezentace věty. Pro zjednoznačnění slov existuje velké množství metod, některé z nich pracují s frekvencí výskytu slova v určitém korpusu, jiné zase mohou využívat kontext, ve kterém je slovo použité.

2.1.6 Pragmatická analýza

V lingvistice je pragmatika vědní obor, který lze definovat, jako studium významu (slov, textu) v určitém kontextu.

Tato vrstva analýzy se zabývá použitím jazyka v určitých situacích a z kontextu v dané situaci porozumět textu. Cílem je „číst mezi řádky“ a vyčíst z textu informace, které v něm nemusí být obsažené. Toto vyžaduje mnoho znalostí, včetně pochopení záměrů, plánů a cílů daného textu. Proto *NLP* systémy pro tento typ analýzy mohou využívat například znalostní báze.

2.2 Oblasti použití zpracování přirozeného jazyka

Zpracování přirozeného jazyka se využívá v mnoha aplikacích, ve skutečnosti každá aplikace, která pracuje s textem může používat *NLP*. Některé z aplikací, které využívají zpracování přirozeného jazyka jsou:

- Strojový překlad – jedná se o aplikaci, která se snaží o automatický překlad textu z jednoho přirozeného jazyka do jiného. Využívají se zde různé vrstvy analýzy.
- Syntéza řeči – je generování lidské řeči například z nějakého psaného textu.
- Rozpoznání řeči – je, na rozdíl od syntézy řeči, převod z mluvené řeči na text.
- Výtah z textu – jedná se, jak už název napovídá, o zkrácení původního textu (výtah důležitých informací), na kratší text, který bude shrnovat informace z původního textu.
- Vyhledávání informací – vyhledávání určité informace z nějakých informačních zdrojů.

- Extrakce informací – zaměřuje se na rozpoznání, označení a extrakci informací do určité strukturované reprezentace. Tato strukturovaná reprezentace může být dále použita v nějakých aplikacích (např. odpovídání na otázky).
- Odpovídání na otázky – vypíše uživateli textovou odpověď na jeho dotaz. Na rozdíl od vyhledávání informací, které uživateli poskytuje seznam dokumentů obsahujících vyžadovanou informaci.

3 Metody pro porovnávání textu

Další oblastí zpracování přirozeného jazyka je porovnávání sémantické podobnosti textu [8] (*STS – Semantic Textual Similarity*), je to jedna ze základních disciplín tohoto oboru. Předpokladem pro porovnávání je, že máme alespoň nějaké dva textové fragmenty (slova, fráze, věty, odstavce nebo i celé dokumenty) a cílem porovnávání je odhadnout sémantickou podobnost těchto textových fragmentů.

U jednotlivých slov se ještě může rozlišovat mezi sémantickou podobností a souvislostí. Sémantická podobnost dvou slov udává, jak moc jsou si daná slova významově podobná. Čím větší je podobnost dvou slov, tím je větší pravděpodobnost, že při prohození těchto slov ve větě, bude věta mít stejný význam. Kdežto souvislost dvou slov udává, jaká je pravděpodobnost, že se slova mohou vyskytovat ve stejném kontextu.

Cílem této práce je přeložit věty z anglického jazyka do českého a následně porovnávat sémantickou podobnost těchto vět. Systémy pro porovnávání podobnosti textu většinou pracují s daty (v tomto případě páry vět k porovnání), která byla nejprve ohodnocena lidmi. Příklad dvou vět k porovnání:

```
Dva psi si hrají na dvorku.
Dva psi si hrají na trávě.
```

Tyto dvě věty jsou ohodnoceny mírou podobnosti (skórem). U tohoto příkladu jsou věty ohodnoceny skórem 4,6. Tato míra podobnosti se hodnotí od 0 do 5, kde skóre 0 znamená, že si věty nejsou vůbec významově podobné a skóre 5 znamená, že věty mají stejný význam.

Metod pro výpočet sémantické podobnosti textových fragmentů, v našem případě dvou vět, existuje mnoho [11]. Tyto metody se mohou používat samostatně nebo se mohou použít jejich různé kombinace. Velké množství těchto metod je úzce spjato se strojovým učením.

3.1 Strojové učení

Strojové učení [14](*Machine learning*) je vědní disciplína, která patří do oboru umělé inteligence. Jedná se o procesy, algoritmy, které umožňují systému se „učit“. „Učením“ se rozumí automatické zlepšování systému (např. jeho struktury, chování nebo znalostí) na základě nějakých zkušeností.

Základním a nejdůležitějším principem strojového učení je generalizace systému, což je schopnost vhodně a dostatečně natrénovaného sytému pracovat s daty, které nikdy neviděl (data, která nebyla součástí trénovací množiny).

Oblasti využití strojového učení lze nalézt téměř ve všech oborech lidské činnosti (lékařství, biologie, ekonomika, robotika, výzkum vesmíru atd.). Strojové učení se obvykle dělí na dvě hlavní kategorie [15] – učení s učitelem a učení bez učitele.

3.1.1 Učení s učitelem

Základním předpokladem učení s učitelem (*Supervised learning*) je existence trénovací množiny dat. Trénovací množina se skládá z dvojic vstupních objektů x (typicky vektor příznaků) a požadovaného výstupu y. Cílem je natrénovat systém pomocí trénovací množiny, aby ze vstupních objektů dokázal vyprodukovat výstup.

$$D = \{(x_i, y_i)\}_{i=1}^{N}, \tag{3.1}$$

kde D je trénovací množina a N je počet jednotlivých trénovacích příkladů. Příkladem trénovacího vstupu x_i může být vektor čísel (např. vektor obsahující dvě čísla reprezentující výšku a váhu člověka), ale obecně se může jednat o jakýkoliv strukturovaný objekt (např. dvě věty). Stejně tak výstup y_i může být prakticky jakýkoliv. Pokud je výstupem konečná množina tříd (např. třídy *spam* a *ham*), jedná se o klasifikační problém (klasifikaci) – rozdělení dat do několika tříd (kategorií). Pokud počet výstupních tříd je v intervalu reálných čísel, jedná se o regresní problém (regresi).

3.1.2 Učení bez učitele

Na rozdíl od učení s učitelem, učení bez učitele (*Unsupervised learning*) nemá k dispozici žádné znalosti o třídě výstupu, trénovací množina se skládá pouze ze vstupních dat.

$$D = \{(x_i)\}_{i=1}^{N}. \tag{3.2}$$

Podstatou činnosti učení bez učitele je hledání společných vlastností vstupních dat. Jednou z metod k nalezení společných vlastností dat je shluková analýza, která seskupuje data do různých skupin na základě vzájemné podobnosti nebo odlišnosti dat.

Učení bez učitele se může používat u následujících aplikací:

- Automatická klasifikace dokumentů
- Analýza sociálních sítí
- Segmentace trhu pro účely marketingu
- Strojový překlad
- Předpovídání živelných katastrof
- Analýza astronomických dat

3.2 Metody pro předzpracování textu

Metody předzpracování textu se využívají téměř ve všech oblastech zpracování přirozeného jazyka. Proto před samotným použitím metod pro výpočet sémantické podobnosti textových fragmentů, je potřeba obsah těchto fragmentů určitým způsobem předzpracovat, aby se s ním dalo v těchto metodách dále pracovat. V této kapitole budou dále popsány základní metody pro předzpracování textu.

3.2.1 Tokenizace

První z metod předzpracování textu je tokenizace, což je proces, který rozdělí text na informační celky – tzv. *tokeny*. Zároveň tento proces může, ale nemusí odstranit interpunkci a jiné speciální znaky.

Token je jednotka textu, většinou se jedná o grafické slovo (řetězec znaků oddělený mezerou nebo některými speciálními znaky). Interpunkce a i jiné speciální znaky mohou být samostatné *tokeny*.

Příklad tokenizace:

Text před tokenizací: `Pes stojí v písku.`

Text rozdělený na tokeny:

`Pes`	`stojí`	`v`	`písku`	`.`

3.2.2 Lemmatizace

Lemmatizace je proces, kdy se slovům (*tokenům*), přiřadí jejich slovníkový tvar. V podstatě se jedná o hledání základních tvarů slov (u podstatných jmen – první pád, jednotné číslo, u sloves – infinitiv atd.). Problém v tomto procesu může nastat, pokud má slovo více významů, v tomto případě se přiřadí *tokenu* jeho základní tvar na základě kontextu.

Lemma je základní tvar (slovníková podoba) slova. Například pro sloveso *jsou* bude jeho lemma sloveso v infinitivu, tedy *být*.

3.2.3 Stemming

Stemming je metoda, která „ořezává“ od daného slova (*tokenu*) jeho předpony, přípony a koncovky (jedná se o snahu nalezení „kmenu“ slova). Nalezený *stem* slova nemusí mít žádný význam, jelikož metoda pouze odstraní předpony, přípony a koncovky, například pokud se u slova *kočka* odstraní koncovka, vznikne *stem kočk*, který sám o sobě nemá žádný význam.

3.2.4 Part-of-speech tagging

Tato metoda přiřadí každému slovu (*tokenu*) ve větě *POS tag* (informaci o slovním druhu, popřípadě další informace, například u podstatných jmen

jejich pád, rod, číslo atd.)

POS tag je ve většině případů nějaká značka (například řetězec znaků, který reprezentuje informace o daném *tokenu* ve větě). Příklad jednoduchého *POS tagu* může být třeba pouze jeden znak, který reprezentuje slovní druh daného slova. U slovního spojení `Bílý trpaslík` by potom *POS tagy* pro jednotlivá slova byly:

Slovo `Bílý` by mělo *POS tag* – „A“ (Adjective – přídavné jméno)

Slovo `trpaslík` by mělo *POS tag* – „N“ (Noun – podstatné jméno)

3.3 Metody založené na znalostních bázích

Tento přístup se používá převážně pro výpočet sémantické podobnosti jednotlivých slov. Princip těchto metod je založen na poloze slov v ontologii, přičemž podobná slova leží blízko sebe. Proto základem pro tyto metody jsou vztahy mezi jednotlivými slovy (synonyma, antonyma, hyperonyma, hyponyma atd.).

Ontologie Jedná se o explicitní, formální popis určité problematiky. Ontologie je složena z definic pojmů a vztahů mezi těmito pojmy. Datový model ontologie, reprezentující určité znalosti, obsahuje následující čtyři základní prvky:

1. Jedinec – je to základní prvek ontologie, může se jednat o živý nebo neživý objekt.
2. Třída – jedná se o množinu jedinců, která má určité společné vlastnosti (jedinci určitého typu).
3. Atributy – popisují určité vlastnosti, charakteristiku či parametry jedince.
4. Vazba – už podle názvu je patrné, že se jedná o nějaké propojení mezi dvěma jedinci, toto propojení může být, buďto jednostranné nebo oboustranné.

3.3.1 Manuálně vytvářené ontologie

Tento princip je časově velice náročný, protože se musí ručně vytvořit a označkovat dostatečně rozsáhlá databáze slov. Jednou z nejznámějších a největších lexikálních databází je *WordNet* [2]. *WordNet* je ručně vyráběná databáze slov pro anglický jazyk.

Slova v této databázi jsou spojována do takzvaných *synsetů* (množina synonym), slova v *synsetech* mají stejný sémantický význam. Pro každý slovní tvar je definován jeho význam a jeho sémantické vazby na ostatní slovní tvary (např. synonyma).

3.3.2 Automaticky generované ontologie

Výhodou automaticky generovaných ontologií je, že obsahují značně větší množství dat než manuálně vytvářené ontologie. Na druhou stranu data v těchto ontologiích nejsou zdaleka tak přesná. *YAGO* nebo *DBPedia* jsou asi jedni z nejznámějších zástupců automaticky generovaných ontologií.

3.4 Lexikální a syntaktická podobnost

V této kapitole budou popsány metody, které pracují s lexikálními a syntaktickými informacemi v textu. Mnoho z těchto metod využívá slova ohodnocená pomocí metriky TF-IDF.

3.4.1 TF-IDF

Tato metoda ohodnotí každé slovo jeho relevancí pro daný dokument. Relevance slova v dokumentu se spočítá pomocí kombinace dvou hodnot *tf* a *idf*. Dále se toto ohodnocení slov používá pro velké množství dalších metod.

Četnost slova v dokumentu (*tf – Term frequency*) je hodnota, kolikrát se slovo vyskytuje v dokumentu. V některých případech se tato hodnota normalizuje délkou dokument (počtem slov v dokumentu). V jiných případech se může používat logaritmická četnost slova. Vzorec pro výpočet složky *tf* není pevně daný.

Normalizovaná četnost slova:

$$tf(t_i, d_j) = \frac{n_{i,j}}{\sum_{k=1}^{N} n_{k,j}}, \tag{3.3}$$

kde $n_{i,j}$ představuje počet výskytů slova t_i v určitém dokumentu d_j, ve kterém je N slov.

Logaritmická četnost slova:

$$tf(t_i, d_j) = 1 + \log tf(t_i, d_j). \tag{3.4}$$

Inverzní četnost v dokumentech (*idf – Inverse document frequency*) udává důležitost daného slova a vypočítá se pomocí následují rovnice:

$$idf(t_i, D) = \log \frac{|D|}{df_i}, \tag{3.5}$$

kde $|D|$ je počet všech dokumentů a df_i představuje počet dokumentů, ve kterých se slovo t_i vyskytuje. Z této rovnice je tedy patrné, že v čím více dokumentech se dané slovo vyskytuje, tím je významově méně důležité. Výsledná hodnota *tf-idf* se může vypočítat jednoduchým vynásobením hodnot *tf* a *idf*.

3.4.2 LCS (Longest Common Substring)

LCS nebo také nejdelší společný podřetězec, je algoritmus pro nalezení nejdelšího společného podřetězce libovolného počtu řetězců. Jelikož cílem práce je porovnávání dvou textových fragmentů (vět) budou se porovnávat pouze dva řetězce.

Algoritmus nalezení nejdelšího společného podřetězce může být definován vztahem:

$$LCS(S1, S2) = \max_{1 \leq i \leq m, 1 \leq j \leq n} LCS_{Suff}(S1_{1 \cdots i}, S2_{1 \cdots j}), \tag{3.6}$$

kde m je délka řetězce $S1$ a n je délka řetězce $S2$. A LCS_{Suff} je funkce, kterou lze jednoduše definovat jako hledání společných sufixů všech možných prefixů řetězců $S1$ a $S2$.

3.4.3 Překrytí n-gramů

Překrytí n-gramů (*n-gram overlaps*) je metoda, která rozdělí oba dva porovnávané textové fragmenty na několik n-gramů, záleží na délce textového fragmentu.

U porovnávání podobnosti dvou vět existuje několik možností, jaké n-gramy lze vytvořit. Mohou se použít jednotlivá slova ve větách pro vytvoření dvou množin n-gramů, které se následně budou porovnávat. U tohoto případu se jedná o překrytí slov, pokud se pracuje s „unigramy“. Můžeme pracovat i s předzpracovanými větami a pro vytvoření n-gramů použít lemmata jednotlivých slov. Máme tedy dvě množiny n-gramů, které mezi sebou porovnáme. Shoda nastane, pokud se daný n-gram vyskytuje v obou množinách. Ohodnocení n-gramů pro výpočet koeficientu podobnosti (viz níže) může být pro každý n-gram hodnota 1 nebo lze použít sofistikovanější metodu a ohodnotit každý n-gram součtem hodnot *ifd* (viz kapitola 3.4.1) pro jednotlivá slova v n-gramu.

Pro výpočet syntaktické podobnosti dvou vět můžeme použít n-gramy složené z *POS tagů* jednotlivých slov ve větách. Nemusí se však pracovat pouze se slovy ve větách, ale lze použít i jednotlivé znaky.

n-gram

Pro metody, které pracují s textem, je n-gram posloupnost n po sobě jdoucích položek v určitém textu. Tyto položky mohou být jednotlivé znaky, slova, slovní spojení, klidně i jednotlivé věty (obsah položek muže být v podstatě jakýkoliv). Pokud n-gram obsahuje pouze jednu položku, nazývá se „unigram“, posloupnost dvou položek je „bigram“, n-gram složený ze tří položek je „trigram“.

Výpočet koeficientu podobnosti

Například pro slova *závod* a *voda* použijeme rozdělení na trigramy. Pro slovo *závod* nám tedy vzniknou trigramy: *záv*, *ávo* a *vod*. Pro slovo *voda* budou „trigramy“ pouze dva: *vod* a *oda*. Existuje několik možností, jak pro vzniklé množiny vypočítat koeficient jejich podobnosti. Jednotlivé n-gramy, jak již bylo zmíněno výše, mohou být ohodnoceny hodnotou 1 nebo součtem hodnot *idf*.

Jaccardův koeficient podobnosti:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}, \tag{3.7}$$

kde A a B jsou množiny n-gramů. Z rovnice je patrné, že pokud jsou obě množiny stejné (např. stejná slova), výsledný koeficient bude 1. Hodnoty koeficientu podobnosti se pohybují v intervalu od 0 do 1. Pro výše uvedený příklad dvou slov, by výsledný koeficient byl 0,25. Slova mají jeden společný trigram *vod* a po sjednocení množin je počet unikátních trigramů roven čtyřem.

3.5 Distribuční sémantika

V této kapitole budou popsány metody, které jsou více zaměřeny na sémantický význam jednotlivých slov v textu. Tyto metody jsou založeny na distribuční hypotéze, která říká, že slova, která se vyskytují v podobných kontextech mají podobný význam. Základním přístupem těchto metod je mapování slov do vektorového prostoru s vysokou dimenzí. Podobná slova leží blízko u sebe v tomto vektorovém prostoru, to znamená, že mají podobné vektory, které tyto slova reprezentují. K vytvoření těchto vektorů je zapotřebí velké množství textu, ze kterého se pomocí metod statistické analýzy určí, v jakých kontextech se dané slovo vyskytuje.

Sémantickou podobnost dvou slov lze spočítat jako vzdálenost (podobnost) jejich vektorů. Pro porovnání dvou vět se mohou vytvořit vektory, které budou reprezentovat jednotlivé věty. Tyto vektory se dají vytvořit lineární kombinací vektorů jednotlivých slov obsažených v těchto větách a následně lze vypočítat vzdálenost těchto nově vzniklých vektorů. Metod založených

na distribuční hypotéze je mnoho, zde jsou popsány pouze některé z nich.

3.5.1 Vzdálenost vektorů

Vzdálenost (podobnost) dvou vektorů se může vypočítat několika způsoby. Jedním z nejznámějších způsobů je použití *euklidovské vzdálenosti* dvou vektorů.

Euklidovská vzdálenost:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{N}(y_i - x_i)^2}, \tag{3.8}$$

kde $\mathbf{x}$ a $\mathbf{y}$ jsou vektory reprezentující slovo a N je dimenze vektorů. Ve většině případů se ale pro určení podobnosti dvou vektorů používá kosinová podobnost, která určí podobnost dvou vektorů jako kosinus úhlu, který svírají.

Kosinová podobnost:

$$cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} = \frac{\sum_{i=1}^{N} x_i y_i}{\sqrt{\sum_{i=1}^{N} x_i^2 \sum_{i=1}^{N} y_i^2}} \tag{3.9}$$

3.5.2 LSA

Latentní sémantická analýza – *LSA* [11] (*Latent semantic analysis*) je asi jedna z nejznámějších metod distribuční sémantiky. Základem této metody je vytvoření matice A, matice výskytů jednotlivých slov v dokumentech (dokument může představovat i jednotlivé věty), pro ohodnocení těchto slov se používají hodnoty *tf-idf* pro dané slovo. S rostoucím celkovým počtem slov v dokumentech nebo s rostoucím počtem dokumentů roste dimenze této matice. Dále se tedy na matici A použije metoda *SVD* (*Singular Value Decomposition*), která rozloží matici na tři samostatné matice pomocí následující

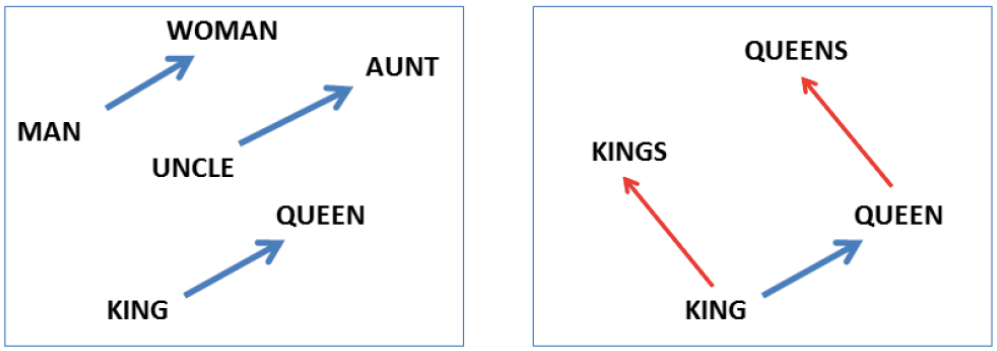
rovnice:

$$A = USV^T, \tag{3.10}$$

kde matice U je ortogonální matice a V je transponovaná ortogonální matice. Sloupce v matici U jsou ortogonální vlastní vektory $A \times A^T$ a sloupce matice V jsou ortogonální vektory $A^T \times A$. Matice S je diagonální matice, která obsahuje odmocniny vlastních čísel z matice U nebo V, ty jsou na hlavní diagonále seřazeny v sestupném pořadí. Po tomto singulárním rozkladu jsou řádky matice U vektory jednotlivých slov a řádky matice V jsou vektory jednotlivých dokumentů.

3.5.3 Word2Vec

Word2Vec je v poslední době jedna z nejpoužívanějších metod (nástrojů) pro vytváření mnohorozměrných vektorů reprezentujících význam slova [12]. Pro vytváření vektorů jednotlivých slov tato metoda využívá neuronové sítě. Navíc vektory natrénované metodou *word2vec* obsahují informace o některých lingvistických jevech (viz obrázek 3.1).



Obrázek 3.1: Ukázka vztahů mezi vektory jednotlivých slov, obrázek pochází z [13]

Metoda *word2vec* poskytuje dva různé různé algoritmy pro natrénování vektorů. Prvním z těchto algoritmů je *CBOW* (*Continuous Bag-Of-Words*), druhý se nazývá *Skip-gram*.

Model *CBOW* se snaží na základě předem nastaveného rozsahu okna kontextu určit aktuální slovo. Tento model nebere ohled na pořadí jednotlivých slov v kontextovém okně, proto název *bag-of-words*. Na druhou stranu, model *skip-gram* se snaží předpovědět kontext daného slova v rozsahu kontextového okna.

3.6 Kombinace metod pro porovnávání textu

Pro určení sémantické podobnosti dvou vět lze použít výše popsané metody samostatně, nebo může být použita kombinace těchto metod. Kombinace těchto metod je v podstatě regresní problém. Cílem regresních metod je nalézt mapování ze vstupních hodnot na výstupní hodnoty (v tomto případě podobnost dvou vět).

Vstupní hodnoty zde budou výsledky jednotlivých metod pro porovnávání vět, lze je uspořádat do d-rozměrného vektoru $\mathbf{x}$ z prostoru $\mathbb{R}^d$, kde d je počet použitých metod.

Pro nalezení výsledných podobností z kombinací výsledků jednotlivých metod, může být použito učení s učitelem. Natrénuje se model z trénovacích dat, což jsou dvojice: dvě věty a jejich podobnost (podobnost těchto dvou vět anotovaná lidmi). Takto natrénovaný model poté pomocí nějaké regresní metody ze vstupu dvou vět vyhodnotí jejich podobnost.

Regresních metod existuje velké množství, některé z nich jsou uvedené níže, pro ukázku je zde popsaná jen jedna 3.6.1.

- Lineární regrese
- SVM (*Support Vector Machine*) [16]
- Regresní rozhodovací stromy
- Regrese pomocí neuronových sítí

3.6.1 Lineární regrese

Jedná se asi o nejjednodušší způsob regrese. Cílem je nalezení hypotézy h, pomocí které lze mapovat vstupní hodnoty x na výstupní y. Hypotéza h pro

lineární regresy vyjadřuje rovnice 3.11.

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n = \theta^T x, \tag{3.11}$$

kde x_0 pro všechna natrénovaná data se rovná 1 a θ je vektor parametrů dané rovnice. Pro nalezení těchto parametrů lze použít například metodu nejmenších čtverců.

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2, \tag{3.12}$$

kde m je počet trénovacích dat, $x^{(i)}$ a $y^{(i)}$ jsou dvojice trénovacích dat. Cílem je nalézt parametry θ, aby hodnota $J(\theta)$ byla co nejmenší.

4 Korelace výsledných podobností

Pro testovací data (dvojice vět k porovnání) získáme výslednou podobnost jednotlivých vět pomocí metod popsaných v kapitole 3. Máme tedy podobnost dvou vět určenou lidmi a podobnost získanou použitím metod pro porovnávání vět. Nyní by bylo vhodné určit, jak byly tyto metody úspěšné. Toho lze dosáhnout pomocí korelace. Obvykle se používají dva základní typy korelace: *Pearsonova korelace* a *Spearmanova korelace*.

Výslednou korelaci vyjadřuje reálné číslo v intervalu od -1 do 1. Čím více se koeficient korelace v absolutní hodnotě blíží 1, tím více jsou veličiny více lineárně závislé.

4.1 Pearsonova korelace

Tento typ korelace udává míru lineární závislosti dvou veličin, je definován následujícím vztahem:

$$\rho = \frac{cov(X, Y)}{\sigma(X) \times \sigma(Y)}, \tag{4.1}$$

kde $cov(X, Y)$ je kovariance vyjádřená rovnicí 4.2 a $\sigma(X)$ je směrodatná odchylka X definovaná rovnicí 4.3.

$$cov(X, Y) = E(X \times Y) - E(X) \times E(Y), \tag{4.2}$$

$$\sigma(X) = \sqrt{E(X^2) - E^2(X)}, \tag{4.3}$$

kde $E(X)$ je střední hodnota X a $E(Y)$ je střední hodnota Y. Výpočet

výběrového korelačního koeficientu udává rovnice 4.4

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}, \tag{4.4}$$

kde $\bar{x}$ a $\bar{y}$ představují výběrové průměry.

4.2 Spearmanova korelace

Spearmanova korelace udává míru lineární závislosti mezi pořadím jednotlivých položek vektorů, r_x a r_y jsou vektory, které reprezentují pořadí jednotlivých hodnot vektorů x a y. Potom je korelace definována rovnicí 4.5.

$$r = 1 - \frac{6\sum_{i=1}^{n}(r_{x_i} - r_{y_i})^2}{n(n^2 - 1)}, \tag{4.5}$$

kde n je počet prvků jednoho z vektorů.

5 Popis řešení

Jednotlivé dvojice vět, pro které má být vypočtena jejich sémantická podobnost, jsou uspořádány v textových souborech. Na jedné řádce je dvojice vět k porovnání, tyto věty jsou od sebe odděleny tabulátorem. Soubory, které obsahují dvojice vět jsou označeny jako *input* (např. *STS.input.images.txt*). K těmto souborům existuje vždy ještě další textový soubor označený *gs* (*golden standard*), ten vždy obsahuje na každém řádku podobnost dvou vět ohodnocenou několika lidmi a následně zprůměrovanou do hodnoty v tomto souboru. Pro tyto dvojice souborů (např. *STS.input.images.txt* a *STS.gs.images.txt*) platí, že dvojice vět a jejich podobnost jsou v obou souborech vždy na stejném řádku.

Všechny věty pro testování podobnosti byly dodány v angličtině. Jelikož cílem této práce bylo testování podobnosti dvojic vět v českém jazyce, bylo potřeba vytvořit český korpus. Toho nešlo docílit jinak, než anglické věty ručně přeložit do češtiny. Bylo přeloženo 1200 párů vět, podobnost těchto vět byla ponechána z původních anglických vět.

Základem této práce byl dodaný systém *STS* od skupiny *NLP* Fakulty aplikovaných věd ZČU. Vývojáři tohoto systému se s ním účastnili řešení problému výpočtu sémantické podobnosti textu – *SemEval's STS 2016*. Tento problém patří k jednomu z nejpopulárnějších úkolů soutěže *SemEval*. S výsledky získanými pomocí tohoto systému se umístili na krásném druhém místě.

5.1 Systém STS

Sytém *STS* [3] je implementovaný v programovacím jazyce *Java*, slouží jednak pro výpočet sémantické podobnosti dvou vět v jednom jazyce (angličtina), ale i pro porovnávání dvou vět v různých jazycích, konkrétně pro výpočet podobnosti vět v angličtině a ve španělštině. Pro výpočet podobnosti mezi anglickými a španělskými větami, byl nejdříve použitý *Google translator* pro přeložení španělských vět do angličtiny, a dále už se vzájemně porovnávaly dvě anglické věty.

Jelikož metody pro výpočet podobnosti textu většinou pracují s již předzpracovaným textem, zásadní částí systému je použití metod pro předzpracování textu. Systém *STS* pro předzpracování textu (*tokenizaci*, *lemmatizaci*, *POS tagging* atd.) využívá knihovnu *Standford CoreNLP*, která poskytuje nástroje pro analýzu přirozeného jazyka. Tato knihovna obsahuje metody pro předzpracování textu pouze pro některé lidmi běžně používané jazyky, bohužel čeština nepatří mezi jazyky, které tato knihovna podporuje.

Model pro určení sémantické podobnosti je natrénován pomocí učení s učitelem (systém dokáže vytvořit model i pomocí učení bez učitele, ale tento způsob nebyl v této práci použitý), kde jsou jako trénovací data použity předzpracované věty a jejich podobnost.

Dále jsou implementovány metody pro výpočet podobnosti jednotlivých párů vět. Tyto metody jsou použity jako příznaky (features) pro trénování modelu. Z výsledků těchto metod jsou vytvořeny vektory reálných čísel (podobností vypočítaných pomocí těchto metod pro jednotlivé páry vět). Tyto vektory slouží jako vstupní trénovací data. Některé z použitých metod pro výpočet podobnosti jsou popsány v kapitole 5.1.1.

5.1.1 Metody systému STS pro výpočet podobnosti

Metody pro výpočet podobnosti, které systém *STS* využívá lze rozdělit do následujících skupin.

Lexikální a syntaktické

- Překrytí n-gramů jednotlivých slov nebo jejich lemmat. Systém pracuje s různými hodnotami (od 1 do 4) pro n (počet slov obsažených v jednom n-gramu). Ohodnocení n-gramů se vypočte součtem hodnot *idf* jednotlivých slov v n-gramu. Pro výpočet podobnosti je zde využit Jaccardův koeficient podobnosti – viz kapitola 3.4.3 a *Containment Coefficient*.
- Překrytí n-gramů *POS tagů* jednotlivých slov. Hodnoty pro n používá stejné, jako metoda výše, stejně tak koeficienty pro výpočet podobnosti.
- Překrytí n-gramů jednotlivých znaků ve větě, kde n je počet znaků obsažených v jednom n-gramu.

- Dále systém používá metody pro porovnání řetězců znaků jednotlivých vět, jsou to metody: *LCS* (*Longest Common Substring*), *Longest Common Subsequence* a *Greedy String Tiling*.

Sémantické

Další metody, které systém *STS* používá pro výpočet podobnosti, jsou metody distribuční sémantiky. Tyto metody využívají reprezentaci jednotlivých slov v podobě vektorů. Jelikož se pro natrénování modelů pro tyto metody vyžaduje velké množství textu, trénuje se učením bez učitele. K vytvoření těchto modelů vývojáři systému *STS* použili algoritmy *Paragraph2Vec*, *GloVe* a *Word2Vec*.

- Sémantická kompozice – jedná se o metodu, která porovnává význam obou vět. Toho se docílí vytvořením vektorů pro reprezentaci vět. Tyto vektory se vytvoří lineární kombinací vektorů reprezentující jednotlivá slova ve větě, vektory jednotlivých slov jsou ještě ohodnoceny hodnotou *tf-idf*, aby se vzala v potaz jejich relevance jednotlivých slov ve větě. Následně se vypočítá podobnost vět pomocí kosinové podobnosti (kapitola 3.5.1) vektorů vět.
- *Paragraph2Vec* [9] – tato metoda pracuje s vektory reprezentující větší úseky textu. Poté se pro tyto vektory vypočítá jejich podobnost pomocí kosinové podobnosti.

Další použité metody v systému *STS* jsou detailněji popsány v publikaci [3].

5.1.2 Regresní metody

Pro natrénování regresního modelu vývojáři experimentovali s různými druhy regrese:

- Lineární regrese
- Regrese gaussovských procesů
- *SVM* regrese
- Regrese pomocí rozhodovacích stromů

Všechny výše popsané regresní metody jsou implementovány v knihovně *WEKA*.

5.2 Knihovna WEKA

WEKA [1][6] je knihovna implementována v programovacím jazyce *Java*, je vyvíjena na *University of Waikato*. Jedná se o open source knihovnu, která obsahuje soubor různých algoritmů strojového učení pro zpracování úkolů z oblasti dolování dat. *WEKA* obsahuje nástroje na předzpracování, klasifikaci, regresi, shlukování a vizualizaci dat.

Základní komponenty této knihovny jsou:

- *Instances*, *Instance*
- *Filter*
- *Classifier*, *Clusterer*
- *Evaluating*
- *Attribute selection*

Instances, *Instance* jsou komponenty určené k uchovávání dat, se kterými ostatní části této knihovny pracují. Komponenta *Classifier* je část knihovny, která obsahuje algoritmy pro natrénování různých druhů klasifikátorů a regresních metod. Tyto metody byly v systému *STS* použity pro natrénování modelu.

5.3 Integrace systému lemmatizer do systému STS

Jak již bylo zmíněno v kapitole 5.1 systém *STS* pracuje s předzpracovaným textem, proto jsem musel na přeložené věty použít nějaký nástroj, který dokáže předzpracovat text v českém jazyce. To znamená použít na české věty *tokenizaci*, *lemmatizaci*, *stemming* a *POS tagging*.

Pro tokenizaci vět jsem použil *Tokenizér* založený na pravidlech. Rozlišují se dva typy *tokenizérů*: statistický (tokenizuje na základě strojového učení) nebo *tokenizér* založený na pravidlech. Použitý *tokenizér* rozdělí větu na jednotlivé *tokeny*, kde jeden *token* představuje jedno slovo, číslo, interpunkci nebo jiný speciální znak. *Tokenizér* tedy tokenizuje text podle mezer, interpunkce a ostatních speciálních znaků (např. znak pro dolar „$“).

Pro lemmatizaci [7], *stemming* a *POS tagging* jsem upravil a následně použil systém *lemmatizer*. Tento systém obsahuje již natrénované modely pro lemmatizaci a *POS tagging* pro český jazyk. Při lemmatizaci se pracuje s *tokeny*, pro které se pomocí *lemma modelu* získají jejich lemmata. U *POS taggingu* je použitý *tag model*, který pro jednotlivé *tokeny* vrací jejich *POS tagy*.

POS tag [7] je značka složená z alfanumerických znaků, některé z těchto znaků se určují pouze pro některé slovní druhy. Význam jednotlivých znaků v *POS tagu* podle jejich pořadí:

1. První znak představuje slovní druh *tokenu* (např. pokud první znak je *N* jedná se o podstatné jméno).
2. *SUBPOS* – detailnější informace o slovním druhu (např. pokud první znak bude *V* – sloveso a *SUBPOS* bude *f*, jedná se o sloveso v infinitivu).
3. Rod (např. *F* – ženský rod).
4. Číslo – zda se jedná o jednotné nebo množné číslo (např. pro slovo „potoky“ by značka byla *P* – množné číslo).
5. Pád – číslo v jakém pádu *token* je.
6. Přivlastňovací rod
7. Přivlastňovací číslo
8. Osoba – číslo od 1 do 3 podle osoby slovesa.
9. Čas (např. *R* – budoucí čas)
10. Stupeň – číslo od 1 do 3 (např. pro slovo „větší“ bude stupeň 2, pro slovo „největší“ by stupeň byl 3).
11. Negace – zda se jedná o negaci slova (např. slovo „nebyl“ je negované).

U *stemmingu* se stejně jako u tokenizace rozlišují dva typy: statistický přístup a *stemming* založený na pravidlech (vytvořená pravidla pro daný jazyk). Pro vytvoření *stemů* z *tokenů* jsem použil přístup založený na pravidlech (*CzechStemmerLight* a *CzechStemmerAgressive*). Poté jsem pro vytvoření *stemů* použil i statistický *HPStemmer* [4]. Pro testování popsané v kapitole 6 jsem používal *stemy* vytvořené pomocí *stemmeru* založeného na pravidlech – *CzechStemmerLight*.

Tyto procesy pro předzpracování dat jsem použil pro všechny soubory s přeloženými větami. Data získaná pomocí těchto procesů (lemmata, *POS tagy* a *stemy*) jsou uložena do souborů (pro každý proces samostatný soubor) ve stejné struktuře, jako jsou uloženy přeložené věty. Pro každý soubor s větami tak vznikne několik nových souborů obsahujících lemmata, *POS tagy* a *stemy* pro *tokeny* jednotlivých vět.

5.4 Abstrakce dat

Aby práce s předzpracovanými daty byla snazší a nemusely se používat všechny soubory obsahující lemmata, *POS tagy* nebo *stemy*, implementoval jsem třídu *DataToXMLConverter*. Tato třída obsahuje metody pro načtení všech souborů s předzpracovanými daty a jejich následné sloučení do *XML* souboru (viz kód 5.1).

```
<Pair ID="121">
  <Sentence>
    <Text>Egypt zvyšuje bezpečnostní opatření před pro-Morsi protesty.
    </Text>
    <Token Lemma="Egypt" POS="NNIS1" Stem="egypt">Egypt</Token>
    <Token Lemma="zvyšovat" POS="VB" Stem="zvyšuj">zvyšuje</Token>
    <Token Lemma="bezpečnostní" POS="AANP4" Stem="bezpečnostn">
    bezpečnostní</Token>
    <Token Lemma="opatření" POS="NNNP4" Stem="opatřen">opatření</Token>
    <Token Lemma="před" POS="RR" Stem="před">před</Token>
    <Token Lemma="pro" POS="RR" Stem="pro">pro</Token>
    <Token Lemma="-" POS="Z" Stem="-">-</Token>
    <Token Lemma="Morsi" POS="C" Stem="mors">Morsi</Token>
    <Token Lemma="protest" POS="NNIP4" Stem="protest">protesty</Token>
    <Token Lemma="." POS="Z" Stem=".">.</Token>
  </Sentence>
  <Sentence>
    <Text>Egypt se připravuje na masivní protesty.</Text>
```

```
19      <Token Lemma="Egypt" POS="NNIS1" Stem="egypt">Egypt</Token>
20      <Token Lemma="se" POS="P7" Stem="se">se</Token>
21      <Token Lemma="připravovat" POS="VB" Stem="připravuj">připravuje
22      </Token>
23      <Token Lemma="na" POS="RR" Stem="na">na</Token>
24      <Token Lemma="masivní" POS="AAFS4" Stem="masivn">masivní</Token>
25      <Token Lemma="protest" POS="NNIP1" Stem="protest">protesty</Token>
26      <Token Lemma="." POS="Z" Stem=".">.</Token>
27    </Sentence>
28  </Pair>
```

Kód 5.1: Ukázka jednoho páru vět uloženého v *XML* souboru.

Jak může být z ukázky kódu 5.1 patrné, *DataToXMLConverter* vytvoří *XML* soubor obsahující jednotlivé předzpracované páry vět. Element *Pair* obsahuje vždy dvě věty, atribut *ID* je číslo řádky, na které se věty v původním souboru nacházely. *Sentence* je element obsahující samotnou větu. Tento element obsahuje jak originální text věty v elementu *Text*, tak také elementy předzpracovaných *tokenů*.

5.5 Výsledný systém

Výsledný systém načte jednotlivé páry vět z *XML* souborů. Z těchto vět a jejich podobností se natrénuje regresní model. Regresní model se vytvoří a natrénuje pomocí metod implementovaných v knihovně *WEKA*. Systém používá následující metody pro určení sémantické podobnosti vět reprezentující jednotlivé příznaky (features) modelu:

- *String features*
  1. *Longest Common Subsequence*
  2. *Longest Common Substring*
  3. *Greedy String Tiling*
- *N-grams features*
  1. Překrytí *n*-gramů slov (*Word n-gram*)
  2. Překrytí *n*-gramů znaků (*Character n-gram*)
- *Vectors features*
  1. Kosinová podobnost *tf-idf* vektorů vět

2. Kosinová podobnost vektorů reprezentující věty (*Word Vector Composition Sum*)

- *Double features* – kombinace dvou příznaků. Celý set použitých příznaků se zkombinuje mezi sebou. Vždy se vynechá kombinace příznaku sama se sebou.
- *Triple features* – kombinace tří příznaků, funguje na stejném principu jako *double features*. Jedná se o kombinaci tří různých příznaků.

Různé kombinace těchto metod, představující jednotlivé příznaky, byly použity při experimentech – tyto experimenty jsou popsány v kapitole 6.

6 Experimenty

Tato kapitola obsahuje popis různých experimentů prováděných na upraveném systému *STS* pro český jazyk. Dále jsou v této kapitole uvedeny také výsledky těchto provedených experimentů.

6.1 Trénovací a testovací data

Celý český korpus obsahuje 1200 párů přeložených vět z angličtiny. Tyto věty byly rozděleny na 850 trénovacích párů (600 dvojic vět ze souboru *Images* z roku 2015 a 250 dvojic vět ze souboru *Headlines* z roku 2014) a 350 testovacích párů (100 dvojic vět ze souboru *Images* z roku 2014, 150 dvojic vět ze souboru *Images* z roku 2015 a 100 dvojic vět ze souboru *Headlines* z roku 2015). U všech těchto párů vět byla ponechána stejná podobnost, jako u anglických vět, ze kterých byly přeloženy.

Dále bylo zapotřebí vytvořit vektorovou reprezentaci jednotlivých slov, pro jejíž vytvoření potřebujeme velké množství textu v daném jazyce. K tomuto účelu jsem použil soubor obsahující český text z Wikipedie o velikosti 4,7 GB. Pro vytvoření modelu vektorů jsem použil oba algoritmy *Word2Vec* (*CBOW – Continous Bag-of-Words* a *Skip-gram*). Oba tyto algoritmy vytvořily 300 dimenzionální vektory pro 773952 slov.

Aby bylo možné provádět experimenty s různými vektory slov, tak jsem na soubor s českým textem z Wikipedie použil lemmatizaci a *stemming*. Vznikly tak dva nové soubory, které obsahují lemmata a *stemy* původního textu. Na tyto nové české korpusy jsem použil metodu *CBOW* pro vytvoření vektorů pouze pro lemmata a pro *stemy*. Opět vznikly 300 dimenzionální vektory pro 672519 lemmat a pro 401230 *stemů*.

Nakonec bylo potřeba vytvořit české *tf-idf* modely. Pro vygenerování těchto modelů jsem také použil český text z Wikipedie. Tento model obsahuje dvě *HashMap*. Obě tyto mapy používají jako klíče jednotlivé *tokeny* z celého korpusu. První mapa má pro každý klíč uložený počet jeho výskytů v celém korpusu. Elementy v druhé mapě, uložené pro jednotlivé klíče, představují hodnoty *idf* daného klíče (*tokenu*).

Tf-idf modely jsem vytvořil jednak pro slova obsažená v českém korpusu z Wikipedie, tak pro lemmata a *stemy* z vytvořených korpusů. Následně jsem vytvořil další *tf-idf* modely pouze pro sled n po sobě jdoucích znaků, k tomuto záměru byl též použit český korpus. Vygeneroval jsem *tf-idf* modely pro sled 2, 3, 4 a 5 po sobě jdoucích znaků.

6.2 Výsledky experimentů

Trénování modelu pro testování sémantické podobnosti vět je velmi náročné pro operační paměť stroje. Proto trénování modelu a následné vyhodnocení výsledné podobnosti testovacích párů vět trvá i několik desítek minut (samozřejmě také záleží na jakém stroji je trénování spuštěno). Doba běhu trénování záleží na tom, jaké příznaky (metody pro výpočet podobnosti vět) jsou použity pro natrénování modelu, a také na použité regresní metodě (např. lineární regrese).

Pro vyhodnocení výsledků jsem použil *Pearsonovu korelaci* (viz kapitola 4.1) mezi hodnotami podobností vět ohodnocenými lidmi a hodnotami podobností vět vygenerovanými systémem *STS*.

V tabulce 6.1 jsou uvedeny výsledky pro model trénovaný pouze jedním typem příznaků. Pro jednotlivé příznaky uvedené v této tabulce jsem použil pouze slova obsažená ve větách, bez jejich předzpracování (nepoužil jsem lemmatizaci ani *stemming*). Testovací korpusy *Images* z roku 2014 a *Headlines* obsahují 100 párů testovacích vět a *Images* z roku 2015 obsahují 150 párů vět. Z tabulky 6.1 je vidět, že nejlepších výsledků dosahují metody, které porovnávají řetězce znaků. Výsledky pro soubor *Images* z roku 2014 jsou výrazně horší než u druhých dvou souborů, věty v *Images* z roku 2014 jsou typově trošku odlišné od vět použitých pro trénování modelu, proto nedosahují tak dobrých výsledků. Celkově trénovací korpus je poměrně malý a zaměřený pouze na jednoduché a krátké věty.

	Pearsonova korelace		
Model	**2014**	**2015**	
Použité příznaky	**Images**	**Images**	**Headlines**
Longest Common Subsequence	0,5786	0,7120	0,6993
Longest Common Substring	0,4331	0,5444	0,5886
Greedy String Tiling	0,6043	0,7647	**0,7983**
Všechny *String features*	**0,6551**	**0,7932**	0,7932
Idf vážené *n*-gramy slov	0,5027	0,6614	0,6432
Idf vážené znakové *n*-gramy	0,5777	0,7624	0,7869
POS n-gramy	0,5247	0,5387	0,5618
Tf-idf	0,5151	0,6209	0,5892
Kompozice vektorů *CBOW*	0,6411	0,6992	0,6256
Kompozice vektorů *Skip-gram*	0,6017	0,6488	0,6673

Tabulka 6.1: Ukázka vlivu jednotlivých typů příznaků použitých při trénováni modelu (použitá regresní metoda pro trénování modelu u všech příznaků byla lineární regrese).

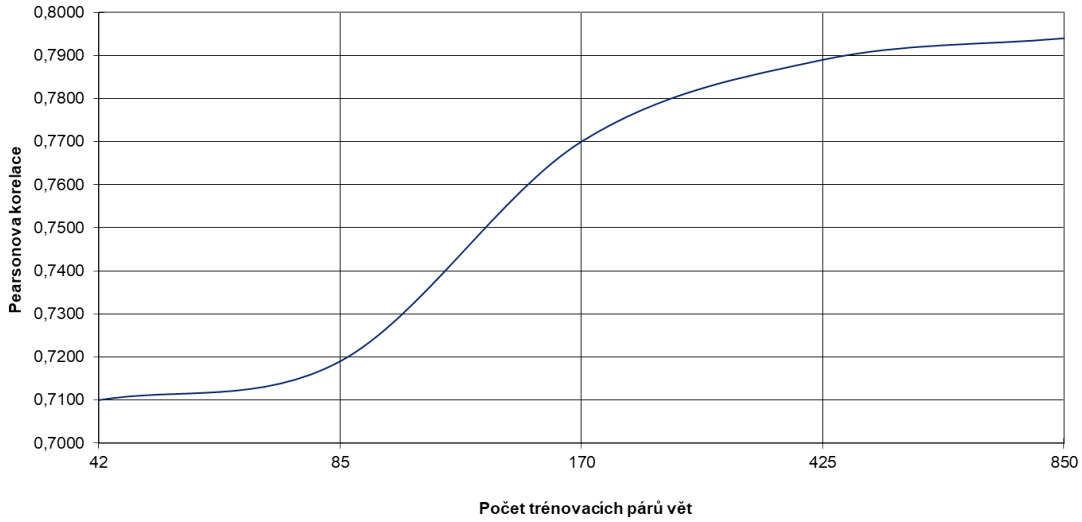
V další tabulce 6.2 jsou uvedeny hodnoty Pearsonovy korelace pro postupně přidávané příznaky do trénovaného modelu, sloupec *počet příznaků* udává, kolik bylo použito příznaků pro natrénování modelu. Jsou zde porovnané příznaky pro věty bez předzpracování, po lemmatizaci a po *stemmingu*. V této tabulce je uvedena Pearsonova korelace pro všechny tři testovací korpusy dohromady.

Nejlepších výsledků dosahuje model natrénovaný všemi příznaky s použitím *stemmingu* a modelu vektorů vytvořeným metodou *skip-gram* (Pearsonova korelace: 0,7939). Rozdíly mezi výsledky, kterých dosahují modely s použitím lemmat a modely s použitím *stemů*, jsou minimální. Rozdíly mezi výsledky, kterých dosahují modely, které pracují s větami bez použití lemmatizace nebo *stemmingu* a modely, které používají předzpracované věty jsou již výraznější. Proto lze říci, že rozšíření systému o lemmatizaci a *stemming* vedlo ke znatelnému zlepšení výsledků. Použití dvojité kombinace všech příznaků nevedlo ke zlepšení výsledků a naopak prodloužilo dobu trénování modelu.

č	**Model**	**Pearsonova korelace**	**Počet příznaků**
1.	N-gram příznaky – *word*	0.6140	9
2.	N-gram příznaky – *lemma*	0,6959	9
3.	N-gram příznaky – *stem*	0,7319	9
4.	1. + *String features – word*	0,7732	17
5.	2. + *String features – lemma*	0,7897	17
6.	3. + *String features – stem*	0,7829	17
7.	4. + syntaktické příznaky	0,7704	25
8.	5. + syntaktické příznaky	0,7860	25
9.	6. + syntaktické příznaky	0,7865	25
10.	7. + vektorové přízn. – *CBOW*	0,7796	27
11.	7. + vektorové přízn. – *skip-gram*	0,7814	27
12.	8. + vektorové přízn. – *CBOW*	0,7917	27
13.	8. + vektorové přízn. – *skip-gram*	0,7924	27
14.	9. + vektorové přízn. – *CBOW*	0,7910	27
15.	9. + vektorové přízn. – *skip-gram*	**0,7939**	27
16.	11. + použití *double features*	0,7331	352
17.	13. + použití *double features*	0,7545	352
18.	15. + použití *double features*	0,7477	352

Tabulka 6.2: Výsledky Pearsonovy korelace pro celý testovací korpus. Nárůst výsledné korelace přidáváním jednotlivých typů příznaků, porovnání použití slov, lemmat a *stemů*.

Z grafu (viz obrázek 6.1) je vidět Pearsonova korelace všech výsledných podobností testovacích vět v závislosti na počtu párů vět použitých k natrénování modelu. Zajímavé je, že největší nárůst je mezi modelem natrénovaným 85 páry vět a modelem natrénovaným 170 páry vět. Pravděpodobně je to dáno tím, že trénovací a testovací věty jsou typově hodně podobné, a proto k určení relevantní podobnosti testovacích vět stačí model natrénovaný 170 páry vět. S dále narůstajícím počtem trénovacích párů vět se nepatrně zvyšuje výsledná korelace.



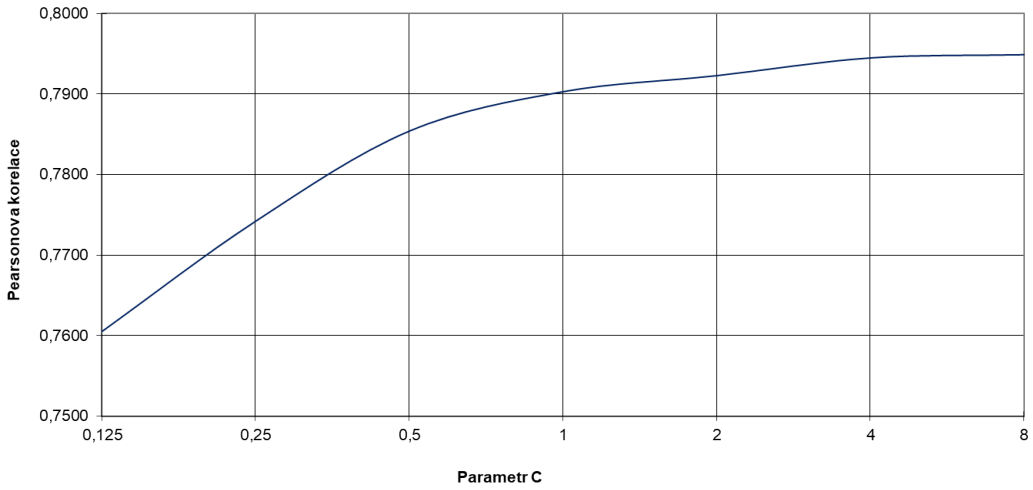
Obrázek 6.1: Výsledky Pearsonovy korelace pro modely natrénované stejnými příznaky, ale různým počtem trénovacích párů vět.

	Pearsonova korelace		
	2014	**2015**	
Model	**Images**	**Images**	**Headlines**
STS v EN *SemEval 2016*	–	0,8776	0,8398
Lineární regrese	**0,7337**	**0,8254**	0,7918
Regrese gaussovských procesů	0,7283	0,8194	0,7986
SVM regrese	0,7283	0,8238	**0,7999**

Tabulka 6.3: Pearsonova korelace podobností testovacích dat s použitím různých regresních metod (porovnání s nejlepšími výsledky původního systému *STS*, druhý nejlepší systém *SemEval 2016*). Nastavení regrese gaussovských procesů bylo – $gamma = 0,125$ a nastavení SVM regrese bylo – $gamma = 0,03125$ a $C = 8$.

V tabulce 6.3 jsou nejlepší dosažené výsledky Pearsonovy korelace podobností testovacích vět pro model natrénovaný různými druhy regrese. Pro porovnání jsou v této tabulce uvedeny nejlepší hodnoty dosažené původním systémem *STS* pro anglická data. Uvedené hodnoty pro porovnání jsou pouze orientační, protože české testovací věty jsem překládal z jiných ang-

lických vět, než pro které jsou zde uvedené korelace jejich podobností. Na obrázku 6.2 je vidět vývoj Pearsonovy korelace výsledných podobností testovacích párů vět pro měnící se parametr „C“ *SVM* regrese [16]. Jednoduše řečeno parametr „C“ určuje, jak moc přesně chceme klasifikovat každý trénovací pár, při hodně vysokém „C“ může dojít k přetrénování modelu.



Obrázek 6.2: Pearsonova korelace pro model natrénovaný SVM regresí s měnícím se parametrm „C“.

	Pearsonova korelace		
	2014	**2015**	
Model	**Images**	**Images**	**Headlines**
Nejlepší model *STS* české věty	**0,7337**	**0,8254**	0,7918
Model *STS* 850 párů angl. vět	0,6928	0,8021	0,8060
Model *STS* 3000 párů angl. vět	0,6900	0,8149	**0,8198**

Tabulka 6.4: Porovnání Pearsonovy korelace podobností testovacích vět v češtině a testovacích vět automaticky přeložených do angličtiny.

Pro poslední prováděný experiment byly české testovací věty automaticky přeloženy zpět do angličtiny použitím *Google translatoru*. Model jsem natrénoval na originálních anglických větách s použitím stejných příznaků

jako pro nejlepší model natrénovaný z českých vět. Výsledky tohoto experimentu jsou uvedeny v tabulce 6.4.

Model pro první experiment byl natrénován z 850 trénovacích párů vět. Jedná se o stejné páry vět, ze kterých, jejich ručním přeložením, vznikl český trénovací korpus. Pro druhý experiment byl model natrénován větším počtem trénovacích párů anglických vět, konkrétně jsem použil celkem 3000 párů vět z *SemEval 2014* a z *SemEval 2015*. Výsledky těchto experimentů dosahují lepších hodnot Pearsonovy korelace pouze u testovacích párů vět z *Headlines*.

7 Závěr

V rámci této práce jsem se seznámil s existujícími metodami pro výpočet sémantické podobnosti jednotlivých slov i celých vět. Jelikož metod pro výpočet sémantické podobnosti textových fragmentů je mnoho, seznámil jsem se pouze s některými z nich.

Jedním z cílů této práce bylo vytvořit český korpus skládající se z dvojic českých vět. Celkem jsem přeložil 1200 párů anglických vět do českého jazyka. Podobnost těchto nových českých párů vět byla ponechána stejná jako u původních anglických párů vět, přičemž 850 dvojic vět bylo použito pro trénování modelu a zbylých 350 párů vět bylo ponecháno pro testování jejich sémantické podobnosti.

Dalším cílem této práce bylo adaptovat existující *STS* systém na český korpus vět. Tento systém byl vyvinutý pro výpočet sémantické podobnosti dvou anglických vět. Jedna ze základních úprav systému spočívala v předzpracování českých vět. České věty byly předzpracovány pomocí systému *lemamtizer*, vzniklá lemmata, *stemy* a *POS tagy* byly uloženy do souborů. Pro zjednodušení práce s těmito daty, jsem tato data sloučil do *XML* souborů. Dále bylo zapotřebí metodám umožnit pracovat se *stemy* jednotlivých slov, protože původní systém pracoval pouze s lemmaty nebo *POS tagy*. Některé z metod pro výpočet sémantické podobnosti vět pracují s vektorovou reprezentací jednotlivých slov, tyto vektory slov jsem vytvořil pomocí metod *Word2Vec* použitých na rozsáhlém českém textu.

Po implementaci systému schopného vypočítat sémantickou podobnost dvou českých vět, jsem začal provádět experimenty. Nejprve jsem testoval každou z metod pro výpočet sémantické podobnosti vět samostatně, následně jsem zkoušel jejich různé kombinace. Pro natrénování modelu, který určuje podobnost vět, jsem použil různé regresní metody. Nejlepších výsledků dosahoval model s použitím lineární regrese natrénovaný všemi příznaky, přičemž příznaky jsou jednotlivé metody. Vylepšení systému o lemmatizaci a *stemming* vedlo k dosažení lepších výsledků.

Dle mého názoru, práce splňuje všechny body zadání. Jedním z možných vylepšení by mohlo být vytvoření rozsáhlejšího českého korpusu pro natrénování modelu, který určuje podobnost českých vět.

Literatura

[1] *Weka 3: Data Mining Software in Java* [online]. [cit. 2017/06/06]. Dostupné z: `http://www.cs.waikato.ac.nz/ml/weka/`.

[2] *WordNet A lexical database for English* [online]. [cit. 2017/05/24]. Dostupné z: `http://wordnet.princeton.edu/wordnet/`.

[3] Brychcín, T. – Svoboda, L. UWB at semeval-2016 task 1: Semantic textual similarity using lexical, syntactic, and semantic information. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*, 16, s. 588–594, San Diego, California, June 2016.

[4] Brychcín, T. – Konopík, M. HPS: High precision stemmer. *Information Processing and Management*. 2015, 51, s. 68–91.

[5] Collobert, R. et al. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*. August 2011, s. 2493–2537.

[6] Frank, E. – Hall, M. A. – H, I. *The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques"*. Fourth Edition. Morgan Kaufmann, 2016.

[7] Hana, J. et al. Manual for Morphological Annotation Revision for the Prague Dependency Treebank 2.0. Technical report, ÚFAL MFF UK, 2005.

[8] Jimenez, S. SERGIOJIMENEZ at SemEval-2016 Task 1: Effectively Combining Paraphrase Database, String Matching, WordNet and Word Embedding for Semantic Textual Similarity. In *Proceedings of SemEval-2016*, s. 749–757, San Diego, California, June 2016.

[9] Lau, J. H. – Baldwin, T. An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, s. 78–86, Berlin, Germany, August 2016.

[10] Liddy, E. D. Natural Language Processing. In *Encyclopedia of Library and Information Science*, 2nd Ed. NY: Marcel Decker, Inc, 2001.

[11] Majumder, G. et al. Semantic Textual Similarity Methods, Tools, and Applications: A Survey. In *Computación y Sistemas*, Vol. 20, No. 4, s. 647–665, 2016.

[12] Mikolov, T. et al. Efficient Estimation of Word Representations in Vector Space. 2013.

[13] Mikolov, T. – Yih, W. – Zweig, G. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of NAACL-HLT 2013*, s. 746–751, Atlanta, Georgia, June 2013.

[14] Murphy, K. P. *Machine Learning A Probabilistic Perspective*. The MIT Press, 2012. ISBN 9780262306164.

[15] Shalev-Shwartz, S. – Ben-David, S. *Understanding Machine Learning From Theory to Algorithms*. Cambridge University Press, 2014. ISBN 978-1-107-05713-5.

[16] Smola, A. – Schoelkopf, B. A tutorial on support vector regression. Technical report, NeuroCOLT2, 1998. NC2-TR-1998-030.

A Uživatelská příručka

A.1 Přeložení programu

Přiložené DVD obsahuje dva programy, jsou to *STS* – program pro určení sémantické podobnosti vět a *lemmatizer* – program pro předzpracování vět. Překlad obou dvou programů probíhá pomocí nástroje *Maven*, stačí v adresáři projektu zadat příkaz: `mvn install`.

A.2 Spouštění a obsluha programu

A.2.1 Lemmatizer

Pro předzpracování vět je nutné spustit aplikaci `PreprocessingMain` v projektu *lemmatizer*. Tato aplikace se spouští s jedním parametrem. Tento parametr je cesta k souboru, který obsahuje dvojice vět, pro které chceme určit jejich sémantickou podobnost. Příklad spuštění aplikace: `mvn exec:java -Dexec.mainClass=cz.zcu.fav.liks.sentiment.PreprocessingMain -Dexec.args="data/semeval2014/STS.input.headlines.txt"`.

Pro správnou funkčnost aplikace je nutné, aby textový soubor s větami používal kódování *UTF-8*. Další podmínkou je správná struktura vět v souboru – vždy dvě věty k porovnání na jedné řádce oddělené tabulátorem.

Aplikace vytvoří tři nové soubory, které obsahují lemmata, *POS tagy* a *stemy*. Tyto soubory jsou dále potřeba pro vyhodnocení podobnosti vět.

A.2.2 STS

Pro vyhodnocení sémantické podobnosti českých vět jsem vytvořil aplikaci `CzechMain`. Tato aplikace se spouští buďto s jedním, se čtyřmi nebo s pěti argumenty. Tuto aplikaci lze spustit stejným příkazem, který je uvedený v předchozí kapitole A.2.1, pouze se musí změnit `mainClass=sts.CzechMain` a argumenty.

- Spuštění aplikace s jedním argumentem – tento argument představuje

cestu k *XML* souboru s předzpracovanými větami. Tento soubor automaticky vzniká při spuštění aplikace s více argumenty.

- Spuštění aplikace se čtyřmi argumenty – první z těchto argumentů je cesta k soubor s větami, jedná se o stejný soubor, který byl použitý při předzpracování vět. Další tři argumenty jsou cesty k souborům vzniklých při předzpracování, je nutné, aby byli v pořadí lemma, *POS tag* a *stem*.

- Spuštění aplikace s pěti argumenty – první čtyři argumenty jsou stejné, jako u spuštění aplikace se čtyřmi argumenty. Pátý argument je název *XML* souboru, který aplikace automaticky vytvoří.

Výstupem aplikace je soubor, který obsahuje vypočtené sémantické podobnosti vět. Podobnost dvou vět je vždy na stejném řádku, na kterém jsou v souboru s větami tyto dvě věty.