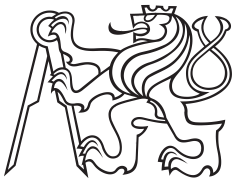
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Hodnocení reakcí čtenářů na obsah mediálních zpráv

Analýza sentimentu českých a anglických reakcí pomocí *Word Embeddings*

Lukáš Zíb

Vedoucí: Ing. Radek Mařík, CSc.
Obor: Softwarové inženýrství
Květen 2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Zíb** Jméno: **Lukáš** Osobní číslo: **468019**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Hodnocení reakcí čtenářů na obsah mediálních zpráv

Název bakalářské práce anglicky:

Evaluation of readers reactions to the content of media news

Pokyny pro vypracování:

Hlavní cíl: Vyzkoušet analýzu sentimentu nad českými texty.
1. Proveďte rešerši metod, které popisují řetězec zpracování při zpracování analýzy sentimentu textu, využívající vnoření textu do vektorového prostoru.
2. Vyberte vhodnou sestavu metod a vytvořte příslušný implementační řetězec zpracování.
3. Implementaci ověřte experimenty s anglickými texty.
4. Vhodně rozšířenou implementaci ověřte i výsledky nad českými mediálními texty.
5. Proveďte diskusi získaných výsledků a identifikujte kritické body zpracování.

Seznam doporučené literatury:

[1] Tomas Mikolov. Distributed Representations of Words and Phrasesand their Compositionality. 2013. https://papers.nips.cc/paper/5021-distributed-representations-of-words-andphrases-and-their-compositionality.pdf
[2] Tomas Mikolov & Kai Chen & Greg Corrado & Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. 2013. https://arxiv.org/pdf/1301.3781v3.pdf
[3] Ian Goodfellow & Yoshua Bengio & Aaron Courville. Deep Learning. 2016. https://www.deeplearningbook.org
[4] Maas, Andrew L. & Daly, Raymond E. & Pham, Peter T. & Huang, Dan & Ng, Andrew Y. & Potts, Christopher. Learning Word Vectors for Sentiment Analysis. 2011. http://www.aclweb.org/anthology/P11-1015

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Radek Mařík, CSc., katedra telekomunikační techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **10.02.2021** Termín odevzdání bakalářské práce: **21.05.2021**

Platnost zadání bakalářské práce: **30.09.2022**

Ing. Radek Mařík, CSc. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Poděkování

Především děkuji mému vedoucímu Ing. Radkovi Maříkovi, CSc. za trpělivost, rady v problematice a nasměrování při zhotovování implementace a náplně této práce.
Dále bych rád poděkoval za veškerou podporu a pochopení ze strany mé rodiny.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu a jiné zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 21. května 2021

Abstrakt

Náplní práce je implementace algoritmů strojového učení v jazyce Python, pomocí kterých je vyhodnocen postoj autora v libovolném textu. Vhodnými texty pro takové zpracování jsou různé recenze a reakce lidí, které obvykle vyjadřují kladný nebo záporný postoj vůči popisované záležitosti.
Cílem implementace a tedy celé práce je ověření existence vhodných prostředků pro vyhodnocení postoje z českých textů – reakcí na zprávy ohledně pandemie Covid-19. Práce obsahuje, mimo vyhodnocení těchto reakcí, také porovnání vyhodnocení veřejně dostupných recenzí v angličtině a češtině. Poukazuje tak na odlišnosti českého jazyka oproti anglickému.

Klíčová slova: Analýza sentimentu, Word Embedding, klasifikace reakcí, zpracování přirozeného jazyka, Scikit-learn, Gensim

Vedoucí: Ing. Radek Mařík, CSc.
Fakulta elektrotechnická,
Technická 2,
Praha 6 - Dejvice

Abstract

General idea of this work is to implement machine learning algorithms in Python, which are used to evaluate the author's attitude in any kind of text. Suitable texts for such evaluation are various types of reviews and reactions, which usually express a positive or negative sentiment towards the described issue.
The goal of the implementation and thus of the whole work is to verify the existence of suitable means to evaluate the sentiment of Czech texts – reactions to media news concerning the Covid-19 pandemic. This work contains, in addition to the evaluation of these reactions, also a comparison of the evaluation of public reviews in English and Czech. The work points at the differences between the Czech and English languages.

Keywords: Sentiment Analysis, Word Embedding, classification of reactions, natural language processing, Scikit-learn, Gensim

Title translation: Evaluation of readers reactions to the content of media news — Sentiment Analysis of Czech and English reactions while using Word Embeddings

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

Analýza sentimentu anglického textu není novinkou a tato záležitost se rozšířila i do dalších přirozených jazyků. Nyní se dostáváme k takovému postupu, který by byl schopen analyzovat i složitější jazyky, konkrétně češtinu.
Čeština má oproti angličtině mnohem více možností formování slov, a tak je obtížnější stroj naučit slova rozpoznávat a detekovat, která slova mají stejný význam a použití.
V tomto ohledu pomáhá vnoření slov do vektorového prostoru (*Word Embedding*[1]).

1.1 Popis práce

Tato práce ukazuje, zda lze metody *Word Embeddings* při zpracování angličtiny aplikovat úspěšně na český text.
Pro porovnání úspěšností zpracování těchto jazyků je využita právě analýza sentimentu, rozpoznávající kladné a záporné texty na základě vektorů slov.

Práce představuje implementaci analýzy sentimentu klasickými kroky [2] s využitím právě zmíněných metod *Word Embeddings*.

[1] *Word Embedding* – vektor reprezentující význam slova, zachovávající kontext použití [Kar18]

[2] Příprava textů, vytvoření *Word Embedding* modelu, trénování klasifikátoru a nakonec testování klasifikace s pomocí *Word Embedding* modelu – více v kapitole *3 Analýza sentimentu*

Část I

Rešerše

Tato část obsahuje informace o řešené problematice a implementaci analýzy sentimentu.
Nejprve je obsažen popis, co vlastně znamená sentiment, následovaný konkrétními informacemi o knihovnách použitých v implementaci a nakonec je popis samotné analýzy sentimentu.
Kapitoly jsou takto seřazené, neboť se postup analýzy sentimentu opírá o konkrétní metodu, využívající knihovny popisované v následující kapitole *2 Knihovny a možné implementace*.

1.2 Krátce o sentimentu

Sentiment, nebo-li postoj či mínění, je v této práci považován za vyjádření emocí autora vyhodnocovaného textu. Použité texty v této práci jsou založené právě na vyjádření emocí a postoje obsaženého v recenzích a komentářích. Tyto texty jsou obvykle subjektivní názory a měly by tak poskytnout snadno nalezitelné části, které vystihují pozitivní nebo negativní postoj – sentiment. Avšak „snadno nalezitelné“ jsou pro člověka, který hovoří přirozeným jazykem jako je angličtina nebo čeština. Pro stroj je porozumnění textu a správné nalezení postoje autora textu složitější. Problematika hledání sentimentu je více popsána v této části práce, převážně v kapitole *3 Analýza sentimentu*.

Kapitola 2

Knihovny a možné implementace

Problematika analýzy sentimentu staví na základních kamenech, které se používají ve strojovém učení. V této kapitole je přehled těchto základních kamenů, které jsou dále použity v Analýze Sentimentu.
Jedná se pouze o klíčové knihovny a jejich použití, úplný seznam použitých knihoven je na prvních řádcích zdrojových souborů.

2.1 Zpracování textu, grafy a další podpůrné knihovny

Před jakoukoliv prací s modelem na *Word Embedding* a klasifikací je nutné zpracovat a pročistit vstupní data. K načtení a zobrazení dat z csv souborů jsou použité knihovny *pandas* a *matplotlib.pyplot*.
Texty ze vstupních dat jsou běžné věty jako řetězce písmen, které je vždy nutné rozdělit na jednotlivá slova – tokenizace[1].

Po tokenizaci textů následuje jejich normalizace (převedení na malá písmena) spolu s odstraněním bezvýznamných slov. Následuje odstranění bezvýznamných tokenů [2].
Jako poslední transformaci textů je možné transformovat tokeny představující slova na jejich kořenovou formu. K tomu je využit *PorterStemmer* z knihovny *Gensim* pro transformaci anglických slov a *czech_stemmer* modul [Gom10] pro transformaci českých slov[3].

Před vytrénováním klasifikačního modelu je s pomocí *Scikit-learn* [PVG+20] knihovny *sklearn.model_selection* rozdělen dataset textů a sentimentů na dvě části – část pro trénování klasifikačního modelu (60 - 80 %) a testovací část pro ověření přesností jeho predikcí.

Na závěr je pro zobrazení výsledného zpracování *Word Embeddings* použita opět knihovna *matplotlib* spolu s konvertory *sklearn.decomposition.PCA* a *sklearn.manifold.TSNE*. Tyto konvertory postupně zredukují vícedimenzionálními vektory (reprezentující slova) projekcí na 2-D vektory, které lze ilustrovat v grafu.

[1] rozdělení řetězců znaků na části nazývané tokeny – řetězce znaků, které představují jednotlivá slova, odkazy, emotikony apod.[Dai19]

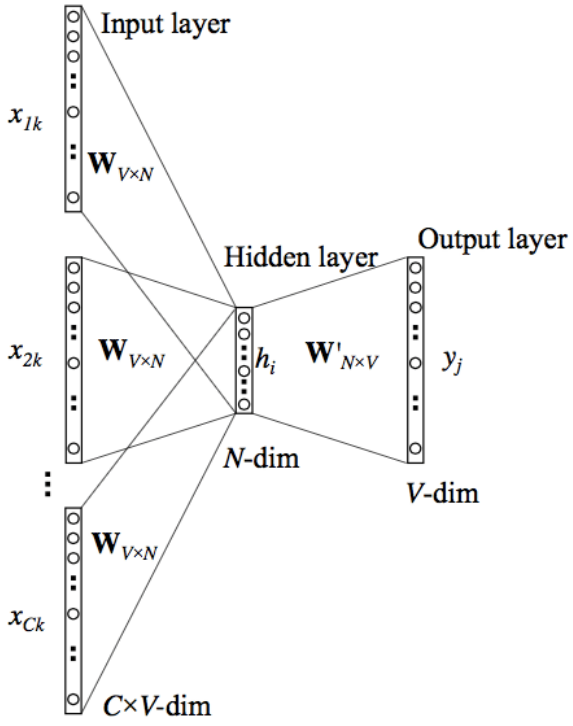
[2] pro převod slov z textů pomocí *Word2Vec* modelu z knihovny *Gensim*[Ře19] je toto odstranění tokenů spíše doporučené

[3] modul je veřejně dostupný skript v Pythonu od Luís Gomese, nachází se v nezměněné formě ve zdrojovém adresáři jako „src/czech_stemmer.py“

2.2 Word Embedding model

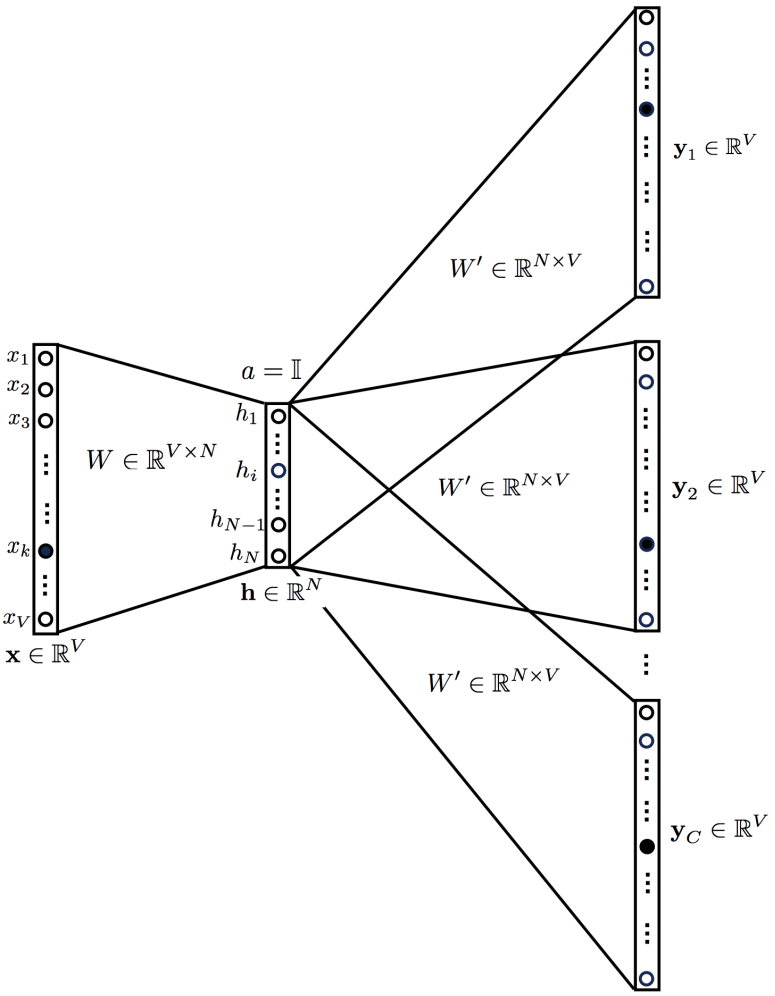
K zachycení povahy slov byl použit model Word2Vec od Gensim [Ře19], který představuje rozšíření původní implementace převodu slova na vektor od Tomáše Mikolova [Mik13, MCCD]. Word2Vec model využívá neuronové sítě k predikci vektoru hledaného slova a to dvěma možnými způsoby: *CBOW* a *skipgram*.

Jedná se v podstatě o převedení přirozeného jazyka na jazyk čísel, a to díky postupnému rozlišování slov pomocí neuronové sítě – na vstupu je pár slov jako posloupnost samostatných písmen a na výstupu je výsledný vektor reprezentující jejich povahu a použití v kontextu (např. *CBOW* model *Common Bag-of-Words* – viz. obrázek 2.1).



Obrázek 2.1: CBOW model neuronové sítě pro vytváření *Word Embeddings* – převzato z [Kar18]

Opakem *CBOW* modelu (nebo také algoritmu) je postup *skip-gram*, který na obrázku reprezentujících krajní vrstvy neuronové sítě vypadá zrcadlově převrácený – viz obrázek 2.2.

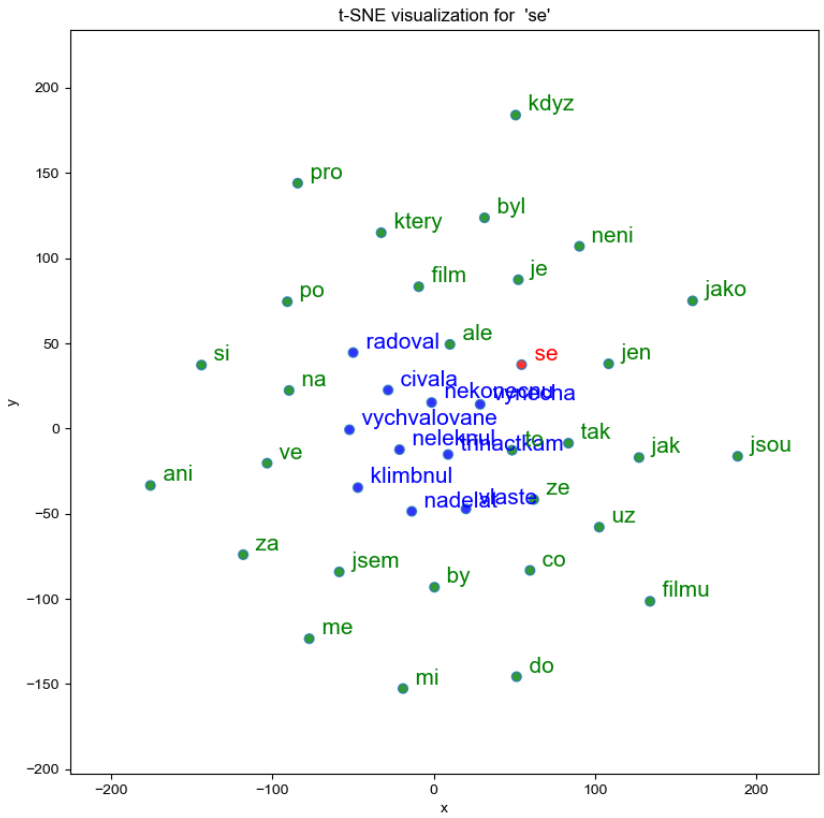


Obrázek 2.2: Skip-gram model neuronové sítě pro vytváření *Word Embeddings* – převzato z [Kar18]

Výsledkem je pak prostor, kde má každé slovo své souřadnice. Přičemž některá slova jsou si blíže právě kvůli použití v podobném kontextu – k vidění ve vizualizaci 300-D prostoru s vybranými slovy v obrázku 2.3

V momentě, kdy existuje *Word Embedding* reprezentace běžných slov, která ohodnotíme (pozitivní či negativní), můžeme vektory nových slov k těmto ohodnoceným slovům připodobnit.
Model *Word2Vec* v Gensim knihovně obsahuje instanci *WordEmbeddingsKeyedVectors* [Ře], která poskytuje jednoduché rozhraní pro porovnávání vektorů[5]. Tím lze určit sentiment všech slov a následně určit celkový sentiment věty či textu.

[5]metody, které vrací seznam slov a jejich *Embedding* reprezentace na základě vzdáleností vektorů k hledanému slovu či seznamu slov



Obrázek 2.3: Ukázka slov vnořených do 300-D prostoru vektorů – t-SNE[4] vizualizace *Word Embeddings*

Nebo naopak můžeme znát celkové sentimenty vět s podobným obsahem a podle nich určovat sentimenty pouze částí textů. Tyto částečné sentimenty pak skládáním dohromady opět vedou na celkový sentiment věty/textu, který ještě nebyl ohodnocen.
Více informací o určování sentimentu je v následující kapitole *2.3 Klasifikační modely*.

2.3 Klasifikační modely

Pro konečné vyhodnocení textu byly použity modely z knihovny Scikit-learn, které obsahují rozsáhlé množství klasifikátorů s podobným použitím [PVG+20].
Zejména trénování a testování modelů je řízeno zcela shodnými metodami „fit(array of shape (n_samples, n_features), list)“ a „predict(array)“.

V následujících podkapitolách jsou shrnutí použitých klasifikátorů. Jedná se pouze o výběr z mnoha modelů knihovny Scikit-learn[PVG+20]

2.3.1 Decision Tree

Klasifikátor *Decision Tree* představuje metodu učení, která na základě pouze vstupních hodnot rozhoduje o jejich kategorii.
Kategorie vstupních dat mohou být různého počtu, a lze tak na základě vstupu určovat nejen sentiment (pozitivní či negativní), ale také tématiku nebo zmínky míst, osob apod..
Složitost trénování i použití je logaritmická v závislosti na počtu vstupních dat [PVG+20]:

$$\begin{aligned} \text{trénování} \dots O(n_{celkem\,tokenu} \cdot \log(n_{pocet\,textu})) \ , \\ \text{použití} \dots O(\log(n_{pocet\,textu})) \qquad\qquad . \end{aligned}$$

Díky tomu je trénování poměrně rychlé a především je rychlé následné použití.

Hlavní nevýhodu představuje náchylnost k vyhodnocení podobných vstupů odlišně, jelikož se model rozhoduje na základě jednoduchého číselného porovnávání.
Vychýlení v jednotkách složek vstupních vektorů může mít za následek zcela opačný výstup, nebo také nemusí.

Klasifikátor *DecisionTreeClassifier* představuje jednoduchou pomůcku s poměrně nejistými výsledky, téměř bez ohledu na kvalitu *Word Embedding* modelu, vytvářející vstupní vektory.

Klasifikátor v principu vytváří binární strom od kořenového uzlu a přidává větve podle trénovacích dat tak, aby v listech byly data s přibližně shodnými „vlastnostmi“, mající stejné ohodnocení.
Při zařazení nového vstupu se postupuje větvemi od kořene stromu, dokud vstup nemá shodu s vlastnostmi a s ohodnocením. Pokud toto procházení stromu dojde do listu, který má oproti vstupu odlišné vlastnosti nebo ohodnocení, dochází k větvení tohoto listu, vzniká nový uzel a nové listy.

Založení *DecisionTreeClassifier* objektu je řízeno především parametry [PVG+20]:

- *criterion* = „gini“ nebo „entropy“ – funkce pro měření kvality dělení v uzlu
- *splitter* = „best“ nebo „random“ – strategie pro větvení – určení porovnávacího parametru
- *max_depth* – celé číslo pro omezení hloubky stromu, tímto parametrem lze „omezit“ čas při vyhledávání ohodnocení
- *min_samples_splitint* – minimální počet vstupních dat, která musí existovat v uzlu, aby došlo k rozvětvení

Použití stromu a klasifikátoru je v podstatě série porovnávání (větší, menší, shodný) dokud se klasifikátor nedostane na konec stromu. Vytvoření takového stromu a porovnávání v uzlech je upraveno především zmíněnými parametry, ale existuje řada dalších parametrů, kterými lze klasifikátor nastavit. Toto nastavení má následně dopad především na rychlost vyhledávání a přesnost klasifikace.

2.3.2 SGD – Stochastic Gradient Descent

Algoritmus sestupu v poli podle gradientu (*Gradient Descent algorithm*) je součástí mnoha algoritmů pro strojové učení, který optimalizuje funkci $f(x)$ pomocí změny parametru, aby bylo dosaženo minima/maxima funkce $f(x)$ [GBC16]. Díky tomu je nalezen ideální parametr x pro následné „trasování“ datových bodů.
Stochastická varianta sestupu podle gradientu je oproti klasické variantě výrazně rychlejší a efektivnější díky výběru malého množství bodů, podle kterých se gradient počítá [Sri19].
Jde tak o hrubší aproximaci gradientů v datovém poli, která nedává nejlepší výsledek, ale díky tomu trvá algoritmus zlomek času. Složitost SGD je linární podle velikosti dat[GBC16].

Zatímco SGD je obecná optimalizační procedura, klasifikace pomocí *SGDClassifier* tuto proceduru využívá pouze při některém z nastavení. Tato nastavitelnost se řídí parametrem „loss“, který je prvotně („defaultně“) nastaven na „hinge“ [6]. Takové nastavení vytváří lineární *Support Vector Machine* (SVM) [7].
Obvykle klasifikátor vytváří rozhodovací rovinu vektorů, na níž se provádí projekce samotných vektorů [PVG+20]. Posléze, podle rozmístění vektorů a jejich ohodnocení se modeluje přímka, která vektory rozdělí podle ohodnocení.
Výhodou této metody je rychlost trénování i klasifikace. Na druhou stranu je obtížněji nastavitelný, aby dosahoval ideálních výsledků [PVG+20].

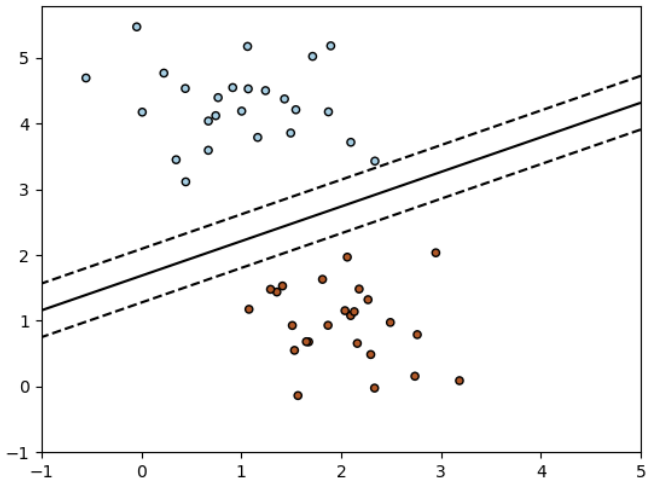
Jedná se o známý nástroj, který má díky své efektivnosti velké využití při zpracování velkých dat. Složitost trénování je lineární podle počtu vstupních dat:

$$O(n_{epochs} n_{celkem\,tokenu} \overline{p}) \ ,$$
kde $\overline{p}$ je průměr nenulových hodnot na vstupu.

V principu jde o rozdělení vektorů do dvou skupin podle jejich příbuznosti. Na začátku tedy existuje dataset souřadnic, ve kterém jsou body se shodným ohodnocením jistým způsobem shlukované (až na výjimky). Úkolem *SGDClassifier* je tyto shluky nalézt a odlišit tak, aby většina dat se shodným ohodnocením byla na jedné straně rozdělení, zatímco data s odlišným ohodnocení byla na straně druhé – viz ukázka v obrázku 2.4.

[6] v implementaci je použita hodnota *loss*=„squared_hinge“
[7] SVM je obecným postupem pro odlišení vektorů v prostoru na dvě či více částí [PVG+20]

Klasifikátor tohoto rozdělení dosahuje metodou SVM, hledající v prostoru vektorů takovou „šedou zónu“, která množinu vektorů co možná nejlépe rozdělí na dvě části, ve kterých by se vyskytovaly vektory pouze shodného ohodnocení (až na výjimky).
Při hledání této zóny se nejprve odhaduje rozdělení, které by vektory úspěšně odlišilo. Následně je toto rodzělení vyhodnoceno a upraveno tak, aby byla „šedá zóna“ co možná největší. K tomuto účelu se vyhledává množina vektorů ze vstupní množiny, označované jako *Support Vectors*, které definují hranice „šedé zóny“. [Pup18]
Rozdělení s „šedou zónou“ je vidět v názorné ukázce 2-D prostoru vektorů v obrázku 2.4.



Obrázek 2.4: Ukázka rozdělení roviny podle pomocí SGD klasifikátoru

Založení *SGDClassifier* objektu je řízeno především parametry [PVG+20]:

- *loss* = „hinge“, „squared_hinge“, „perceptron“, ... – funkce pro výpočet „ztrátovosti“ modelu
- *penalty* = „l2“, „l1“ nebo „elasticnet“ – definuje způsob regularizace modelu [8]
- *alpha* – konstantní číslo, kterým se násobí dopad regularizace
- *max_iter* – číslo definující horní limit pro počet trénovacích epoch

[8] úprava komplexnosti modelu s cílem zvýšit jeho přesnost

2.3.3 Neuronová síť – MLP klasifikátor

Princip neuronové sítě je mít několik desítek až stovek uzlů, ve kterých se provádí jednoduché početní operace. Na vstupu je seznam čísel z předchozí vrstvy, které slouží jako parametry funkce, vracející jedno číslo, které se následně distribuje do všech uzlů další vrstvy. [Tutxx]
První vrstvu sítě tvoří vždy vstupní data (obvykle seznam několika čísel) a na výstupu je číslo nebo specifikovaný počet čísel, která indikují ohodnocení. Například pro tři typy sentimentů (pozitivní, neutrální a negativní) by na výstupu byly 3 čísla, každé s hodnotou od 0 do 1. Tyto hodnoty znamenají pravděpodobnosti, že jsou vstupní data pozitivní, neutrální nebo negativní[9].
Základem vývoje neuronové sítě je obvykle „zpětná propagace argumentů“[10], která mění parametry vnitřních funkcí každého uzlu. Tyto parametry funkce se označují jako váhy vstupních parametrů (každý spoj mezi dvěma uzly má svojí váhu) a jde o koeficienty pro násobení konkrétních vstupů. [Tutxx]

Použití neuronové sítě ke klasifikaci je výpočetně náročnější než předchozí zmíněné klasifikátory *DecisionTree Classifier* a *SGD Classifier*.
Při dostatečném objemu trénovacích vektorů a výstupů by však měl poskytnout přesnější predikce. Navíc může být klasifikátor použit nejen k určování sentimentu, ale rozdělení do více kategorií – podobně jako *2.3.1 Decision Tree*.

Název třídy modelu je *MLPClassifier*[11]. Jedná se o klasické využití neuronové sítě pro nalezení zobrazení $f : R^m \rightarrow R^n$, kde m je rozměr vstupního vektoru *Word Embedding* a n je rozměr výstupního vektoru, který činí 2 nebo 3[12].
Toto zobrazení se modeluje zmíněnou „zpětnou propagací argumentů“. Ve výsledku pak mají některé uzly větší význam na vyhodnocení oproti jiným.

Další nevýhodou tohoto typu klasifikátoru je složitější nastavení. Oproti *DecisionTree* a SGD klasifikátoru je složitější najít ideální nastavení [PVG+20]. O jeho časové náročnosti svědčí i složitost trénování:

$$O(n_{celkem\,tokenu} \cdot h^k \cdot o \cdot i) \ ,$$

kde h je počet neuronů ve skryté vrstvě, k značí počet skrytých vrstev,

[9] pro 3 sentimenty jsou 3 pravděpodobnosti, každá říkající jak moc je pravděpodobné, že je vstup právě daný sentiment
[10] z anglického překladu *backpropagation*
[11] Multi-layer Perceptron [PVG+20]
[12] podle toho, zda zahrnujeme neutrální sentiment, vždy ale dostáváme alespoň pravděpodobnost pozitivního a negativního sentimentu

o je počet výstupních neuronů a i je počet iterací (epoch).

Tento typ klasifikace vycházel časově náročnější během testování vzniklé implementace (oproti *DecisionTree* a SGD klasifikátorům). Implementace je však založena na používání CPU (bez počítání na grafické kartě). Kdyby byla použita grafická karta, tak by zpracování dat vhodně implementovanou neuronovou sítí bylo podstatně rychlejší [13].

Založení *MLPClassifier* objektu je řízeno především parametry [PVG+20]:

- *hidden_layer_sizes* – počet neuronů ve skrytých vrstvách neuronové sítě
- *solver* = „lbfgs“, „sgd“ nebo „adam“ – funkce pro určení váhy na výstupu každého uzlu
- *max_iter* – číslo definující horní limit pro počet trénovacích epoch

[13] faktor zrychlení se odvíjí od výkonu konkrétní karty a použité implementace klasifikátoru

Kapitola 3

Analýza sentimentu

Analýza Sentimentu textu znamená proces určení, zda zkoumaný text vyjadřuje celkově pozitivní, negativní nebo neutrální postoj [Sel20].
Toho lze obvykle dosáhnout statistickým měřením ohodnocených slov v daném textu – při převaze pozitivních výrazů je následně i celý text vyhodnocen pozitivně a naopak. Existují ale i jiné přístupy, které již hraničí se strojovým porozumněním významu vět.

3.1 Shrnutí problematiky

Určení sentimentu textů v přirozeném jazyce (angličtina, čeština, ...) pomocí počítače s sebou nese mnoho problémů.
Největším problémem je naučit počítač rozpoznat smysl vět a slov. Toho lze dosáhnout různými způsoby, ale vždy pomůže zjednodušení jednotlivých slov a vět. Zjednodušením textů sice ztrácíme jistou míru informací a riskujeme úplné obrácení významu vět, ale počítač je schopen slova snáze rozpoznat.

Základní přístup k textu je zastoupení slov čísly – indexace. Tím dosáhneme možnosti vyhledávání informací o jednotlivých slovech, která ale musí být zpracována, aby stroj odhalil příbuznosti (jiný pád, čas slovesa, stupňování přídavných jmen, ...).
Díky vytvoření takového korpusu je možné vyhodnotit sentiment podle známých slov ve zkoumaném textu. Bohužel s rostoucím počtem slov v korpusu roste lineárně i časová náročnost, avšak bez dostatečného množství „známých“

slov mizí šance na úspěšné vyhodnocení textu.

Pro ještě snazší rozpoznání slov existují metody k reprezentování slov pomocí vektorů (tzv. *Word Embedding*), které nesou povahu a příbuznost jednotlivých slov. Tato reprezentace poskytuje náhled na příbuznost slov a každé slovo lze tak vyhodnotit jako spíše pozitivní nebo negativní [Nab18]. Více o metodě převedení slova na vektor je k nalezení v kapitole *2.2 Word Embedding model*.

3.2 Metody analýzy

Hlavní rozdělení, jak analyzovat text a vyhodnotit jeho výsledný sentiment, je následující (převzato z [WM14]):

- Přístup založený na lexikonu slov
  - Použití slovníku ohodnocených slov
  - Vytvoření korpusu slov z textu
- Strojové učení s učitelem – *Supervised Machine Learning*
  - Klasifikace podle rozhodovacího stromu – viz *2.3.1 Decision Tree*
  - Lineární klasifikace (včetně použití neuronové sítě)
  - Klasifikace založená na pravidlech
  - Pravděpodobnostní klasifikace
- Strojové učení bez učitele – *Unsupervised Machine Learning*
- Hybridní přístup – kombinace strojového učení s použitím lexikonu

Pro tuto práci byla použita metoda založená na strojovém učení s učitelem a předně klasifikace pomocí neuronové sítě (posané v *2.3.3 Neuronová síť – MLP klasifikátor*). Pro porovnání úspěšnosti analýzy sentimentu, a tedy přesnosti klasifikace, byly použity další 2 přístupy strojového učení s učitelem: *Decision Tree* a lineární rozdělení vektorem – viz kapitola *2.3 Klasifikační modely* popisující vybrané 3 klasifikační metody.

3.3 Postup analýzy sentimentu v implementaci

Klíčové kroky v analýze sentimentu textu jsou vždy příprava textu, trénování klasifikačního modelu a následné otestování a použití. Pro účely této práce je před trénováním klasifikačního modelu tvorba *Word Embedding* modelu a tudíž i zkrácená příprava textu.
Níže jsou proto uvedené pouze důležité kroky při použití *Word Embedding* modelu a i tak se může postup lišit při použití odlišných knihoven.

1. Získání a předpřipravení datasetu s texty
2. Příprava textů
   a. Rozdělení textů do seznamů slov
   b. Pročistění seznamů a úprava slov (*doporučené*)
      (i) Odstranění nesmyslných znaků a převedení všech písmen na malá bez diakritiky
      (ii) Odstranění *stop-words*[1]
      (iii) Odstranění předpon a přípon slov [2]
3. Příprava a vytrénování *Word Embedding* modelu
   a. Trénování modelu na části/celém datasetu
   b. Zefektivnění modelu – použití *KeyedVectors*[3]
4. Příprava a vytrénování klasifikačního modelu
   a. Trénování modelu na části datasetu – texty připravené pro trénování
5. Testování klasifikačního modelu
   a. Předpověď sentimentů a vyhodnocení úspěšnosti (pomocí *confusion matrix*) připravených textů pro testování
6. Použití modelů na reálné texty – bez ohodnocení, ale připravené shodně jako texty datasetu

[1] slova, která nepřidávají/nemění význam věty. V angličtině se jedná např o: *the*, *at*, *which* – převzato z [Tej20]
[2] v angličtině nazýváno *stemming* a *lemmatization* – techniky zpracování přirozeného jazyka, které převádí slova do základní formy [Jab18]
[3] zachování pouze tokenů a příslušných vektorů, které pokrývají „celý“ vektorový prostor [Ře19]

3.4 Problematika s češtinou

Problémy s českým jazykem oproti anglickému vychází z větší variability jazyka (více pádů, synonym, ...) a obtížné rozpoznatelnosti předpon a přípon slova. Další problematikou je stavba vět, která je oproti angličtině benevolentnější.
Navíc i obyčejné věty v sobě mohou skrývat více negací, které při špatném zpracování textu mohou změnit celý význam věty.

To vše se projevuje na přípravě textu při pročištění seznamů slov na:

1. převedení slov s odlišnou diakritikou na zcela shodná
   většinou se jedná o podstatná jména mající odlišné významy. To vede k méně přesným aproximacím slov na vektory, jelikož se shodně vypadající slova používají ve zcela odlišných kontextech, a tudíž by měly vést na 2 odlišné vektory.
2. správnou detekci *stop-words*
   – spojky, částice, některá zájmena a další odstranitelná slova, která „pouze“ spojují části věty, ale nemají informační hodnotu
3. odstranění předpony a přípony slov
   to je jistě přinejmenším problematické a možná vyjde ve výsledku tato redukce více matoucí, než ponechání slov v původním tvaru

Kvůli problematičtější přípravě textu je také obtížnější správně vytrénovat *Word Embedding* model – některá příbuzná slova mohou mít velmi odlišné vektory a naopak.
Problémy se samotnou klasifikací by nastat neměly, ale kvůli méně přesně určeným vektorům jsou některá slova matoucí při finálním určení sentimentu – místo výrazného určení sentimentu nevýrazné či naopak.

Část II

Implementace a vyhodnocení

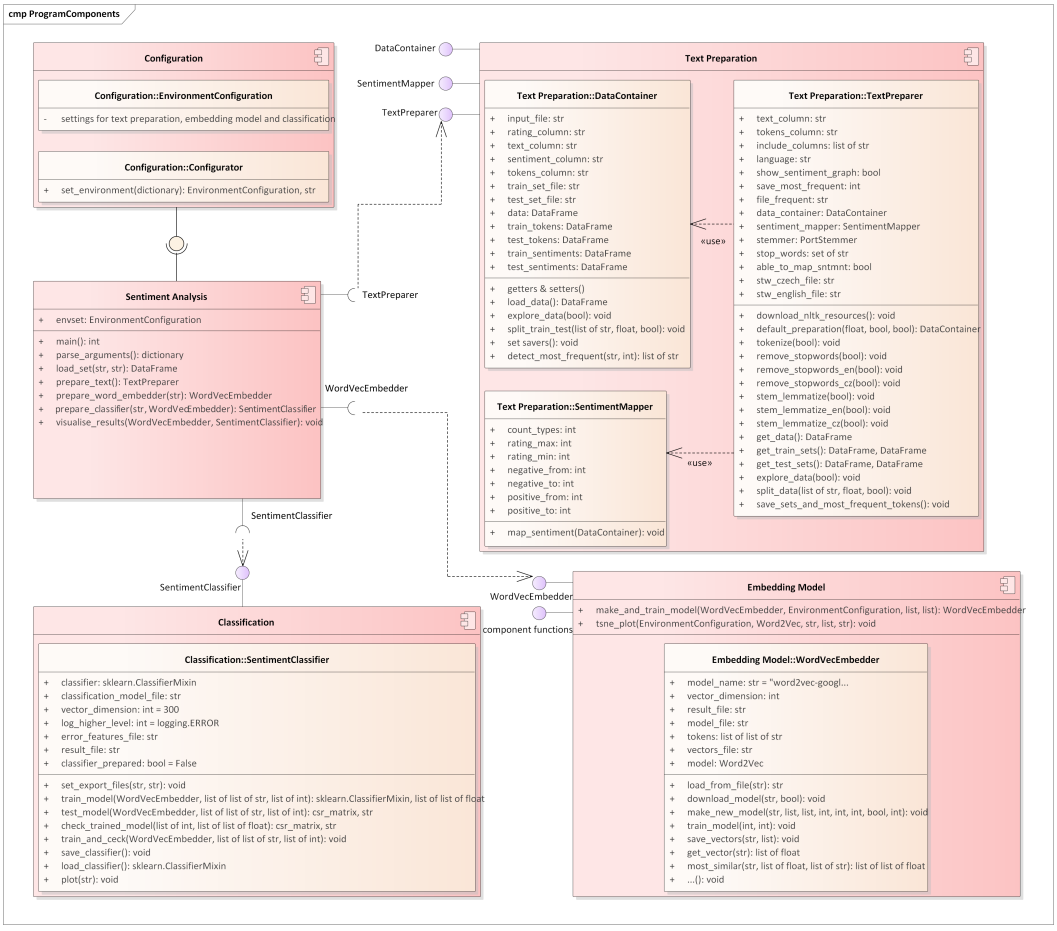
Kapitola 4

Plán Implementace

Hlavním cílem je implementovat v Pythonu analýzu sentimentu dostatečně variabilní, aby jí šlo bez velkých odlišností použít pro anglické i české texty. Následně implementaci odzkoušet s různým nastavením na obou jazycích a výstupy porovnat.
Na konec provést diskuzi výsledného porovnání jazyků a hypotetizovat nad možným zlepšením zpracování českého textu.

4.1 Realizace

Implementace je rozdělena do jednoho řídícího skriptu a třech hlavních skriptů pro zpracování textů, trénování a použití *Word Embedding* modelu a trénování a použití klasifikačního modelu. Veškeré chování programu je řízeno konfiguračním objektem, který načítá své proměnné ze vstupního JSON souboru.



Obrázek 4.1: Diagram hlavních komponent programu – kvůli nečitelnosti je diagram přiložen v přílohách u práce

V obrázku 4.1 je původní struktura programu, která byla rozšířena o další třídu podobnou *SentimentMapper* v komponentě pro přípravu textu.
Kvůli pozdnímu dodání českých reakcí na zprávy, které přibližně měsíc před koncem práce nebyly očekávány, bohužel nezbyl čas na úpravu podpůrných materiálů.
Původní použití však bylo zachováno, pouze doplněno kódem speciálně na získané reakce.

Hierarchie adresáře s daty a programem je v příloze *C Elektronické přílohy*. Program je spustitelný ve verzi Pythonu 3.7 a vyšší. Na nižší verzi nebyl kód testován.

4.2 Činnosti

Nejprve proběhlo seznámení s problematikou analýzy sentimentu, převodu textu na vektor a rešerše k použití knihoven v implementaci.
Následovala implementace programu s využitím základů z některých tutoriálů [Meg18, Baa20] a hledání použitelných textů v anglickém a českém jazyce. Po odzkoušení funkčnosti programu byly nasazeny nalezené texty s různým nastavením zpracování textů, trénování *Word Embedding* modelu a použití klasifikačních modelů. Ty nejúspěšnější výsledky byly následně podrobeny detailnějším nastavením a náročnějšímu trénování převodu slov na vektor. Nakonec bylo otestováno několik zpracování se stejným nastavením pro ověření úspěšnosti daného nastavení.
To vše doprovázelo psaní tohoto dokumentu a komentování kódu.

Kromě rešerše, která z větší části proběhla před plánováním implementace a následujícím zaznamenáním výsledků, práce obnášela následující činnosti.

1. Ujasnění knihoven, postupu a problematiky 35h
2. Implementace – pro zpracování AJ a ČJ 75h
3. Popis implementací a dokumentace tříd kódu 30h
4. Diskuze,vyhodnocení a ladění textu práce 20h

Veškeré časy jsou souhrnné orientace z plánu. Celkem bylo odhadováno přibližně 20 *man-days*[1] na výše zmíněné činnosti (bez konzultací s vedoucím a výpočetního času). Podrobný časový plán je k nalezení v příloze *D Plán implementace*.

Reálné trvání implementace a dalších činností bylo **145** hodin.
Toto číslo je nižší především ze dvou důvodů. Ujasnění knihoven a postupu proběhlo částečně před trasováním stráveného času. Implementace se zkrátila o jednotky hodin díky nalezené implementaci českého *stemmeru* pro získání

[1] v češtině „člověko-hodina“ znamená množství času, které 1 člověk vypracuje, v násobkách 8 hodin (obvyklá pracovní doba jednoho dne)

kořenů slov.

Nebýt netrasované části a usnadnění implementace, tak by reálné trvání bylo o cca 15 hodin delší. Jde pouze o odhad, ale znamenalo by to odpracování **20 *man-days*** z původně plánovaných 20 *man-days*.

Kapitola 5

Otestované metody

Shrnutí úspěšností s použitím odlišných klasifikátorů, různých postupů při zpracování textu a především odlišné parametrizace *Word2Vec* modelu. Veškeré obměny přitom byly zkoumány jak na anglickém, tak českém textu – bylo použito přibližně shodné množství recenzí [1]. Použité anglické recenze byly v průměru delší, a tudíž vhodnější na trénování modelů oproti českým.

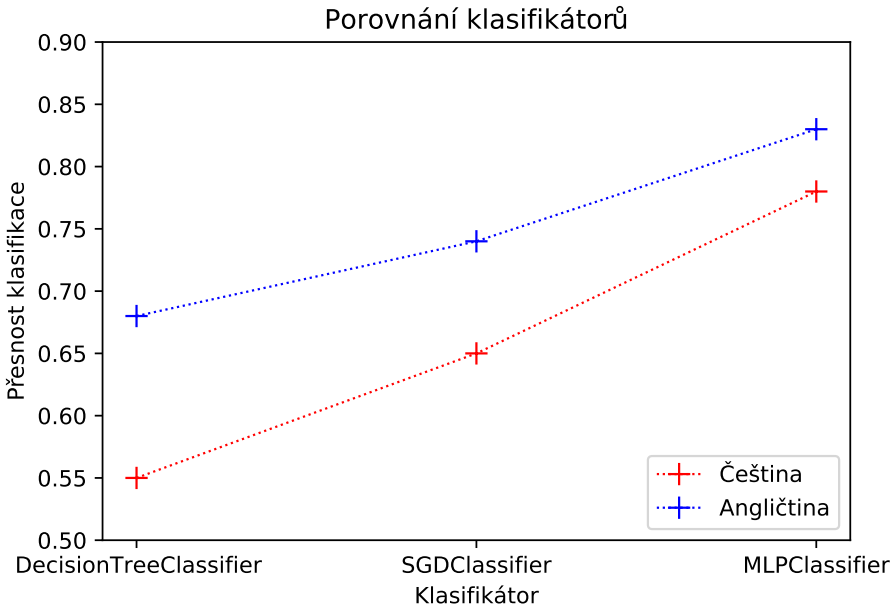
Ke konci práce byly získány ohodnocené reakce na české zprávy, které tvoří druhou část výsledků. Tato data jsou však mnhem menším datasetem, než data, použitá pro porovnání jazyků.

Použitá data byla získána ze 3 různých zdrojů, které jsou popsány v kapitole *A.2 Vstupní texty a jiná data.*

[1] anglických recenzí bylo použito prvních 91 368, stejně jako českých recenzí, které byly použity všechny

5.1 Klasifikátor

Z vyzkoušených klasifikátorů byl výrazně nejúspěšnější *MLPClassifier* využívající neuronovou síť a za ním *SGDClassifier*. Mezi *DecisionTree* klasifikátorem a klasifikace podle Stochastického gradientu nejsou v angličtině tak velké rozdíly a je možné, že za určitého nastavení je úspěšnější *DecisionTree Classifier*.

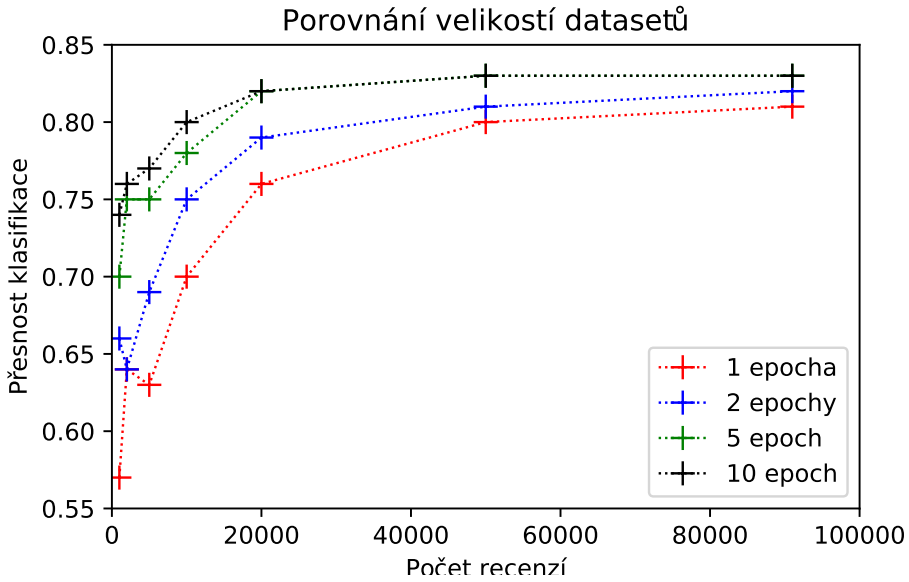


Obrázek 5.1: Porovnání klasifikátorů nad angličtinou a češtinou

Toto porovnání bylo měřeno pouze se změnou jazyku – bez změny v přípravě textů a trénování Word2Vec modelu.
Většina testů probíhala dále s pomocí neuronové sítě v podobě úspěšného *MLPClassifier*, který byl zároveň nejpomalejší na trénování.

5.2 Word Embedding model

Věškeré pokusy byly prováděny pouze s modelem Word2Vec z knihovny *Gensim* s různým nastavením parametrů. Byly vyzkoušené různé počty trénovacích epoch, minimální frekvence slov, velikosti vektorů a jiné. Většina těchto změn měla pouze nepatrné následky za použití shodné velikosti dat. Největší rozdíl tvořila právě data, která při malém množství (stovky záznamů) nedokázala vycvičit model na příliš velkou diverzitu vektorů a celkovou úspěšnost analýzy. Při těchto malých dávkách dat se projevila změna v počtu epoch poměrně výhodnější.
Jak je z grafu *5.2 Závislost velikosti datasetů* vidět, menší velikosti datasetů (do 5000 recenzí) mají větší náklonnost k pochybení.
Z tohoto důvodu, by menší datasety měly být testované vícekrát a porovnávat buď průměrnou nebo maximální přesnost, aby tato porovnání dávala smysl.



Obrázek 5.2: Závislost úspěšnosti na velikosti datasetů

Lze usoudit, že model na vytrénování potřebuje jisté množství zpracování záznamů a když je záznamů méně (do jednotek tisíc) je dobré modely trénovat hlouběji – oproti velkému množství záznamů, kde úspěšnost trénování Word2Vec modelu pouze jednou vychází pouze o jednotky procent horší.
Z grafu na obrázku 5.2 si lze povšimnout přiblížení vzrůstu úspěšnosti k logaritmické křivce – to platí především pro větší počty trénovacích epoch *Word2VEc* modelu.

5.3 Příprava textů

Vyzkoušené odlišnosti přípravy textů byly pouze v celkové redukci slov na jejich kořeny a odstranění „bezvýznamných“ slov – *stop-words*.

Příprava textu	Čeština	Angličtina
Odstranění stop-words a převedení na kořeny	78% - 79%	83%
Pouze odstranění stop-words	78%	83%
Pouze převedení slov na kořeny	78%	83%
Ponechány stop-words a původní tvary slov	78%	83%

Polovina testů pro češtinu s odstraněním *stop-words* a převedení na kořeny vycházela 78% a druhá polovina vycházela 79%.

Tabulka 5.1: Přesnosti klasifikace při odlišné přípravě textu

Jelikož pro klasifikaci vět a celých recenzí byly použity vektory reprezentující slova, změny v přípravě textů se výrazně neprojevovaly – jak lze vidět v tabulce 5.1. Záleží opět na velikosti datasetu, který byl v ukázaných porovnání vždy shodný.

Kapitola 6

Výsledky

Porovnávané výsledky jsou vždy zaměřeny pouze na jednu ze tří problematik – příprava textů, *Word Embedding* model a klasifikátor.
Základ konfigurace je ukázán v tabulkách 6.1, 6.2 a 6.3. Od tohoto základu se konfigurace vždy lišily pouze v jedné z těchto tabulek, ostatní nastavení konfigurace byla ponechána jak pro české, tak anglické texty.

Parametr	Výchozí hodnota
remove_stopwords	True
stem_words	True

Tabulka 6.1: Konfigurace přípravy textu

Parametr	Výchozí hodnota	Další hodnoty
vector_dimension	300	30, 50, 100, 200, 500
train_epochs	30	1, 2, 5, 10
minimal_word_frequency	1	0, 5
use_skipgram	True	False

Tabulka 6.2: Konfigurace trénování Word2Vec modelu

Parametr	Výchozí hodnota	Další hodnoty
model_class_name	MLPClassifier	DecisionTreeClassifier, SGDClassifier

Tabulka 6.3: Konfigurace klasifikace – pouze změna klasifikátoru

6.1 Anglické texty

Anglické texty z yelp.co[Yel20] nebyly zcela využity – kvůli malému množství českých dat, se kterými probíhala porovnání. Avšak i na zlomku celkového objemu získaných dat byla analýza sentimentu poměrně úspěšná při použití *MLP klasifikátoru* [1].
Ostatní klasifikátory vykazovaly při použitém množství dat o poznání nižší úspěšnost (řádově desítky procent). U *SGD* a *DecisionTree* klasifikátoru by mohl větší dataset znamenat znatelnější zvýšení přesnosti klasifikace oproti již úspěšnému *MLP klasifikátoru*. Porovnání klasifikátorů je v tabulce níže 6.4.

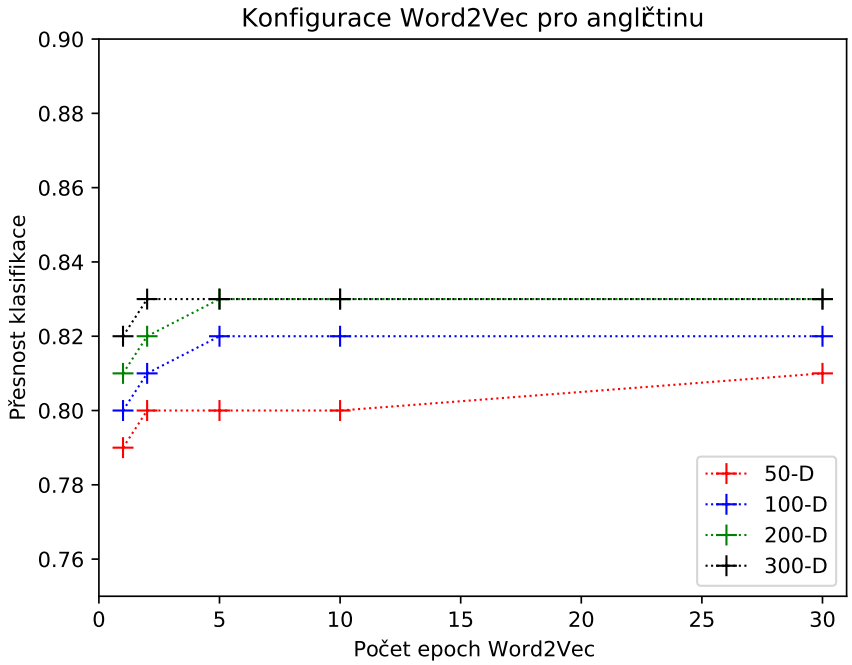
Klasifikátor	Přesnost
DecisionTree	68%
SGD	74%
MLP	83%

Tabulka 6.4: Porovnání přesností klasifikace angličtiny na jednotlivých klasifikátorech

Nejvyšší úspěšnosti klasifikace 83% bylo dosaženo při největším použitém počtu epoch na *Word Embedding* modelu, a také více rozměrů vektorů reprezentujících jednotlivá slova. Další změny jako ponechání *stop-words* nebo nastavení minimální četnosti slova pro trénování *Word2Vec* modelu neměly tak znatelný efekt.
Podivuhodný je však rozdíl při použití algoritmů trénování *Word2Vec* modelu. Úspěšnost *CBOW* typu *Word2Vec* modelu obecně zaostává za modelem typu *skip-gram*. Odlišnost těchto typů spočívá v použití kontextu odhadovaného slova – viz kapitola *2.2 Word Embedding model*. Lze tedy říci, že je vhodnější odhadovat vektor slova i s kontextem pouze z hledného slova, než odhadovat pouze vektor slova z kontextu a hledného slova.

V grafu níže je porovnání algoritmů na trénování *Word2Vec* modelu a některých dimenzí vektorů v závislosti na počtu epoch trénování.

[1] klasifikátor využívající neuronovou síť – více v rešerši *2.3.3 Neuronová síť – MLP klasifikátor*



Obrázek 6.1: Úspěšnost zpracování angličtiny v závislosti na počtu epoch pro různé dimenze

6.2 České texty

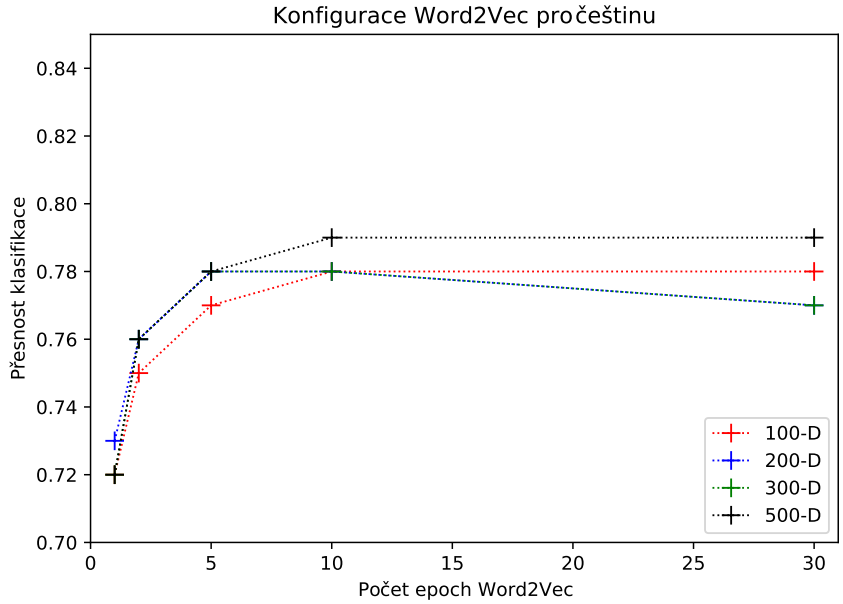
Stejně jako u anglických textů je znatelně vyšší úspěšnost klasifikace pomocí neuronové sítě v podobě *MLP klasifikátoru* – viz tabulka 6.5. Platí také vyšší úspěšnosti při použití vícerozměrných vektorů, trénovacích epoch *Word2Vec* modelu a především použití *skip-gram* – viz graf v obrázku 6.2 níže.

Klasifikátor	Přesnost
DecisionTree	55%
SGD	65%
MLP	78%

Tabulka 6.5: Porovnání přesností klasifikace češtiny na jednotlivých klasifikátorech

Tato porovnání byla prováděna nad recenzemi z portálu ČSFD.cz[2].

[2]více info o zdrojích textů v kapitole *A.2 Vstupní texty a jiná data*



Obrázek 6.2: Úspěšnost zpracování češtiny v závislosti na počtu epoch pro různé dimenze

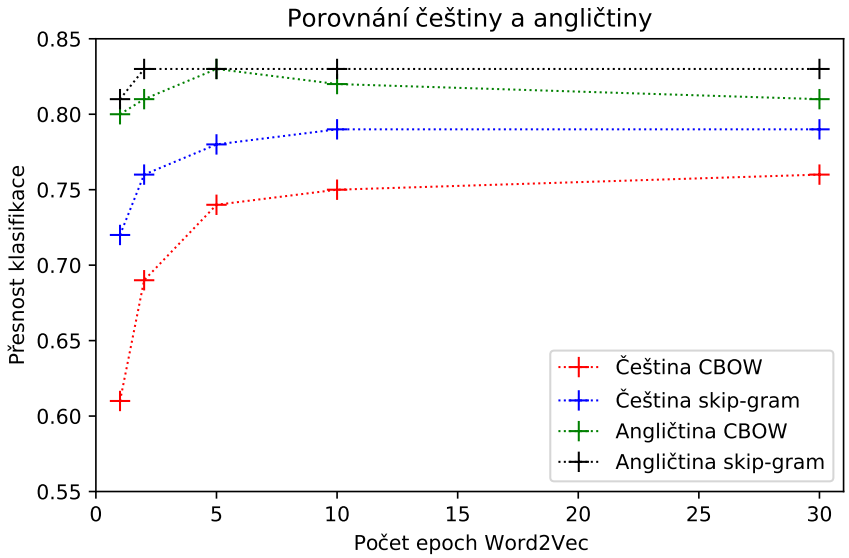
6.3 Porovnání zpracování jazyků

Z porovnání přesností klasifikace závislé na počtu epoch trénování *Word Embedding* modelu (viz graf v *6.3 Porovnání češtiny a angličtiny*) a z porovnání přesnosti při stejné konfiguraci (tabulka 6.6) lze vidět, že anglické texty jsou lépe vyhodnocené, ale s rozdílem pouze okolo 5%.

Typ Word2VEc modelu	Přesnost angličtiny	Přesnost češtiny
CBOW	81%	76%
skip-gram	83%	79%

Tabulka 6.6: Porovnání nejlepších výsledků pro angličtinu a češtinu s CBOW nebo skip-gram modelem Word2Vec

To, že oba jazyky lze vyhodnotit s přibližně 80% přesností klasifikace, je jistý úspěch. Je však jisté, že by šly metody ještě zlepšit, aby se vyhodnocení českého textu ještě více přiblížilo anglickému. Nejvíce jistě pomůže objemnější dataset – o tom přímo vypovídá fakt, že čeština obsahuje asi 250 tisíc slov[Bra20], zatímco angličtina obsahuje přes 170 tisíc [Dex]. Je tak pravděpodobnější, že *Word Embedding* model narazí na slova, která nezná a nebo je jich málo, aby dostatečně určovala použití v kontextu.



Obrázek 6.3: Porovnání shodných konfigurací nad češtinou a angličtinou s oběma algoritmy Word2Vec modelu, v závislosti na počtu epoch

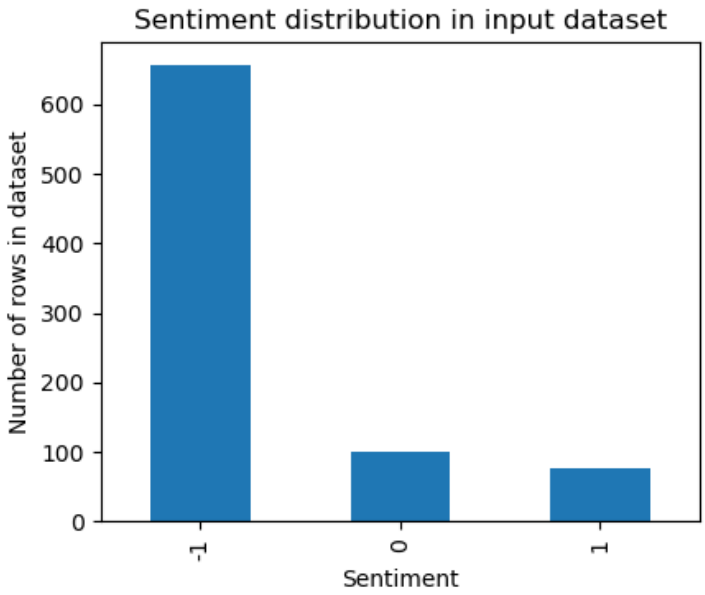
Další důležitý poznatek přináší porovnání úspěšnosti s ohodnocením při odlišné přípravě textu. Výsledné přesnosti klasifikace nejsou příliš odlišné. Je to jistě proto, že byl použitý *Word Embedding* model, využívající neuronové sítě, a stejně tak klasifikační model využívá neuronovou síť.
Díky použití neuronových sítí bylo zařazení textu zřetelnější. To mělo za následek zvýšení celkové přesnosti klasifikace testovaných dat, a tedy celkové úspěšnosti analýzy sentimentu.

6.4 Reakce na české zprávy

Aby bylo možné porozumět výsledkům je nutné nejprve zmínit zpracování dat, která se velmi liší od porovnávacích datasetů. Oba porovnávací datasety obsahovaly sentiment u každého záznamu a především obsahují mnohem více záznamů.

6.4.1 Zpracování dat

Reakce na zprávy byly získány a ohodnoceny pouze několiv dní před ukončením bakalářské práce. Z tohoto důvodu nebylo příliš času na dostatečné prozkoumání ideálního postupu pro vytrénování klasifikačního a *Word Embedding* modelu.
Celý dataset je příliš malý (pouze 833 záznamů) a z většiny negativní (viz graf *6.4 Distribuce sentimentu v reakcích na zprávy*), což jsou velmi nevyhovující podmínky pro smysluplné hledání sentimentu. Klasifikátor téměř vždy odhaduje text jako negativní a celková úspěšnost se řídí prakticky poměrem negativních záznamů a celkového počtu záznamů.



Negativní sentiment je vyjádřen v cca 79% reakcí.

Obrázek 6.4: Distribuce sentimentu v reakcích na zprávy

Kvůli malému množství dat bylo nemožné vytrénovat *Word2Vec* model

s dostatečnou kvalitou, aby byla jednotlivá slova a celé texty dostatečně rozeznatelné pro klasifikátor. Při počtu epoch na trénování *Word2Vec* do 10-ti byly téměř všechny texty klasifikovány jako negativní. To vedlo sice k téměř 80% úspěšnosti, ale byla to také jasná známka selhání celé analýzy sentimentu.
Z těchto důvodů jsou veškeré výsledky zobrazovány až od 20-ti trénovacích epoch, přičemž i do 50-ti epoch se někdy vyskytla chyba klasifikace.

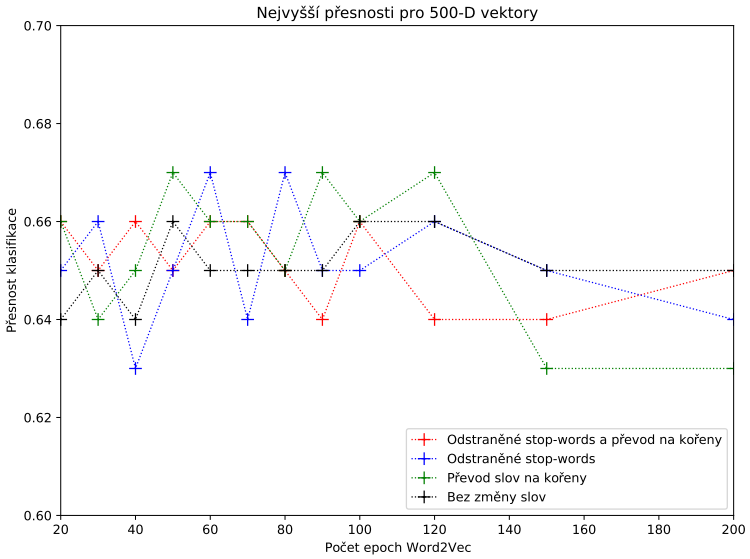
6.4.2 Výsledné testy

Výsledky vyhodnocení reakcí čtenářů na zprávy o nemoci Covid-19 jsou dle očekávání o poznání méně úspěšné. Hlavním problémem je jistě množství dat a kvůli pozdnímu vytvoření datasetu nebylo možné příliš měnit implementaci a nalézt ideální konfigurace.
V následujících grafech je ukázána podobnost úspěšnosti několika různých konfigurací. Jedná se o pouhou část provedených testů. Ostatní pokusy dopadly prakticky shodně – v rozmezí 60% až 70% celkové přesnosti klasifikace a bez znatelného zlepšení při větším počtu trénovacích epoch na *Word2Vec* model.
Testovaná data byla získána pouze pětkrát pro každou konfiguraci a z těchto výsledků je ukázán nejlepší, nebo průměrný (vždy ze všech 5-ti testů). Některé testy bohužel vykazovaly nesmyslná měření s přesnostmi klasifikace 0% pro pozitivní nebo neutrální sentimentu. Obvykle se jednalo pro minimální počty epoch trénování, a proto jsou v těchto výsledcích pouze počty epoch od 20 výše.

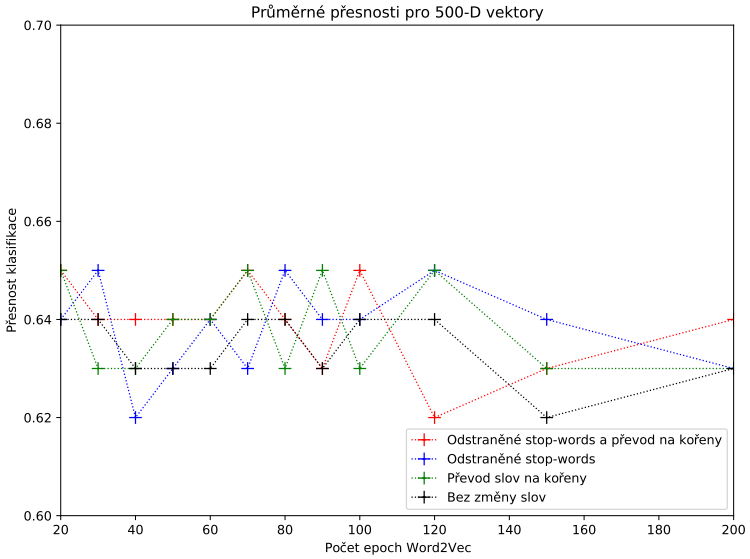
Zatímco v grafu s nejlepšími přesnostmi (obrázek 6.5) jsou výrazné odchylky, tak průměr (obrázek 6.6) se pohybuje většinou mezi 63% a 65%.
Zajímavé je, že s přibývajícím tréninkem *Word Embedding* modelu v obou porovnání dochází k mírnému poklesu přesností. Je ale jisté, že tyto výsledky jsou více nahodilé, a nevykazují žádný vzestup přesnosti.

Podobně nahodilé jsou i výsledky při porovnání různých kritérií na minimální výskyt slova pro trénování *Word2Vec* modelu – viz obrázek *6.7 Nejlepší úspěšnosti reakcí podle frekvence slov.*

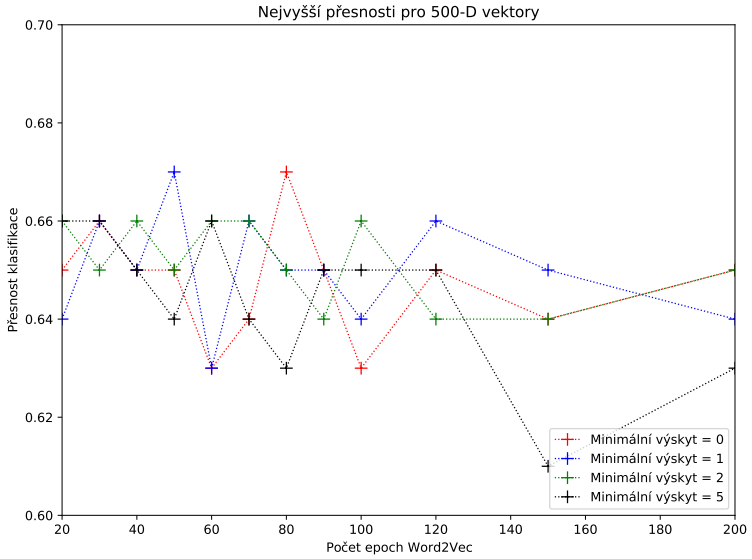
Klasifikace reakcí na zprávy nevykazuje progres ani při porovnání velikosti vektorů – viz obrázky 6.8 a 6.9. Testované byly pouze 100, 300 a 500 dimenzí velké vektory, na které byla slova převáděna.



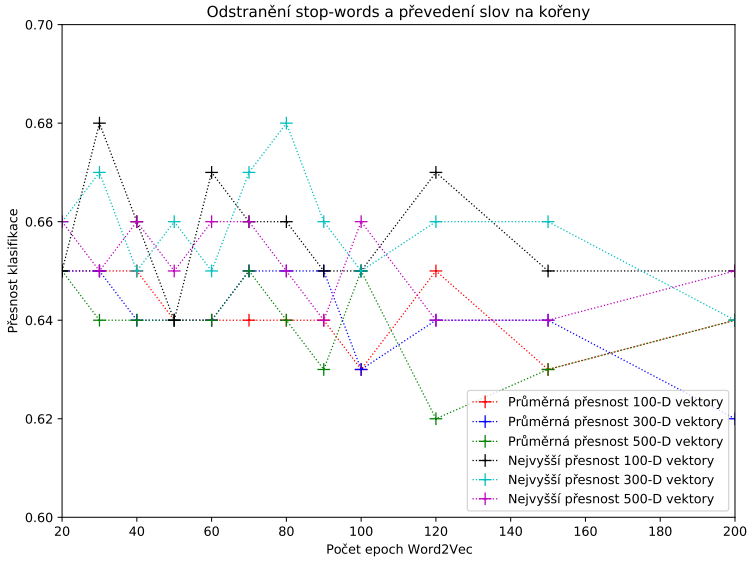
Obrázek 6.5: Porovnání nejúspěšnějších klasifikací 500-D vektorů na různě připravených reakcích, v závislosti na počtu trénovacích epoch



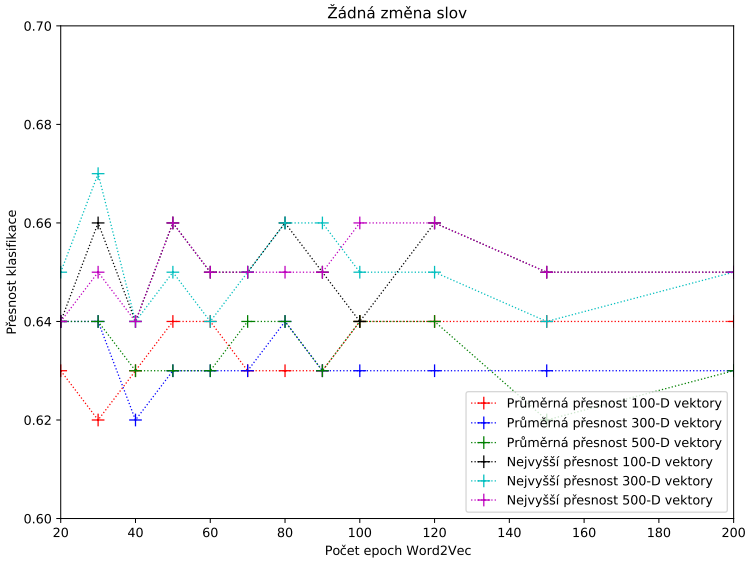
Obrázek 6.6: Porovnání průměru přesnosti klasifikací 500-D vektorů na různě připravených reakcích, v závislosti na počtu trénovacích epoch



Obrázek 6.7: Porovnání nejúspěšnějších klasifikací 500-D vektorů za použití různých minimálních frekvencích slov, v závislosti na počtu trénovacích epoch



Obrázek 6.8: Porovnání klasifikací při odstranění *stop-words* a převedení slov na kořeny, v závislosti na počtu trénovacích epoch



Obrázek 6.9: Porovnání klasifikací při zanechání *stop-words* a ponechání slov v původním tvaru, v závislosti na počtu trénovacích epoch

Kapitola 7

Diskuze výsledků

Vždy při zpracování textů hraje největší roli samotný text – především množství dat, ale i kvalita v podobě řádných mezer a slov bez chybějících písmen. Mimo to lze, konkrétně při analýze sentimentu, nejvíce ovlivnit použití metod a samotných modelů pro převod slov na *embeddings* a následnou klasifikaci textů.
Další důležitou část tvoří správný Klasifikátor. Obecně lze při použití strojového učení říci, že neuronové sítě vedou k lepším výsledkům, bohužel ale za cenu výpočetní náročnosti.

Nutné je také podotknout, že veškerá porovnání proběhla na minimální variabilitě nastavení programu a samotnou implementaci lze vylepšit doplněním o jiné *Word Embedding* i klasifikační modely.

7.1 Zpracování angličtiny a češtiny

Úspěšnost klasifikace anglických textů byla dle očekávání vyšší. Tento rozdíl však s použitím *MLP klasifikátoru* a dostatečně vytrénovaného *Word Embedding* modelu ukazuje na celkovou úspěšnost analýzy sentimentu.
Při bližším zkoumání českého datasetu je vidět, že za stejného počtu recenzí je tento text výrazně menší oproti anglickým recenzím. To ještě více podporuje anglické texty.

Rozdíl přibližně 5% v přesnosti klasifikace za daných podmínek lze považovat za úspěch pro češtinu, a tedy použitelnost metodik pro analýzu českého textu.

7.2 Reakce na české zprávy

Jak již bylo zmíněno, reakce na zprávy tvořily příliš malý a nevyvážený dataset, který je velmi nevhodný na kvalitní analýzu sentimentu. U tohoto datasetu je vhodnější předpokládat vždy negativní postoj a k tomu stačí prohlédnout histogram distribuce sentimentu *6.4 Distribuce sentimentu v reakcích na zprávy*.

Kdyby dat bylo alespoň 5-krát více, je možné, že by se klasifikace dostala na přibližně podobnou úroveň, jako je tomu u českých textů z ČSFD.cz [HB13].

Jsem přesvědčen, že kdyby šlo reakce blíže prozkoumat a ideálně přidat další, tak bych dosáhl vhodnějšího nastavení programu a nejspíš i kódu.
Vzhledem k malému množství, by ale bylo nemožné přesáhnout 75% úspěšnosti a přiblížit se tak k analýze recenzí z ČSFD.

Kapitola 8

Závěr

Dle očekávání byly provedené analýzy sentimentu úspěšnější při zpracování anglických textů. Ty jsou jednodušší na zpracování díky menšímu počtu slov v jazyce a jednoduché slovostavbě.
Jelikož ale byly použity modely využívající neuronové sítě, odlišnosti ve slovostavbě se nijak výrazně nejprojevily. Největší dopad na zpracování textu má jednoznačně celková velikost trénovacího datasetu.

Z tohoto důvodu bylo zpracování reakcí na zprávy poněkud neúspěšné při určení sentimentu. Jelikož celkový počet ohodnocených textů nepřesahoval tisíc.
Naproti tomu byly desítky tisíc recenzí pro porovnání češtiny a angličtiny, které tak dokázaly dostatečně připravit *Word Embedding* model i klasifikátor.

Analýza sentimentu v češtině je zcela reálná záležitost, která oproti analýze angličtiny vyžaduje pouze více vstupních dat. V obou případech je ideální použití moderních neuronových sítí jak pro „zachycení významu“ vět, tak pro výsledné určení postoje autora textu.

Přílohy

Příloha A

Zdroje dat a částí programu

A.1 Části kódu

Výrazná část původního kódu byla převzata z ukázky od Pierre Megret[Meg18]: www.kaggle.com/pierremegret/gensim-word2vec-tutorial .
Tento kód byl následně doplněn o části ze stránek Gensimu[1] a příprava textů z ukázky od Dipika Baad[Baa20] zde: medium.com/swlh/sentiment-classification-using-word-embeddings-word2vec-aedf28fbb8ca .
Výsledný kód představuje rozdělení částí do vlastních tříd s rozšířenou konfigurovatelností.

Pro zjištění kořenů českých slov byl v nezměněné podobě použit *czech_stemmer*, jehož autorem je Luís Gomes[2].
Tento kód byl pro účely bakalářské práce mírně pozměněn.

Další shlédnuté ukázky analýzy sentimentu či použití *Word2Vec*:

- Analýza sentimentu s *Scikit-learn*[Mal18]: stackabuse.com/python-for-nlp-sentiment-analysis-with-scikit-learn
- Návod na analýzu sentimentu s Pythonem[Sel20]: towardsdatascience.com/a-beginners-guide-to-sentiment-analysis-in-python-95e354ea84f6

[1] radimrehurek.com/gensim/models/word2vec.html [Ře19]
[2] odkaz na stránky: research.variancia.com/czech_stemmer

- Analýza sentimentu s *Word2Vec* [Var17]: www.kaggle.com/varun08/sentiment-analysis-using-word2vec
- Použití klasifikátorů z *Scikit-learn*[PVG+20]: scikit-learn.org/stable/user_guide.html

A.2 Vstupní texty a jiná data

Celkem byly použity pouze 2 datasety určené pro analýzu sentimentu a jiná strojová zpracování textů. Anglické texty byly získány od „Yelpu“ zde: www.yelp.com/dataset .
Jako zdroj českých textů byly použity rezence z protálu ČSFD přes prostředníka: liks.fav.zcu.cz/sentiment .

Mimo porovnání anglického a českého jazyka byly použity reakce čtenářů na zprávy o nemoci Covid-19. Tyto texty však kvůli nedostatečnému času na zpracování prošly pouze konfigurací implementace, která byla vhodná pro porovnání jazyků.
Reakce na zprávy byly získány z českých zpravodajských serverů[3] a ohodnoceny studenty fakulty sociálních věd Univerzity Karlovy z oboru žurnalistiky ve spolupráci s Václavem Moravcem.

Součástí implementace jsou také seznamy *stop-words*[Ded20], které byly získány zde: www.kaggle.com/heeraldedhia/stop-words-in-28-languages .
Použité obrázky a grafy, které nebyly vytvořené v rámci bakalářské práce, nesou u sebe odkaz na literaturu, ze které byly obrázky převzaty.

[3] portály idnes.cz, seznamzpravy.cz, novinky.cz, blesk.cz a lidovky.cz

Příloha B

Literatura

[Baa20] Dipika Baad, *Sentiment classification using word embeddings (word2vec)*, March 2020.

[Bra20] Bc. Lenka Brabencová, *Slovní zásoba v českém jazyce*, Prevopisně.cz (2020).

[Dai19] Shaumik Daityari, *How To Perform Sentiment Analysis in Python 3 Using the Natural Language Toolkit (NLTK)*, Digital Ocean, September 2019.

[Ded20] Heeral Dedhia, *Stop words in 28 languages*, kaggle.com, 2020.

[Dex] Allison Dexter, *How many words are in the english language?*, Word Counter.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep learning*, MIT Press, 2016.

[Gom10] Luís Gomes, *czech_stemmer*, 2010.

[HB13] Ivan Habernal and Tomáš Brychcín, *Unsupervised Improving of Sentiment Analysis Using Global Target Context*, Proceedings of RANLP 2013, Association for Computational Linguistics, 2013, p. TBD.

[Jab18] Hafsa Jabeen, *Stemming and lemmatization in python*, Data Camp (2018).

[Kar18] Dhruvil Karani, *Introduction to word embedding and word2vec*, Towards data science (2018).

[Mal18] Usman Malik, *Python for nlp: Sentiment analysis with scikit-learn*, Stack Abuse, 2018.

[MCCD] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, *Efficient estimation of word representations in vector space*, Tech. report.

[Meg18] Pierre Megret, *Gensim Word2Vec Tutorial*, kaggle, October 2018.

[Mik13] Tomas Mikolov, *Distributed representations of words and phrasesand their compositionality*, Tech. report, 2013.

[Nab18] Javaid Nabi, *Machine Learning — Word Embedding & Sentiment Classification using Keras*, October 2018.

[Pup18] Rushikesh Pupale, *Support Vector Machines(SVM) — An Overview*, Towards data science (2018).

[PVG+20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: User Guide*, Scikit-learn (2020).

[Sel20] Natassha Selvaraj, *A Beginner's Guide to Sentiment Analysis with Python*, Towards data science, September 2020.

[Sri19] Aishwarya V Srinivasan, *Stochastic Gradient Descent — Clearly Explained*, Towards data science (2019).

[Tej20] Sai Teja, *Stop words in nlp*, Medium (2020).

[Tutxx] Tutorialspoint, *Artificial Intelligence - Neural Networks*, Tutorials Point (20xx).

[Var17] Divya Varun, *Sentiment analysis using word2vec*, kaggle, 2017.

[WM14] Hoda Korashy Walaa Medhataba, Ahmed Hassan, *Sentiment analysis algorithms and applications: A survey*, Ain Shams Engineering Journal (2014).

[Yel20] *Yelp open dataset*, November 2020.

[Ře] Radim Řehůřek, *Gensim*.

[Ře19] ———, *Word2Vec embeddings*, November 2019.

Příloha C

Elektronické přílohy

Popis adresáře s přílohami, nahraného v elektronické podobě na stránkách dspace.cvut.cz spolu s touto prací:

```
./
├── Components/
│
├── Data/
│   ├── Configurations/ - konfigurace programu a skripty na vytvoření
│   │   konfigurací
│   ├── Input/ - ukázka vstupních dat, která byla použita
│   └── Results/ - výsledky porovnání konfigurací
│
└── Software/
    ├── stopwords/
    ├── classification.py
    ├── configuration.py
    ├── czech_stemmer.py
    ├── embedding_model.py
    ├── sentiment_analysis.py - hlavní skript programu
    └── text_preparation.py
```

Příloha D

Plán implementace

Zde je původní plán práce z počátku roku 2021 doplněn pouze o poslední kapitolu s výslednými časy.

Hodnocení reakcí ćtenářů na obsah mediálních zpráv – Plán na implementaci analýzy sentimentu českých textů

Autor: Lukáš Zíb

Interval implementace: 01.01. - 21.5. 2021

1 Cíle

1. Nalézt a zpracovat problematiky spojené se zpracováním a následným ohodnocením českého textu.

   - Sepsat obecný postup analýzy sentimentu.
   - V každém kroku postupu zvážit existenci možných komplikací.
   - Zohlednit obecné odlišnosti češtiny a angličtiny.
   - Kroky obecného postupu analýzy logicky navazují – od improtu dat až po použití vytrénovaného modelu.
   - Postup analýzy počítá s různými modely, které nemusí potřebovat některé z uvedených kroků.
   - Jsou rozlišeny kroky pro zpracování anglického a českého textu – za předpokladu, že bude výsledná implementace odlišná.
     V případě odlišností vzniknou 3 individuální postupy (obecný, pro anglické texty a pro české texty).
   - Nalezená problematika s českým textem reflektuje rozdílnosti v postupu analýzy a naopak.

2. Připravit implementaci v Pythonu, která dokáže klasifikovat sentiment anglických textů.

   - Implementace by měla co nejlépe reflektovat obecný postup analýzy sentimentu.
     Může se lišit v závislosti na použitých knihovnách.
   - Připravit implementaci dostatečně modulární pro použití jiného jazyku.
     Zachováním podstatné části kódu budou lépe vidět odlišnosti.
   - Výsledný model implementace dokáže odhadnout sentiment použitých anglických textů.
     Lze očekávat přesnost cca 80% – v závislosti na použitých knihovnách a přípravy textu.

3. Implementaci pro angličtinu rozšířit patřičnými způsoby, aby dokázala klasifikovat sentiment českých krátkých textů (recenze, komentáře či věcná vyjádření).

   - Zachovat podstatné části kódu pro snadné odlišení potřeb jazyků.

- Výsledný model implementace dokáže odhadnout sentiment použitých českých textů.
  Lze očekávat přesnost cca 65% – v závislosti na použitých knihovnách a přípravě textu.

4. Detekovat a objasnit odlišnosti a slabiny implementace při zpracování českých textů oproti anglickým.
   Tento cíl reflektuje nalezené problematiky (1. z cílů).
   Odlišnosti vyplývají z odlišností kódů.

   - Odlišnosti implementací jsou rozdílné pouze v postupech analýzy textu pro češtinu a angličtinu.

5. Provést diskuzi – zda a případně jak by bylo možné nedostatky minimalizovat.

2 Činnosti

Celou práci lze rozdělit na obecné milníky, kterými jsou: *příprava a odhalení rizik*, následá *implementace*, *vyhodnocení* a *doplňující text práce*.
Nutno podotknout, že popis jednotlivých činností v bakalářské práci je zahrnut v činnosti samotné, ale je to především tento popis, který nejspíš prodlouží očekávané odhady času – viz tabulka 1 Harmonogram hlavních činností.

Tabulka 1: Harmonogram hlavních činností

Činnost	*Odhad času*	*Počátek*	*Konec*
Ujasnění knihoven	10h	11.1. 2021	7.2. 2021
Postup analýzy	8h	11.1. 2021	14.2. 2021
Detekce problematiky	19h	11.1. 2021	14.2. 2021
Implementace pro AJ	55h	1.2. 2021	14.3. 2021
Implementace pro ČJ	20h - 24h	1.3. 2021	25.4. 2021
Popis implementací	29h	8.3. 2021	9.5. 2021
Diskuze a vyhodnocení	10h - 15h	15.4. 2021	9.5. 2021
Ladění textu práce	9h	26.4. 2021	23.5. 2021
CELKEM	**157h - 166h**		

V odhadovaném čase chybí zahrnutí konzultací s vedoucím a výpočetní čas počítače (doba, po kterou probíhá trénování modelů).

2.1 Budoucí činnosti

Tato kapitola obshuje výpis podčinností, které jistě bude nutné řešit. Je možné, že bude nutné podniknout další činnosti, zde neuvedené.
U podčinností s otazníkem je možné, že nebude dávat smysl jejich provedení

Tabulka 2: Plán činností

(Pod)činnost	*Odhad času*
Ujasnění knihoven	**10,0h**
Embedding model	5,0h
Klasifikační model	2,5h
Vizualizace grafů	2,5h
Postup analýzy	**8,0h**
Obecný postup na Embedding model	2,5h
Odlišit češtinu	5,5h
Detekce problematiky	**19,0h**
Odlišnosti jazyků	4,0h
Hlavní knihovny	9,0h
Obecné problémy analýzy	3,0h
Obecné problémy *Word Embeddings*	3,0h
Implementace pro AJ	**55,0h**
Struktura tříd/souborů	6,0h
Obecný kód analýzy	12,0h
Příprava textu	12,0h
Třída a funkce pro Embedding model	12,0h
Funkce pro Klasifikační model	9,0h
Příprava více variant modelů a zpracování textu	4,0h
Implementace pro ČJ	**20,0h - 24,0h**
Příprava textu	12,0h
Třída a funkce pro Embedding model	8,0h
? Příprava více variant modelů a zpracování textu	4,0h
Popis implementací	**29,0h**
Vysvětlení WordEmbeddings	11,0h
Gramatické zpracování textu	9,0h
Méně kritické části	9,0h
Diskuze a vyhodnocení	**9,5h - 15,0h**
Porovnání variant anglické verze	5,5h
? Porovnání variant české verze	5,5h
Porovnání anglické a české verze	4,0h
Ladění textu práce	**9,0h**
Kontrola návaznosti	3,0h
Gramatika	2,0h
Informace a data	4,0h

(viz tabulka 2 Plán činností). Je to hlavně proto že, modely mohou být s českými texty neúspěšné a bylo by irelevantní porovnávat neúspěšné výsledky mezi sebou.

3 Realizace plánu

Plánovaná práce byla téměř celá časově trasovaná. Chybí pouze pár hodin, které jsem strávil rešerší analýzy a knihoven, ještě před důsledným trasováním.
Tento čas odhaduji na zhruba 1 *man-day*. Zároveň jsem ušetřil asi 1 *man-day* s řešením českého převodníku slov na kořeny, který jsem našel implementovaný.
V tabulce 3 níže je souhrn časů z plánu a skutečně odvedené práce. Nutno dodat, že k závěru práce bylo trasování času do konkrétní činnosti spíše odhadováno.

Tabulka 3: Harmonogram hlavních činností

Činnost	*Odhad času*	*Naměřený čas*	*Oprava času*
Ujasnění knihoven	10h	10,5h	+2h
Postup analýzy	8h	2,5h	+6h
Detekce problematiky	19h	16,0h	
Implementace pro AJ	55h	42,5h	
Implementace pro ČJ	20h	18,5h	+7h
Popis implementací	29h	34,5h	
Diskuze a vyhodnocení	10h	12,0h	
Ladění textu práce	9h	8,5h	
CELKEM	**157h**	145h + 15h	

Mohu konstatovat, že skutečně strávený čas odpovídá naplánovanému času. Jelikož došlo k mírným odchylkám, tolerance 3h je jistě na místě.

Myslím ale, že bylo možné na práci strávit celkově více času a velmi tím posílit fakta i zpříjemnit použití implementace.
Celkový čas odpovídá spíše minimu k naplnění cílů, přičemž toto naplnění bylo pár týdnů před závěrem práce velmi nejisté, kvůli zpoždění získání českých reakcí na zprávy.