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

Diplomová práce

Klasifikace dokumentů s použitím hierarchické reprezentace

Plzeň 2018 Jakub Morávka

Místo této strany bude zadání práce.

Prohlášení

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

V Plzni dne 26. června 2018

Jakub Morávka

Abstract

Recently, distributed representations of words or longer pieces of text have gotten quite an attention. In contrast with traditional representations like *Bag-of-words*, distributed representations have the advantage of being able to capture semantic meaning. The main goal of this work was to try document classification via neural networks, which is built upon hierarchical distributed representations of documents. In case of hierarchical representation, document is represented as a collection of its segments (e,g. paragraphs or sentences). There is an assumption that representation of pieces of text at higher hierarchy level (than just words) can capture more abstract semantic meaning. That might lead to a document representation of better quality and thus increase the accuracy of classification. The results of conducted experiments show that some hierarchical distributed representations achieve better accuracy than most of the non-hierarchical ones, although they do not outperform some of the available results of other approaches.

Abstrakt

Distribuovaným reprezentacím slov či větších úseků textu se v posledních několika letech dostává poměrně velké pozornosti. Na rozdíl od tradičních reprezentací typu *Bag-of-words*, je výhodou distribuovaných reprezentací schopnost zachytit sémantický význam. Cílem této práce bylo vyzkoušet klasifikaci neuronovými sítěmi, postavenou nad hierarchickou distribuovanou reprezentací dokumentů. Při hierarchické reprezentaci je dokument reprezentován po částech jeho přirozené hierarchické struktury (např. dokument $\rightarrow$ odstavce $\rightarrow$ věty $\rightarrow$ slova). Předpoklad je takový, že reprezentace částí textu ve vyšší úrovni hierarchie by mohla zachytit abstraktnější sémantický význam. Díky tomu by, co se týče úspěšnosti klasifikace, mohla celková reprezentace dokumentu být kvalitnější. Dosažené výsledky ukazují, že některé hierarchické distribuované reprezentace většinou poráží nehierarchické distribuované reprezentace a tudíž mají potenciál, přestože nepřekonaly některé dostupné výsledky jiných prací.

Obsah

Seznam obrázků

Seznam tabulek

Seznam vzorců

1 Úvod

S rostoucím množstvím elektronických dokumentů vzrůstá také potřeba tyto dokumenty nějakým způsobem třídit. Manuální kategorizace každého dokumentu je časově velice náročná záležitost a proto je důležitá automatická kategorizace dokumentů. Klasifikace (kategorizace) dokumentů je jednou z mnoha úloh zpracování přirozeného jazyka. Cílem této úlohy je rozhodnout, do které kategorie z definované množiny kategorií dokument patří.

Aby bylo možné s dokumenty nakládat, je nutné reprezentovat je ve vhodné formě – jako vektory reálných čísel. Běžný způsob je *Bag-of-words* reprezentace. Při této reprezentaci je dokument reprezentován vektorem, jehož položky odpovídají četnostem (mohou být různě vážené) slov v dokumentu. Přes svoji jednoduchost je tato metoda hojně používána různými algoritmy klasifikace (k-nejbližších sousedů, Naivní Bayesův klasifikátor, *Support vector machines*) a dosahuje dobrých výsledků. Problémem těchto reprezentací je vysoká dimenzionalita; délka vektoru je rovna velikosti slovníku a vektor má na většině pozic nuly.

V poslední době je veliká pozornost věnována také reprezentacím pomocí nízko-dimenzionálních, hustých vektorů. Existuje mnoho metod pro výpočet těchto vektorů, mezi nimi například různé algoritmy maticové faktorizace a redukce dimenzionality, nebo populární modely založené na neuronových sítích. Důležitou skupinou těchto modelů jsou distribuované reprezentace, které jsou nejčastěji založené na úkolu nalézt takovou reprezentaci slova, která zachytí jeho sémantický význam na základě kontextu, ve kterém se toto slovo vyskytuje. Některé tyto metody také (na rozdíl od *Bag-of-words*) berou v úvahu pořadí, v jakém se slova v textu vyskytují.

Tato práce se zabývá klasifikací novinových článků do velkého počtu kategorií, kdy jeden článek může náležet více kategoriím zároveň. Pro vytvoření systému, který dokáže dokumentu přiřadit kategorie, je nutné mít trénovací data. K dispozici jsou dvě datové sady dokumentů s přiřazenými kategoriemi. První je sada českých novinových článků „Czech text document corpus v2.0“ České tiskové kanceláře, druhá sada je poměrně rozsáhlá kolekce anglických novinových článku „Reuters corpus volume 1“ zpravodajské agentury Reuters.

Hlavním cílem práce je vyzkoušet, jestli má smysl reprezentovat dokument pomocí distribuovaných reprezentací za využití přirozené hierarchie dokumentu. Při hierarchické reprezentací se dokument nechápe jako složený z jednotlivých slov, ale z větších kusů textu, např. z odstavců či vět. Předpoklad je zde takový, že reprezentace na vyšší úrovni hierarchie zachytí abstraktnější sémantickou informaci, než je toho schopná reprezentace po jednotlivých slovech. Tato informace může v kontextu klasifikace dokumentů přinést kvalitnější distribuovanou reprezentaci a tím pádem napomoci vyšší úspěšnosti klasifikace.

Klasifikace dokumentů na základě navržené hierarchické reprezentace (a některých *baseline* reprezentací vybraných pro porovnání) bude probíhat pomocí neuronových sítí. Konkrétně bude věnována pozornost dvěma architekturám neuronových síti – klasickému vícevrstvému perceptronu a konvoluční neuronové síti, která se začala používat pro klasifikaci textových dokumentů teprve před několika lety.

1.1 Přehled kapitol

Druhá kapitola popisuje paradigma klasifikační úlohy a způsoby hodnocení úspěšnosti klasifikace. Třetí kapitola uvádí stručný přehled základních metod klasifikace dokumentů a podrobněji neuronové sítě. Ve čtvrté kapitole se nachází seznámení s předzpracováním textu a s tradičními a distribuovanými reprezentacemi textových dokumentů. Pátá kapitola popisuje použité datové sady a některé jejich důležité statistické údaje. Šestá kapitola pojednává o hierarchických reprezentacích a navržených metodách, sedmá stručně shrnuje implementovanou aplikaci. V osmé kapitole se nachází seznam provedených experimentů a jejich výsledky.

2 Klasifikace dokumentů

Tato kapitola popisuje problematiku klasifikace dokumentů, obecnou klasifikační úlohu a některé vybrané metriky, kterými se hodnotí úspěšnost klasifikace.

2.1 Klasifikační úloha

Klasifikace (někdy také kategorizace) dokumentů se zabývá problémem zařazování dokumentů do kategorií. Cílem klasifikační úlohy [65] je přiřadit každému dokumentu d z množiny dokumentů D kategorii (nebo více kategorií) z množiny kategorií C. Formálně $D \times C \rightarrow \{0, 1\}$, tzn. každému páru $< d_j, c_i > \in D \times C$ přiřazuje hodnotu 0 nebo 1, což znamená, že dokumentu d_j není, respektive je, přiřazena kategorie c_i. Jinak řečeno je dokumentu d_j přiřazen vektor $C^j = (c_1^j, c_2^j, ..., c_k^j) \in \{0, 1\}^k$, kde $1 \leq j \leq |D|$ a $k = |C|$.

2.1.1 Typy klasifikační úlohy

Klasifikační úlohy lze rozdělit podle typu učení na učení s učitelem, učení bez učitele a kombinované učení [26; 28, s. 184], nebo podle podle velikosti množiny C a počtu nenulových pozic vektorů $C^j, 1 \leq j \leq |D|$ na binární klasifikaci, *multi-class* klasifikaci a *multi-label* klasifikaci [81].

Učení s učitelem

Při učení s učitelem (angl. *supervised learning*) se k trénování klasifikátoru používají anotovaná data – to znamená, že je u nich určena kategorie, do které patří. Kategorie jsou tedy předem pevně určeny.

Učení bez učitele

Na rozdíl od učení s učitelem, nejsou u učení bez učitele (*unsupervised learning*) data anotována – není u nich určena kategorie. Stejně tak kategorie nejsou předem pevně dány, přičemž často není ani určen jejich počet. Úloha klasifikace dokumentů za použití učení bez učitele [8; 73] se také často nazývá **shlukování** (*clustering*). Principem je zařazení dokumentů do shluků (kategorií) na základě jejich podobnosti, která může být různě definována.

Kombinované učení

Kombinované učení (*semi-supervised learning*) kombinuje oba předchozí typy a obvykle se používá, pokud je k dispozici málo dat s přiřazenou kategorií a velké množství dat bez kategorie. V tomto případě lze natrénovat klasifikátor na anotovaných datech, klasifikovat data bez anotací a poté přetrénovat klasifikátor navíc s těmi klasifikovanými daty, která byla zařazena do kategorií s nejvyšší pravděpodobností. Tato metoda se nazývá *self-training*, ale existují i další [56].

Binární klasifikace

Pokud $k = 2$ a $|\{c \in C; c = 1\}| = 1$, tedy existují pouze dvě kategorie a dokument patří do jedné z nich, jedná se o binární klasifikaci. Příkladem je typická úloha rozhodnutí jestli e-mail je spam nebo není (např. [78]).

Multi-class klasifikace

Pokud $k > 2$ a $|\{c \in C; c = 1\}| = 1$, jde o *multi-class* (více-třídní, ale většinou se do češtiny nepřekládá) klasifikaci. Každému dokumentu je přiřazena přesně jedna kategorie z množiny kategorií velikosti více než dva.

Výstupem většiny klasifikátorů pro daný dokument ovšem není přímo vektor hodnot 0 a 1, ale obecně jakýchkoliv reálných hodnot – nejčastěji z intervalu $\langle 0, 1\rangle$, což značí pravděpodobnost[1], že dokument patří do kategorie c_i. V tomto případě se hodnota 1 do vektoru dosadí na pozici, kde je pravděpodobnosti nejvyšší, hodnota 0 na všechny ostatní pozice.

Multi-class a binární klasifikaci také lze souhrnně nazvat *single-label* klasifikace.

Multi-label klasifikace

Pokud $k > 2$ a $0 \leq |\{c \in C; c = 1\}| \leq k$, tzn. existuje více než 2 kategorie a dokument může být zařazen do libovolného množství z těchto kategorií, jedná se o *multi-label* (do češtiny se nepřekládá) klasifikaci[2] [30; 70; 84].

Stejně jako u *multi-class* klasifikace, není výstup klasifikátoru přímo vektor hodnot 0 a 1. Zde se obvykle provádí **prahování** [19] (angl. *thresholding*).

[1] Ale nemusí to být pravděpodobnost; vždy záleží na tom, jak je definovaný výstup klasifikátoru.

[2] Pokud nebude řečeno jinak, jsou nadále termíny „kategorie“ a „*label*“ zaměnitelné.

Hodnota 1 bude dosazena do vektoru C^j na pozice, kde $c_i \geq t$, kde t je definovaný **práh** (*threshold*) – ten může být buď pevně určen předem (často např. 0.5, pokud jsou výstupem klasifikátoru pravděpodobnosti), zjištěn experimentálně, nebo je možné algoritmicky zjistit nejvhodnější hodnotu během trénování klasifikátoru [17; 49; 85], což ale může být relativně složité v porovnání s oběma předchozími metodami. Na ostatní pozice, tj. kde $c_i < t$, bude dosazena hodnota 0.

2.2 Hodnocení úspěšnosti klasifikace

Existují mnohé evaluační metriky, kterými lze zhodnotit, nakolik je klasifikační systém úspěšný. Většina jich vychází z **konfúzní matice** [20; 55] (někdy také pod názvem kontingenční tabulka), které se věnuje kapitola 2.2.2. Není ovšem vhodné testovat úspěšnost klasifikátoru na datech, na kterých byl natrénován – proto se data různě dělí(viz kapitolu 2.2.1), aby byly výsledné metriky důvěryhodné.

Tato kapitola také obsahuje seznam některých často používaných metrik a způsob jejich výpočtu – důraz je kladen především na metriky, které lze využívat pro *multi-class* klasifikaci. Názvy metrik budou přednostně uváděny v angličtině, protože české překlady jsou často zavádějící. Např. obě slova *Accuracy* a *Precision* se často překládají jako *Přesnost*.

Existují dvě skupiny metrik podle způsobu jakým se počítají [70] – *label-based* (kapitola 2.2.3) a *example-based* (kapitola 2.2.5), přičemž většinu metrik lze počítat oběma způsoby. U prvního způsobu se navíc ještě uvádějí dvě možnosti, jak metriky vypočítat [70] – mikro-průměrování a makro-průměrování (kapitola 2.2.4).

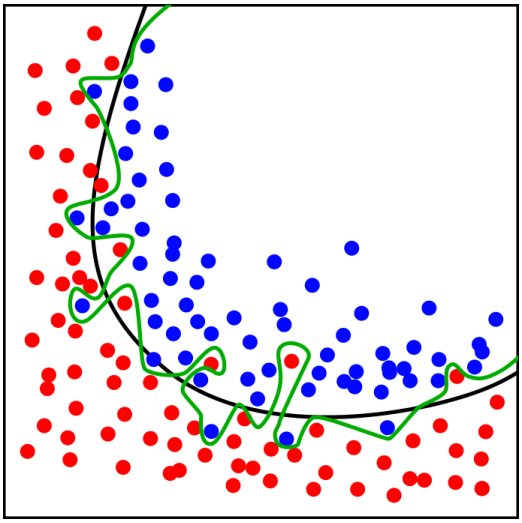
2.2.1 Dělení dat

Při implementaci klasifikátoru je běžná praxe model různě ladit – na každou úlohu nebo data není možné použít stejné hyperparametry[3], proto se hledají pro konkrétní úlohu nejvhodnější konfigurace – model se opakovaně trénuje a testuje pro různé hyperparametry. Pokud by byl model laděn na

[3]Jako „parametry“ se označují vlastnosti modelu, které jsou upravovány učícím algoritmem, např. váhy v neuronových sítích. „Hyperparametry“ jsou předem nastavené globální parametry, které se neupravují procesem učení. Jde např. o počet neuronů ve vrstvě neuronové sítě, počet iterací učení apod.

datech, na kterých je trénován, může lehko dojít k **přeučení** (*overfitting*[4]) – klasifikátor se dobře naučí předpovídat trénovací data, ale neporadil by si dobře s daty, která dosud neviděl – neměl by schopnost tzv. **generalizace**. Přeučení může nastat i tak, že proces učení klasifikátoru běží déle, než je vhodné – klasifikátor může zachytit různý náhodný šum, který je specifický pro data, na kterých byl trénován.

Příklad přeučení lze vidět na obrázku 2.1, kde je úkolem kategorizovat data do dvou kategorií (modrá a červená). Zelená křivka reprezentuje přeučení – přesně dělí trénovací data (tj. chyba na trénovacích datech je nulová), ale lze očekávat, že chybovost na nových datech, která klasifikátor dosud neviděl, bude vyšší, než u černé křivky.



Obrázek 2.1: Příklad přeučení klasifikátoru. Zelená křivka reprezentuje overfitting. *Obrázek převzat z [12].*

Existuje také opačný problém, tzv. **nedoučení** (*underfitting*[5]), kdy je model nedostatečně komplexní, aby zachytil vztahy v datech. Příkladem může být pokus rozdělit dvě kategorie na obrázku 2.1 lineární funkcí.

Nejčastější způsob dělení dat, aby bylo co nejlépe zabráněno přeučení, je na 3 disjunktní množiny – **trénovací**, **validační** a **testovací** data. Je vhodné, aby tyto 3 množiny měly pokud možno podobné vlastnosti – např. u klasifikace dokumentů by měla být podobná distribuce kategorií v jednotlivých množinách. Přestože neexistuje žádné univerzální pravidlo, často se data dělí

[4]V češtině se často používají i anglické termíny *underfitting* a *overfitting*.
[5]Viz poznámku pod čarou 4.

50%/25%/25% nebo 60%/20%/20%. Existují také různé metody, jak zvolit vhodné rozdělení podle vlastností dostupné datové sady [24].

Trénovací data

Na těchto datech je model pouze natrénován, resp. jsou nastaveny jeho parametry – např. optimální váhy v neuronových sítích.

Validační data

Na validačních datech (někdy anglicky *development set* nebo také *dev set*) je model laděn – zkouší se různé hyperparametry, pro které klasifikátor bude dosahovat nejlepší úspěšnosti. Také se hledá vhodný počet iterací, jak dlouho má učení klasifikátoru trvat. Tato praktika se označuje jako *early-stopping* [57] (včasné zastavení) – učení se nechá běžet tak dlouho, než se začne úspěšnost klasifikace zhoršovat na validačních datech, což značí, že se klasifikátor začal přeučovat na trénovacích datech. Tato metoda validace, tj. existuje předem pevně určená množina validačních dat, se nazývá *hold-out validation* [58].

Existuje také často používaná metoda zvaná **křížová validace** [58] (*cross-validation*). Jedna její verze se nazývá *k-fold cross validation* a funguje tak, že je trénovací množina rozdělena na k dílů, klasifikátor opakovaně trénován na $k-1$ dílech (vždy s vynecháním jiného dílu) a na jednom zbylém dílu otestován. Výsledky z k testů jsou potom zprůměrovány. Nejčastěji se používá $k = 5$ nebo $k = 10$ [28, s. 184].

Testovací data

V momentě, kdy je klasifikátor odladěn, je jeho finální úspěšnost získána z testovacích dat – tato data klasifikátor nikdy dříve neviděl. Po získání finálních výsledků na testovacích datech už by klasifikátor neměl být nijak laděn. Tato třetí množina se používá především proto, že k přeučení může dojít i na validační množině [50]. Tak dlouho jsou laděny hyperparametry a nastavení klasifikátoru, až je nalezena nejlepší možná konfigurace, která je ale specifická pouze pro evaluaci na validačních datech. Takto získané výsledky nejsou důvěryhodné – proto je nakonec úspěšnost klasifikátoru získána z testovacích dat.

2.2.2 Konfúzní matice

Pro výpočet metrik je nejprve nutné sestavit konfúzní matici. Aby to bylo možné, je samozřejmě nutné pro každý dokument znát kategorii, do které dokument patří, a kategorii, kterou předpověděl systém. Nejprve bude konfúzní matice vysvětlena na binární klasifikaci a poté rozšířena pro *multi-class* (*multi-label*) klasifikaci.

Binární klasifikace

Konfúzní matici lze vidět v tabulce 2.2, kde sloupce **ANO** a **NE** značí, jestli dokument patří do sledované kategorie (např. úloha, jestli e-mail je nebo není spam), a řádky **ANO** a **NE** značí, jak kategorii určil klasifikátor.

		Očekávaná předpověď	
		ANO	**NE**
Předpověď	**ANO**	*tp*	*fp*
	NE	*fn*	*tn*

Tabulka 2.2: Konfúzní matice.

Význam jednotlivých položek tabulky, ze kterých je většina metrik počítána, je vysvětlen v následujícím seznamu, kde *true* znamená správnou předpověď klasifikátoru a *false* špatnou.

- *tp (true positive)* – dokument patří do předpovězené kategorie
- *fp (false positive)* – dokument nepatří do předpovězené kategorie
- *fn (false negative)* – dokument patří do kategorie, která nebyla předpovězena
- *tn (true negative)* – dokument nepatří do kategorie, která nebyla předpovězena

Je zřejmé, že je žádoucí, aby hodnoty *fp* a *fn* byly co nejmenší.

Multi-class **klasifikace**

Pokud existuje více tříd než 2, má konfúzní matice tvar uvedený v tabulce 2.3, kde lze vidět modelový případ zařazení dokumentů do 3 kategorií politika, ekonomika a sport. Pro zjednodušení je klasifikační úloha modelového případu pouze *single-label*.

		Očekávaná kategorie		
		Politika	**Ekonomika**	**Sport**
Předpovězená kategorie	**Politika**	40	5	6
	Ekonomika	12	37	4
	Sport	1	5	46

Tabulka 2.3: Zařazení dokumentů do kategorií – modelový případ.

V každém sloupci tabulky je uvedeno rozložení všech dokumentů, náležících jedné kategorii, jak byly kategorie předpovězeny klasifikačním systémem. Tzn. například v kategorii sport je 56 $(6 + 4 + 46)$ dokumentů, ale z toho 6 jich bylo špatně zařazeno do kategorie politika a 4 do kategorie ekonomika.

Pro výpočet metrik ovšem není tento tvar konfúzní matice intuitivní. Proto lze nahlížet na *multi-class* klasifikaci do n kategorií jako na n binárních klasifikací a sestavit pro každou kategorii zvlášť binární konfúzní matici – např. pro kategorii Sport je tato matice sestavena v tabulce 2.4. Lze jít ještě o krok dále a sečíst *tp*, *tn*, *fp* a *fn* přes všechny kategorie, což lze pro modelový případ vidět v tabulce 2.5. Součet všech položek konfúzní matice sestavené pro jednu kategorii se musí rovnat počtu dokumentů (k), pro celkovou konfúzní matici potom $k \cdot n$.

	ANO	**NE**
ANO	46	6
NE	10	94

Tabulka 2.4: Konfúzní matice – kategorie Sport.

	ANO	**NE**
ANO	123	33
NE	33	279

Tabulka 2.5: Konfúzní matice – všechny kategorie.

Takto je možné sestavit konfúzní matice pro jednotlivé kategorie a pro jejich součet i pro *multi-label* klasifikaci. Součet všech položek celkové konfúzní matice se musí opět rovnat $k \cdot n$.

2.2.3 *Label-based* metriky

Metriky uvedené v této kapitole se běžně používají pro binární klasifikaci, proto se zde opět využívá fakt, že se lze chovat ke klasifikaci do n kategorií jako k n binárním klasifikacím. Celkový výsledek pak bude průměrem metrik vypočtených z binárních konfúzních matic jednotlivých kategorií – odtud název *label-based*.

Precision

Precision [20; 69] udává poměr, kolik dokumentů zařazených klasifikátorem do kategorie opravdu do této kategorie náleží. Tomu odpovídá vzorec 2.1.

$$P = \frac{tp}{tp + fp} \tag{2.1}$$

Výpočet precision.

Recall

Recall [20; 69] (úplnost) znamená, kolik dokumentů klasifikátor označil kategorií c v poměru k počtu všech dokumentů, které do této kategorie náleží.

$$R = \frac{tp}{tp + fn} \tag{2.2}$$

Výpočet recall.

F-measure

F-measure [20; 69] (F-skóre, F-míra) je harmonický průměr *precision* (P) a *recall* (R) – vzorec 2.3[6]. Používá se harmonický průměr H_n, spíše než aritmetický A_n, protože aritmetický průměr není penalizován horší z obou průměrovaných metrik – vždy platí $H_n \leq A_n$.

$$F = \frac{2 * P * R}{P + R} \tag{2.3}$$

Výpočet F-skóre.

[6]Konkrétně jde o F1-skóre, kde jednička značí, že oba členy průměru mají stejnou váhu, ale jelikož se v této práci jiná váha nepoužívá, bude se o něm mluvit jako o F-skóre.

2.2.4 Mikro- a makro-průměrování metrik

Způsob výpočtu metrik v předchozí kapitole je tzv. **makro-průměr**, ale často se používá ještě druhý způsob, **mikro-průměr** [69; 70]. Jak již bylo výše řečeno, makro-průměrovaná metrika se počítá pro každou kategorii zvlášť (tzn. z tabulky 2.4) a poté je z těchto hodnot vypočten aritmetický průměr, kdežto mikro-průměrovaná je vypočtena ze součtů tp, fp, tn a fn přes všechny kategorie (tzn. pro modelový případ by to bylo z tabulky 2.5). Z toho vyplývá, že mikro-průměr dává kategorii váhu, která je úměrná počtu dokumentů, které se v ní nachází, kdežto makro-průměr přiřazuje všem kategoriím stejnou váhu.

Oba způsoby výpočtu lze vidět zapsané obecným vzorcem v 2.4 a 2.5, kde $C = \{c_j : j = 1...n\}$ je množina všech kategorií.

$$E_\mu = E(\sum_{c=1}^{n} tp_c, \sum_{c=1}^{n} fn_c, \sum_{c=1}^{n} fp_c, \sum_{c=1}^{n} tn_c) \tag{2.4}$$

Mikro-průměrovaná metrika.

$$E_M = \frac{1}{n} \sum_{c=1}^{n} E(tp_c, fn_c, fp_c, tn_c) \tag{2.5}$$

Makro-průměrovaná metrika.

Poměrně často může být užitečné a důležité uvádět hodnoty metrik pro oba způsoby výpočtu. Platí to zvláště, pokud distribuce dokumentů v kategoriích je nerovnoměrná (např. v některých kategoriích je 20krát více dokumentů, než v jiných). Lze očekávat, že se klasifikátor hůře naučí předpovídat málo zastoupené kategorie, což se výrazně neprojeví na mikro-průměru, kdežto makro-průměr je tím ovlivněn podstatně více.

Bude-li například existovat klasifikační úloha se třemi kategoriemi, kde v testovací množině náleží 100, 1000 a 2000 dokumentů kategoriím A, B a C a po předpovědi klasifikátoru vzniknou pro jednotlivé kategorie konfúzní matice v tabulkách na 2.6, budou hodnoty F-skóre $F_\mu \approx 89\%$ a $F_M \approx 64\%$. 89% je pro některé úlohy velice slušná hodnota mikro-průměrovaného F-skóre, ale je třeba přihlédnout i k makro-průměrovanému, které už je pouze 64%, protože jej snižuje kategorie A, kterou klasifikátor ve většině případů nedokázal správně předpovědět.

A	**ANO**	**NE**
ANO	10	100
NE	90	2900

B	**ANO**	**NE**
ANO	900	140
NE	100	1960

C	**ANO**	**NE**
ANO	1850	100
NE	150	1000

Tabulka 2.6: Konfúzní matice modelové úlohy.

2.2.5 *Example-based* metriky

Na rozdíl od *label-based* metrik, *example-based* metriky se počítají pro každý dokument zvlášť a poté průměrují [70]. Tento způsob výpočtu existuje i pro výše uvedené metriky – definice pro *precision*, *recall* a *F-measure* lze vidět na vzorcích 2.6, kde k je počet všech dokumentů, Y_i je množina kategorií, do kterých dokument patří, a Z_i množina kategorií, kterou pro daný dokument klasifikátor předpověděl.

$$\begin{aligned} P &= \frac{1}{k}\sum_{i=1}^{k}\frac{|Y_i \bigcap Z_i|}{|Z_i|} \quad R = \frac{1}{k}\sum_{i=1}^{k}\frac{|Y_i \bigcap Z_i|}{|Y_i|} \\ F &= \frac{1}{k}\sum_{i=1}^{k}\frac{2 * P * R}{P + R} = \frac{1}{k}\sum_{i=1}^{k}\frac{2 * |Y_i \bigcap Z_i|}{|Y_i| + |Z_i|} \end{aligned} \tag{2.6}$$

Example-based verze některých metrik – precision, recall a F-skóre.

Dále se často používají *accuracy* a *Hamming loss*.

Accuracy

Často se uvádí dvě různé verze metriky *accuracy* (někdy v češtině „úspěšnost“ nebo „správnost“[7]) – pro binární a *multi-label* klasifikaci, přičemž zde je samozřejmě podstatná ta druhá – tzv. *multi-label accuracy*[8]. Její význam je poměr správně předpovězených kategorií k počtu všech označených kategorií (správných a předpovězených). Zapsanou vzorcem ji lze vidět na 2.7.

$$A_{ML} = \frac{1}{k}\sum_{i=1}^{k}\frac{|Y_i \bigcap Z_i|}{|Y_i \bigcup Z_i|} \tag{2.7}$$

Výpočet accuracy (multi-label).

[7] Ale termín „úspěšnost“ nebude používán, protože může být zavádějící. „Úspěšností klasifikace/klasifikátoru“ se má na mysli celková kvalita klasifikátoru, nejen jedna metrika.

[8] Nebude-li řečeno jinak, každým dalším výskytem *accuracy* se má na mysli tato verze, tj. *multi-label accuracy*.

Multi-label accuracy byla zavedena proto, že binární *accuracy* nelze pro klasifikaci do více tříd použít, protože do výpočtu zahrnuje i správně nepředpovězené kategorie (tj. *true negative* položka v konfúzní matici) – viz vzorec 2.8.

$$A_B = \frac{tp + tn}{tp + tn + fp + fn} \tag{2.8}$$

Výpočet accuracy (binární).

Zahrnutí *true negative* značně zkresluje výsledek, pokud by byla tato verze použita pro klasifikaci do více kategorií. Například bude-li existovat testovací množina obsahující 10000 dokumentů, kde každých 100 dokumentů patří do jiné kategorie (tzn. je 100 kategorií), a klasifikátor nepředpoví správně kategorii ani pro jediný dokument, bude v konfúzní matici pro každou kategorii stále $tp = 0, fp = 100, fn = 100, tn = 9800$ (za předpokladu, že pro každou kategorii jsou všechny předpovědi klasifikátoru rovnoměrně rozloženy mezi ostatní kategorie). Výsledná *accuracy* potom bude 98%, což na první pohled vypadá jako velice slušný výsledek, přestože klasifikátor správně nezařadil ani jeden dokument. *Multi-label accuracy* by pro tento případ vyšla 0%, což odpovídá realitě.

Hamming loss

Hamming loss [70; 76] (Hammingova ztráta) je stejně jako *multi-label accuracy* určena pro *multi-label* klasifikaci a udává, jak často je předpovězena nesprávná kategorie (měla být předpovězena, ale nebyla, nebo neměla být, ale byla), resp. poměr špatně předpovězených kategorií k počtu správných kategorií. Vypočítat ji lze podle vzorce 2.9, kde k je počet všech dokumentů, $|Y|$ počet kategorií, do kterých dokument patří, $\oplus$ symetrický rozdíl dvou množin (disjunktní sjednocení), Y_i množina kategorií, do kterých dokument patří, a Z_i množina kategorií, kterou pro daný dokument klasifikátor předpověděl. Jelikož je to ztrátová funkce, cílem je, aby byla její hodnota co nejmenší.

$$H = \frac{1}{k} \sum_{i=1}^{k} \frac{|Y_i \oplus Z_i|}{|Y|} \tag{2.9}$$

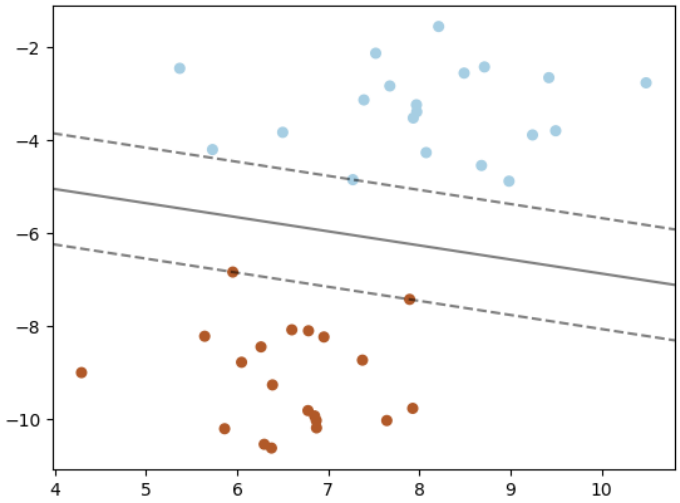
Hammingova ztráta.

3 Metody klasifikace dokumentů

Tato kapitola krátce zmiňuje některé známé metody strojového učení (k-nejbližších sousedů, *Support vector machines*, Naivní bayesovský klasifikátor a Maximální entropie), používané ke klasifikaci dokumentů, a podrobněji pak neuronové sítě, kterým se tato práce věnuje. Nakonec je uvedena krátká zmínka o *multi-label* klasifikaci těmito metodami.

3.1 Support Vector Machines

Cílem algoritmu *Support Vector Machines* [47, s. 319] (SVM; do češtiny se obvykle nepřekládá) je nalézt nadrovinu, která dělí (v případě binární klasifikace) prostor příznaků tak, že body v prostoru (např. dokumenty), náležící odlišným kategoriím, budou ležet na opačných stranách této nadroviny. Aby byla chyba klasifikace co nejmenší, je žádoucí umístit tuto nadrovinu tak, aby byla co nejdále od nejbližších příslušníků obou kategorií (tzv. *maximal margin*, neboli „maximální okraje“) – viz obrázek 3.1. Přerušované čáry se nazývají *support vectors*.

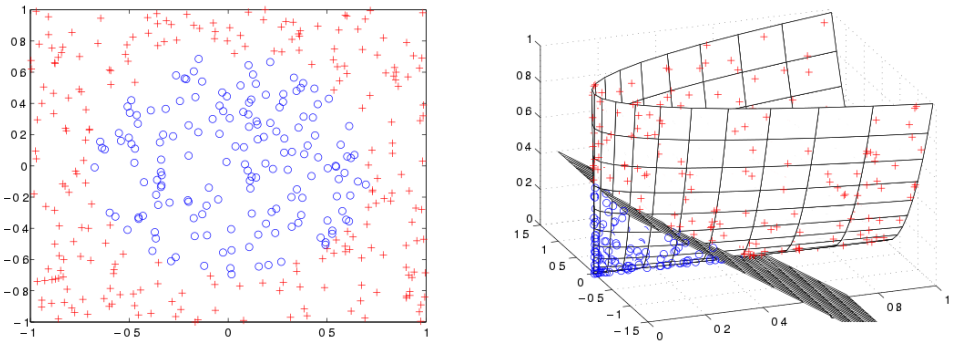


Obrázek 3.1: Support vector machines – *lineárně separovatelný případ* [9].

[9] Převzato z `http://scikit-learn.org/stable/auto_examples/svm/plot_separating_hyperplane.html`.

Tento případ, kdy lze kategorie rozdělit nadrovinou, se označuje jako **lineárně separovatelný**. Obecně toto ovšem neplatí a proto existuje i verze pro lineárně neseparovatelné kategorie, kdy se při hledání optimální nadroviny navíc penalizují špatně umístěné body – tzn. hledá se nadrovina, jejíž okraje jsou co největší a zároveň je špatně umístěno co nejméně bodů – pomocí parametrů lze ovládat, které kritérium je důležitější.

Také existuje nelineární SVM, kdy jsou data transformována do vyšší dimenze, ve které už je lze rozdělit nadrovinou. Tento postup se označuje jako *kernel trick* a konkrétní transformace je určena „jádrovou funkcí“ (*kernel function*). Příklad lze vidět na obrázku 3.2.



Obrázek 3.2: Příklad nelineárního SVM [45].

3.2 Naivní bayesovský klasifikátor

Naivní bayesovský klasifikátor [47, s. 258] (*Naive Bayes classifier*, NBC) vychází z pravděpodobnostního modelu, kde jsou dokumentu, reprezentovaného vektorem $\mathbf{x} = (x_1, x_2, ..., x_n)$, přiřazeny pravděpodobnosti $p(C_k|\mathbf{x})$ pro každou kategorii C_k. Tato pravděpodobnost je získána za použití **Bayesovy věty** podle vzorce 3.1.

$$p(C_k|\mathbf{x}) = \frac{p(C_k)p(\mathbf{x}|C_k)}{p(\mathbf{x})} \tag{3.1}$$

Bayesova věta.

Část vzorce ve jmenovateli lze vynechat, protože není podmíněna kategorií a hodnoty příznaků jsou konstantní. Naivní bayesovský klasifikátor lze pak zkombinovat s rozhodovacím pravidlem a zapsat vzorcem 3.2, tzn. pro

daný dokument se hledá kategorie y, pro kterou je pravděpodobnost nejvyšší. Část vzorce $p(C_k)$ není nic jiného, než apriorní pravděpodobnost, že dokument patří do kategorie C_k, což je poměr počtu dokumentů v trénovací množině, které patří do kategorie C_k, k celkovému počtu dokumentů v trénovací množině. Část vzorce $p(x_i|C_k)$ pak reprezentuje pravděpodobnost výskytu příznaku x_i (nejčastěji slova) v dokumentech, které patří do kategorie C_k, což je relativní frekvence výskytu tohoto slova ve všech dokumentech, patřících do kategorie C_k.

$$y = \underset{k \in 1,2,\ldots,|C|}{\arg\max} (p(C_k) \prod_{i=1}^{n} p(x_i|C_k)) \tag{3.2}$$

Naivní bayesovský klasifikátor.

3.3 Maximální entropie

Základem techniky klasifikace maximální entropií [7] je modelování distribuce pravděpodobnosti přiřazení určitého výstupu podle daného vstupu (v případě klasifikace kategorie c podle dokumentu d). Hledá se takový statistický model $p(c|d)$, který je konzistentní se všemi vstupními daty (tj. jaký dokument patří do jaké kategorie), ale jinak uniformní.

Zavádí se tzv. omezení (angl. *constraints*) ve tvaru rovnice 3.3, kde $\widetilde{p}(x)$ je empirická distribuce dokumentů v trénovací množině, $p(c|d)$ hledaný model, $\widetilde{p}(d,c)$ empirická distribuce dokumentů v kategoriích v trénovací množině a $f(d,c)$ charakteristická funkce, která nabývá hodnoty 1, pokud dokument d patří do kategorie c a obsahuje daný příznak, jinak 0. Celá pravá strana rovnice pak má význam očekávané hodnoty f s ohledem na distribuci $\widetilde{p}(d,c)$, levá strana očekávané hodnoty f s ohledem na hledaný model $p(c|d)$.

$$\sum_{d,c} \widetilde{p}(d) p(c|d) f(d,c) = \sum_{d,c} \widetilde{p}(d,c) f(d,c) \tag{3.3}$$

Omezení statistického modelu trénovacími daty.

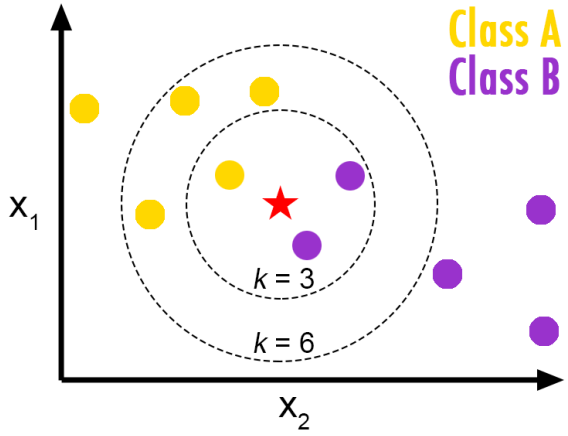
Tyto rovnosti (pro každý příznak jedna) splňuje nekonečně mnoho modelů $p(c|d)$. Jako nejlepší model p^* z množiny všech možných modelů C se proto vybírá ten, který má maximální entropii $H(p)$ – vzorec 3.4. Vypočtení takového modelu, který splňuje zavedená omezení a má maximální entropii, se provádí např. metodou *Improved Iterative Scaling* [7].

$$p* = \arg\max_{p \in C}(H(p)) = \arg\max_{p \in C}(\tilde{p}(d)p(c|d)\log p(c|d)) \tag{3.4}$$

Maximální entropie statistického modelu.

3.4 Algoritmus k-nejbližších sousedů

Algoritmus k-nejbližších sousedů [53] (*K-nearest neighbors*, zkracuje se na KNN nebo kNN) je jedna ze základních a jednoduchých metod, využitelných pro klasifikaci dokumentů. Trénovací fáze spočívá v umístění dokumentů do n-rozměrného prostoru (podle toho, kolik dimenzí má vektor, kterým jsou dokumenty reprezentovány). Prakticky jde pouze o „zapamatování“ vektorů (a příslušné kategorie, do které dokument patří), nic není počítáno – jde o tzv. *lazy learning* (líné učení). V testovací fázi je potom vektor dokumentu umístěn do stejného n-rozměrného prostoru a nalezeno k nejbližších dokumentů[10] (odtud název metody) – dokument je zařazen do kategorie, která je v k nejbližších dokumentech zastoupena nejčastěji. Příklad lze vidět na obrázku 3.3, kde *n=2*. Nový červený dokument bude klasifikován do kategorie B, pokud *k=3*, ale do kategorie A, pokud *k=6*.



Obrázek 3.3: Příklad klasifikace dokumentu algoritmem k-nejbližších sousedů[11].

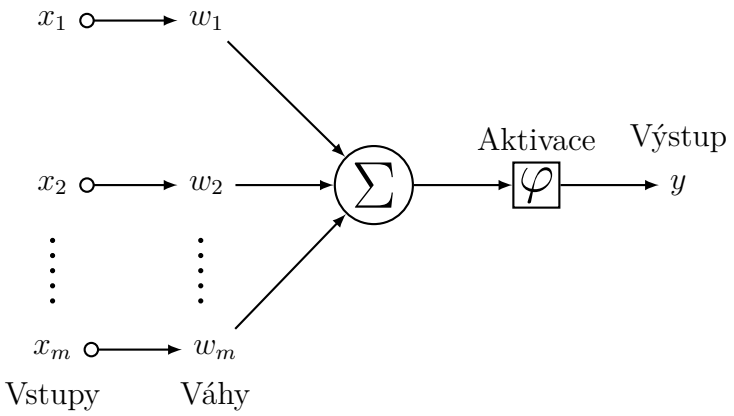
[10] Používají se různé způsoby výpočtu vzdálenosti dokumentů, např. Eukleidovská vzdálenost.

[11] Převzato z `https://helloacm.com/a-short-introduction-to-k-nearest-neighbors-algorithm/`.

3.5 Neuronové sítě

Protože je oblast neuronových sítí poměrně rozsáhlá, budou zmíněny pouze základní definice a principy; existuje velké množství článků, knih atd., kde se lze dozvědět více [18; 34]. Největší pozornost bude věnována dvěma architekturám neuronových sítí – vícevrstvému perceptronu a konvolučním neuronovým sítím.

Neuronové sítě (ANN[12]) jsou výpočetní systémy volně modelované podle biologické neuronové sítě. Jsou složeny z různě vzájemně propojených jednotek (uzlů), které se v kontextu neuronových sítí nazývají **neurony**[13] (někdy v angl. také *units*). Každé **spojení** (*connection*) může přenášet signál (v kontextu ANN reálné číslo). Formálně je ANN uspořádaná trojice (N, V, w), kde N je množina neuronů a V množina $\{(i,j)|i,j \in N\}$ neboli množina spojení mezi neurony i a j. Funkce $w : V \to \mathbb{R}$ definuje váhy, kde $w_{i,j}$ je váha spojení mezi neurony i a j [34, s. 38]. Základní model neuronu lze vidět na obrázku 3.4.



Obrázek 3.4: Základní model neuronu.

Neuron má m vstupních spojení, kde $(x_1, x_2, ..., x_m)$ jsou **vstupy** (signály) a $(w_1, w_2, ..., w_m)$ **váhy**, kterými jsou vstupy váženy. Pro získání výstupu neuronu y je aplikována **aktivační funkce** φ na součet vážených vstupů – vzorec 3.5. Jako aktivační funkce se často používá sigmoid, tanh, nebo ReLU [18, s. 15] (z angl. *Rectified Linear Unit*).

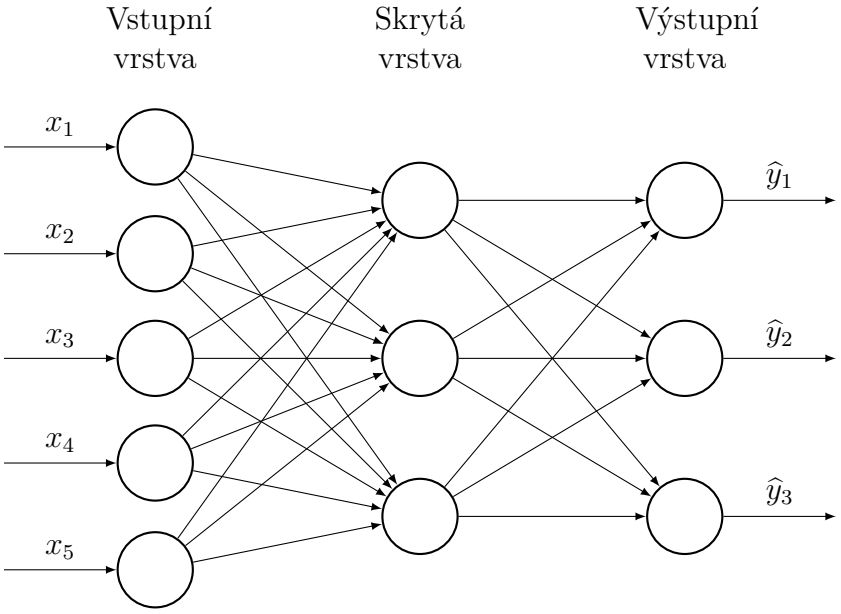
[12]Zkratka ANN znamená *Artificial neural networks* (umělé neuronové sítě). Slovo *artificial* se používá kvůli odlišení od biologických neuronových sítí. Avšak bude-li kdykoliv dále zmíněna „neuronová síť“, vždy se má na mysli ANN. Používána bude i ekvivalentní zkratka NN.

[13]Stejně jako u neuronových sítí, formálně správný název by byl „umělý neuron“ (*artificial neuron*), ale používán bude pouze název „neuron“.

$$y = \varphi(\sum_{a=1}^{m} w_a x_a) \tag{3.5}$$

Výpočet výstupu neuronu.

Obvykle je neuronová síť uspořádána do vrstev, kde každá vrstva může mít různý počet neuronů. Příklad jednoduché neuronové sítě lze vidět na obrázku 3.5. První vrstva se nazývá **vstupní** (angl. *input layer*). Neurony v této vrstvě nemají vstupní spojení jako takové (a tedy ani váhy), protože neexistuje předchozí vrstva, ale přijímají vstup (dokument) v podobě číselného vektoru, zde vektor $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5)$. Stejně tak poslední vrstva, která se nazývá **výstupní** (*output layer*), nemá následovníka a její výstup je výstupem neuronové sítě jako celku. Síť na obrázku má 3 výstupní neurony, což, pokud by šlo o klasifikaci, odpovídá klasifikaci do 3 kategorií, kde vektor reálných čísel $\widehat{\mathbf{y}} = (\widehat{y}_1, \widehat{y}_2, \widehat{y}_3)$ jsou předpovědi pro jednotlivé kategorie. Ostatní vrstvy neuronové sítě jsou tzv. **skryté vrstvy** (*hidden layers*), které nemají žádné spojení s okolím neuronové sítě, pouze s ostatními vrstvami.



Obrázek 3.5: Příklad jednoduché neuronové sítě.

Pokud jsou všechny neurony jedné vrstvy spojeny s každým neuronem další vrstvy, jde o ***fully-connected*** (plně propojenou) vrstvu. Pokud existují v síti spojení pouze směrem od vstupní vrstvy k výstupní, jako na obrázku 3.5, jedná se o ***feed-forward*** (dopřednou) neuronovou síť[18, p. 11; 34,

p. 45] (FNN). Neuronové sítě, které mají i jiná spojení (může to být buď směrem od výstupní vrstvy ke vstupní, nebo v rámci jedné vrstvy, ať už je neuron spojen sám se sebou nebo s jinými neurony v dané vrstvě), se nazývají **rekurentní neuronové sítě**[18, p. 77; 34, p. 48] (RNN).

Neuronová síť se trénuje změnou vah spojení neuronů. Tyto změny je samozřejmě nutné vypočítat. Prvním krokem je **ztrátová funkce** [18, s. 21] (*loss function*), která vyhodnocuje, jak velká je chyba výstupu (jak hodně se očekávaný výstup liší od skutečného). Cílem je tedy hodnotu této funkce minimalizovat, což se provádí **gradientním sestupem** (*gradient descent*). Metoda, kterou lze pro neuronové sítě gradient ztrátové funkce vypočítat, se nazývá ***backpropagation of error***[14][18, s. 27; 34, s. 102], jejímž cílem je propagace chyby od výstupní vrstvy směrem zpět a podle ní úprava vah jednotlivých spojení.

3.5.1 Vícevrstvý perceptron

Za vícevrstvý perceptron [34, s. 98] (MLP z angl. *multi-layer perceptron*) se označuje taková *feed-forward* neuronová síť, která je složena pouze z *fully-connected* vrstev a má alespoň jednu skrytou vrstvu, tzn. výše uvedený obrázek 3.5 je vícevrstvý perceptron.

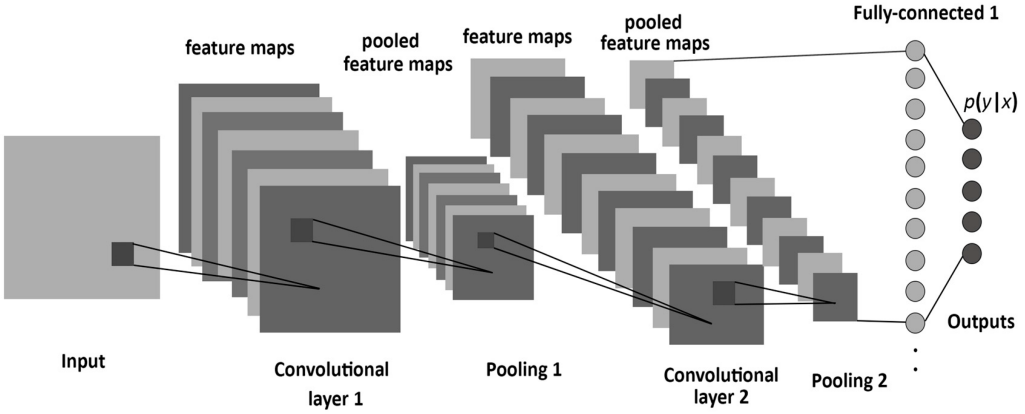
Neuronová síť, která z uvedených podmínek nesplňuje pouze přítomnost alespoň jedné skryté vrstvy (je složena pouze ze vstupní a výstupní vrstvy), se nazývá *single-layer* perceptron [34, s. 86].

3.5.2 Konvoluční neuronová síť

Konvoluční neuronová síť[23; 18, s. 42] (*convolutional neural network*, CNN) je taková neuronová síť, která využívá operaci konvoluce. V mnoha ohledech je podobná jako MLP – je *feed-forward*, má skryté vrstvy a typicky obsahuje jednu nebo více plně propojených vrstev. Navíc ale obsahuje **konvoluční vrstvu** a ***pooling* vrstvu**. Obvyklá architektura je taková, že se několikrát opakuje konvoluční a *pooling* vrstva a poté následují plně propojené vrstvy – obrázek 3.6. Vstupem konvoluční neuronové sítě jsou obvykle dvourozměrná data. Historicky se nejvíce používá na klasifikaci (a další úlohy) obrazových materiálů [35], ale poslední dobou se rozmáhá i použití na úlohy zpracování přirozeného jazyka (*Natural language processing*, NLP) – např. klasifikace

[14] Ale běžně se používá jen *backpropagation*, případně jen *backprop*; zkracuje se na BP. Český název by byl „zpětná propagace chyby“, ale více se používá anglický název.

sentimentu [32], ale i klasifikace (českých) článků [40]. To ale obvykle vyžaduje dokumenty reprezentovat dvourozměrně. Výstupy dvou nových vrstev se nazývají **mapy příznaků** (*feature maps*).



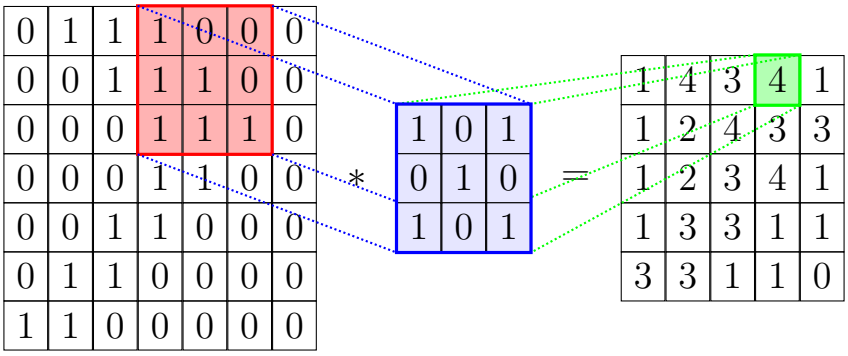
Obrázek 3.6: Ukázka schématu architektury konvoluční neuronové sítě [2].

Pro úlohy NLP je důležitá také tzv. ***embedding* vrstva**, která je určená pro vytváření vektorových reprezentací slov.

Konvoluční vrstva

Konvoluční vrstva aplikuje na výstup z předchozí vrstvy operaci konvoluce. Konvoluce se ovšem neaplikuje na celý vstup najednou, ale postupně na jeho podmnožiny – obrázek 3.7 – vrstva není plně propojená, ale pouze lokálně (*locally-connected layer*). Velikost oblasti, ze které se konvoluce počítá, je specifikována hyperparametrem – tzv. **filtr** (*filter*) neboli **jádro** (*kernel*). Filtr je postupně posunován po celém vstupu – posun filtru je určen hyperparametrem, který se nazývá **krok** (*stride*). Krok může být menší, než je velikost filtru, tzn. jednotlivé oblasti se budou překrývat. Také lze specifikovat různý krok pro každý směr posunu, stejně tak může filtr mít různé rozměry. S filtrem jsou asociovány trénovatelné váhy (v obrázku modrá matice). Při výpočtu výstupu vrstvy se provádí konvoluce pro danou oblast a matici vah – jednotlivé příznaky oblasti jsou násobeny příslušnou váhou a poté sečteny (skalární součin matic).

Konvoluční vrstva běžně obsahuje větší počet filtrů (obvykle desítky) stejných rozměrů – každý má svojí vlastní sadu trénovatelných vah. Obrázek 3.7 znázorňuje pouze jeden filtr (a tedy jednu výslednou mapu příznaků), ale ve skutečnosti má konvoluční vrstva 4 rozměry – 2 jsou určeny rozměry filtru, třetí počtem filtrů. Tyto 3 rozměry určují rozměr výstupu celé vrstvy



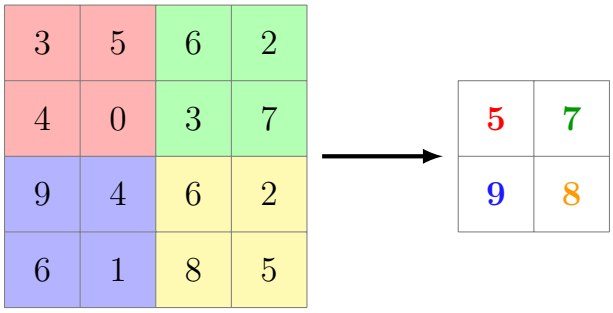
Obrázek 3.7: Příklad konvoluce[15]*, kde* $filter = (3, 3)$ *a* $stride = (1, 1)$*. Pro zjednodušení jsou příznaky i váhy pouze 0 a 1, ale obecně jsou to reálná čísla.*

– výstupem není jedna dvourozměrná mapa příznaků, ale tolik map příznaků, kolik má vrstva filtrů. Čtvrtý rozměr určuje hloubku filtru – bude-li totiž vstupem konvoluční vrstvy (*Convolutional layer 2* na obr. 3.6) výstup jiné konvoluční vrstvy (resp. *pooling* vrstvy, viz další kapitola), resp. tolik map příznaků, kolik má předchozí mapa filtrů, má filtr v aktuální vrstvě jinou sadu vah pro každou vstupní mapu příznaků. Pokud je vstupem konvoluční vrstvy pouze jedna mapa příznaků (např. dokument reprezentovaný jako matice ve vstupní vrstvě CNN – *Convolutional layer 1* na obr. 3.6), je hloubka filtru rovna jedné.

Pooling **vrstva**

Pooling vrstva se typicky umisťuje hned za konvoluční vrstvu. Její funkcí je snížení velikosti mapy příznaků, což má za efekt snížení celkového počtu parametrů a výpočtů v neuronové síti a tím pádem pomáhá i proti přeučení. Tato vrstva ale nemá žádné vlastní trénovatelné parametry. Stejně jako u konvoluční vrstvy, i zde je specifikován filtr a krok posunu filtru – jednotlivé oblasti, určené filtrem, na sebe obvykle bez překryvu a mezer navazují (krok je stejně veliký jako filtr). *Pooling* vrstva ovšem nad každou oblastí neprovádí konvoluci, ale tzv. *pooling* – nejčastěji *max pooling*, kdy je z každé oblasti vybráno maximum – viz obrázek 3.8. Existuje také *avg pooling* (průměr oblasti) a *L2 pooling* (L2-norma oblasti).

[15] S drobnými úpravami převzato z `https://github.com/PetarV-/TikZ/tree/master/2D%20Convolution`.



Obrázek 3.8: Příklad max pooling*, kde* $filter = (2, 2)$ *a* $stride = (2, 2)$.

Embedding **vrstva**

Embedding vrstva je obvykle první vrstvou neuronové sítě a poskytuje vstup následující konvoluční vrstvě. Tato vrstva funguje jako vyhledávací tabulka vektorových reprezentací slov – obsahuje matici reálných čísel, jejíž rozměry jsou $|V| \times n$, kde $|V|$ je velikost slovníku a n délka vektorů, které reprezentují jednotlivá slova – každá řádka matice reprezentuje jedno slovo slovníku. Vstupem *embedding* vrstvy je seznam indexů slov ve slovníku (reprezentace vstupního dokumentu), podle kterých budou v matici vyhledána příslušná slova. Z těchto vektorů slov je vytvořena mapa příznaků, která slouží jako vstup konvoluční vrstvy. Tato vrstva (resp. matice $|V| \times n$) je pochopitelně trénovatelná stejně jako všechny ostatní vrstvy sítě, aby byly vektorové reprezentace slov optimalizovány pro úlohu NLP, pro kterou je neuronová síť trénována.

3.6 *Multi-label* klasifikace

Většina popsaných metod je v základní podobě *single-label* klasifikací, některé jen binární. Pro použití *single-label* metody pro *multi-label* klasifikaci lze často využít pouze prahování (viz kapitola 2.1.1), ale většinou existují i komplikovanější způsoby, kdy je přímo v algoritmu klasifikační metody zohledněno, že dokument může náležet více kategoriím(NBC [79], kNN [13; 86], ME [88], NN [85]). V případě binární klasifikace se pak často využívá tzv. *one-vs-all* strategie, kdy je natrénováno tolik binárních klasifikátorů, kolik existuje kategorií, a každý z nich rozhoduje o zařazení dokumentu do jedné příslušné kategorie (SVM [43]).

4 Reprezentace dokumentů

Nejčastěji je dokument reprezentován jako vektor (někdy matice) reálných čísel; jednotlivé položky vektoru se často nazývají **příznaky**. Před vlastní tvorbou příznakového vektoru je často text různě předzpracován nebo se provádí výběr příznaků.

Reprezentaci dokumentů příznakovým vektorem (maticí) lze rozdělit na dva způsoby – *sparse* reprezentaci (kapitola 4.2) a *dense* reprezentaci (kapitola 4.3) – „řídká“ a „hustá“ reprezentace[16]. Kapitola 4.4 pak stručně zmiňuje metody výběru relevantních příznaků.

4.1 Předzpracování textu

Smyslem předzpracování textu je najít v dokumentech jednotlivá slova (tokenizace) a případně odstranit slova (stop slova) a znaky, které nenesou žádnou informaci o tom, do které kategorie by dokument mohl patřit. Slova lze také nahradit jejich jinou formou (stematizace a lemmatizace), která by měla nést lepší informaci.

Tokenizace

Tokenizace je proces rozdělení dokumentu na jednotlivá slova. Nejjednodušší metoda je dělení podle mezer (resp. bílých znaků), ale to je obvykle nedostačující. Například každé slovo na konci věty by mělo u sebe tečku. Proto se navíc odstraňují všechny nežádoucí znaky (někdy i všechna čísla), nebo je také možné rovnou rozdělit dokument na jednotlivá slova podle všech znaků, které nejsou písmeny. Velice často se také velká písmena nahrazují malými.

Stop slova

Za stop slova (*stop words*) se označuje seznam slov, která jsou explicitně z textu odstraněna. Jedná se o slova jako jsou spojky, předložky a další slova, u kterých lze očekávat, že nemají žádný význam pro správné určení kategorie. Tento seznam slov je obvykle pevně předem daný[17], ale ne pro každou úlohu je vhodný ten samý. Například slova jako „ok“, „not“ nebo

[16] České překlady se spíše nepoužívají.

[17] Např. `https://github.com/miso-belica/sumy/tree/dev/sumy/data/stopwords`

„never“ pravděpodobně nemají význam při klasifikaci novinových článků, ale pro úlohu určení sentimentu mohou být užitečná.

Stematizace

Stematizace [47] (*stemming*) je proces nalezení kmenu slova (resp. nahrazení všech slov odvozených od stejného kmenu jedním „slovem“). Toto lze provádět buď algoritmicky, kdy se podle určitých pravidel, která jsou samozřejmě pro každý jazyk jiná, odtrhávají a mění koncovky slova, nebo statisticky. Příkladem prvního způsobu je velice známý *Porter stemmer* [54], ale existují i další [3]. Stejně tak statistických přístupů existuje několik [3].

Lemmatizace

Lemmatizace [47] (*lemmatization*) je proces přeměny slova na základní (slovníkový) tvar (resp. nahrazení všech slov odvozených od stejného základu jedním slovem), tzv. lemma. Na rozdíl od stematizace, zde je výsledkem existující slovo.

Part-of-speech značkování

Part-of-speech (POS) značkování [46] je proces přiřazení slovního druhu[18] jednotlivým slovům. Toho lze využít např. odstraněním slovních druhů, které mohou být pro klasifikaci (či jinou úlohu) zbytečné.

4.2 *Sparse* reprezentace

Sparse reprezentace znamená, že dokument (případně slovo) je reprezentován vektorem, který má na většině pozic nuly. Každá položka vektoru (příznak) odpovídá jednomu slovu ve slovníku a tedy délka vektoru se rovná počtu slov ve slovníku. Mezi nejčastěji používané *sparse* reprezentace patří *one-hot* a *Bag-of-words* model (BoW). Těmto reprezentacím, kdy každá položka reprezentujícího vektoru odpovídá jednomu slovu slovníku, se také někdy říká „lokální reprezentace“.

Je zřejmé, že vektor může být velice dlouhý (náročné na paměť apod.), pokud existuje v použité datové sadě hodně unikátních slov. Velice často se proto slovník (a tím pádem i délka vektoru) omezuje na několik desítek tisíc nejfrekventovanějších slov.

[18] Ale nemusí to vždy být jen slovní druhy. Někdy se POS značkování rozšiřuje například na označení interpunkce, která může být pro některé úlohy NLP důležitá.

One-hot reprezentace

Při *one-hot* reprezentaci je každé slovo reprezentováno vektorem, kde pouze jediná pozice obsahuje hodnotu 1, ostatní jsou nulové. Tato pozice odpovídá indexu daného slova ve slovníku. Dokument je potom reprezentován maticí, složenou po řádkách ze slov dokumentu.

Pro většinu klasifikačních algoritmů je ovšem potřeba, aby byly dokumenty reprezentovány maticí (vektorem) konstantních rozměrů, proto se z každého dokumentu použije několik počátečních slov (nejčastěji několik set). Pokud je dokument kratší, je obvykle matice doplněna nulami.

Co-occurence vektor

Tato reprezentace vychází z *co-occurence matrix* (matice spolu-výskytů), do které se zanáší, jak často se různá slova vyskytují blízko sebe (např. v jedné větě). Tato matice má tolik sloupců a řádků, kolik je slov ve slovníku. Na pozici $m_{i,j}$ bude celé číslo, které udává, kolikrát se slova i a j vyskytují u sebe. Slovo je potom reprezentováno příslušnou řádkou resp. sloupcem matice (tato matice je symetrická). Tuto reprezentaci lze chápat jako kontext slova.

Bag-of-words model

Bag-of-words model reprezentuje každý dokument vektorem, který má nenulové hodnoty na pozicích, odpovídajících indexům slov, obsažených v tomto dokumentu. Existují různé varianty, jak tyto hodnoty získat nebo vypočítat. Mezi základní a často používané metody patří tyto:

- **Binární vektor** – ve vektoru je na dané pozici hodnota 1, pokud dokument odpovídající slovo obsahuje, jinak 0.
- **Frekvence** – na každé pozici vektoru je číslo, které značí počet výskytů slova v dokumentu. Buď lze jako absolutní počet výskytů, nebo relativní k délce dokumentu (tzn. počet výskytů je vydělen celkovým počtem slov dokumentu).
- **Vážená frekvence** [63; 64; 81] (*tf-idf*) – kombinuje relativní frekvenci slova v dokumentu (*tf*, neboli *term frequency*) s „důležitostí“ slova v celém korpusu (*idf*, *inverse document frequency*). Pro slovo i v dokumentu j, ji lze vypočíst vzorcem 4.1, kde $n_{i,j}$ je počet výskytů slova i v dokumentu j, $|d_j|$ počet slov v dokumentu j, N celkový počet dokumentů a df_i počet dokumentů, které obsahují dané slovo.

$$w_{i,j} = \frac{n_{i,j}}{|d_j|} \cdot \log_e \frac{N}{df_i} \tag{4.1}$$

Výpočet tf-idf.

Existují i pokročilejší metody výpočtu vážených frekvencí [36; 44], ale těmi se tato práce nebude zabývat.

4.3 *Dense* reprezentace

Tato kapitola bude nejprve (a převážně) pojednávat o *dense* reprezentacích jednotlivých slov (až dále o reprezentacích dokumentů), protože základní metody generování *dense* reprezentací jsou většinou navržené pro generování slovních reprezentací. Metody pro vytváření *dense* reprezentací dokumentů (případně jiných kusů textu, složených ze slov) většinou staví na reprezentacích slov (nebo na algoritmech, které je vytváří). Při *dense* reprezentaci má příznakový vektor na většině pozic nenulové hodnoty. Také bývá výrazně kratší, než u *sparse* reprezentací, což mj. může být vhodné pro některé algoritmy klasifikace, např. pro konvoluční neuronové sítě, kde by při použití *sparse* reprezentací bylo moc trénovatelných parametrů[19]. Nicméně za určitých úprav je možné klasifikovat pomocí CNN nad BoW reprezentací [29].

Některé *dense* reprezentace vycházejí z *co-occurence* matice[20], např. singulární dekompozice [31] (*singular value decomposition*, SVD), kdy je cílem zachovat informaci (kontext slova), ale zároveň drasticky snížit dimenzi matice[21]. Další metodou pro tvorbu *dense* reprezentací, která vychází z počítání slov (kontextů slov) v korpusu, je latentní sémantická analýza [37] (*latent semantic analysis*[22], LSA), kde se vektory slov (resp. matice těchto vektorů) vytváří použitím SVD[23] na matici počtů slov v dokumentech (řádky slova, sloupce dokumenty). Dále lze pro tvorbu *dense* reprezentací využít např.

[19]Což teoreticky nedělá problém učícímu algoritmu jako takovému, ale síť by mohla zabrat daleko více paměti a učení by trvalo podstatně déle.

[20]Ne nezbytně je to matice spolu-výskytů slov. Některé metody používají matici četností výskytů slov v dokumentech

[21]Metoda SVD nebyla vyvinuta za tímto účelem. Je to metoda maticové faktorizace, kterou lze využít pro redukci dimenzionality, ale výsledné vektory lze za *dense* reprezentaci považovat.

[22]Někdy také *latent semantic indexing*, LSI.

[23]LSA není jedná metoda, která používá SVD jako svoji součást. Existuje např. pokročilejší *LexVec* [61; 62], kde je kombinováno několik dalších technik pro získání kvalitnějších distribuovaných reprezentací slov.

analýzu hlavních komponent (*principal component analysis*, PCA), což je metoda pro redukci dimenzionality se snahou zachovat nejdůležitější příznaky[24]. Mezi novější metody patří např. poměrně hojně využívaná metoda GLoVe [52], kde jsou vektory slov počítány z *co-occurence* matice specificky upravenou váženou metodou nejmenších čtverců.

4.3.1 Distribuované reprezentace

Distribuovaná reprezentace se vysvětluje tak, že jeden příznak reprezentujícího vektoru obsahuje informace o více slovech a zároveň je informace o jednom slovu obsažena (distribuována) ve více příznacích [25]. Dále nejsou tyto reprezentace tzv. *count-based* (založené na počítaní spolu-výskytů slov jako výše uvedené metody), ale *prediction-based* – modely založené většinou na neuronových sítích předpovídají slovo z jeho kontextu [5; 6; 15; 25] (případně naopak).

Reprezentace slova (případně jiné entity) krátkým *dense* vektorem reálných čísel se často nazývá ***embedding***[25]. Smyslem těchto reprezentací je zachycení skrytých významů a vztahů slov. Vektor je umístěn ve spojitém vektorovém prostoru a očekává se, že sémanticky podobné entity budou reprezentovány podobnými vektory.

4.3.2 Distribuované reprezentace slov

Známým nástrojem pro generování distribuovaných reprezentací slov na základě předpovědí slova z kontextu je *word2vec*[26] [48]. *Word2vec* je důležitý, protože navržené algoritmy pro generování distribuovaných reprezentací používají jednodušší modely neuronových sítí než předchozí přístupy, což zvyšuje efektivitu výpočtu a dovoluje trénovat na více datech či méně výkonných přístrojích – oproti modelu *Neural Probabilistic Language Model* [6], ze kterého *word2vec* vychází, jsou v *projection* vrstvě modelu vektory slov průměrovány místo skládání za sebe a jedna skrytá vrstva modelu je odstraněna úplně.

Mezi další pomerně moderní přístupy patří např. *fastText* [9], který je v podstatě rozšířením modelů *word2vec* o zahrnutí informace „uvnitř“ slova – znakové n-gramy. Slovo je v kontextu *fastText* chápáno jako součet n-gramů,

[24]Podobně jako SVD, PCA je především nástroj pro redukci dimenzionality, ale výsledné vektory lze použít jako *dense* reprezentaci. Podobných metod existuje více [67].

[25]Do češtiny se nepřekládá.

[26]Nyní jsou ale pod názvem *word2vec* známé spíše algoritmy pro vytvářeni distribuovaných reprezentací, které původní nástroj implementoval.

které toto slovo obsahuje, což může podle autorů být užitečné především pro jazyky s ohebnými slovními druhy, kde jedno slovo může kvůli skloňování a časování nabývat mnoha různých tvarů.

Výhodou těchto slovních reprezentací je jejich přenositelnost – často je možné pro určitou úlohu NLP použít předtrénované slovní vektory (např. autoři metody *fastText* poskytují volně ke stažení vektory natrénované z mnoha různých jazyků[27]), které byly natrénované na velkém množství dat (miliardy slov), místo generování vektorů vlastních, které nemusí být tak kvalitní (např. protože byly trénovány na nedostatečném množství dat). Také lze šetřit úsilí a čas, které by musely být generování vektorů věnovány.

Word2vec

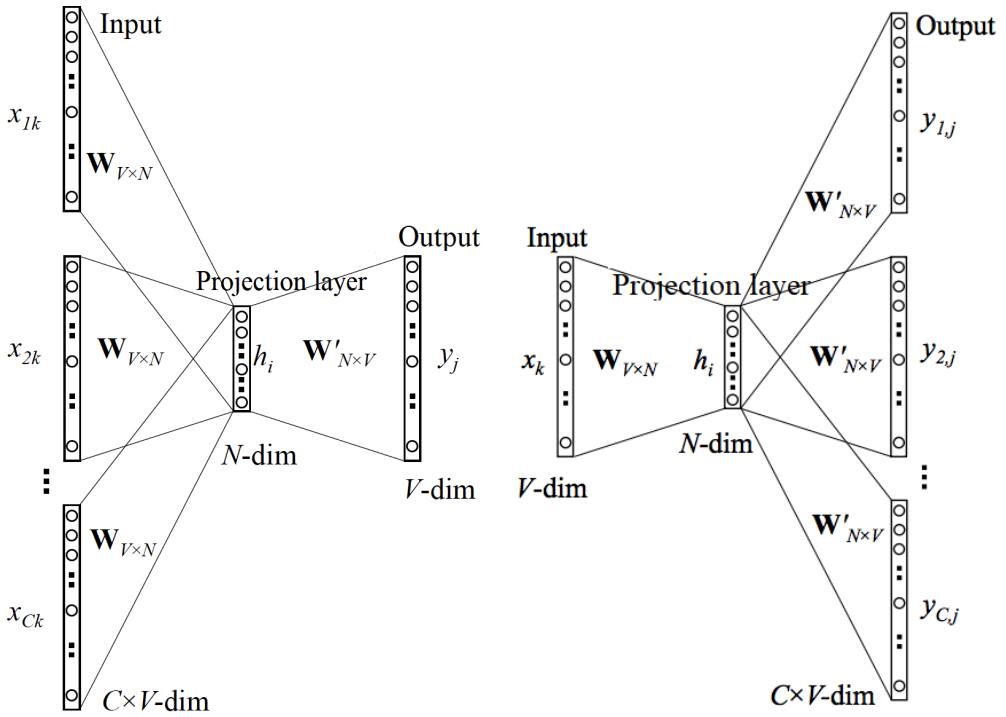
Pod *word2vec* spadají dvě metody výpočtu distribuovaných reprezentací slov. První z nich je **CBOW** (*continuous bag-of-word*), kdy je slovo předpovídáno z kontextu (několik okolních slov), a ***Skip-gram*** (SG), kdy je kontext předpovídán ze slova. CBOW nebere v úvahu pořadí slov (odtud *bag-of-words* v názvu), ale *Skip-gram* přiřazuje bližším slovům větší váhu než vzdálenějším. Tyto dvě architektury jsou znázorněny na obrázku 4.1, kde C je délka kontextu (počet slov v okolí aktuálního slova), V počet slov ve slovníku a N délka reprezentací slov (resp. počet neuronů ve skryté vrstvě). Výsledné slovní vektory jsou po trénování obsaženy ve sloupcích matice vah W mezi vstupní a *projection* vrstvou.

Embedding vrstva

Za příbuznou distribuovaným reprezentacím slov lze do určité míry považovat i *embedding* vrstvu (kapitola 3.5.2). Podstatný rozdíl je, že slovní vektory *embedding* vrstvy nejsou trénovány přímo z textu, ale jsou prakticky pouze dalšími trénovatelnými váhami neuronové sítě a jsou tedy optimalizovány přímo pro konkrétní NLP úlohu. Obvykle je nelze jen tak vzít a využít v jiné úloze. Na druhou stranu je možné, že ze stejného důvodu jsou pro jednu konkrétní úlohu, pro kterou byly natrénovány, kvalitnější reprezentací slov, než obecnější slovní vektory vytvořené např. metodou *word2vec*.

Váhy *embedding* vrstvy (slovní vektory) ale nemusí být inicializovány náhodně – do vrstvy lze vložit předtrénované vektory (např. *word2vec*) a tím přinést externí informaci, což může pro některé úlohy NLP zvýšit výslednou

[27] `https://fasttext.cc/docs/en/crawl-vectors.html`



(a) CBOW *(b) Skip-gram*

Obrázek 4.1: Architektura word2vec *modelů*[28].

kvalitu natrénovaných slovních vektorů [32], resp. tyto úlohy dosáhnou lepších výsledků díky kvalitnějším slovním vektorům. Také je možné zakázat trénování vah *embedding* vrstvy, což znamená, že vložené vektory zůstanou nezměněny. Použití *embedding* vrstvy s vloženými vektory je ekvivalentní jejímu nahrazení normální vstupní vrstvou (tj. místo sekvence indexů pro vyhledávací tabulku *embedding* vrstvy bude vstupem vstupní vrstvy matice slovních vektorů).

4.3.3 Distribuované reprezentace vět a dokumentů

Distribuované reprezentace lze vytvářet i pro větší jednotky textu, než slova (např. věty nebo dokumenty). Mezi základní jednoduché metody patří např. průměrování slovních vektorů, případně různě vážené průměrování. Na těchto přístupech lze stavět a různě je vylepšovat [4; 16; 80]. Jedna z komplikovanější metod generování distribuovaných reprezentací vět využívá derivační strom pro určení pořadí, v jakém jsou slova v dokumentu pomocí určitých

[28] Obrázky s malými úpravami převzaty z `https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/`.

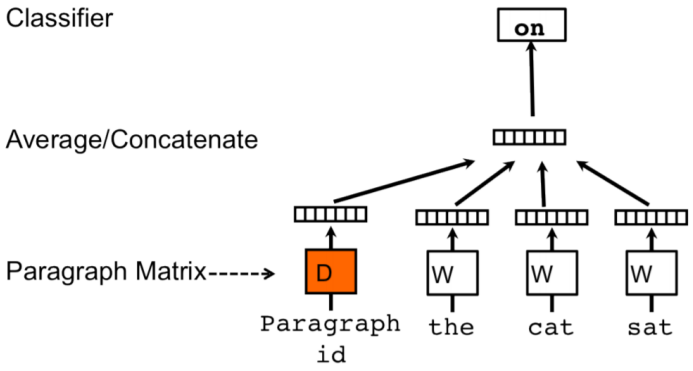
operací kombinována[29] [68]. Dále existuje známá metoda *Paragraph vector*[30] [39], což je rozšíření obou modelů *word2vec*. V těchto modelech při předpovědi slova z kontextu (resp. kontextu ze slova) hraje roli kromě jednotlivých slov i dokument, ve kterém se slovo nachází. Další poměrně známou metodou je *Sent2vec* [51], která pro získání reprezentací vět algoritmicky staví na modelu CBOW z *word2vec*, ale stejně jako *fastText* využívá znakové n-gramy.

Také je možné dokument (větu) reprezentovat maticí místo vektoru, kdy je tato matice po řádkách složena z reprezentací slov. Stále platí omezení, které bylo zmíněno u *one-hot* reprezentace (kap. 4.2) – obvykle je nutné určit pevný počet slov, aby měly matice pro různé dokumenty (věty) stejné rozměry.

Doc2vec

Doc2vec (*Paragraph vector*) rozšiřuje oba modely výpočtu distribuovaných reprezentací slov z *word2vec* pro výpočet distribuovaných reprezentací dokumentů (resp. jiných, libovolně dlouhých kusů textu).

- ***Distributed Memory*** (DM) – Tento model (viz obrázek 4.2) je rozšířením CBOW a je mu velice podobný. Jediný rozdíl je, že k předpovědi slova je kromě jeho kontextu navíc použit vektor dokumentu. Každý dokument je reprezentován jedním sloupcem (trénovatelné) matice D (Stejně jako slova jsou reprezentována ve sloupcích matice W).



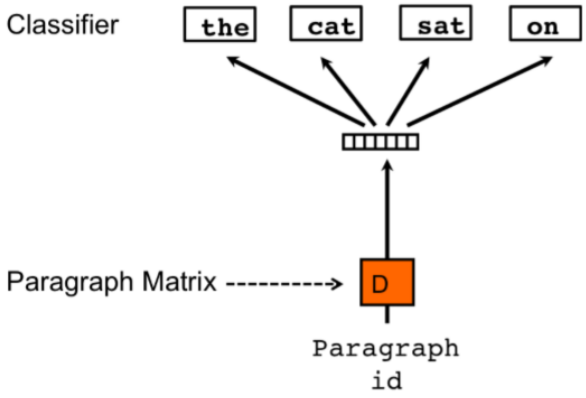
Obrázek 4.2: Architektura modelu DM[31].

[29] Samozřejmě jde o operace, u kterých záleží na pořadí operandů.

[30] Známá i pod názvem *doc2vec*.

[31] Obrázek převzat z `https://medium.com/@amarbudhiraja/understanding-document-embeddings-of-doc2vec-bfe7237a26da`.

- ***Distributed Bag of Words*** – DBOW (obr. 4.3) je variací modelu *Skip-gram*. Pro předpověď kontextu slova je místo slova použit pouze vektor dokumentu. V původní implementaci [39], tzv. *pure DBOW*, jsou trénovány pouze vektory dokumentů, ale existují některé implementace (např. ve frameworku Gensim [59]), které dovolují trénovat zároveň vektory slov stejným způsobem jako ve *Skip-gram* modelu.



Obrázek 4.3: Architektura modelu DBOW[32].

4.4 Výběr příznaků

Často může být vhodné snížit počet příznaků, resp. vybrat vhodné příznaky (odstranit nevhodné). U *sparse* reprezentací mj. kvůli zmenšení vektorů, ale obecně je tím často zlepšena úspěšnost klasifikace. Za základní metody, nezávislé na konkrétních datech, by se daly považovat i metody předzpracování textu, jako jsou např. prosté odstranění stop slov a POS značkování, nebo v případě *sparse* reprezentací omezení velikosti slovníku. Existují však i komplexnější metody, souhrnně pod názvem „výběr příznaků“ (angl. *feature selection*), kde je důležitost každého slova vypočítána z konkrétních dat.

Základem všech metod výběru příznaků je předpoklad, že každé slovo (příznak) přispívá správné klasifikaci dokumentu různou měrou. Stejně jako u předzpracování textu je ve většině případů žádoucí odstranit z dokumentů ta slova, která nesou malou nebo žádnou informaci. Mezi známé metody výběru příznaků [22; 82] patří například: dokumentová frekvence (DF), *Information gain* (IG), *Mutual Information* (MI), GSS koeficient, *Term strength* (TS), nebo chí-kvadrát (χ^2).

[32]Obrázek převzat z `https://medium.com/@amarbudhiraja/understanding-document-embeddings-of-doc2vec-bfe7237a26da`.

5 Datové sady

Pro testování úspěšnosti klasifikace navrženými metodami jsou k dispozici dva korpusy – český a anglický. Jejich popis následuje v kapitolách 5.1 a 5.2.

Všechny dokumenty z obou datových sad byly podrobeny standardnímu textovému předzpracování - nahrazení velkých písmen malými, odstranění čísel, stop slov[33] a různého šumu (interpunkce apod.). Přestože vyžadují tyto metody poměrně malé úsilí, obvykle zlepšují výsledek klasifikace [66; 72; 77].

Dále přicházela v úvahu lemmatizace. Přestože se díky ní může výrazně zmenšit slovník všech unikátních slov korpusu a u českého korpusu se lemmata již nacházejí (viz kapitolu 5.1), vliv na úspěšnost klasifikace je diskutabilní; některé zdroje uvádějí pozitivní [26; 27], jiné negativní [10; 75] vliv. Vzhledem k tomu, že zkoumat vliv různých nadstandardních metod předzpracování na kvalitu klasifikace není předmětem této práce, nebude lemmatizaci věnována větší pozornost.

Ze stejných důvodů nebylo využito ani POS značkování – není cílem práce jej zkoumat a bylo by nutné jej provést pro anglický korpus, protože se POS značky nacházejí jen u českého korpusu.

5.1 Czech Text Document Corpus v 2.0

Tato datová sada[34], představená v [33], obsahuje novinové články vydané Českou tiskovou kanceláří[35] (ČTK). Byla vytvořena za účelem testování a porovnávání různých přístupů ke klasifikaci dokumentů. Hodí se také pro *multi-label* klasifikaci, protože každý dokument je zařazen do více kategorií. Korpus obsahuje ke každému článku (dokumentu) také výsledky morfologické analýzy (vytvořené nástrojem UDPPipe[36]). Datová sada je licencována pod Attribution-NonCommercial-ShareAlike 3.0 Unported License[37].

[33] Použity byly seznamy stop slov z odkazu v kapitole 4.1, tj. `https://github.com/miso-belica/sumy/blob/dev/sumy/data/stopwords`

[34] Ke stažení na `http://ctdc.kiv.zcu.cz/`.

[35] `http://www.ctk.cz/`

[36] `http://www.aclweb.org/anthology/K/K17/K17-3009.pdf`

[37] `https://creativecommons.org/licenses/by-nc-sa/3.0/`

5.1.1 Struktura

Název každého dokumentu je ve formátu *id_kat1_kat2_...._katN.přípona*, kde *id* je pěticiferné identifikační číslo dokumentu a *kat1* až *katN* jsou kódy kategorií, do kterých dokument náleží (v příloze B seznam kategorií). Dokument je reprezentován 5 následujícími soubory:

- **.txt* – Prostý text článku.
- **.tok* – Text rozdělený na jednotlivá slova (tokeny) včetně interpunkce.
- **.lemma* – Stejné jako *název.tok*, ale všechna slova jsou nahrazena příslušným lemmatem.
- **.pos* – Part-of-speech (POS) značky, které určují slovní druh jednotlivých slov, nebo označují interpunkci.
- **.conll* – Článek v CoNLL-U[38] formátu. V příloze A lze nalézt ukázku jedné věty v tomto formátu.

Zajímavými soubory jsou **.tok*, protože není potřeba řešit tokenizaci dokumentů, **.lemma*, pokud budou použita pro klasifikaci místo původních slov příslušná lemmata, a **.conll*, protože obsahuje dokument rozdělený po větách, což lze využít jako základ pro hierarchickou metodu reprezentace dokumentu. Přestože POS značky mohou mít pozitivní vliv [10; 26] na kvalitu klasifikace, nejsou zde vzhledem k zadání práce důležité, a proto jim nebude věnována pozornost.

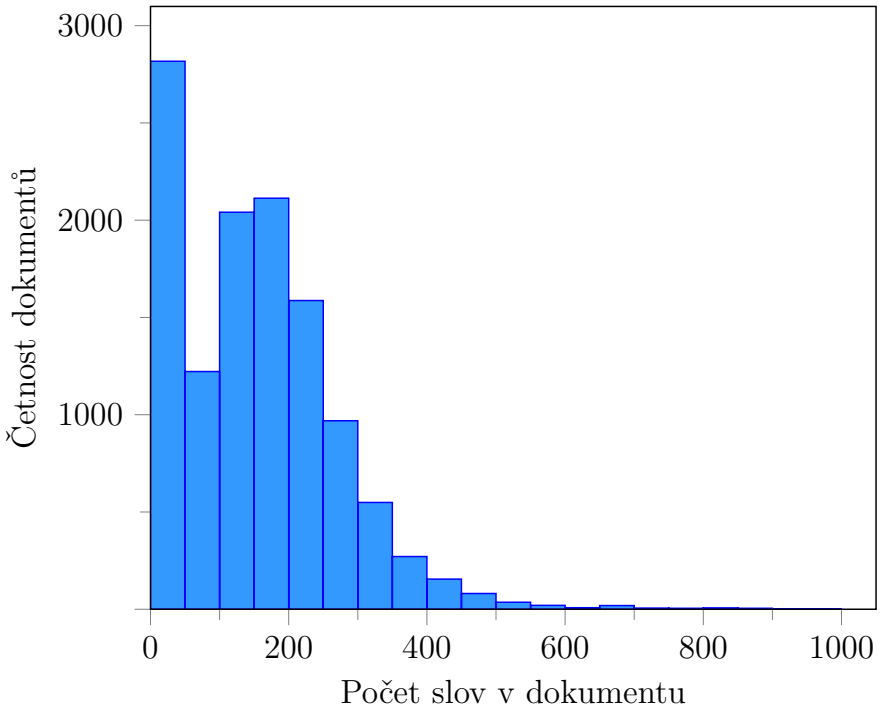
Samozřejmě je možné rozdělit původní dokument na věty znovu, avšak při letmém prozkoumání několika dokumentů se zdá, že jsou rozděleny na věty převážně správně. Dále také **.conll* soubory obsahují věty rozdělené na jednotlivá slova a příslušná lemmata, která ale, jak bylo řečeno v úvodů kapitoly, nebudou využita. Vzhledem k tomu, že soubory **.conll* obsahují všechny potřebné informace, budou využity pouze tyto soubory.

5.1.2 Statistické údaje

Korpus obsahuje 11955 trénovacích dokumentů a 2735 validačních, ze kterých je ale 197 (7, 2%) odstraněno kvůli omezení nedostatečně zastoupených kategorií (viz dále v kapitole 5.1.2). Aplikace tohoto omezení na trénovací sadu z ní neodstraní žádný dokument.

[38] `http://universaldependencies.org/docs/format.html`

V histogramu 5.1 lze vidět distribuci četností trénovacích dokumentů podle počtu slov v plně předzpracovaných dokumentech – s odstraněnou interpunkcí, čísly a stop slovy. Z histogramu je patrné, že nejvíce dokumentů, kolem 2900, má méně než 50 slov. Několik dokumentů v histogramu chybí – jedná se o dokumenty, které mají více, než 1000 slov. Celkově jde o 41 dokumentů. Jeden dokument (včetně vynechaných dokumentů) má průměrně 162 slov, medián je 182.



Obrázek 5.1: Distribuce dokumentů podle četností slov (bez interpunkce).

V tabulce 5.2 je v prvním sloupci celkový počet slov v trénovací množině, ve druhém sloupci počet unikátních slov a ve třetím sloupci počet unikátních lemmat s interpunkcí (**i**), bez interpunkce (**bi**) a konečně bez interpunkce, stop slov a čísel (**bis**). Ve čtvrtém sloupci je uvedeno procentuální snížení velikosti slovníku (počtu unikátních slov) při použití lemmat místo původních slov. Hodnoty byly naměřeny ze souborů **.conll*. Pro spočítání unikátních slov byla ve všech slovech velká písmena nahrazena malými, jinak by stejné slovo uprostřed a na začátku věty bylo počítáno jako dvě unikátní slova, což je nežádoucí[39].

[39]Pokud se bude dále mluvit o předzpracovaných dokumentech, vždy se předpokládá nahrazení velkých písmen malými, pokud nebude řečeno jinak.

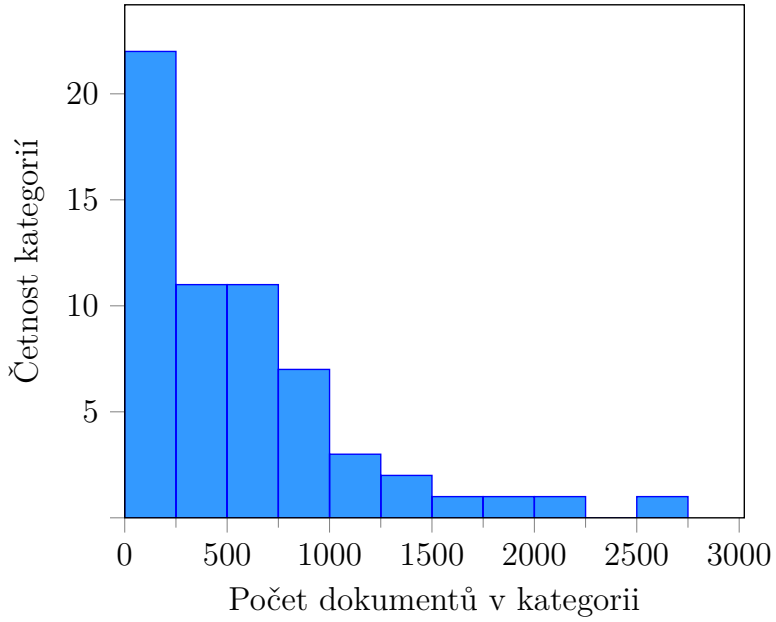
	Počet slov	**Unikátní slova**	**Unikátní lemmata**	**Snížení počtu v %**
i	3505965	150899	82986	45,0
bi	2978678	145536	78368	46,2
bis	1931523	137402	70343	48,8

Tabulka 5.2: Počet slov v české datové sadě podle různé úrovně předzpracovaní.

Kategorie dokumentů

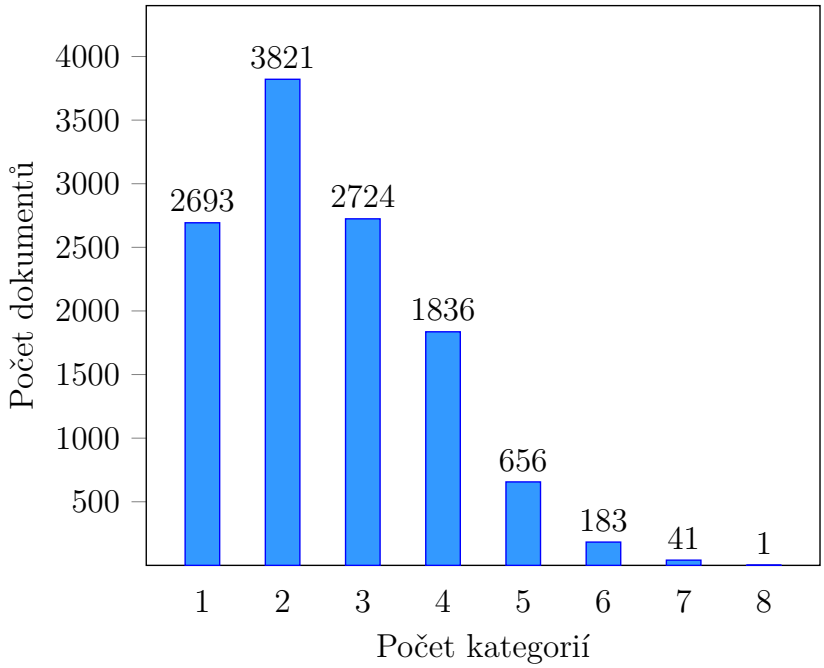
Každý dokument může (ale nemusí) náležet do více kategorií. Celkově existuje 60 kategorií (seznam s četnostmi v trénovací množině v příloze B), z nichž ale bude pro klasifikaci po vzoru [33] použito 37 nejčetnějších, protože ostatní mají statisticky nevýznamnou četnost. Poslední neplatné kategorii přísluší 258 dokumentů, kdežto první platné 262 dokumentů.

Na obrázku 5.3 lze vidět histogram počtů dokumentů v jednotlivých kategoriích v trénovací množině. Kategorie, na které nebude při klasifikaci brán zřetel, přibližně odpovídají prvnímu sloupci. Dokumenty, které nemají po odstranění neplatných kategorií žádnou přiřazenou kategorii, jsou taktéž označeny za neplatné a při trénování i testování klasifikátoru jsou ignorovány – z trénovací množiny se to netýká žádného dokumentů, z validační množiny je tímto odstraněno 197 dokumentů z celkového počtu 2735, tj. $7,2\%$.



Obrázek 5.3: Distribuce kategorií podle počtu trénovacích dokumentů.

Jak bylo výše řečeno, každý dokument může náležet různému počtu kategorií. Na obrázku 5.4 se nachází histogram počtu kategorií u jednotlivých dokumentů v trénovací množině (po odstranění neplatných kategorií). Nejčastěji mají dokumenty dvě kategorie, přičemž průměrný počet je $2,553$.



Obrázek 5.4: Histogram počtu kategorií u dokumentů (po odstranění neplatných kategorií) v trénovací množině.

5.2 Reuters Corpus Volume 1 (RCV1)

Korpus RCV1, představený v [60] a detailně popsaný v [42], obsahuje 806791 novinových článků vydaných zpravodajskou agenturou Reuters[40] mezi 20. srpnem 1996 a 19. srpnem 1997. Každý článek je obsažen v samostatném strukturovaném `xml` souboru (ukázka v příloze C) s názvem ve formátu `idnewsML.xml`, kde `id` je 5 nebo 6 ciferné unikátní identifikační číslo dokumentu. Kromě textu obsahuje soubor nadpis, různé kódy (*topic codes*, *region codes* a *industry codes*, resp. kategorie, místo původu a průmyslový kód), místo a datum původu a další. Pro tuto práci je zajímavý text, nadpis a kategorie (*topic code*).

Kategorie, v celkovém počtu 103, jsou uspořádány do hierarchie. Každá kategorie, kromě čtyř kořenových, má rodičovskou kategorii (případně více

[40] `https://www.reuters.com/`

úrovní rodičů). Pro trénování a testování klasifikátoru jsou ale všechny kategorie považovány za rovnocenné.

Některé soubory obsahují chyby, které byly pokud možno opraveny podle [42], nebo byly příslušné dokumenty označeny za neplatné a zahozeny. V [42] autoři nazvali upravený korpus RCV1-v2, zde bude nazván **RCV1-v2a**, protože provedené úpravy se mírně liší – např. u některých dokumentů chybí *Region code* a proto tyto dokumenty nejsou zahrnuty v RCV1-v2. Pro účely této práce chybějící *Region code* není podstatný a dokument bude zahrnut v RCV1-v2a.

Proces přeměny RCV1 na RCV1-v2a zde sestává pouze z těchto dvou kroků:

1. Dokumenty, které nemají žádné kategorie, byly odstraněny. Týká se to 2364 dokumentů.

2. U některých dokumentů v RCV1 je špatně hierarchie kategorií; přestože dokument má přiřazenou kategorii níže v hierarchickém stromu, chybí mu jedna nebo více rodičovských kategorií. Všechny chybějící kategorie byly doplněny – celkově 25402 v 14786 dokumentech.

5.2.1 Statistické údaje – RCV1-v2a

Po vzoru [42] jsou pro trénování klasifikátoru použity dokumenty s identifikačním číslem mezi 2286 a 26150 (včetně), což odpovídá dokumentům publikovaným mezi 20. a 31. srpnem 1996. Ostatní dokumenty, s identifikačním číslem mezi 26151 a 810956, jsou použity jako testovací množina. Celkově je tedy 23149 trénovacích dokumentů a 781278 testovacích, dohromady 804427[41].

V tabulce 5.5 lze vidět počty slov[42] v trénovací sadě (23149 dokumentů) – v jednotlivých sloupcích se nachází celkový počet slov, počet unikátních slov, průměrný počet slov v jednom dokumentu a medián počtu slov. V první řádce (**bi**) jsou údaje pro text pouze s odstraněnou interpunkcí[43], ve druhé (**bis**) pro plně předzpracované dokumenty, tj. i s odstraněnými stop slovy a čísly.

[41] Čísla se mírně liší od počtu dokumentů v RCV1-v2 v [42], protože autoři odstraňují i dokumenty, kterým chybí *region codess*. Konkrétně se jedná o 13 dokumentů z testovací množiny.

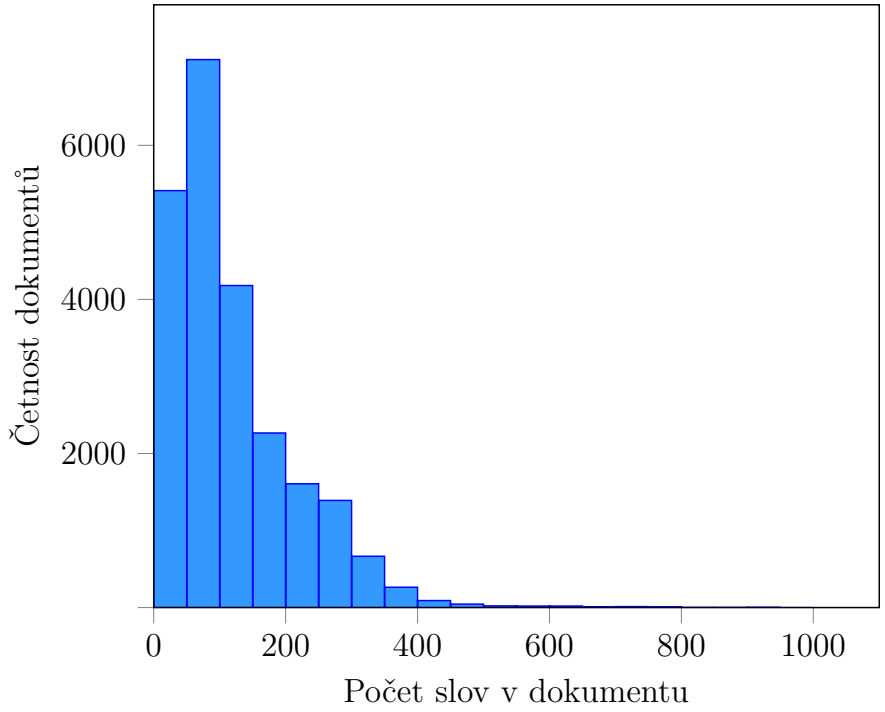
[42] Do textu dokumentu je zahrnut i nadpis článku.

[43] A stejně jako u českého korpusu s nahrazenými velkými písmeny.

	Počet slov	**Unikátní slova**	**Průměrně slov**	**Medián slov**
bi	5659953	75348	245	155
bis	2798217	68494	121	85

Tabulka 5.5: Celkové a průměrné počty slov v anglickém korpusu podle různé úrovně předzpracovaní.

V histogramu 5.6 je zanesena distribuce dokumentů v trénovací množině podle počtu slov (plně předzpracované, tzn. odpovídá druhé řádce tabulky 5.5). Lze vypozorovat, že jsou opět dominantní dokumenty s nízkým počtem slov – přibližně 12500 dokumentů má méně, než 100 slov.



Obrázek 5.6: Distribuce dokumentů podle četností slov (bez interpunkce, stop slov a čísel).

Kategorie

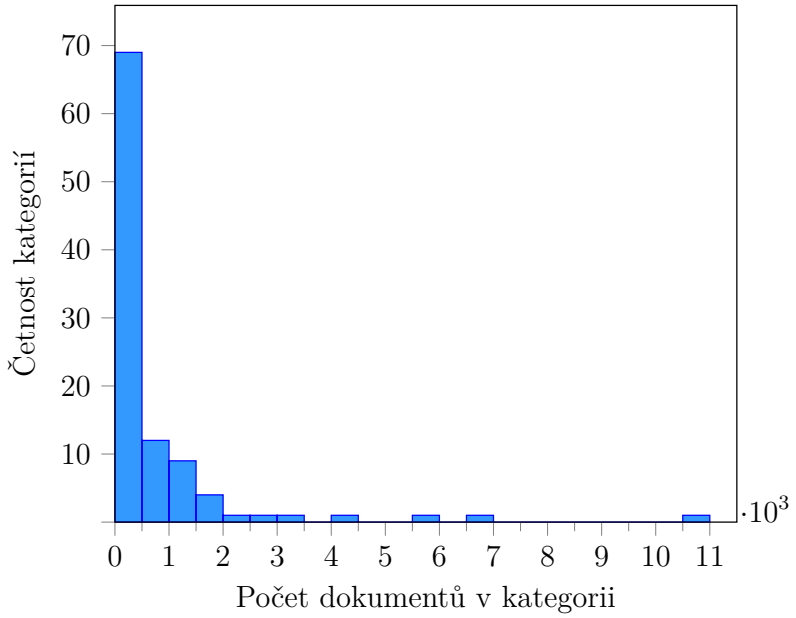
Z celkových 103 kategorií je v trénovací množině 101 kategorií, které jsou přiřazené alespoň jednomu dokumentu. Dvě přebývající kategorie budou pro testování ignorovány; z testovací množiny se to týká pouhých 57 dokumentů. Seznam těchto 103 kategorií lze nalézt v příloze D, kde přeškrtnutím jsou vyznačeny 2 ignorované kategorie.

Kvůli stromové hierarchii kategorií (kde jsou pouze 4 kořenové kategorie) a poměrně podrobnému dělení na jednotlivé kategorie v listech stromu je velký rozdíl v počtu dokumentů patřících do jednotlivých kategorií (především mezi listovými kategoriemi a těmi vysoko v hierarchii). Z tabulky 5.7 lze vypozorovat, že přestože listových kategorií je přibližně 80%, jejich celkový počet přiřazení je pouze 38%.

Podmnožina korpusu	**Kategorie**	**Počet kategorií**	**Počet přiřazení**
Trénovací	listy hierarchie	80	27776 (38%)
	ostatní	21	45921 (62%)
Vše	listy hierarchie	82	986225 (38%)
	ostatní	21	1620684 (62%)

Tabulka 5.7: Počet přiřazení kategorií pro trénovací množinu a pro všechny dokumenty v RCV1-v2a.

Z celkového počtu 73697 přiřazení v trénovací množině jich náleží 20117 čtyřem kořenovým kategoriím, z čehož 10786 přiřazení náleží jediné z nich. Na druhé straně existuje 33 kategorií, které mají méně než 100 přiřazení (některé i méně než 10) a dohromady pouhých 1347 přiřazení. Z histogramu na obrázku 5.8 je patrné, že převážná většina kategorií (necelých 70) má méně než 500 přiřazených dokumentů.



Obrázek 5.8: Distribuce kategorií podle počtu trénovacích dokumentů.

Tyto informace nejsou nikterak překvapující, vzhledem k hierarchickému uspořádání kategorií – každé přiřazení v listu stromu znamená i přiřazení všech jeho předků až ke kořenu stromu. Jsou ale důležité, vzhledem k tomu, že při trénování klasifikátoru nebude na hierarchii brán zřetel a všechny kategorie budou považovány za rovnocenné. Tzn. i přes tento podstatný nedostatek nebudou kategorie RCV1-v2a nijak omezeny a pro trénování (a testování) bude použito všech 101.

Pro trénování neuronové sítě (a obecně většiny učících se klasifikačních metod) je však několik desítek exemplářů v jedné kategorii obvykle nedostatečný počet. Lze očekávat, že klasifikátor bude relativně dobře odhadovat kategorie ve vyšších vrstvách hierarchie, ale s málo četnými listovými kategoriemi si tak dobře neporadí. Pokud se tato domněnka prokáže, mohlo by to být dobře vidět mj. na nižším makro-průměrovaném F1 skóre (v porovnání s mikro-průměrovaným), resp. na všech makro-průměrovaných evaluačních metrikách.

6 Hierarchická reprezentace

Za hierarchickou reprezentaci se označuje taková reprezentace, kde je využita hierarchická struktura částí dokumentu – tzn. dokument se chápe jako složený z několika dílů, které také mohou být dále děleny. Smyslem je zachytit pomocí distribuovaných reprezentací sémantický význam plynoucí z hierarchie – jednotlivé věty či odstavce mohou mít mírně odlišný význam (v kontextu příslušenství ke kategoriím) a je možné, že reprezentace na nich založená ponese kvalitnější informaci pro klasifikaci dokumentů.

Použité datové sady bohužel nejsou na hierarchickou reprezentaci nejvhodnější, protože dokumenty nejsou nikterak strukturovány. Jediná hierarchie plyne přímo ze syntaxe přirozeného jazyka – text je rozdělen na věty a věty na slova. Tento fakt omezuje možnosti hierarchické reprezentace. Dále reprezentování dokumentu po slovech (ať už průměr slov nebo matice, kde každá řádka odpovídá jednomu slovu), což není neobvyklá praktika, nelze považovat za hierarchickou reprezentaci jako takovou. Za použití konvoluční neuronové sítě s *embedding* vrstvou byla reprezentace po slovech použita např. v [32] pro klasifikaci sentimentu nebo v [40] pro kategorizaci dokumentů na českém korpusu, který je použit také v této práci.

6.1 Související práce

Hierarchická struktura dokumentů je využita pro reprezentaci např. v [74], kde autoři klasifikují vědecké články, které jsou díky použití stejné šablony hierarchicky strukturované *document* > *section* > *subsection* > *paragraph* > *subparagraph*. Reprezentace dokumentu je zde rekurzivně vytvořena jako součet (po složkách) vektoru distribuované reprezentace, získaného z *doc2vec*, a jednotlivých vektorů z matice, která je výsledkem aplikace SVD na matici vektorových reprezentací potomků.

Také existuje přístup, kdy je pro zachycení přirozené hierarchie v dokumentu (věty a slova) navržena speciální architektura rekurentní neuronové sítě – tzv. *Hierarchical attention network* [83] (HAN). V této architektuře jsou nejprve slova jednotlivých vět transformována v *dense* reprezentace a poté jsou z nich vybrána důležitá slova, ze kterých bude vytvořena reprezentace věty. Další vrstvy HAN provádí to samé pro věty – vytvoření reprezentace vět z vektorů slov a poté výběr důležitých vět, ze kterých je vytvořena reprezen-

tace dokumentu, která je pak použita jako příznakový vektor pro klasifikaci. Další práce využívající tento přístup (pro klasifikaci sentimentu), tj. přirozená hierarchie dokumentu je zahrnuta přímo v architektuře neuronové sítě (autoři zde experimentují kromě RNN také s CNN), je např. [87].

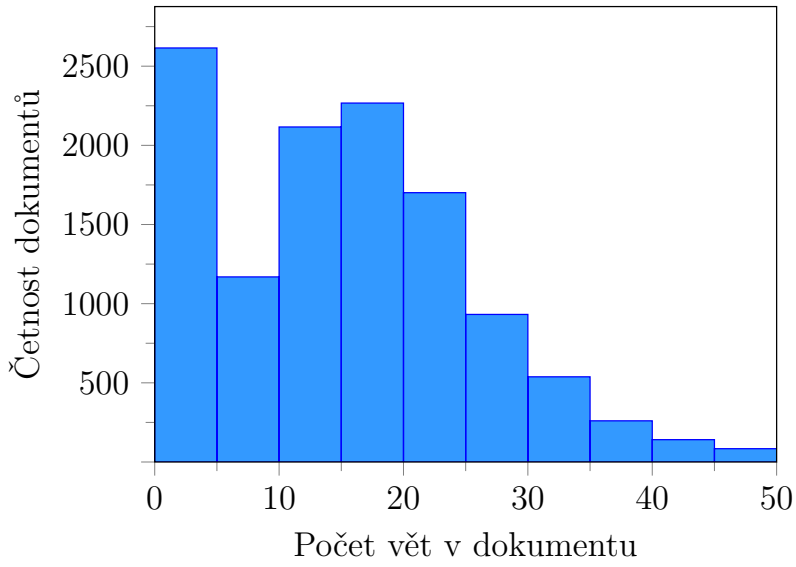
6.2 Navržená metoda reprezentace

Kvůli chybějící (hierarchické) struktuře dokumentů v dostupných datových sadách zůstává pouze jediná možnost, jak dokumenty reprezentovat hierarchicky – po větách. Vyzkoušeny budou následující dva způsoby.

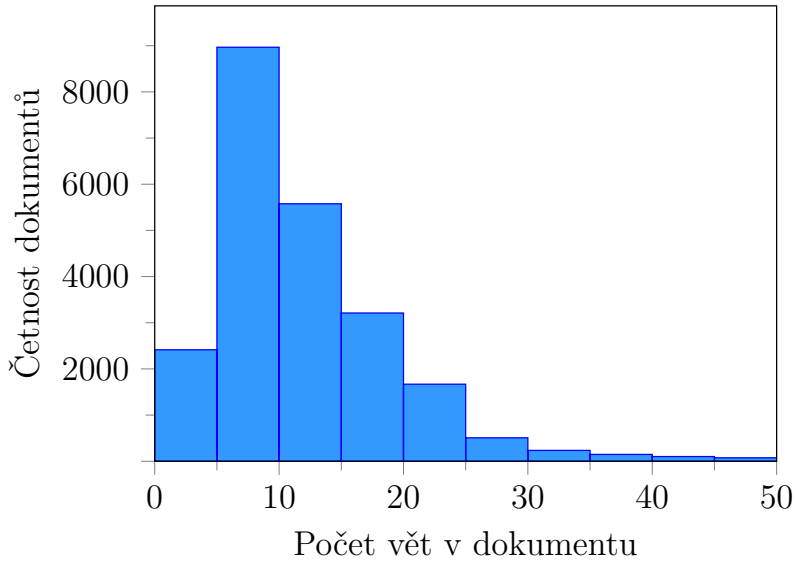
- Dokument bude složen z distribuovaných reprezentací vět, které budou získány z *doc2vec* nebo *Sent2vec* – tzn. dokument bude reprezentován maticí o rozměrech $N \times L$, kde N je délka vektorů, reprezentujících jednotlivé věty, a L počet vět, které budou z každého dokumentu použity. Pokud bude mít dokument méně než L vět, bude matice doplněna nulami. Tato matice bude použita jako vstup CNN.
- Druhý způsob je obměnou prvního – reprezentace jednotlivých vět budou získány jako průměr distribuovaných reprezentací slov (získaných např. z *word2vec* nebo *fastText*).

Důležitý může být také počet vět, které budou z každého dokumentu použity. Na obrázcích 6.1 a 6.2 lze vidět histogramy počtu vět v dokumentech pro český a anglický korpus. V obou histogramech byly vynechány dokumenty, které mají více než 50 vět – u české datové sady se jedná o 132 dokumentů, u anglické o 242 dokumentů.

Průměrně (včetně dokumentů vynechaných z histogramu) má dokument české datové sady $15,6$ vět, medián je 15. Vhodný počet vět, kterým reprezentovat dokument, by mohl být buď medián či průměr, nebo takový počet, aby pro většinu dokumentů nebylo ztraceno velké množství vět. Takový počet by mohl být např. 30, protože 10922 ($91,4\%$) dokumentů z trénovací množiny má 30 vět nebo méně a tudíž by neztratilo žádnou větu. V anglickém korpusu je průměrný (včetně dokumentů vynechaných z histogramu) počet vět v dokumentu $12,2$, medián 10. Při reprezentaci 20 větami by žádnou větu neztratilo 20640 dokumentů z trénovací množiny, tj. $89,2\%$. V kapitole 8 bude proveden experiment, jaký počet vět je nejvhodnější.



Obrázek 6.1: Histogram počtu vět v dokumentech v trénovací množině českého korpusu.



Obrázek 6.2: Histogram počtu vět v dokumentech trénovací množiny anglického korpusu.

7 Implementace

Práce byla implementována v jazyce Python – především proto, že pro tento jazyk existují hojně používané knihovny s implementacemi neuronových sítí a distribuovaných reprezentací. Konkrétně byla pro generování distribuovaných reprezentací použita knihovna Gensim [59] a pro implementaci neuronových sítí Keras [14].

Z knihovny Gensim byly použity implementace *word2vec* a *doc2vec*. Knihovna umožňuje trénovat také *fastText*, ale z nezjištěného důvodu běželo trénování na testovacím počítači nepřijatelně dlouho dobu (v řádu hodin až dnů pro český korpus). Proto byla pro generování *fastText* vektorů externě použita původní implementace *fastText* algoritmu v jazyce C++[44,45], která běží mnohem rychleji (v řádu maximálně desítek minut pro větší počet iterací) – výsledné slovní vektory lze potom pomocí knihovny Gensim importovat. Pro natrénování *Sent2vec* modelu byla také použita původní implementace v jazyce C++[46,47] – Gensim poskytuje wrapper nad C++ implementací a lze tedy natrénovaný model načíst[48].

Keras poskytuje vysoko-úrovňové API, které abstrahuje nad některými známými knihovnami pro implementaci neuronových sítí (momentálně existuje podpora pro Theano, TensorFlow a CNTK – *Computational Network Toolkit*) – použita byla knihovna TensorFlow [1]. Výpočet lze provádět na procesoru nebo na grafické kartě.

7.1 Struktura programu

Program je implementován jako knihovna (tj. není sám o sobě spustitelný), která obsahuje třídy a funkce pro předzpracování dokumentů, načtení před-

[44] Ke stažení na `https://github.com/facebookresearch/fastText/releases/tag/v0.1.0`

[45] S drobnou úpravou zdrojového kódu – před kompilací bylo zakomentováno uložení celého modelu, který na disku zabírá až několik gigabajtů. Ukládány jsou pouze natrénované vektory slov.

[46] Ke stažení na `https://github.com/epfml/sent2vec`

[47] S mírnou úpravou zdrojového kódu, aby byl program zkompilovatelný na Windows.

[48] *Sent2vec* ovšem musí nejprve být nainstalovaný (viz příručka F.1) v úložišti Python balíků. Tato instalace není nutná, aby byla aplikace spustitelná (ale v tomto případě samozřejmě nelze *sent2vec* model importovat).

zpracovaných dokumentů, vytváření jejich číselných reprezentací, trénování různých modelů neuronových sítí a konečně výpočet některých metrik úspěšnosti klasifikace (z kapitoly 2.2) na validačních datech. Program je členěn do 4 balíků – `input`, `network`, `tools` a `utils`.

Je na uživateli, aby vytvořil krátký skript, který za pomoci implementované knihovny načte data, natrénuje síť a vypočte a uloží výsledky. Příloha F.2 obsahuje návod, jak by tento skript měl vypadat a jak knihovnu používat. Na přiloženém DVD se dále nachází vzorové spustitelné skripty.

7.1.1 `tools`

Balík obsahuje dva soubory `prepare_czech.py` a `prepare_english.py`, které obsahují funkce pro předzpracování českého a anglického korpusu. Pro oba korpusy je každý dokument rozdělen na věty a slova a jsou odstraněna stop slova, interpunkce a čísla. Pro anglický korpus jsou navíc doplněny chybějící kategorie z hierarchie kategorií a odstraněny dokumenty, kterým chybí kategorie (viz kapitola 5.2).

7.1.2 `input`

Balík `input` obsahuje třídy a funkce pro zpracování vstupu.

- Soubor `document_source.py` – tento soubor obsahuje třídy, které načítají předzpracované dokumenty a ve vhodném formátu je poskytují ostatním částem knihovny.
- Soubor `distributed_representation.py` – obsahuje třídy, které pomocí knihovny Gensim trénují distribuované reprezentace (včetně navržené hierarchické reprezentace) a načítají předdtrénované slovní vektory nebo modely.
- Soubor `input_provider.py` – zde se nachází třídy, jejichž úkolem je transformace načtených dokumentů na číselné reprezentace, kterým bude rozumět neuronová síť.
- Soubor `input_provider_factory.py` – *factory* funkce pro vytváření tříd z `input_provider.py` podle zadaných parametrů.

7.1.3 `network`

Zde se provádí vytvoření modelu neuronové sítě knihovnou Keras (soubor `model.py`) a poté trénování tohoto modelu a jeho evaluace (soubor `nn.py`).

7.1.4 `utils`

Obsahuje různé pomocné funkce, které jsou využívány celým programem, případně uživatelem.

- `string_utils.py` – funkce pro zacházení s textovým řetězcem; konverze na číslo apod.
- `nn_utils.py` – obsahuje funkce pro výpočet metrik úspěšnosti klasifikace.
- `docs_utils.py` – zde jsou k nalezení funkce pro zacházení s načteným dokumentem (např. vytvoření generátoru poskytujícího souvislý text ze seznamu dokumentů).
- `print_utils.py` – funkce pro zpracování výsledků metrik a jejich tisk do souboru nebo konzole.

8 Testování

Tato kapitola popisuje provedené experimenty a jejich výsledky. Nejprve jsou provedeny předběžné experimenty např. pro výběr vhodného algoritmu generování distribuovaných reprezentací slov a podobně, poté finální experimenty pro porovnání *baseline* s hierarchickými metodami reprezentace.

Předběžné experimenty často nejsou provozovány na celé testovací množině nebo jsou provozovány bez ideálního nastavení (např. testování na validační množině nebo s méně iteracemi trénování) především kvůli časové náročnosti – toto si lze dovolit, protože v rámci jednoho experimentu (tzn. konstantní konfigurace) jde pouze o porovnání jednotlivých přístupů mezi sebou, ne o porovnání s výsledky jiných experimentů, případně s externími výsledky. Výsledky kteréhokoliv z těchto experimentů tedy **nejsou** porovnatelné s výsledky jiných experimentů.

Některé přístupy v jednotlivých experimentech budou vybrány jako *baseline* pro porovnání s ostatními experimenty, resp. pro provedení finálních experimentů – především otestování navržené hierarchické metody reprezentace dokumentů a její porovnání s *baseline*. Experimenty pro vybrané metody reprezentace budou znovu provedeny s jednotnou konfigurací, aby byly porovnatelné mezi sebou, s hierarchickými reprezentaci a případně s výsledky jiných prací.

8.1 Konfigurace experimentů

Tato sekce popisuje konfiguraci, která byla použita při experimentovaní. Jsou zde uvedena ta nastavení, která jsou společná většímu počtu experimentů (případně všem). Nejprve je uvedena konfigurace společná předběžným i finálním experimentům, poté ta, která se liší. Další případná specifická konfigurace bude uvedena u jednotlivých experimentů.

8.1.1 Společná konfigurace

Všechny experimenty byly prováděny pro anglický korpus pro 101 kategorií, přítomných v trénovací množině, a pro český korpus s 37 nejčetnějšími kategoriemi. Vhodný práh pro *multi-label* klasifikaci byl zjištěn experimentálně pro maximalizaci mikro F-skóre – $0,4$ pro všechny experimenty s výjimkou

klasifikace pomocí MLP nad BoW reprezentací, kde je $0,5$, a klasifikace pomocí CNN s trénovatelnou *embedding* vrstvou, kde je nejvhodnější práh $0,3$.

Některé použité metody reprezentace vyžadují omezení velikosti slovníku – jmenovitě BoW a *embedding* vrstva. Slovník je v těchto případech vždy omezen na 20 tisíc nejčetnějších slov. Dále, pokud nebude řečeno jinak, jsou délky vektorů distribuovaných reprezentací (*word2vec*, *doc2vec* atd.) vždy 150. U metod, kde je nutné omezit počet slov (např. pro maticový vstup CNN), bude použito (pokud nebude řečeno jinak) vždy 150 prvních slov z každého dokumentu. Podle [40] může sice úspěšnosti klasifikace prospět větší počet slov, ale bohužel počítač, na kterém byly experimenty spouštěny, nemá dostatečně kvalitní hardware, aby zvládl výpočet CNN pro více slov (v kombinaci s délkou vektorů – je sice možné použít kratší vektory a více slov, nebo naopak, ale hodnoty 150 a 150 byly zvoleny jako vhodný kompromis).

Všechny testy byly prováděny na počítači s 64 bitovým Windows 10, 64 bitovým Pythonem verze 3.6.3, procesorem Intel i5-7500 3.4GHz, 8GB RAM a grafickou kartou NVIDIA GeForce GTX 1060 s 3GB paměti. Na jiné konfiguraci (především výrazně méně RAM nebo grafické paměti) nemusí být provedené experimenty zopakovatelné.

Architektura neuronové sítě

Pro všechny provedené experimenty v této práci byly použity stejné architektury MLP a CNN jako v [40]. MLP má dvě skryté vrstvy o 1024 a 512 neuronech a počet neuronů vstupní vrstvy se rovná délce vektoru, kterým je reprezentován dokument. Výstupní vrstva má tolik neuronů, do kolika kategorií se klasifikuje (37 pro český korpus, 101 pro anglický).

Použitá CNN má jednu sekvenci konvoluční a *pooling* vrstvy, kde konvoluční vrstva používá 40 filtrů velikosti $min(L, 16) \times 1$, kde L je počet vektorů (slov nebo vět), kterými je reprezentován dokument, tzn. konvoluce probíhá na jednom příznaku 16 vektorů (případně je použit menší filtr, pokud je dokument reprezentován menším počtem vektorů). *Pooling* vrstva provádí *max pooling* filtrem velikosti $(L - min(L, 16)) \times 1$ (nebo 1×1, pokud náhodou $L = 16$), což znamená, že je *max pooling* proveden vždy na jednom příznaku všech řádek mapy příznaků z předchozí konvoluční vrstvy. Výsledkem je pak 40 vektorů délky N, což je šířka matice, kterou byl reprezentován dokument (resp. délka vektorů slov nebo vět, ze kterých je dokument složen). Násle-

duje jedna plně propojená vrstva, která má 256 neuronů. Výstupní vrstva je stejná jako u MLP. Vstupem CNN je buď sekvence L slovníkových indexů v případě použití *embedding* vrstvy, nebo matice velikosti $N \times L$ složená z L vektorových reprezentací slov nebo vět délky N, pokud *embedding* vrstva není použita.

Jako aktivace konvolučních a plně propojených vrstev byla použita funkce ReLU, výstupní vrstvy mají za aktivaci funkci sigmoid.

8.1.2 Konfigurace předběžných experimentů

Trénování pro předběžné experimenty pro český korpus bylo prováděno na trénovací množině a testování na validační (viz kapitolu 5.1.2) a pro anglický korpus trénování na trénovací množině a testování na prvních 10 až 100 tisících dokumentech z testovací množiny (konkrétní počet se liší pro různé experimenty). Pro všechny tyto experimenty byla neuronová síť trénována 5 iteracemi.

8.1.3 Konfigurace finálních experimentů

Finální experimenty pro porovnání jednotlivých reprezentací (včetně hierarchické) byly pro český korpus po vzoru [40] prováděny křížovou validací na trénovací množině s $k = 5$, tzn. trénovací množina byla rozdělena na 5 dílů, kdy byl opakovaně jeden díl (vždy jiný) odložen pro otestování a na 4 pětinách natrénována neuronová síť. Celkový výsledek je pak průměr z těchto 5 běhů. Pro anglický korpus bude pro trénování a testování použita trénovací a testovací množina beze změny[49] (viz kapitolu 5.2.1). Každý výsledek, uvedený pro tyto experimenty, je průměrem alespoň z 5 samostatných spuštění. Počet trénovacích epoch neuronových sítí byl 20 s výjimkou klasifikace MLP nad BoW reprezentací, kde je použito 5 epoch, a CNN s *embedding* vrstvou, kde je použito 10 epoch.

8.2 Předběžné experimenty

Tato kapitola uvádí předběžné experimenty, které byly provedeny pro porovnání různých příbuzných přístupů pro tvorbu reprezentací nebo např.

[49]Pro anglický korpus nebyla použita žádná validační množina, protože všechny metody byly laděny na validační množině českého korpusu. Na anglickém korpusu byly spuštěny pouze pro získání výsledků.

pro zjištění vhodného počtu vět při hierarchické reprezentaci. Cílem je mj. vybrat vhodný *baseline* pro porovnání s hierarchickými metodami.

8.2.1 Příznaky BoW reprezentace

Tento experiment má za cíl zjistit nejlepší způsob, jak získat či vypočítat jednotlivé příznaky vektoru při BoW reprezentaci – viz kapitola 4.2. Výsledky pro český korpus lze vidět v tabulce 8.1. Jako nejvhodnější způsob, přestože s malým rozdílem, se jeví absolutní frekvence, tedy příznak je určen počtem výskytů slova v dokumentu. Tato reprezentace dokumentu při klasifikaci pomocí MLP bude použita jako *baseline*.

Výpočet příznaku	binární	frekvence (absolutní)	frekvence (relativní)	tf-idf
$\mathbf{F}_\mu$ [%]	78,73	**78,98**	78,19	75,85
$\mathbf{F}_\mathbf{M}$ [%]	71,99	**73,16**	71,92	69,24

Tabulka 8.1: Porovnání výpočtu příznaků při BoW reprezentaci (na validační množině českého korpusu).

8.2.2 Distribuované reprezentace slov

Cílem tohoto experimentu je porovnat různé algoritmy pro generování distribuovaných reprezentací slov. Všechny vektory, použité v tomto experimentu, mají délku 300 a pro reprezentaci dokumentu pro CNN bylo použito 100 slov.

Předtrénované vektory

Zde byly otestovány předtrénované vektory stažené z internetu. V následujícím seznamu jsou uvedeny zdroje předtrénovaných vektorů. Metoda GLoVe sice funguje jinak, než ostatní uvedené (založena na *co-occurence* matici, ne na předpovídání pomocí modelu neuronové sítě), ale je vhodné ji vyzkoušet, protože se také poměrně hojně používá.

- *word2vec (skip-gram)* – Natrénováno na anglických GoogleNews, staženo z `https://code.google.com/archive/p/word2vec/`.
- *fastText (skip-gram)* – Natrénováno na anglické Wikipedii, staženo z `https://fasttext.cc/docs/en/pretrained-vectors.html`.
- *GLoVe* – Natrénováno na anglické Wikipedii a korpusu Gigaword[50], staženo z`https://nlp.stanford.edu/projects/glove/`.

[50] `https://catalog.ldc.upenn.edu/LDC2011T07`

Použity byly pouze vektory pro 20 tisíc nejčastějších slov (ostatní slova jsou ignorována). Také byly předtrénované vektory porovnány s vektory natrénovanými pomocí *word2vec*. V tabulce 8.2 lze vidět výsledky z otestování na prvních 20 tisících dokumentech testovací množiny anglického korpusu. Z předtrénovaných vektorů nejlépe vychází *word2vec*. Trénované vektory jsou mírně horší než předtrénované, což je pravděpodobně způsobeno tím, že předtrénované vektory byly trénovány na neporovnatelně větším množství dat (řádově miliardy slov) a tím pádem mají kvalitnější reprezentaci především pro méně frekventovaná slova. Z toho vyplývá, že použití předtrénovaných vektorů má smysl i při úloze klasifikace dokumentů do velkého množství kategorií. Předtrénované vektory ale dále využívány nebudou ze dvou důvodů. Za prvé při délce 300 většinou počítač, kde testování probíhalo, nezvládne výpočet CNN. Za druhé pro češtinu nejsou *word2vec* vektory dostupné ke stažení.

Metoda	*word2vec (trained)*	*word2vec*	*fastText*	*GLoVe*
$\mathbf{F}_\mu$ [%]	79,18	**79,98**	78,64	79,63
$\mathbf{F}_\mathbf{M}$ [%]	48,01	**49,56**	45,43	48,06

Tabulka 8.2: Porovnání předtrénovaných dense *reprezentací slov (na prvních 20 tisících dokumentech testovací množiny anglického korpusu).*

Trénované vektory

Porovnány byly také trénované vektory – natrénované na trénovací množině českého korpusu. *GLoVe* se nepovedlo zprovoznit, a proto zde byly vyzkoušeny pouze *word2vec* a *fastText* – pro oba modely CBOW a SG. Pro trénovaní obou metod a obou modelů bylo použito 20 iterací. V tabulce 8.3 lze najít výsledky.

Metoda	*word2vec* (CBOW)	*word2vec* (SG)	*fastText* (CBOW)	*fastText* (SG)
$\mathbf{F}_\mu$ [%]	70,77	**72,97**	71,37	71,88
$\mathbf{F}_\mathbf{M}$ [%]	59,64	**64,08**	63,03	61,92

Tabulka 8.3: Porovnání trénovaných distribuovaných reprezentací slov (na validační množině českého korpusu).

Nejlépe vychází *word2vec* (SG), pročež bude používán pro generování distribuovaných reprezentací slov v ostatních experimentech. Také bude klasifikace pomocí CNN, kde je dokument reprezentován jako matice, složená po řádkách ze slov, použita jako *baseline*.

8.2.3 Distribuované reprezentace vět

Při tomto experimentu byly vyzkoušeny různé přístupy pro generování distribuovaných reprezentací dokumentů, které potom slouží jako příznakový vektor pro klasifikaci pomocí MLP.

Předtrénované modely

Cílem tohoto experimentu je vyzkoušet, jestli má smysl používat předtrénované modely. V tabulce 8.4 lze vidět porovnání dvou předtrénovaných *doc2vec* modelů (*wiki* a *apnews*) a jednoho natrénovaného na trénovací množině anglického korpusu (*train*) – délka vektoru byla 300. Otestovány byly na prvních 20 tisících dokumentech z testovací množiny. Z tabulky je patrné, že nejlépe vychází trénovaný model a je tedy lepší jej použít místo předtrénovaných modelů (pokud je pominut fakt, že při použití předtrénovaného modelu se šetří čas).

Model	*wiki* (DBOW)	*apnews* (DBOW)	*train* (DBOW)
$\mathbf{F}_{\mu}$ [%]	80,02	80,05	**80,88**
$\mathbf{F}_{\mathbf{M}}$ [%]	50,04	50,53	**51,92**

Tabulka 8.4: Porovnání předtrénovaných modelů, generujících distribuované reprezentace dokumentů (na prvních 20 tisících dokumentech testovací množiny anglického korpusu).

Trénované modely

Tento experiment porovnává dva algoritmy pro vytváření distribuovaných reprezentací dokumentů (*Sent2vec* a *doc2vec* v obou variantách DBOW a DM) a metodu, kdy je reprezentace dokumentu vytvořena zprůměrováním 150 slovních vektorů z *word2vec*. V tabulce 8.5 lze vidět výsledky pro český korpus a délku vektorů 100.

Algoritmus	*Sent2vec*	*doc2vec* (DM)	*doc2vec* (DBOW)	*word2vec* (průměr)
$\mathbf{F}_{\mu}$ [%]	75,88	62,48	**76,73**	75,04
$\mathbf{F}_{\mathbf{M}}$ [%]	67,97	47,66	**70,19**	67,09

Tabulka 8.5: Porovnání algoritmů pro vytváření distribuovaných reprezentací dokumentů (na validační množině českého korpusu).

Jako nejlepší vychází *doc2vec* ve variantě DBOW a bude při klasifikaci pomocí MLP použit jako *baseline*. Zajímavé je, že autoři *doc2vec* uvádějí jako

lepší variantu DM [39]. Další výzkumy jsou ale konzistentní s tímto experimentem [21; 38], tedy uvádějí, že DBOW předčí DM ve většině úloh. Zajímavým poznatkem je také to, že průměr *word2vec* vektorů zaostává poměrně málo, F_μ o $1,69\%$.

8.2.4 *Embedding* vrstva CNN

V tomto experimentu jsou porovnány různé strategie inicializace a trénování *embedding* vrstvy konvoluční neuronové sítě. Porovnáno bude následující:

- *random* – *embedding* vrstva CNN je inicializována náhodně a trénuje se jako každá jiná vrstva neuronové sítě.
- *insert-trainable* – do *embedding* vrstvy budou vloženy natrénované slovní vektory z *word2vec* a vrstva **bude** dále trénovatelná.
- *insert-static* – do *embedding* vrstvy budou vloženy natrénované slovní vektory, ale vrstva **nebude** dále trénovatelná.
- *cnn* – v CNN nebude použita *embedding* vrstva a na vstup budou vkládány dokumenty jako matice *word2vec* vektorů – tento přístup je ekvivalentní s *insert-static* a proto by měly být výsledky těchto dvou přístupů velice podobné.

V tabulce 8.6 lze najít výsledky tohoto experimentu pro český i anglický korpus (test na prvních 100 tisících dokumentech). Z těchto výsledků lze vypozorovat, že se potvrdil předpoklad, že *cnn* a *insert-static* mají podobné výsledky. Také je zřejmé, že *random* a *insert-trainable* jsou srovnatelné, avšak s tím rozdílem, že *random* má o něco lepší F_M. Oba tyto přístupy ale výrazně předčily *insert-static*, resp. *cnn* – je tedy bezpochyby lepší nechat neuronovou síť trénovat náhodně inicializovanou *embedding* vrstvu, než používat či vkládat natrénované slovní vektory z *word2vec*, a to i v případě, že by byla *embedding* vrstva s vloženými vektory dále trénovatelná. Natrénované slovní vektory zde nepřináší žádnou informaci, která by pomohla zvýšit úspěšnost klasifikace. Pravděpodobné je, že *embedding* vrstva vytvoří reprezentace slov šité na míru konkrétní úloze, kdežto předem natrénované vektory jsou více obecné.

Zajímavé je, že autor původních pokusů aplikace CNN na klasifikaci textu uvádí opačné výsledky [32] - náhodná inicializace nejhorší ($76,1\%$ na vybrané úloze klasifikace sentimentu filmových recenzí) a vložení předtrénovaných vektorů do *embedding* vrstvy výrazně lepší ($81,1\%$ pro *insert-static*

Strategie	*random*	*insert -trainable*	*insert -static*	*cnn*
Český korpus				
$\mathbf{F}_\mu$ [%]	**76,98**	76,52	72,49	72,67
$\mathbf{F}_\mathbf{M}$ [%]	**71,15**	68,45	62,34	62,99
Anglický korpus				
$\mathbf{F}_\mu$ [%]	**81,70**	81,57	78,28	78,99
$\mathbf{F}_\mathbf{M}$ [%]	**55,78**	54,52	43,21	42,67

Tabulka 8.6: Porovnání strategií inicializace a trénování embedding vrstvy (na validační množině českého korpusu a prvních 20 tisících dokumentech testovací množiny anglického korpusu).

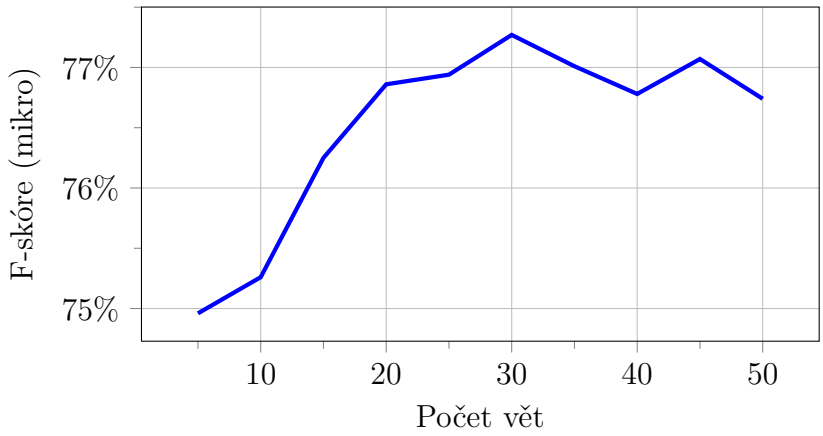
a $81,5\%$ pro *insert-trainable*). Výsledky v tabulce 8.6 jsou ovšem konzistentní s experimenty v [41], provedenými na českém korpusu, který je použit také v této práci – zde autoři také uvádějí jako nejlepší *random* ($84,32\%$), poté *insert-trainable* ($83,81\%$) a jako nejhorší *insert-static* ($73,98\%$). Tento rozdíl oproti [32] může plynout z odlišnosti úloh (klasifikace sentimentu vs. klasifikace článků do mnoha kategorií) – je možné, že pro některé úlohy jsou předtrénované vektory vhodné, ale pro jiné úlohy nikoliv. Také může mít velký vliv odlišná architektura neuronové sítě – v [41] je vyzkoušena i stejná architektura jako v [32] a rozdíly mezi náhodnou inicializací a vloženými vektory se liší výrazně méně ($80,21\%$ vs. $79,36\%$), avšak náhodná inicializace stále dosahuje mírně lepších výsledků.

Klasifikace pomocí CNN s náhodně inicializovanou *embedding* vrstvou bude použita jako *baseline*.

8.2.5 Počet vět při hierarchické reprezentaci

Cílem tohoto experimentu bylo vhodný počet vhodný vět, kterými při hierarchické reprezentaci reprezentovat dokument. Dobrými kandidáty může být průměr či medián, nebo takový počet, aby většina dokumentů nepřišla o žádnou větu. V histogramu lze vidět závislost úspěšnosti klasifikace (F_μ) na počtu použitých vět pro český korpus. Věty zde byly vytvořeny jako průměr *word2vec* vektorů.

Z grafu je zřejmé, že je vhodné volit spíše větší počet vět. Do 30 vět úspěšnost klasifikace stabilně stoupá, poté začíná kolísat. Proto bude pro hierarchickou reprezentaci českých dokumentů zvolen počet vět **30**. Pro anglické dokumenty bude použito **20** vět, což pro trénovací množinu anglického kor-



Obrázek 8.7: Závislost úspěšnosti klasifikace na maximálním počtu vět, ze kterých je vytvořena reprezentace dokumentu (na validační množině českého korpusu).

pusu přibližně odpovídá stejnému procentu dokumentů (jako při použití 30 vět v českém korpusu[51]), které nepřijdou o žádnou větu.

8.2.6 Rekapitulace

Z provedených experimentů vyplynulo (pro tuto úlohu klasifikace a použité datové sady a architektury neuronových sítí) několik poznatků.

- Nejlepší reprezentací slov z vyzkoušených reprezentací je *word2vec* ve variantě *skip-gram*. Dále je zřejmé, že má smysl (alespoň pro některé úlohy NLP) používat předtrénované *word2vec* vektory, které dosahují v provedeném experimentu mírně lepších výsledků, než trénované vektory.

- Nejlepší reprezentací dokumentů z vyzkoušených reprezentací je *doc2vec* ve variantě DBOW a není nutné použít předtrénované modely pro zvýšení úspěšnosti klasifikace (nicméně jsou srovnatelné s modelem natrénovaným přímo na datové sadě, na které bude provozováno trénování ANN).

- Při použití *embedding* vrstvy je lepší náhodná inicializace, než vložení natrénovaných slovních vektorů.

- Nejvhodnější počet vět pro hierarchickou reprezentaci je takový, že většina dokumentů nepřijde o žádnou větu (a tedy informaci) – pro český korpus je to 30 vět, pro anglický bylo zvoleno 20 vět.

[51] Pro český korpus a 30 vět je to 91, 4%, pro anglický a 20 vět 89, 2%.

Některé z provedených experimentů byly vybrány pro použití jako *baseline* pro porovnání s hierarchickými reprezentacemi. Tučný text značí pojmenování *baseline*. Pod tímto jménem se na něj bude dále odkazovat. Vybrané *baseline* budou pochopitelně znovu natrénovány podle popisu v kapitole 8.1.3, aby byly porovnatelné.

- **mlp-bow** – MLP + BOW reprezentace, kde každý příznak BOW vektoru je roven počtu výskytů příslušného slova v dokumentu.
- **cnn-word** – CNN + *word2vec*, kde každá řádka matice, která reprezentuje dokument, je vektor slova z *word2vec*.
- **mlp-doc** – MLP + *doc2vec*, kde je dokument reprezentován jedním vektorem, získaným z *doc2vec*.
- **cnn-emb** – CNN + *embedding* vrstva (náhodná inicializace) – dokument je reprezentován sekvencí slovníkových indexů pro vyhledávací tabulku *embedding* vrstvy.

8.3 Finální experimenty

V kapitolách 8.3.1 (český korpus) a 8.3.2 (anglický korpus) se nachází tabulky s výsledky porovnání opětovně natrénovaných *baseline* a hierarchických metod reprezentace s jednotnou konfigurací trénování a testování. Také je zde přítomno porovnání s dostupnými výsledky jiných prací.

8.3.1 Česká datová sada

V tabulce 8.8 jsou uvedeny výsledky (procentuální hodnota mikro a makro *precision*, *recall* a *F-measure* a *mutli-label accuracy*) porovnání opětovně natrénovaných *baseline* s hierarchickými metodami reprezentace pro český korpus. **Hier-doc** značí hierarchickou reprezentaci, kdy jsou reprezentace jednotlivých věty vytvořené pomocí *doc2vec* a **hier-avg** značí hierarchickou reprezentaci, kdy jsou reprezentace vět vytvořené jako průměr *word2vec* (*skip-gram*) vektorů. Horizontální čarou jsou odděleny nedistribuované reprezentace od distribuovaných. Tučně je vyznačena nejvyšší hodnota v každém sloupci, přičemž nejdůležitější je sloupec $\mathbf{F}_\mu$, případně $\mathbf{F_M}$ a $\mathbf{A_{ML}}$.

Pro český korpus konzistentně vychází distribuované metody reprezentace (ať už hierarchické, či nikoliv) hůře než BOW a náhodně inicializovaná *embedding* vrstva. Je možné, že je to způsobeno nedostatečným počtem dokumentů, na kterých je *word2vec* či *doc2vec* trénován. Problémem při malém

[%]	$\mathbf{P}_\mu$	$\mathbf{R}_\mu$	$\mathbf{F}_\mu$	$\mathbf{P_M}$	$\mathbf{R_M}$	$\mathbf{F_M}$	$\mathbf{A_{ML}}$
mlp-bow	85,64	**82,23**	**83,90**	86,84	**81,01**	**83,55**	**78,75**
cnn-emb	**86,17**	79,81	82,85	**87,20**	78,39	82,19	77,04
mlp-doc	82,11	78,40	80,21	83,81	76,01	79,31	73,35
cnn-word	81,15	73,62	77,37	83,41	71,20	76,08	70,43
hier-doc	79,81	72,19	75,80	81,73	69,36	74,18	68,52
hier-word	81,96	78,34	80,10	83,69	76,65	79,53	74,01

Tabulka 8.8: Výsledky pro český korpus (změřeno křížovou validací na trénování množině).

počtu dokumentů se může stát např. velká míra flexe v českém jazyce. Jednotlivé tvary jednoho slova se nevyskytují tolikrát, aby pro ně byla natrénována dostatečně kvalitní distribuovaná reprezentace. S tímto problémem by mohla pomoci lemmatizace nebo stematizace, případně použití předtrénovaných vektorů, které byly trénovány na podstatně větším korpusu. Avšak předtrénované *word2vec* vektory bohužel momentálně nejsou volně dostupné ke stažení.

Pokud by byly v tabulce 8.8 ponechány pouze distribuované reprezentace, tj. **mlp-bow** a **cnn-emb** by byly zanedbány, vychází nejlepší hierarchická metoda (**hier-word**) srovnatelně s nejlepší distribuovanou nehierarchickou metodou reprezentace (**mlp-doc**) – $\mathbf{F}_\mu$ se liší o $0,11$ % ve prospěch nehierarchické a $\mathbf{F_M}$ o $0,22$ % ve prospěch hierarchické. Hierarchické metody tedy konkurují nehierarchickým, ale u použitého českého korpusu je pravděpodobně nejprve nutné zabývat se problémy nastíněnými v předchozím odstavci.

Zajímavým úkazem je podstatně horší ($\Delta\mathbf{F}_\mu = 4,30$ %) výsledek **hier-doc** oproti **hier-word**. Toto je v kontrastu s experimentem v kapitole 8.2.2, kde pro klasifikaci pomocí MLP vycházela reprezentace dokumentu jako průměr *word2vec* vektorů všech slov o $0,8$ % ($\mathbf{F}_\mu$) hůře, než reprezentace dokumentu jedním *doc2vec* vektorem. Z toho lze vyvodit, že *doc2vec* je v porovnání s průměrem *word2vec* vektorů méně kvalitní reprezentací kratších kusů textu, v tomto případě vět, ale pro delší kusy textu (celé dokumenty) už to tak není.

V tabulce 8.9 lze nalézt porovnání vybraných řádek z tabulky 8.8 s výsledky jiných prací, kde je použita stejná metoda testování pro získání výsledků

(křížová validace na trénovací množině s $k = 5$) jako v této práci. Čarou jsou odděleny externí výsledky.

[%]	$\mathbf{P}_\mu$	$\mathbf{R}_\mu$	$\mathbf{F}_\mu$
mlp-bow	85,64	82,23	83,90
cnn-emb	86,17	79,81	82,85
hier-word	81,96	78,34	80,10
Brychcín & Král 2014 [11] (ME)	89,0	75,6	81,7
Lenc & Král 2017 [40] (MLP)	83,7	83,6	83,9
Lenc & Král 2017 [40] (CNN-best)	86,4	82,8	84,7
Lenc & Král 2017 [40] (CNN-150)	-	-	83,1

Tabulka 8.9: Porovnání výsledků pro český korpus s dostupnými externími výsledky.

Výsledky **mlp-bow** a Lenc & Král 2017 (MLP) jsou dle očekávání velice podobné, protože je použita stejná architektura NN a reprezentace dokumentu. To samé platí o **cnn-emb** a Lenc & Král 2017 (CNN-150). Rozdíl mezi CNN-best a CNN-150 je počet použitých slov z dokumentu a délka vektorových reprezentací *embedding* vrstvy – CNN-150 je výsledek pro stejný počet slov a délku reprezentací jako v **cnn-emb** (tj. 150 a 150), CNN-best je nejlepší dosažený výsledek. Lze usoudit, že s rostoucí délkou vektoru a počtem slov by úspěšnost **cnn-emb** také rostla. Hierarchická metoda si ovšem nevede dobře ani v porovnání s Brychcín & Král 2014, kde byla klasifikace provedena metodou maximální entropie za použití několika pokročilejších metod výběru příznaků.

8.3.2 Anglická datová sada

V tabulce 8.10 jsou uvedeny výsledky porovnání opětovně natrénovaných *baseline* s hierarchickými metodami reprezentace pro anglický korpus. Význam značení je stejný jako v tabulce 8.8. Zde si distribuované reprezentace v porovnání s nedistribuovanými vedou mírně lépe, než u českého korpusu, ale přesto většinou vychází hůře. Je možné, že 23 tisíc dokumentů v trénovací množině je pořad nedostatečný počet. Také nelze vyloučit, že použité architektury sítě nejsou pro distribuované reprezentace nejvhodnější.

Důležité je, že hierarchická reprezentace **hier-word** dosáhla celkově nejlepšího výsledku $\mathbf{F}_\mu$, přestože s rozdílem pouze $0,14$ % nad **mlp-bow**. Nad nehierarchickými distribuovanými metodami reprezentace **cnn-word** a **mlp-doc**

[%]	$\mathbf{P}_\mu$	$\mathbf{R}_\mu$	$\mathbf{F}_\mu$	$\mathbf{P}_\mathbf{M}$	$\mathbf{R}_\mathbf{M}$	$\mathbf{F}_\mathbf{M}$	$\mathbf{A}_\mathbf{ML}$
mlp-bow	**86,15**	78,23	82,01	67,14	**52,43**	**56,64**	**77,35**
cnn-emb	80,96	**80,56**	80,76	63,78	51,91	54,46	75,82
cnn-word	80,29	74,63	77,36	59,06	43,07	47,35	71,56
mlp-doc	80,15	75,09	77,54	62,68	47,09	51,59	71,17
hier-doc	81,14	78,84	77,86	56,36	46,93	48,56	71,74
hier-word	85,33	79,20	**82,15**	**67,77**	51,01	55,86	77,25

Tabulka 8.10: Výsledky pro anglický korpus (změřeno na celé testovací množině).

má náskok skoro 5 %, z čehož lze usoudit, že hierarchická (distribuovaná) reprezentace po větách má potenciál v porovnání s distribuovanou reprezentací po slovech nebo celých dokumentech. Je pravděpodobné, že s rostoucí kvalitou distribuovaných reprezentací *word2vec* a *doc2ec* by rostla i úspěšnost klasifikace využívající tyto reprezentace (tedy včetně hierarchických reprezentací).

Porovnání **hier-doc** a **hier-word** je konzistentní s českým korpusem. Reprezentace vět jako *doc2vec* vektor je výrazně horší ($\Delta\mathbf{F}_\mu = 4,29$ %), než průměr *word2vec* vektorů. Podobný je také dobrý výsledek **cnn-emb**. *Dense* reprezentace slov natrénovaná v *embedding* vrstvě (tj. laděná pro konkrétní úlohu klasifikace) je lepší, než obecné distribuované reprezentace z *word2vec* a *doc2vec*, ovšem s výjimkou **hier-word**, což jen dále potvrzuje, že hierarchická reprezentace po větách má za určitých okolností smysl.

Nezanedbatelná výhoda je také to, že při použití hierarchických reprezentací má konvoluční vrstva v použité architektuře CNN méně parametrů. Použití 30 vět oproti 150 slovům znamená pouze pětinu paremetrů, což má za důsledek kromě rychlejšího trénování sítě i menší spotřebu paměti. Díky tomu lze použít delší vektory. V tabulce 8.11 se nachází porovnání metody **word-hier** z předchozí tabulky, **word-hier-300** pro tu samou metodu reprezentace, ale s délkou vektorů 300, a **word-hier-pretrained**, kde jsou pro vytvoření reprezentace věty použity předtrénované vektory délky 300 (natrénované na GoogleNews[52]). Úspěšnost klasifikace **word-hier** a **word-hier-300** je takřka stejná, z čehož vyplývá, že hierarchická metoda v tomto případě netěží z větší délky vektorů (ale samozřejmě pro tento závěr by bylo

[52] `https://code.google.com/archive/p/word2vec/`

třeba vyzkoušet další velikosti, protože je možné, že 150 je málo, ale 300 už moc, a nejlepšího výsledku by dosáhly vektory délky třeba 250). Nicméně předtrénované vektory **hier-word-pretrained** mírně předčí trénované obou velikostí, což opět ukazuje, že předtrénované vektory mohou být užitečné i v úloze klasifikace do mnoha kategorií.

[%]	$\mathbf{F}_\mu$	$\mathbf{F_M}$	$\mathbf{A_{ML}}$
hier-word	82,15	55,86	77,25
hier-word-300	81,96	56,18	76,98
hier-word-pretrained	**82,20**	**58,70**	**77,51**

Tabulka 8.11: Výsledky hierarchické metody pro anglický korpus pro délku vektorů 300.

V tabulce 8.12 se nachází porovnání nejlepší hierarchické metody s výsledky jiných prací. Lewis et al. 2004 je výsledek SVM *baseline* z článku, seznamujícího s korpusem RCV1 (resp. RCV1-v2). Hierarchické metody tuto *baseline* mírně překonávají. Na druhou stranu ale mírně zaostávají za dalším dostupným výsledkem Soucy & Guy 2005, kde je pro klasifikaci RCV1-v2 použit algoritmus kNN spolu s pokročilejšími *supervised* metodami váhování příznaků.

[%]	$\mathbf{F}_\mu$
hier-word	82,15
hier-word-pretrained	82,20
Lewis et al. 2004 [42] (SVM)	81,6
Soucy & Guy 2005 [71] (kNN)	83,3

Tabulka 8.12: Porovnání výsledků hierarchické metody s dostupnými externími výsledky pro anglický korpus.

9 Závěr

Provedené experimenty ukázaly, že pro český korpus jsou hierarchické reprezentace v porovnání s ostatními distribuovanými lepší nebo srovnatelné. Avšak všechny distribuované (včetně hierarchických) jsou horší, než BoW a *embedding* vrstva. U anglického korpusu je situace jiná. Přestože většina distribuovaných reprezentací opět zaostává za nedistribuovanými, hierarchická metoda, kde je věta reprezentována průměrem slov z *word2vec* (**hier-word**), dosáhla celkově nejlepšího výsledku $\mathbf{F}_\mu = 82,15$ %. S tím souvisí zajímavý fakt, že (soudě podle výsledků **hier-doc** a **hier-word**) pro reprezentaci krátkých kusů textu (vět) vychází výrazně lépe průměr *word2vec* vektorů, než jediný vektor získaný z *doc2vec* modelu ($\mathbf{F}_\mu$ o $4,30$ % a $4,29$ % pro jednotlivé korpusy), přestože pro reprezentaci celých dokumentů tomu tak není.

V porovnání s externími výsledky jsou na tom výsledky nejlepší hierarchické metody o něco hůře, ale je nutné zohlednit také to, že nebyly použity žádné pokročilejší metody předzpracování (např. lemmatizace) nebo výběru relevantních příznaků, které obvykle zvyšují úspěšnost klasifikace [26; 71]. Lze očekávat, že při použití některých z těchto metod by se úspěšnost hierarchických metod zlepšila. Také by mohlo pomoci zavést metodu výběru příznaků až na úrovni hierarchie, tzn. vybrat pro každou větu pouze důležitá slova, případně pro dokument důležité věty.

Co se týká distribuovaných reprezentací, vyzkoušených v této práci, mají hierarchické metody vzhledem k dosaženým výsledkům potenciál a určitě nabízí poměrně široký prostor pro další výzkum. Mohlo by ale být vhodnější zabývat se hierarchií u klasifikace dokumentů, které jsou strukturované do více úrovní, než je pouze přirozená hierarchie vět a slov (např. [74]). Pro klasifikaci dokumentů, které obsahují pouze přirozenou hierarchickou strukturu, může být vhodným směrem výzkumu použití architektur sítě, ve kterých je přirozená hierarchie dokumentu přímo zahrnuta [83; 87].

Obecně horší výsledek distribuovaných reprezentací v porovnání s nedistribuovanými by mohl být způsoben několika faktory. Je možné, že distribuované reprezentace ještě nejsou na takové úrovni, aby pro všechny úlohy NLP konzistentně porážely *sparse* reprezentace jako je BoW, případně *embedding* vrstvu CNN. Důležité je také to, že jsou distribuované reprezentace tréno-

vané na jiné úloze (předpověď (z) kontextu) než je klasifikace, a proto nemusí být pro klasifikaci ideální. Za zmínku stojí také fakt, že hyperparametry trénování distribuovaných reprezentací a neuronových sítí nebyly v rámci této práce nijak laděny – s výjimkou zjištění vhodného počtu iterací trénování (kvůli nedoučení a přeučení), vhodné hodnoty pro prahování *multi-label* klasifikace a vhodného algoritmu pro generování distribuovaných reprezentací (CBOW vs. Skip-gram a DM vs. DBOW). Nelze vyloučit, že s jinými hyperparametry (např. tvar a velikost filtru konvoluční vrstvy apod.) by si distribuované reprezentace vedly lépe. Nicméně je pravděpodobné, že některé hyperparametry architektur obou sítí byly nějakým způsobem laděny v rámci práce, odkud byly architektury přejaty [40].

V rámci provedených experimentů bylo také zjištěno několik poznatků, které souvisí s tématem této práce. Navzdory tvrzení autorů *doc2vec*, že model DM poskytuje kvalitnější reprezentace než DBOW [39], byl zjištěn přesný opak (což je ale konzistentní s některými dalšími pracemi [21; 38]). Také se ukázalo, že *embedding* vrstva nezíská žádnou výhodu, pokud do ní budou vloženy natrénované reprezentace slov, ale je lepší, když jsou inicializovány náhodně a trénovány jako součást neuronové sítě (opět navzdory tvrzení [32], ale konzistentně s [41]).

Přehled zkratek a termínů

ČTK – Česká tisková kancelář

NLP – *Natural Language Processing* (zpracování přirozeného jazyka)

NBC – *Naive bayes classifier* (naivní bayesovský klasifikátor)

KNN, **kNN** – Algoritmus k-nejbližších sousedů (*k-nearest neighbor*)

SVM – *Support Vector Machines*

ME – Maximální entropie

ANN, **NN** – *Artifical neural network*, (umělá) neuronová síť

ReLU – *Rectified linear unit* (rektifikovaná lineární jednotka)

FNN – *Feed-forward* neuronová síť

RNN – Rekurentní neuronová síť

MLP – *Multi-layer perceptron* (vícevrstvý perceptron)

CNN – Konvoluční neuronová síť

BP – *backpropagation of error*, *backpropagation*

POS – z angl. „Part-of-speech“ – slovní druh

TF – *Term frequency* (frekvence slova)

IDF – *Inverse document frequency* (inverzní dokumentová frekvence)

TF-IDF, **tf-idf** – Algoritmus váhování *Term frequency-inverse document frequency*

BoW, **BOW** – *Bag-of-words* (způsob reprezentace dokumentů)

SVD – *Singular-value decomposition* (singulární dekompozice)

LSA – *Latent semantic analysis* (latentní sémantická analýza)

CBOW – *Continuous bag-of-words*

SG – *Skip-gram*

DM – *Distributed memory*

DBOW – *Distributed bag-of-words*

RCV1 – Reuters Corpus Volume 1

RCV1-v2, **RCV1-v2a** – Reuters Corpus Volume 1 verze 2 a verze 2a

CoNLL – *Computer Natural Language Learning* (Počítačové učení přirozeného jazyka), formát textových souborů

HAN – *Hierarchical attention network*

Literatura

[1] Abadi, M. et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Dostupné z: https://www.tensorflow.org/.

[2] Albelwi, S. – Mahmood, A. A Framework for Designing the Architectures of Deep Convolutional Neural Networks. *Entropy*. 2017, 19, 6. ISSN 1099-4300. doi: 10.3390/e19060242. Dostupné z: http://www.mdpi.com/1099-4300/19/6/242.

[3] Anjali, M. – Jivani, G. A Comparative Study of Stemming Algorithms. *International Joural of Computer Technology and Applications*. 2011, 2, s. 1930–1938.

[4] Arora, S. – Liang, Y. – Ma, T. A simple but tough-to-beat baseline for sentence embeddings. *International Conference on Learning Representations*. 2017.

[5] Bengio, Y. New Distributed Probabilistic Language Models. Technical Report 1215, Département d'informatique et recherche opérationnelle, Université de Montréal, 2002. Dostupné z: http://www.iro.umontreal.ca/~lisa/pointeurs/TR1215.ps.

[6] Bengio, Y. et al. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.* March 2003, 3, s. 1137–1155. ISSN 1532-4435. Dostupné z: http://dl.acm.org/citation.cfm?id=944919.944966.

[7] Berger, A. L. – Pietra, V. J. D. – Pietra, S. A. D. A Maximum Entropy Approach to Natural Language Processing. *Comput. Linguist.* March 1996, 22, 1, s. 39–71. ISSN 0891-2017. Dostupné z: http://dl.acm.org/citation.cfm?id=234285.234289.

[8] Bisht, S. – Paul, A. Document Clustering: A Review. *International Journal of Computer Applications*. July 2013, 73, 11.

[9] Bojanowski, P. et al. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606*. 2016.

[10] Bonatti, R. et al. Effect of Part-of-Speech and Lemmatization Filtering in Email Classification for Automatic Reply. In *AAAI Workshop: Knowledge Extraction from Text*, 2016.

[11] Brychcín, T. – Král, P. Novel unsupervised features for Czech multi-label document classification. In *Mexican International Conference on Artificial Intelligence*, s. 70–79. Springer, 2014.

[12] Chabacano. Diagram showing overfitting of a classifier, 2008. Dostupné z: `https://commons.wikimedia.org/wiki/File:Overfitting.svg`. [Online; citováno 23. května 2018].

[13] Chiang, T.-H. – Lo, H.-Y. – Lin, S.-D. A Ranking-based KNN Approach for Multi-Label Classification. In Hoi, S. C. H. – Buntine, W. (Ed.) *Proceedings of the Asian Conference on Machine Learning*, 25 / *Proceedings of Machine Learning Research*, s. 81–96, Singapore Management University, Singapore, 04–06 Nov 2012. PMLR. Dostupné z: `http://proceedings.mlr.press/v25/chiang12/chiang12.pdf`.

[14] Chollet, F. – others. Keras. `https://keras.io`, 2015.

[15] Collobert, R. – Weston, J. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, s. 160–167, New York, NY, USA, 2008. ACM. doi: 10.1145/1390156.1390177. Dostupné z: `http://doi.acm.org/10.1145/1390156.1390177`. ISBN 978-1-60558-205-4.

[16] De Boom, C. et al. Representation Learning for Very Short Texts Using Weighted Word Embedding Aggregation. *Pattern Recogn. Lett.* September 2016, 80, C, s. 150–156. ISSN 0167-8655. doi: 10.1016/j.patrec.2016.06.012. Dostupné z: `https://doi.org/10.1016/j.patrec.2016.06.012`.

[17] Elisseeff, A. – Weston, J. A Kernel Method for Multi-Labelled Classification. In *In Advances in Neural Information Processing Systems 14*, s. 681–687. MIT Press, 2001.

[18] Epelbaum, T. *Deep learning: Technical introduction* [online]. 09 2017. Dostupné z: `https://arxiv.org/pdf/1709.01412v2.pdf`.

[19] Fan, R.-E. – Lin, C.-J. A study on threshold selection for multi-label classification. Technical report, National Taiwan University, 05 2018.

[20] Fawcett, T. An Introduction to ROC Analysis. *Pattern Recogn. Lett.* June 2006, 27, 8, s. 861–874. ISSN 0167-8655. doi: 10.1016/j.patrec.2005.10.010. Dostupné z: `http://dx.doi.org/10.1016/j.patrec.2005.10.010`.

[21] Fisher, G. A. – Israni, M. Exploring Optimizations to Paragraph Vectors, 2017.

[22] Galavotti, L. S. – Fabrizioand Simi, M. Experiments on the Use of Feature Selection and Negative Evidence in Automated Text Categorization. In Borbinha, T. J. B. (Ed.) *Research and Advanced Technology for Digital Libraries*, s. 59–68, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. ISBN 978-3-540-45268-3.

[23] Goodfellow, I. – Bengio, Y. – Courville, A. Convolutional Networks. In *Deep Learning*. MIT: MIT Press, 2006. 9, s. 326–366. Dostupné z: `http://www.deeplearningbook.org/contents/convnets.html`.

[24] Guyon, I. A Scaling Law for the Validation-Set Training-Set Size Ratio. In *AT & T Bell Laboratories*, 1997.

[25] Hinton, G. E. – McClelland, J. L. – Rumelhart, D. E. Distributed Representations. In Rumelhart, D. E. – McClelland, J. L. – PDP Research Group, C. (Ed.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*. Cambridge, MA, USA: MIT Press, 1986. Distributed Representations, s. 77–109. Dostupné z: `http://dl.acm.org/citation.cfm?id=104279.104287`. ISBN 0-262-68053-X.

[26] HRALA, M. Automatická klasifikace dokumentů s podobným obsahem [online]. Master's thesis, University of West Bohemia, Faculty of Applied Sciences, Plzeň, 2012. Dostupné z: `https://theses.cz/id/7satjf/`.

[27] Hrala, M. – Král, P. Evaluation of the Document Classification Approaches. In *Proceedings of the 8th International Conference on Computer Recognition Systems CORES 2013*, s. 877–885, Heidelberg, 2013. Springer International Publishing. ISBN 978-3-319-00969-8.

[28] James, G. et al. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014. ISBN 1461471370, 9781461471370.

[29] Johnson, R. – Zhang, T. Effective Use of Word Order for Text Categorization with Convolutional Neural Networks. *CoRR*. 2014, abs/1412.1058.

[30] Kafrawy, P. E. – Mausad, A. – Esmail, H. Article: Experimental Comparison of Methods for Multi-label Classification in different Application Domains. *International Journal of Computer Applications*. March 2015, 114, 19, s. 1–9.

[31] Kalman, D. A singularly valuable decomposition: The SVD of a matrix. *College Math Journal*. 1996, 27, s. 2–23.

[32] Kim, Y. Convolutional Neural Networks for Sentence Classification. *CoRR*. 2014, abs/1408.5882.

[33] Král, P. – Lenc, L. Czech Text Document Corpus v 2.0. *CoRR*. 2017, abs/1710.02365. Dostupné z: http://arxiv.org/abs/1710.02365.

[34] Kriesel, D. *A Brief Introduction to Neural Networks*. David Kriesel, 2007. Dostupné z: http://www.dkriesel.com/en/science/neural_networks.

[35] Krizhevsky, A. – Sutskever, I. – Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In Pereira, F. et al. (Ed.) *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc.: Curran Associates, Inc., 2012. s. 1097–1105. Dostupné z: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[36] Lan, M. et al. A Comprehensive Comparative Study on Term Weighting Schemes for Text Categorization with Support Vector Machines. In *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web*, WWW '05, s. 1032–1033, New York, NY, USA, 2005. ACM. doi: 10.1145/1062745.1062854. Dostupné z: http://doi.acm.org/10.1145/1062745.1062854. ISBN 1-59593-051-5.

[37] Landauer, T. – Foltz, P. – Laham, D. An introduction to latent semantic analysis. *Discourse processes*. 1998, 25, s. 259–284. Dostupné z: http://lsa.colorado.edu/papers/dp1.LSAintro.pdf.

[38] Lau, J. H. – Baldwin, T. An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation. *CoRR*. 2016, abs/1607.05368.

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

[40] Lenc, L. – Král, P. Deep Neural Networks for Czech Multi-label Document Classification. *CoRR*. 2017, abs/1701.03849.

[41] Lenc, L. – Král, P. Word Embeddings for Multi-label Document Classification. In *RANLP*, 2017.

[42] Lewis, D. D. et al. RCV1: A New Benchmark Collection for Text Categorization Research. *J. Mach. Learn. Res.* December 2004, 5, s. 361–397. ISSN 1532-4435. Dostupné z: http://dl.acm.org/citation.cfm?id=1005332.1005345.

[43] Li, X. – Guo, Y. Active learning with multi-label SVM classification. In *IJCAI International Joint Conference on Artificial Intelligence*, s. 1479–1485, 08 2013.

[44] Liu, M. – Yang, J. An improvement of TFIDF weighting in text categorization. *International Conference on Computer Technology and Science*. 2012. doi: 10.7763/IPCSIT.2012.V47.9. Dostupné z: `https://pdfs.semanticscholar.org/6876/855902ca8fc7bfd74fc3eb8ec139c3bfe145.pdf`.

[45] Machine Learner. Nonlinear SVM example illustration, 2014. Dostupné z: `https://commons.wikimedia.org/wiki/File:Nonlinear_SVM_example_illustration.svg`. [Online; citováno 23. května 2018].

[46] Manning, C. D. – Schütze, H. *Foundations of Statistical Natural Language Processing*, s. 341–380. MIT Press, 1999.

[47] Manning, C. D. – Raghavan, P. – Schütze, H. *An Introduction to Information Retrieval*, s. 32–35. Cambridge University Press, online edition, 2009. Dostupné z: `https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf`.

[48] Mikolov, T. et al. Efficient Estimation of Word Representations in Vector Space. *CoRR*. 2013, abs/1301.3781. Dostupné z: `http://arxiv.org/abs/1301.3781`.

[49] Nam, J. et al. Large-scale Multi-label Text Classification - Revisiting Neural Networks. *CoRR*. 2013, abs/1312.5419.

[50] Ng, A. Y. Preventing "Overfitting"of Cross-Validation Data. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, s. 245–253, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. Dostupné z: `http://dl.acm.org/citation.cfm?id=645526.657119`. ISBN 1-55860-486-3.

[51] Pagliardini, M. – Gupta, P. – Jaggi, M. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. *CoRR*. 2017, abs/1703.02507. Dostupné z: `http://arxiv.org/abs/1703.02507`.

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

[53] Peterson, L. E. K-nearest neighbor. *Scholarpedia*. 2009, 4, 2, s. 1883. doi: 10.4249/scholarpedia.1883. revision #136646.

[54] Porter, M. F. Readings in Information Retrieval. In Sparck Jones, K. – Willett, P. (Ed.) *Readings in Information Retrieval*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997. An Algorithm for Suffix Stripping, s. 313–316. Dostupné z: http://dl.acm.org/citation.cfm?id=275537.275705. ISBN 1-55860-454-5.

[55] Powers, D. M. W. Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. *Journal of Machine Learning Technologies*. 2011, 2, 1, s. 37–63.

[56] Prakash, V. J. – Nithya, L. M. A Survey on Semi-Supervised Learning Techniques. *CoRR*. 2014, abs/1402.4645. Dostupné z: http://arxiv.org/abs/1402.4645.

[57] Prechelt, L. Early Stopping - but when? In *Neural Networks: Tricks of the Trade, volume 1524 of LNCS, chapter 2*, s. 55–69. Springer-Verlag, 1997.

[58] Refaeilzadeh, P. – Tang, L. – Liu, H. Cross-Validation. In *Encyclopedia of Database Systems*, 532–538, s. 532–538, 01 2009.

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

[60] Rose, T. – Stevenson, M. – Whitehead, M. The reuters corpus volume 1-from yesterday's news to tomorrow's language resources. *Proceedings of the Third International Conference on Language Resources and Evaluation*. 2002, s. 29–31. Dostupné z: http://about.reuters.com/researchandstandards/corpus/LREC_camera_ready.pdf.

[61] Salle, A. – Idiart, M. – Villavicencio, A. Matrix Factorization using Window Sampling and Negative Sampling for Improved Word Representations. *CoRR*. 2016, abs/1606.00819.

[62] Salle, A. – Idiart, M. – Villavicencio, A. Enhancing the LexVec Distributed Word Representation Model Using Positional Contexts and External Memory. *CoRR*. 2016, abs/1606.01283.

[63] Salton, G. – Wong, A. – Yang, C. S. A Vector Space Model for Automatic Indexing. *Commun. ACM*. November 1975, 18, 11, s. 613–620. ISSN 0001-0782. doi: 10.1145/361219.361220. Dostupné z: http://doi.acm.org/10.1145/361219.361220.

[64] Salton, G. – Buckley, C. Term-weighting Approaches in Automatic Text Retrieval. *Inf. Process. Manage.* August 1988, 24, 5, s. 513–523. ISSN 0306-4573. doi: 10.1016/0306-4573(88)90021-0. Dostupné z: http://dx.doi.org/10.1016/0306-4573(88)90021-0.

[65] Sebastiani, F. Machine Learning in Automated Text Categorization. *ACM Comput. Surv.* March 2002, 34, 1, s. 1–47. ISSN 0360-0300. doi: 10.1145/505282.505283. Dostupné z: http://doi.acm.org/10.1145/505282.505283.

[66] Sharma, D. – Jain, S. Evaluation of Stemming and Stop Word Techniques on Text Classification Problem. *International Journal of Scientific Research in Commputer Science and Engineering.* 2015, 3. ISSN 2320-7639. Dostupné z: https://pdfs.semanticscholar.org/c5d1/db4509ea7fe81f6c9f0a5f2db8339ef7cb2d.pdf.

[67] Sobisek, L. – Řezanková, H. Srovnání metod pro redukci dimenzionality aplikovaných na ordinální proměnné. *Acta Oeconomica Pragensia.* 01 2011, 19, s. 3–19. ISSN 0572-3043.

[68] Socher, R. et al. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 2011.

[69] Sokolova, M. – Lapalme, G. A systematic analysis of performance measures for classification tasks. *Inf. Process. Manage.* 2009, 45, s. 427–437.

[70] Sorower, M. S. A literature survey on algorithms for multi-label learning. Technical report, Department of Computer Science. Oregon State University, 2010.

[71] Soucy, P. – Mineau, G. W. Beyond TFIDF Weighting for Text Categorization in the Vector Space Model. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, s. 1130–1135, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc. Dostupné z: http://dl.acm.org/citation.cfm?id=1642293.1642474.

[72] Srividhya, V. – Anitha, R. Evaluating preprocessing techniques in text categorization. *International Journal of Computer Science and Application.* 2010. ISSN 0974-0767. Dostupné z: http://sinhgad.edu/ijcsa-2012/pdfpapers/1_11.pdf.

[73] Steinbach, M. – Karypis, G. – Kumar, V. A comparison of document clustering techniques. In *In KDD Workshop on Text Mining*, 2000.

[74] Teofili, T. par2hier: towards vector representations for hierarchical content. *Procedia Computer Science*. 2017, 108, s. 2343 – 2347. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2017.05.077. Dostupné z: http://www.sciencedirect.com/science/article/pii/S1877050917306154. International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland.

[75] Toman, M. – Tesar, R. – Jezek, K. Influence of Word Normalization on Text Classification, 2006. Dostupné z: http://textmining.zcu.cz/publications/inscit20060710.pdf.

[76] Tsoumakas, G. – Katakis, I. Multi-Label Classification: An Overview. *International Journal of Data Warehousing and Mining*. 09 2009, 3, s. 1–13.

[77] Uysal, A. K. – Gunal, S. The Impact of Preprocessing on Text Classification. *Inf. Process. Manage.* January 2014, 50, 1, s. 104–112. ISSN 0306-4573. doi: 10.1016/j.ipm.2013.08.006. Dostupné z: http://dx.doi.org/10.1016/j.ipm.2013.08.006.

[78] W.A, A. – S.M, E. Machine Learning Methods for Spam E-Mail Classification. *International Journal of Computer Science and Information Technology*. 02 2011, 3.

[79] Wei, Z. et al. A naive Bayesian multi-label classification algorithm with application to visualize text Search Results. *International Journal of Advanced Intelligence*. 2011, 3, 2, s. 173–188.

[80] Wieting, J. et al. Towards Universal Paraphrastic Sentence Embeddings. *CoRR*. 2015, abs/1511.08198. Dostupné z: http://arxiv.org/abs/1511.08198.

[81] Yang, Y. – Joachims, T. Text categorization. *Scholarpedia*. 2008, 3, 5, s. 4242. doi: 10.4249/scholarpedia.4242. revision #91858.

[82] Yang, Y. – Pedersen, J. O. A Comparative Study on Feature Selection in Text Categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, s. 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. Dostupné z: http://dl.acm.org/citation.cfm?id=645526.657137. ISBN 1-55860-486-3.

[83] Yang, Z. et al. Hierarchical Attention Networks for Document Classification. In *HLT-NAACL*, s. 1480–1489, 01 2016.

[84] Zhang, M. – Zhou, Z. A Review On Multi-Label Learning Algorithms. *IEEE Transactions on Knowledge and Data Engineering*. 05 2013. ISSN 1041-4347. doi: 10.1109/TKDE.2013.39. Dostupné z: https://www.computer.org/csdl/trans/tk/2014/08/06471714-abs.html.

[85] Zhang, M.-L. – Zhou, Z.-H. Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization. *IEEE Trans. on Knowl. and Data Eng.* October 2006, 18, 10, s. 1338–1351. ISSN 1041-4347. doi: 10.1109/TKDE.2006.162. Dostupné z: http://dx.doi.org/10.1109/TKDE.2006.162.

[86] Zhang, M.-L. – Zhou, Z.-H. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*. 2007, 40, 7, s. 2038 – 2048. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2006.12.019. Dostupné z: www.sciencedirect.com/science/article/pii/S0031320307000027.

[87] Zheng, J. et al. A Hierarchical Neural-Network-Based Document Representation Approach for Text Classification. *Mathematical Problems in Engineering*. 2018, 2018. doi: https://doi.org/10.1155/2018/7987691. Article ID 7987691.

[88] Zhu, S. et al. Multi-labelled Classification Using Maximum Entropy Method. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, s. 274–281, New York, NY, USA, 2005. ACM. doi: 10.1145/1076034.1076082. Dostupné z: http://doi.acm.org/10.1145/1076034.1076082. ISBN 1-59593-034-5.

A Ukázka českého dokumentu ve formátu CoNLL-U

```
# newdoc
# newpar
# sent_id = 3
# text = Věří, že se vztahy nezhorší.
1       Věří    věřit   VERB    VB-S---3P-AA--- Aspect=Imp|Mood=Ind|
    Number=Sing|Person=3|Polarity=Pos|Tense=Pres|VerbForm=Fin|Voice
    =Act 0 root _    SpaceAfter=No
2       ,       ,       PUNCT   Z:------------- _       5       punct   _
          _
3       že      že      SCONJ   J,------------- _       5       mark    _
          _
4       se      se      PRON    P7-X4---------- Case=Acc|PronType=Prs|
    Reflex=Yes|Variant=Short 5 expl:pass      _       _
5       vztahy  vztah   NOUN    NNIP1-----A---- Animacy=Inan|Case=Nom|
    Gender=Masc|Number=Plur|Polarity=Pos 1    ccomp   _       _
6       nezhorší        zhoršit PROPN   NNMS1-----A---- Animacy=Anim|
    Case=Nom|Gender=Masc|NameType=Sur|Number=Sing|Polarity=Pos 5
    nmod _      SpaceAfter=No
7       .       .       PUNCT   Z:------------- _       1       punct   _
          SpaceAfter=No
```

B Seznam kategorií českého korpusu s četnostmi

Zkratka	Název	Četnost[53]
aut	Automobilový průmysl	431
bos	Bohemika	~~186~~
bsk	Sklářský průmysl	~~16~~
bua	Burzy akciové	~~188~~
buk	Burzy komoditní	~~258~~
bup	Burzy peněžní	365
bur	Burzy	687
cen	-	~~25~~
che	Chemický a farmaceutický průmysl	262
den	Zpravodajské deníky	600
dpr	Doprava	1061
dre	Dřevozpracující průmysl	~~44~~
efm	Firmy	2011
ekl	Životní prostředí	765
eko	Ekologie	~~40~~
ene	Energie	769
eur	Evropská unie - zprávy	553
fin	Finanční služby	684
for	Parlamenty a vlády	1293
fot	Fotbal - zprávy	~~165~~
hok	Hokej - zprávy	~~196~~
hut	Hutnictví	~~35~~
kat	Neštěstí a katastrofy	~~68~~
kul	Kultura	1105
mag	Magazínový výběr	1375
mak	Makroekonomika	838
med	Média a reklama	336
met	Počasí	698
mix	Mix	396
mot	Motorismus	~~32~~

[53]V trénovací množině. Přeškrtnutím jsou vyznačeny četnosti kategorií, na které při klasifikaci nebude brán zřetel kvůli nízkému výskytu.

Zkratka	**Název**	**Četnost**[53]
nab	Náboženství	~~185~~
obo	Obchod	684
odb	Práce a odbory	639
pit	Telekomunikace a IT	645
pla	Plány zpravodajství ČTK	~~33~~
pod	Politika ČR	1834
pol	Politika	2762
prg	Pragensia	325
prm	Lehký průmysl	~~185~~
ptr	Potravinářství	337
reg	Region	~~27~~
sko	Školství	438
slo	Slovenika	~~74~~
slz	Služby	927
sop	Sociální	464
spc	-	~~3~~
spl	Životní styl	753
spo	Sportovní zpravodajství	1068
sta	Stavebnictví a reality	787
str	Strojírenství	573
sur	Suroviny	356
tlk	Telekomunikace	~~173~~
tok	Textil	~~43~~
tur	Cestovní	511
vat	Věda a technika	~~102~~
zah	Zahraniční	~~28~~
zak	Kriminalita a právo	1687
zbr	Zbraně	~~17~~
zdr	Zdravotnictví	961
zem	Zemědělství	544

C Ukázka anglického *xml* dokumentu

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<newsitem itemid="100000" id="root" date="1996-10-07" xml:lang="en
    ">
<title>USA: NYCE cotton closes up on Tropical Storm Josephine.</
    title>
<headline>NYCE cotton closes up on Tropical Storm Josephine.</
    headline>
<dateline>NEW YORK 1996-10-07</dateline>
<text>
<p>Light speculative buying buoyed NYCE cotton futures to a higher
    close as Tropical Storm Josephine was poised to deluge an
    already soaked Southeast U.S. with more rain, industry sources
    said.</p>
<p>The storm, forecast to hit Florida Tuesday, could bring between
    two to five inches of rain to Georgia and the Carolinas.</p>
<p>&quot;Most areas from northern Florida to Georgia to the
    Carolinas...will receive two to five inches (of rain),&quot;
    Jon Davis, a meteorologist with Weather Services Corp, said.</p
    >
<p>But volume was estimated at only 6,000 lots as players remained
    cautious ahead of USDA's monthly supply and demand report,
    released Friday.</p>
<p>Players also awaited this week's import data due Wednesday, and
    export sales due to be released Thursday.</p>
<p>December cotton closed 0.92 cent higher at 77.65 cents a lb
    after trading from 77.74 to 76.90. Deferreds gained 0.83 to
    0.30 cent.</p>
<p>--Suzanne Rostler, New York Commodities 212-859-1646</p>
</text>
<copyright>(c) Reuters Limited 1996</copyright>
<metadata>
<codes class="bip:countries:1.0">
  <code code="USA">
    <editdetail attribution="Reuters BIP Coding Group" action="
        confirmed" date="1996-10-07"/>
  </code>
```

```
</codes>
<codes class="bip:topics:1.0">
  <code code="M14">
    <editdetail attribution="Reuters BIP Coding Group" action="
        confirmed" date="1996-10-07"/>
  </code>
  <code code="M141">
    <editdetail attribution="Reuters BIP Coding Group" action="
        confirmed" date="1996-10-07"/>
  </code>
  <code code="MCAT">
    <editdetail attribution="Reuters BIP Coding Group" action="
        confirmed" date="1996-10-07"/>
  </code>
</codes>
<dc element="dc.date.created" value="1996-10-07"/>
<dc element="dc.publisher" value="Reuters Holdings Plc"/>
<dc element="dc.date.published" value="1996-10-07"/>
<dc element="dc.source" value="Reuters"/>
<dc element="dc.creator.location" value="NEW YORK"/>
<dc element="dc.creator.location.country.name" value="USA"/>
<dc element="dc.source" value="Reuters"/>
</metadata>
</newsitem>
```

D Seznam kategorií anglického korpusu

Zkratka	Název[54]
C11	STRATEGY/PLANS
C12	LEGAL/JUDICIAL
C13	REGULATION/POLICY
C14	SHARE LISTINGS
C15	PERFORMANCE
C151	ACCOUNTS/EARNINGS
C1511	ANNUAL RESULTS
C152	COMMENT/FORECASTS
C16	INSOLVENCY/LIQUIDITY
C17	FUNDING/CAPITAL
C171	SHARE CAPITAL
C172	BONDS/DEBT ISSUES
C173	LOANS/CREDITS
C174	CREDIT RATINGS
C18	OWNERSHIP CHANGES
C181	MERGERS/ACQUISITIONS
C182	ASSET TRANSFERS
C183	PRIVATISATIONS
C21	PRODUCTION/SERVICES
C22	NEW PRODUCTS/SERVICES
C23	RESEARCH/DEVELOPMENT
C24	CAPACITY/FACILITIES
C31	MARKETS/MARKETING
C311	DOMESTIC MARKETS
C312	EXTERNAL MARKETS
C313	MARKET SHARE
C32	ADVERTISING/PROMOTION
C33	CONTRACTS/ORDERS
C331	DEFENCE CONTRACTS
C34	MONOPOLIES/COMPETITION
C41	MANAGEMENT

[54]Přeškrtnutím jsou vyznačeny kategorie, které se nenachází v trénovací množině.

Zkratka	**Název**[54]
C411	MANAGEMENT MOVES
C42	LABOUR
CCAT	CORPORATE/INDUSTRIAL
E11	ECONOMIC PERFORMANCE
E12	MONETARY/ECONOMIC
E121	MONEY SUPPLY
E13	INFLATION/PRICES
E131	CONSUMER PRICES
E132	WHOLESALE PRICES
E14	CONSUMER FINANCE
E141	PERSONAL INCOME
E142	CONSUMER CREDIT
E143	RETAIL SALES
E21	GOVERNMENT FINANCE
E211	EXPENDITURE/REVENUE
E212	GOVERNMENT BORROWING
E31	OUTPUT/CAPACITY
E311	INDUSTRIAL PRODUCTION
~~E312~~	~~CAPACITY UTILIZATION~~
E313	INVENTORIES
E41	EMPLOYMENT/LABOUR
E411	UNEMPLOYMENT
E51	TRADE/RESERVES
E511	BALANCE OF PAYMENTS
E512	MERCHANDISE TRADE
E513	RESERVES
E61	HOUSING STARTS
E71	LEADING INDICATORS
ECAT	ECONOMICS
G15	EUROPEAN COMMUNITY
G151	EC INTERNAL MARKET
G152	EC CORPORATE POLICY
G153	EC AGRICULTURE POLICY
G154	EC MONETARY/ECONOMIC
G155	EC INSTITUTIONS
G156	EC ENVIRONMENT ISSUES
G157	EC COMPETITION/SUBSIDY
G158	EC EXTERNAL RELATIONS
G159	EC GENERAL

Zkratka	**Název**[54]
GCAT	GOVERNMENT/SOCIAL
GCRIM	CRIME, LAW ENFORCEMENT
GDEF	DEFENCE
GDIP	INTERNATIONAL RELATIONS
GDIS	DISASTERS AND ACCIDENTS
GENT	ARTS, CULTURE, ENTERTAINMENT
GENV	ENVIRONMENT AND NATURAL WORLD
GFAS	FASHION
GHEA	HEALTH
GJOB	LABOUR ISSUES
~~GMIL~~	~~MILLENNIUM ISSUES~~
GOBIT	OBITUARIES
GODD	HUMAN INTEREST
GPOL	DOMESTIC POLITICS
GPRO	BIOGRAPHIES, PERSONALITIES, PEOPLE
GREL	RELIGION
GSCI	SCIENCE AND TECHNOLOGY
GSPO	SPORTS
GTOUR	TRAVEL AND TOURISM
GVIO	WAR, CIVIL WAR
GVOTE	ELECTIONS
GWEA	WEATHER
GWELF	WELFARE, SOCIAL SERVICES
M11	EQUITY MARKETS
M12	BOND MARKETS
M13	MONEY MARKETS
M131	INTERBANK MARKETS
M132	FOREX MARKETS
M14	COMMODITY MARKETS
M141	SOFT COMMODITIES
M142	METALS TRADING
M143	ENERGY MARKETS
MCAT	MARKETS

E Struktura přiloženého DVD

- `bin` – obsahuje ukázkové skripty
  - `data` – použité datové sady
    - `cz` – předzpracovaný a původní český korpus
    - `en` – předzpracovaný anglický korpus
  - `pretrained` – některé předtrénované slovní vektory (*word2vec*, *GLoVe*, *fastText*) a *sent2vec* model
  - `vysledky` - složka, kam ukázkové skripty ukládají výstup
    - `experimenty` – zdroje výsledků experimentů, které jsou uvedeny v textu DP
- `poster` – zdrojový soubor a pdf posteru
- `text` – zde se nachází text diplomové práce (pdf)
  - `src` – obsahuje zdrojový kód textu a další soubory nutné pro přeložení do pdf
- `src` – zdrojové kódy pro komplikaci nebo instalaci
  - `dp` – zdrojový kód a instalační skripty implementované knihovny
  - `external_tools` – zdrojový kód a kompilační skripty *fastText* a *sent2vec*

F Příručka

F.1 Instalace

Pro správnou funkci programu je nutné mít nainstalovaný 64 bitový Python verze alespoň 3.6.3[55] (na nižších verzích nebo 32 bitovém Pythonu není zaručen bezchybný běh). Pro kompilaci externí implementace *fastText* a *Sent2vec* je dále nutné mít předkladač C++ (vyzkoušeno na verzi MinGW-W64 gcc 4.8.1[56], autoři uvádějí gcc minimálně verze 4.8). Komplikace proběhne standardně pomocí `makefile` příkazem *make* ve složce `src/external_tools/fasttext` pro *fastText* a ve složce `src/external_tools/sent2vec` pro *Sent2vec*. V obou případech bude vygenerován v příslušné složce spustitelný soubor *fasttext.exe*.

Pro instalaci implementované knihovny je nutné ve složce `src/dp` postupně spustit následující příkazy. První nainstaluje závislosti programu, druhý nainstaluje program do lokálního úložiště Python balíků[57]. Pokud se z nějakého důvodu instalace knihovny nepovede[58], je možné skript na její používání umístit přímo do složky `src/dp` – moduly knihovny budou importovány skriptem lokálně místo z úložiště Python balíků.

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

Tyto příkazy nainstalují verzi TensorFlow (resp. implementované knihovny) pro procesor. Přestože trénování neuronových sítí apod. trvá na procesoru déle než na grafické kartě, je instalace TensorFlow pro grafickou kartu poměrně komplikovaná[59], a proto instalační konfigurace, která je umístěna na přiloženém DVD, nainstaluje CPU verzi. Pro instalaci TensorFlow (resp. implementované knihovny) pro grafickou kartu je nutné provést 3 následující kroky.

[55] `https://www.python.org/downloads/release/python-363/`

[56] `https://sourceforge.net/projects/mingwbuilds/files/host-windows/releases/4.8.1/64-bit/`

[57] Nejčastěji `c:/Program Files/Python36/Lib/site-packages/`

[58] Ale pouze instalace implementované knihovny, závislosti z `requirements.txt` jsou nezbytné.

[59] Pokyny na `https://www.tensorflow.org/install/`

- Mít NVIDIA grafickou kartu s kompatibilitou *CUDA Compute Capability 3.5*
- Nainstalovat CUDA toolkit 9.0[60] a cuDNN v7.0[61].
- Změnit v `requirements.txt` řádku „tensorflow“ na „tensorflow-gpu“ a spustit výše uvedené příkazy stejně jako při instalaci pro procesor.

Dále pro instalaci *Sent2vec* do úložiště Python balíků, aby bylo možné importovat pomocí implementované natrénované *sent2vec* modely, je nutné ve složce `src/external_tools/sent2vec/src` spustit postupně následující příkazy. Zdrojový kód *Sent2vec* byl mírně upraven, aby šel program zkompilovat na Windows. Neupravenou verzi lze stáhnout z Git repozitáře na `https://github.com/epfml/sent2vec`. Knihovnu lze samozřejmě využívat i pokud instalace *Sent2vec* neproběhne, ale v tomto případě nelze importovat natrénovaný *Sent2vec* model.

```
pip install -r requirements.txt
py setup.py build_ext
pip install .
```

F.2 Návod k použití

F.2.1 Formát datových sad

Aby byla knihovna schopna načíst dokumenty je nezbytné, aby byly pojmenovány `id_c1_c2_..._cn.sent`, kde `id` je unikátní číselný identifikátor a `c1`, `c2`, ..., `cn` jsou názvy kategorií – například `0003_pol_kul.sent`. Tyto dokumenty obsahují prostý text, kde na každé řádce je jedna věta, která je složená ze slov oddělených mezerou.

Pro generování těchto souborů lze použít dvě funkce `prepare_czech_ctk()` a `prepare_english_rcv()`, které se nachází v nainstalované knihovně v modulech `dp_moravka/tools/prepare_czech` a `dp_moravka/tools/prepare_english`. Obě funkce uživatele dialogovými okny požádají o složku, kde se nachází dokumenty v původním formátu, a o složku, kam mají být soubory generovány. Na přiloženém DVD je ovšem v původním formátu přítomen

[60] `https://docs.nvidia.com/cuda/cuda-quick-start-guide/index.html`
[61] `https://developer.nvidia.com/cudnn`

pouze český korpus ve složce `bin/data/cz`. Anglický i zkomprimovaný zabírá moc diskového prostoru (několik gigabajtů). Na DVD jsou samozřejmě umístěny také oba korpusy v předzpracované podobě (a tedy ve formátu, který program vyžaduje) ve složkách `bin/data/cz` a `bin/data/en`.

F.2.2 Ovládání knihovny

Pro využití knihovny je nutné napsat krátký Python skript (pojmenovaný např. `skript.py`), který lze pak spustit příkazem `py skript.py`. Budou zde popsány základní principy a možnosti API, ale pro zjištění například všech parametrů nějaké funkce je nutné přečíst si dokumentaci v kódu knihovny – všechny třídy, metody a funkce, které lze použít, mají podrobnou dokumentaci. Na přiloženém DVD je uloženo několik vzorových (podrobně komentovaných) skriptů, ze kterých lze vidět použití API knihovny nejlépe.

Níže lze vidět základní ukázku takového skriptu. Pod ním je pak uveden sled činností, které musí skript provádět, a třídy a funkce, které lze využít.

```
from moravka_dp.input.document_source import DocsReaderSimple
from moravka_dp.input.input_provider import
                                    create_input_provider_word2vec
from moravka_dp.network.model import create_model_cnn
from moravka_dp.network.nn import NN
from moravka_dp.utils.print_utils import print_results_table

path_train ='some path'
path_test ='some path'
out_path ='some path'

# (1) create document reader
docsCz =DocsReaderSimple(folder_train=path_train, folder_test=path_test,
                                    num_most_common_labels=37)

# get documents and labels
docs_train =docsCz.get_docs(train=True)
docs_eval =docsCz.get_docs(train=False)
all_labels =docsCz.get_all_labels()

# (2) create input provider
inputProvider =create_input_provider_word2vec(docs=docs_train,
                                    vector_length=100, num_words=200)

# (3) create model
model =create_model_cnn(dim_out=len(all_labels), input_size=200,
                                    embedding_length=100)
```

```
# (4) train neural network
nn = NN(model, inputProvider, all_labels)
nn.train(docs_train, epochs=10)

# (5) evaluate neural network
results =nn.evaluate(docs_eval)

# (6) save results to file
print_results_table(results, file_name=out_path)
```

1. **Načtení dokumentů** – Pro načtení dokumentů ze souboru lze použít třídy `DocsReaderSimple` a `DocsReaderSplit`. Obě třídy mají metodu `get_all_labels` pro získání seznamu všech kategorií, nalezených v názvech načtených souborů. Dále mají tyto třídy metodu `get_docs`, která přečte ze souborů dokumenty a vrátí jejich seznam.

2. **Transformace dokumentů** – Dále je nutné vytvořit instanci třídy, která je schopná dokumenty transformovat na číselné reprezentace, kterým bude rozumět neuronová síť. Tuto transformaci provádí třídy v souboru `input_provider.py`, ale pro vytvoření instancí těchto tříd slouží *factory* funkce v souboru `input_provider_factory.py`.

   - `create_input_provider_word2vec` – Vytvoří instancí třídy, která převádí dokumenty na *word2vec* reprezentace.
   - `create_input_provider_word2vec_emb` – Vytvoří instancí třídy, která převádí dokumenty na *word2vec* reprezentace, a navíc poskytuje metodu `get_embeddings` pro získání matice vektorů všech slov ve slovníku. Tuto matici lze vložit do *embedding* vrstvy CNN.
   - `create_input_provider_word2vec_hier` – To samé jako `create_input_provider_word2vec`, ale reprezentace dokumentu je vytvářena po větách (jako průměr slov), ne po jednotlivých slovech.
   - `create_input_provider_pretrained_word_vectors` – To samé jako `create_input_provider_word2vec`, ale předtrénované vektory jsou načítány ze souboru.
   - `create_input_provider_pretrained_word_vectors_emb` – To samé jako `create_input_provider_word2vec_emb`, ale předtrénované vektory jsou načítány ze souboru.
   - `create_input_provider_pretrained_word_vectors_hier` – To samé jako `create_input_provider_word2vec_hier`, ale předtrénované slovní vektory jsou načteny ze souboru.

- `create_input_provider_doc2vec` – Vytvoří instanci třídy, která převádí dokument na *doc2vec* reprezentaci.
- `create_input_provider_doc2vec_hier` – To samé jako `create_input_provider_doc2vec`, ale dokument není reprezentován jedním vektorem, ale po větách.
- `create_input_provider_sent2vec` – Vytvoří instanci třídy, sloužící pro převod dokumentu na *Sent2vec* reprezentaci. Předtrénovaný model *Sent2vec* je načten z určeného souboru.
- `create_input_provider_bow` – Vytvoří instanci třídy, která vytváří *baf-of-words* reprezentace dokumentů.

3. **Model neuronové sítě** – Dalším krokem je inicializace modelu neuronové sítě. K dispozici jsou 3 funkce.
   - `create_model_mlp` – Tato funkce vytvoří model vícevrstvého perceptronu podle zadaných parametrů.
   - `create_model_cnn` – Vytvoří model konvoluční sítě podle zadaných parametrů (bez *embedding* vrstvy).
   - `create_model_cnn_emb` – Vytvoří model konvoluční sítě s *embedding* vrstvou podle zadaných parametrů.
4. **Trénování sítě** – Nejprve je nutné vytvořit instanci třídy `NN`, která ke své činnosti potřebuje model sítě, transformátor dokumentů (*inputProvider*) a seznam všech kategorií. Poté lze zavolat metodu `train` pro natrénování sítě na zadaných dokumentech.
5. **Evaluace sítě** – Evaluace sítě se provádí metodou `evaluate` třídy `NN`. Kromě dokumentů, na kterých se má evaluace provést, je možné specifikovat seznam prahů. Metoda vrací tabulku výsledků – každý sloupec odpovídá metrikám vypočteným pro jeden z daných prahů.
6. **Výpis a uložení výsledků** – Posledním krokem celého procesu je výsledky upravit do čitelné podoby (zaokrouhlení, přidání hlavičky tabulky atd.) a vypsat do konzole nebo uložit do souboru, což provádí funkce `print_results_table`. Výstup může vypadat např. jako tabulka F.1 (pro prahy 0,2, 0,4 a 0,6).

F.2.3 Použití externích programů

Kompilací *fastText* (na DVD složka `src/external_tools/fasttext`) nebo *sent2vec* (složka `src/external_tools/sent2vec`) vznikne spustitelný soubor *fasttext.exe*, který se ovládá z příkazové řádky. Pokud se do příkazové

THRESHOLD	0.2	0.4	0.6
true positives	4252	3832	3459
false positives	2088	1102	619
true negatives	86569	87555	88038
false negatives	997	1417	1790
precision (micro)	67.07 %	77.67 %	84.82 %
recall (micro)	81.01 %	73.0 %	65.9 %
fmeasure (micro)	73.38 %	75.26 %	74.17 %
precision (macro)	65.79 %	77.12 %	84.92 %
recall (macro)	72.23 %	63.38 %	56.8 %
fmeasure (macro)	67.73 %	68.03 %	65.85 %
accuracy (ML)	70.48 %	71.8 %	70.04 %
hamming loss	0.62	0.47	0.41

Tabulka F.1: Ukázka výstupu s vypočtenými metrikami (pro prahy 0,2, 0,4 a 0,6).

řádky zadá v příslušné složce pouze `fasttext.exe`, program vypíše nápovědu.

Pro natrénování např. *skip-gram* modelu pomocí *fastText* je nutné ve složce, kde je *fastText* zkompilovaný (`src/external_tools/fasttext`), zadat následující příkaz.

```
fasttext skipgram -output OUTFILE -input INFILE
```

OUTFILE je název souboru, kam budou uloženy natrénované vektory. INFILE je název souboru s textem, na kterém se má *fastText natrénovat*. Pro *Sent2vec* je příkaz podobný (ve složce, kde je *sent2vec* zkomplikovaný, tj. `src/external_tools/sent2vec`).

```
fasttext sent2vec -output OUTFILE -input INFILE
```

Pro vyzkoušení programů je ve složkách `src/external_tools/fasttext` a `src/external_tools/sent2vec` umístěn soubor `cz_train.sent`, který obsahuje po řádkách všechny věty z trénovací množiny českého korpusu. Dále se ve složce `bin/pretrained` nachází natrénované slovní vektory z *fastText* a natrénovaný *sent2vec* model.

F.2.4 Seznam vzorových skriptů

Na přiloženém DVD se ve složce `bin` nachází několik vzorových skriptů pro použití knihovny. Všechny jsou podrobně okomentovány.

- `prepare_cz.py` – Předzpracování českého korpusu. Český korpus v originální podobě (formát CoNLL-U) je umístěn ve složce `bin/data/cz`, ale nejprve je nutné jej rozbalit.
- `prepare_en.py` – Předzpracování anglického korpusu. Kvůli své velikosti se anglický korpus v původní podobě na DVD nenachází.
- `mlp_bow.py` – BOW reprezentace českých dokumentů a klasifikace pomocí MLP.
- `mlp_cross_validation.py` – BOW reprezentace českých dokumentů a klasifikace pomocí MLP. Ukázka provedení křížové validace (dělení množiny dokumentů na části a průměrování výsledků).
- `mlp_cz_doc2vec.py` – *Doc2vec* reprezentace českých dokumentů a klasifikace pomocí MLP.
- `mlp_en_doc2vec.py` – *Doc2vec* reprezentace anglických dokumentů a klasifikace pomocí MLP.
- `mlp_sent2vec_pretrained.py` – *Sent2vec* reprezentace (z předtrénovaného *Sent2vec* modelu) českých dokumentů a klasifikace pomocí MLP.
- `cnn_embedding_insert.py` – Reprezentace dokumentů jako sekvence slovníkových indexů. Do netrénovatelné *embedding* vrstvy CNN jsou vloženy vektory natrénované ve *word2vec*.
- `cnn_embedding_random.py` – Reprezentace dokumentů jako sekvence slovníkových indexů. *Embedding* vrstva CNN je inicializována náhodně.
- `cnn_hier_doc2vec.py` – Hierarchická reprezentace dokumentů (věty z *doc2vec*). Klasifikace pomocí CNN.
- `cnn_hier_word2vec.py` – Hierarchická reprezentace dokumentů (věty jako průměr slovních vektorů z *word2vec*). Klasifikace pomocí CNN.
- `cnn_word2vec_pretrained.py` – Načtení stažených předtrénovaných *word2vec* vektorů a klasifikace pomocí CNN.
- `cnn_hier_words_pretrained.py` – Hierarchická reprezentace dokumentů (věty jako průměr předtrénovaných [na českém korpusu] *fastText* vektorů). Klasifikace pomocí CNN.