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

Diplomová práce

Kategorizace zákaznické zpětné vazby nezávislá na jazyce

Plzeň 2023 Pavel Průcha

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2022/2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Pavel PRŮCHA**
Osobní číslo:	**A20N0102P**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Téma práce:	**Kategorizace zákaznické zpětné vazby nezávislá na jazyce**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se s metodami vícejazyčného zpracování textu a úlohou kategorizace zákaznické zpětné vazby.
2. Vytvořte testovací korpus z recenzí vybraného řetězce na Google Reviews.
3. Navrhněte metodu, která bude kategorizovat recenze napsané v různých jazycích do předem definovaných kategorií. U každé zmínky kategorie bude také detekovat polaritu.
4. Metodu implementujte a vytvořte jednoduchý demonstrátor.
5. Otestujte kvalitu kategorizace a výsledky zhodnoťte.

Rozsah diplomové práce: **doporuč. 50 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

dodá vedoucí diplomové práce

Vedoucí diplomové práce: **Doc. Ing. Josef Steinberger, Ph.D.**
Katedra informatiky a výpočetní techniky

Datum zadání diplomové práce: **9. září 2022**
Termín odevzdání diplomové práce: **18. května 2023**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 11. října 2022

Prohlášení

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

V Plzni dne 12. května 2023

Pavel Průcha

Poděkování

Rád bych poděkoval vedoucímu diplomové práce doc. Ing. Josefovi Steinbergerovi, Ph.D. za odborné vedení a cenné rady při zpracování práce.

Abstract

Categorization of customer feedback is used by various companies to improve the quality of the product or service they offer. This thesis deals with language-independent categorization of customer feedback. Conversion between languages uses vector space transformation using a transformation matrix and machine translation. The data corpus for training and testing classifiers is created from reviews of the chain *McDonald's*, in which the sentiment in selected categories is then manually annotated. In this way, the training corpus is created from Czech reviews and the test corpus from English and German reviews. Data are tested in various combinations primarily on neural networks *CNN* and *LSTM* with word embeddings *word2vec* and *fasttext*. The most successful combination of models is *LSTM* with *fasttext*, which are used in the work demonstrator.

Abstrakt

Kategorizaci zákaznické zpětné vazby využívají různé společnosti pro zkvalitnění produktu nebo služby, kterou nabízí. Tato diplomová práce se zabývá kategorizací zákaznické zpětné vazby nezávislé na jazyce. Konverze mezi jazyky využívá transformaci vektorového prostoru pomocí transformační matice a strojový překlad. Datový korpus pro trénování a testování klasifikátorů je vytvořen z recenzí řetězce *McDonald's*, ve kterých je následně manuálně označený sentiment ve vybraných kategoriích. Tímto způsobem je vytvořený trénovací korpus z českých recenzí a testovací korpusy z anglických a německých recenzí. Data jsou v různých kombinacích testována primárně na neuronových sítích *CNN* a *LSTM* s vektorovou reprezentací textu *word2vec* a *fasttext*. Nejúspěšnější kombinací modelů je *LSTM* s *fasttext*, která je použita v demonstrátoru práce.

Obsah

1 Úvod

Tato práce se věnuje kategorizaci zákaznické zpětné vazby nezávislé na jazyce. Kategorizace je technika, která se využívá při zpracování přirozeného jazyka nebo-li *NLP* (z anglického *natural language processing*). Zpracování přirozeného jazyka je obor, který se úzce dotýká matematiky, informatiky a lingvistiky. Tento obor má mnoho využití – analýza mínění, detekce tématiky, rozpoznávání jazyka, extrakce klíčových slov, kategorizace a mnoho dalšího. Výsledkem *NLP* je počítačový program, který pomocí dat „porozumí“ problému a vrátí jeho řešení nebo výsledek. Zkoumání přirozeného jazyka má několik rovin, které se liší svým zaměřením a nástroji, které potřebujeme pro jejich zpracování. Klasifikace zákaznické zpětné vazby patří do roviny sémantické. [4]

Zákaznická zpětná vazba bývá často recenze, kterou zákazník píše na webové stránky daného podniku, který navštívil, nebo na stránkách, které o podnicích sbírají informace nebo o nich píší. Kategorizace je přiřazení textu do několika tématických celků a určuje tak, čeho všeho se text týká. U recenzí, které hodnotí restauraci, mohou být kategorie *„jídlo“*, *„nápoje“* nebo *„obsluha“*.

Při kategorizaci textu se obvykle zároveň detekuje polarita sentimentu dané kategorie. Polarita sentimentu určuje, jaký je emocionální tón textu. Tedy jestli je text pro danou kategorii pozitivní nebo negativní. Příkladem kategorizace textu a určení polarity sentimentu je věta *„Vaří zde výborně, ale kafe je slabé a studené a personál neschopný.“*. Výsledkem bude, že pro kategorii *„jídlo“* je text pozitivní, ale pro kategorie *„nápoje“* a *„obsluha“* je text negativní.

Konceptem této práce je malý podnik restaurace v České republice, která chce vylepšit své služby v několika specifických kategoriích na základě zákaznické zpětné vazby získané z recenzí na *Google Reviews*. Protože se ale jedná o malý podnik, má pro klasifikaci jen malý počet recenzí v českém jazyce. Tato data je potřeba nejprve podle daných kategorií označkovat (manuálně určit sentiment textu). Restaurace je blízko hranic s Německem, proto by aplikace měla být schopná klasifikovat i anglické a německé recenze. Zpracování recenzí jiného než českého jazyka je možné klasifikovat pomocí překladu a následně stejného postupu jako u českých recenzí nebo za použití transformace vektorového prostoru cizího jazyka.

Práce je organizovaná do devíti kapitol. V první části práce v kapitole `Zpracování přirozeného jazyka` se práce zabývá úlohou kategori-

zace zákaznické zpětné vazby. V této kapitole jsou klíčové části rozděleny do sekcí `Předzpracování textu`, `Reprezentace textu` a `Strojové učení`. Dále jsem se seznámil s metodami vícejazyčného zpracování textu, které jsem popsal v kapitole `Vícejazyčné zpracování slovních vektorů`. Po seznámení s problematikou jsem vytvořil korpus z recenzí vybraného řetězce *McDonald's* na *Google Reviews*. Postup a problémy s tím spojené zmiňuji v kapitole `Vytvoření testovacího korpusu`. Následně v kapitole `Implementace` popisuji architekturu zdrojového kódu a implementaci klíčových částí. V kapitole `Experimenty a testování` testuji a představuji navrženou metodu s jejími výsledky, která úspěšně kategorizuje recenze napsané v různých jazycích do předem definovaných kategorií a u každé této kategorie detekuje polaritu. Následně v kapitole `Hodnocení výsledků` jsou zhodnoceny výsledky testování. V závěru v kapitole `Demonstrátor aplikace` ukazuji demonstrátor vybrané metody a způsob, jak ho použít.

2 Zpracování přirozeného jazyka

Přirozeným jazykem se myslí jazyk, který lidé používají ke každodenní komunikaci. Jsou to jazyky jako čeština, angličtina, nebo polština.

Tyto jazyky se časem vyvíjely často přes desítky tisíc let, proto je poměrně složité je definovat pomocí explicitních pravidel. Přestože umělé jazyky jsou vytvořené účelově a také se v průběhu času vyvíjí, musí dodržovat jednoduchá pravidla, která jazyk definují. Takovými jazyky jsou například jazyky programovací a matematické zápisy.

Je samozřejmé, že technologie založené na *NLP* jsou stále rozšířenější. Například mobilní telefony už poměrně dlouhou dobu podporují prediktivní rozpoznávání textu a rukopisu, webové vyhledávače umožňují přístup k informacím uzavřeným v nestrukturovaném textu, strojový překlad nám umožňuje získávat texty napsané v angličtině a číst je v češtině, textová analýza nám umožňuje detekovat sentiment v tweetech nebo recenzích. Poskytováním přirozenějších rozhraní *human-machine* a sofistikovanějším přístupem k uloženým informacím hraje jazykové zpracování zásadní roli ve vícejazyčné informační společnosti. [11]

Obor zpracování přirozeného jazyka začal ve 40. letech 20. století, kdy si díky druhé světové válce lidé uvědomili důležitost překladu z jednoho jazyka do druhého a chtěli vytvořit stroj, který by tento překlad zautomatizoval. Od té doby se *NLP* vyvíjí až do dnešní doby. [17]

Jak bylo psáno v kapitole `Úvod`, zkoumání přirozeného jazyka má několik rovin, které se liší svým zaměřením a nástroji, které potřebujeme pro jejich zpracování. Takových rovin je celkem šest. První z nich je **Lexikální** rovina, do které patří slovní zásoba a její proměny. Tedy *„Co dané slovo znamená?“*. Druhou je rovina **morfologická**, kam patří slovní tvary a způsob tvoření nových slov. Smyslem této roviny je *„Jak skloňovat dané slovo?“*. Další rovina je rovina **syntaktická**. Zde jde o shlukování slov do větších celků neboli frází a organizace frází ve větě. U této roviny se ptáme *„Je nějaký rozdíl mezi frází A a frází B?“*. Další rovinou je rovina **logická**. Obsahuje reprezentaci vět pomocí logických formulí. Do **sémantické** roviny patří význam celku a jednotlivých částí a v poslední **pragmatické** rovině hledáme význam promluvy v kontextu. [4]

Zpracování přirozeného jazyka má velký rozsah aplikací pro počítačové manipulace přirozeného jazyka. V jednoduchých aplikacích může jít o triviální počítání frekvencí slov k porovnání různých stylů psaní a v těch složitějších jde o „porozumění“ tomu, co lidé řeknou nebo napíší, alespoň do té míry, že na ně dokážeme dávat užitečné odpovědi. [11]

Dobrým příkladem *NLP* dnešní doby je **Chat Generative Pre-training Transformer** (zkratka *GPT*). *ChatGPT* ukazuje významný pokrok v *NLP*. Jde o technologii umělé inteligence vyvinutou společností *OpenAI*. Tato technologie se vyznačuje velkou přesností a rychlostí. Jde o chatbota (počítačový program určený k automatizované komunikaci s lidmi [26]), který „rozumí“ lidskému přirozenému jazyku, čte, co uživatel napíše, a následně odpovídá. Za krátkou sekundu dokáže zpracovat miliardy slov a úspěšně odpovědět na zadanou zprávu. Pracuje na základě před-trénované neuronové sítě na obrovské datové sadě textu. Díky ní a vstupnímu textu generuje odpověď. Velmi pokroková je zde kvalita porozumění chatbota kontextu konverzace s uživatelem, které dosahuje pomocí *self-attention* mechanismu. Tento mechanismus dokáže hodnotit váhu důležitosti různých slov a frází na základě relevance k tématu. Protože *ChatGPT* je schopný řešit i složitější úkoly a problémy, je použitelný pro velkou škálu aplikací. Přestože ani *ChatGPT* není bezchybný a ne vždy jsou jeho odpovědi přesně takové, jaké můžeme očekávat, ukazuje, že obor *NLP* se stále úspěšně vyvíjí. [30]

Důležitými prvky *NLP* je předzpracování textu a vektorová reprezentace textu popsané v následujících sekcích.

2.1 Předzpracování textu

Aby se s daty dalo dobře pracovat, je nutné, aby prošla několika procesy, které tato data připraví na hlavní operace. Tento komplex procesů se nazývá předzpracování. U předzpracování textu do něj mimo jiné patří *odstranění stop slov, tokenizace, stemming, lematizace, lowercase* a *truecase*.

Důvodem předzpracování je, že všechny informace, které dostaneme, nemusí být nutně užitečné. Předzpracováním informací se může snížit počet slov sjednocením slov stejného významu a odstraněním slov a symbolů irelevantní pro téma, které nás zajímá. Tím se tedy zmenší i velikost modelu pro následnou práci, ale model se stane relevantnějším. [8]

2.1.1 Stop slova

Odstranění **stop slov** bývá jeden z první kroků předzpracování textu. Jde o slova, která se běžně používají a odstraňují z věty jako předzpracování v různých úlohách zpracování přirozeného jazyka. Každý nástroj používá trochu jinou sadu seznamu stop slov, které odstraňuje, v závislosti na tématu úlohy, protože v různých tématech je relevance a irelevance klíčových slov jiná. Příklady často používaných stop slov jsou: *'a', 'aby', 'ačkoli', 'ahoj', 'ale', 'ani', 'asi', 'až', 'bez'* atd. [9]

2.1.2 Tokenizace

Tokenizace je proces předzpracování textu, při kterém je text rozdělen do pole po částech, které se nazývají tokeny. Tokeny jsou jednotlivá slova textu, která jsou často navíc upravována, podle způsobu tokenizace. V rámci tohoto procesu může být odstraněna interpunkce i diakritická znaménka a písmena slov mohou být převedena na malá. Celková tokenizace pomáhá provádět transformace na každé slovo zvlášť. Příkladem převedení je věta *'Příliš žluťoučký kůň úpěl ďábelské ódy.'* převedená na tokeny *'prilis', 'zlutoucky', 'kun', 'upel', 'dabelske' a 'ody'*.[8]

2.1.3 Stemming

Stemming je dalším krokem procesu předzpracování textu. Zde se každé slovo redukuje na kořen slova odstraněním přípony a předpony. Stemming se používá často v *SEO* aplikacích (optimalizace pro vyhledávače z anglického *search engine optimization*), vyhledávání na webu a v různém získávání informací. Pokud se kořen někde shoduje v textu, pomáhá získat všechny související dokumenty při hledání. [8]

2.1.4 Lematizace

Lematizace je podobná procesu stemming. Na rozdíl od stemmingu , používá gramatická pravidla a slovník pro mapování slov do kořenové formy. Slova *'jsem'* a *'je'* převede na slovo *'být'*, *'vykoupit'* a *'koupila'* převede na *'koupit'*.

2.1.5 Lowercase, truecase, interpunkce, diakritika

Dalšími kroky předzpracování textu může být *lowercase*, *truecase*, *odstranění interpunkce* a *odstranění diakritiky*.

Lowercase je jednoduché převedení všech písmen na písmena malá. Výhodou je sjednocení slov začínající velkým písmenem na začátku věty, jejichž normální forma velké písmeno na začátku nemá, a stejných slov s malým písmenem uprostřed věty. Nevýhodou je sjednocení slov, která mají jiný význam s malými a velkými písmeny.

Řešením tohoto problému je metoda **truecase**, při které velká písmena zůstanou u slov, kde v normální formě velká písmena jsou. To jsou většinu jména a názvy.

Odstranění diakritiky se používá, pokud očekáváme, že některá část dat může být psaná bez diakritiky nebo je diakritika použitá chybně a chceme tato slova opět sjednotit.

Ze stejného důvodu se dá odstranit i **interpunkce**. Ta ale může sloužit k odlišení jednotlivých částí textu, což může být prospěšné.

2.2 Reprezentace textu

Běžný počítač umí dobře pracovat s čísly. Ve výpočtech i kódování je výrazně rychlejší než člověk, proto na něj v mnoha ohledech spoléháme a necháme ho pracovat za nás. To, s čím ale počítač pracovat neumí, je jazyk v syrové podobě. Řešením tohoto problému je transformovat slova na něco, čemu počítač rozumí a to jsou číselné vektory. Tomuto kroku se říká reprezentace textu a jde o klíčovou část cesty k modelu strojového učení. Reprezentaci textu můžeme rozdělit na dvě části: diskrétní reprezentace textu a distribuční reprezentace textu. [12]

2.2.1 Diskrétní reprezentace textu

V **diskrétní reprezentaci textu** jsou slova reprezentována jejich odpovídajícími indexy k jejich pozici ve slovníku z korpusu [12].

Některé používané reprezentace textu jsou *One-Hot encoding, Bag of words* a *Tf-Idf*. Pro tyto reprezentace je potřeba nejprve vytvořit slovník z korpusu trénovacích dat. Slovník obsahuje každé jedinečné slovo korpusu.

One-hot

Nejzákladnější reprezentace textu je **One-hot**. Tato reprezentace vytvoří takový vektor pro každé slovo ve větě, že velikost vektoru je rovna velikosti slovníku, tedy počtu různých slov korpusu. Vektoru přiřadí jednou hodnotu 1, podle pozice daného slova ve slovníku, a ve zbytku vektoru budou jen hodnoty 0.

Příklad dvou sousloví a jejich výsledná *One-Hot* reprezentace:

jedna jedna dva tri dva jedna tri
[[1,0,0,0], [1,0,0,0], [0,1,0,0], [0,0,1,0], [0,1,0,0]], [1,0,0,0], [0,0,1,0]]
tri ctyri dva tri
[0,0,1,0],[0,0,0,1],[0,1,0,0],[0,0,1,0]

Bag of words

U reprezentačního modelu **Bag of words** (zkratka BOW) se vektor vytváří jako pole frekvencí každého slova ze slovníku ve větě [8]. Velikost vektoru je tedy rovna velikosti slovníku vytvořeného z korpusu pro trénování dat.

Příklad dvou sousloví a jejich výsledná *Bag of words* reprezentace:

jedna jedna dva tri dva jedna tri
[3,2,2,0]
tri ctyri dva tri
[0,1,2,1]

Tf-Idf

Z výše zmíněných je nejsložitější textová reprezentace **Tf-Idf**. Počítá se vynásobením lokální a globální složky. Lokální složka je frekvence termů (z anglického *term frequency*, zkratka TF) a globální složka je inverzní frekvence dokumentu (z anglického *inverse document frequency*, zkratka IDF). [10]

Pro výpočet je nejprve nutné spočítat váženou frekvenci termu pro každý dokument *d* vlastnící term *t* udávající relevanci termu v dokumentu. Vypočítá se podle vzorce:

$$tf_{t,d} = \log(1 + wf_{t,d})$$

kde $wf_{t,d}$ je frekvence termu *t* v dokumentu *d* a znamená, kolikrát se term v dokumentu vyskytuje.

Druhým důležitým vzorcem je vzorec:

$$idf_t = \log \frac{N}{df_t},$$

Tento vzorec počítá inverzní frekvenci termu v dokumentu *t*. Jedná se o důležitost termu. *N* je počet dokumentů a df_t je frekvence dokumentů, ve kterých se term *t* vyskytl. Čím častěji se slovo vyskytuje v dokumentech, tím méně je důležité.

Složka výsledného vektoru dokumentu *d* se vypočítá vynásobením $wf_{t,d}$ a idf_t pro daný term *t* je tedy vzorec:

$$tf - idf_{d,t} = tf_{t,d} \quad idf_t,$$

a výsledkem je *tf-id* $f_{d,t}$ vektor. [27]

2.2.2 Distribuční reprezentace textu

Distribuční reprezentací textu se myslí reprezentace, kde reprezentace slova je vektor fixní velikosti složený z kladných i záporných reálných čísel a reprezentace slova není nezávislá nebo se vzájemně nevylučuje s jiným slovem. Takže informace o slovu je distribuována ve vektoru, kterým je reprezentována. To se liší od diskrétní reprezentace, kde je každé slovo považováno za jedinečné a nezávislé. [13]

Distribuční reprezentace textu může být statická nebo dynamická.

Statická distribuční reprezentace textu

Nevýhodou statické distribuční reprezentace textu je, že slovní vektory slučují sémantiku *polysémů* (slova, která mohou mít více významů) a po tréninku se nemění podle kontextu. Příkladem je slovo „kolo“, které může mít význam jízdního kola, součást automobilového vozidla nebo Loukoťové kolo. [18]

K nejznámějším statickým distribučním reprezentacím textu patří *Word2vec, GloVe* a *Fasttext*.

Word2vec

Tradičním způsobem reprezentace slov je diskrétní reprezentace slov jako je *one-hot*. Nevýhodou této reprezentace je, že nelze odvodit žádný vztah mezi dvěma slovy. Tento problém řeší reprezentační model **word2vec**. Ten využívá kontext cílových slov tak, že použije okolní slova k reprezentaci slov cílových pomocí neuronové sítě, jejíž skrytá vrstva kóduje reprezentaci slova. Model projde všechna slova v textu a přiřadí jim dva vektory – jeden vektor reprezentuje slovo, když je kontextové, a druhý vektor reprezentuje slovo, když se jedná o slovo cílové. [20]

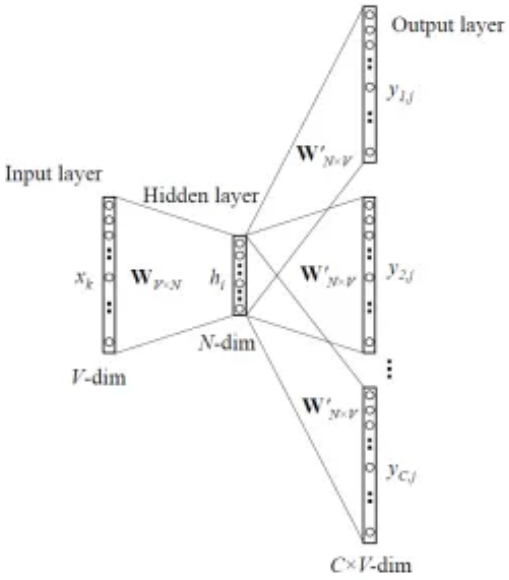
Existují dva typy **Word2Vec** – *Skip-gram* a *Continuous Bag of Words* (dále jen CBOW).

Typ **skip-gram** funguje tak, že vstupem je cílové slovo a výstupem jsou slova, která cílové slovo obklopují. Příkladem je věta „V této restauraci vaří

výborně“. Pokud by vstupem bylo slovo „vaří“, pak by výstupem byla slova „V“ „této“ „restauraci“ a „výborně“. Všechna vstupní i výstupní data jsou zakódována pomocí *one-hot* a mají stejnou dimenzi. V závěru výstupní vrstvy je aktivována funkce *softmax*. [20]

Aktivační funkce je funkce, která transformuje vážený součet neuronu tak, aby byl výstup nelineární. Výsledkem aktivační funkce je číslo od 0 do 1, které vyjadřuje pravděpodobnost, že patří do dané třídy. Na rozdíl od funkce *sigmoid*, která se používá pro binární klasifikaci, je funkce *softmax* využívána spíše pro klasifikace více tříd. Výsledkem je pravděpodobnost pro každou třídu. [16]

Prvek výstupního vektoru ukazuje pravděpodobnost výskytu daného slova v kontextu. Struktura neuronové sítě typu *skip-gram* je znázorněna na obrázku 2.1 na straně 17.

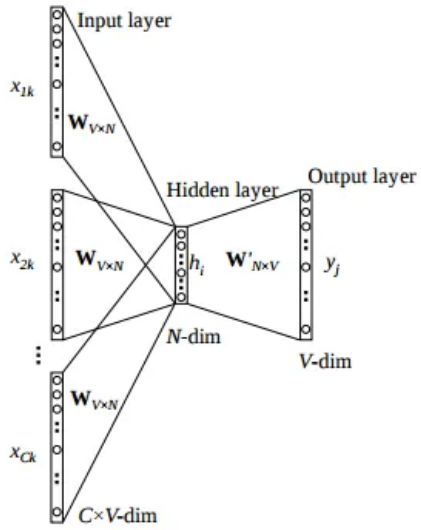


Obrázek 2.1: Struktura neuronové sítě typu *skip-gram*, zdroj: [21]

Typ **CBOW** se liší od typu *skip-gram* tím, že zaměňuje vstup a výstup. Cílem je zjistit, jaké slovo se s největší pravděpodobností objeví vzhledem ke kontextu. Rozdíl mezi oběma typy je v tom, jak se generují vektory slov. U *CBOW* se všechny příklady s cílovým slovem vloží do sítě a výstupem je průměr extrahované skryté vrstvy. Příkladem jsou věty „V této restauraci vaří výborně“ a „O této restauraci se hodně povídá“ a cílové slovo je „restauraci“. Pro získání vektorové reprezentace cílového slova musíme do

neuronové sítě vložit obě věty a vypočítat průměr hodnot ve skryté vrstvě. [20]

Struktura neuronové sítě typu *CBOW* je znázorněna na obrázku 2.2 na straně 18.



Obrázek 2.2: Struktura neuronové sítě typu *CBOW*, zdroj: [21]

Fasttext

Textová reprezentace **Fasttext** byla v roce 2016 vytvořená laboratoří *Facebooku AI Research* jako vylepšení původní *word2vec*. Funguje na podobném principu jako *word2vec*. Nicméně *fasttext* nevkládá do neuronové sítě celá slova, ale pouze n-gramy. N-gramy jsou sekvence *n* po sobě jdoucích znaků. Například trigramy pro slovo „dobrý“ budou: „dob“, „obr“ a „brý“. Výsledným vektorem pro slovo „dobrý“ bude součet n-gramů „dob“, „obr“ a „brý“. Díky tomu je možné získat i vektory slov, které se přímo nenacházejí ve vstupních datech, protože je velmi pravděpodobné, že jejich n-gramy v datech jsou a mohou se z nich tak poskládat. To je velká výhoda reprezentace *fasttext*. [20]

2.2.3 Dynamická distribuční reprezentace textu

Na rozdíl od statické distribuční reprezentace textu může dynamická mít pro stejné slovo různou vektorovou reprezentaci. Tím nedochází k falešně stejnému významu u polysémů. Mezi používané reprezentace patří *ELMO* a *BERT*. [18]

ELMO

Embeddings from Language Models (česky *Vektory z jazykových modelů*, dále jen *ELMO*) jsou modely využívající hluboké neuronové sítě a obousměrné jazykové modelování. Tento model byl představen v roce 2018 výzkumníky z *Allen Institute for Artificial Intelligence*. *ELMO* vytváří dynamické vektorové reprezentace, které zohledňují i kontext, ve kterém se dané slovo vyskytuje, na rozdíl od statické distribuční reprezentace textu, jako je *Word2vec, Fasttext* nebo *GloVe*.

ELMO se skládá ze tří částí. První část je síť pro kódování slov, která vytváří reprezentaci slova na základě jeho vlastní podoby. Druhou částí je síť pro kódování vět. Tato část vytváří reprezentaci celé věty, která zahrnuje informace o sousedících slovech, jde tedy o kontext. Poslední část kombinuje reprezentaci slova s jeho kontextem do výsledné reprezentace daného slova. Tyto části jsou společně trénovány na velkém množství dat, které je založené na kontextu, ve kterém se slovo vyskytuje.

ELMO využívá také techniku obousměrného jazykového modelování. To znamená, že sítě jsou trénovány na predikci následujícího slova v obou směrech - od začátku věty a od konce věty. Díky tomu se zajistí, že reprezentace slov jsou vytvářeny na základě celého kontextu věty a ne jen na základě jednoho směru věty. [23]

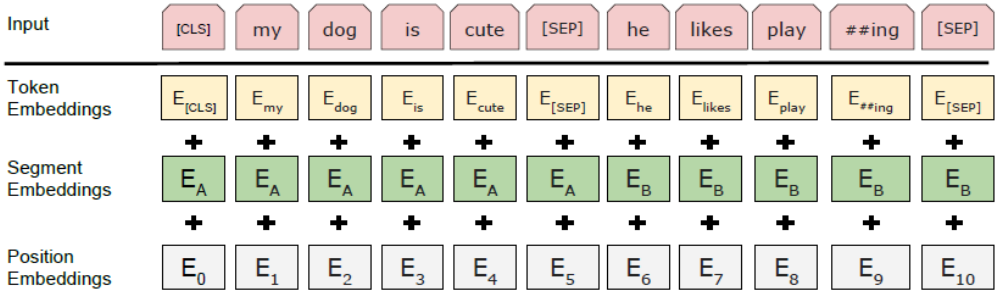
BERT

Bidirectional Encoder Representations from Transformers (česky *Obousměrné kódovací reprezentace z transformátorů*, dále jen *BERT*) je před-trénovaná technika strojového učení založená na transformátorech pro zpracování přirozeného jazyka a je vyvinutá společností *Google* v roce 2019.

BERT používá obousměrný kontext jazykového modelu. Snaží se maskovat oběma směry, tedy jak zleva doprava, tak zprava doleva, aby vytvořil mezilehlé tokeny, které se použijí pro úlohy predikce. Z toho důvodu se mu říká obousměrný. *ELMO* se zaměřuje na kontextovou závislost slov, zatímco *BERT* je zaměřen na modelování vztahů mezi slovy v celém textu.

Vstupní reprezentace do modelu *BERT* je součtem vektoru tokenu, segmentačního vektoru a vektoru pozice a dodržuje maskovací strategii pro model, aby předpověděl správné slovo v kontextu. Příklad je vidět na obrázku 2.3 na straně 20.

BERT využívá síť transformátorů a *self-attention* mechanismu, který se učí kontextový vztah mezi slovy a je vyladěn tak, aby mohl vykonávat další úkoly. [14]



Obrázek 2.3: Součet vektorů tokenů (*Token Embeddings*), segmentů (*Segment Embeddings*) a pozic (*Position Embeddings*), zdroj: [14]

2.2.4 Podobnost vektorů

Ve vektorovém prostoru, kde každé slovo má svůj vektor, nám podobnost vektorů udává míru, jak jsou si dvě slova v daném prostoru navzájem podobná. Tato míra se většinou získává pomocí **kosinové podobnosti** (z anglického *cosine similarity*) a vypočítá se kosinem úhlu těchto vektorů.

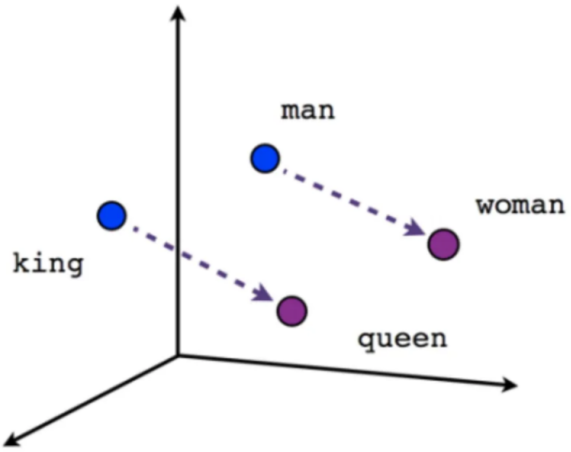
Vzorec pro kosinovou podobnost je následující:

$$\cos(\theta) = \frac{\mathbf{a} \cdot \mathbf{b}}{\mathbf{a} \quad \mathbf{b}} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} \sqrt{\sum_{i=1}^{n} b_i^2}}$$

Kde **a** a **b** jsou dva vektory v prostoru a θ je úhel mezi nimi. $\mathbf{a} \cdot \mathbf{b}$ znamená skalární součin mezi vektory **a** a **b** a **|a|** a **|b|** jsou délky vektorů **a** a **b**.

Výsledkem podobnosti vektorů je desetinné číslo 0 až 1. Pokud vyjde číslo 0, úhel vektorů svírá 90° a slova si nejsou významově podobná. Pokud vyjde číslo 1, jde o úhel 0° a slova jsou stejná nebo významově velmi podobná. Čísla mezi 0 až 1 udávají podobnost podle hodnoty.[15]

Vektory získané odečtením dvou příbuzných slov někdy vyjadřují smysluplný pojem, jako je rod, povolání nebo slovesný čas. Jak je znázorněno na obrázku 2.4 na straně 21, podobnost vektorů *„king“* a *„queen“* by měla být přibližně stejná jako podobnost vektorů *„man“* a *„woman“*.



Obrázek 2.4: Podobnost vektorů pohlaví, zdroj: [21]

2.3 Strojové učení

Strojové učení je obor umělé inteligence, který se zabývá studiem algoritmů a technik, které umožňují počítačům automaticky se učit z dat a tak zlepšovat své výkony v konkrétní úloze bez explicitního programování. Strojové učení se snaží vytvořit obecné modely, které dokážou zpracovávat data a provádět úlohy bez potřeby přesného programování pro každou jednotlivou situaci. Čím více dat mají k dispozici, tím více se aplikace zlepšují a stávají se přesnějšími. Strojové učení se využívá ve všech oblastech kolem nás. [6]

V závislosti na povaze dat a požadovaném výsledku můžeme modely strojového učení rozdělit na strojového učení *S učitelem* (z anglického *supervised*) a *Bez učitele* (z anglického *unsupervised*).

Učení s učitelem

V algoritmech učení s učitelem se stroj učí pomocí příkladů. Modely kontrolovaného učení se skládají z dvojic dat vstupu a výstupu, kterým se říká trénovací data. Cílem je naučit stroj, aby na specifický vstup odpověděl specifickým výstupem. Systému se předkládají tyto dvojice a ten začíná určovat korelační podobnosti, rozdíly a další logické body. Čím více trénovacích dat dostane, tím více přesnější bude. Modely učení s učitelem se používají v mnoha aplikacích, se kterými se setkáváme každý den. Jsou to nástroje pro doporučení produktů, aplikace pro analýzu provozu jako je *Waze*, které předpovídají nejrychlejší trasu v různých denních dobách a spoustu dalších. [6]

Učení bez učitele

V modelech učení bez učitele chybí výstup, kterého by se stroj mohl držet. Stroj studuje vstupní data a snaží se identifikovat vzory a korelace pomocí všech relevantních a dostupných dat. Je to metoda pozorování a samostatného učení ze zkušeností a chování okolí. U strojů je „zkušenost“ definována množstvím dat, která jsou vložena a zpřístupněna. Čím více dat přijde, tím více může stroj lépe kategorizovat a identifikovat odlišnosti. Mezi běžné příklady aplikací učení bez dozoru patří rozpoznávání obličeje, analýza sekvence genů, průzkum trhu a kybernetická bezpečnost. [6]

2.3.1 Analýza sentimentu

Analýza sentimentu je analytická technika, která využívá strojové učení a zpracování přirozeného jazyka k určení emocionálního tónu komunikace. Je to způsob, kterým společnosti vyhodnocují zpětnou vazbu od spotřebitele. Může jít o telefonní hovory, emaily, recenze na internetu, příspěvky na sociálních sítích a mnoho dalšího. Tím mohou zlepšit své služby nebo produkty a zvýšit tak zájem zákazníků. Často se také používá pro automatickou cenzuru příspěvků na diskuzních fórech, kdy je příspěvek automaticky zablokován, pokud je jeho emocionální tón příliš silný nebo útočný.

Počet a typ emocionálních tónu se může lišit na základě společnosti, typu komunikace, množství dat a spoustě dalších faktorů. Primární typy jsou *pozitivní*, *negativní* a případně *neutrální*. Někdy se mohou objevit i *silně pozitivní* a *silně negativní* emocionální tóny. [19]

2.3.2 Metriky vyhodnocení

Mezi základní metriky klasifikace patří *precision* (přesnost pro pozitivní metriky), *recall* (úplnost), *F1 score* (výkonnost klasifikačního modelu) a *accuracy* (přesnost klasifikace). Hodnoty se počítají pomocí TP (true positive) – správně klasifikovaných pozitivních příkladů, TN (true negative) – správně klasifikovaných negativních příkladů, FP (false positive) – špatně klasifikovaných pozitivních příkladů a FN (false negative) – špatně klasifikovaných negativních příkladů.

Precision

Klasifikační metrika **precision** měří podíl správně klasifikovaných pozitivních příkladů k celkovému počtu pozitivních příkladů.

Vzorec metriky je následující:

$$\text{precision} = \frac{TP}{TP + FP}$$

Recall

Klasifikační metrika **recall** vyjadřuje, jakou část pozitivních příkladů model identifikoval správně. Pokud je *recall* vyšší, znamená to, že model má vysokou schopnost rozpoznat pozitivní případy.

Vzorec metriky je následující:

$$\text{recall} = \frac{TP}{TP + FN}$$

F1 score

Klasifikační metrika **F1 score** je harmonický průměr mezi *precision* a *recall*. Vyšší hodnota **F1 score** indikuje lepší kvalitu klasifikace.

Vzorec metriky je následující:

$$\text{F1 score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2TP}{2TP + FP + FN}$$

Accuracy

Klasifikační metrika **accuracy** měří podíl správně klasifikovaných příkladů k celkovému počtu příkladů.

Vzorec metriky je následující:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

2.3.3 Klasifikační algoritmy

Klasifikace je proces rozdělování objektů do skupin na základě jejich společných vlastností. Jedná se o strojové učení s učitelem, kde text je vstup a výsledný sentiment je výstup.

Naivní Bayesovský klasifikátor

Naivní Bayes je jednoduchý algoritmus, který klasifikuje text na základě pravděpodobnosti výskytu událostí.

Vzorec pro výpočet pravděpodobnosti klasifikace daného vzorku x do třídy C vychází z Bayesova pravidla:

$$P(C \mid x) = \frac{P(x \mid C) \cdot P(C)}{P(x)}$$

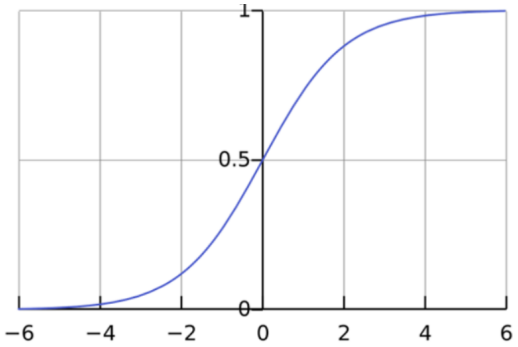
Kde $P(C \mid x)$ je podmíněná pravděpodobnost, že vzorek x patří do třídy C. $P(x \mid C)$ je pravděpodobnost, že vzorek x bude pozorován v třídě C. $P(C)$ je apriorní pravděpodobnost třídy C. A $P(x)$ je pravděpodobnost, že vzorek x bude pozorován. [3]

Logistická regrese

Dalším pravděpodobnostním algoritmem používaným jako klasifikátor je **Logistická regrese**. Patří mezi regresní modely a využívá k predikci pravděpodobnosti přítomnosti třídy logistickou funkci jako rovnici mezi x a y. Logistická funkce mapuje y jako *sigmoid* funkci x:

$$\frac{1}{1 + e^{-x}}$$

Pokud vykreslíme rovnici logistické regrese, bude vypadat jako na obrázku 2.5 na straně 24. Je zde vidět, že vrací pouze hodnoty mezi 0 a 1.

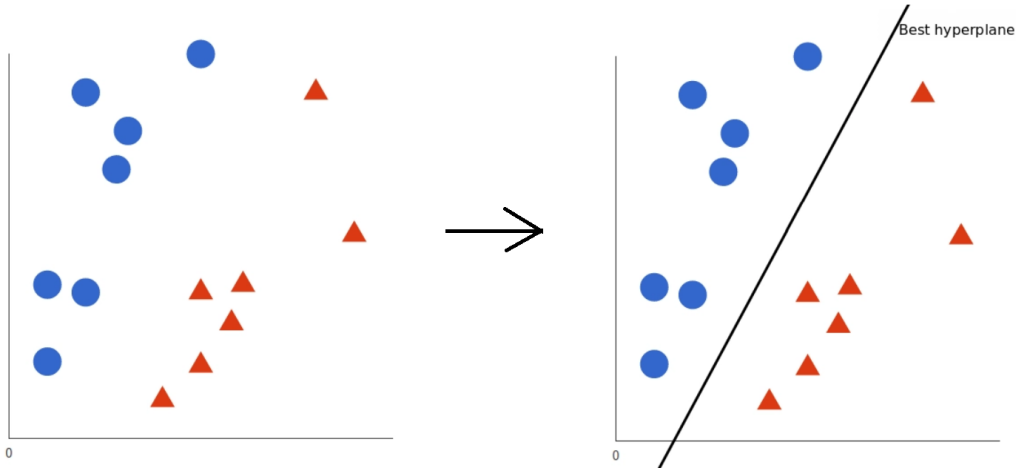


Obrázek 2.5: Vykreslená rovnice logistické regrese, zdroj: [2]

Metoda podpůrných vektorů

Metoda podpůrných vektorů (z anglického *Support vector machines, dále jen SVM*) je rozhodovací algoritmus. Cílem algoritmu je určit nejlepší rozhodovací hranici, která rozděluje vektory mezi ty, které do dané třídy patří a které ne. *SVM* se rozděluje na lineární a nelineární

Základní princip **Lineární SVM** je vidět na obrázku 2.6 na straně 25, kde na levé straně je prostor s modrými a červenými objekty a na pravé straně je výsledek po nalezení nejlepší hranice mezi barevnými objekty. *SVM* se snaží najít hranici tak, aby byla co nejdále od nejbližšího modrého a zároveň i červeného objektu.

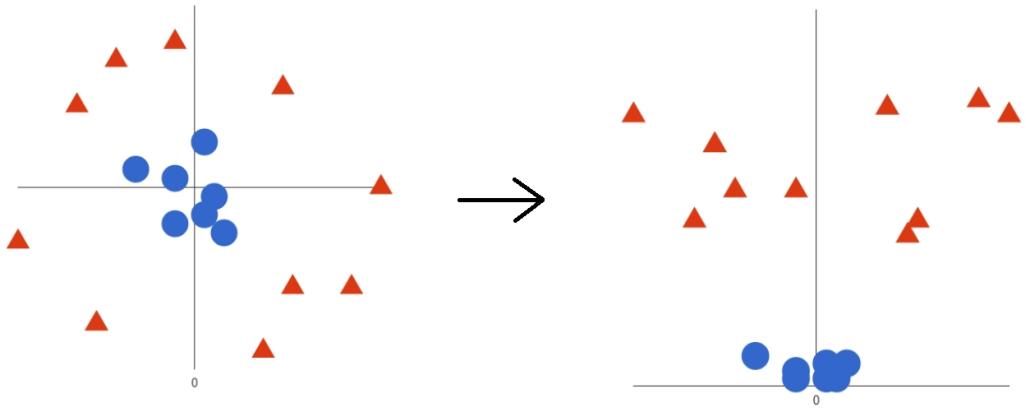


Obrázek 2.6: Nalezení nejlepší hranice (*best hyperplane*) mezi modrými a červenými objekty, zdroj: [7]

Často se stává, že data nejsou lineárně separovaná jako v předchozím příkladu. Tento problém řeší **Nelineární SVM** pomocí *jádrového triku* (z anglického *The kernel trick*). Smysl je v tom, že se provede transformace dat z lineárního prostoru do prostoru vyšší dimenze, kde problém není. Taková transformace může být výpočetně nákladná z důvodu mnoha dimenzí s komplikovanými výpočty. Výpočet pomocí *jádrového triku* nepočítá s vektory, ale pouze se skalárním součinem mezi vektory. Využívá se tak *jádrová funkce* (z anglického *kernel function*). Výběr *jádrové funkce* závisí na tom, jak vypadají data. Pokud je to možné, bývá nejvhodnější použít *Lineární jádrovou funkci*, protože je jednoduchá a rychlá na výpočet. [7]

K dalším nejpoužívanějším jádrovým funkcím patří *Polynomiální jádrová funkce*, *Gaussovská jádrová funkce* a *Sigmoidální jádrová funkce*. [28]

Na obrázku 2.7 na straně 26 je vidět nalezení nového prostoru v případě nelineárně separovaných dat.



Obrázek 2.7: Nalezení nejlepší hranice (*best hyperplane*) mezi modrými a červenými objekty v novém prostoru, zdroj: [7]

Ve srovnání s novějšími algoritmy, jako jsou neuronové sítě, má *SVM* dvě velké výhody: vyšší rychlost a lepší výkon do omezeného počtu vzorků. Algoritmus tedy může být vhodnější volbou pro klasifikaci textu, pokud je datová sada velká jen v řádech tisíců vzorků. [7]

Neuronové sítě

Umělá neuronová síť (ANN – z anglického *Artificial neural networks*) je model inspirovaný strukturou biologického mozku, který se používá v oblasti strojového učení a umělé inteligence. Umělé neurony se zde nazývají uzly a jsou shlukovány dohromady v několika vrstvách. Tyto uzly působí navzájem paralelně. Jsou schopné zpracovávat vstupní data a vytvářet komplexní modely pro řešení různých úloh. Když uzel obdrží numerický signál, zpracuje ho a odešle signál ostatním neuronům, které jsou s ním spojené. Stejně jako v lidském mozku, tak i u umělých neuronů má posílení nervů za následek lepší rozpoznávání vzorů, lepší odbornost a lepší celkové učení. [6]

Neuronové sítě můžeme na základě síťových topologií rozdělit na dopředné neuronové sítě a rekurentní nebo rekurzivní neuronové sítě.

Dopředná neuronová síť

Dopřednou neuronovou síť (z anglického *Feedforward neural network*, dále jen *FFNN*) bych rád ukázal na příkladu. Příklad je na obrázku 2.8 na straně 28. Na příkladu jsou tři vrstvy: L_1 je *vstupní vrstva*, která odpovídá

vstupnímu vektoru *(*x_1, x_2, x_3*)* a termu *+1*. Tato vrstva obsahuje elementy vstupního vektoru. Druhou vrstvou L_2 je *skrytá vrstva*, jejíž výstup není viditelný jako výstup sítě. Poslední vrstvou je *výstupní vrstva* L_3, která odpovídá výstupnímu vektoru S_1. Vrstvy L_2 a L_3 obsahují základní výpočetní prvky – neurony. Šipky mezi neurony reprezentují tok v síti. Každé takové spojení má určitou váhu, jejíž hodnota řídí signál mezi neurony. Neuronová síť se učí úpravami těchto vah pomocí informací, které mezi neurony protékají. Neurony v každé vrstvě přečtou informace z předchozích neuronů, ty zpracují a vygenerují výstup pro neurony v další vrstvě. Výsledná forma hypotéz $h_{w,b}(x)$ odpovídá po natrénování výstupním datům.

Ve skryté vrstvě neurony přijímají vstup od x_1, x_2, x_3 a *+1* termu a generují hodnotu $f(W^t x) = f\left(\sum_{i=1}^{3} W_i x_i + b\right)$ aktivační funkce *f*, kde W_i jsou váhy jednotlivých spojení elementů, *b* je *bias*. Funkce *f* bývá buď *sigmoid* funkce a *hyperbolická tangens* funkce nebo *Rectified Linear Unit* funkce (ReLU).
Rovnice funkce *sigmoid*:

$$f(W^t x) = sigmoid(W^t x) = \frac{1}{1 + exp(-W^t x)}$$

Rovnice funkce *hyperbolický tangens*:

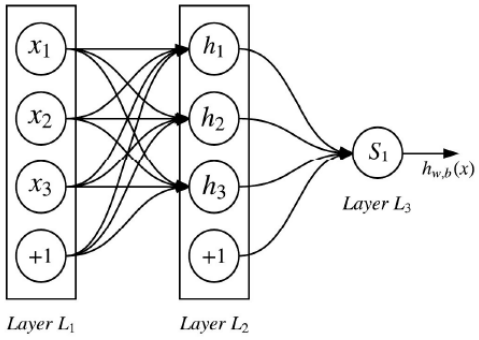
$$f(W^t x) = tanh(W^t x) = \frac{e^{W^t x} - e^{-W^t x}}{e^{W^t x} + e^{-W^t x}}$$

Rovnice funkce *ReLU*:

$$f(W^t x) = ReLU(W^t x) = max(0, W^t x)$$

Funkce *sigmoid* výsledné číslo vsune do hodnoty mezi 0 a 1, kde 0 znamená, že prvek do třídy určitě nepatří a 1 znamená, že prvek do třídy určitě patří. Funkce *hyperbolického tangensu* má rozsah -1 až 1. U funkce *ReLU* jsou všechny hodnoty nižší než 0, mapovány na 0. Funkce *ReLU* bývá oblíbená z důvodu snadného výpočtu a dobrého výkonu. V poslední vrstvě může být použita funkce *softmax* pro finální klasifikaci. Tato funkce je zobecněním logistické funkce a mapuje N-rozměrný vektor *X* libovolných reálných hodnot na N-rozměrný vektor $\sigma(X)$ skutečných hodnot v rozsahu 0 až 1.
Rovnice funkce *softmax* je následující:

$$\sigma(X)_j = \frac{e^{x_j}}{\sum_{n=1}^{N} e^{x_n}}, proj = 1, ..., n$$



Obrázek 2.8: Příklad *Dopředné neuronové sítě*, zdroj: [31]

Obvyklým postupem při trénování neuronové sítě je *stochastický gradientový sestup* prostřednictvím zpětného šíření (z anglického *backpropagation*), aby se minimalizovala ztráta *cross-entropy*, což je ztrátová funkce (z anglického *loss-function*) pro výstup funkce *softmax*. Nejprve jsou vypočítány gradienty ztrátové funkce vzhledem k vahám od poslední skryté vrstvy k výstupní vrstvě a poté jsou rekurzivně vypočítány gradienty výrazů vzhledem k vahám mezi horními vrstvami sítě – aplikací řetězového pravidla zpětným způsobem. S těmito gradienty se odpovídajícím způsobem upraví váhy mezi vrstvami. Jedná se o iterativní proces zpřesňování a trvá, dokud nejsou splněna určitá kritéria pro ukončení. [31]

Konvoluční neuronová síť

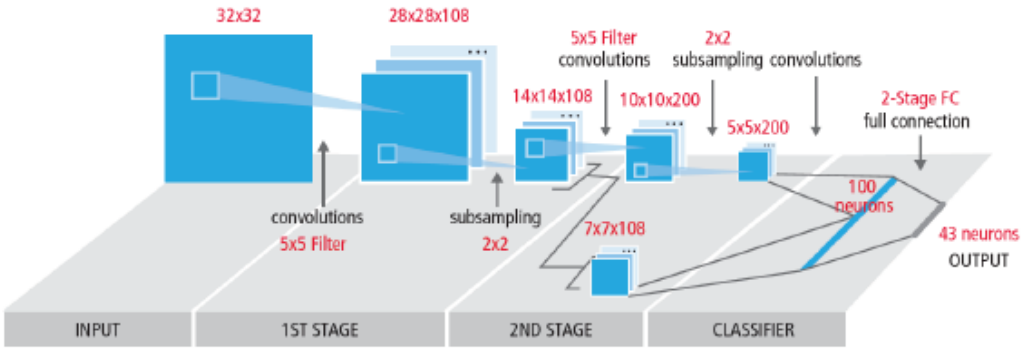
Speciálním typem *FFNN* je **Konvoluční neuronová síť** (z anglického *Convolutional neural network*, dále jen *CNN*). Tento model je inspirován zrakovou kůrou v mozku. Zraková kůra obsahuje mnoho buněk, pomocí kterých mozek detekuje světlo v malých oblastech zorného pole. Tyto buňky slouží jako filtry před vstupním prostorem. *CNN* se skládá z několika konvolučních vrstev, které stejně jako buňky mají funkci filtru.

Zde je pro ukázku a snazší pochopení příklad *CNN* na rozeznání dopravních značek na obrázku 2.9 na straně 29.

Vstupem *CNN* je obrázek o šířce 32 pixelů, výšce také 32 pixelů a jedním vstupním kanálem, tedy(32x32x1). V první fázi (stage) se použije filtr o velikost (5x5x1) ke skenování obrazu. Každá oblast ve vstupním obrázku, na kterou filtr promítá, je receptivní pole. To, čemu se říká filtr, je ve skutečnosti pole vah. Jak filtr konvoluje (neboli se posouvá), násobí hodnoty vah s původními hodnotami pixelů obrázku. Výsledná čísla z násobení se sečtou do jednoho čísla, které reprezentuje receptivní pole. Když filtr dokončí skenování obrázku, získáme *aktivační mapu* o velikosti (28x28x1). V *CNN*

se běžně používá hned několik různých filtrů. V tomto příkladu se aplikuje celkem 108 filtrů a po první fázi tedy máme 108 aplikačních map. Tím končí první konvoluční vrstva.

Po konvoluční vrstvě se většinou používá vrstva *podvzorkování* (z anglického *subsampling* nebo *pooling*). Vrstva *podvzorkování* postupně zmenšuje prostorovou velikost reprezentace. Snižuje počet prvků a výpočetní složitost sítě, přesto zachovává nejdůležitější informace. Tato vrstva je použita i zde a zmenšuje rozměry na (14x14x108). Výstup první fáze je zároveň vstupem fáze druhé, kde se použijí nové filtry o velikosti (5x5x108), kde 108 je počet aktivačních map v předchozí fázi. V poslední fázi *CNN* už používá plně propojenou vrstvu a pro klasifikaci funkci *softmax* s výstupními třídami. [31]



Obrázek 2.9: Příklad *Konvoluční neuronové sítě*, zdroj: [31]

Rekurentní neuronová síť

Dalším typem neuronové sítě je **Rekurentní neuronová síť** (z anglického *Recurrent neural network*, dále jen *RNN*). Nazývá se *rekurentní*, protože spojení mezi neurony tvoří řízený cyklus. Výhodou *RNN* je, že může používat vnitřní paměť na zpracování sekvence vstupů. Tím se myslí, že provádí stejný úkol pro každý prvek sekvence tak, že jeho výstup je závislý na předchozích výpočtech.

RNN má tolik časových kroků, kolik je délka vstupu (počet slov). Příklad *RNN* s třemi časovými kroky je na obrázku 2.10 na straně 30, kde x_t je vstupní vektor v časovém kroku t. Skrytý stav v časovém kroku t je označen h_t a je vypočítán aktivační funkcí na základě předchozího skrytého stavu a vstupu v aktuálním časovém kroku:

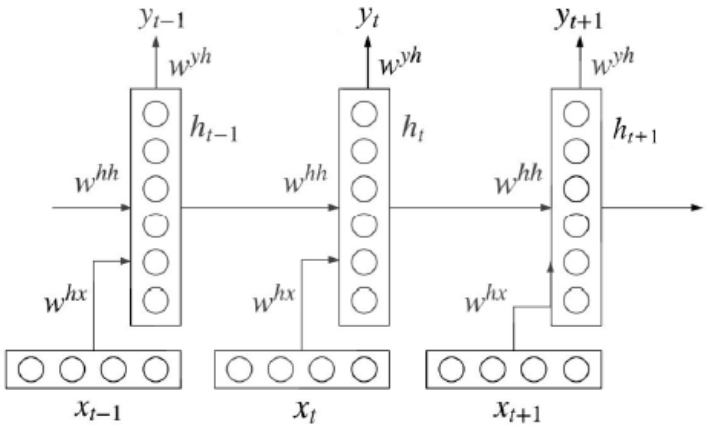
$$h_t = f\left(w^{hh}h_{t-1} + w^{hx}x_t\right)$$

kde aktivační funkce f je většinou *hyperbolický tangens* nebo *ReLU* funkce, w^{hx} je matice vah k podmínění vstupu x_t a w^{hh} je matice vah, která se pou-

žívá k podmínění předchozího skrytého stavu h_{t-1}. y_t je pravděpodobnostní rozdělení ve slovní zásobě v kroku t. Pokud bychom chtěli predikovat slovo ve větě, toto by byl vektor pravděpodobností napříč slovní zásobou. Výpočet y_t se vypočítá pomocí funkce *softmax*:

$$y_t = softmax\Big(w^{yh} h_t\Big)$$

Právě skrytý stav h_t je nazýván jako paměť sítě, která uchovává informace o tom, co se stalo v předchozích krocích. Zatímco *FFNN* používá jiné parametry pro každou vrstvu, *RNN* má stejné parametry ve všech krocích sítě. Mění se jen vstupy. *RNN* se může podívat zpět v krocích kvůli „mizejícímu gradientu“ a „explodujícímu gradientu“ jen v omezeně dlouhých sekvencích. [31]



Obrázek 2.10: Tři časové kroky příkladu *Rekurentní neuronové sítě*, zdroj: [31]

Neuronová síť s dlouhodobou a krátkodobou pamětí

Speciálním typem *RNN* je **Neuronová síť s dlouhodobou a krátkodobou pamětí** (z anglického *Long short term memor*, dále jen *LSTM*) která byla navržena pro řešení problémů sekvencí a časových řad. Hlavním rozdílem mezi *LSTM* a běžnými *RNN* je, že *LSTM* má schopnost uchovávat informace po dlouhou dobu, aniž by došlo k degradaci kvality těchto informací. Oproti *RNN* má místo jedné vrstvy čtyři a také má dva stavy – skrytý stav a stav buňky, ale stejně jako *RNN* obsahuje časové kroky. Stav buňky obsahuje paměťovou buňku, která uchovává informace o historii vstupů sítě a zapomínání informací. Skrytý stav je výstupem *LSTM* vrstvy a obsahuje informace o stavu buňky, která je zpracována pomocí transformací v rámci

vrstvy. *LSTM* obsahuje celkem tři brány – *brána zapomínání* (z anglického *forget gate*), *vstupní brána* (z anglického *input gate*) a *výstupní brána* (z anglického *output gate*). Brány jsou části paměťové buňky, což je struktura, která uchovává po určitou dobu informace. Brány řídí tok informací z paměťové buňky a do paměťové buňky.

Příkladem této neuronové sítě je obrázek 2.11 na straně 32, kde stav buňky je označen *C*, skrytý stav je označen *h* a vstupní data jsou označená *x*. *LSTM* se v časovém kroku *t* pomocí funkce *sigmoid* nejprve rozhoduje, jaké informace se ze stavu buňky vypíší. Funkce použije výstup z předchozí skryté vrstvy h_{t-1} a aktuální vstup x_t. Výstupem je číslo od 0 do 1, kde 0 znamená „úplně zapomenout“ a 1 znamená „úplně zachovat“. Toto se děje v bráně, která se nazývá *brána zapomínání*.
Výpočet *sigmoid* funkce je následující:

$$f_t = \sigma\Big(W^f x_t + U^f h_{t-1}\Big)$$

Následně se rozhodne, jaké nové informace do stavu buňky uloží. Toto rozhodování má dvě části. První část se děje ve *vstupní bráně*. Zde se opět podle funkce *sigmoid* rozhoduje, které hodnoty *LSTM* aktualizuje. Druhým krokem je funkce *hyperbolického tangensu*, která vytvoří vektor nových hodnot C_t. Vektor je následně přidán do stavu buňky. Kombinací těchto dvou částí se aktualizuje stav. Po těchto krocích se aktualizuje stav předchozí buňky C_{t-1} na aktuální stav C_t pomocí rovnice:

$$C_t = f_t \quad C_{t-1} + i_t \quad C_t$$

Výpočet funkce *sigmoid* v první části:

$$i_t = \sigma\Big(W^i x_t + U^i h_{t-1}\Big)$$

Výpočet funkce *hyperbolického tangensu* v druhé části:

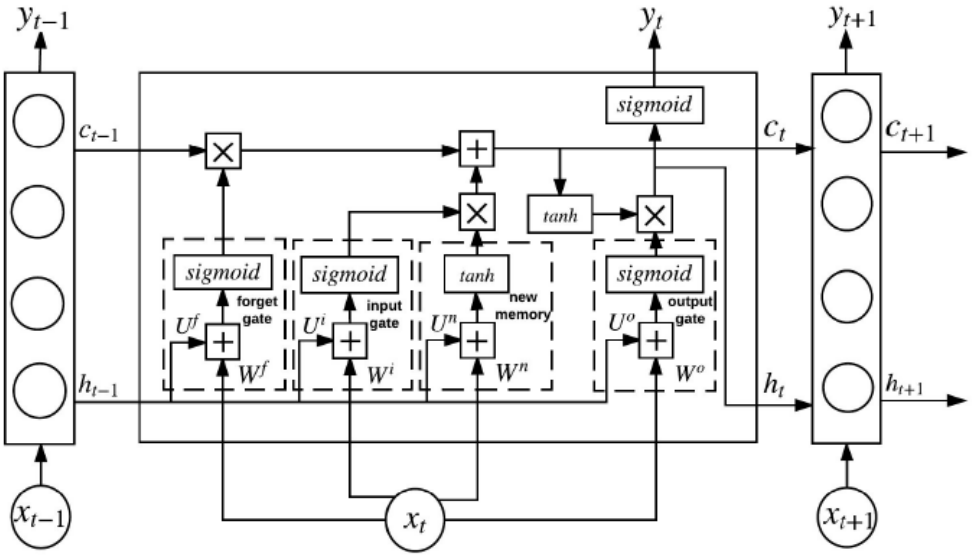
$$C_t = tanh\Big(W^n x_t + U^n h_{t-1}\Big)$$

V poslední části *LSTM* se rozhoduje o výstupu. Ten je závislý na stavu buňky. Ve *výstupní bráně* se pomocí funkce *sigmoid* rozhodne, která část informace projde na výstup. Poté se na stav buňky použije funkce *hyperbolický tangens* a vynásobí s výsledkem funkce *sigmoid*. Výsledek je výstupem buňky. [31]
Rovnice funkce *sigmoid* ve *výstupní bráně*:

$$o_t = \sigma\Big(W^o x_t + U^o h_{t-1}\Big)$$

Rovnice funkce *hyperbolický tangens* na stav buňky:

$$h_t = o_t \quad tanh(C_t)$$



Obrázek 2.11: Příklad *Neuronové sítě s dlouhodobou a krátkodobou pamětí*, zdroj: [31]

3 Vícejazyčné zpracování slovních vektorů

Vícejazyčné zpracování slovních vektorů (z anglického *cross-lingual word embeddings*) se využívá pro nalezení podobností mezi vektory slov v různých jazycích. Tím je možné převést význam slova z jednoho jazyka do druhého. Tato technika se používá ve strojovém učení a oblasti jazykových technologií, jako je překlad mezi jazyky, *multilingvní* klasifikace (trénování modelu klasifikace textu umožňující pracovat s více různými jazyky), analýza sentimentu pro různé jazyky, vyhledávání informací v různých jazycích nebo generování vícejazyčného obsahu. [22]

Použití modelů vícejazyčných slovních vektorů má dva hlavní důvody. Prvním důvodem je porovnání významu slov napříč různými jazyky. To je často důležité při strojovém učení nebo vyhledávání informací. Druhým aspektem je přenos modelu mezi jazyky poskytnutím společného prostoru, na příklad mezi jazyky s mnoha zdroji a jazyky, kde je zdrojů málo.

Modely se liší zejména s ohledem na data, která používají, v následujících dvou dimenzích:

- **Typ zarovnání** – Tyto metody využívají různé typy dvojjazyčných supervizních signálů (dat, která metoda vyžaduje, aby se naučila zarovnat prostor pro reprezentaci napříč jazyky), které zavádějí silnější nebo slabší supervizi.
- **Porovnatelnost** – Tyto metody vyžadují buď paralelní zdroje dat (přesné překlady do různých jazyků) nebo srovnatelná data, která jsou si nějakým způsobem podobná.

Jsou tři různé typy zarovnání, které jsou používány různými metodami. Jde o typické zdroje dat pro paralelní i porovnatelná data na základě následujících typů zarovnání: [25]

- **Zarovnání slov** – Většinou se můžeme setkat s paralelně zarovnanými daty ve formě dvojjazyčného slovníku, kde je vždy slovo z jednoho jazyka a jeho překlad v jazyce druhém. Paralelní zarovnání slov lze také získat automatickým zarovnáním slov v paralelním korpusu, který lze použít k vytvoření dvojjazyčných slovníků. Paralelní korpus obsahuje stejné texty v různých jazycích zobrazené vedle sebe.

- **Zarovnání vět** – K zarovnání vět je potřeba paralelní korpus. Metody většinou využívají *Europarl* korpus. Ten je nejčastějším zdrojem trénovacích dat u metod zarovnání vět. Obsahuje text zarovnaný do vět z jednání Evropského parlamentu.

- **Zarovnání dokumentů** – Paralelní data zarovnání dokumentů vyžadují zvlášť přeložené dokumenty v různých jazycích. Dobrým příkladem je webový portál *Wikipedia*, který obsahuje informační dokumenty v několika jazycích.

3.1 Modely zarovnání na úrovni slov

Paralelní data zarovnání na úrovni slov mají tři různé přístupy: [25]

- **Přístup založený na mapování** – Tento přístup nejdříve natrénuje jednojazyčné reprezentace slov na velkých jednojazyčných korpusech. Poté se pomocí těchto reprezentací snaží vytvořit transformační matici, která mapuje reprezentaci slov v jednom jazyce na reprezentaci druhého jazyka. Tuto transformaci se učí ze zarovnání slov nebo dvojjazyčných slovníků.

- **Přístup založený na pseudo-multijazyčných korpusech** – Tento přístup používá jednojazyčné vektorové reprezentační metody do automaticky vytvořených korpusů, které obsahují slova ze zdrojového i cílového jazyka. Může jít i o nějakým způsobem poškozené korpusy.

- **Metody spojení** – Metody berou paralelní text jako vstup a minimalizují jednojazyčné ztráty ve zdrojovém a cílovém jazyce.

Přístupy založené na mapování

Přístupy založené na mapování patří k nejoblíbenějším modelům díky jednoduchosti a snadnému použití. Jejich cílem je naučit se mapování z jednojazyčných vektorových prostorů do prostoru společného pro oba jazyky. Přístupy založené na mapování jsou čtyři: [25]

- **Regresní metody** – tyto metody mapují vektory zdrojového jazyka do cílového jazykového prostoru maximalizací jejich podobností

- **Ortogonální metody** – tyto metody mapují vektory zdrojového jazyka, aby maximalizovala jejich podobnost s vektory cílového jazyka, ale transformace je omezená ortogonalitou.

- **Kanonické metody** – tyto metody mapují vektory obou jazyků do nového sdíleného prostoru, což maximalizuje jejich podobnost.
- **Marginálni metody** – tyto metody mapují vektory zdrojového jazyka, aby se maximalizovala rezerva mezi správnými překlady

Regresní metody

Oblíbenou metodou mapování je *metoda lineární transformace*. Metoda vznikla díky pozorování, že vektory stejných slov v různých jazycích ukazují podobné geometrické útvary po použití správné lineární transformace. To dokazuje, že je možné transformovat vektorový prostor zdrojového jazyka *z* na vektorový prostor cílového jazyka *c* pomocí transformační matice $W^{z \to c}$, dále jen *W*.

Pomocí slovníku překladů *n* slov, kde $w_1^z, ..., w_n^z$ jsou slova ze zdrojového jazyka a $w_1^c, ..., w_n^c$ jsou slova z cílového jazyka, se vypočítá transformační matice W. Vypočítá se použitím stochastického gradientního sestupu tím, že se minimalizuje druhá mocnina euklidovské vzdálenosti (střední kvadratická chyba, z anglického *mean squared error*, zkratka *MSE*) mezi naučenou jednojazyčnou vektorovou reprezentací zdrojového slova x_i^z, která je transformována pomocí matice W, a jednojazyčnou vektorovou reprezentací jeho překladu x_i^c.
Výpočet *MSE*:

$$\Omega_{\text{MSE}} = \sum_{i=1}^{n} ||Wx_i^z - x_i^c||^2$$

Vztah může být zapsán i ve formě matic:

$$\Omega_{\text{MSE}} = ||WX^z - X^c||_F^2$$

kde X^z je matice vektorů slov zdrojového jazyka a X^c je matice vektorů slov cílového jazyka. Pokud na rovnici budeme nahlížet jako na řešení nejmenších čtverců (z anglického *least squares solution*) v systému lineárních rovnic můžeme odvodit vztah:

$$WX^z = X^c$$

Nyní pokud převedeme X^z na pravou stranu, dostaneme:

$$W = (X^{zT}X^z)^{-1}X^{zT}X^c$$

kde X^{zT} je transponovaná matice vektorů zdrojových slov. [25]

Ortogonální metody

Regresní metodu je možné vylepšit tak, že omezíme transformaci *W* na ortogonální. To znamená, že bude platit vztah:

$$W^T W = I$$

kde *I* je jednotková matice a W^T transponovaná matice *W*. Řešením **ortogonální metody** je rovnice:

$$W = VU^T$$

která se dá vypočítat pomocí jednotkového rozkladu matice (z anglického *Singular Value Decomposition*, zkratka *SVD*), pro který platí:

$$A = U\Sigma V^T$$

kde *A* je rozkládaná matice, *U* je *m*m* ortogonální matice, Σ je *m*n* diagonální matice s nezápornými prvky a V^T je transponovaná *n*n* ortogonální matice *V* [29]. Pro výpočet transformační matice *W* získáme transponováním matic V^T a *U* matice *V* a U^T z rozkladu: [25]

$$X^{cT} X^z = U\Sigma V^T$$

Kanonické metody

Kanonické metody používají pro mapování *Kanonickou korelační analýzu* (z anglického *Canonical Correlation Analysis*, dále jen CCA). Lineární projekce vytváří pouze jednu transformační matici $W^{z\rightarrow c}$ pro transformaci vektorového prostoru zdrojového jazyka do prostoru cílového jazyka. Naproti tomu *CCA* vytváří transformační matici pro zdrojový jazyk $W^{z\rightarrow}$ a transformační matici pro cílový jazyk $W^{c\rightarrow}$. Promítne tak pomocí transformačních matic oba prostory do nového společného prostoru, který je odlišný od obou prostorů.

Korelaci mezi projektovanými vektory zdrojového jazyka $W^{z\rightarrow} x_i^z$ a jeho odpovídajícími vektory projektovaného cílového jazyka $W^{c\rightarrow} x_i^c$ můžeme zapsat následovně:

$$\rho(W^{z\rightarrow} x_i^z, W^{c\rightarrow} x_i^c) = \frac{\mathrm{cov}(W^{z\rightarrow} x_i^z, W^{c\rightarrow} x_i^c)}{\sqrt{\mathrm{var}(W^{z\rightarrow} x_i^z)\mathrm{var}(W^{c\rightarrow} x_i^c)}}$$

kde *cov* je kovariance a *var* je variance. Následně se *CCA* snaží maximalizovat korelaci (mezi projektovanými vektory $W^{z\rightarrow}x_i^z$ a $W^{c\rightarrow}x_i^c$ pomocí vzorce: [25]

$$\Omega_{\text{CCA}} = -\sum_{i=1}^{n} \rho(W^{z\rightarrow}x_i^z, W^{c\rightarrow}x_i^c)$$

4 Vytvoření testovacího korpusu

V této práci je velký rozdíl od spousty podobných úloh zaměřených na klasifikaci sentimentu v tom, že jsem nemohl použít žádná hotová data. Jak bylo psáno v kapitole Úvod, má tato práce praktický význam. Její koncept je restaurace, kde je málo dat v češtině a v zájmu je pouze několik specifických kategorií, u kterých nás zajímá sentiment, tedy zda má text pozitivní nebo negativní emocionální tón (více v sekci Analýza sentimentu).

Vytváření vlastního datového korpusu je ojedinělá záležitost a velmi zajímavá část této práce. Běžným procesem při trénování klasifikace v různých oblastech a úlohách je využití hotových datových korpusů. Příkladem takového korpusu, který by se dal pro tuto práci použít je **SemEval-2016 Task 5**, kde jednou z domén, na které se zaměřuje, je klasifikace recenzí restaurací. Tento korpus obsahuje celkem 39 datasetů s tím, že 19 jich je určeno k trénování a 20 k testování. Možnost použití je v celkem 8 jazycích, mezi které patří angličtina i němčina, ale čeština se zde nevyskytuje. Kromě recenzí na restaurace je zde dalších 6 různých domén. Jeho součástí je 70790 manuálně označkovaných *ASBA* (*Aspect-Based Sentiment Analysis*, česky Analýza sentimentu založená na aspektech). *ASBA* je úloha zpracování přirozeného jazyka, jejímž cílem je identifikovat a extrahovat sentiment specifických aspektů nebo součástí produktu nebo služby. Z těchto *ASBA* bylo 47654 označkovaných větných celků pro všech 8 jazyků a dále 7 domén a 23136 označkovaných plných textů pro 6 jazyků a 3 domény včetně domény restaurace. Čísla vypadají hezky, ale pokud se zaměříme pouze na označkované texty v anglickém jazyce pro doménu restaurace, zjistíme, že je v tomto korpusu pouze 350 textů pro trénování a 90 pro testování. To je pro experimenty velice málo. Na druhou stranu se dá očekávat, že kvalita těchto dat bude výrazně vyšší než data, které jsem vytvořil sám. Označkování těchto dat má několik fází a pracovali na nich mnohem zkušenější lidé včetně odborných lingvistů. [24]

Cílem této praktické části je tedy vyhledat hrubá data a tato data následně označkovat, aby se dala použít pro trénování a testování klasifikačního modelu.

4.1 Získání dat

Mým prvním úkolem bylo připravit hrubá data z recenzí vybraného řetězce na *Google Reviews*. Vybraným řetězcem je jeden z největších světových řetězců rychlého občerstvení *McDonald's*. Důvodem je jeho oblíbenost a rozšířenost v celém světě. Díky tomu by mělo být snadné získat velké množství recenzí ve více jazycích.

Data jsem vyhledal a získal ve formátu *json* pomocí služby *Google Maps Reviews Scraper* platformy **Apify**. Platforma Apify se používá k extrakci dat z webových stránek, automatizací internetových procesů a zpracováním dat pomocí umělé inteligence. [1]

Díky službě *Google Maps Reviews Scraper* stačí zadat url adresu místa na *google* mapách, zadat upřesňující parametry a získat tak datovou sadu recenzí. Tímto způsobem jsem získal hrubá data recenzí řetězce *McDonald's* pro jazyk český, anglický a německý.

4.2 Označkování dat

Aby data byla použitelná pro trénování klasifikátoru a následné testování, je nutné data označkovat. Označkováním dat se rozumí přiřadit ke každé recenzi emocionální tón pro každou kategorii. Emocionální tón, neboli sentiment, rozlišuji v této práci na pozitivní a negativní. U sentimentu recenzí se často používá i neutrální emocionální tón, ale vzhledem k malému počtu výskytů potenciálně neutrálních recenzí pro jednotlivé kategorie jsem jej nezavedl. Nebyl by dostatek dat pro dostačující trénování a testovaní neutrálního tónu a klasifikace by neměla takový význam.

Velikost trénovacích dat bývá výrazně vyšší než u dat testovacích. Protože korpus českých recenzí slouží jako primární testovací data, má celkem 3000 označkovaných recenzí. Korpusy pro anglické a německé recenze mají každý 500 označkovaných recenzí. Obecně platí, že s větším počtem dat roste i přesnost určení sentimentu. U takového počtu dat nelze očekávat extrémně velkou přesnost, ale i tak by měl stačit na úspěšné detekování většiny recenzí.

Doba trvání značkování je různá a záleží na zkušenosti se značkováním i na zkušenosti se samotným textem. Po každé označené recenzi se tak značkování nepatrně zrychluje. U cizích jazyků značkování trvá déle kvůli nutnému překladu. Má průměrná doba označení 100 recenzí byla přibližně jedna hodinu času. Využitím transformace jsem tedy ušetřil přibližně 25 hodin pro každý cizí jazyk, které bych strávil značkováním recenzí.

4.2.1 Popis kategorií

Kategorie, které budou značkovány, jsem vybral na základě vzorku o několika recenzích. Označkování recenzí bylo rozděleno do devíti kategorií: *General* (obecné), *Food* (jídlo), *Drink* (nápoje), *Sta* (obsluha), *Speed* (rychlost), *Cleanness* (čistota), *Prices* (ceny), *Environment* (prostředí) a *Occupancy* (obsazenost). Text recenze může patřit do několika kategorií níže popsaných. Text musí vždy obsahovat alespoň jednu kategorii.

General

Text spadá do kategorie **General**, pokud je hodnocena restaurace jako celek. Jde o obecný pohled na návštěvu recenzenta.

Příklady textu klasifikovaného do kategorie **General**:

- **positive** – Super. ; Ok. ; V celku spokojenost. ; Doporučuji.
- **negative** – Sem už nikdy. ; Velké zklamání. ; Za mě už nikdy. ; Nedoporučuji.

Food

Do kategorie **Food** patří text, který hodnotí jídlo. Tedy chuť, kvalitu, čerstvost, teplotu, množství a podobně.

Příklady textu klasifikovaného do kategorie **Food**:

- **positive** – S jídlem si dal někdo práci. ; Pochutnal jsem si ; Kvalita a chutnost
- **negative** – S jídlem to nemá nic společného. ; Nechutné žrádlo. ; Přesolené hranolky

Drink

U kategorie **Drink** platí kategorizace podobně jako u kategorie *Food*. Tedy hodnocení chuti, kvality, teploty a množství. Ale jedná se o nápoje.

Příklady textu klasifikovaného do kategorie **Drink**:

- **positive** – Mají dobrou kávu ; Colu tu mám nejradši
- **negative** – Ani pití se nedá pít. ; Káva je studená ; Nejhorší kafe, co jsem kdy pila

Staff

Kategorie **Staff** je v textu označena, pokud se hodnotí personál restaurace. Hodnotí se jeho schopnosti, rychlost i povaha. Rychlost personálu patří do kategorie *Sta* i *Speed*.
Příklady textu klasifikovaného do kategorie **Staff**:

- **positive** – Vynikající obsluha. ; Rychlost vyřízení objednávek. ; Sympatický a usměvavý personál. ; Rychlá a příjemná obsluha
- **negative** – Popletli mi objednávky. ; Dnešní večerní směna katastrofa. ; Nepříjemná obsluha

Speed

Pokud se v textu hodnotí rychlost, ať už rychlost objednávek nebo rychlost personálu, je označena kategorie **Speed**.

Příklady textu klasifikovaného do kategorie **Speed**:

- **positive** – Velmi rychle se odbavíte. ; Klasika rychlého jídla. ; Rychlá obsluha
- **negative** – Poslední dobou jim to trvá. ; Extrémně pomalé DriveThru ; Dneska jim to trvalo 30 minut

Cleanness

Kategorie **Cleaness** je označena, pokud se v textu hodnotí čistota restaurace a okolí. Může jít o čistotu podlahy, nepořádek v kuchyni, drobky na stole, vynesení odpadkových košů i čistotu toalet.

Příklady textu klasifikovaného do kategorie **Cleanness**:

- **positive** – Poměrně čisto ; Chválím toaletu, byla čistá. ; Hezky uklizeno
- **negative** – Často velmi špinavé ; Odpadkových košů by mohlo být víc. ; Všude jsou drobky a odpadky.

Prices

Pokud se v textu hodnotí ceny restaurace, patří text do kategorie **Prices**.

Příklady textu klasifikovaného do kategorie **Prices**:

- **positive** – Za pár kaček dostaneš dobré jídlo. ; Dobré ceny
- **negative** – Vyšší cena za jídlo. ; Předražené.

Environment

Hodnocení prostředí patří do kategorie **Environment**. Může jít o stížnosti na hluk nebo ocenění vnitřní i okolní části restaurace.

Příklady textu klasifikovaného do kategorie **Environment**:

- **positive** – Tady je jedno z hezčích prostředí. ; Po rekonstrukci krásné místo. ; Moderně zařízené
- **negative** – Uprostřed hlučné a přecpané jídelny. ; Zastaralý interiér

Occupancy

Do kategorie *Occupancy* náleží hodnocení množství lidí v restauraci. Pokud je málo lidí, jde o pozitivní hodnocení a pokud jde o velké množství lidí, jde o hodnocení negativní.

Příklady textu klasifikovaného do kategorie **Environment**:

- **positive** – Málo lidí, klid. ; Všude spousta místa ; Dneska ani noha
- **negative** – Večer je tu nacpáno. ; Velké fronty ; Každý den plno

4.2.2 Ověření správnosti značkování

Každá osoba má jiný pohled na věc a trochu odlišné vnímání textu. U značkování dat může z tohoto důvodu jedna osoba vidět v textu obsaženou jednu z kategorií a druhá ne. Něco takového samozřejmě může způsobit komplikace. Příkladem je věta, která se objevuje v recenzích restaurací: „Je to tu nechutný!“. V této větě může recenzent vyjadřovat, že se mu v restauraci nelíbí a šlo by o kategorii *General*, nebo tím chce poukázat na to, že v restauraci nemají dobré jídlo. Zde se může označit jedna, druhá nebo obě varianty. Pokud data značkuje pouze jeden člověk, jsou data sice jednotná, ale pokud označkovaná data nebo výsledek klasifikace zpracovává někdo jiný, může dojít k chybám. Řešením je zobecnit tento pohled vnímání textu tím, že se porovná označkování více lidí na stejném textu a následně se sjednotí odlišnosti.

Na 100 recenzích jsme já a vedoucí této práce označkovali text podle kategorií a jejich popisu. Poté jsem vyhodnotil výsledky. Vypočítané metriky

porovnání jsou v tabulce 4.1 na straně 44. V pravé části tabulky je navíc vidět počet shodných označení (*hit*) a počet odlišných označení (*miss*) pro každou detekci kategorie i jejího sentimentu. Z tabulky je na první pohled poznat, že pokud se vyskytla nějaká odlišnost, šlo o detekci kategorie. Pokud jsme oba správně detekovali kategorii, u sentimentu jsme se vždy shodli. Nejvíce odlišných označení má kategorie *General* a to celkem 16. Důvodem je, že tato kategorie je hodnocením obecného pohledu na restauraci, proto má nejširší škálu možností, jak se vyjádřit k hodnocení. Různý pohled jsme měli na kategorii *Environment*, kde je *miss* 8. Klíčovou nejasností zde bylo hodnocení parkoviště, u kterého jsme neměli stejný názor, zda má do kategorie patřit. Ostatní kategorie byly v pořádku nebo odlišnosti nebyly zásadní.

Po diskuzi s vedoucím práce jsem na základě těchto výsledků upravil hodnotící kritéria značkování kategorií.

Jsou i části textu, kde chce zjevně recenzent něco pochválit, nebo naopak zkritizovat. Typ sentimentu by byl zjevný podle tónu mluveného slova, ale ne z textu. Příkladem je věta *„Hranolky stojí 99Kč“*. Zde jde o hodnocení ceny (kategorie *Prices*), ale zda jde o pozitivní nebo negativní sentiment už záleží na úhlu pohledu čtenáře a je těžké názor sjednotit.

4.2.3 Porovnání korpusů a kategorií

V každé kategorii každého jazyka se liší počet pozitivního a negativního sentimentu. V tabulce 4.2 na straně 45 je vidět četnost sentimentu v jednotlivých kategorií pro každý jazyk. Ve spodní části tabulky jsou celkové součty sentimentů. Vypočítané poměry sentimentu k počtu recenzí jsou v tabulce 4.3 na straně 46 i s celkovým poměrem pro daný jazyk. Díky těmto datům se dají vyčíst zajímavé předpoklady a relace napříč kategoriemi a jazyky. Data zároveň mohou vysvětlovat určité odchylky a nepřesnosti ve výsledcích klasifikace.

Průměrný počet detekovaných kategorií na jednu recenzi se ve všech jazycích pohybuje okolo 1.50 označení na recenzi s tím, že v anglických recenzích je poměr trochu vyšší (1.60). Kde se ale jazyky liší výrazněji, je sentiment. V českém jazyce znatelně převládá pozitivní sentiment (0.90 pozitivního sentimentu na recenzi) před negativním (0.62). To může znamenat, že čeští recenzenti mají ve zvyku hodnotit restauraci spíše kvůli chvále než kritice. U anglického a německého jazyka jsou poměry mezi sentimenty poměrně vyrovnané.

Kategorie **General** je pro český jazyk zastoupena nejvíce. Vyskytuje se v celkem 36% recenzích. V označkovaných recenzích má tato kategorie 75% pozitivních recenzí. Má jich tak nejvíce ze všech kategorií v českém jazyce. V

	Recall	Precision	Accuracy	F1-score	hit	miss
General	0.84	0.88	1.0	1.0	84	16
sentiment	1.0	1.0	1.0	1.0	28	0
Food	0.96	0.87	0.95	0.91	95	5
sentiment	1.0	1.0	1.0	1.0	24	0
Drink	1.0	1.0	1.0	1.0	100	0
sentiment	1.0	1.0	1.0	1.0	3	0
Staff	0.97	0.90	0.96	0.93	96	4
sentiment	1.0	1.0	1.0	1.0	27	0
Speed	0.97	0.93	0.96	0.95	96	4
sentiment	1.0	1.0	1.0	1.0	35	0
Cleaness	1.0	1.0	1.0	1.0	100	0
sentiment	1.0	1.0	1.0	1.0	5	0
Prices	1.0	1.0	1.0	1.0	100	0
sentiment	1.0	1.0	1.0	1.0	5	0
Environ.	0.8	0.8	0.92	0.80	92	8
sentiment	1.0	1.0	1.0	1.0	16	0
Occup.	0.86	0.86	0.98	0.86	98	2
sentiment	1.0	1.0	1.0	1.0	6	0

Tabulka 4.1: Četnost sentimentu v jednotlivých kategorií pro každý jazyk, jejich součty pro každý jazyk a poměr k počtu recenzí

anglickém jazyce má tato kategorie 28%, ale poměr pozitivních a negativních recenzí má velmi vyrovnaný (54% a 48%). U německého jazyka je také poměr této kategorie k počtu recenzí velký (32%) a opět zde převládá pozitivní sentiment (67%). U všech jazyků jde o vysoký poměr, což se u obecného hodnocení dá očekávat. U nás i v Německu se zdá tento způsob stravování oblíbenější.

Nejvíce zastoupenou kategorií pro anglický i německý jazyk je kategorie **Staff**. Ta má poměr 54% u anglického a 47% u německého jazyka. U češtiny je poměr pouze 27%. U všech jazyků je sentiment velmi rovnoměrný. Z toho se dá soudit, že v těchto zahraničních zemích více záleží na schopnostech a povaze personálu než u nás.

Naopak nejméně zmiňovaná kategorie je **Prices**. Tato kategorie má u češtiny 3%, angličtiny 2% a u němčiny dokonce jen 1%. Ceny tedy u hodnocení restaurace v řetězci *McDonald's* nejsou zajímavým kritériem. K méně zajímavým kategoriím pro všechny jazyky také patří **Drink, Cleaness, Environment** a **Occupancy**.

U více zastoupených kategorií je zajímavý rozdíl v sentimentu u katego-

rie **Speed**. Zatímco u českého i německého jazyka jsou poměry sentimentu vyrovnané, u anglického jazyka výrazně převládá ten negativní (72%). Důvodem může být v anglickém obecném pojmenování tohoto řetězce *„fast food“* (rychlé jídlo), kvůli kterému očekávají větší rychlost odbavení než u nás.

Druhá zajímavá odchylka sentimentu je u kategorie **Food**. Zde jsou poměry sentimentu vyrovnané u angličtiny a němčiny, ale u češtiny převládá pozitivní sentiment (69%). To může být způsobeno lepší kvalitou jídla nebo naopak nižším očekáváním.

	Čeština			**Angličtina**			**Němčina**		
	Pos	*Neg*	*Sum*	*Pos*	*Neg*	*Sum*	*Pos*	*Neg*	*Sum*
General	821	269	1090	76	64	140	108	54	162
Food	561	250	811	78	72	150	80	65	145
Drink	137	61	198	9	19	28	3	5	8
Staff	385	419	804	116	156	272	124	111	235
Speed	434	344	778	32	82	114	56	64	120
Cleaness	118	114	232	22	19	41	28	44	72
Prices	33	69	102	8	1	9	1	3	4
Environ.	145	21	166	15	6	21	16	3	19
Occup.	56	298	354	2	21	23	6	17	23
Součet	2690	1845	4535	358	440	798	422	366	758

Tabulka 4.2: Četnost sentimentu v jednotlivých kategorií pro každý jazyk s jejich součty

4.3 Problémy při vytváření korpusu

Díky tomu, že jsem osobně přečetl každou označkovanou recenzi mohu analyzovat rozdíly a případné problémy, které kvůli rozdílům mohou nastat. Každá země má trochu jinou kulturu a i to se podepíše na stylu psaní recenzí. Když porovnám styl psaní německých a anglických recenzí, tak u německých recenzí jsem se setkal s více přímým přístupem k vyjádření emocionálního tónu k hodnoceným kategoriím. Zatímco anglické recenze jsou více „upovídané“. To znamená, že v nich recenzenti často píší i hodně textu, který se přímo netýká hodnocených kritérií (na příklad, kam měli namířeno, než restauraci navštívili) nebo obsáhlý popis problému, kvůli kterému jsou nespokojení se službou. Tím vzniká množství slov, která nejsou relevantní směrem ke kategoriím a mohou ovlivnit kategorizaci. V českých recenzích

	Čeština			Angličtina			Němčina		
	Pos	*Neg*	*Cel*	*Pos*	*Neg*	*Cel*	*Pos*	*Neg*	*Cel*
General	0.75	0.25	0.36	0.54	0.46	0.28	0.67	0.33	0.32
Food	0.69	0.31	0.27	0.52	0.48	0.30	0.55	0.45	0.29
Drink	0.69	0.31	0.07	0.32	0.68	0.06	0.38	0.63	0.02
Staff	0.48	0.52	0.27	0.43	0.57	0.54	0.53	0.47	0.47
Speed	0.56	0.44	0.26	0.28	0.72	0.23	0.47	0.53	0.24
Cleaness	0.51	0.49	0.08	0.54	0.46	0.08	0.39	0.61	0.14
Prices	0.32	0.68	0.03	0.89	0.11	0.02	0.25	0.75	0.01
Environ.	0.87	0.13	0.06	0.71	0.29	0.04	0.84	0.16	0.04
Occup.	0.16	0.84	0.12	0.09	0.91	0.05	0.26	0.74	0.05
Celkem	0.90	0.62	1.51	0.72	0.88	1.60	0.84	0.73	1.52

Tabulka 4.3: Poměr sentimentu v jednotlivých kategoriích pro každý jazyk. *Pos* a *Neg* vyjadřují poměr sentimentu vůči celkovému počtu sentimentů v dané kategorii a *Cel* vyjadřuje poměr počtu detekce kategorie vůči počtu všech recenzí.

se nejvíce objevují „nicneříkající“ recenze, u kterých nelze označit žádnou kategorii.

Při vytváření korpusu jsem narazil na několik dalších problémů, které mohou ovlivnit analýzu sentimentu a stojí za zmínku. Jedním z problémů je různé použití slov a významů v různých jazycích. To může být způsobené kulturou i jazykem. Příkladem je pojmenování řetězce *McDonald's*. Zatímco v německých recenzích se velmi často vyskytuje správný název *„McDonald's“*, v českých recenzích se návštěvníci obrací spíše k hovornějšímu pojmenování *„Mekáč“* a v anglických používají obecné *„Fast food“*. Toto pojmenování v anglických recenzích navíc přináší další dva problémy. Jedním z nich je kolize s anglickým spojením „fast food“ (česky *rychlé jídlo*), kdy chce recenzent pochválit restauraci za rychlé jídlo. Druhým problémem je, že návštěvníci kvůli tomuto pojmenování často přistupují k ironii, pokud si myslí, že o rychlé jídlo nejde.

Dalším vícejazyčným problémem je stavba samotného jazyka. Příkladem je němčina, kde se podstatná jména píší s velkým počátečním písmenem a s malým písmenem pak mohou znamenat něco jiného. Příkladem je slovo *„Essen“* (česky *jídlo*) a *„essen“* (česky *jíst*).

Dalším problémem, kterým se liší recenze podle jazyků, je problém záporů. V záporné větě se s drtivou většinou skrývá negativní recenze. Ve spisovném anglickém jazyce by záporná věta měla obsahovat pouze jeden zápor a tím snadno detekovat zápornou větu. V češtině může být v jedné

větě záporů několik a přesto může jít o kladně postavenou větu. Příkladem je věta *„nevaří tu vůbec špatně“*, která obsahuje hned tři negativní slova, ale ve výsledku je to věta pozitivní.

Všechny tyto problémy mají vliv na přesnost analýzy sentimentu recenzí, ale některé se dají do jisté míry redukovat pomocí správně postaveného předzpracování textu, které se pak může mírně lišit pro každý jazyk.

Posledním problémem ke zmínění je rozložení dat mezi jednotlivé kategorie. Protože každý recenzent je jiná osoba, tak jinak píše a používá jiná slova. Proto je potřeba mít dostatečně velký korpus na učení klasifikátoru, aby mohl pojmout co největší škálu možných recenzí pro každou kategorii. Kromě velkého počtu je důležité i rozložení dat. Nevyvážený korpus může způsobit, že klasifikátor bude mít tendenci se zaměřit převážně na jednu část sentimentu a bude tedy mít nižší schopnost přesně klasifikovat příklady z části druhé. Ideální poměr sentimentu (pozitivní/negativní) i existence sentimentu (existuje/neexistuje) je jedna ku jedné, ale pokud recenze obsahují více kategorií, pak je velmi těžké takového výsledku dosáhnout.

Z tabulky četnosti sentimentu 4.2 na straně 45 je vidět, že četnost některých kategorií je velmi slabá, proto bude těžké je správně detekovat. Kategorie s dobrou četností vzhledem k velikosti korpusu pro češtinu jsou *General, Food, Sta* a *Speed*. Zároveň je u několika kategorií vidět špatný poměr mezi pozitivním a negativním sentimentem. Naopak kategorie, kde je poměr pěkný jsou *Sta , Speed* a *Cleanness*. V korpusu pro angličtinu i němčinu jsou kategorie *General*, *Food*, *Sta* a *Speed* také nejlépe zastoupeny.

Čistě na základě této statistiky by se dalo očekávat, že velmi dobré výsledky analýzy sentimentu budou mít kategorie *Sta* a *Speed* a pěkné výsledky by mohly mít i kategorie *General* a *Food*.

5 Implementace

Program této práce jsem implementoval v programovacím jazyce **Python** verze 3.9.13. Tento jazyk jsem zvolil z několika důvodů. Prvním z nich je, že tento jazyk je velmi oblíbený a má tedy velmi výraznou a aktivní komunitu. Díky tomu je snadné vyhledat cenné rady a návody při řešení problémů spojených s implementací. Druhou výhodou je množství knihoven. Python má rozsáhlou knihovnu pro práci s textovými daty, jako jsou recenze. Knihovny jako *NLTK*, *Scikit-learn*, *PyTorch* nebo *Pandas* jsou velmi populární a nabízejí širokou škálu funkcí pro zpracování a klasifikaci textových dat.

V této kapitole popisuji architekturu programu a implementaci klíčových částí aplikace.

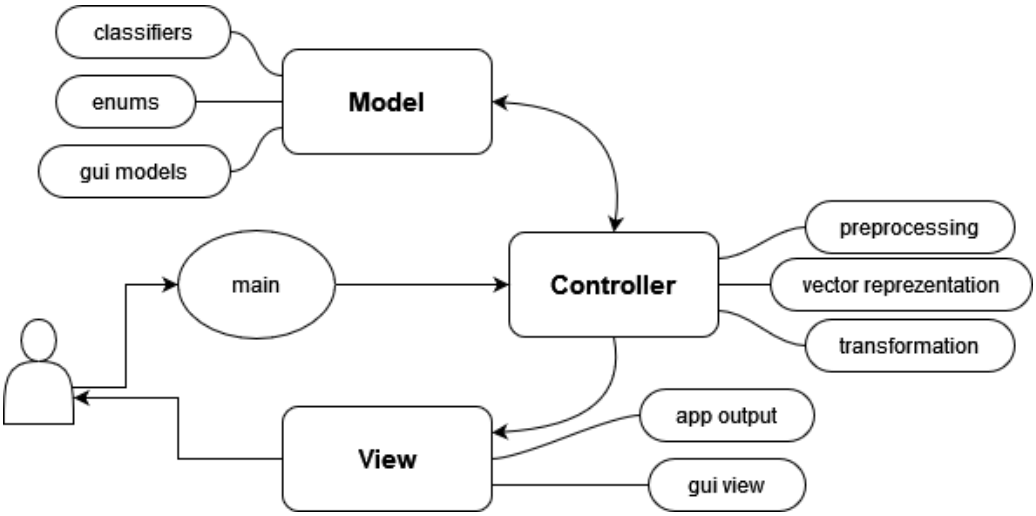
5.1 Architektura programu

Architekturu programu jsem zvolil na základě standardní architektury **MVC** (Model-View-Controller), která je zobrazena na obrázku 5.1 na straně 49. Základní myšlenkou *MVC* je oddělení logiky od výstupu a rozdělení kódu na vrstvy *Model, View* a *Controller*.

Model obsahuje objektovou část programu. Ta zahrnuje *classifiers, enums* a *gui models*. *Classifiers* jsou objekty klasifikátorů, které obsahují metody potřebné k trénování a testování modelů. *Enums* obsahuje enumerace sentimentu a jazyka. *Gui models* se používá pro ukládání použitých modelů.

Vrstva **Controller** řídí celý průběh programu a využívá k tomu *preprocessing, vector reprezentation* a *transformation*. Soubor *preprocessing* obsahuje metody použité pro předzpracování textu, *vector reprezentation* obsahuje metody pro vektorovou reprezentaci textu a *transformation* metody pro výpočet transformační matice. Vstupem této vrstvy je soubor *main*, který zpracovává vstupní parametry od uživatele. *Controller* komunikuje s vrstvou *Model* a odesílá data na vrstvu *View*.

Poslední vrstvou je **View**, která se stará o výstup směrem k uživateli. Ta se skládá z *app output* zprostředkovávající textový výstup (konzole a logger) a *gui view* obsluhující gui demonstrátoru popsaného v kapitole `Demonstrátor aplikace`.



Obrázek 5.1: Architektura programu

5.2 Průběh programu

V této části bych chtěl popsat postup programu s ukázkou některých zajímavých částí kódu.

Po spuštění příkazu `python main.py [parameters]` se spustí program a *main* zpracuje parametry a zkontroluje, zda jsou zadané správně. Pokud jsou, rozběhne se řadič programu. Řadič následně na základě parametrů započne vytváření modelu vektorové reprezentace textu nebo spustí proces klasifikace recenzí. Při vytvoření modelu se nejprve načte datový *feed* (zdroj dat), na data je aplikováno předzpracování dat, jsou odstraněny stop slova a prázdné tokeny a nakonec je model uložen ve formátu *fasttext* nebo *word2vec*.

Ukázka kódu vytvoření vektorového modelu:

```
1     # load feed
2     top_data_df = pd.read_excel(args.feed_path,
          sheet_name="Sheet1")
3     # preprocessing
4     result = preprocessing.lower_split(top_data_df, args
          .lang, check_lang=False)
5     # remove stop words
6     preprocessing.remove_bad_words(result, args.lang)
7     # remove empty tokens
8     result = [x for x in result if x != ['']]
9     # save vector model
10    if args.model_type == 'ft':
```

```
        make_fasttext_model(result, fasttext_file=args.
            model_path)
    elif args.model_type == 'w2v':
        make_word2vec_model(result, word2vec_file=args.
            model_path)
    else:
        app_output.exception(f"Model type {args.
            model_type} not found.")
```

Zajímavá část se vyskytuje v následující části kódu odstranění stop slov, kde se nejdříve načtou stop slova ze souboru a následně se rychle a efektivně, pouze na jedné řádce kódu pomocí filtru a lambda výrazu, nahradí každá recenze recenzí bez stop slov:

```
    # read stop words
    with open(filename, encoding='utf-8') as f:
        stop_words = [line.strip() for line in f.
            readlines()]
    # remove stop words from sentences
    for sentence in sentences:
        sentence[:] = list(filter(lambda word: word not
            in stop_words, sentence))
```

V případě, že se jedná o proces klasifikace, načtou se české recenze a proběhne opět předzpracování. U předzpracování bych rád ukázal metodu `split_line`, která má několik fází. První fází je odstranění diakritiky závislé na jazyce, protože každý jazyk používá jinou diakritiku. Druhou fází je odsazení důležité punktace a závorek, které oddělují klíčové části recenze od sebe. Třetí fází je smazání symbolů „'“ a „-“ z důvodu sloučení slov typu *„don't“* na *„dont“* a *„nebude-li“* na *„nebudeli“*. Čtvrtou fází je nahrazení některých symbolů mezerou, aby naopak ke sloučení nedošlo. Další fází je sloučení vícečetných mezer kvůli snadnější tokenizaci a poslední fází je odstranění mezer na začátku a na konci recenze:

```
def split_line(line, lang):
    # remove diacritics
    if lang == Language.CZECH.value:
        line = remove_diacritics_cs(line)
    elif lang == Language.GERMAN.value:
        line = remove_diacritics_de(line)
    # punctuation offset
    line = re.sub("([.,!?()])", r" \1 ", line)
```

```
    # remove unwanted symbols ', -
    line = re.sub(r"[\'\-]", "", line)
    # replace some unwanted symbols with space
    line = re.sub(r'[":;]', " ", line)
    # united spaces
    line = re.sub("\s{2,}", " ", line)
    # remove spaces at the beginning and end of line
    return line.lstrip().rstrip()
```

Pokud jde o vícejazyčnou klasifikaci, musí se stejně zpracovat i recenze cizího jazyka. V případě vícejazyčné klasifikace pomocí překladu, přeloží se recenze před předzpracováním do cílového jazyka pomocí knihovny **Easy-NMT**:

```
def translate_data(data_df, lang_from, lang_to):
    model = EasyNMT('opus-mt')
    data_df['Text'] = [model.translate(x, target_lang=
       lang_to, source_lang=lang_from) for x in data_df[
       'Text']]
```

Dalším krokem je načtení modelu vektorové reprezentace textu pro daný typ a daný jazyk. Za předpokladu, že jde o vícejazyčnou klasifikaci pomocí transformace, je nutné vypočítat transformační matici. Výpočet transformační matice pomocí ortogonální metody je v následující ukázce:

```
def compute_transform_matrix_orthogonal(target_model,
   source_model, filename):
    # read dictionary <target_word
       translate_of_target_word>
    with open(filename, encoding="utf-8") as f:
        matrix_t = None
        matrix_s = None
        for line in f:
            line = line.rstrip()
            wt, ws = line.split()
            # get words' vectors
            try:
                vec_t = target_model[wt]
                vec_s = source_model[ws]
            except:
                continue
            # matrix assembly
```

```
            if matrix_t is None:
                matrix_t = vec_t
                matrix_s = vec_s
            else:
                matrix_t = np.vstack((matrix_t, vec_t))
                matrix_s = np.vstack((matrix_s, vec_s))
        # transposition of the target matrix
        matrix_t_T = matrix_t.T
        # multiplication of transposed target matrix and
            source matrix
        X = np.dot(matrix_t_T, matrix_s)
        # Singular Value Decomposition
        U, s, V_T = np.linalg.svd(X)
        V = V_T.T[:, :len(s)]
        U_T = U.T
        trans_matrix = np.dot(V, U_T)
    return trans_matrix.T
```

Funkčnost metody jsem kontroloval pomocí *kosinové podobnosti* vektoru slova a transformovaného vektoru slova překladu pomocí transformační matice:

```
def cosine_similarity(vec_a, vec_b):
    return np.dot(vec_a, vec_b) / (np.linalg.norm(vec_a)
        * np.linalg.norm(vec_b))
```

Dalším krokem procesu je samotná klasifikace. Nejdříve proběhne klasifikace detekující každou kategorii v recenzích a následně klasifikace detekující sentiment pro každou kategorii. Rozhodnutí o tom, která metoda vektorové reprezentace textu a typ klasifikátoru se použije, závisí na parametrech od uživatele. Jako ukázku klasifikátoru bych chtěl uvést neuronovou síť *LSTM*. Z důvodu rozsáhlého zdrojového kódu je v příloze v sekci `Zdrojový kód LSTM`. Kód je z důvodu větší přehlednosti upraven. Obsahuje Inicializaci neuronové sítě a metody pro trénování a testování.

6 Experimenty a testování

V této kapitole jsou popsány použité metody pro zpracování a klasifikaci recenzí a výsledky testování. Běžně se používají tři různá vyhodnocení průměrné hodnoty metriky. **Macro průměr** jednoduše sečte výsledky metriky všech tříd a součet vydělí počtem tříd. **Vážený průměr** se vypočítá jako součet všech výsledků dané metriky vynásobených váhami. Váhou je zde počet skutečných prvků dané třídy vydělený počtem všech prvků. **Micro průměr** vypočítává globální průměr dané metriky. Tedy pro *TP* (*true positive*) platí $TP = TP_1 + TP_2 + ... + TP_n$, kde *n* je počet tříd. Pro *TN, FP* a *FN* platí stejný vztah. Tento průměr počítá podíl správně klasifikovaných prvků ze všech prvků, což je to samé jako celková přesnost (*AC*).

Výsledky jsou zaznamenány v tabulkách a jejich hodnoty jsou hodnoty vážený průměr pro *F1 – F1 score* (kvalita klasifikace), *AC – accuracy* (přesnost) nebo-li *micro průměr* a metrika *MC – majority class* (třída většiny). Metrika *MC* se používá k vyhodnocení modelů, které klasifikují data do jedné třídy a nezajímají se o přesnost klasifikace do ostatních tříd. Výsledkem této metriky je zastoupení pouze nejčastější třídy v datech. Slouží tedy jako hranice, pod kterou by se přesnost (*AC*) experimentu neměla dostat, aby byla úspěšnější než klasifikace na jednu třídu. Výsledné hodnoty experimentů byly vypočítány pomocí výpočtu průměru po odstranění extrémů (z anglického *trimmed mean*). Proces byl spuštěn pětkrát, nejvyšší a nejnižší hodnota byla odstraněna a se zbytkem hodnot se udělal aritmetický průměr. Každá kategorie má dvě části, které jsou zobrazeny v tabulkách níže. Její první řádek ukazuje úspěšnost detekce existence kategorie a druhý řádek ukazuje úspěšnost detekce sentimentu kategorie ze všech recenzí, kde daná kategorie existuje. Trénování a testování v českém jazyce je popsané v sekci `Jednojazyčná klasifikace textu` a trénování v českém jazyce, ale testovaní na datech v jazyce anglickém a německém je popsáno v sekci `Vícejazyčná klasifikace textu`.

6.1 Předzpracování dat

V procesu předzpracování dat byl na text aplikován *lowercasing*, *odstraněna diakritika, odstranění* a *odsazení některých znaků*. Znaky, které jsou odsazené, jsou závorky a interpunkce. Tyto znaky jsou oddělené mezerami od slov. Jsou zachovány, protože mohou mít význam pro klasifikaci textu tím, že oddělují od sebe části, které k sobě nepatří, určují vztahy mezi slovy a

větami nebo vyznačují důležité body v textu. Odstraněné znaky jsou nahrazeny mezerou, aby se tím nesloučila slova oddělená pouze tímto znakem. Výjimkou jsou znaky „-“ a „’“, které se vyskytují ve slovech jako jsou v češtině „nebo-li“ nebo v angličtině „don’t“. Smyslem je, aby se tady jednotlivé části neodlučovaly a vznikla slova „neboli“ a „dont“. Pokud po zpracování vznikne na nějakém místě více mezer vedle sebe, jsou sloučeny do jedné, aby se nevytvářely prázdné *tokeny*. Následně byl text rozdělen na jednotlivé *tokeny* podle mezer. Nakonec byla vyhozena slova ze seznamu *stop slov*.

6.2 Reprezentace textu

Pro reprezentaci textu byly použity dva modely diskrétní reprezentace textu pro klasifikátory *SVM* a *Logistická regrese* a statické distribuční reprezentace textu *word2vec* a *fasttext* pro neuronové sítě. Dynamická distribuční reprezentace by se pravděpodobně jevila jako lepší model, ale pro jejich použití je potřeba velký výkon a paměť, proto jsem se rozhodl využít pouze statickou. Distribuční modely byly vytvořeny pomocí knihovny **gensim**. Oba modely mají dimenzi vektorů *300* s minimální četností slov *2*, to znamená, že slova, která se vyskytnou v datovém korpusu jen jednou, nebudou součástí slovníku. Obě vektorové reprezentace textu byly vytvořeny z neoznačkovaných recenzí na řetězec *McDonald’s*. U *word2vec* je použitý takzvaný *padding*. *Padding* je technika, která se používá k vyrovnání délky vstupních sekvencí (délka věty) v textu, takže každá sekvence má stejnou délku. To se provádí přidáním nulových prvků na konec kratších sekvencí a místo slov, která nejsou ve slovníku, takže všechny sekvence mají stejnou délku jako nejdelší sekvence v sadě recenzí. *Fasttext* pro slova, která nejsou obsažena ve slovníku, generuje vektory pomocí n-gramů. Velikosti slovníku reprezentací *word2vec* a *fasttext* podle jazyka jsou v tabulce 6.1 na straně 54.

	velikost slovníku
čeština	31908
němčina	51095
angličtina	23536

Tabulka 6.1: Velikosti slovníků podle jazyka

6.3 Jednojazyčná klasifikace textu

V jednojazyčné klasifikaci textu jsem použil *SVM, Logistickou Regresi* a neuronové sítě *LSTM* a *CNN*. Standardní klasifikátory jsem implementoval pomocí knihovny **sklearn** a neuronové sítě pomocí knihovny **PyTorch**.

Neuronová síť *LSTM* obsahuje trénovací algoritmus *Skip-gram*, *skrytý stav* o velikosti *256* a *dropout* s hodnotou *0,5*. *Dropout* je technika neuronových sítí, která zabraňuje přetrénování sítě pomocí náhodného vypouštění vstupů. Optimalizace proběhla pomocí metody *Adam* s rychlostí učení *0,001*. (*learning rate*).

CNN obsahuje konvoluční vrstvu s filtry o velikosti (1, 2, 3, 5) a vrstvu *podvzorkování*. I zde proběhla optimalizace pomocí metody *Adam* s rychlostí učení *0,001*.

Obě neuronové sítě mají v případě vícejazyčného procesu dvě *embedding* vrstvy. Jednu pro testovací data a druhou pro trénovací data s vektory následně násobenými transformační maticí.

Označkovaná česká data (3000) jsou náhodně rozdělena v poměru 5:1 na trénovací a testovací data. Výsledky klasifikace standardních klasifikátorů jsou v tabulce 6.2 na straně 56 a výsledky neuronových síti jsou v tabulce 6.3 na straně 57.

6.4 Vícejazyčná klasifikace textu

Cross-lingual, neboli vícejazyčná, klasifikace textu používá jiný jazyk pro trénování a jiný pro testování. Zde jsem použil pouze neuronové sítě *LSTM* a *CNN* se stejnými parametry jako v *jednojazyčné* klasifikaci. Jako vícejazyčný model jsem zvolil model zarovnání na úrovni slov a sice přístup založený na mapovaní. Transformační matice byla vypočítána *ortogonální metodou* za pomocí slovníku o přibližném počtu 50 slov s jejich překladem. Slovník jsem vytvořil ze tří set nejvíce četných slov z datového zdroje obou jazyků a použil jen významová slova. Implementovaný a testovaný byl i výpočet transformační matice pomocí *regresní metody*, ale výsledky zdaleka nedosahovaly úspěšnosti jako u *ortogonální metody*. Pro každý jazyk byla data testována pomocí transformační matice a pomocí překladu. Při překladu byly recenze přeloženy pomocí knihovny **EasyNMT**.

Všechna označená česká data (3000) jsou použita jako trénovací data a všechna označena data cizího jazyka (500) jsou použita jako data testovací. Výsledky klasifikace pro německý jazyk pomocí překladu jsou v tabulce 6.4 na straně 58 a pomocí transformační matice jsou v tabulce 6.5 na straně

CZ	SVM tfidf		SVM bow		LR tfidf		LR bow		
	F1	AC	F1	AC	F1	AC	F1	AC	MC
General	0.82	0.82	0.81	0.81	0.82	0.82	**0.83**	**0.82**	0.57
sentiment	**0.92**	**0.93**	0.91	0.92	0.83	0.85	0.92	0.92	0.76
Food	0.85	0.84	**0.89**	**0.89**	0.77	0.80	0.85	0.86	0.69
sentiment	**0.87**	**0.87**	0.85	0.85	0.73	0.78	0.84	0.85	0.69
Drink	0.96	0.96	**0.97**	**0.97**	0.93	0.95	0.96	0.96	0.93
sentiment	**0.89**	**0.90**	0.87	0.88	0.75	0.80	0.87	0.88	0.73
Staff	0.91	0.92	**0.95**	**0.95**	0.83	0.95	0.93	0.94	0.72
sentiment	0.91	0.91	**0.93**	**0.93**	0.88	0.88	0.92	0.92	0.50
Speed	0.92	0.93	**0.95**	**0.95**	0.86	0.88	0.94	0.94	0.75
sentiment	**0.97**	**0.97**	0.94	0.94	0.96	0.96	0.92	0.92	0.52
Cleaness	0.96	0.96	**0.98**	**0.98**	0.91	0.93	0.97	0.97	0.92
sentiment	0.89	0.89	0.87	0.87	**0.92**	**0.91**	0.87	0.87	0.62
Prices	0.97	0.98	**0.98**	**0.98**	0.96	0.97	0.97	0.98	0.96
sentiment	0.90	0.90	**0.91**	**0.90**	0.76	0.81	**0.91**	**0.90**	0.76
Environ.	0.97	0.97	**0.98**	**0.98**	0.95	0.96	0.97	0.97	0.94
sentiment	0.83	0.88	0.83	0.88	0.83	0.88	0.83	0.88	0.52
Occup.	0.95	0.95	**0.97**	**0.97**	0.90	0.92	0.96	0.96	0.88
sentiment	0.86	0.87	0.83	0.83	0.83	0.86	**0.87**	**0.89**	0.80
Průměr	0,91	0,91	0,92	0,92	0,86	0,88	0,91	0,91	

Tabulka 6.2: Výsledky klasifikace pro trénování i testování v českém jazyce. F1 – f1-score, AC – accuracy, MC – majority class, bow – bag of words, tfidf – tf-idf

59. Pro anglický jazyk pomocí překladu jsou v tabulce 6.6 na straně 60 a pomocí transformační matice v tabulce 6.7 na straně 61.

CZ	LSTM ft		LSTM w2v		CNN ft		CNN w2v		
	F1	AC	F1	AC	F1	AC	F1	AC	MC
General	0.83	0.83	0.82	0.82	**0.87**	**0.87**	0.84	0.84	0.57
sentiment	0.96	0.96	0.94	0.94	0.95	0.95	**0.97**	**0.97**	0.76
Food	0.90	0.90	0.87	0.87	**0.91**	**0.91**	0.88	0.88	0.69
sentiment	0.87	0.88	0.87	0.88	0.90	0.90	**0.93**	**0.93**	0.69
Drink	**0.95**	**0.95**	**0.95**	**0.95**	0.89	0.93	0.89	0.93	0.93
sentiment	0.80	0.81	0.82	0.83	**0.90**	**0.90**	0.88	0.88	0.73
Staff	0.94	0.94	0.94	0.94	**0.95**	**0.95**	0.94	0.94	0.72
sentiment	0.96	0.96	0.93	0.93	0.96	0.96	0.96	0.96	0.50
Speed	0.94	0.94	0.94	0.94	**0.96**	**0.96**	0.95	0.95	0.75
sentiment	0.93	0.93	0.94	0.94	**0.97**	**0.97**	0.96	0.96	0.52
Cleaness	**0.98**	**0.98**	0.97	0.97	0.95	0.95	0.95	0.95	0.92
sentiment	0.87	0.87	0.89	0.89	0.94	0.94	**0.96**	**0.96**	0.62
Prices	**0.97**	**0.97**	0.95	0.97	0.95	0.96	0.95	0.96	0.96
sentiment	0.72	0.76	0.72	0.76	**0.86**	**0.86**	0.84	0.84	0.62
Environ.	**0.96**	**0.96**	0.94	0.95	0.91	0.94	0.91	0.94	0.94
sentiment	0.83	0.88	0.83	0.88	**0.93**	**0.94**	**0.93**	**0.94**	0.88
Occup.	**0.96**	**0.96**	0.95	0.96	**0.96**	**0.96**	0.95	0.95	0.88
sentiment	0.72	0.80	0.72	0.80	**0.87**	**0.89**	**0.87**	**0.89**	0.80
Průměr	0,89	0,90	0,89	0,90	0,87	0,93	0,92	0,93	

Tabulka 6.3: Výsledky klasifikace pro trénování i testování v českém jazyce pomocí neuronových sítí. F1 – f1-score, AC – accuracy, MC – majority class, ft – fasttext, w2v – word2vec

DE	LSTM ft		LSTM w2v		CNN ft		CNN w2v		
translate	F1	AC	F1	AC	F1	AC	F1	AC	MC
General	**0.81**	**0.81**	0.79	0.79	0.75	0.74	0.75	0.74	0.65
sentiment	0.89	0.89	0.85	0.85	**0.90**	**0.90**	0.88	0.88	0.76
Food	**0.90**	**0.90**	0.87	0.87	0.87	0.88	0.86	0.86	0.63
sentiment	0.81	0.81	0.77	0.77	**0.84**	**0.84**	0.82	0.82	0.55
Drink	**0.96**	**0.97**	0.95	0.96	**0.96**	**0.97**	0.95	0.96	0.96
sentiment	0.87	0.88	0.75	0.75	**0.89**	**0.88**	0.87	0.88	0.63
Staff	**0.89**	**0.89**	0.85	0.85	0.84	0.84	0.82	0.82	0.53
sentiment	0.85	0.85	0.85	0.85	**0.88**	**0.88**	0.86	0.86	0.53
Speed	**0.94**	**0.94**	0.90	0.90	0.90	0.90	0.89	0.89	0.76
sentiment	0.89	0.89	0.87	0.88	**0.93**	**0.93**	0.91	0.91	0.53
Cleaness	**0.94**	**0.94**	0.90	0.90	**0.94**	**0.94**	0.79	0.86	0.86
sentiment	0.77	0.78	0.75	0.75	**0.81**	**0.81**	**0.81**	**0.81**	0.61
Prices	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
sentiment	0.64	0.75	0.64	0.75	**0.83**	**0.75**	**0.83**	**0.75**	0.75
Environ.	0.95	0.96	0.95	0.96	0.94	0.96	0.94	0.96	0.96
sentiment	0.77	0.84	0.77	0.84	**0.87**	**0.89**	0.74	0.79	0.84
Occup.	0.93	0.93	0.93	0.93	**0.94**	**0.94**	0.92	0.93	0.95
sentiment	0.63	0.74	0.63	0.74	0.63	0.74	0.63	0.74	0.74
Průměr	0,86	0,87	0,83	0,85	0,87	0,88	0,85	0,86	

Tabulka 6.4: Výsledky klasifikace pro trénování v českém jazyce a testování v německém jazyce za pomoci překladu. F1 – f-score, AC – accuracy, MC – majority class

DE	LSTM ft		LSTM w2v		CNN ft		CNN w2v		
t-matrix	F1	AC	F1	AC	F1	AC	F1	AC	MC
General	**0.76**	**0.72**	0.74	0.74	0.71	0.72	0.75	0.71	0.65
sentiment	**0.87**	**0.86**	0.84	0.84	0.82	0.83	0.77	0.79	0.67
Food	0.91	0.91	0.90	0.90	**0.92**	**0.92**	0.90	0.90	0.63
sentiment	**0.83**	**0.83**	0.75	0.75	0.63	0.68	0.78	0.79	0.55
Drink	**0.97**	**0.97**	0.95	0.97	0.95	0.96	0.95	0.96	0.96
sentiment	**0.60**	**0.63**	**0.60**	**0.63**	0.38	0.38	0.38	0.38	0.63
Staff	**0.91**	**0.91**	0.88	0.89	0.90	0.90	0.89	0.89	0.53
sentiment	**0.89**	**0.89**	0.83	0.84	0.88	0.88	0.84	0.85	0.53
Speed	**0.92**	**0.92**	0.87	0.88	0.90	0.90	0.85	0.85	0.76
sentiment	**0.86**	**0.86**	**0.86**	**0.86**	0.85	0.86	0.78	0.79	0.53
Cleaness	**0.96**	**0.96**	0.89	0.91	0.93	0.94	0.91	0.93	0.86
sentiment	**0.79**	**0.79**	**0.79**	**0.79**	0.68	0.68	0.67	0.68	0.61
Prices	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
sentiment	0.64	0.75	0.64	0.75	**0.77**	**0.75**	**0.77**	**0.75**	0.75
Environ.	**0.95**	**0.96**	0.94	0.96	0.94	0.96	0.94	0.96	0.96
sentiment	0.77	0.84	0.77	0.84	0.77	0.84	0.77	0.84	0.84
Occup.	0.94	0.95	0.94	0.95	0.94	0.95	0.93	0.95	0.95
sentiment	0.63	0.74	0.63	0.74	0.63	0.74	0.63	0.74	0.74
Průměr	0,84	0,86	0,82	0,84	0,80	0,82	0,79	0,82	

Tabulka 6.5: Výsledky klasifikace pro trénování v českém jazyce a testování v německém jazyce za pomoci transformační matice. F1 – f-score, AC – accuracy, MC – majority class

EN	LSTM ft		LSTM w2v		CNN ft		CNN w2v		
translate	F1	AC	F1	AC	F1	AC	F1	AC	MC
General	**0.75**	**0.75**	0.72	0.71	0.74	0.75	0.72	0.72	0.67
sentiment	**0.87**	**0.87**	0.84	0.84	0.85	0.85	0.85	0.85	0.54
Food	0.84	0.85	0.87	0.87	**0.89**	**0.89**	0.84	0.84	0.62
sentiment	0.87	0.87	0.86	0.86	0.88	0.88	**0.89**	**0.89**	0.52
Drink	0.91	0.93	0.87	0.91	**0.92**	**0.93**	0.87	0.91	0.91
sentiment	0.70	0.71	**0.81**	**0.82**	0.70	0.71	0.78	0.79	0.68
Staff	**0.79**	**0.79**	0.75	0.76	0.75	0.76	0.71	0.72	0.55
sentiment	0.89	0.89	0.88	0.88	**0.90**	**0.90**	0.88	0.88	0.57
Speed	0.92	0.92	0.91	0.91	**0.93**	**0.93**	0.87	0.88	0.77
sentiment	**0.93**	**0.93**	0.92	0.92	0.91	0.91	0.89	0.89	0.72
Cleaness	**0.97**	**0.97**	0.93	0.93	0.95	0.95	0.93	0.93	0.91
sentiment	0.78	0.78	0.76	0.76	0.79	0.80	**0.95**	**0.9**5	0.54
Prices	0.98	0.98	0.98	0.98	0.98	0.98	0.96	0.98	0.98
sentiment	0.73	0.67	**0.90**	**0.89**	0.71	0.67	**0.90**	**0.89**	0.89
Environ.	0.94	0.96	**0.95**	**0.96**	0.94	0.96	0.94	0.96	0.96
sentiment	0.60	0.71	0.60	0.71	0.60	0.71	0.60	0.71	0.71
Occup.	0.92	0.93	0.92	0.93	0.92	0.93	**0.93**	**0.93**	0.95
sentiment	0.87	0.91	0.87	0.91	0.87	0.91	**0.95**	**0.96**	0.91
Průměr	0,85	0,86	0,85	0,87	0,85	0,86	0,86	0,87	

Tabulka 6.6: Výsledky klasifikace pro trénování v českém jazyce a testování v anglickém jazyce za pomoci překladu. F1 – f-score, AC – accuracy, MC – majority class

EN	LSTM ft		LSTM w2v		CNN ft		CNN w2v		
t-matrix	F1	AC	F1	AC	F1	AC	F1	AC	MC
General	0.75	0.75	0.73	0.74	**0.76**	**0.76**	0.67	0.72	0.67
sentiment	**0.69**	**0.70**	0.64	0.67	0.60	0.65	0.67	0.70	0.54
Food	**0.87**	**0.87**	0.79	0.80	0.82	0.83	0.81	0.82	0.62
sentiment	**0.72**	**0.73**	0.70	0.71	0.69	0.72	0.66	0.70	0.52
Drink	**0.93**	**0.94**	0.91	0.93	0.87	0.91	0.87	0.91	0.91
sentiment	**0.72**	**0.71**	0.51	0.50	0.67	0.68	0.44	0.46	0.68
Staff	**0.87**	**0.87**	0.82	0.82	0.82	0.82	0.78	0.78	0.55
sentiment	0.75	0.75	**0.86**	**0.86**	0.69	0.70	0.84	0.84	0.57
Speed	0.**89**	**0.89**	0.88	0.87	0.87	0.87	0.89	0.88	0.77
sentiment	**0.93**	**0.93**	0.76	0.75	0.92	0.92	0.80	0.79	0.72
Cleaness	0.90	0.92	0.94	0.95	**0.96**	**0.96**	0.87	0.91	0.91
sentiment	**0.72**	**0.73**	0.65	0.66	0.48	0.59	0.60	0.66	0.54
Prices	**0.98**	**0.98**	**0.98**	**0.98**	0.96	0.98	0.96	0.98	0.98
sentiment	0.84	0.89	0.84	0.89	0.84	0.89	**0.90**	**0.89**	0.89
Environ.	0.94	0.96	0.94	0.96	0.94	0.96	0.94	0.96	0.96
sentiment	0.60	0.71	0.60	0.71	0.60	0.71	0.60	0.71	0.71
Occup.	0.92	0.94	0.92	0.94	0.92	0.94	**0.93**	**0.95**	0.95
sentiment	0.87	0.97	0.87	0.91	0.87	0.91	0.87	0.91	0.91
Průměr	0,83	0,84	0,80	0,82	0,79	0,82	0,78	0,81	

Tabulka 6.7: Výsledky klasifikace pro trénování v českém jazyce a testování v anglickém jazyce za pomoci transformační matice. F1 – f-score, AC – accuracy, MC – majority class

7 Hodnocení výsledků

Podle výsledků z kapitoly `Experimenty a testování` se dá vyhodnotit několik závěrů. Porovnatelné hodnoty jsou v rámci *Jednojazyčné klasifikace textu* a v rámci *Vícejazyčné klasifikace textu*. Celkovému hodnocení se věnuje sekce *Celkové hodnocení*.

7.1 Hodnocení jednojazyčné klasifikace textu

V *jednojazyčné klasifikaci textu* se mohou porovnat klasifikátory mezi sebou a to standardní (*Suppor vector machines* a *Logistic regression*) i neuronové sítě (*Longshort term memory* a *Convolutional neural networks*). Také se zde mohou porovnat vektorové reprezentace stejného klasifikátoru – *bag of words* a *tf-idf* u standardních klasifikátorů a *fasttext* a *word2vec* u neuronových sítí.

Standardní klasifikátory v *jednojazyčné klasifikaci textu* mají obecně velmi dobré výsledky. U standardních klasifikátorů je *Accuracy* u všech kategorií i jejich sentimentů vyšší než *Majority class*. *F1-score* se pohybuje velmi vysoko. V porovnání standardních klasifikátorů i jejich textových reprezentací žádný zásadní rozdíl není.

U neuronových sítí jsou výsledky také velmi dobré. *Accuracy* není u žádné z kategorií ani jejich sentimentů nižší než *Majority class*. Nicméně u některých kategorií s velkou *Majority class* se *AC* a *MC* rovnají (příkladem je kategorie *Drink* a *Prices* u *CNN* s reprezentací *fasttext*), ale to se dá u neuronových sítí s malým a nesymetrickým počtem trénovacích dat očekávat. Potvrzuje to i tvrzení ze sekce `Experimenty a testování`, že s malými daty může být *SVM* úspěšnější než neuronové sítě. U kategorií s vyšším zastoupením dat a lepším rozdělením sentimentu dosahují neuronové sítě většinou lepších výsledků než standardní klasifikátory. To se týká převážně kategorií *General, Food, Sta* a *Speed*. U těchto kategorií má neuronová síť *CNN* lepší výsledky než *LSTM*. Naopak u kategorií se slabším zastoupení dat je *LSTM* většinou úspěšnější. Mezi textovými reprezentacemi *fasttext* a *word2vec* nejsou zásadní rozdíly.

7.2 Hodnocení vícejazyčné klasifikace textu

V tomto hodnocení hodnotím jen *vícejazyčnou klasifikaci textu*. I zde se mohou porovnávat klasifikátory mezi sebou, ale pouze neuronové sítě. Budu po-

rovnávat vícejazyčnou klasifikaci pomocí ortogonální transformační matice a pomocí překladu. Porovnat se mohou výsledky mezi anglickým a německým jazykem, ale je třeba pamatovat, že testování proběhlo na jiných datech.

Hodnocení klasifikace německých recenzí

Klasifikace německých recenzí na českých trénovacích datech pomocí ortogonální transformační matice i pomocí překladu má velmi dobré výsledky. Pomocí překladu jsou většinou výsledky o něco lepší, ale v některých případech je úspěšnější transformační matice a to za použití *LSTM* a *fasttext* v detekci kategorie *Cleaness* a detekci kategorie i sentimentu kategorie *Sta* . *LSTM* a *fasttext* je obecně u výpočtů pomocí transformační matice viditelně nejúspěšnější. U překladu je tato kombinace u detekce kategorií také nejúspěšnější, zatímco u detekce sentimentu je většinou lepší *CNN* s reprezentací *fasttext*.

Některé kategorie s nízkým počtem dat se drží na hranici *MC* a některé jsou dokonce pod touto hranicí. To se týká sentimentu kategorie *Environment* u překladu a kategorie *Drink* s klasifikátorem *CNN* u transformační matice. Když se podíváme na tabulku četností sentimentu jednotlivých kategorií 4.2 na straně 45, všimneme si, že třída *Environment* je v trénovacích datech obsažená pouze 166-krát a v testovacích datech pro německý jazyk 19-krát a u třídy *Drink* mají trénovací data 198 výskytů a testovací dokonce jen 8. Tato nepřesnost je tedy pochopitelná.

Hodnocení klasifikace anglických recenzí

Výsledky klasifikace anglických recenzí na českých trénovacích datech pomocí překladu a detekce kategorií pomocí ortogonální matice jsou také velmi dobré. Ale detekce sentimentu transformace je zde méně kvalitní. Přesto jediný výsledek *AC*, který je nižší než *MC* je u detekce sentimentu kategorie *Drink* u klasifikace za použití vektorové reprezentace *word2vec*. Pod hranicí je i detekce sentimentu kategorie *Prices* u překladu pomocí *fasttext*.

Překlad je tedy i zde, až na některé výjimky, úspěšnější než transformace. Jednou z výjimek je detekce kategorie u tříd *Drink* a *Sta* pomocí klasifikátoru *LSTM* s reprezentací *fasttext*. A druhou je detekce sentimentu kategorie *Prices* a detekce kategorie *Occupancy* pomocí *CNN* a *word2vec*.

Za použití transformace je nejúspěšnější opět klasifikátor *LSTM* v kombinaci s *fasttext*. U překladu je *fasttext* úspěšnější než *word2vec*, s výjimkou detekce sentimentu kategorií *Prices* a *Occupancy*, kde je úspěšnější *word2vec* s klasifikátorem *CNN*.

7.3 Shrnutí výsledků experimentů

Na silných kategoriích jako je *General*, *Food*, *Sta* a *Speed* má obecně výrazně lepší výsledky klasifikace na německých datech a to i pomocí transformační matice i pomocí překladu. Horší úspěšnost anglických dat může být způsobena problémy popsanými v sekci `Problémy při vytváření korpusu` nebo nedostatečně velkým slovníkem reprezentace dat. Jako nejlepší kombinace klasifikátoru a vektorové reprezentace textu pro transformaci jazykového prostoru do prostoru jiného jazyka se prokázala neuronová síť *LSTM* s *fasttext*.

7.4 Celkové hodnocení

Výpočty z předchozí kapitoly slouží spíše k hodnocení a rozpoznání rozdílů v detekci kategorie a určení sentimentu pro recenze, kde daná recenze existuje. Celkový výsledek je lépe vidět na celkové klasifikaci sentimentu na všech recenzích pro každou kategorii. Výsledky jsou zobrazeny *micro průměrem* nebo-li *accuracy* (AC) v tabulkách 7.1 na straně 64 pro překlad německých recenzí, 7.2 na straně 65 pro použití transformace s německými recenzemi, 7.3 na straně 65 pro překlad na anglických recenzích a 7.4 na straně 66 pro použití transformace s anglickými recenzemi. Ve výsledné tabulce 7.5 na straně 66, která zobrazuje průměry napříč všech kombinací vícejazyčné klasifikace na testovacích datech angličtiny i němčiny s použitím překladu i transformace. Tabulka potvrzuje nejúspěšnější kombinaci klasifikační metody *LSTM* s vektorovou reprezentací textu *fasttext*.

DE MT	LSTM ft	LSTM w2v	CNN ft	CNN w2v
General	0,72	0,67	0,66	0,66
Food	0,73	0,67	0,74	0,71
Drink	0,85	0,72	0,85	0,84
Staff	0,76	0,73	0,74	0,71
Speed	0,84	0,79	0,85	0,81
Cleaness	0,74	0,67	0,76	0,69
Prices	0,74	0,74	0,74	0,74
Environ.	0,81	0,81	0,86	0,76
Occup.	0,69	0,69	0,70	0,68
Průměr	0,76	0,72	0,77	0,73

Tabulka 7.1: Micro průměr detekce sentimentu pro recenze německého jazyka pomocí překladu. *ft* – fasttext, *w2v* – word2vec

DE TM	LSTM ft	LSTM w2v	CNN ft	CNN w2v
General	0,64	0,61	0,59	0,59
Food	0,75	0,67	0,62	0,72
Drink	0,61	0,60	0,36	0,36
Staff	0,82	0,74	0,79	0,75
Speed	0,78	0,74	0,77	0,67
Cleaness	0,76	0,72	0,64	0,63
Prices	0,74	0,74	0,74	0,74
Environ.	0,81	0,81	0,81	0,81
Occup.	0,70	0,70	0,70	0,70
Průměr	0,73	0,70	0,67	0,66

Tabulka 7.2: Micro průměr detekce sentimentu pro recenze německého jazyka pomocí transformace. *ft* – fasttext, *w2v* – word2vec

EN MT	LSTM ft	LSTM w2v	CNN ft	CNN w2v
General	0,65	0,60	0,63	0,61
Food	0,74	0,75	0,78	0,74
Drink	0,66	0,75	0,66	0,72
Staff	0,70	0,67	0,68	0,64
Speed	0,86	0,84	0,85	0,78
Cleaness	0,76	0,71	0,77	0,89
Prices	0,65	0,87	0,65	0,87
Environ.	0,69	0,69	0,68	0,68
Occup.	0,85	0,85	0,85	0,89
Průměr	0,73	0,75	0,73	0,76

Tabulka 7.3: Micro průměr detekce sentimentu pro recenze anglického jazyka pomocí překladu. *ft* – fasttext, *w2v* – word2vec

EN TM	LSTM ft	LSTM w2v	CNN ft	CNN w2v
General	0,52	0,50	0,50	0,51
Food	0,64	0,57	0,60	0,57
Drink	0,67	0,47	0,62	0,42
Staff	0,65	0,71	0,58	0,66
Speed	0,82	0,65	0,81	0,70
Cleaness	0,68	0,62	0,56	0,60
Prices	0,87	0,87	0,87	0,87
Environ.	0,69	0,69	0,68	0,69
Occup.	0,87	0,87	0,87	0,86
Průměr	0,71	0,66	0,68	0,65

Tabulka 7.4: Micro průměr detekce sentimentu pro recenze anglického jazyka pomocí transformace. *ft* – fasttext, *w2v* – word2vec

	LSTM ft	LSTM w2v	CNN ft	CNN w2v
DE MT	0,76	0,72	0,77	0,73
DE TM	0,73	0,70	0,67	0,66
EN MT	0,73	0,75	0,73	0,76
EN TM	0,71	0,66	0,68	0,65
Průměr	0,733	0,708	0,713	0,700

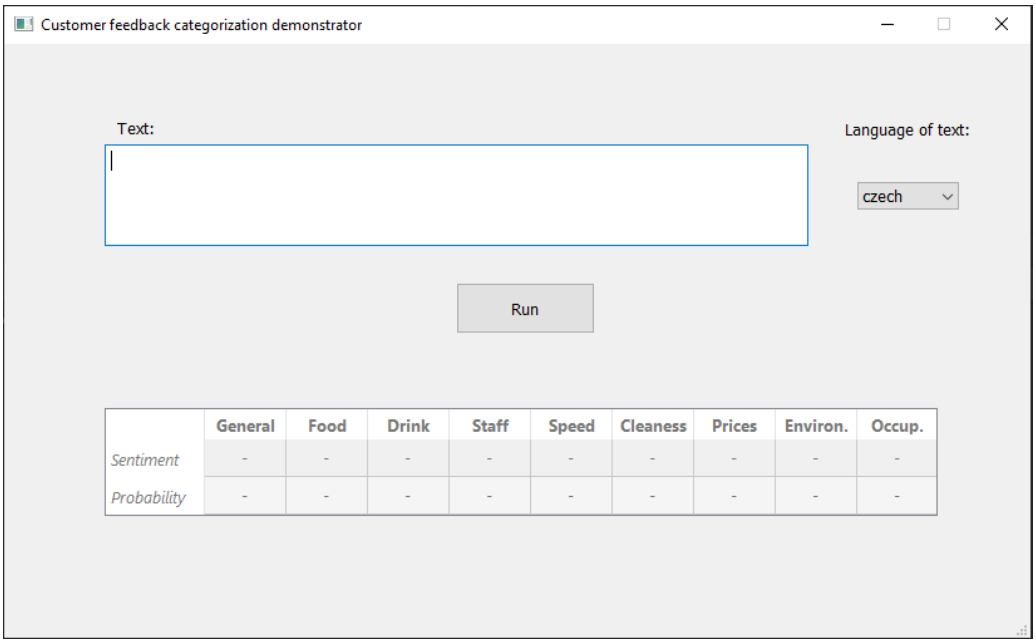
Tabulka 7.5: Celkové výsledky klasifikace. *DE* – německá testovací data, *EN* – anglická testovací data, *MT* – překlad, *TM* – transformační matice, *ft* – fasttext, *w2v* – word2vec

8 Demonstrátor aplikace

Jednoduchý demonstrátor byl implementován pomocí programovacího jazyka **Python** verze 3.9.13. Grafické uživatelské prostředí (gui) demonstrátoru bylo vytvořené knihovnou **PyQt5**. Aplikace demonstruje využití kategorizace a detekce polarity sentimentu na textu zadaném uživatelem díky gui. Demonstrátor programu je možné spustit příkazem `python gui.py`. Žádné parametry nepotřebuje. Protože se jedná o demonstrátor, jsou vybrány nejlépe hodnocené modely na základě experimentů. Modelem reprezentací textu byl tedy zvolen *fasttext*, klasifikační model je neuronová síť *LSTM* a pro výpočet transformační matice je zvolena *ortogonální metoda*. Více kombinací metod je možné vyzkoušet pomocí konzolové aplikace, jejíž použití je popsáno v sekci `Uživatelská dokumentace` včetně instalační příručky.

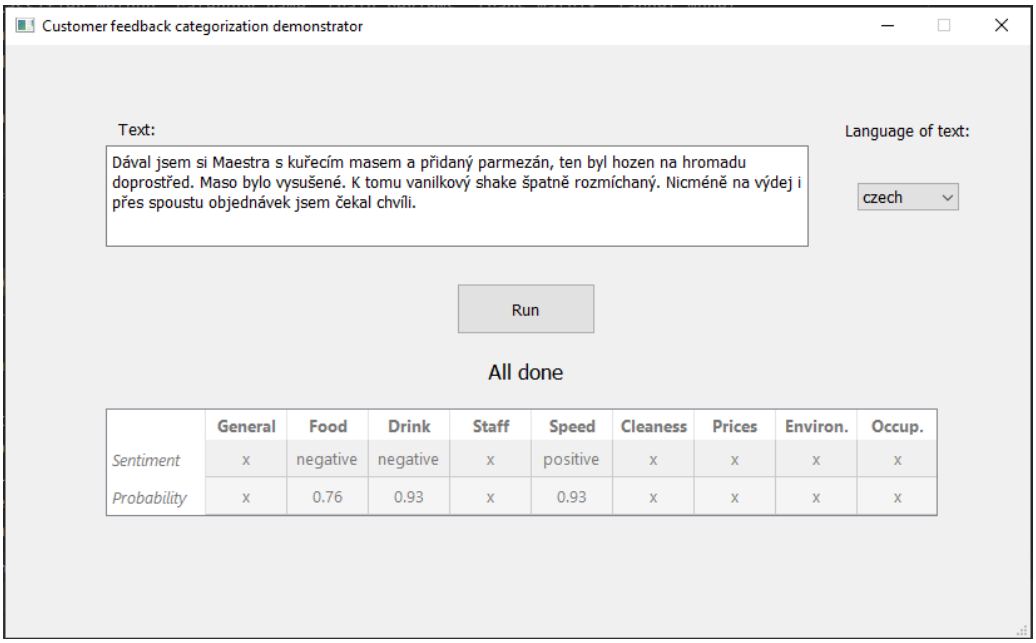
Program očekává vektorový model *fasttext* pro každý jazyk uložený v souboru `Vstupni_data/vec_model/ft_cs.bin`, analogicky *en* nebo *de* pro angličtinu a němčinu. Modely tedy musí být předem vytvořené konzolovou aplikací. Pro urychlení klasifikace se naučené klasifikátory pro každou kategorii ukládají do složky `Vstupni_data/saved_classifiers/`. Při dalším spuštění programu se tyto klasifikátory opět načtou ze souboru a klasifikace bude tedy výrazně rychlejší.

Po spuštění příkazu se objeví gui aplikace, která je vidět na obrázku 8.1 na straně 68.

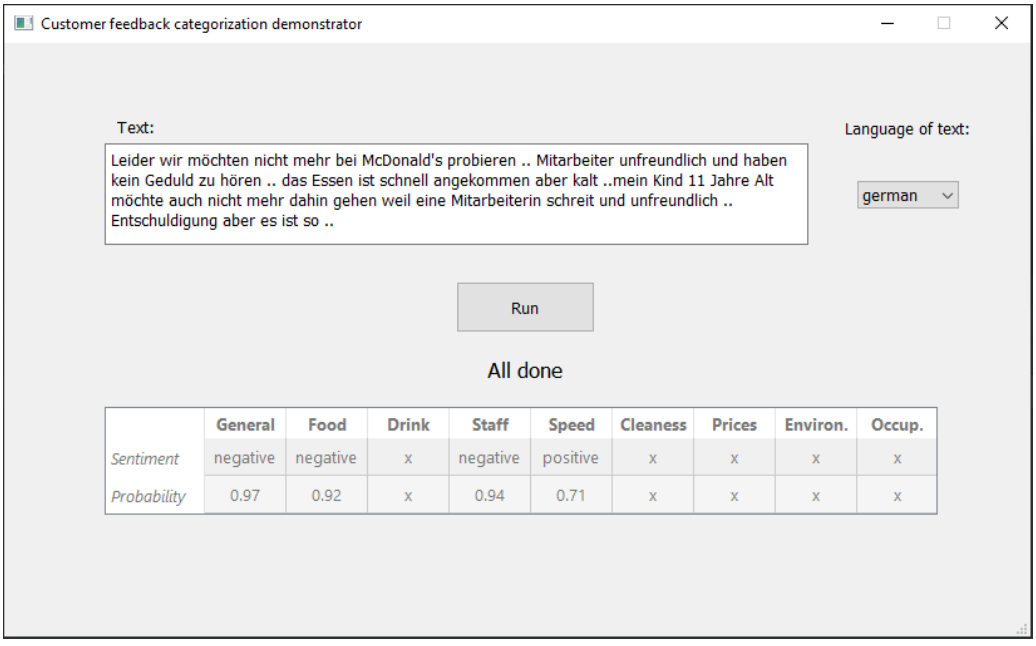


Obrázek 8.1: Gui aplikace po spuštění aplikace

Před spuštěním tlačítkem **Run**, je potřeba do volného pole označeným *Text* vložit text, který bude klasifikován a zvolit jazyk textu (*Language of text*) – *czech, english* nebo *german*. Při zvolení českého jazyka bude klasifikace jednojazyčná, při zvolení jiného jazyka bude klasifikace vícejazyčná s pomocí transformační matice. Po stisknutí tlačítka **Run** proběhne klasifikace do všech tříd. Pokud klasifikátor určí, že třída je přítomna v textu, v tabulce na řádce *Sentiment* vypíše sentiment třídy a na řádce *Probability* vypíše pravděpodobnost sentimentu. Pokud podle klasifikátoru třída přítomná není, vypíše do buněk symbol "x". Příklad výsledné jednojazyčné klasifikace je vidět na obrázku 8.2 na straně 69 a příklad vícejazyčné klasifikace je vidět na obrázku 8.3 na straně 69.



Obrázek 8.2: Gui aplikace po dokončení jednojazyčné klasifikace



Obrázek 8.3: Gui aplikace po dokončení vícejazyčné klasifikace

9 Závěr

Tato práce je pojata hlavně jako práce praktická, kde klíčovou částí je vytvoření vlastního datového korpusu, který je následně zpracován pro trénovací a testovací data klasifikace. V rámci diplomové práce jsem se seznámil s předzpracováním textu pro úlohu kategorizace zákaznické zpětné vazby a s kategorizací samotnou. Pro kategorizaci jsem v práci vyzkoušel klasifikační algoritmy *SVM* a *Logistickou regresi* pro jednojazyčnou klasifikaci.

Následně jsem se také seznámil s metodami vícejazyčného zpracování textu a dvě z nich jsem v práci využil na experimenty. První metoda pracuje na přístupu založeném na mapování. Jde o *ortogonální* metodu transformace mezi jazykovými prostory, která probíhá pomocí transformační matice. Druhou možností vícejazyčného zpracování textu je jednoduchý překlad textu z jednoho jazyka do druhého a dále pracovat jako s jednojazyčnou klasifikací. Pro vícejazyčnou klasifikaci pomocí transformace i překladu jsem vyzkoušel neuronové sítě *LSTM* a *CNN*. Tyto neuronové sítě jsem pro porovnání vyzkoušel i v jednojazyčné klasifikaci.

Po seznámení s modely jsem vytvořil testovací korpus z recenzí vybraného řetězce na *Google Reviews*. Tímto řetězcem byl řetězec rychlého občerstvení *McDonald's*. Skutečné recenze z tohoto řetězce jsem vyhledal v českém, německém a anglickém jazyce, recenze jsem označkoval v kategoriích *General*, *Food*, *Drink*, *Sta* , *Speed*, *Cleanness*, *Prices*, *Environment* a *Occupancy* a připravil ke zpracování. Českých recenzí jsem označkoval 3000, protože jde o primární trénovací data. Recenzí anglického a německého jazyka jsem označkoval 500. Označení 100 recenzí trvá přibližně jednu hodinu. Použitím transformace se tedy ušetří 50 hodin času strávených nad značkováním recenzí. Tato část byla velice zajímavá, protože jsem se zde dozvěděl mnoho poznatků nejen k následnému zpracování dat, ale i k rozdílnému chování v jiných kulturách v psané podobě.

Klasifikátory jsem otestoval a následně zhodnotil výsledky testování. Standardní klasifikátory *SVM* a *Logistická regrese* byly testovány v kombinaci s vektorovými reprezentacemi textu *BOW* a *Tf-Idf* a modely neuronových sítí byly testovány s kombinací vektorových reprezentací textu *word2vec* a *fasttext*. Pro porovnání jsem neuronové sítě otestoval i v jednojazyčné klasifikaci, kde byly úspěšnější než standardní algoritmy. Výhodou standardních algoritmů je výrazně vyšší rychlost než u neuronových sítí. U vícejazyčné klasifikace se překlad ukázal jako přesnější než transformace pomocí matic. Nevýhodou překladu je čas strávený na překladu textu a potřeba

internetového připojení. Jako nejúspěšnější klasifikační metodu pro kategorizaci recenzí v různých jazycích do předem definovaných kategorií, u kterých detekuje polaritu, jsem na základě výsledků testování zvolil *LSTM* v kombinaci s vektorovou reprezentací textu *fasttext*. Tuto kombinaci jsem použil pro vytvoření jednoduchého grafického demonstrátoru, v kterém je možné klasifikaci vyzkoušet. Kromě demonstrátoru je možné využít i konzolovou aplikaci, která má větší výběr klasifikačních modelů.

Práce je postavená na konceptu restaurace s malým počtem dat. Nevýhodou malého počtu dat je horší přesnost klasifikace a nerovná distribuce sentimentu v trénovacích i testovacích datech. Naopak výhodou je, že každá nepřesnost nebo chyba se výrazně projeví a je tedy snadnější tyto nepřesnosti detekovat a napravit a tak lépe připravit klasifikační proces i pro případná větší množství dat. Výsledný program by se dal využít v libovolné restauraci pro kategorizaci zákaznické zpětné vazby ve formě recenzí ve zmíněných kategoriích. Stále se v programu nachází velký prostor pro rozšíření. V první řadě, by se dal samotný demonstrátor rozvinout o funkce konzolové aplikace. Zajímavé by bylo využít datum a čas recenzí a pomocí časové osy určovat například problémy v jednotlivých směnách. Dalším krokem by mohlo být využití *Business Inteligence* a z obdržených recenzí získat užitečné grafy a diagramy, které pomohou lépe zobrazit nedostatečnosti v podniku.

Seznam obrázků

Literatura

[1] About · Apify. Dostupné z: https://apify.com/about.

[2] What is Logistic Regression? Dostupné z: https://aws.amazon.com/what-is/logistic-regression/.

[3] Machine learning NLP text classification algorithms and Models. Dostupné z: https://www.projectpro.io/article/machine-learning-nlp-text-classification-algorithms-and-models/523#mcetoc_1fle0r0758.

[4] *Zpracování přirozeného jazyka aneb NLP* [online]. Fakulta informatiky Masarykovy univerzity, 2001-2023. [cit. 2023/2/2]. Dostupné z: https://nlp.fi.muni.cz/cs/ZpracovaniPrirozenehoJazyka.

[5] Pytorch. Dostupné z: https://pytorch.org/get-started/locally/.

[6] Co Je to strojové učení?: Definice, typy A příklady: SAP insights. Dostupné z: https://www.sap.com/cz/insights/what-is-machine-learning.html.

[7] Support Vector Machines (SVM) algorithm explained, Jun 2017. Dostupné z: https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/.

[8] Baad, D. Sentiment classification with bow, Apr 2020. Dostupné z: https://medium.com/swlh/sentiment-classification-with-bow-202c53dac154.

[9] Baad, D. Sentiment classification using CNN in Pytorch, Apr 2020. Dostupné z: https://towardsdatascience.com/sentiment-classification-using-cnn-in-pytorch-fba3c6840430.

[10] Baad, D. Sentiment classification for restaurant reviews using TF-IDF, Apr 2020. Dostupné z: https://medium.com/swlh/sentiment-classification-for-restaurant-reviews-using-tf-idf-42f707bfe44d.

[11] Bird, S. – Klein, E. – Loper, E. *Natural language processing with python*. O'Reilly, 2009. ISBN 0596516495.

[12] Chandran, S. Introduction to text representations for Language processing-part 1, Nov 2021. Dostupné z: https://towardsdatascience.com/introduction-to-text-representations-for-language-processing-part-1-dc6e8068b8a4.

[13] Chandran, S. Introduction to text representations for Language processing-part 2, Nov 2021. Dostupné z: https://towardsdatascience.com/introduction-to-text-representations-for-language-processing-part-2-54fe6907868.

[14] Devlin, J. et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, s. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. Dostupné z: https://aclanthology.org/N19-1423.

[15] Fatih, K. Cosine similarity. Dostupné z: https://www.learndatasci.com/glossary/cosine-similarity/.

[16] Furnieles, G. Sigmoid and Softmax functions in 5 minutes, Sep 2022. Dostupné z: https://towardsdatascience.com/sigmoid-and-softmax-functions-in-5-minutes-f516c80ea1f9.

[17] Gallagher, S. – Rafferty, A. – Wu, A. NLP - Owerview, 2004. Dostupné z: https://cs.stanford.edu/people/eroberts/courses/soco/projects/2004-05/nlp/overview_history.html.

[18] Hu, J. An overview for text representations in NLP, Mar 2020. Dostupné z: https://towardsdatascience.com/an-overview-for-text-representations-in-nlp-311253730af1.

[19] Jay. Analýza Sentimentu NLP Pomocí Pythonu, Apr 2022. Dostupné z: https://hashdork.com/cs/nlp-sentiment-analysis-using-python/.

[20] Kung-Hsiang, H. S. Word2vec and FastText word embedding with Gensim, Sep 2018. Dostupné z: https://towardsdatascience.com/word-embedding-with-word2vec-and-fasttext-a209c1d3e12c.

[21] Nss. An intuitive understanding of word embeddings: From count vectors to word2vec, Oct 2020. Dostupné z: https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/.

[22] Ormazabal, A. et al. Analyzing the Limitations of Cross-lingual Word Embedding Mappings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, s. 4990–4995, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1492. Dostupné z: https://aclanthology.org/P19-1492.

[23] Peters, M. E. et al. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, s. 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. Dostupné z: https://aclanthology.org/N18-1202.

[24] Pontiki, M. et al. SemEval-2016 Task 5: Aspect Based Sentiment Analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, s. 19–30, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/S16-1002. Dostupné z: https://aclanthology.org/S16-1002.

[25] Ruder, S. – Vulić , I. – Søgaard, A. A Survey of Cross-lingual Word Embedding Models. *Journal of Artificial Intelligence Research*. aug 2019, 65, s. 569–631. doi: 10.1613/jair.1.11640. Dostupné z: https://doi.org/10.1613%2Fjair.1.11640.

[26] Schlicht, M. The Complete Beginner's Guide to Chatbots, May 2018. Dostupné z: https://chatbotsmagazine.com/the-complete-beginner-s-guide-to-chatbots-8280b7b906ca.

[27] Simha, A. Understanding TF-IDF for Machine Learning, Oct 2021. Dostupné z: https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/.

[28] Team, D. Kernel functions-introduction to SVM Kernel amp; Examples, Mar 2021. Dostupné z: https://data-flair.training/blogs/svm-kernel-functions/.

[29] Vincent, R. J. Singular value decomposition (SVD)-working example, Aug 2021. Dostupné z: https://medium.com/intuition/singular-value-decomposition-svd-working-example-c2b6135673b5.

[30] Wampler, M. The technology behind Chat GPT-3, Jan 2023. Dostupné z: https://www.clearcogs.com/post/the-technology-behind-chat-gpt-3.

[31] Zhang, L. – Wang, S. – Liu, B. ArXiv, Jan 2018. Dostupné z: https://arxiv.org/ftp/arxiv/papers/1801/1801.07883.pdf.

Seznam zkratek

- **NLP** - Natural language preocessing
- **GPT** - Generative Pre-training
- **SEO** - Search engine optimization
- **BOW** - Bag of words
- **TF** - Term frequency
- **IDF** - Inverse document frequency
- **ELMO** - Embeddings from Language Models
- **BERT** - Bidirectional Encoder Representations from Transformers
- **ANN** - Artificial neural networks
- **FFNN** - Feedforward neural networks
- **ReLU** - Rectified Linear Unit
- **CNN** - Convolutional neural networks
- **RNN** - Recurrent neural networks
- **LSTM** - Long Short Term Memory
- **CCA** - Canonical Correlation Analysis
- **CBOW** - Continuous Bag of Words
- **MSE** - Mean squared error
- **SVD** - Singular Value Decomposition
- **ASBA** - Aspect-Based Sentiment Analysis
- **MVC** - Model-View-Controller

Přílohy

A Zdrojový kód LSTM

```
class LongShortTermMemory(nn.Module):
    def __init__(...):
        super(LongShortTermMemory, self).__init__()

        self.hidden_dim = hidden_dim
        self.no_layers = no_layers

        # target embeddings
        vec_model_train = gensim.models.KeyedVectors.
           load(model_filename_train)
        weights_train = vec_model_train.wv
        self.embedding_train = nn.Embedding.
           from_pretrained(torch.FloatTensor(
           weights_train.vectors))

        # source embeddings
        if model_filename_test:
            vec_model_test = gensim.models.KeyedVectors.
               load(model_filename_test)
            weights_test = vec_model_test.wv
            self.embedding_test = nn.Embedding.
               from_pretrained(torch.FloatTensor(
               weights_test.vectors))
        else:
            self.embedding_test = self.embedding_train

        self.lstm = nn.LSTM(input_size=weights_train.
           vector_size, hidden_size=self.hidden_dim,
                            num_layers=no_layers,
                               batch_first=True)
        # dropout layer
        self.dropout = nn.Dropout(drop_prob)
        # linear layer
        self.fc = nn.Linear(self.hidden_dim, output_dim)
        # sigmoid layer
```

```
28          self.sig = nn.Sigmoid()
29          # transformation matrix
30          self.trans_matrix = None if trans_matrix is None
                else torch.from_numpy(trans_matrix).float().
               to(device)
31
32      def forward(self, x, hidden, train_input):
33          batch_size = x.size(0)
34          # target embeddings
35          if train_input:
36              embeds = self.embedding_train(x)
37          # source embeddings
38          else:
39              embeds = self.embedding_test(x)
40              if self.trans_matrix is not None:
41                  # transformation
42                  for i in range(len(embeds)):
43                      embeds[i] = torch.matmul(self.
                           trans_matrix, embeds[i].T).T
44          # lstm out
45          lstm_out, hidden = self.lstm(embeds, hidden)
46          lstm_out = lstm_out.contiguous().view(-1, self.
               hidden_dim)
47          # dropout and fully connected layer
48          out = self.dropout(lstm_out)
49          out = self.fc(out)
50          # sigmoid function
51          sig_out = self.sig(out)
52          # reshape to be batch_size first
53          sig_out = sig_out.view(batch_size, -1)
54          # get last batch of labels
55          sig_out = sig_out[:, -1]
56          # return last sigmoid output and hidden state
57          return sig_out, hidden
58
59      def init_hidden(self, batch_size, device):
60          # Create two new tensors with sizes no_layers x
               batch_size x hidden_dim,
61          # initialized to zero, for hidden state and cell
                state of LSTM
62          h0 = torch.zeros((self.no_layers, batch_size,
               self.hidden_dim)).to(device)
```

```
63          c0 = torch.zeros((self.no_layers, batch_size,
                self.hidden_dim)).to(device)
64          hidden = (h0, c0)
65          return hidden
66
67
68  def training_LSTM(...):
69      # make vector index map
70      X_train = [vector_reprezentation.
            make_vector_index_map(...) for line in X_train]
71      X_train = np.array(X_train)
72      Y_train = Y_train_sentiment.to_numpy()
73      # create dataloader
74      train_data = TensorDataset(torch.from_numpy(X_train)
            , torch.from_numpy(Y_train))
75      train_loader = DataLoader(train_data, shuffle=True,
            batch_size=batch_size)
76
77      no_layers = 2
78      output_dim = 3
79      hidden_dim = 256
80
81      # lstm model init
82      lstm_model = LongShortTermMemory(...)
83
84      # to device (cpu/gpu)
85      lstm_model.to(device)
86
87      lr = 0.001
88      criterion = nn.BCELoss()
89      optimizer = torch.optim.Adam(lstm_model.parameters()
            , lr=lr)
90
91      # function to predict accuracy
92      def acc(pred, label):
93          pred = torch.round(pred.squeeze())
94          return torch.sum(pred == label.squeeze()).item()
95
96      clip = 5
97      epochs = 5
98
99      for epoch in range(epochs):
```

```
100         train_losses = []
101         train_acc = 0.0
102         lstm_model.train()
103         # init hidden state
104         h = lstm_model.init_hidden(batch_size, device)
105         for inputs, labels in train_loader:
106             inputs, labels = inputs.to(device), labels.
                to(device)
107             # create new variables for the hidden state
108             h = tuple([each.data for each in h])
109             lstm_model.zero_grad()
110             output, h = lstm_model(inputs, h, True)
111             # calculate the loss and perform backprop
112             loss = criterion(output, labels.float())
113             loss.backward()
114             train_losses.append(loss.item())
115             # calculate accuracy
116             accuracy = acc(output, labels)
117             train_acc += accuracy
118             # prevent the exploding gradient problem in
                LSTM
119             nn.utils.clip_grad_norm_(lstm_model.
                parameters(), clip)
120             optimizer.step()
121
122     return lstm_model
123
124 def testing_LSTM(...):
125     bow_cnn_predictions = []
126     # lstm evaluation
127     lstm_model.eval()
128     with torch.no_grad():
129         i = 0
130         for tags in X_test:
131             # make vector index map
132             vec = vector_reprezentation.
                make_vector_index_map(...)
133             inputs = np.expand_dims(vec, axis=0)
134             torch.from_numpy(inputs).float().to(device)
135             inputs = torch.from_numpy(inputs).to(device)
136             batch_size = 1
137             # init hidden state
```

```
138             h = lstm_model.init_hidden(batch_size,
                    device)
139             h = tuple([each.data for each in h])
140             # get prediction
141             output, h = lstm_model(inputs, h, False)
142             if output < 0.5:
143                 bow_cnn_predictions.append(0)
144             else:
145                 bow_cnn_predictions.append(1)
146             i += 1
147     # compare with true labels and print result
148     app_output.output(classification_report(
            Y_test_sentiment, bow_cnn_predictions))
```

B Uživatelská dokumentace

Uživatelská dokumentace obsahuje popis a použití konzolové aplikace. Pro spuštění je nutné mít nainstalovaný jazyk **Python** verze 3.9.13. Součástí programu je textový soubor *requirements.txt*, kde jsou potřebné knihovny pro správný chod programu včetně doporučených verzí. Tyto knihovny je možné nainstalovat pomocí příkazu:

```
pip install -r requirements.txt
```

V případě stroje se systémem *GPU NVIDIA* je možné nainstalovat knihovnu **torch** s parametrem *–index-url https://download.pytorch.org/whl/cu117*, aby program mohl fungovat na *GPU* a provádět paralelní výpočty. [5]

B.1 Konzolová aplikace

Konzolovou aplikaci je možné spustit zadáním příkazu `python main.py` a několika povinným a podmíněně povinným parametrům:

- **–a (action)** – Udává jakou akci by měl program vykonat. Tento parametr je vždy povinný a podmiňuje povinnost ostatních parametrů.
  - 'mono' – definuje jednojazyčnou klasifikaci
  - 'cross' – definuje vícejazyčnou klasifikaci s použitím transformační matice
  - 'translate' – definuje vícejazyčnou klasifikaci s použitím překladu

  - 'model' – definuje vytvoření modelu

- **–mt (model type)** – Udává typ vektorových modelů. Tento parametr je vždy povinný.
  - 'ft' – definuje typ modelu *fasttext*
  - 'w2v' – definuje typ modelu *word2vec*
  - 'tfidf' – definuje typ modelu *tf-idf*
  - 'bow' – definuje typ modelu *bow*

- **–mp (model path)** – Udává cestu ke jménu souboru obsahující vektorový model pro jazyk určený k trénování. Tento parametr je povinný pokud je typ modelu *word2vec* nebo *fasttext*.

- **–mptest (model path test)** – Udává cestu ke jménu souboru obsahující vektorový model pro jazyk určený k testování. Tento parametr je povinný pokud je akce programu *'model', 'cross'* nebo *'translate'* a typ vektorového modelu je *word2vec* nebo *fasttext*.

- **–cm (model type)** – Udává typ klasifikačních modelů. Tento parametr je vždy povinný.
  - 'lstm' – definuje typ modelu *fasttext*
  - 'cnn' – definuje typ modelu *word2vec*
  - 'svm' – definuje typ modelu *Support vector machines*
  - 'logreg' – definuje typ modelu *Logistic regression*
  - 'dectree' – definuje typ modelu *Decision tree*

- **–rp (reviews path)** – Udává cestu ke jménu souboru obsahující recenze pro trénování. Tento parametr je povinný pokud je akce programu *'mono', 'cross'* nebo *'translate'*.

- **–rptest (reviews path test)** – Udává cestu ke jménu souboru obsahující recenze pro testování. Tento parametr je povinný pokud je akce programu *'cross'* nebo *'translate'*.

- **–l (language)** – Udává jazyk dat pro trénování. Od jazyka se odvíjí předzpracování textu, překlad a slovník pro výpočet transformační matice. Tento parametr je vždy povinný.

- –**ltest (language test)** – Udává jazyk dat pro testování. Tento parametr je povinný, pokud je akce programu *'cross'* nebo *'translate'*.
- –**fp (feed path)** – Udává cestu ke jménu souboru obsahující recenze pro vytvoření vektorového modelu. Tento parametr je povinný, pokud je akce programu *'model'*.

Klasifikátory neuronové sítě (*lstm, cnn*) mohou být pouze v kombinaci s textovou reprezentací *word2vec* a *fasttext* a ostatní klasifikátory (*Support vector machines, Logistic regression* a *Decision tree*) mohou být pouze v kombinaci s reprezentací textu *bow* a *tf-idf*. Vícejazyčná transformace může být použita jen s klasifikátory neuronových sítí (*lstm, cnn*).

Po dokončení běhu programu jsou výsledky zobrazeny v samotné konzoli a v textovém souboru *log.txt*.

Příklady použití

Protože možností a kombinací použití programu je mnoho, uvedu několik příkladů spuštění podle toho, co se od programu očekává (při zachování struktury adresáře přiloženého archivu přílohy, bude soubor `data/` nahrazen cestou `../../Vstupni_data/`):

- Pro vytvoření vektorového modelu *word2vec* pro český jazyk z datového setu *feed_cs.xlsx* se spustí příkaz:

  ```
  python main.py -a model -mp data/vec_model/w2v_cs.bin -l cs
  -fp data/feed/feed_cs.xlsx -mt w2v
  ```

- Pro jednojazyčnou klasifikaci s vektorovým modelem *tf-idf* a klasifikátorem *svm* pro český jazyk se spustí příkaz:

  ```
  python main.py -a mono -rp data/review/reviews_cs.xlsx -l cs
  -mt tfidf -cm svm
  ```

- Pro vícejazyčnou klasifikaci za pomoci transformační matice s vektorovým modelem *word2vec* a klasifikátorem *lstm* pro česká trénovací data a německá testovací data se spustí příkaz:

  ```
  python main.py -a cross -rp data/review/reviews_cs.xlsx -mp
  data/vec_model/w2v_cs.bin -l cs -rptest data/review/reviews_de.xlsx
  -mptest data/vec_model/w2v_de.bin -ltest de -mt w2v -cm lstm
  ```

- Pro vícejazyčnou klasifikaci za pomoci překladu s vektorovým modelem *fasttext* a klasifikátorem *lstm* pro česká trénovací data a německá testovací data se spustí příkaz:

```
python main.py -a translate -rp data/review/reviews_cs.xlsx
-mp data/vec_model/ft_cs.bin -l cs -rptest data/review/reviews_de.xlsx
-ltest de -mt ft -cm lstm
```

C Adresářová struktura

- **Text_prace** – soubory napsané diplomové práce
  - **A20N0102P_DP.pdf** – text práce ve formátu *pdf*
  - **latex** – zdrojové soubory textu práce včetně obrázků
- **Poster**
  - **Prucha_Pavel_2023.pdf** – poster ve formátu *pdf*
  - **Prucha_Pavel_2023.pub** – poster ve formátu *pub*
- **Aplikace_a_knihovny**
  - **src** – všechny vytvořené zdrojové kódy
  - **docs** – vygenerovaná dokumentace ze zdrojových kódů
  - **navod.txt** – návod na sestavení a spuštění
- **Vstupni_data** – data, která jsou nezbytná pro běh aplikace
  - **dictionary** – slovníky slov a jejich překladů pro každý jazyk
  - **feed** – zdroje recenzí k vytvoření modelu vektorové reprezentace textu pro každý jazyk
  - **review** – označkované recenze pro každý jazyk
  - **stopwords** – soubory stop slov pro každý jazyky
  - **vec_model** – soubor určený pro modely vektorové reprezentace textu
- **Vysledky** – výstupní soubory naměřených a vypočtených výsledků experimentů
- **Readme.txt** – popis aktuální adresářové struktury