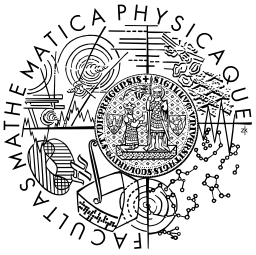


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Jakub Dargaj

Detekce intenzity v postojové analýze češtiny

Ústav formální a aplikované lingvistiky

Vedoucí bakalářské práce:	Mgr. Aleš Tamchyna
Studijní program:	Informatika
Studijní obor:	IOI

Praha 2017

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Název práce: Detekce intenzity v postojové analýze češtiny

Autor: Jakub Dargaj

Ústav: Ústav formální a aplikované lingvistiky

Vedoucí bakalářské práce: Mgr. Aleš Tamchyna, Ústav formální a aplikované lingvistiky

Abstrakt: Postojová analýza sa zaoberá automatickou extrakciou subjektívnych informácií z textu. Cieľom práce je predpovedať intenzitu postoja v českých textoch. Na riešenie tejto úlohy sme pripravili dataset filmových hodnotení užívateľov Česko-Slovenskej filmovej databázy. Porovnávame niekoľko metód strojového učenia, pričom sa zameriavame na extrakciu číselných atribútov z textových dát. S využitím konvolučných neurónových sietí a korpusovo závislého trénovania vektorových reprezentácií slov sa nám podarilo prekonať základné modely a dosiahnuť presnosť podobnú najnovším výsledkom v tejto oblasti. V práci taktiež analyzujeme model logistickej regresie na porovnanie použitých jazykových prostriedkov medzi recenziami s rôznymi stupňami hodnotenia.

Klíčová slova: postojová analýza, strojové učení, počítačová lingvistika

Title: Detection of Intensity in Sentiment Analysis of Czech

Author: Jakub Dargaj

Institute: Institute of Formal and Applied Linguistics

Supervisor: Mgr. Aleš Tamchyna, Institute of Formal and Applied Linguistics

Abstract: Sentiment analysis is concerned with automatic extraction of subjective information from text. The goal of this thesis is to predict the intensity of attitude in Czech texts. In order to solve this task, we prepared a dataset of movie reviews by users of Czech-Slovak Film Database. We compare several machine learning methods, focusing on feature extraction from text data. Using convolutional neural networks and corpus-dependent training of word embeddings, we surpassed basic models and achieved accuracy similar to the most recent results in this field. We also analyze the logistic regression model in order to compare the vocabulary used in reviews with different ratings.

Keywords: sentiment analysis, machine learning, computational linguistics

Táto práca by nevznikla bez niekoľkých ľudí, ktorým by som sa chcel týmto spôsobom poďakovať.

Vedúcemu práce, Mgr. Alešovi Tamchynovi, ďakujem za množstvo cenných odporúčaní, komentárov a rýchlu dostupnosť počas celej doby tvorenia tejto práce. Mgr. Kateřine Veselovskej, Ph.D. patrí takisto vďaka za navedenie tým správnym smerom a za konzultácie problémov, s ktorými som sa pri práci stretol. Obom ďakujem aj za motiváciu, spolupráca s Vami bola naozaj príjemná.

V neposlednom rade sa chcem poďakovať priateľom a rodine za veľkú podporu a prekazenie každého môjho pokusu túto prácu vzdať.

Ďakujem!

Obsah

Úvod

Od vynálezu počítača prebehlo niekoľko dekád a informačná spoločnosť výrazne pokročila. Dnes je už samozrejmosťou, že počítače ovplyvňujú takmer každú stránku ľudského života a zvládajú úlohy, ktoré sa kedysi zdali nepredstaviteľné. Umelá inteligencia dokáže riadiť vozidlá, pomáha lekárom pri diagnostike ochorení a nevidiacim cez mobilnú aplikáciu popisuje okolité objekty. V mnohých činnostiach je dokonca efektívnejšia než človek. Otázky jej možného využitia sa pomaly začínajú meniť na obavy o bezpečnosti umelej inteligencie. Aspoň v jednej oblasti by sme však oproti počítačom mohli mať prevahu. Dokážu stroje pociťovať emócie?

Na túto otázku, ktorá možno znie skôr ako sci-fi, by sme mohli odpovedať jednoducho. Počítač, nech je v akejkoľvek podobe, je stále iba kopou kovových súčiastok poprepájaných káblami, ktoré ak správne spolupracujú, robia to, čo od nich človek očakáva. Čo ak ale počítače pokročia a budú dôveryhodne prejavovať emócie bez toho, aby ich naozaj pociťovali? Budeme môcť objektívne posúdiť, či je takýto prejav skutočný alebo ide len o výstup algoritmu, ktorý kopíruje správanie niekoľkých ľudí?

Minimálne čítanie emócií počítačmi neznie až tak nereálne a presne o to sa v tejto práci pokúsime. Veľmi vtipne túto problematiku zobrazuje jedna epizóda populárneho seriálu, kde jej hlavná postava, geniálny introvert Sheldon Cooper, má problém s vnímaním emócií svojich priateľov. Kvôli tomu dostáva detektor emócií — zariadenie, ktoré s vysokou presnosťou identifikuje šťastné, smutné, nahnevané a nadšené tváre a pomáha mu viesť priateľské vzťahy.

Samozrejme, okrem toho by spomínaný detektor mohol mať aj naozajstné praktické využitie. Čítať emócie nemusíme len z tváre osôb, ale napríklad aj z textu. Nie každý text má nejaký emocionálny význam, a preto môže byť jednou z úloh detektora rozlíšiť, či je text objektívny alebo nie. Keď nejaká spoločnosť poskytuje služby alebo vydá svoj nový produkt na trh, zrejme ju bude zaujímať, ako sú s ním zákazníci spokojní a čo o ňom píšu. Preto je snahou čo najpresnejšie rozpoznať, či text vyjadruje pozitívne alebo negatívne emócie, prípadne identifikovať hodnotenú vlastnosť produktu. Všetkými týmito úlohami sa v širšom kontexte zaoberá postojová analýza.

Bežná postojová analýza rozlišuje len pozitívne a negatívne, prípadne neutrálne texty. Po vydaní nového produktu by okrem polarity hodnotení spoločnosť možno rada vedela, či výrobok dostáva len mierne pozitívne reakcie, alebo sú z neho zákazníci nadšení. Cieľom tejto práce je detekovať polaritu textu na širšej škále ako iba pozitívne verzus negatívne. Na to vytvoríme dataset filmových hodnotení užívateľov Česko-Slovenskej filmovej databázy, na ktorom použijeme metódy inšpirované najnovšími výsledkami v oblasti postojovej analýzy. V prvých troch kapitolách popíšeme tieto metódy strojového učenia, konkrétne logistickú regresiu a konvolučné neurónové siete. V kapitolách 4 a 5 navrhneme niekoľko prístupov k tejto úlohe a experimentálne ich porovnáme, šiesta kapitola je venovaná lexikálnej analýze textových hodnotení.

1. Lineárny model

Jedným z triviálnych spôsobov, ako z textu automaticky získať postoj autora, je vytvoriť slovník pozitívnych a negatívnych slov daného jazyka. Pre konkrétny vstup, napríklad recenziu produktu alebo tweet, ľahko spočítame počet slov oboch slovníkov a podľa toho určíme, či je sentiment textu pozitívny alebo negatívny, prípadne neutrálny. Tento prístup môžeme samozrejme ďalej rozšíriť na komplexný systém pravidiel, no už v takto zjednodušenom modeli by bolo vytvorenie slovníkov časovo náročné a použiteľné iba pre jeden jazyk.

Tu prichádza na rad strojové učenie, ktoré je hlavnou príčinou rozvoja postojovej analýzy v posledných rokoch. Namiesto manuálnej tvorby pravidiel študujeme všeobecné algoritmy, pomocou ktorých sa dokážu počítače naučiť z vlastných skúseností. Okrem toho, že väčšinu práce prenecháme počítaču a jeden algoritmus môžeme použiť na problémy z rôznych oblastí, sú tieto algoritmy často efektívnejšie a dosahujú lepšie výsledky.

V tejto kapitole, založenej na James a kol. (2014) a Mitchell (1997), predstavíme typy úloh, ktoré sa dajú pomocou strojového učenia riešiť. Popíšeme jednoduchý algoritmus, ktorý sa používa na klasifikáciu, a v závere uvedieme niekoľko spôsobov, ako zistiť, či je vybraný algoritmus na danú úlohu vhodný.

1.1 Strojové učenie

V strojovom učení sa snažíme čo najlepšie predpovedať výstupnú hodnotu zo vstupných dát. V závislosti na tom, či výstupné hodnoty máme k dispozícii, rozlišujeme dva hlavné typy úloh.

V učení s učiteľom (*supervised learning*) sa model naučí vzťah medzi vstupnými dátami a výstupnou hodnotou. Povedzme, že máme k dispozícii údaje o reprezentatívnej vzorke obyvateľov ČR, napríklad vek, lokalitu, dosiahnuté vzdelanie a priemerný mesačný príjem. Počítač na tejto vzorke naučíme vzťah medzi vstupnými parametrami a výškou príjmu. Ak natrénovanému modelu následne zadáme informácie o osobe, ktorej príjem nepoznáme, dostaneme predpovedanú výšku príjmu podľa vzťahu, ktorý sa naučil na trénovacích dátach.

Naopak, v učení bez učiteľa (*unsupervised learning*) výstupnú hodnotu nemáme k dispozícii. Aj napriek tomu môže byť počítač schopný nájsť určitú štruktúru v dátach. Príkladom je zhluková analýza, pri ktorej sa dáta snažíme rozdeliť na niekoľko navzájom odlišných skupín tak, aby sa prvky vnútri skupín čo najviac podobali. Pretože v experimente máme k vstupným dátam dostupnú aj výstupnú hodnotu, budeme sa v práci ďalej zaoberať iba metódami pre učenie s učiteľom.

Podľa toho, z akej množiny vyberáme výstupnú hodnotu, delíme úlohy na klasifikáciu a regresiu. Pri regresii je výstup spojitý, sem patrí aj spomínaná predikcia príjmu na základe demografických údajov. V klasifikačnej úlohe je zas výstupná hodnota z konečnej množiny, špeciálne pri binárnej klasifikácii máme na výber z dvoch hodnôt.

Okrem tvorby prediktívnych algoritmov je veľmi dôležitou súčasťou procesu strojového učenia príprava a spracovanie dát. Ak pracujeme s neštruktúrovanými dátami, akými sú obrázky alebo texty, je prvým krokom získať z dát údaje v číselnej podobe. Tento proces nie je jednoduchý a závisí od neho konečná úspešnosť

algoritmu, preto mu venujeme celú 3. kapitolu. Nateraz predpokladajme, že máme dáta v štruktúrovanej forme, tak ako v príklade o predikcii príjmu.

1.2 Logistická regresia

Logistická regresia je metóda strojového učenia určená pre binárnu klasifikáciu. Jej názov je síce zavádzajúci, no je odvodený od lineárnej regresie, na ktorej je táto metóda založená.

V lineárnej regresii predpokladáme, že výstupná hodnota je lineárne závislá na vstupných hodnotách. Presnejšie, ak máme na vstupe vektor premenných $x = \langle x_1, \ldots, x_m \rangle$, výstupnú hodnotu modelujeme funkciou

$$h(x) = \beta_0 + \sum_{i=1}^{m} \beta_i \, x_i,$$

tiež nazývanou hypotéza. Koeficienty $\beta_0, \ldots, \beta_m$ sú reálne čísla a hľadáme také, ktoré minimalizujú stratovú funkciu. V tomto prípade je to súčet vzdialeností predikcie od skutočnej výstupnej hodnoty y_i cez všetky trénovacie príklady. Formálne, ak máme k dispozícii n príkladov, stratová funkcia je v tvare

$$L(\beta) = \sum_{i=1}^{n} (h(x_i) - y_i)^2.$$

Lineárnu regresiu môžeme priamočiaro rozšíriť na binárny klasifikátor. Ak sú výstupné hodnoty z množiny $\{0, 1\}$, zvolíme vhodnú hraničnú hodnotu t. Pre $h(x) > t$ priradíme x výstupnú hodnotu 1, inak 0. Upravíme aj stratovú funkciu a namiesto súčtu vzdialeností je možné použiť napríklad podiel nesprávne klasifikovaných príkladov.

Takéto rozšírenie je však trochu nepraktické. Od binárneho klasifikátora by sme chceli, aby jeho výstupom bola pravdepodobnosť, že daný vstup patrí do danej skupiny. Hypotéza použitá v lineárnej regresii má ale obor hodnôt $\mathbb{R}$, takže sa ľahko stane, že výstup padne mimo intervalu $[0, 1]$.

Potrebujeme spojitú rastúcu funkciu, ktorá má definičný obor $\mathbb{R}$ a obor hodnôt $[0, 1]$. Tieto vlastnosti má tzv. sigmoida

$$g(z) = \frac{1}{1 + e^{-z}}.$$

Hodnotu $g(h(x))$ teraz už môžeme interpretovať ako pravdepodobnosť, že $y = 1$. Takto sme z lineárnej regresie odvodili logistickú regresiu, ktorej hypotéza vyzerá nasledovne

$$h(x) = \frac{1}{1 + e^{-\beta_0 - \sum_{i=1}^{m} \beta_i \, x_i}}.$$

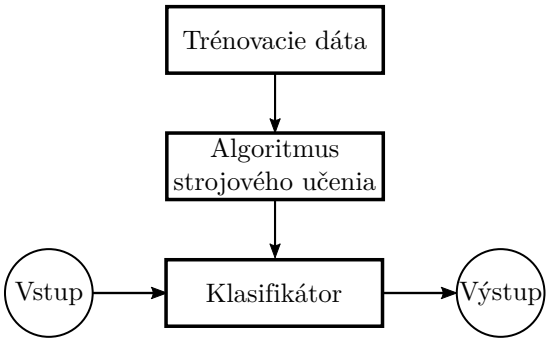
Na hľadanie koeficientov hypotézy použijeme metódu maximálnej vierohodnosti, MLE (*maximum likelihood estimation*). Základná myšlienka spočíva v tom, že sa snažíme maximalizovať pravdepodobnosť $h(x)$ pre vstupy s $y = 1$ a minimalizovať ju pre zvyšok. Maximalizujeme teda hodnotu funkcie

$$l(\beta) = \prod_{i:y_i=1} h(x_i) \prod_{j:y_j=0} (1 - h(x_j)).$$

Konečne sa dostávame k stratovej funkcii pre logistickú regresiu. Predchádzajúca funkcia má maximum v bode, v ktorom má minimum stratová funkcia

$$L(\beta) = \sum_{i=1}^{n} y_i \, log(1 - h(x_i)) \, + (1 - y_i) \, log(h(x_i)).$$

Na záver v stručnosti zhrňme popísaný postup. Klasifikátor na trénovacích dátach naučíme koeficienty lineárnej funkcie. Ak chceme predpovedať výstupnú triedu pre daný vstup, použijeme naň hypotézu s natrénovanými koeficientmi. Výstupom hypotézy je odhadovaná pravdepodobnosť, že tento vstup patrí do triedy 1. Podľa tejto hodnoty mu priradíme výslednú triedu, spravidla sa volí hranica 0,5.



Obr. 1.1: Schéma procesu strojového učenia.

1.3 Metóda najväčšieho spádu

V predchádzajúcej sekcii sme vysvetlili, že trénovanie modelu v podstate znamená hľadanie parametrov hypotézy tak, aby sme minimalizovali stratovú funkciu. Na to využijeme znalosti z optimalizácie.

Uvedieme iteratívnu metódu, ktorá hľadá lokálne minimum funkcie tak, že sa posunie proti smeru gradientu funkcie v aktuálnom bode. Presnejšie, na začiatku nastavíme koeficienty na $\beta^0 = 0$ a v kroku $k + 1$ aktualizujeme

$$\beta^{k+1} = \beta^k - \alpha \, \nabla L(\beta^k),$$

pre vhodne zvolený parameter α. Tento krok opakujeme, až pokiaľ nedosiahneme lokálne minimum. Detailnejšie je táto metóda popísaná v (Boyd a Vandenberghe, 2004, s. 466–475).

Výhodou metódy najväčšieho spádu je, že je univerzálne použiteľná na rôzne optimalizačné úlohy. U konvexných funkcií je každý lokálny extrém zároveň globálnym extrémom, takže nám ani nevadí, že touto metódou nájdeme iba lokálne minimum. V ostatných prípadoch je výsledok citlivý na počiatočnej voľbe koeficientov, preto je lepšie algoritmus niekoľkokrát zopakovať a počiatočné koeficienty zvoliť náhodne. Aby algoritmus skonvergoval, potrebujeme správne zvoliť hodnotu parametra α, ktorá nemusí byť konštantná a môže sa s pribúdajúcimi iteráciami znižovať.

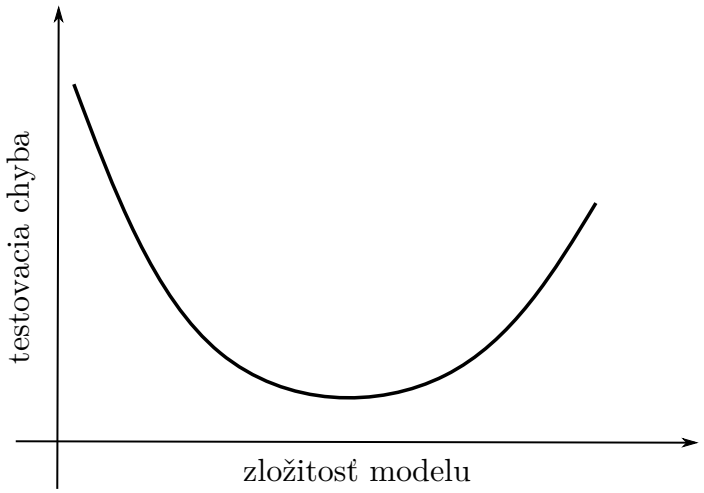
1.4 Regularizácia

V procese strojového učenia zvyčajne dostupné dáta rozdelíme do dvoch disjunktných množín. Na trénovacej vzorke model natrénujeme a následne vyhodnotíme jeho chybovosť na testovacej vzorke. Model, ktorý je príliš zložitý, zachytí štruktúru trénovacích dát veľmi presne, no nedokáže ju zovšeobecniť na inú vzorku. Tomuto javu hovoríme pretrénovanie a existuje niekoľko spôsobov, ako sa mu vyhnúť. Jedným z nich je regularizácia.

Cieľom regularizácie modelu je zníženie väzby medzi vstupnými atribútmi a predpovedanou hodnotou. Inými slovami, chceme penalizovať koeficienty, ak sú príliš veľké, čo sa u stratovej funkcie logistickej regresie prejaví nasledovne

$$L(\beta) = \sum_{i=1}^{n} y_i \left(1 - h(x_i)\right) + (1 - y_i)\, h(x_i) + \lambda \left\|\beta\right\|_k.$$

Tu je $\lambda \geq 0$ parameter modelu a $\|\beta\|_k$ norma vektoru β. Najčastejšie používané sú $k = 1$ (L1 regularizácia) a $k = 2$ (L2 regularizácia). Hlavný rozdiel medzi nimi je v tom, že L1 regularizácia dokáže niektoré koeficienty zmenšiť až na nulu. Atribúty s nulovým koeficientom nemajú žiaden vplyv na výstup, preto ich môžeme z celého procesu učenia vylúčiť.



Obr. 1.2: Všeobecný vzťah testovacej chyby a zložitosti modelu.

Parameter λ vyjadruje silu regularizácie. Voľbou $\lambda = 0$ dostávame pôvodnú stratovú funkciu, s jej rastúcou hodnotou zmenšujeme viazanosť modelu na trénovacie dáta. Pre $\lambda \to \infty$ dostaneme koeficienty blízke nule a model logistickej regresie sa zjednoduší na model, ktorý každému vstupu vráti priemernú výstupnú hodnotu na trénovacích dátach.

1.5 Princíp one-vs-all

Binárna klasifikácia je špeciálnym prípadom klasifikácie, ak výstup nadobúda dve možné hodnoty. Logistickú regresiu však môžeme zovšeobecniť aj na úlohy, kde dáta rozdeľujeme do troch alebo viacerých tried. Princíp *one-vs-all* využíva

binárny klasifikátor na to, aby určil pravdepodobnosť, s akou daný vstup patrí do každej z tried, a nakoniec preň vyberie triedu s najvyššou pravdepodobnosťou.

Uvažujme úlohu, v ktorej výstup y nadobúda hodnoty z množiny $\{0, ..., n-1\}$, jedná sa teda o klasifikáciu do n tried. Postavíme n binárnych klasifikátorov, pričom i-tý klasifikátor rozdeľuje vstupy do skupín $\{i\}$ a $\{0,...,n-1\} \setminus \{i\}$. Teraz využijeme to, že hypotéza logistickej regresie má obor hodnôt interval $[0, 1]$. Ak označíme hypotézu i-tého klasifikátora h_i, tak naša hypotéza ako funkcia vstupu x má tvar

$$h(x) = \underset{i=0...n-1}{\operatorname{argmax}} \; h_i(y = i|x).$$

Existujú aj iné riešenia všeobecnej klasifikačnej úlohy, no všetky najpoužívanejšie postupy ju redukujú na binárnu klasifikáciu. Napríklad princíp *one-vs-one* postaví binárny klasifikátor pre každú dvojicu tried a pre daný vstup vyberie triedu, ktorá je klasifikovaná najčastejšie.

1.6 Vyhodnotenie modelu

Popísali sme základné algoritmy používané v strojovom učení a predstavili sme aj niektoré ich vylepšenia. Doteraz sme však nepovedali, ako modely medzi sebou porovnať a vybrať z nich ten najlepší.

Na začiatku dostupné dáta rozdelíme na testovaciu množinu, ktorú použijeme až na záverečné porovnanie modelov, a vývojový dataset, slúžiaci na ladenie hyperparametrov (t.j. parametrov modelu). Napríklad u logistickej regresie sa rozhodujeme, ktorý typ regularizácie použiť, aká je správna hodnota parametra λ, hranica pri binárnej klasifikácii a podobne. Na to si vývojový dataset opäť rozložíme na trénovacie a testovacie príklady. Pri tom si musíme dať pozor, aby boli pravdepodobnostné rozdelenia príkladov vo všetkých množinách približne rovnaké.

Aby sme mohli určiť chybovosť modelu na testovacích dátach, či už pri ladení jeho hyperparametrov, alebo pri záverečnom vyhodnotení, si potrebujeme zaviesť nejakú metriku, ktorá nám povie, ako dobre model predpovedá testovacie príklady. Samozrejme, pre každý typ úlohy sa používa iná metrika, preto uvedieme niekoľko metrík samostatne pre regresiu a klasifikáciu.

Na analýzu chýb klasifikátora sa používa matica konfúzie. Pri n možných triedach je to matica C veľkosti $n \times n$, pričom na pozícii $C_{i,j}$ je počet tých príkladov patriacich do triedy i, ktoré klasifikátor zaradil do triedy j. Podiel správne zaradených príkladov vyčítame z matice konfúzie jednoducho,

$$\sum_{i=0}^{n-1} C_{i,i} / \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} C_{i,j},$$

je to podiel súčtu čísel na diagonále. Táto metrika sa nazýva presnosť (*accuracy*). Pre každú výstupnú triedu t nás môže ďalej zaujímať, koľko percent príkladov zaradených do t tam naozaj patrí (*precision*), alebo akú časť príkladov z tejto triedy sme do nej zaradili (*recall*). Keď chceme zohľadniť obe skutočnosti, použijeme F1 skóre danej triedy definované ako harmonický priemer týchto dvoch metrík.

Pri regresii predikujeme spojitú veličinu a väčšinou správnu hodnotu netrafíme presne. Preto sa na vyhodnotenie predikcie používa vzdialenosť od skutočnej hodnoty. Na testovacej vzorke tieto vzdialenosti spriemerujeme, čím dostaneme výslednú kvalitu modelu. Stredná absolútna chyba (*mean absolute error*) modelu s hypotézou h je

$$MAE = \frac{1}{n} \sum_{i=0}^{n-1} |h(x_i) - y_i|.$$

V praxi sa ale častejšie používa stredná kvadratická chyba (*mean squared error*), pretože kladie väčší dôraz na tie príklady, ktorých predikcia je od skutočnej hodnoty veľmi vzdialená. Táto metrika má tvar podobný ako stratová funkcia u lineárnej regresie:

$$MSE = \frac{1}{n} \sum_{i=0}^{n-1} (h(x_i) - y_i)^2.$$

1.7 Krížová validácia

Vo všeobecnosti platí, že čím viac dát máme k dispozícii, tým lepší model dokážeme natrénovať. Často je však dostupné len obmedzené množstvo dát, z ktorých ešte nejakú časť vyhradíme na testovanie modelu. Ani podiel testovacích dát nemôže byť príliš malý, aby bol výsledok dostatočne reprezentatívny.

S cieľom ušetriť trénovacie dáta a zároveň poskytnúť dostatok príkladov na vyhodnotenie modelu sa používa krížová validácia (*cross-validation*). Ide o veľmi jednoduchý princíp — vývojové dáta rozdelíme na vopred stanovený počet oddielov rovnakej veľkosti. Viac oddielov znamená viac dát použitých na trénovanie a presnejší výsledok testu, no zároveň sa celý proces vyhodnocovania spomalí.

Každý oddiel použijeme raz ako testovaciu sadu na vyhodnotenie modelu natrénovanom na všetkých zvyšných oddieloch. Výsledky testov na jednotlivých oddieloch spriemerujeme, čím dostaneme konečnú kvalitu modelu.

2. Hlboké učenie

Dnes je už samozrejmosťou, že počítače prekonávajú človeka v mnohých úlohách. Okrem bežných činností nám ich efektivita pomohla vyriešiť otvorené matematické problémy či poraziť najlepšieho hráča šachu. Hlavným rysom týchto úloh je to, že sa dajú ľahko formalizovať a preložiť do sveta počítačov. Naopak, úlohy, ktoré sa zdajú byť banálne a založené skôr na ľudskej intuícii než na logickom myslení, ako napríklad rozlíšiť objekty na obrázku, slová v reči alebo sentiment recenzie, sú pre počítače veľkou výzvou.

Ako sa ľudia dokázali naučiť, ako na obrázku rozoznať psa od mačky a prečo je toto pre počítač ťažká úloha? Čo sa stane, ak sa pokúsime modelovať nervový systém človeka? Tvorcovia neurónových sietí našli inšpiráciu práve v biológii ľudského mozgu. Jeho štruktúra je už po narodení prispôsobená na to, aby si mohol vytvoriť vlastné pravidlá správania na základe skúseností.

Neurónová sieť je zložená z jednoduchých buniek, neurónov, ktoré sú medzi sebou vzájomne poprepájané. Ide o veľmi zjednodušený matematický model fungovania mozgu, v ktorom neurón prijme informáciu od susedných neurónov pomocou svojich dendritov, spracuje informáciu a ak dosiahne určitý potenciál, aktivuje sa a pošle výstupný signál ďalším neurónom. Proces učenia znamená posilňovanie, prípadne oslabovanie synaptických spojení medzi neurónmi.

V hlbokom učení sú tieto neuróny usporiadané do vrstiev. Jednotlivé vrstvy reprezentujú to, ako celá sieť spracúva informácie. Prvá vrstva prijíma vstup po jednoduchých častiach, napríklad obrázok po pixeloch. Každá ďalšia vrstva postaví zložitejší koncept na základoch predchádzajúcej vrstvy. V prípade obrázku sa dostávame od pixelov k hranám, obrysom až po komplexné objekty.

V tejto kapitole sa budeme venovať tomu, ako preložiť biologickú neurónovú sieť do reči matematiky. Začneme jej základnou stavebnou jednotkou, neurónom, z ktorého postavíme orientovaný graf rozdelený na vrstvy. Popíšeme proces učenia siete znova ako optimalizáciu stratovej funkcie a predstavíme niektoré jej drobné vylepšenia. Bližšie informácie o fungovaní neurónových sietí sú dostupné v Goodfellow a kol. (2016) a Haykin (1998), z ktorých čerpá aj táto kapitola.

2.1 Model neurónu

Biologický popis neurónu nám dáva návod, ako postaviť neurón ako výpočtovú jednotku. Táto jednotka musí byť dosť jednoduchá na to, aby sa dala popísať nejakou známou matematickou funkciou, ale zároveň dostatočne komplexná na riešenie úloh strojového učenia.

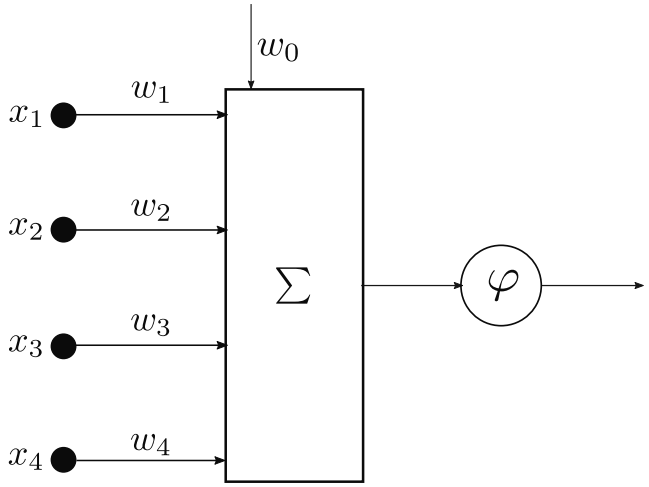
Na vstupe očakáva neurón vektor x predpísanej dĺžky. Spracovanie informácie znamená, že vektor x skalárne vynásobíme vektorom váh w a pričítame k tomu skalár w_0. Tieto váhy sú uložené v tele neurónu a menia sa v procese učenia. Reprezentujú silu synaptických spojení a v celej neurónovej sieti sú váhové vektory jej jedinou pamäťou.

Nakoniec sa neurón aktivuje, čo znamená, že výsledok skalárneho súčinu zobrazí nejakou vopred stanovenou funkciou. Veľmi často sa používa sigmoida, ktorú sme videli aj pri logistickej regresii, no v nasledujúcej sekcii uvedieme ďalšie zaujímavé funkcie. Neurón ako funkcia vstupu x s aktivačnou funkciou sigmoidou má

tvar

$$y = \frac{1}{1 + e^{w \cdot x + w_0}},$$

kde y je výstup neurónu.



Obr. 2.1: Model neurónu so vstupným vektorom dĺžky 4 a aktivačnou funkciou φ.

Ak je neurón súčasťou neurónovej siete, jeho výstup sa zvyčajne posunie ďalej ako vstup ďalším neurónom. Aj samotný neurón nám však môže poslúžiť ako dobrý klasifikátor. Všimnime si, že neurón ako funkcia je totožný s hypotézou logistickej regresie. Namiesto metódy maximálnej vierohodnosti tu však k optimalizácii stratovej funkcie používame tzv. online učenie. To prebieha v niekoľkých iteráciách, kde v každej iterácii dostáva neurón na vstupe jeden trénovací príklad za druhým, pričom si neurón upravuje svoje váhy. Tento algoritmus tu však nebudeme rozoberať detailne, keďže jeho zovšeobecnenú verziu použijeme k trénovaniu celej siete.

2.2 Aktivačné funkcie

Vidíme, že základná stavebná jednotka neurónovej siete je dostatočne silná na modelovanie logistickej regresie. Okrem sigmoidy sa však v neurónových sieťach používajú aj iné aktivačné funkcie. Táto sekcia obsahuje prehľad tých najpoužívanejších.

Ako najjednoduchšiu aktivačnú funkciu môžeme použiť **identitu**. Samostatný neurón s identitou predstavuje prediktor založený na lineárnej regresii. Táto funkcia sa nám môže hodiť, no ak použijeme identitu ako aktivačnú funkciu všetkých neurónov v sieti, celou sieťou získame opäť len lineárnu funkciu.

Ďalšou jednoduchou funkciou, ktorá je v obore kladných čísel totožná s identitou, je **ReLU** (*Rectified Linear Unit*). Narozdiel od identity nám však do siete prináša nelinearitu a v posledných rokoch sa používa dokonca častejšie ako sigmoida. Táto funkcia je definovaná ako

$$relu(x) = max(0, x).$$

Navyše je ReLU inšpirovaná biologickým popisom neurónu; doposiaľ sme vynechali to, že neurón sa aktivuje, iba ak dosiahne určitý potenciál. V prípade ReLU to znamená, že ak je potenciál záporný, neurón sa neaktivuje a ďalej pošle nulu.

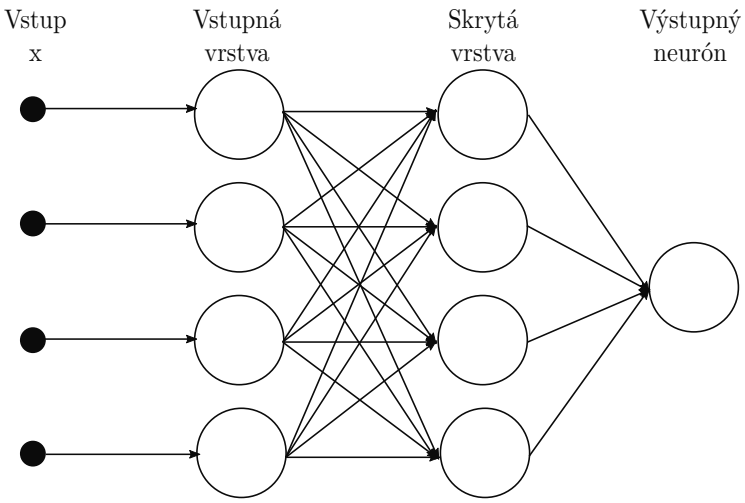
Klasifikáciu do viacerých tried sme pri logistickej regresii redukovali na niekoľko binárnych klasifikačných úloh. V zložitejších modeloch sa hodí zovšeobecnenie sigmoidy, a to funkcia **softmax**. Pre n výstupných tried a vstupný vektor x vráti softmax vektor dĺžky n, ktorého zložky sú reálne čísla z intervalu $[0, 1]$ a ich súčet je 1. Funkcia je definovaná ako

$$softmax(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}.$$

Softmax sa od predchádzajúcich funkcií líši tým, že prijíma vektory dĺžky n, takže nejde o aktivačnú funkciu jedného neurónu, ale celej vrstvy.

2.3 Dopredná neurónová sieť

Teraz, keď už poznáme všetky ingrediencie na stavbu neurónovej siete, si ukážeme jej základnú architektúru. Aj keď je celá sieť opäť len matematická funkcia, alebo výpočetná jednotka, ktorá prijíma vstup a vráti výstupnú hodnotu, najlepšie sa predstavuje ako orientovaný graf.



Obr. 2.2: Príklad doprednej neurónovej siete s jednou skrytou vrstvou.

Vrcholy grafu tvoria neuróny a orientovaná hrana znamená, že medzi danými neurónmi je spojenie. Každý neurón má množinu vstupných a výstupných hrán, tak ako sme to popisovali v modeli neurónu. Navyše, hrany sú ohodnotené váhami a trénovanie siete znamená hľadanie optimálnych váh.

V jednoduchej doprednej neurónovej sieti sú neuróny rozdelené do vzájomne usporiadaných vrstiev. Hranou sú spojené iba dvojice neurónov zo susedných vrstiev, pričom smerujú vždy od nižšej vrstvy k vyššej. Najnižšia, vstupná vrstva, obsahuje vstupný vektor. Zvyčajne nasleduje aspoň jedna skrytá vrstva a najvyššie je výstupná vrstva. Výstup týchto neurónov je výstupom celej siete.

Na trénovanie neurónovej siete sa používa algoritmus spätnej propagácie, ktorý pozostáva z dvoch fáz. Prebieha opäť v niekoľkých iteráciách, príklad po príklade. Predpokladajme, že sieť má na vstupe príklad z trénovacej vzorky.

V doprednej fáze sú váhy hrán zafixované a vstup prechádza sieťou postupne po vrstvách, až sa aktivujú výstupné neuróny a dostaneme výstup. V spätnej fáze sa získaný výstup porovná s požadovanou výstupnou hodnotou. Chybový signál sa posiela naspäť do nižších vrstiev, kde sa upravujú hodnoty váh na jednotlivých hranách.

Algoritmus spätnej propagácie hľadá také váhy, ktoré minimalizujú zvolenú stratovú funkciu pomocou metódy najväčšieho spádu. Existujú tri varianty tejto metódy, ktoré sa líšia v tom, koľko príkladov naraz spracujú (Ruder (2016)). Keď sme optimalizovali stratovú funkciu logistickej regresie, použili sme celý dataset naraz. Naopak, v tejto kapitole sme hovorili o on-line učení príklad po príklade. V skutočnosti sa však používa kompromisná varianta, kedy spätná fáza algoritmu prebehne po spracovaní várky n príkladov. Táto verzia algoritmu sa nazýva *mini-batch gradient descent*.

2.4 Krížová entropia

Od stratovej funkcie neurónovej siete chceme, aby sa strata datasetu dala napísať ako priemer strát jednotlivých trénovacích príkladov, a navyše požadujeme, aby to bola funkcia závislá iba od výstupu neurónov vo výstupnej vrstve siete.

Ak je neurónová sieť určená pre regresiu, môžeme použiť ako stratovú funkciu strednú kvadratickú chybu. Pri klasifikačných úlohách sa nám ponúka niekoľko možností. Najjednoduchšie je použiť podiel nesprávne klasifikovaných príkladov (t.j. klasifikačnú chybu), no podobne ako v logistickej regresii to nie je vhodná voľba pre trénovanie modelu.

Výstupom klasifikátora sú pravdepodobnosti, že daný vstup patrí do danej triedy, pričom klasifikujeme do triedy s najvyššou pravdepodobnosťou. Použitím klasifikačnej chyby strácame akúkoľvek informáciu o tom, ako sú tieto pravdepodobnosti rozložené a či sa učíme tým správnym smerom.

Tento problém rieši krížová entropia. Jej špeciálny prípad pre dve triedy sme už odvodili z metódy maximálnej vierohodnosti u logistickej regresie. Krížová entropia pre n tried a daný vstup x má tvar

$$H(x) = -\sum_{i=1}^{n} y_i \, log_2(h(x)_i),$$

kde y je jednotkový vektor s jednotkou na pozícii prislúchajúcej triede, do ktorej patrí vstup x, a $h(x)$ je výstup klasifikátora — vektor predikovaných pravdepodobností. Krížová entropia celej vzorky je priemerom cez všetky príklady.

2.5 Algoritmus ADAM

Metóda najväčšieho spádu je veľmi dobrým základom v optimalizácii, no sama osebe má veľa problémov. Výpočet gradientu je pri funkciách stoviek premenných veľmi drahá operácia, ktorá sa v hlbokých neurónových sieťach používa príliš často. To sa v praxi rieši stochastickou verziou tejto metódy (*stochastic gradient*

descent), kde sa gradient odhaduje podľa podmnožiny trénovacej sady, veľkosti niekoľko sto príkladov. Táto podmnožina je vybraná náhodne a v každej iterácii sa zmení.

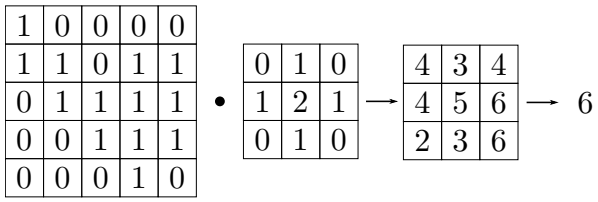
SGD je už dostatočne rýchly, no stále existujú nevyriešené otázky. Prvou je voľba veľkosti kroku. Príliš malý krok vedie k pomalej konvergencii, naopak veľkým krokom by sme mohli preskočiť optimum a neskonvergovať nikdy. Pri riedkych dátach sa dostaneme k zmene niektorých parametrov častejšie ako pri iných, preto ich je lepšie upravovať v rôznom rozsahu. Poslednou otázkou zostáva, ako optimalizovať nekonvexné funkcie, ktoré môžu mať viacero lokálnych optím.

Metóda ADAM (*Adaptive Moment Estimation*), popísaná v Kingma a Ba (2014), je heuristika, ktorá v praxi veľmi dobre rieši vyššie spomenuté problémy. Detaily algoritmu sú nad rámec tejto práce, no ide o adaptívnu metódu založenú na momentovej metóde. Tá sa od SGD líši tým, že si okrem lokálneho gradientu pamätá aj predchádzajúce kroky. Jeden krok je potom lineárnou kombináciou gradientu v aktuálnom bode a predchádzajúceho kroku. Adaptívne metódy sa hodia práve na riedke dáta, pretože viac upravujú menej frekventované parametre.

2.6 Konvolúcia

Doprednú neurónovú sieť teraz rozšírime o novú vrstvu. Táto vrstva predspracuje vstup siete tzv. konvolúciou a svoj výstup pošle po aktivovaní ďalšej vrstve. Konvolučné neurónové siete dosiahli najväčší úspech v klasifikácii obrázkov a na tejto úlohe si vysvetlíme princíp konvolúcie.

Majme na vstupe zadaný obrázok vo forme dvojrozmernej matice. Na túto maticu predtým, ako príde na vstup neurónom, aplikujeme filter. Filter je menšia matica, napríklad veľkosti 5×5, ktorú posúvame postupne po vstupnom obrázku. Filter sa skalárne vynásobí s prekrývajúcou sa časťou vstupu a výsledok sa uloží do novej matice. Napríklad, ak je filtrom matica obsahujúca samé jednotky, pixely obrázku v každom regióne veľkosti 5×5 spriemerujeme. Týchto filtrov použijeme niekoľko stoviek aj v rôznych veľkostiach a výsledok filtra zobrazíme aktivačnou funkciou, napríklad ReLU. Na jej výstup sa aplikuje *pooling*, čo môže byť napr. spriemerovanie prvkov matice alebo výber maxima.



Obr. 2.3: Aplikácia filtra 3x3 a max–pooling na čiernobielom obrázku.

Svoje miesto si konvolučné siete našli aj v počítačovej lingvistike. Slová vo vete môžeme reprezentovať vektormi reálnych čísel (viď Sekcia 3.2) a usporiadať ich do matice. Filtre, tentokrát už fixnej šírky, postupne aplikujeme na skupiny susedných slov vo vete. Opäť použijeme niekoľko rôznych veľkostí filtrov a na výstup každého filtra aplikujeme *pooling*, čím dostaneme vektor fixnej dĺžky ako výstup, aj keď vstupné vety môžu mať rôzny počet slov.

Hodnoty filtrov sa sieť naučí tak isto ako váhy hrán. Okrem rôznych veľkostí filtrov môžeme meniť ďalšie parametre konvolúcie. Musíme sa rozhodnúť, či filter aplikujeme len na platné regióny vstupu, alebo vstupný vektor doplníme zľava aj sprava nulami tak, aby sme mohli začať na prvom slove. Ďalším parametrom je dĺžka kroku, t.j. o koľko slov posunieme filter v jednom kroku.

2.7 Dropout

Hlboké neurónové siete dokážu modelovať aj veľmi komplikované vzťahy medzi vstupmi a ich výstupmi, avšak aj tu pri nedostatku dát dochádza k pretrénovaniu. U logistickej regresie sme tento problém vyriešili regularizáciou pomocou penalizačnej funkcie. Učenie neurónovej siete zas zastavíme v iterácii, kedy sa testovacia chyba začne zväčšovať. Srivastava a kol. (2014) však predstavujú aj iný spôsob, ako pretrénovaniu predísť.

Architektúru neurónovej siete budeme počas trénovania upravovať. Vopred si zvolíme pravdepodobnosť prerušenia p. Pred spracovaním príkladu zjednodušíme sieť tak, že každý neurón v skrytej vrstve dočasne odstránime s pravdepodobnosťou p, nezávisle na zvyšných neurónoch, a váhy necháme pôvodné.

Po dotrénovaní už k sieti pristúpime trochu inak. Všetky neuróny budú v sieti prítomné, takže architektúra bude pôvodná. Naopak, upravíme váhy hrán tak, že každú z nich vynásobíme $1 - p$. Potom je stredná hodnota výstupu každého neurónu v trénovacej fáze rovná hodnote výstupu v testovacej fáze.

Užitočnosť tejto metódy je nielen v riešení problému pretrénovania. Na záverečnú sieť sa dá pozerať ako na kombináciu zjednodušených sietí, z ktorých je síce každá iná, ale zároveň medzi sebou zdieľajú váhy. Kombinácia rôznych modelov sa v strojovom učení osvedčila, najmä ak ide o modely, ktoré sa od seba výrazne líšia. Uvedená technika je zaujímavým prístupom, ako skombinovať neurónové siete bez nutnosti explicitného trénovania odlišných architektúr.

3. Extrakcia atribútov

Pri všetkých doteraz popísaných modeloch sme predpokladali, že na vstupe dostaneme číselný vektor. Aby sme tieto modely mohli použiť na textové dáta, potrebujeme nájsť spôsob, ako na číselné vektory previesť vety. Od tejto operácie zároveň požadujeme, aby sme stratili čo najmenšie množstvo informácie. Od jednoduchého modelu, ktorý zráta počet jednotlivých slov vo vete, prejdeme k reprezentácii slov vektormi, ktoré majú navyše veľmi zaujímavé vlastnosti.

3.1 N-gramový model

Predpokladajme, že na vstupe máme českú vetu, napríklad

Člověk pochopí sám, až jej uvidí a vyslechne.

Tak ako sú obrázky zložené z pixelov, aj vetu rozdelíme na malé jednotky, tzv. tokeny. Ako základné stavebné jednotky vety sa bežne používajú znaky alebo slová, v tejto kapitole sa budeme venovať práve slovám ako tokenom.

Prvým krok je tokenizácia textu, t.j. rozdelenie viet na tokeny. Obvykle sa interpunkčné znamienka zanedbávajú alebo ich môžeme chápať ako samostatné tokeny. Ďalej môžeme vetu predspracovať tak, že všetky písmená prevedieme na malé, prípadne odstránime diakritiku. Po takto zvolenej tokenizácii pôvodnej vety dostaneme zoznam slov

[*clovek, pochopi, sam, az, jej, uvidi, a, vyslechne*].

Nasleduje vektorizácia slov, pri ktorej máme k dispozícii slovník — buď vopred daný slovník všetkých slov daného jazyka, alebo pri väčšom množstve textu si tento slovník vyrobíme zo vstupných dát. Každé slovo nahradíme *one-hot* vektorom, ktorý má jednotku na pozícii daného slova v slovníku. V najjednoduchšom modeli, nazvanom *bag-of-words*, je vektor reprezentujúci vetu alebo blok textu prostý súčet *one-hot* vektorov pre jednotlivé slová.

Výsledný vektor má ale dve zásadné negatíva. Dĺžka vektora je rovná veľkosti slovníka, ktorá býva rádovo v desaťtisícoch, a to aj po predchádzajúcom predspracovaní, ktoré sme popísali vyššie. Predstavme si napríklad logistickú regresiu. Aby sa model správne naučil niekoľko desiatok tisíc parametrov, potrebuje k tomu primerané množstvo dát. Čiastočným riešením je vynechanie slov, ktoré sa v dátach vyskytujú príliš často (*stopwords*), v češtine sú to najmä spojky, predložky a ďalšie bezvýznamové slová. Podobne, málo frekventované slová je tiež vhodné zo slovníka vynechať.

Ďalším nedostatkom takejto reprezentácie je strata informácie o poradí slov. Vo výslednom vektore je na pozícii i počet výskytov i-teho slova slovníka vo vete. Porovnajme napríklad vetu

„Moc nudný film, navíc ani nebyl nijak originální“,

s vetou, ktorá má rovnakú vektorovú reprezentáciu ako veta s opačnou polaritou

„Moc originální film, navíc ani nebyl nijak nudný“.

Riešením je pozerať sa na n-tice za sebou idúcich slov, tzv. n-gramy. Doteraz sme si pamätali iba výskyt unigramov, no ak k tomu pridáme bigramy, trigramy či dlhšie úseky slov, pôvodné usporiadanie vety sa už dá dopočítať väčšinou jednoznačne. Tu ale narastá dĺžka výstupného vektora s veľkosťou slovníka ešte rýchlejšie, preto je mimoriadne dôležité brať do úvahy iba frekventované n-gramy.

Problémy sme z menšej časti vyriešili, no stále platí, že ak chceme zachovať čo najviac informácie z textu, potrebujeme na to vektory veľkej dĺžky. Tieto vektory by sa nám hodilo zobraziť do priestoru nižšej dimenzie, na čo využijeme neurónovú sieť.

3.2 Embedding

Úloha znie jednoducho — previesť slová na vektory čísel danej dĺžky. Navyše môžeme využiť už ktorúkoľvek dostupnú reprezentáciu slova, t.j. buď ako zoznam znakov, *one-hot* vektor alebo index slova v slovníku. V princípe by nám stačila akákoľvek rozumná funkcia z množiny takýchto reprezentácií do $\mathbb{R}^d$ pre zvolenú dimenziu výstupného vektora d. Takýmto prevodom by sme oproti pôvodnej reprezentácii však nič nezískali. Našim cieľom bude nájsť takú funkciu, ktorá zachová syntaktické a sémantické vzťahy medzi slovami a ich obrazmi.

Mikolov a kol. (2013) predstavili veľmi efektívny spôsob hľadania tejto funkcie, tzv. *continuous skip-gram model*. Vytvoríme neurónovú sieť s jednoduchou architektúrou, ktorá rieši úplne inú úlohu. Tou úlohou je pre vstupné slovo predpovedať okolité slová vo vete.

Trénovanie prebieha na veľkom množstve viet. Pre každé slovo sa pozrieme na niekoľko predchádzajúcich a rovnaký počet nasledujúcich slov vo vete. Dvojice týchto slov použijeme ako trénovacie príklady pre neurónovú sieť. Napríklad z pôvodnej tokenizovanej vety [*clovek, pochopi, sam, az, jej, uvidi, a, vyslechne*] pre slovo *jej* dostaneme pri veľkosti okna 2 tieto dvojice:

(jej, sam),
(jej, az),
(jej, uvidi),
(jej, a).

Vzdialenejším dvojiciam sa dáva menšia váha a výstupom siete je pravdepodobnosť, že sa druhé slovo v dvojici vyskytuje v blízkosti prvého slova.

Neurónová sieť má na vstupe *one-hot* vektor, prvé slovo z dvojice. Nasleduje skrytá vrstva s d neurónmi, kde d je dimenzia požadovaného vektora, a za ňou výstupná vrstva s aktivačnou funkciou softmax, rovnakej veľkosti ako vstup. Výstup je zoznam pravdepodobností, že náhodne vybrané slovo v okolí vstupného slova je práve slovo príslušné výstupnému neurónu.

Keď sa pozrieme bližšie na skrytú vrstvu, vidíme, že funguje presne ako požadovaná funkcia, zobrazuje *one-hot* vektory na reálne vektory dimenzie d (*word embeddings*). Zachovanie vzťahov medzi slovami je vlastnosť tejto funkcie, ktorá vyplýva z definície našej úlohy. Uveďme si na niekoľkých príkladoch, prečo je to tak.

Prvou dôležitou vlastnosťou embeddingov je, že podobné slová sa zobrazia na podobné vektory. Podobnosť vektorov určujeme pomocou kosínusovej podobnosti, ktorá je pre dva vektory $u, v \in \mathbb{R}^d$ definovaná ako kosínus uhla medzi nimi,

$$sim(u, v) = \frac{u \cdot v}{|u|\,|v|}.$$

Napríklad synonymá sa v trénovacej vzorke vyskytujú v rovnakom kontexte, čo znamená, že výstupné pravdepodobnosti pre jednotlivé slová sú u synoným podobné. Preto je prirodzené, že výstup skrytej vrstvy je podobný pre synonymá či slová vyskytujúce sa zvyčajne v rovnakom kontexte, takže blízko seba budú vektory reprezentujúce farby, mestá, štáty a podobne.

Ďalšou zaujímavosťou je zachovanie vzťahu medzi dvojicami slov, ktorý je zachytený v rozdieli vektorov. Takýmto vzťahom je napríklad superlatív, takže platí, že vektor

$$W(nejvetsi) - W(velky) + W(maly)$$

je s vysokou pravdepodobnosťou najbližšie vektoru

$$W(nejmensi),$$

kde W je zobrazujúca funkcia. Okrem morfo-syntaktických vzťahov ako minulý čas a množné číslo, sú v rozdieli vektorov zachytené aj sémantické vzťahy typu muž–žena či hlavné mesto štátu.

Model *skip-gram* nie je jediným modelom prezentovaným v Mikolov a kol. (2013), no dosahuje najvyššiu presnosť v zachovaní sémanticko–syntaktických vzťahov, a to aj v porovnaní s predchádzajúcimi dostupnými modelmi. Zároveň jeho autori poukazujú na dôležitosť veľkosti trénovacích dát, na natrénovanie siete použili dataset obsahujúci viac než 700 miliónov slov.

Aj napriek tomu, že embeddingy slov majú pozoruhodné vlastnosti, pre účely sentimentálnej analýzy by sa nám hodila ešte jedna vlastnosť, ktorá im chýba. Slová opačnej polarity sú často syntakticky totožné a vyskytujú sa v podobných kontextoch, takže ich odpovedajúce vektory sa zobrazia blízko seba. Ak by to bolo naopak, samotné embeddingy by takmer vyriešili predpovedanie postoja z textu. Stále sa však môžeme držať pôvodného plánu, a to použiť predspracované vektory ako vstup zložitejším modelom.

4. Popis úlohy

Ako sme už uviedli, postojová analýza rieši niekoľko úloh spojených s identifikovaním postoja autora, emocionálneho významu textu či detekcie subjektivity. Cieľom tejto práce je okrem postoja autora predpovedať aj jeho intenzitu v špecifickej doméne — filmových recenziách.

Výhodou tejto domény je najmä dobrá dostupnosť dát pre učenie s učiteľom, pretože textové recenzie sú často spojené s číselným hodnotením filmu. To nám umožňuje preskočiť jednu z prvých fáz procesu strojového učenia, manuálnu anotáciu dát, ktorá býva časovo náročná a závislá na subjektivite autora. V tejto kapitole popíšeme, ako sme vytvorili dataset diváckych recenzií, v krátkosti ho zanalyzujeme a uvedieme predchádzajúce experimenty na podobných datasetoch.

4.1 Príprava dát

Česko-Slovenská filmová databáza obsahuje vyše tri a pol milióna verejne dostupných komentárov k päťsto tisíc filmom. Táto databáza však nie je dostupná v ľahko spracovateľnom formáte, preto na získanie recenzií využívame presne definovanú štruktúru jej webových stránok.

Pre predstavu uvádzame príklad recenzie spolu s jej metadátami, ako súčasť zdrojového kódu stránky konkrétneho filmu:

```
<h5 class="author">
       <a href="/uzivatel/75-gouryella/">
       gouryella
       </a>
</h5>

<img src=".../stars/5.gif"
       class="rating"
       width="40"
       alt="*****"
/>

<p class="post">
       Tenhle film je jako zjeveni z jineho sveta -
       skoda, ze se takovych netoci vice.
       <span class="date desc">(7.6.2002)</span>
</p>
```

Z recenzie v tomto formáte extrahujeme textové hodnotenie a počet hviezdičiek ako hodnotu atribútu *alt* tagu *img*. Ako sme neskôr zistili, tento postup nebol bezchybný a do triedy s nula hviezdičkami sa dostali aj recenzie, ktorým číselné hodnotenie chýbalo. Ak by ich bolo veľa, mohlo by to spochybňovať vhodnosť datasetu na našu úlohu, no ako uvidíme v nasledujúcej sekcii, našťastie všetkých nulových recenzií nie je viac ako 6%.

Navyše sme pridali aj identifikátor autora recenzie zložený z unikátneho čísla a mena užívateľa, názov filmu a jeho žáner, ktoré sú v uložené v úvodnej časti

zdrojového kódu. Tieto údaje síce v práci nevyužívame, no mohli by byť dôležité pre iné úlohy, napríklad pre odporúčacie systémy. Po spracovaní vyššie uvedeného zdrojového kódu dostaneme nasledovný výsledok:

```
<rating
       genre="Drama / Krimi"
       movie="Vykoupeni z veznice Shawshank"
       stars="5"
       user="75-gouryella">
       Tenhle film je jako zjeveni z jineho sveta -
       skoda, ze se takovych netoci vice.
</rating>
```

Výsledný dataset tvorí dohromady 10000 recenzií, ktoré sme náhodne vybrali tak, aby boli čo najmenej viazané na konkrétne filmy a zároveň zostalo zachované rozdelenie číselného hodnotenia. Pre každý z rokov 1987 až 2016 sme vybrali 500 najviac hodnotených filmov vydaných v danom roku. Týmto sme získali 15000 filmov, pre každý z nich sme vybrali prvú recenziu z prehľadovej stránky o filme. Z týchto komentárov sme vyfiltrovali české texty a nakoniec sme odstránili recenzie najstarších filmov tak, aby zostalo 10000 recenzií.

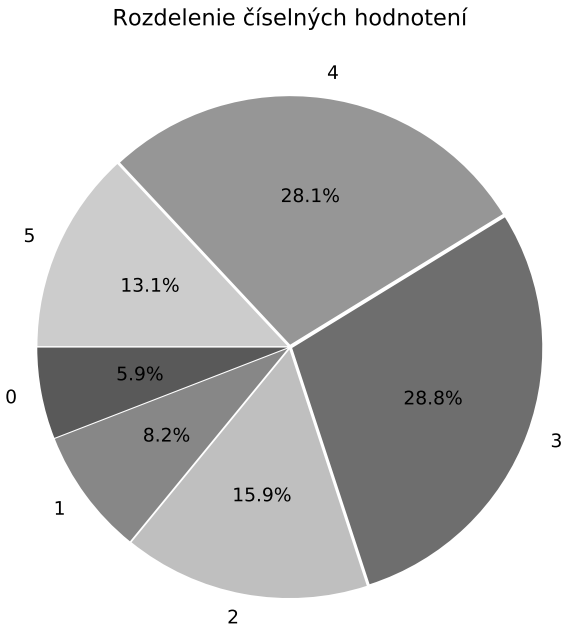
Na odstránenie komentárov v inom jazyku ako českom (najmä v slovenčine) sme použili knižnicu *langdetect* (Shuyo, 2010), konkrétne jej port do jazyka Python. Táto knižnica využíva naivný bayesovský klasifikátor na detekciu vyše päťdesiatich jazykov s viac ako 99% presnosťou, vrátane češtiny a slovenčiny.

4.2 Popis datasetu

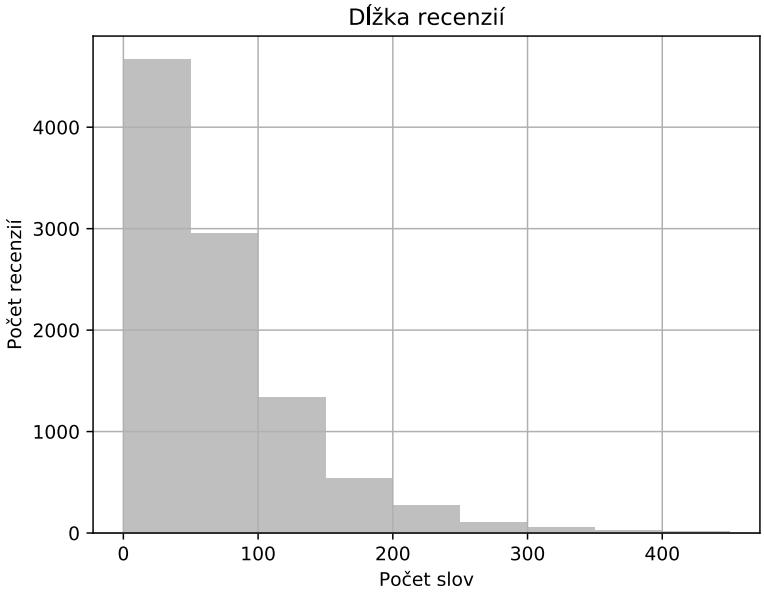
Každá recenzia obsahuje dva údaje, ktoré sú pre našu úlohu dôležité — text a počet hviezdičiek. Na základe textu ako vstupnej hodnoty sa pokúsime predpovedať počet hviezdičiek, ktorý predstavuje intenzitu negatívneho či pozitívneho postoja autora.

Počet hviezdičiek sa pohybuje na škále od 0 do 5, kde 0 znamená najhoršie hodnotenie. Výstup teda nadobúda šesť rôznych hodnôt. Veľmi dôležitou informáciou je rozdelenie počtu hviezdičiek v datasete, ktoré indikuje aj obtiažnosť úlohy. Porovnajme napríklad rovnomerné rozdelenie s prípadom, kedy približne 90% recenzií má hodnotenie 3 — je zrejmé, ktorá úloha je ťažšia. Diváci nepoužívajú extrémne hodnotenia veľmi často, preto ani rozdelenie nášho datasetu nie je rovnomerné. Presné rozdelenie je zobrazené na obrázku 4.1.

Druhým dôležitým faktorom je dĺžka komentárov. Príliš krátkym komentárom môže postoj autora chýbať, čo ilustrujú recenzie „Zůstaň sám sebou.“, „Temperamentní Španělé.“ a „Už bylo hodně pozdě.“, nachádzajúce sa v datasete. Naopak, u neprimerane dlhých hodnotení sa zas často prelínajú pozitívne a negatívne segmenty, pričom celkový sentiment textu je ťažké odhadnúť. Z obrázka 4.2 vidíme, že väčšina recenzií je kratších ako sto slov a až na niekoľko extrémnych prípadov môžeme považovať 300 slov ako maximálnu dĺžku recenzie.



Obr. 4.1: Graf rozdelenia číselných hodnotení v datasete.



Obr. 4.2: Histogram počtu slov v recenziách.

4.3 Predchádzajúce experimenty

Zaujímavosťou detekcie intenzity postoja je to, že na túto úlohu sa dá pozerať z dvoch rôznych pohľadov. Na jednej strane ide o klasifikačnú úlohu, kde príklady rozdeľujeme do šiestich rôznych kategórií. Tieto kategórie sú však navzájom usporiadané a majú numerickú hodnotu, takže predikciu hodnotenia môžeme riešiť aj regresiou.

Tieto prístupy porovnali Gupta a kol. (2010) na datasete obsahujúcom hodnotenia reštaurácií. Ich výsledky naznačujú, že modely určené pre klasifikáciu prekonávajú regresiu, a to dokonca aj pri porovnaní ich strednej absolútnej chyby.

Ďalej uvádzame práce, ktoré na riešenie podobných úloh využívajú konvolučné neurónové siete. Kim (2014) predstavuje rôzne modely založené na konvolučných neurónových sieťach a porovnáva ich úspešnosť na siedmich datasetoch z rôznych domén. Vo väčšine prípadov prekonávajú dovtedajšie najlepšie výsledky.

Na rovnakej množine datasetov (a ďalších dvoch) testuje hyperparametre konvolučnej neurónovej siete Zhang a Wallace (2015). Výsledkom práce sú všeobecné odporúčania k používaniu aktivačných funkcií, regularizácie, poolingu a ďalších parametrov u týchto modelov.

V ďalšom článku (dos Santos a de C. Gatti, 2014) sa zas autori zameriavajú na extrakciu atribútov na úrovni znakov, slov a dokonca aj viet. Tento prístup aplikovali na dvoch datasetoch, obsahujúcich tweety a filmové hodnotenia.

Všetky tri články pracujú s nejakou variantou datasetu Stanford Sentiment Treebank (Socher a kol., 2013), približne rovnakej veľkosti ako náš dataset. Hlavný rozdiel je v tom, že obsahuje iba päť výstupných tried a namiesto celých recenzií tvoria dataset anglické vety, ktoré sú z recenzií extrahované.

Pretože je dataset podobný nášmu, v tabuľke úvadzame výsledky zo spomínaných troch článkov. Naším cieľom je vyvinúť model, ktorý dosiahne úspešnosť na takejto úrovni.

Autor	**Presnosť najlepšieho modelu v %**
Kim	48,00
Zhang a Wallace	47,08
dos Santos a de C. Gatti	48,30

Tabuľka 4.1: Výsledky modelov na datasete SST

5. Experiment

Táto kapitola je venovaná návrhu a ladeniu modelov určených na detekciu intenzity postoja. Na základe doterajších výsledkov k úlohe pristupujeme ako ku klasifikácii do šiestich tried.

Pripravený dataset sme na začiatku náhodne rozdelili na vývojovú vzorku veľkosti 9000 recenzií a testovaciu vzorku s 1000 recenziami. Na vyladenie všetkých popísaných modelov používame desaťnásobnú krížovú validáciu, pričom maximalizujeme ich presnosť (*accuracy*). V závere kapitoly vyladené modely porovnávame na testovacej vzorke.

5.1 Základný model

Analýza datasetu, špeciálne rozdelenia počtu hviezdičiek, nám dala informáciu o zložitosti úlohy. Túto analýzu teraz prevedieme na vývojových dátach, aby sme vytvorili tzv. *baseline* model, t.j. triviálny model určený na dolný odhad úspešnosti.

Hodnotenie	**Podiel v %**
0	5,98
1	8,14
2	16,02
3	28,96
4	27,71
5	13,19

Tabuľka 5.1: Rozdelenie číselných hodnotení vo vývojom datasete

Rozdelenie číselných hodnotení, zobrazené v tabuľke 5.1, sa podobá na rozdelenie v celom datasete. To znamená, že náhodné rozdelenie dát prebehlo v poriadku. Ako *baseline* navrhujeme klasifikátor, ktorý každému príkladu priradí 3 hviezdičky. Takýto model dosahuje na vývojovej vzorke presnosť 28,96%.

5.2 Logistická regresia

Ďalej popíšeme klasifikátor založený na logistickej regresii, ktorý neskôr použijeme k analýze jazykových prostriedkov použitých v jednotlivých triedach recenzií.

Použitiu modelu predchádza extrakcia atribútov, ktorú sme previedli nasledujúcim spôsobom. Najprv sme v textoch recenzií nahradili veľké písmená malými. Takýto text sme tokenizovali, čiže rozdelili na slová definované ako súvislé alfanumerické reťazce, za využitia NLTK tokenizátora (Loper a Bird, 2002). Na získanie číselných atribútov zo spracovaného textu sme použili n-gramový model. Text takto vektorizujeme, pričom sa obmedzíme na unigramy a bigramy. Navyše slovník obsahuje iba také unigramy a bigramy, ktoré majú dokumentovú frekvenciu nanajvýš 0,7, čím sme odstránili korpusovo závislé *stopwords*. Veľkosť slovníka sme obmedzili na päťdesiat tisíc, odstránili sme najmenej frekventované n-gramy.

Získaný vektor používame ako vstup modelu logistickej regresie, implementovanej v knižnici *scikit-learn* (Pedregosa a kol., 2011). Klasifikáciu do šiestich tried riešime princípom *one-vs-all*. Na tomto modeli sme otestovali vplyv použitej regularizácie (s *l1* a *l2* penalizáciou) a tiež to, ako zníži presnosť zmenšenie veľkosti slovníka. Výsledky získané krížovou validáciou uvádzame v tabuľke 5.2.

Veľkosť slovníka	**l1**	**l2**
1000	41,54	40,80
10000	44,11	43,56
50000	44,63	44,00

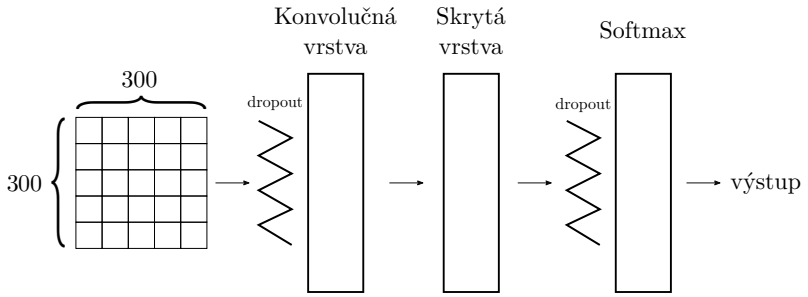
Tabuľka 5.2: Presnosť logistickej regresie pre rôzne veľkosti slovníka a použité penalizačné funkcie

Na záverečné porovnanie modelov a analýzu jazykových prostriedkov použijeme logistickú regresiu s *l1* regularizáciou a veľkosťou slovníka 50000.

5.3 Návrh architektúry neurónovej siete

Predchádzajúca kapitola obsahuje niekoľko výsledkov, ktoré ukazujú úspešnosť konvolučných neurónových sietí v postojovej analýze. V tejto sekcii navrhneme sieť určenú špeciálne pre našu úlohu.

Recenzie predspracujeme podobne ako pri logistickej regresii, akurát namiesto n-gramového modelu použijeme predtrénované embeddingy pre zobrazenie slov na vektory. Pretože neurónová sieť očakáva vstup fixnej dĺžky, zvolíme dĺžku recenzií 300 slov. Kratšie recenzie doplníme nulovými vektormi a koniec dlhších recenzií zahodíme. Skrátenie textu však nie je problém, pretože sa týka menej ako 1% recenzií.



Obr. 5.1: Architektúra použitej neurónovej siete.

Po vstupnej vrstve nasleduje konvolučná vrstva s niekoľkými filtrami veľkosti 4. Filtre aplikujeme len na platné regióny vstupu a max–poolingom dosahujeme konštantnú dĺžku jej výstupu. Aktivačnou funkciou tejto vrstvy je $ReLU$. Ďalej nasleduje skrytá vrstva s aktivačnou funkciou sigmoidou, ktorej výstup sa posunie *softmax* vrstve. Výstupom je trieda s najvyššou pravdepodobnosťou.

Na hrany vstupujúce do konvolučnej a *softmax* vrstvy aplikujeme dropout s rovnakou pravdepodobnosťou prerušenia. Pri trénovaní je stratovou funkciou

krížová entropia a na jej minimalizáciu používame algoritmus ADAM. Implementácia siete je jednoduchá vďaka knižnici Keras (Chollet a kol., 2015).

Model s jednoduchou konvolučnou vrstvou porovnáme s modelom, ktorý používa filtre rôznych veľkostí. Pre druhý model sme zvolili filtre veľkosti 2, 3 a 5, a to z každého druhu rovnaký počet filtrov. Na nájdenie vhodných parametrov sme okrem odporúčaní zo spomínaných článkov použili grid-search, čo je prehľadanie všetkých kombinácií z množiny hodnôt parametrov.

5.4 Trénovanie embeddingov

V práci sme použili embeddingy natrénované na českej Wikipédii za použitia vylepšeného skip-gram modelu, ktorý popisujú Bojanowski a kol. (2016). Ich vlastnosti predvedieme na niekoľkých príkladoch.

francie	**zelená**	**km**	**xbox**	**mnoho**
durancie	modrá	kilometrů	playstation	několik
francii	zazelená	jihozápadně	xboxu	spoustu
francouzska	žlutá	severozápadně	playstationu	spousta
itálie	červená	severovýchodně	gamecube	omnoho
belgie	zelenomodrá	kilometru	konzoli	řadu
anglie	modrozelená	kilometrům	nintendo	spousty
francouzsku	rudozelená	jihovýchodně	dreamcast	nemnoho
tourcoing	zezelená	západně	konzole	množství
bourgueil	trávozelená	kilometry	wii	spoustě
británie	černozelená	kilometr	nintendu	spoustou

Tabuľka 5.3: Desať najbližších embeddingov pre niektoré slová

Tabuľka 5.3 zobrazuje desať najbližších vektorov po zobrazení slov *Francie, zelená, km, XBOX* a *mnoho*. Vidíme, že sa podobné slová naozaj zobrazujú na blízke vektory. Táto implementácia skip-gram modelu dáva väčšiu váhu morfologickým vlastnostiam slov ako sémantickým.

Stále však zostávajú zachované aj sémantické vlastnosti slov. Ak označíme W funkciu zobrazujúcu slová na vektory, tak v tomto natrénovanom modeli naozaj platí, že

$$W(\textit{Francie}) - W(\textit{Paříž}) \approx W(\textit{Itálie}) - W(\textit{Řím}).$$

Zachytené sú aj povolania známych osobností,

$$W(\textit{Einstein}) - W(\textit{vědec}) \approx W(\textit{Jágr}) - W(\textit{hokejista}),$$

či stupňovanie prídavných mien,

$$W(\textit{velký}) - W(\textit{větší}) \approx W(\textit{chladný}) - W(\textit{chladnější}).$$

Použité embeddingy majú zaujímavé vlastnosti, ktoré sú spôsobené aj natrénovaním na veľkom korpuse. Môžeme však zvoliť aj iný prístup, a to namiesto použitia predtrénovaných vektorov získať embeddingy priamo z nášho datasetu. Síce je jeho veľkosť omnoho menšia, takže výsledné vektory by mali byť menšej kvality, no mohli by mať vlastnosti dôležité pre túto špecifickú úlohu. Navyše trénovanie embeddingov môžeme vložiť do navrhnutej neurónovej siete tak,

že pred vstupnú vrstvu pridáme ďalšiu vrstvu, ktorá zobrazuje one-hot vektory na embeddingy stanovenej dimenzie.

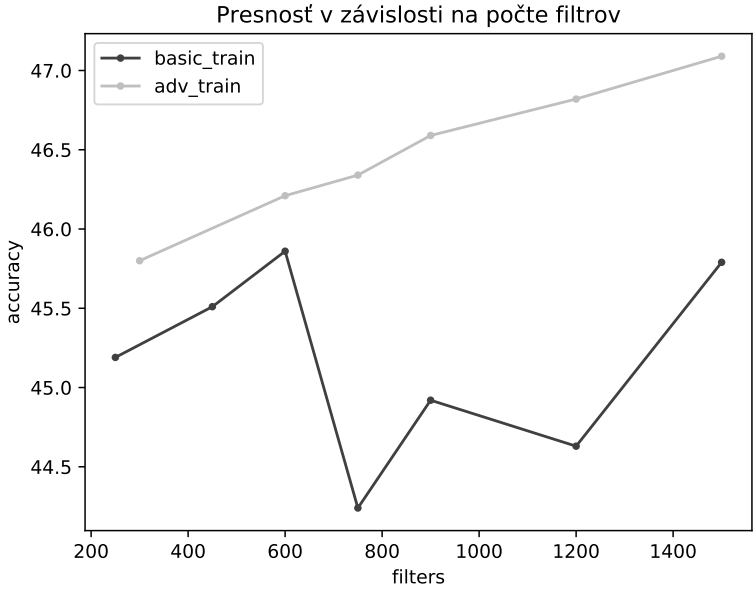
Nič nezabraňuje ani kombinácii týchto dvoch prístupov. Embeddingová vrstva je v princípe tabuľka, ktorá definuje embeddingy slov. Na začiatku jej trénovania je inicializovaná nulami, prípadne náhodne. Takúto inicializáciu môžeme nahradiť predtrénovanou tabuľkou embeddingov, ktorá sa trénovaním siete ešte upraví pre túto konkrétnu úlohu.

5.5 Vplyv veľkosti konvolučnej vrstvy

V tejto chvíli máme k dispozícii šesť rôznych modelov, ktorých základnú architektúru popisuje obrázok 5.1. Tri modely majú všetky filtre veľkosti 4, zvyšné tri modely majú filtre veľkostí 2, 3 a 5, a to z každej veľkosti rovnaký počet. Trojice modelov sa odlišujú v rôznych prístupoch k trénovaniu embeddingov.

Každý model sme nahrubo odladili pomocou grid-search a na vývojových dátach sa zdá, že inicializácia embeddingovej vrstvy predtrénovanými vektormi a jej následné korpusovo závislé trénovanie prekonáva zvyšné dva prístupy. Výrazne zaostáva statické používanie predtrénovaných vektorov, ktoré sa ukazuje byť na použitých dátach nevhodné.

Teraz sa zameriame na presnejšie vyladenie modelov s najlepším prístupom k embeddingom. Konkrétne, u oboch modelov zisťujeme vplyv počtu filtrov v konvolučnej vrstve, ktorý sme doteraz vôbec neladili. Výsledky zobrazuje graf 5.2.



Obr. 5.2: Vplyv počtu filtrov na presnosť modelu.

Najlepší počet filtrov sme následne použili aj u zvyšných modelov.

Kvôli tomu, že konvolúcia prijíma vektory fixnej dĺžky, sme stanovili dĺžku recenzií na 300 a chýbajúce slová sme doplnili prázdnymi vektormi. Čo by sa ale stalo, ak by sme všetky recenzie skrátili na prvých 5, resp. 10 slov? To sme otestovali u najlepšieho modelu (t.j. s 3 veľkosťami filtrov a embeddingovou vrstvou

inicializovanou predtrénovanými embeddingmi) s pôvodným počtom filtrov. Došli sme k zaujímavému záveru, a to, že aj prvých niekoľko slov recenzie niekedy stačí na detekciu číselného hodnotenia. Presnosť sa samozrejme o niečo zhoršila, sledovanie prvých desiatich slov znížilo presnosť modelu o 7,38%; ak sme sa pozerali iba na prvých päť, presnosť sa zhoršila oproti pôvodnému modelu o 8,09%. Hlavným dôvodom takéhoto zníženia presnosti bolo orezanie záveru recenzie, ktorý tiež býva veľmi informatívny.

5.6 Výsledné modely

Po hrubom odladení všetkých popísaných modelov a presnejšom hľadaní optimálneho počtu konvolučných filtrov sme dostali parametre, s ktorými jednotlivé modely dosahujú najvyššiu presnosť. Hodnoty týchto parametrov sú uvedené v tabuľke 5.4. Prvá časť názvu modelu zodpovedá veľkostiam filtrov (*basic* pre jednu veľkosť, *adv* pre 3 rôzne veľkosti), druhá časť prístupu k embeddingom (*zero* nepoužíva predtrénované embeddingy, *nontr* áno, ale ďalej ich netrénuje a *train* je kombináciou týchto prístupov).

Parameter/Model	**basic zero**	**basic nontr**	**basic train**	**adv zero**	**adv nontr**	**adv train**
Veľkosť várky	4	4	32	4	32	32
Počet iterácií	2	2	3	2	3	3
Max. dĺžka recenzie	300	300	300	300	300	300
Veľkosť slovníka	10000	20000	20000	20000	10000	20000
Dimenzia embeddingov	100	300	300	200	300	300
Počet filtrov	600	600	600	1500	1500	1500
Veľkosť filtrov	4	4	4	2, 3, 5	2, 3, 5	2, 3, 5
Šírka skrytej vrstvy	250	250	250	250	250	250
Dropout	0.2	0.4	0.2	0.4	0.2	0.4

Tabuľka 5.4: Konfigurácie odladených modelov.

5.7 Záverečné porovnanie modelov

Model	**Presnosť v %**	**MSE**
baseline	27,4	1,788
linear	42,4	1,686
basic_zero	48,4	1,558
basic_nontr	41,6	1,561
basic_train	45,8	1,542
adv_zero	43,8	1,701
adv_nontr	38,9	1,765
adv_train	47,7	1,434

Tabuľka 5.5: Výsledky odladených modelov na testovacích dátach.

Dohromady sme získali osem modelov, a to baseline, logistickú regresiu a šesť modelov založených na konvolučnej neurónovej sieti. S parametrami uvedenými

v predchádzajúcej sekcii sme ich následne natrénovali na celej vývojovej vzorke a v tejto sekcii ich vyhodnotíme na testovacej vzorke.

Doposiaľ sme pre každý model maximalizovali jeho presnosť. Pre objektívne porovnanie modelov je dôležité si uvedomiť, že existuje niekoľko druhov chýb, ktoré môže klasifikátor spraviť. Ak sa pomýli o jednu triedu, presnosť sa zníži rovnako, ako keď recenzii s veľmi negatívnym hodnotením priradí päť hviezdičiek. Kvôli tomu u všetkých modelov sledujeme aj strednú kvadratickú chybu (*MSE*). Prehľad ich výsledkov obsahuje tabuľka 5.5.

Konvolučné neurónové siete porazili logistickú regresiu, no ukázalo sa, že použiť predtrénované embeddingy a ďalej ich netrénovať je horšie nielen v porovnaní s ostatnými prístupmi, ale aj s jednoduchým modelom, akým je logistická regresia. Na prístupe k embeddingom teda veľmi záleží a pre každý z nich je vhodné zvoliť iné hodnoty parametrov. Presnosť našich najlepších modelov dosiahla podobné hodnoty ako konvolučné siete v predchádzajúcich experimentoch na datasete SST.

6. Analýza recenzií

Navrhli sme a porovnali niekoľko modelov založených na odlišných metódach. S využitím predtrénovaných embeddingov a konvolučných neurónových sietí sme dosiahli úspešnosť porovnateľnú s najnovšími výsledkami v postojovej analýze. Aj napriek tomu naše modely predpovedajú intenzitu postoja vo viac ako polovici prípadov nesprávne. V tejto kapitole sa pozrieme na to, aké chyby robí náš model najčastejšie, pokúsime sa nájsť ich príčinu a porovnáme použité jazykové prostriedky pre jednotlivé triedy recenzií.

6.1 Analýza chýb

Model *basic_zero* dosiahol na testovacích dátach najvyššiu presnosť, zatiaľ čo stredná kvadratická chyba modelu *adv_train* bola výrazne najnižšia. Tieto dva modely teraz detailnejšie preskúmame, na čo využijeme ich matice konfúzie získané pri testovaní, uvedené v tabuľkách 6.1 a 6.2.

Skutočnosť/Predikcia	**0**	**1**	**2**	**3**	**4**	**5**
0	**17**	18	6	5	2	6
1	4	**41**	24	7	10	2
2	2	36	**50**	35	18	5
3	1	12	28	**152**	70	11
4	1	20	12	69	**179**	35
5	2	4	2	12	64	**38**

Tabuľka 6.1: Matica konfúzie modelu *adv_train*.

Už na prvý pohľad vidíme rozdielnosť týchto modelov. Zatiaľ čo *adv_train* rozdeľuje recenzie celkom vyvážene, *basic_zero* výrazne preferuje triedu 4, do ktorej klasifikuje takmer polovicu príkladov.

Skutočnosť/Predikcia	**0**	**1**	**2**	**3**	**4**	**5**
0	**14**	12	6	7	12	3
1	6	**37**	16	12	13	4
2	2	15	**51**	29	43	6
3	3	7	13	**120**	123	8
4	5	5	11	40	**225**	30
5	2	3	1	5	74	**37**

Tabuľka 6.2: Matica konfúzie modelu *basic_zero*.

Ak by sme u oboch modelov tolerovali chybu ± 1 triedu, presnosť modelu *adv_train* by stúpla na 86%, u modelu *basic_zero* by dosiahla 84,2%. To znamená, že väčšina chýb nie je veľmi závažná a znamená pochybenie o jednu triedu. Ak by sa nám podarilo presne rozdeliť recenzie s tromi hviezdičkami od štyroch, znamenalo by to zlepšenie o 14 až 16 percent. To je však jeden z najväčších problémov tejto úlohy. Rozlíšenie týchto dvoch tried je náročné aj pre človeka, zvlášť v prípade, ak sa číselné hodnotenia nezhodujú s textom. Pre ilustráciu obtiažnosti tejto úlohy uvádzame dve recenzie:

Sakra úletový film, kočka proti psovi, více žánrů dohromady, to všechno celkem zábavný koktejl.

Roztodivné a zároveň roztomilé peklíčko...

Aj keď to tak na prvý pohľad nevyzerá, prvá recenzia s veľmi pozitívnym textom má tri hviezdičky, zatiaľ čo druhá štyri.

Model *basic_zero* tiež klasifikuje niekoľko negatívnych recenzií do triedy 4 či 5 a naopak. Ide už o závažnú chybu, preto sa pozrieme na niekoľko takých príkladov.

Recenzii s jednou hviezdičkou

Tento film mě moc zklamal. Zhruba ve třetíne jsem už věděla, jaká bude poenta a i přesto jsem doufala, že se třeba spletu a nebo že v tom prostě bude ještě něco víc. Ale nebylo! Ani vynikající herecký výkon obou hlavních představitelů nic nezachránil. Škoda

klasifikátor priradil päť. Tento text je celkom náročný, pretože obsahuje pozitívne aj negatívne segmenty. Mätúce je najmä spojenie „vynikající herecký výkon“. Naopak, aj u textu s hodnotením nula hviezdičiek, ktorý tak aj na prvý pohľad pôsobí, sa klasifikátor pomýlil o štyri triedy:

První díl byl humus a druhý díl pokračuje v nastolené odpadní vlně.

Niekedy sa zas neurónová sieť zdá byť inteligentnejšia ako samotný autor komentára. Tejto recenzii s piatimi hviezdičkami predpovedala len jednu:

Díky bohu za internet se svou volnou morálkou pro zobrazování čehokoliv. Z tohoto by se dramaturgové na Kavčích horách asi sesypali a ministr pro národnostní menšiny by následně označil tohle dílo za nešťastné využívání koncesionářských poplatků. V době, kdy se každý ustaraně ohlíží přes rameno, aby snad někoho neurazil, oprašuje tahle dvojka svoje legendární lekce němčiny a opět se dívá na svět tím pravým árijsko-sudetským pohledem.

6.2 Analýza pomocou tf-idf

Naším cieľom je teraz porovnať, ako sa s intenzitou hodnotenia menia jazykové prostriedky použité v recenziách a aké sú hlavné odlišnosti medzi jednotlivými triedami po lexikálnej stránke.

Jednoduchým prístupom je vytvorenie štatistiky, ktorá každému slovu priradí jeho frekvenciu výskytu v danej množine textov. Presnejšie, definujeme štatistiku **tf** (*term frequency*) ako:

tf(slovo) = (počet výskytov slova v texte) / (počet všetkých slov v texte).

Pre každú zo šiestich tried recenzií vypočítame túto štatistiku a porovnáme slová, ktorá majú najvyššiu hodnotu *tf*. Je však pravdepodobné, že vo všetkých triedach by najfrekventovanejšie slová boli približne rovnaké a touto analýzou by sme nezískali žiadnu informáciu okrem množiny *stopwords* v českom jazyku.

Aby sme vo výsledku získali najdôležitejšie slová špecifické pre danú triedu, potrebujeme dať menšiu váhu slovám, ktoré sa vyskytujú vo všetkých triedach. Táto váha sa nazýva **idf** (*inverse document frequency*) a je definovaná ako

$$idf(slovo) = log\frac{N}{df(slovo)} \ ,$$

kde N je počet všetkých tried a $df(slovo)$ je počet tried obsahujúcich slovo. Štatistika **tf–idf** je súčinom *tf* a *idf* a použijeme ju k lexikálnej analýze recenzií.

0	**1**	**2**
lev	přeskakuji	nezachrání
medvěd	nezachrání	35
glóbus	nanic	nudí
křišťálový	cher	průměr
uctívám	jackie	nezajímavé

3	**4**	**5**
průměr	75	perfektní
55	perfektní	dokonalý
65	atmosférou	nejoblíbenější
neurazí	spokojenost	paterson
titule	nudit	nádherný

Tabuľka 6.3: Najfrekventovanejšie slová v jednotlivých triedach.

Pre predstavu uvádzame v tabuľke 6.3 pre všetky triedy päť slov s najväčšou hodnotou *tf-idf*. Tabuľka s väčším počtom slov je súčasťou príloh tejto práce.

Prekvapivé sú najdôležitejšie slová v recenziách bez hviezdičky. Názvy filmových cien sa tam vyskytujú preto, že trieda s nula hviezdičkami obsahuje aj recenzie, ktorým číselné hodnotenie pôvodne chýbalo, viď sekcia 4.1.

Medzi ďalšími frekventovanými slovami v triedach 0 a 1 sú rôzne negatívne slová a najmä vulgarizmy. Recenzie s dvoma a troma hviezdičkami zas obsahujú množstvo neutrálnych slov, ako nezachrání, nuda, průměr, nezajímavé, neurazí či nenadchne. Niektoré recenzie taktiež obsahujú percentuálne hodnotenie filmu priamo v texte. Táto informácia samozrejme uľahčuje ich klasifikáciu, no ani na základe nej nemôžeme predpovedať triedu jednoznačne. Napríklad 85% je jedným z najdôležitejších slov ako u štvorhviezdičkových, tak aj u päťhviezdičkových recenzií.

6.3 Analýza modelu logistickej regresie

Konvolučná neurónová sieť s vhodnými parametrami identifikovala intenzitu postoja autora najlepšie zo všetkých modelov. Ani logistická regresia však za najlepším modelom veľmi nezaostávala a jej výhodou oproti neurónovým sieťam je možnosť model ľahko analyzovať. Funkcia zobrazujúca vstupný vektor atribútov na výstupnú triedu je jednoduchá a navyše atribúty reprezentujú počet výskytov jednotlivých slov, resp. n-gramov v texte.

V našom prípade je natrénovaný model logistickej regresie zložený zo šiestich binárnych klasifikátorov, pričom každý z nich je popísaný hypotézou — lineárnou funkciou, na ktorú je aplikovaná sigmoida. Ak označíme $h(x)$ hypotézu, x_i vstupnú premennú znamenajúcu počet výskytov slova, resp. n-gramu w_i vo vstupnom texte a β_i koeficient pri tejto premennej, tak pridanie jedného slova w_i spô-

sobí zväčšenie hodnoty

$$log\frac{h(x)}{1-h(x)}$$

o β_i. Presná interpretácia tejto hodnoty v tomto prípade ani nie je potrebná, dôležité je, že n-gramy môžeme rozdeliť na tie, ktorých výskyt výstupnú pravdepodobnosť zvyšuje, znižuje či na ňu nemá žiaden vplyv. Navyše, podľa hodnoty koeficientov β_i môžeme usporiadať všetky n-gramy a vybrať z nich tie najdôležitejšie.

Táto vlastnosť logistickej regresie nám umožňuje previesť podobnú analýzu ako podľa štatistiky *tf-idf*. Výhodou *tf-idf* bola jej nezávislosť na modeli, v tomto prípade sa zas môžeme pozrieť na to, čo sa model naučil a ktoré slová považuje za dôležité. Využijeme aj vlastnosť *l1* regularizácie, a to, že väčšina koeficientov je rovná nule. Slov s nenulovým koeficientom tak nezostáva veľa (800 až 3000, v závislosti od triedy) a môžeme ich bližšie preskúmať.

0	**1**	**2**
napadá mě	2 10	4 10
medvědi	odpadu	40
zhovadilosti	zmatená	druhá za
kterého by	se úplně	35
hovně	výsměch	kecám

3	**4**	**5**
6 10	8 10	kdy jsou
55	75	10 10
60	tak originální	politiku
těšila	7 10	nebyl vůbec
65	čtyřka	volbou

Tabuľka 6.4: Najdôležitejšie slová a bigramy získané logistickou regresiou.

V tabuľke 6.4 uvádzame prehľad najdôležitejších pozitívnych slov a bigramov pre jednotlivé triedy. Kompletná tabuľka sa nachádza v prílohe práce.

Z tabuľky vidíme, že model sa naozaj naučil číselné hodnotenia obsiahnuté v texte. Recenzie obsahujúce číselné hodnotenie priamo v texte (buď ako číslovku, percentá alebo hodnotenie na škále od 1 do 10) tvoria dohromady približne až 29,6% všetkých recenzií. Ak by sa na základe tohto číselného hodnotenia dal predpovedať počet hviezdičiek, už v spojení s baseline klasifikátorom by sme mohli dosiahnuť presnosť okolo 50%. Bohužiaľ, číselné hodnotenie v texte neurčuje jednoznačne počet hviezdičiek. Presnejšie povedané, klasifikátor založený na pravidlách, ktorý recenziám bez číselného hodnotenia v texte priradí 3 hviezdičky (najfrekventovanejšiu triedu) a zvyšným recenziám priradí počet hviezdičiek podľa čísla v texte, dosiahol na vývojových dátach presnosť 33,66%, čo je oproti baseline zlepšenie iba o 4,7%.

Naopak, klasifikátor nepovažuje za dôležité názvy filmových cien, namiesto toho sa u nízkych hodnotení naučil vulgarizmy. Niektoré bigramy, ako napríklad „napadá mě“ či „kdy jsou“, sú až prekvapivo dôležité. Sami osebe nemajú žiaden emocionálny význam, preto predpokladáme, že ich dôležitosť je spôsobená menšou

veľkosťou datasetu. Pri niekoľkonásobne väčšom množstve dát by sa ich náhodný frekventovanejší výskyt v jednej triede eliminoval a klasifikátor by ich nepovažoval za také dôležité.

	0	**1**	**2**	**3**	**4**	**5**
pozitívne	7	5	10	7	14	15
negatívne	5	3	9	11	8	1

Tabuľka 6.5: Počet superlatív medzi kladnými a zápornými atribútmi v jednotlivých triedach.

Ak tieto čudné slová vynecháme, u päťhviezdičkových recenzií opäť objavíme veľmi pozitívne slová, ako geniální, fascinující či vynikající. Špeciálnu kategóriu tvoria superlatívy, ktorých je medzi päťdesiatimi najdôležitejšími atribútmi u 5 hviezdičiek hneď niekoľko. Väčšinou platí, že slovo začínajúce predponou nej- je superlatív (výnimkou je napríklad slovo „nejasný“). Túto aproximáciu sme použili na odhad počtu superlatívov medzi nenulovými atribútmi u jednotlivých tried, čo zobrazuje tabuľka 6.5. Superlatívy sú najviac zastúpené v recenziách s 5 hviezdičkami a takisto pomáhajú oddeliť štvorhviezdičkové recenzie od tých negatívnejších.

Záver

V tejto práci sme sa venovali automatickej detekcii intenzity postoja v textoch krátkej až strednej dĺžky. Na tieto účely sme z užívateľských hodnotení filmov na webe Česko-Slovenskej filmovej databázy vytvorili dataset, pričom sme sa snažili zachovať pravdepodobnostné rozdelenie hodnotení a odstrániť závislosť recenzií na konkrétnych filmoch. Tieto recenzie pôvodne obsahovali ďalšie metadáta ako meno užívateľa a informácie o filme. Modelovanie užívateľov a filmov by určite zlepšilo naše výsledky, no filmové recenzie nám slúžili len ako ľahko dostupný dataset určený na testovanie použitých metód.

Na riešenie úlohy sme mohli použiť množstvo metód strojového učenia. Aktuálne výsledky experimentov v oblasti postojovej analýze naznačovali, že tento typ úlohy v súčasnosti najlepšie zvládajú konvolučné neurónové siete. Takúto sieť sme implementovali a porovnali s jednoduchým modelom založeným na logistickej regresii. Zamerali sme sa aj na extrakciu atribútov z textových dát, konkrétne sme otestovali tri rôzne prístupy k ich prevodu na vektory reálnych čísel.

Výsledky experimentov viedli k trom hlavným pozorovaniam. Po prvé, potvrdila sa úspešnosť konvolučných neurónových sietí na tejto úlohe, no na druhej strane ani jednoduchý model, akým je logistická regresia, nedopadol oveľa horšie.

Druhým dôležitým výsledkom je, že používanie vopred natrénovaných embeddingov bez ich ďalšieho trénovania nie je vhodným spôsobom, ako zobraziť slová na reálne vektory. Omnoho lepšie výsledky dosiahli tie modely, v ktorých embeddingy ďalej trénujeme, či už boli inicializované nulami alebo predtrénovanými vektormi.

Analýza natrénovaného modelu logistickej regresie nám umožnila porovnať jazykové prostriedky použité v recenziách s rôznym stupňom hodnotenia. Tretím pozorovaním je, že dôležité slová a slovné spojenia sa medzi jednotlivými triedami recenzií výrazne líšili, okrem číselných hodnotení obsiahnutých priamo v texte, ktoré sa vyskytovali u všetkých tried.

Naše najlepšie modely dosiahli presnosť porovnateľnú s výsledkami najnovších experimentov na podobnej úlohe. Z troch uvedených prác dosiahli na podobnom datasete, Stanford Sentiment Treebank, najvyššiu presnosť 48,30% dos Santos a de C. Gatti (2014). Presnosť nášho najlepšieho modelu bola 48,40%.

Aj keď stopercentná presnosť pravdepodobne nie je dosiahnuteľná, stále je čo zlepšovať. Trénovanie modelov je na bežných procesoroch výpočtovo náročné. Ak by sme mali k dispozícii výkonný grafický procesor, mohli by sme ho využiť k spracovaniu niekoľkonásobne väčšieho množstva dát, jemnejšie odladiť parametre modelov, spracovať text na úrovni znakov či získané modely rôzne kombinovať.

Vývoj všetkých modelov prebiehal na textoch v špecifickej doméne — filmových recenziách. Natrénované modely by pravdepodobne dopadli horšie, ak by sme ich otestovali na textoch z iných oblastí, napríklad na hodnoteniach produktov, reštaurácií či tweetoch. Ďalšou výzvou je preto vývoj modelov, ktoré by boli doménovo nezávislé.

Zoznam použitej literatúry

Bojanowski, P., Grave, E., Joulin, A. a Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Boyd, S. a Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press, New York, NY, USA. ISBN 0521833787.

Chollet, F. a kol. (2015). Keras. `https://github.com/fchollet/keras`.

dos Santos, C. N. a de C. Gatti, M. A. (2014). Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*.

Goodfellow, I., Bengio, Y. a Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Gupta, N., Di Fabbrizio, G. a Haffner, P. (2010). Capturing the stars: Predicting ratings for service and product reviews. In *Proceedings of the NAACL HLT 2010 Workshop on Semantic Search*, SS '10, pages 36–43, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1867767.1867772`.

Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition. ISBN 0132733501.

James, G., Witten, D., Hastie, T. a Tibshirani, R. (2014). *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated. ISBN 1461471370, 9781461471370.

Kim, Y. (2014). Convolutional neural networks for sentence classification. *CoRR*, **abs/1408.5882**. URL `http://arxiv.org/abs/1408.5882`.

Kingma, D. P. a Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, **abs/1412.6980**. URL `http://arxiv.org/abs/1412.6980`.

Loper, E. a Bird, S. (2002). Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1118108.1118117. URL `http://dx.doi.org/10.3115/1118108.1118117`.

Mikolov, T., Chen, K., Corrado, G. a Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, **abs/1301.3781**. URL `http://arxiv.org/abs/1301.3781`.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition. ISBN 0070428077, 9780070428072.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg,

V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. a Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, **12**, 2825–2830.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *CoRR*, **abs/1609.04747**. URL http://arxiv.org/abs/1609.04747.

Shuyo, N. (2010). Language detection library for java. URL http://code.google.com/p/language-detection/.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. a Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D13-1170.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. a Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, **15**(1), 1929–1958. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=2627435.2670313.

Zhang, Y. a Wallace, B. C. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR*, **abs/1510.03820**. URL http://arxiv.org/abs/1510.03820.

Zoznam obrázkov

Zoznam tabuliek

Zoznam použitých skratiek

ADAM adaptive moment estimation

MAE mean absolute error (stredná absolútna chyba)

MLE maximum likelihood estimation (metóda maximálnej vierohodnosti)

MSE mean squared error (stredná kvadratická chyba)

NLTK natural language toolkit

ReLU rectified linear unit

SGD stochastic gradient descent

SST Stanford Sentiment Treebank

tf-idf term frequency – inverse document frequency

Prílohy

Užívateľská dokumentácia

Skripty boli testované na verzii Pythonu 3.5 a požadujú nainštalované tieto moduly:

- NumPy
- Pandas
- NLTK
- Scikit-learn
- Theano
- Keras (bežiaci na Theano)

Trénovanie a testovanie modelov spúšťame skriptom **evaluate.py**, nachádzajúci sa v priečinku **models**, s nasledovnými parametrami:

```
python3 evaluate.py model test_cv gs_eval log param
```

Popis parametrov skriptu:

model názov modelu

test_cv *cv* pre krížovú validáciu na vývojových dátach, *test* pre záverečné testovanie modelu

gs_eval *gs* pre grid search (použiteľný iba pri krížovej validácii), *eval* pre vyhodnotenie modelu s konkrétnymi parametrami

log názov logovacieho súboru, relatívne k priečinku **log**

param názov súboru s parametrami, relatívne k priečinku **parameters**

Príklad použitia skriptu pre záverečné otestovanie modelu založeného na logistickej regresii:

```
python3 evaluate.py linear test eval linear.txt linear.txt
```

Modely sú detailne popísané v texte práce, k dispozícii sú tieto názvy modelov:

linear logistická regresia

basic_zero CNN s filtrami jednej veľkosti, bez predtrénovaných embeddingov

basic_train CNN s filtrami jednej veľkosti, s predtrénovanými embeddingmi a trénovateľnou embedding vrstvou

basic_nontr CNN s filtrami jednej veľkosti, s predtrénovanými embeddingmi a netrénovateľnou embedding vrstvou

adv_zero CNN s filtrami rôznych veľkostí, bez predtrénovaných embeddingov

adv_train CNN s filtrami rôznych veľkostí, s predtrénovanými embeddingmi a trénovateľnou embedding vrstvou

adv_nontr CNN s filtrami rôznych veľkostí, s predtrénovanými embeddingmi a netrénovateľnou embedding vrstvou

Súbor s parametrami modelu ma presne definovanú štruktúru (viď súbor **default_parameters.txt** v priečinku **parameters**). Každý riadok obsahuje meno parametra a jeho hodnoty oddelené medzerou. Pri vyhodnotení modelu s konkrétnymi parametrami sa berie do úvahy len prvá hodnota, pri grid search sa vyskúšajú všetky kombinácie parametrov.

Popis parametrov (a konštánt) modelov CNN:

batch_size veľkosť várky pri trénovaní siete

epochs počet iterácií pri trénovaní siete

pretrained_emb_dims dimenzia predtrénovaných embeddingov, konštanta

num_categories počet výstupných kategórií, konštanta

folds počet oddielov, konštanta

maxlen maximálna dĺžka recenzie

dict_size veľkosť slovníka

embedding_dims dimenzia embeddingov, ak nie sú použité predtrénované

filters počet filtrov

kernel_size veľkosť filtra, ak sú použité filtre jedinej veľkosti

hidden_dims dimenzia skrytej vrstvy

dropout_rate pravdepodobnosť prerušenia hrany

Popis parametrov (a konštánt) modelu logistickej regresie:

num_categories počet výstupných kategórií, konštanta

folds počet oddielov, konštanta

dict_size veľkosť slovníka

max_gram n také, že na vektorizáciu sa použijú unigramy až n-gramy

penalty penalizačná funkcia použitá k regularizácii, možné hodnoty *l1* a *l2*

Logovacie súbory obsahujú výsledky modelov, pri grid search sa uložia výsledky každej kombinácie parametrov a matica konfúzie najlepšieho modelu.

Vývojová dokumentácia

Priložené skripty a dáta majú nasledovnú adresárovú štruktúru:

dataset Dataset a skripty na jeho vytvorenie

- **create_dataset.py** skript na vytvorenie datasetu filmových recenzií
- **split_test_train.py** skript rozdeľujúci dataset na vývojové a testovacie dáta
- **data.xml** celý dataset, 10 000 recenzií
- **train.xml** vývojový dataset, 9 000 recenzií
- **test.xml** testovací dataset, 1 000 recenzií

log logovacie súbory

parameters súbory s parametrami modelov

analysis podklady k analýze

- **positive.txt** 50 najdôležitejších kladných atribútov logistickej regresie pre jednotlivé počty hviezdičiek
- **negative.txt** 50 najdôležitejších záporných atribútov logistickej regresie pre jednotlivé počty hviezdičiek
- **tf_idf_1.txt** 50 najviac frekventovaných unigramov pre jednotlivé počty hviezdičiek podľa TF-IDF
- **tf_idf_2.txt** 50 najviac frekventovaných unigramov a bigramov pre jednotlivé počty hviezdičiek podľa TF-IDF

models skripty

- **csfd.py** pomocné funkcie používané vo všetkých skriptoch — načítanie a spracovanie dát, vektorizácia, rozdelenie do oddielov pri krížovej validácii, spracovanie matice konfúzie, načítanie parametrov modelu
- **embedding.py** pomocné funkcie k spracovaniu predtrénovaných embeddingov
- **evaluate.py** trénovanie a vyhodnotenie modelov, použitie bližšie popísané v užívateľskej dokumentácii
- **analysis_linear.py** analýza podľa atribútov logistickej regresie
- **analysis_tfidf.py** analýza podľa TF-IDF
- **csfd_model_LR.py** model logistickej regresie — definícia triedy
- **csfd_model_CNN.py** model konvolučnej neurónovej siete — definícia triedy

Popis modelov a výsledky analýz sa nachádzajú v texte práce, takisto aj informácie o datasete. Obe triedy modelov, **ModelLR** aj **ModelCNN** majú rovnaké štyri metódy:

___init___() konštruktor; vytvorí sa objekt a parametre modelu sa uložia do jeho atribútov

create_model() definícia architektúry modelu, vracia objekt model

cv() krížová validácia modelu na vývojových dátach, výsledky sa uložia do atribútov modelu confusion_matrix, accuracy a mse

test() testovanie modelu na testovacích dátach, výsledky sa uložia do atribútov modelu confusion_matrix, accuracy a mse

Adresár **dataset** obsahuje okrem vyššie popísaných súborov aj textový súbor s názvom **dataset_embeddings**, z ktorého sa načítavajú predtrénované embeddingy. Vznikol z pôvodného veľkého slovníka embeddingov (Bojanowski a kol., 2016) odstránením slov, ktoré sa v datasete nevyskytovali.