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

Bakalářská práce

Automatické stahování smluvních podmínek z webových stránek

Plzeň 2021 Vojtěch Bartička

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd

Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Vojtěch BARTIČKA**
Osobní číslo:	**A18B0169P**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informatika**
Téma práce:	**Automatické stahování smluvních podmínek z webových stránek**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se s metodami pro stahování obsahu z webových stránek, včetně stránek dynamicky generovaných, a se základními metodami pro zpracování textu.
2. Navrhněte systém pro vyhledávání a automatické stahování smluvních podmínek z webových stránek.
3. Navržený systém implementujte v nejjednodušší funkční podobě a následně i v podobě sofistikovanější.
4. Změřte přesnost a úplnost vytvořených metod, proveďte vzájemné porovnání a kriticky zhodnoťte výsledky.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Ing. Ondřej Pražák**
Nové technologie pro informační společnost

Datum zadání bakalářské práce: **5. října 2020**
Termín odevzdání bakalářské práce: **6. května 2021**

L.S.

Doc. Dr. Ing. Vlasta Radová
děkanka

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

V Plzni dne 26. října 2020

Prohlášení

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

V Plzni dne 6. května 2021

Vojtěch Bartička

Poděkování

Rád bych poděkoval svému vedoucímu Ing. Ondřeji Pražákovi za ochotu a čas, který mi věnoval.

Abstract

The thesis deals with the automatic scraping of terms and conditions and privacy protection information from web pages. It's a part of a larger project which focuses on the analysis of such pages and requires a training dataset for their solution. The goal is to create an application, which will be able to automatically create this dataset. As a baseline solution, a keyword-based system is used. This system allows us to create a dataset, on which we can train a neural network, which further enhances the results. By limiting the capacity of the network and hiding the keywords the network achieves a 6 to 9% improvement in f-score and up to 71% reduction in false positivity compared to the keywords-based system.

Abstrakt

Práce se zabývá automatickým stahováním smluvních podmínek a zásad ochrany údajů z webových stránek. Je součástí většího projektu, který se zabývá analýzou obsahu těchto stránek a potřebuje pro ni trénovací dataset. Cílem je vytvořit aplikaci, která bude schopna tento dataset automaticky vytvořit. Jako základní řešení jsou použita pravidla založená na klíčových slovech. Toto řešení nám umožní získat dataset, pomocí kterého je vytvořena neuronová síť, která dále zlepšuje výsledky. Omezením kapacity sítě a zakrytím klíčových slov pak sítě dosahují zlepšení 6 až 9 % v f-míře, a až 71% snížení falešné pozitivity oproti klasifikaci pomocí klíčových slov.

Obsah

1 Úvod

Zpracování přirozeného jazyka (NLP) je perspektivní obor, který nachází uplatnění v mnoha různých oblastech. Spadají pod něj například klasifikace textových dokumentů, strojový překlad nebo identifikace klíčových slov v textu. V mnoha případech je ale potřeba rozsáhlý dataset, pomocí kterého je možné natrénovat dané řešení.

Práce je součástí projektu na Katedře informatiky a výpočetní techniky Západočeské univerzity v Plzni, který se zabývá analýzou informací o ochraně údajů a podmínek užití webových stránek. Tento projekt potřebuje dataset složený z webových stránek s informacemi o ochraně údajů a podmínkami užití, který bude v rámci projektu manuálně anotován.

Cílem práce je pro tento projekt vytvořit aplikaci, která bude schopna dataset automaticky vytvořit. Nejprve bude potřeba vytvořit program, který zajistí efektivní procházení webu. Poté se navrhne a otestuje způsob, kterým budou relevantní stránky identifikovány. Lze předpokládat, že naivní řešení založená například na klíčových slovech budou mít vysokou míru chybovosti, která se na velkém počtu stránek výrazně projeví. Manuální odstranění těchto chyb by ale bylo časově náročné. Proto budou nashromážděná data použita jako dataset, pomocí kterého bude natrénována neuronová síť, která bude výsledky zlepšovat.

Klasifikace stránek pomocí neuronové sítě bude vyžadovat anotovaný dataset. Jako anotaci použijeme výstup naivního řešení, čímž se vyhneme potřebě dataset manuálně anotovat. Za předpokladu, že chybovost naivního řešení nebude příliš vysoká, bude neuronová síť schopna kvůli převažujícímu počtu správně označených stránek klasifikovat stránky správně.

V první části práce jsou popsány teoretické podklady k procházení webu a neuronovým sítím. V druhé části textu je popsáno řešení problému a jsou zhodnoceny dosažené výsledky.

2 Stahování webových stránek

2.1 Web scraping

Pojem web scraping označuje systematické shromažďování informací z webových stránek. Tento proces lze rozdělit na dvě části, a to stažení stránky a následnou extrakci relevantních informací.

Extrahovaná data mohou být následně použita například pro monitorování cen v internetových obchodech nebo jako trénovací dataset v kontextu zpracování přirozeného jazyka[6].

Web scraping může být prováděn manuálně, nebo automaticky pomocí specializovaných programů, které označujeme jako web crawlery.

2.1.1 Web crawler

Jako web crawler označíme program, který prochází internet. Základní princip fungování je poměrně jednoduchý (Obrázek 2.1). Crawler má stanovenou množinu počátečních stránek, které slouží jako startovní bod. Při navštěvování stránek pak získává odkazy na další stránky, které zařazuje do fronty, a pokud je to potřeba, extrahuje relevantní informace.

S tím, kolik stránek se na internetu nachází, je ale žádoucí definovat nějaká pravidla, podle kterých se crawler rozhodne, zda stránku navštíví, případně v jakém pořadí stránky navštíví. Crawler se například může rozhodovat na základě počtu odkazů na určitou stránku, kdy usoudíme, že stránka, na kterou vede velké množství odkazů, bude pravděpodobně významná, nebo na základě počtu odkazů vedoucích ze stránky, kdy velký počet takových odkazů může znamenat, že se jedná o nějaký katalog stránek[1].

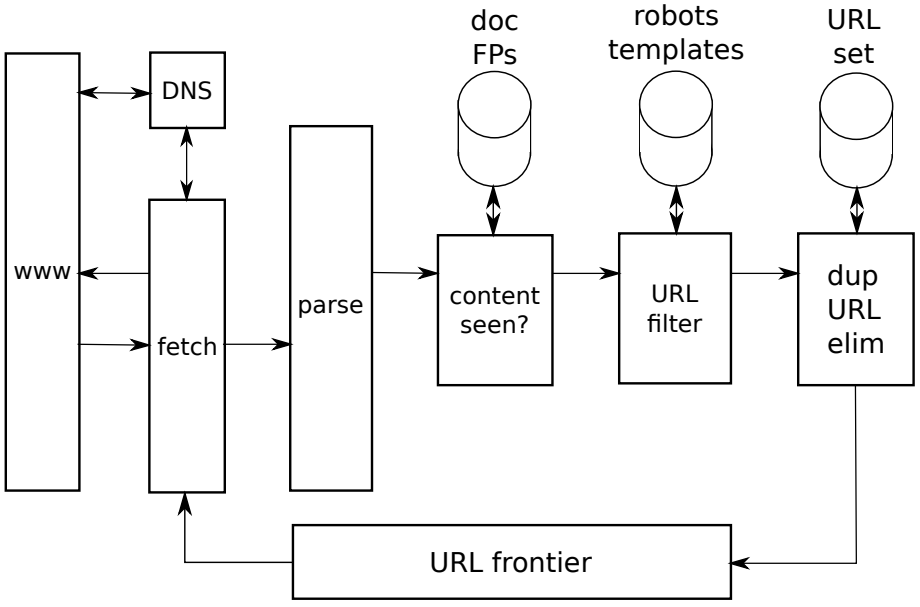
Crawlery, jejichž úkolem je shromáždit velké množství stránek, ale mají komplikovanější strukturu. Je potřeba myslet na to, že rychlost jednovláknového crawleru je limitována odezvou, a je tedy nutné crawler paralelizovat, což přináší komplikovanější implementaci. Zároveň je nutné řešit duplicitu stránek a tzv. spider traps, což jsou stránky, které obsahují nekonečné sekvence odkazů, a mohou být těžko rozlišitelné od normálních stránek.

Crawler může na cílovém serveru způsobit velkou zátěž. Proto definujeme tzv. politeness interval, což je interval mezi požadavky na doménu. Tím požadavky rozprostřeme a serveru tak nezpůsobíme nadměrnou zátěž, která by mohla komplikovat jeho provoz.

Je třeba také respektovat požadavky provozovatelů stránek co se týče

míst, která by crawler neměl prohledávat. K tomu slouží soubor robots.txt, který má známý formát. Crawler tento soubor může a nemusí respektovat[21].

V případě některých crawlerů také potřebujeme udržovat množinu stažených stránek nebo informací o nich aktuální. To je obecně problém, protože různé stránky se mění různě často. Jak často chceme informace aktualizovat závisí na našich požadavcích, ale obecně je žádoucí, aby byly informace co nejaktuálnější. Interval pro opětovné shromáždění informací můžeme určit například skrze průměrný čas mezi změnami, které detekujeme při opětovném navštěvování stránek[1].



Obrázek 2.1: Schéma crawleru. URL frontier je seznam URL k navštívení, URL filter zajišťuje, aby crawler respektoval soubor robots.txt. Obrázek z [21].

2.1.2 Identifikace relevantních údajů

Poté, co proběhne stažení stránky, je potřeba identifikovat relevantní údaje. S postupem času se tato úloha stala složitější kvůli rostoucí rozmanitosti webových stránek.

Pokud je cílem identifikovat relevantní údaje ve stránkách se známou strukturou, úloha je podstatně jednodušší. Jako řešení může posloužit DOM model pro reprezentaci stránky a například XPath pro hledání specifických objektů.

Pokud je cílem extrahovat informace ze stránek, které nemají jednotnou strukturu, stává se úloha výrazně složitější. Existují metody, které se

na základě manuálně anotovaných stránek, kde jsou vyznačeny relevantní informace, naučí tyto informace identifikovat.

Problém ale nastane v případě, že manuálně anotované stránky nejsou dostatečně reprezentativní v kontextu automaticky zpracovávaných dat. Tyto metody pak selhávají na stránkách, ve kterých se nevyskytují vzory z trénovacích dat.

Existují také metody, které nevyžadují manuální anotaci stránek. Místo toho detekují ve stránce se opakující vzory, které pak označí jako relevantní informace. Tyto metody ve srovnání s metodami, které využívají manuálně anotované stránky, podávají méně kvalitní výsledky[24].

Složitost také závisí na druhu informací, které je potřeba ze stránky extrahovat. Například v případě cen produktů v internetových obchodech lze očekávat komplikace, protože stránky nebudou mít obecně jednotnou strukturu a relevantní informace se mohou vyskytovat v různých částech stránky. Pokud je ale naším cílem ze stránek stáhnout obrázky, rozmanitost stránek nebude zásadní komplikací, protože v HTML kódu jsou obrázky jasně definované.

3 Neuronové sítě

3.1 Neurony a neuronové sítě

Ačkoliv existuje mnoho různých druhů neuronových sítí, budeme se soustředit na teorii související s tématem práce.

Neuronová síť je výpočetní systém, jehož cílem je pro určitý vstup vygenerovat výstup co nejpodobnější požadovanému výstupu. Skládá se z neuronů, které jsou mezi sebou propojeny. Jak již bylo zmíněno, neuronová síť transformuje vstup na výstup, tedy má vstupní neurony a výstupní neurony.

Neurony jsou základní stavební jednotky neuronových sítí. Neuron na základě součtu vážených vstupů $w_i x_i$ a aktivační (též označované jako transferové) funkce φ vytvoří signál na svém výstupním propojení (Vzorec 3.1 a Obrázek 3.2). Tento signál pak může sloužit jako vstup jiných neuronů, nebo jako výstup sítě.

Neurony v sítích se seskupují do vrstev, které jsou mezi sebou propojené. Vrstvené sítě mají vstupní vrstvu, výstupní vrstvu, a mohou mít libovolný počet skrytých vrstev mezi nimi (Obrázek 3.1).

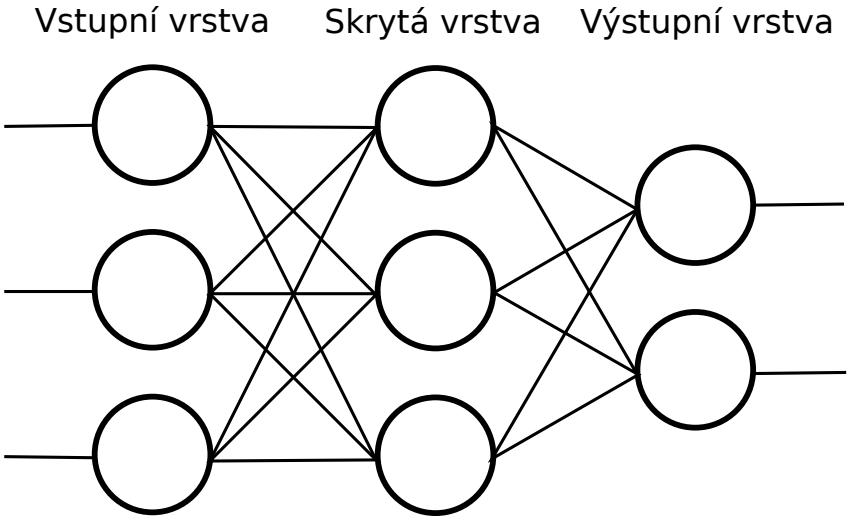
$$y = \varphi\left(\sum_{i=1}^{m} w_i x_i + b\right) \tag{3.1}$$

3.1.1 Aktivační funkce

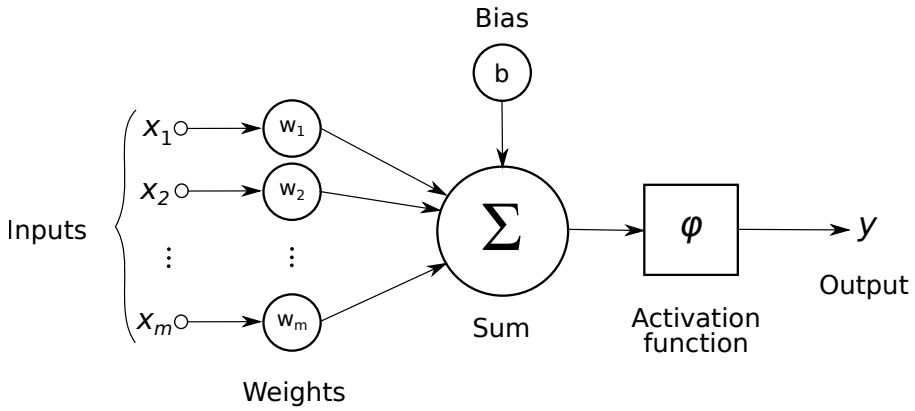
Existuje mnoho aktivačních funkcí, které se v praxi využívají. Aktivační funkce slouží k delinearizaci sítě, což jí umožňuje řešit netriviální (nelineární) problémy. Aktivační funkce ovlivňují chování neuronů a sítě jako celku, zároveň je ale potřeba vzít v potaz, jak náročný je výpočet první derivace aktivační funkce, který je potřeba pro změnu parametrů sítě při trénování.

Sigmoidní funkce

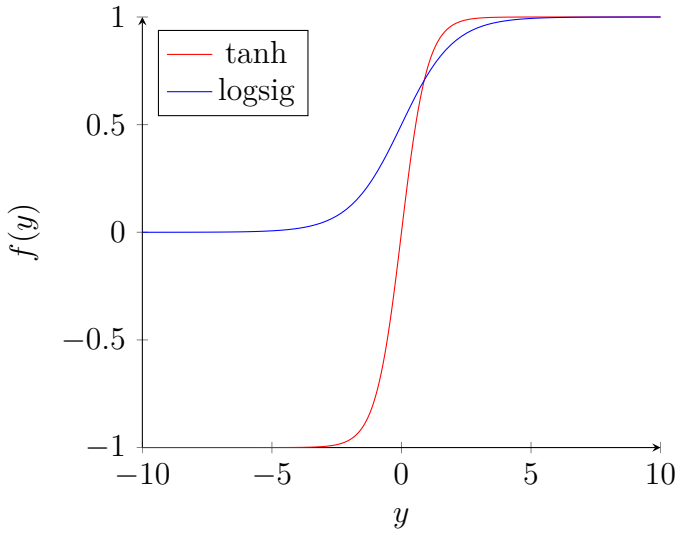
Jako sigmoidní funkci označíme funkci, která je omezená, diferencovatelná, neklesající a má právě jeden inflexní bod. Nevýhodou těchto funkcí je tzv. problém mizejícího gradientu (vanishing gradient problem), který, pokud má síť velké množství vrstev, které používají sigmoidní aktivační funkce, může způsobit neschopnost sítě se učit.



Obrázek 3.1: Schéma neuronové sítě s jednou skrytou vrstvou.



Obrázek 3.2: Schéma neuronu. Bias b se také někdy označuje jako $x_0 w_0$, kde $x_0 = 1$. Obrázek z [16].



Obrázek 3.3: Funkce tanh a logistický sigmoid (logsig).

Existuje více aktivačních funkcí, které do této kategorie spadají. Jednou z nich je například logistická sigmoidní funkce (Vzorec 3.3), nebo hyperbolický tangens (Vzorec 3.2)[12]. Tyto dvě funkce jsou zobrazeny v Obrázku 3.3.

$$y = \frac{e^{y} - e^{-y}}{e^{y} + e^{-y}} \tag{3.2}$$

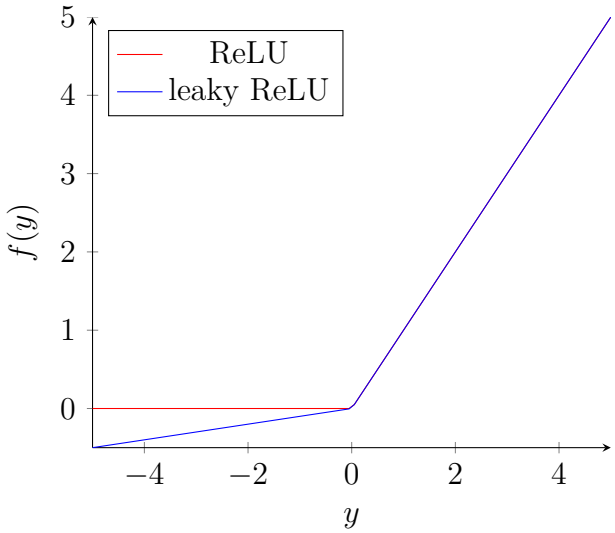
$$y = \frac{1}{1 + e^{-y}} \tag{3.3}$$

V praxi se také často používá softmax jako aktivační funkce výstupní vrstvy. Softmax umožňuje transformovat vstupy na pravděpodobnosti v nebinárních klasifikačních problémech. Je definována jako 3.4, kde se předpokládá, že výstupní vrstva bude mít tolik neuronů, kolik má řešený problém tříd. Softmax se pak počítá pro každý výstupní neuron, resp. pro každou třídu[2].

$$y_i = \frac{e^{y_i}}{\sum_j e^{y_j}} \tag{3.4}$$

Lineární funkce

Lineární aktivační funkce se skládají z několika lineárních segmentů. Jejich výhodou je nenáročnost výpočtu prvních derivací. Navíc jejich derivace nejsou nulové pro nekonečně velké vstupy, narozdíl od sigmoidních funkcí, což pomáhá eliminovat problém mizejícího gradientu[23].



Obrázek 3.4: Funkce ReLU a leaky ReLU.

Pokud bychom použili čistě lineární funkci ve smyslu 3.5 ve všech vrstvách sítě, získali bychom síť, která reprezentuje lineární funkci. Tato aktivační funkce se proto v praxi nepoužívá.

$$y = y \tag{3.5}$$

Především v konvolučních neuronových sítích se ale často používá aktivační funkce ReLU (Rectified Linear Unit)[23]. Pro záporné a nulové vstupy je výsledkem nulový výstup, zatímco kladné vstupy jsou nezměněné předány na výstup (Vzorec 3.6). Problémem může být tzv. „dying ReLU“, kdy velký počet neuronů s ReLU aktivací zůstane při trénování neaktivních (tedy na jejich výstupu bude 0).

Tento problém řeší leaky ReLU funkce, která pro záporné vstupy není rovna nule, ale záporné hodnoty jsou definovány pomalu rostoucím lineárním segmentem (Obrázek 3.4)[12].

$$y = max(0, y) \tag{3.6}$$

3.2 Učení

Učení je proces, při kterém se síť snaží uzpůsobit váhy řešenému problému tak, aby podávala co nejlepší výsledky pro daný trénovací dataset. Dva ze způsobů učení jsou učení s učitelem a bez učitele.

3.2.1 Učení s učitelem

Při učení s učitelem síti poskytneme trénovací dataset, ve kterém pro každý prvek určíme správný výsledek. Síť pak pomocí chybové funkce vypočítá chybu, resp. jak moc se odchýlila od správného výsledku. Tuto chybu pak zpětně propaguje skrz síť směrem ke vstupu, a při tom na základě spočtené chyby upravuje váhy sítě tak, aby chybu zmenšila.

Tento přístup je hojně využívaný v problémech počítačového vidění. Má ale také své nevýhody. Protože je potřeba síti dodat páry vstupu a správného výstupu, je potřeba tyto páry nějakým způsobem vytvořit, ať už manuální anotací, nebo pomocí jiného systému, který páry automaticky vytvoří. Navíc je nutná znalost všech tříd, do kterých bude síť vstupy klasifikovat. Existují problémy, u kterých není možné získat dostatečně velký anotovaný dataset nebo není přede znám počet tříd, na které nelze použít učení s učitelem[4].

3.2.2 Učení bez učitele

Při učení bez učitele se síť snaží najít ve vstupních vzorech společné charakteristiky, na základě kterých pak vstupy klasifikuje. Tento přístup má výhodu v tom, že není potřeba trénovací data anotovat, a že není potřeba předem znát počet klasifikačních tříd. Tento přístup lze využívat například v klasifikaci textu, kdy není předem znám počet tříd, do kterých chceme text klasifikovat. Sítě, které se učí bez učitele jsou například autoencoder sítě nebo samoorganizující se mapy[4].

3.2.3 Zpětná propagace

Při trénování dopředných neuronových sítí (sítí bez cyklů) je potřeba pro úpravu vah sítě spočítat gradient. Zpětná propagace je způsob výpočtu gradientu na základě chyby spočtené chybovou funkcí a vah sítě. Zpětná propagace používá řetězové pravidlo, specifické pořadí provádění numerických operací a ukládání mezivýsledků, které se v průběhu výpočtu mohou opakovat, což ji dělá efektivně implementovatelnou.

Samotný proces změny vah zajišťuje optimalizační algoritmus, nikoliv zpětná propagace, která slouží pouze pro výpočet gradientu. Optimalizační algoritmus pak na základě gradientu upravuje váhy sítě tak, aby se minimalizovala chyba[5]. Existuje mnoho optimalizačních algoritmů, z nich často používané jsou stochastický gradientní sestup a na něm založené algoritmy jako například Adam nebo Adadelta.

3.2.4 Chybové funkce

Chybová funkce se používá pro zjištění chyby výstupu neuronové sítě. V závislosti na řešeném problému je vhodné používat různé chybové funkce. Existuje mnoho chybových funkcí, jako například křížová entropie, střední kvadratická chyba, nebo také kosinová podobnost.

Křížová entropie

Pro binární klasifikační problémy i pro klasifikační problémy s více třídami lze využít křížovou entropii. Pro binární klasifikační problém je definována jako 3.7, kde t_i jsou správné výsledky a y_i jsou výstupy neuronové sítě. Je vhodné ve výstupní vrstvě používat sigmoidní aktivační funkci.

Některé knihovní implementace binární křížové entropie mohou vyžadovat, aby byl výstup sítě jediný skalár. Pak můžeme y_2 definovat jako $1 - y_1$ a t_2 jako $1 - t_1$.

$$L = -\sum_{i=1}^{2} t_i log y_i \tag{3.7}$$

Pro klasifikaci do více tříd můžeme použít kategorickou křížovou entropii. Výstupní vrstva sítě v tomto případě typicky používá aktivační funkci softmax. Kategorickou křížovou entropii pak definujeme jako 3.8, kde t_i jsou správné výsledky, y_i jsou výstupy neuronové sítě a n je počet tříd[7].

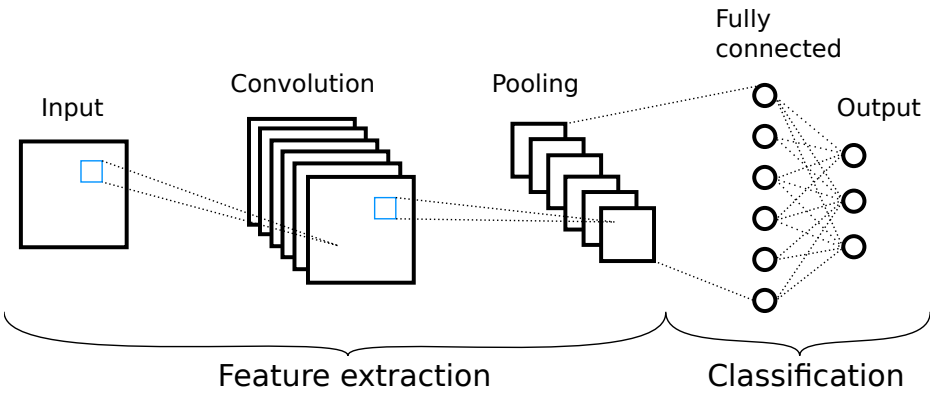
Správné vektory jsou typicky ve formě one-hot vektorů, tedy vektorů s n dimenzemi, kde jsou všechny prvky nulové až na prvek, který označuje správnou třídu. Ten má hodnotu 1. Správný výsledek ale může být také ve formě celého čísla, které označuje příslušnou třídu[22].

$$L = -\sum_{i=1}^{n} t_i log y_i \tag{3.8}$$

Kosinová podobnost

Jako chybovou funkci můžeme použít také kosinovou podobnost. Definujeme ji jako 3.9, kde Y je vektor výstupu sítě, T je správný vektor a n je počet dimenzí těchto vektorů.

$$L = \frac{\sum_{i=1}^{n} T_i Y_i}{\sqrt{\sum_{i=1}^{n} T_i}\sqrt{\sum_{i=1}^{n} Y_i}} \tag{3.9}$$



Obrázek 3.5: Ukázka konvoluční sítě. Obrázek z [19].

3.3 Konvoluční neuronové sítě

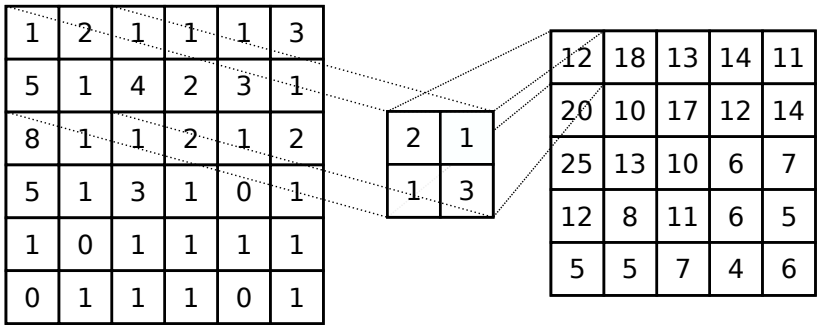
Konvoluční neuronové sítě se primárně používají pro klasifikaci obrázků, ale také pro klasifikaci textu. Skládají se ze tří druhů vrstev - konvolučních vrstev, pooling vrstev a plně propojených vrstev. Sítě mohou také obsahovat embedding vrstvy nebo dropout vrstvy. Konvoluční a pooling vrstvy lze vnímat jako způsob extrakce relevantních informací ze vstupu, zatímco plně propojené vrstvy, které jsou na výstupu sítě, slouží k interpretování extrahovaných informací (Obrázek 3.5).

Zároveň nám konvoluční neuronové sítě v některých aplikacích umožňují snížit počet trénovatelných parametrů oproti tomu, kdybychom použili pouze plně propojené vrstvy. Pokud bychom uvážili obrázek o velikosti 64x64 pixelů se třemi barevnými kanály, vstupní plně propojená vrstva by měla 12 288 trénovatelných vah. Pokud použijeme konvoluční vrstvu s jedním kernelem velikosti 6x6, vstupní vrstva bude mít 108 trénovatelných parametrů (kernel aplikujeme na každý kanál obrázku)[17].

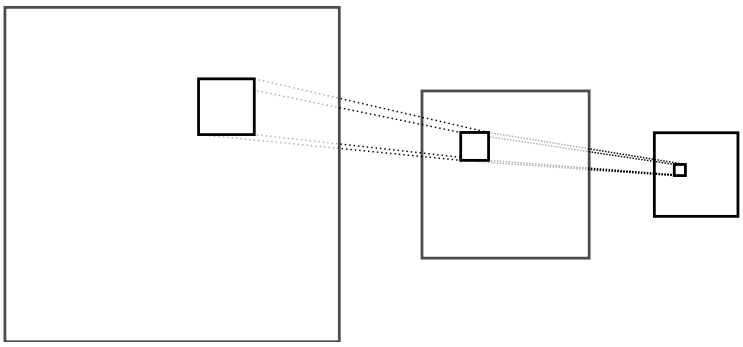
3.3.1 Konvoluční vrstva

Konvoluční vrstva provádí extrakci informací z určitého segmentu aplikováním kernelu, který slouží jako maska, na vstup. Kernel se při zpracování vstupu posouvá po jedné (např. zpracování textu) nebo více (např. zpracování obrazu) dimenzích. Při každém posunu se kernel aplikuje na vstupní data, a výsledkem toho je skalár. Kernel je matice trénovatelných vah, které se tedy v průběhu trénování upravují. Výstupem konvoluční vrstvy je tzv. aktivační mapa. Kernelů může mít konvoluční vrstva více, čímž se zvýší kapacita sítě a její rozlišovací schopnosti.

Můžeme také definovat velikost kroku pro aplikaci kernelů na vstup.



Obrázek 3.6: Ukázka konvoluce nad mapou 6x6 pomocí kernelu 2x2.



Obrázek 3.7: Ukázka řetězení konvolučních vrstev.

V Obrázku 3.6 je znázorněna aplikace kernelu 2x2 na mapu 6x6 s krokem 1, čímž získáme aktivační mapu velikosti 5x5. Pokud bychom zvětšili krok na 2, získali bychom aktivační mapu velikosti 3x3 a eliminovali překryvy mezi aplikacemi kernelů.

Tím, že konvoluční vrstva aplikuje postupně kernely na celý vstup, provádí extrakci užitečných informací. Konvoluční vrstvy se mohou řetězit (Obrázek 3.7), což umožní postupnou detekci vysokoúrovňových struktur ve vstupu. Navíc je možné konvoluční vrstvy prokládat pooling vrstvami (viz. níže)[17].

3.3.2 Pooling vrstvy

Pooling vrstvy v konvolučních neuronových sítích slouží ke zmenšení počtu dimenzí výstupu konvolučních vrstev. Pooling vrstvy mají, podobně jako konvoluční vrstvy, definovaný kernel určitých rozměrů a délku kroku. Narozdíl od konvolučních vrstev kernely nemají trénovatelné parametry, ale reprezentují matematickou funkci, typicky maximum, nebo také průměr.

V praxi se používají kernely o různých rozměrech s délkou kroku tako-

vou, aby se aplikace kernelů na vstup nepřekrývaly. Pro vstup 64x64 bude výstupem pooling vrstvy s kernelem 2x2 a krokem 2 matice o rozměrech 16x16, a dosáhne se tím zmenšení na 25 % původní velikosti[17].

3.3.3 Plně propojené vrstvy

Plně propojené vrstvy tvoří rozhodovací část sítě. Na základě informací, které extrahovaly ze vstupu konvoluční a pooling vrstvy, musí plně propojená vrstva provést samotnou klasifikaci. V síti tedy plní úlohu klasických, plně propojených neuronových sítí[17].

3.4 Metody regularizace neuronových sítí

Při trénování neuronových sítí nastává riziko, že síť bude podávat velmi dobré výsledky nad trénovacími daty, ale špatné výsledky nad testovacími daty. Tento jev se označuje jako overfitting. To může být způsobeno trénovacími daty, které nedostatečně reprezentují problém, který bude síť řešit, příliš velkým počtem epoch nebo příliš velkým modelem. Problém lze řešit použitím regularizačních metod.

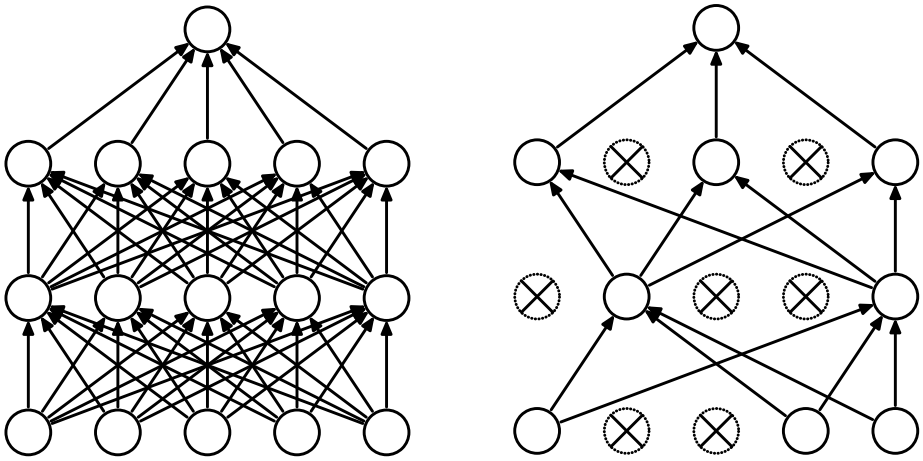
3.4.1 Early stopping

Intuitivní způsob, kterým předejít overfittingu, je zastavit trénování sítě před tím, než k němu dojde. Zastavovací podmínku můžeme definovat pomocí metrik, které se každou trénovací epochu počítají. Pokud se chyba nad testovacím datasetem několik epoch nezmenší nebo začne růst, usoudíme, že dochází k overfittingu a trénování zastavíme[11].

3.4.2 Dropout

Dropout je poměrně nová metoda regularizace, která se ukázala jako velmi efektivní a stala se často používanou. Princip spočívá v tom, že v průběhu trénování neurony s určitou pravděpodobností vypneme, respektive nastavíme jejich výstup na nulu (Obrázek 3.8). Při testování je pak výstup neuronů násoben pravděpodobností zachování neuronu.

Tím síť donutíme nespoléhat na to, že bude mít vždy informace ze všech neuronů. Ta pak musí hledat jiné spojitosti, které by jinak nehledala, protože by existovaly jednodušší alternativy. Dropout lze aplikovat jak na skryté vrstvy, tak na vstupní vrstvy. Autoři ale doporučují u dropoutu vstupní vrstvy používat menší pravděpodobnost vypnutí neuronu[20].



Obrázek 3.8: Příklad sítě před a po aplikaci dropoutu. Obrázek z [20].

3.5 Sémantická reprezentace slov

V kontextu zpracování přirozeného jazyka se pojmem reprezentace slov rozumí vektor s n dimenzemi, který slovo určitým způsobem charakterizuje. Pokud jsou vektory takové, že slova s podobným významem jsou reprezentována podobnými vektory, reprezentaci označíme jako sémantickou reprezentaci slov, tedy reprezentaci na základě jejich významu.

Sémantická reprezentace slov je založena na distribuční hypotéze z oboru lingvistiky, která byla formulována v 50. letech 20. století. Ta říká, že podobná slova mají tendenci se vyskytovat v podobných kontextech[8]. Význam slov lze tedy do určité míry odvodit od jejich kontextu.

Existuje mnoho různých metod pro vytváření sémantických reprezentací slov. Jedním z nich je například term-document matice. Ta slovo reprezentuje jako počet výskytů slova v různých dokumentech. Tento přístup podchycuje význam slov, protože slova, která se často vyskytují v podobných dokumentech, mohou mít podobný význam. Problém ale je velikost vektoru, kterým slovo reprezentujeme, protože jeho délka odpovídá počtu dokumentů.

Jako vhodné se ukázaly sémantické reprezentace využívající vektory s malým počtem dimenzí (50-1000). Menší dimenze zmenšují výpočetní nároky při jejich zpracovávání. Ve většině aplikací v rámci zpracování přirozeného jazyka podávají lepší výsledky než reprezentace využívající dlouhé, řídké vektory a navíc se ukázalo, že jsou schopny lépe vystihnout vztah mezi synonymy[10].

3.5.1 Word2vec

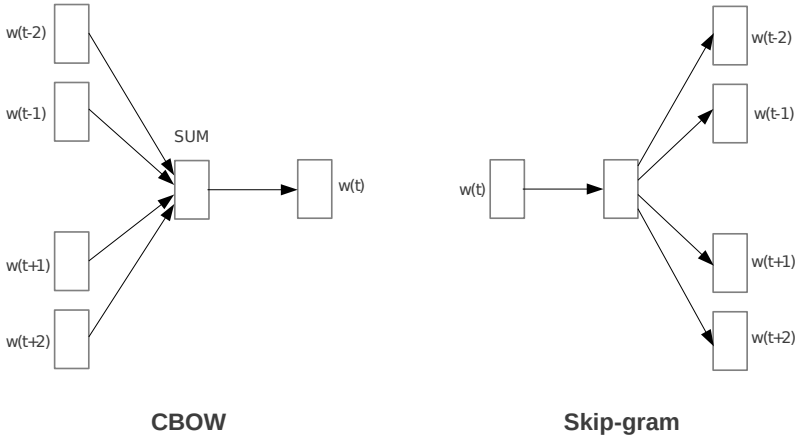
Word2vec je metoda pro vytváření sémantických reprezentací slov, která využívá neuronovou síť s jednou skrytou vrstvou. Jako vstup používá velký textový korpus, na základě kterého vytváří vektorové sémantické reprezentace, které typicky mají několik set dimenzí.

Word2vec používá dvě modelové architektury - CBOW (continuous bag of words) a continuous skip-gram. CBOW předpovídá slova na základě jejich kontextu, zatímco skip-gram předpovídá kontext na základě slov (Obrázek 3.9). Kontext je definován jako okno kolem cílového slova. Zatímco model CBOW nebere v potaz polohu slov v okně, a tedy všechna slova mají stejnou váhu, model skip-gram klade větší důraz na slova, která jsou blízko cílovému slovu.

Jako trénovací algoritmus je možné použít buďto hierarchický softmax nebo negativní vzorkování. V případě hierarchického softmaxu budeme využívat pouze slova a jejich kontext, zatímco pro negativní vzorkování navíc pro dané slovo náhodně vybereme slova, která se v jeho kontextu nevyskytují. Problém hierarchického softmaxu je jeho výpočetní náročnost, kterou lze pomocí negativního vzorkování a použití sigmoidní aktivační funkce snížit[15].

Jako vstup do vstupní vrstvy sítě pro skip-gram model použijeme cílové slovo a jako správný výstup označíme slova, v kontextu kterých se cílové slovo vyskytuje. Pro model CBOW použijeme přesný opak - jako vstup použijeme slova, která se vyskytují v kontextu cílového slova a jako správný výstup označíme cílové slovo[14].

Vektory sémantické reprezentace slov pak získáme jako matici vah sítě. Tato matice se označuje jako tzv. embedding matice.



Obrázek 3.9: Modely CBOW a skip-gram. Obrázek z [14].

3.5.2 GloVe

GloVe je metoda pro vytváření sémantických reprezentací slov. Narozdíl od word2vec nepoužívá pouze informace z lokálního kontextu (okno kolem slova), ale také celkové statistické údaje o textovém korpusu.

GloVe pracuje s maticí výskytů slov $\mathbf{X}$, kde prvek $\mathbf{X}_{ij}$ je počet výskytů slova i v kontextu slova j, kdy kontext je definován jako fixní okno kolem slova. Slova navíc vážíme podle jejich vzdálenosti, a tedy klademe menší důraz na vzdálenější slova.

Metoda se pak snaží minimalizovat výraz 3.10, kde f je váhová funkce, která zajistí, že na časté časté páry slov a zřídka se vyskytující páry slov nebude kladen příliš velký důraz, w_i a $\tilde{w}_j$ jsou vektory pro slovo, resp. kontextové slovo, a b_i a b_j jsou biasy pro slova i a j[18].

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2 \tag{3.10}$$

3.5.3 FastText

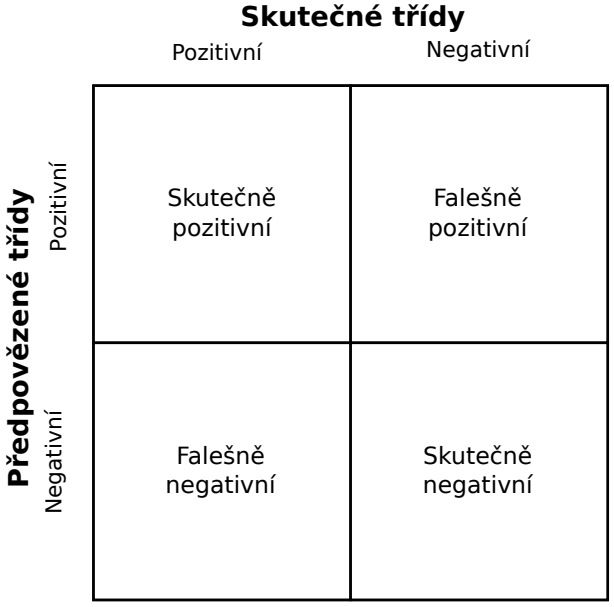
FastText se učí reprezentaci slov na základě znakových n-gramů, ze kterých se slova skládají. V principu se jedná o word2vec model, který ale funguje na úrovni jednotlivých znaků. To nám umožňuje vytvářet reprezentace pro neznámá slova na základě znakových n-gramů, ze kterých se skládají, což není s word2vec ani GloVe možné. Navíc to modelu umožňuje lépe se učit významy předpon a přípon[9][6].

4 Použité metriky

4.1 Matice záměn

Matice záměn umožňuje vizualizovat výsledky klasifikátoru. Řádky matice reprezentují předpovězené třídy a sloupce matice reprezentují reálné třídy. Na diagonále matice jsou pak počty správně klasifikovaných prvků pro dané třídy.

Matici záměn používáme, protože přesnost, úplnost a f-míra nám nedávají dostatečný přehled o chování klasifikátoru. Například z nich nejsou vidět záměny, které snižují přesnost a úplnost, ačkoliv pro nás nemusí představovat takový problém jako například falešně pozitivní klasifikace.



Obrázek 4.1: Matice záměn pro binární klasifikační problém

4.2 Přesnost (precision)

V binárních klasifikačních úlohách je přesnost (precision) procento vzorků, které klasifikátor správně označil jako pozitivní z celkového počtu pozitivně označených vzorků (Vzorec 4.1). V případě, že by měl problém n tříd, počítáme přesnosti pro každou třídu a jako celkovou přesnost označíme jejich

průměr (Vzorec 4.2).

Dílčí přesnosti můžeme spočítat z řádků matice záměn, kterou označíme **M**. Počty správně klasifikovaných vzorků jsou v matici záměn na diagonále. Dílčí přesnost třídy i spočítáme podle vzorce 4.3.

V práci je použita přesnost v tomto smyslu, a pokud je použita přesnost ve smyslu accuracy (viz. níže), je to explicitně zmíněno.

$$\text{přesnost} = \frac{|\text{Správně předpovězena pozitivní}|}{|\text{Předpovězena pozitivní}|} \tag{4.1}$$

$$\text{přesnost} = \frac{1}{n}\sum_{i=1}^{n}\frac{|\text{Správně předpovězena třída i}|}{|\text{Předpovězena třída i}|} \tag{4.2}$$

$$\text{přesnost}_i = \frac{M_{ii}}{\sum_{j=1}^{n} M_{ij}} \tag{4.3}$$

4.3 Přesnost (accuracy)

V klasifikačních úlohách je přesnost (accuracy) počet správně klasifikovaných vzorků ku celkovému počtu klasifikovaných vzorků. V práci je tato definice přesnosti označována jako „přesnost (accuracy)“.

4.4 Úplnost

V binárních klasifikačních úlohách je úplnost procento vzorků, které klasifikátor správně označil jako pozitivní, z celkového počtu vzorků, které skutečně jsou pozitivní (Vzorec 4.4). Podobně jako u přesnosti nelze tento přístup použít na nebinární problémy. Pro nebinární problémy s n třídami úplnosti spočítáme pro každou třídu a jako celkovou úplnost označíme jejich průměr podle Vzorce 4.5.

Dílčí úplnosti můžeme spočítat z řádků matice záměn, kterou označíme **M**. Počty správně klasifikovaných vzorků jsou v matici záměn na diagonále. Dílčí úplnost třídy i spočítáme podle vzorce 4.6.

$$\text{úplnost} = \frac{|\text{Správně předpovězena třída i}|}{|\text{Skutečně třída i}|} \tag{4.4}$$

$$\text{úplnost} = \frac{1}{n}\sum_{i=1}^{n}\frac{|\text{Správně předpovězena třída i}|}{|\text{Skutečně třída i}|} \tag{4.5}$$

$$\text{úplnost}_i = \frac{M_{ii}}{\sum_{j=1}^{n} M_{ji}} \tag{4.6}$$

4.5 F-skóre

F-skóre vyjadřuje vztah mezi přesností (precision) a úplností, který je definován jako jejich harmonický průměr (Vzorec 4.7). Jeho obecnou definicí je F_β-skóre (Vzorec 4.8), které umožňuje klást větší důraz na přesnost nebo úplnost. V našem případě klademe na přesnost i úplnost stejný důraz. F-skóre je počítáno z celkové přesnosti a úplnosti (Vzorce 4.2 a 4.5).

$$F_1 = 2 * \frac{\text{úplnost} * \text{přesnost}}{\text{úplnost} + \text{přesnost}} \tag{4.7}$$

$$F_\beta = (1 + \beta^2) * \frac{\text{úplnost} * \text{přesnost}}{\text{úplnost} + \beta^2 * \text{přesnost}} \tag{4.8}$$

4.6 Interval spolehlivosti

Protože neuronové sítě jsou závislé na náhodné inicializaci počátečních vah a výsledky modelů se stejnou architekturou se tak mohou lišit, je tuto skutečnost potřeba zohlednit při prezentaci výsledků. Proto použijeme pro jejich přesnost, úplnost a f-míru interval spolehlivosti pro střední hodnotu.

Interval spolehlivosti definujeme s vybranou hladinou spolehlivosti α (například 5% nebo 10%). Potom interval spolehlivosti s hladinou spolehlivosti α říká, že 1-α% měření dané veličiny bude v intervalu spolehlivosti.

Pro výpočet nejprve spočítáme průměr n naměřených hodnot μ. Poté spočítáme jejich směrodatnou odchylku σ podle vzorce 4.9. Interval pak definujeme pomocí studentova t-rozdělení se stupněm volnosti $n - 1$ podle vzorce 4.10.

$$\sigma = \sqrt{\frac{1}{n}\left(\sum_{i=1}^{n}(x_i - \mu)^2\right)} \tag{4.9}$$

$$\left(\mu_0 - t_{(1-\frac{\alpha}{2},n-1)}\frac{\sigma}{\sqrt{n}}\ ,\ \mu_0 + t_{(1-\frac{\alpha}{2},n-1)}\frac{\sigma}{\sqrt{n}}\right) \tag{4.10}$$

5 Crawler a klasifikace stránek na základě klíčových slov

5.1 Návrh

Pro shromáždění datasetu je potřeba crawler, který bude procházet internet, identifikovat relevantní stránky a stahovat je. Tento proces je potřeba optimalizovat tak, aby crawler mohl dataset shromáždit v rozumném čase. Je potřeba vyřešit dva problémy, a to stahování dynamicky načítaných stránek a zajištění efektivního procházení internetu.

Crawler bude identifikovat relevantní stránky pomocí klíčových slov. To nám umožní získat dataset, který v případě, že nebude dostatečně kvalitní, poslouží pro trénování neuronových sítí, které použijeme pro dosažení lepších výsledků.

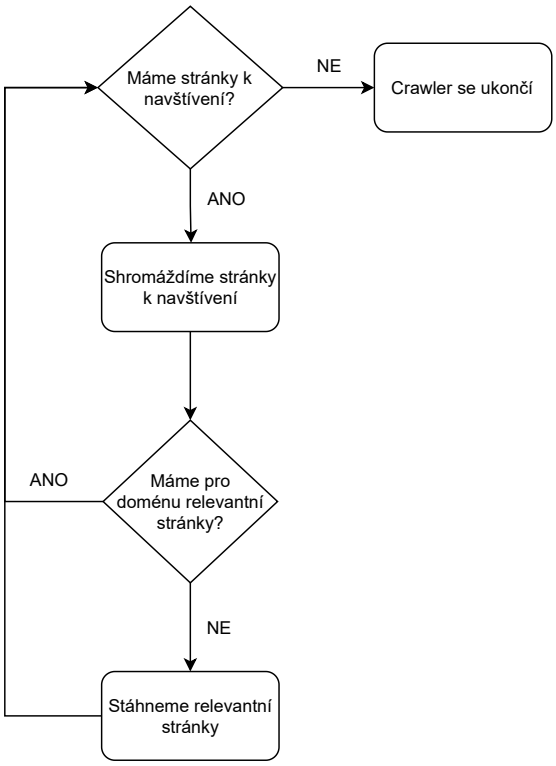
5.1.1 Princip fungování

Crawler si bude udržovat frontu stránek k navštívení a seznam navštívených stránek. Při přidávání stránek k navštívení do fronty bude kontrolovat, jestli je již nenavštívil, aby nedošlo k zacyklení. Každou stránku, kterou navštíví pak stáhne a z HTML kódu extrahuje odkazy. To zajistí knihovna Beautifulsoup, která umožňuje jednoduše vyhledat všechny odkazy na stránce.

Budeme rozlišovat tři druhy stránek - stránky obsahující informace o ochraně osobních údajů, stránky obsahující podmínky užití a ostatní (nerelevantní) stránky.

Z odkazů pak crawler vybere na základě jejich textu a adresy ty, které vedou na potenciálně relevantní stránky. K tomuto účelu budou identifikována klíčová slova, která se v odkazech vedoucích na relevantní stránky vyskytují. Takto identifikované stránky pak stáhne jako relevantní stránky, bez ohledu na jejich obsah.

Jako výchozí bod bude mít crawler seznam stránek, které obsahují velké množství odkazů na různé stránky. Díky tomu bude mít crawler hned na začátku více stránek, čímž zrychlíme stahování relevantních stránek, protože crawler bude mít přístup k více různým doménám.



Obrázek 5.1: Zjednodušené schéma fungování crawleru.

5.1.2 Dynamické načítání stránek

Crawler potřebuje stahovat obsah stránek. Jednoduchým způsobem, jak toho docílit, je například použití knihovny urllib pro Python. Ta umožňuje poslat HTTP požadavek na server a získat odpověď, která obsahuje HTML kód stránky. Tento přístup ale neřeší problém dynamického načítání stránek, které je založené například na JavaScriptovém kódu. Proto bude použit framework Selenium.

Framework Selenium je určen především pro automatizaci testování webových aplikací skrze prohlížeč. Poskytuje rozhraní, které umožňuje provádět interakci s webovým prohlížečem a obsahem stránky. Díky Seleniu můžeme ve webovém prohlížeči spustit JavaScriptový kód, který posune stránku směrem dolů, a můžeme zjistit, zda je stránka již načtena. Bude potřeba počet posunů omezit kvůli stránkám, které načítají velké až „nekonečné“ (např. seznam.cz) množství obsahu.

5.1.3 Efektivita procházení webu

Pokud by crawler navštěvoval každou stránku a hledal v ní relevantní stránky, byl by proces shromažďování stránek příliš pomalý. Protože nás ale zajímají pouze stránky, obsahující podmínky užití a informace o ochraně dat, můžeme počet navštívených stránek snížit.

Předpokládáme, že jedna doména bude mít jedny společné relevantní stránky. Crawler si udržuje seznam navštívených stránek, seznam stránek k navštívení a seznam domén, ze kterých již stáhl relevantní stránky. Stránky k navštívení crawler shromažďuje na každé stránce, ale relevantní stránky stahuje pouze v doménách, pro které ještě relevantní stránky stažené nemá (Obrázek 5.1). Tím se redukuje počet stránek, ve kterých je potřeba relevantní stránky hledat a stahovat. Implementace ale nebere v potaz subdomény. Například domény search.seznam.cz a seznam.cz jsou brány jako různé domény, i když budou mít stejné relevantní stránky. Tento problém je vyřešen ukládáním adres již stažených relevantních stránek a crawler znovu nestahuje relevantní stránky, které již stáhl. Tím se sníží zbytečná duplicita v datech a crawler tráví méně času stahováním stránek.

5.2 Implementace crawleru

Na základě návrhu byl implementován crawler. Jako výchozí body bylo vybráno 11 stránek, které obsahují odkazy na jiné stránky. Jedná se především o katalogy stránek Seznamu.

5.2.1 PDF soubory a JavaScriptové pop-upy

Jako problém se ukázaly PDF soubory a JavaScriptové pop-upy. Selenium neumožňuje specifikovat složku pro stažené soubory za běhu, takže není jednoduché spárovat stažený PDF soubor a doménu, ke které patří. Stejně tak se ukázalo jako obtížné řešit JavaSciptové pop-upy obsahující relevantní informace. S ohledem na jejich malý výskyt jsou takové případy ignorovány.

5.2.2 Efektivita procházení webu

Dále se ukázalo, že rychlost identifikování a stahování relevantních stránek není dostačující. Crawler za 8 hodin stáhl relevantní stránky z pouhých 249 domén. Problém spočíval v tom, že crawler trávil velké množství času v rozsáhlých doménách, kde dále shromažďoval odkazy k navštívení, často vedoucí na stejnou doménu. Byl proto upraven systém shromažďování stránek

k navštívení.

Nový systém při shromažďování stránek k navštívení vybere pouze kombinaci určitého počtu odkazů vedoucích na stejnou doménu, odkazů vedoucích na jinou doménu a odkazů vedoucích přímo na jinou doménu (odkazy, které končí „.cz“). Tím se zajistí, že crawleru nedojdou stránky k navštívení, ale zároveň nebude trávit v jedné doméně příliš mnoho času.

Bylo testováno pět konfigurací crawleru s dobou běhu 8 hodin. Protože crawlery neběžely najednou, výsledky jsou ovlivněny tím, na jaké stránky crawlery narazily a stabilitou internetového připojení. Přesto se ukázalo, že omezení počtu odkazů, které se zařadí do fronty, zvýší počet zpracovaných domén s relevantními stránkami (Tabulka 5.1 a Obrázek 5.2).

Konfigurace	počet domén
všechny-všechny	249
4-8	999
2-4	1 088
1-2	1 257
0.5-1	1 502

Tabulka 5.1: Počet zpracovaných domén s relevantními stránkami. Doba běhu 8 hodin. Konfigurace ve formátu počet odkazů vedoucích na stejnou doménu - počet odkazů vedoucích na jinou doménu

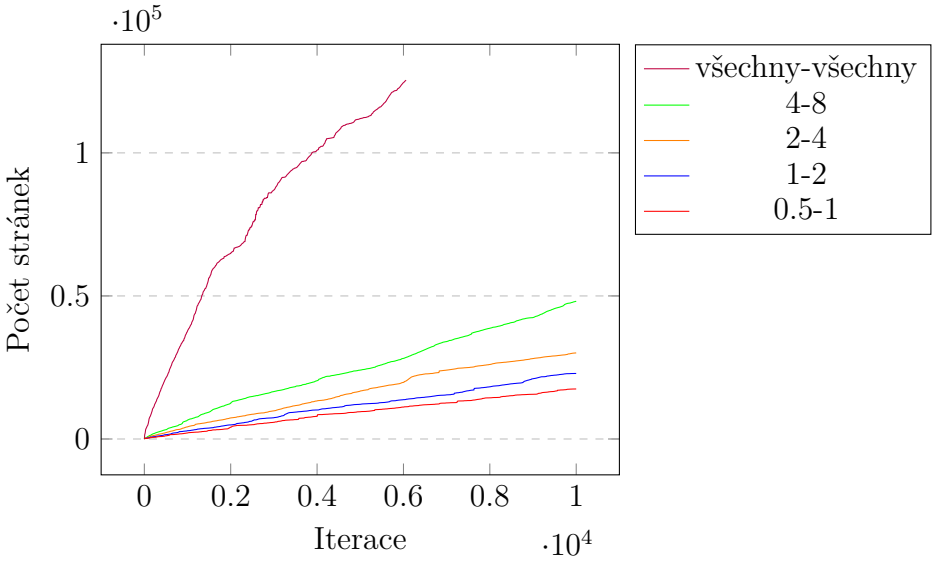
5.3 Klasifikace na základě klíčových slov

Pro klasifikaci pomocí klíčových slov bylo identifikováno celkem 25 slov (16 pro ochranu dat a 9 pro podmínky užití), která se vyskytují v adresách a textech odkazů na relevantní stránky. Tato slova jsou vypsána v Příloze A.

5.3.1 Výsledky

Přesnost klasifikace pomocí klíčových slov byla analyzována na 164 doménách, ze kterých bylo staženo 97 stránek s podmínkami a 417 stránek s informacemi o ochraně údajů. Byla analyzována i chybovost u domén, ze kterých crawler podmínky nebo informace o ochraně údajů nestáhl.

Falešná negativita se neukázala jako problém. Vyskytují se případy, kdy byly relevantní informace v JavaScriptovém pop-upu, ale jedná se o jednotky případů (Tabulka 5.2).



Obrázek 5.2: Počet stránek k navštívení ve frontě v čase. Čas byl omezen na 10 000 iterací, kde se iterací rozumí vyjmutí adresy z fronty a zpracování stránky.

Stav	Počet	Procent
Chyba detekce	3	2.2 %
JS pop-up	5	3.6 %
Na stránce nebyly	129	94.2 %

Tabulka 5.2: Analýza stránek, ve kterých nebyly nalezeny relevantní stránky.

Stav	Počet	Procent
Chyba v kl. s. „podmínky“ - obch., soutěž. a plat.	11	11.3 %
Chyba v kl. s. „podmínky“ - jiné	10	10.3 %
Chyba v kl. s. „užívání“	5	5.2 %
Chyba v kl. s. - jiná	1	1.0 %
Správně stažené	70	72.2 %

Tabulka 5.3: Analýza stránek, stažených jako stránky s podmínkami.

Jako větší problém se ukázala falešná pozitivita. Protože klíčová slova jsou poměrně obecná a používají se i v jiných kontextech, jako relevantní stránky byly identifikovány také stránky o ochraně zdraví, soutěžní podmínky, nebo reklamy na osobní automobily. Obzvlášť v případě osobních automobilů byl počet chybně stažených stránek vysoký (Tabulky 5.3 a 5.4).

Některé z těchto problémů lze řešit upravením pravidel pro klasifikaci, na-

Stav	Počet	Procent
Chyba v kl. s. „osobní“ - auto	75	17.98 %
Chyba v kl. s. „osobní“ - jiné	39	9.35 %
Chyba v kl. s. „ochrana“ - zdraví	10	2.40 %
Chyba v kl. s. „ochrana“ - jiné	10	2.40 %
Chyba v kl. s. „údaj“	10	2.40 %
Chyba v kl. s. „soukromí“	3	0.72 %
Chyba v kl. s. „zpracování“	2	0.48 %
Problém s JS	1	0.24 %
Odkaz pro odstranění cookies	1	0.24 %
Správně stažené	266	63.8 %

Tabulka 5.4: Analýza stránek stažených jako stránky s informacemi o ochraně dat

příklad podmínit klíčové slovo „osobní“ absencí slova „auto“, ale i tak klasifikace pomocí klíčových slov není dostatečně obecná na to, aby bylo možné eliminovat všechny podobné případy falešné pozitivity. Navíc se výskyt podobných, z pohledu klasifikace problematických, slovních spojení mění v čase.

Protože klíčové slovo „osobní“ způsobovalo velký počet falešně pozitivních klasifikací, bylo z klíčových slov odstraněno. Pokud se bude vyskytovat v relevantním kontextu, např. „ochrana osobních údajů“, tak takový odkaz bude zachycen pomocí jiných klíčových slov, které způsobují menší falešnou pozitivitu. Navíc bylo slovo „ochrana“ podmíněno absencí slova „zdraví“. Po aplikaci těchto úprav procento správně stažených stránek může dosáhnout až hodnoty 90,7 %.

Analýza byla provedena na relativně malém počtu stránek a neobsahuje všechny problémy, které se mohou vyskytnout, ale dává nám představu o tom, jaké chyby se budou vyskytovat v datasetu, který pomocí klasifikace na základě klíčových slov vytvoříme.

Lze tvrdit, že míra falešně pozitivity je příliš vysoká na to, aby byl dataset, vytvořený pomocí klasifikace založené na klíčových slovech, použit jako finální dataset. Bude tedy nutné přesnost klasifikace zlepšit pomocí neuronových sítí.

6 Klasifikace pomocí neuronových sítí

6.1 Dataset

Pomocí crawleru a klasifikace na základě klíčových slov byly shromážděny relevantní a nerelevantní stránky. Pro shromáždění nerelevantních stránek byl použit upravený crawler, který nerelevantní stránku identifikoval tak, že odkaz na ni neobsahoval žádné z klíčových slov. Navíc byl počet nerelevantních stránek, shromážděných z jedné domény, omezen na 3 stránky. Tím zajistíme, že v datasetu budou různorodé stránky, a omezíme nadměrnou reprezentaci některých domén.

6.1.1 Testovací a trénovací datasety

	Train	Test	Celkem
Ochrana dat	4 284	1 816	6 100
Podmínky	1 053	446	1 499
Nerelevantní	10 671	4 573	15 244
Celkem	16 008	6 835	22 843

Tabulka 6.1: Počty jednotlivých tříd v datasetu

Výsledný dataset obsahuje celkem 22 843 stránek, z toho 6 100 stránek o ochraně dat, 1 499 podmínek a 15 244 nerelevantních stránek (Tabulka 6.1). Pro trénování byl dataset rozdělen na trénovací a testovací část v poměru 70:30. Poměry tříd v testovacím a trénovacím datasetu jsou stejné.

6.1.2 Manuálně anotované ověřovací datasety

Protože klasifikace pomocí klíčových slov není dostatečně spolehlivá a nemůžeme se spolehnout na kvalitu crawlerem vytvořeného datasetu, byly navíc vytvořeny tři ověřovací datasety, které byly manuálně anotovány. Dva vývojové datasety (Tabulka 6.2), pomocí kterých byly validovány neuronové sítě v průběhu vývoje a experimentování, a jeden kontrolní dataset (Ta-

FP dataset	
Ochrana dat	57
Podmínky	34
Nerelevantní stránky	94
z toho FP	94
Celkem	192

FP+TN dataset	
Ochrana dat	57
Podmínky	34
Nerelevantní stránky	194
z toho FP	94
Celkem	292

Tabulka 6.2: Manuálně anotované vývojové datasety. V tabulkách je uveden počet nerelevantních stránek a kolik z nich crawler identifikoval jako falešně pozitivní (tedy nerelevantní stránky, které jsou v trénovacím a testovacím datasetu označeny jako relevantní). Vlevo je dataset, který obsahuje pouze nerelevantní stránky, které crawler identifikoval jako relevantní, a vpravo je dataset, který navíc obsahuje stránky, které crawler správně identifikoval jako nerelevantní.

	Podmínky	Ochrana dat	Nerelevantní
Podmínky	19	2	18
Ochrana dat	2	87	72
Nerelevantní	0	0	400
Přesnost (precision)	0.6759		
Úplnost	0.8995		
F-míra	0.7718		
Přesnost (accuracy)	0.8433		

Tabulka 6.3: Výsledky manuální anotace kontrolního datasetu. Řádky jsou predikce crawleru (klasifikace pomocí klíčových slov) a sloupce jsou skutečné třídy. Je zřejmé, že se v datech vyskytuje nezanedbatelná falešná pozitivita.

bulka 6.3), který nebyl v průběhu vývoje použit a je určen pro nezávislé zhodnocení kvality finálních modelů.

První vývojový dataset (Tabulka 6.4) obsahuje stránky, které crawler chybně identifikoval jako relevantní, a stránky, které správně identifikoval jako relevantní. V práci je označen jako „FP dataset“. Ten slouží především k ověření toho, jestli sítě neprodukují falešně pozitivní nebo falešně negativní klasifikace. Druhý vývojový dataset (Tabulka 6.5) obsahuje navíc náhodně vybrané stránky, které byly na základě klíčových slov označeny jako nerelevantní. V práci je označen jako „FP+TN dataset“. Ten dává ucelenější přehled o chování sítě.

Kontrolní dataset byl vytvořen náhodným výběrem 600 stránek. Celkem bylo náhodně vybráno 400 stránek, které crawler správně označil jako nerelevantní, 39 stránek, které crawler označil jako podmínky a 161 stránek, které crawler označil jako ochranu dat. Tyto počty poměrově odpovídají testovacím a trénovacím datům. Výsledky manuální anotace datasetu jsou v matici záměn v Tabulce 6.3. Na tomto datasetu byly validovány pouze finální modely v sekci Výsledky.

	Podmínky	Ochrana dat	Nerelevantní
Podmínky	32	3	30
Ochrana dat	2	54	64
Nerelevantní	0	0	0
Přesnost	0.3141		
Úplnost	0.6295		
F-míra	0.4190		
Přesnost (accuracy)	0.4649		

Tabulka 6.4: Matice záměn a metriky klasifikace pomocí klíčových slov pro vývojový dataset FP.

	Podmínky	Ochrana dat	Nerelevantní
Podmínky	32	3	30
Ochrana dat	2	54	64
Nerelevantní	0	0	100
Přesnost	0.6474		
Úplnost	0.8013		
F-míra	0.7162		
Přesnost (accuracy)	0.6526		

Tabulka 6.5: Matice záměn a metriky klasifikace pomocí klíčových slov pro vývojový dataset FP+TN

6.2 Návrh architektury

V potaz připadalo více architektur, které se používají pro zpracování přirozeného jazyka. Kvůli tomu, že síť bude zpracovávat potenciálně dlouhé dokumenty, bylo rozhodnuto ve prospěch použití konvolučních neuronových sítí.

Ty nemají problém s dlouhodobou memorizací jako například LSTM sítě, které v jejich základním provedení mají problémy se zpracováním dlouhých dokumentů[13]. Konvoluční sítě lze navíc jednoduše implementovat pomocí knihovny TensorFlow. Budeme používat standardní architekturu konvolučních neuronových sítí, tedy konvoluční vrstvy prokládané pooling vrstvami a plně propojené vrstvy u výstupu sítě. Třídy budeme reprezentovat pomocí one-hot vektorů a pro výpočet chyby budeme používat kategorickou křížovou entropii. Jako optimalizátor zvolíme Adam.

Pro reprezentaci textu byly zvoleny sémantické word embeddingy. Tento přístup má výhodu v tom, že se nefixujeme na syntaxi slov, ale podchycujeme jejich sémantiku (například slova „údaje“ a „informace“ budou mít podobné vektorové reprezentace, protože se vyskytují v podobném kontextu).

Protože anotace datasetu, i kdyby jen stránek, označených jako relevantní, by byla časově náročná, bylo rozhodnuto trénovat dataset na neanotovaném datasetu. Předpokládáme, že klasifikace pomocí klíčových slov vytvořila více správných než chybných klasifikací. Tento předpoklad potvrdila provedená analýza v sekci 5.1, ale analýza provedená na kontrolním datasetu v sekci 6.2 naznačuje, že by falešná pozitivita mohla být větší.

Idea je taková, že pokud je počet falešně pozitivních stránek v datasetu menší než počet správně klasifikovaných, trénované sítě budou schopny se naučit rozlišovat falešně pozitivní stránky od správně klasifikovaných stránek i přes falešnou pozitivitu v datasetu.

6.3 Předzpracování stránek

Při předzpracování stránek se postupně vytváří slovník, který obsahuje slova a jejich frekvence. Slova, která se vyskytují s frekvencí menší než 15, jsou ze slovníku odstraněna. Stejně tak jsou odstraněna nevýznamová slova [3]. Při zpracování se navíc kontroluje duplicita stránek.

Maximální délka dokumentu byla omezena na 10 000 slov. Pro padding je použit nulový vektor. Padding a zkracování se provádí na konci dokumentu.

6.4 Word Embedding

Pro embedding slov byly použity předtrénované vektory FastText pro český jazyk. Hlavním důvodem byla přímá dostupnost českých embedding vektorů. Protože je FastText postavený na znakových n-gramech, umožňuje vytvářet vektory pro neznámá slova na základě znaků, ze kterých se skládají. Pro slova, která nejsou v předtrénovaných vektorech, jsou vektory pomocí Fast-

Textu vytvořeny. Navíc FastText dobře reprezentuje syntaktickou podobnost slov.

6.5 Intervaly spolehlivosti

V rámci testování architektur neuronových sítí bylo pro každou architekturu trénováno pět modelů. Interval spolehlivosti byl pak vypočten s hladinou spolehlivosti $\alpha = 5\%$. Vhodnější by bylo zvolit větší počet vzorků, které by nám poskytly více reprezentativní interval, ale kvůli časové náročnosti trénování modelů bylo nutné počet vzorků snížit na pět.

6.6 Základní model

Jako výchozí model byl zvolen jednoduchý sekvenční model se dvěma konvolučními vrstvami (viz Tabulka 6.6). Až na výstupní vrstvu byly váhy inicializovány pomocí he_normal inicializace. Rychlost učení byla ponechána na hodnotě 0.001, tedy výchozí hodnotě knihovny TensorFlow.

Vrstva	Parametry
Embedding	-
Conv1D	filters=32, kernel_size=4, init=he_normal
MaxPool1D	pool_size=3
Activation	activation=relu
Conv1D	fiters=32, kernel_size=8, init=he_normal
MaxPool1D	pool_size=8
Activation	activation=relu
Flatten	-
Dense	neurons=32, activation=relu, init=he_normal
Dense	neurons=3, activation=softmax, init=glorot
Parametry učení	batch_size=32, learning_rate=0.001

Tabulka 6.6: Základní model

Výsledky nepřináší žádné zásadní zlepšení oproti klasifikaci pomocí klíčových slov (Tabulka 6.7). Model byl schopen správně identifikovat některé falešně pozitivní stránky, ale falešná pozitivita je stále příliš vysoká. Pokud ale porovnáme matice záměn vývojových datasetů (Tabulky 6.8 a 6.9), můžeme vidět, že crawlerem správně stažené nerelevantní stránky síť klasifikuje správně.

FP dataset	
Přesnost	0.696 ± 0.001
Úplnost	0.708 ± 0.023
F-míra	0.702 ± 0.015

FP+TN dataset	
Přesnost	0.691 ± 0.022
Úplnost	0.834 ± 0.018
F-míra	0.755 ± 0.019

Tabulka 6.7: Metriky základního modelu s klíčovými slovy. Trénování proběhlo v osmi epochách.

	Podmínky	Ochrana dat	Nerelevantní
Podmínky	30	3	18
Ochrana dat	0	54	53
Nerelevantní	0	0	23

Tabulka 6.8: Matice záměn modelu s klíčovými slovy pro FP dataset

Z analýzy stránek, které síť klasifikovala špatně, se zdálo, že se příliš fixuje na klíčová slova. Stránky chybně klasifikované jako ochrana dat často obsahují slova jako „údaje“ a „ochrana“. U podmínek jsou to pak slova „podmínky“ a „užívání“. To je očekávatelný problém, protože všechny relevantní stránky crawler stáhl na základě klíčových slov v odkazech. Lze tedy předpokládat, že se tato klíčová slova budou vyskytovat i v textu stránek, a síť se na ně bude zaměřovat.

Jako pokus o předejití tomuto problému byla ze slovníku odstraněna všechna klíčová slova, včetně jejich pádů a časů, a s takto upraveným slovníkem byla trénována síť se stejnou architekturou. To nepřineslo žádné zlepšení (Tabulka 6.10). Vyskytl se stejný problém, tedy pravděpodobná fixace na klíčová slova. To lze vysvětlit použitím sémantických embeddingů pro reprezentaci slov. Protože embeddingy podchycují sémantiku slova na základě slov, v jejichž okolí se vyskytuje, je pravděpodobné, že v okolí slova „ochrana“ se budou vyskytovat slova s podobnou sémantickou reprezentací. Síť se tak přestane fixovat na klíčová slova, ale začne se fixovat na jejich okolí, které je podobné.

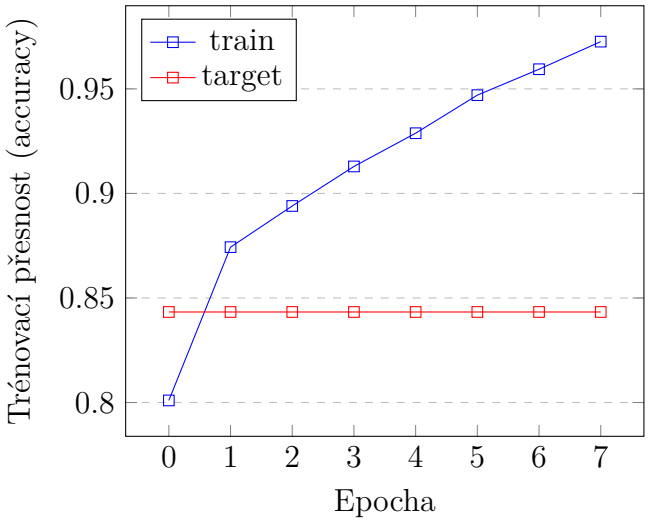
	Podmínky	Ochrana dat	Nerelevantní
Podmínky	30	3	19
Ochrana dat	0	54	55
Nerelevantní	0	0	120

Tabulka 6.9: Matice záměn modelu s klíčovými slovy pro FP+TN dataset

FP dataset	
Přesnost	0.663 ± 0.0260
Úplnost	0.677 ± 0.0226
F-míra	0.670 ± 0.0239

FP+TN dataset	
Přesnost	0.680 ± 0.0236
Úplnost	0.812 ± 0.0186
F-míra	0.740 ± 0.0212

Tabulka 6.10: Metriky základního modelu bez klíčových slov. Trénování proběhlo v osmi epochách.



Obrázek 6.1: Průběh trénování základního modelu s klíčovými slovy. Přesnost je ve smyslu accuracy a červeně je vyznačena cílová přesnost podle kontrolního datasetu.

Tento problém může být také způsoben vyšší než očekávanou falešnou pozitivitou klasifikace pomocí klíčových slov. V datasetu by pak nebyl dostatečně velký počet správně klasifikovaných stránek na to, aby se síť naučila je rozpoznávat od nesprávně klasifikovaných.

Z průběhu trénování (Obrázek 6.1) je ale vidět, že trénovací přesnost jednoho z modelů přesahuje 95 %. To je s ohledem na předpokládanou chybovost v datasetu velmi vysoké číslo. Podle metrik z Tabulky 6.3 bychom se měli za předpokladu, že je kontrolní dataset dostatečně reprezentativní, dostat na přesnost (accuracy) okolo 84 %. Toho se můžeme pokusit dosáhnout snížením počtu epoch.

Pro ověření této hypotézy byl stejný model s i bez klíčových slov trénován 2, 4 a 6 epoch (Tabulky 6.11 a 6.12). Výsledky hypotézu částečně potvrdily. U modelů s klíčovými i bez klíčových slov bylo pozorováno podobné chování.

Stabilita výsledků se zmenšila, což je pochopitelné, protože jsme sítím nedali dostatek času na to, aby se jejich váhy stabilizovaly a inicializace tak

	8 epoch	6 epoch	4 epochy	2 epochy
FP dataset				
Přesnost	0.696 ± 0.001	0.664 ± 0.025	0.693 ± 0.020	0.754 ± 0.019
Úplnost	0.708 ± 0.023	0.690 ± 0.030	0.697 ± 0.016	0.725 ± 0.047
F-míra	0.702 ± 0.015	0.677 ± 0.024	0.695 ± 0.015	0.739 ± 0.028
FP+TN dataset				
Přesnost	0.691 ± 0.022	0.696 ± 0.035	0.729 ± 0.040	0.777 ± 0.028
Úplnost	0.834 ± 0.018	0.823 ± 0.023	0.817 ± 0.006	0.810 ± 0.040
F-míra	0.755 ± 0.019	0.754 ± 0.029	0.770 ± 0.021	0.793 ± 0.015
Přesnost (accuracy)	98.158	96.456	92.602	88.502

Tabulka 6.11: Metriky modelu s klíčovými slovy pro různé počty epoch. Přesnost (accuracy) je průměr všech běhů nad trénovacím datasetem.

	8 epoch	6 epoch	4 epochy	2 epochy
FP dataset				
Přesnost	0.663 ± 0.026	0.690 ± 0.043	0.697 ± 0.050	0.753 ± 0.011
Úplnost	0.677 ± 0.023	0.696 ± 0.043	0.692 ± 0.063	0.706 ± 0.039
F-míra	0.670 ± 0.024	0.693 ± 0.041	0.694 ± 0.054	0.728 ± 0.017
FP+TN dataset				
Přesnost	0.680 ± 0.024	0.715 ± 0.049	0.726 ± 0.039	0.770 ± 0.026
Úplnost	0.812 ± 0.019	0.807 ± 0.024	0.790 ± 0.047	0.783 ± 0.025
F-míra	0.740 ± 0.021	0.758 ± 0.035	0.756 ± 0.040	0.776 ± 0.012
Přesnost (accuracy)	97.274	95.066	91.360	86.602

Tabulka 6.12: Metriky modelu bez klíčových slov pro různé počty epoch. Přesnost (accuracy) je průměr všech běhů nad trénovacím datasetem.

má větší vliv na výsledky. Trénování v šesti a čtyřech epochách nepřineslo zásadní zlepšení. Modely s klíčovými slovy v šesti epochách měly horší výsledky v porovnání s osmi epochami, ale tento rozdíl se ztratil u čtyř epoch. Modely bez klíčových slov vykazovaly zlepšení s každým snížením počtu epoch.

Jako nejlepší se ukázalo trénování modelů ve dvou epochách, kdy modely dosáhly výrazně lepších výsledků, než při trénování v osmi epochách. To je pochopitelné, protože podle Tabulek 6.11 a 6.12 modely po dvou epochách dosahují přesnosti kolem 87 %, což je cílovým 84 % relativně blízko. Modely, které byly schopné správně detekovat falešně pozitivní stránky toho dosahovaly za cenu většího počtu záměn a falešně negativních klasifikací (Tabulka 6.13).

	Podmínky	Ochrana dat	Nerelevantní
Podmínky	22	1	2
Ochrana dat	3	50	28
Nerelevantní	9	5	65

	Podmínky	Ochrana dat	Nerelevantní
Podmínky	26	2	3
Ochrana dat	4	55	53
Nerelevantní	4	0	38

Tabulka 6.13: Matice záměn základních modelů s klíčovými slovy trénovaných 2 epochy pro FP dataset. První matice patří modelu, který je schopen lépe detekovat falešně pozitivní stránky. Druhá matice patří modelu, který toho není schopen do takové míry jako první model.

Byly testovány i další modely s upravenou velikostí kernelů a poolů. Změna parametrů ale neměla větší vliv na chování sítí a výsledky.

6.7 Model s dropout vrstvami

Problémy nedostatečné obecnosti sítí řeší dropout vrstvy, které vypínáním neuronů nutí síť hledat jiné souvislosti a přispívají tak ke zlepšení chování sítě nad reálnými daty. Při trénování základního modelu se ukázalo, že při větším počtu epoch se síť pravděpodobně fixuje na klíčová slova. Proto jsme se pokusili přidáním dropout vrstev do architektury sítě tuto fixaci potlačit.

Mezi plně propojené vrstvy základního modelu byly přidány dropout vrstvy (Tabulka 6.14). Příliš velká pravděpodobnost vypnutí neuronu způsobí neschopnost sítě se učit, zatímco příliš malá pravděpodobnost se nemusí projevit. Proto bylo testováno několik hodnot v rozmezí 0.2 a 0.5.

Pro obě dropout vrstvy byla použita stejná pravděpodobnost. Ačkoliv by bylo vhodnější otestovat všechny kombinace dropoutů a získat tím ucelenější přehled o vlivu dropout vrstev na výsledky, s ohledem na časovou náročnost trénování modelů bylo testování zjednodušeno. Všechny modely byly nejprve trénovány šest epoch. Z těchto modelů byl vybrán nejlepší, u kterého poté bylo ověřeno chování s různými počty trénovacích epoch. Ostatní modely (dropout pravděpodobnosti) byly pak dotestovány na tom počtu epoch, který podal nejlepší výsledky.

Protože se při testování základního modelu ukázalo, že zakrytí klíčových slov nepřináší žádné zlepšení ve výsledcích, byly modely testovány s klíčo-

Vrstva	Parametry
Embedding	-
Conv1D	filters=32, kernel_size=4, init=he_normal
MaxPool1D	pool_size=3
Activation	activation=relu
Conv1D	fiters=32, kernel_size=8, init=he_normal
MaxPool1D	pool_size=8
Activation	activation=relu
Flatten	-
Dropout	dropout_prob={0.2, 0.3, 0.4, 0.5}
Dense	neurons=32, activation=relu, init=he_normal
Dropout	dropout_prob={0.2, 0.3, 0.4, 0.5}
Dense	neurons=3, activation=softmax, init=glorot
Parametry učení	batch_size=32, learning_rate=0.001, keywords=yes epochs=8

Tabulka 6.14: Výchozí model s dropout vrstvami mezi plně propojenými vrstvami

vými slovy.

Testování různých dropout pravděpodobností ukázalo, že hodnota 0.4 poskytuje nejlepší výsledky (Tabulka 6.15). Protože jsme při testování základního modelu dosáhli zlepšení snížením počtu trénovacích epoch, byl model s dropoutem 0.4 testován na nižších počtech epoch (Tabulka 6.16). Jako nejlepší vyšly modely, které byly trénované pět epoch.

Poté byly v pěti epochách trénovány i modely s jinými hodnotami dropoutu (Tabulka 6.17). Potvrdilo se, že hodnota dropoutu 0.4 podává nejlepší výsledky v porovnání s jinými hodnotami.

Problémem používání dropoutu je způsobená nestabilita výsledků. Protože modely jsou trénovány malý počet epoch, jsou více závislé na počáteční inicializaci vah. Dropout vrstvy do sítě pak přidávají další náhodný element, který stabilitu dále zhoršuje.

	dropout 0.5	dropout 0.4	dropout 0.3	dropout 0.2
FP dataset				
Přesnost	0.708 ± 0.010	0.698 ± 0.028	0.698 ± 0.023	0.694 ± 0.025
Úplnost	0.694 ± 0.062	0.733 ± 0.027	0.710 ± 0.030	0.721 ± 0.015
F-míra	0.700 ± 0.034	0.715 ± 0.021	0.704 ± 0.023	0.707 ± 0.020
FP+TN dataset				
Přesnost	0.723 ± 0.028	0.718 ± 0.029	0.718 ± 0.015	0.719 ± 0.021
Úplnost	0.804 ± 0.050	0.838 ± 0.014	0.831 ± 0.025	0.825 ± 0.011
F-míra	0.761 ± 0.028	0.773 ± 0.015	0.770 ± 0.014	0.768 ± 0.009
Přesnost (accuracy)	90.222	91.834	92.828	92.876

Tabulka 6.15: Metriky modelů s různými dropouty. Modely byly testovány s klíčovými slovy a v šesti epochách. Přesnost (accuracy) je průměr všech běhů nad trénovacím datasetem.

	6 epoch	5 epoch	4 epochy	3 epochy	2 epochy
FP dataset					
Přesnost	0.698 ± 0.023	0.725 ± 0.037	0.723 ± 0.017	0.737 ± 0.009	0.729 ± 0.012
Úplnost	0.710 ± 0.030	0.735 ± 0.036	0.714 ± 0.048	0.706 ± 0.023	0.688 ± 0.028
F-míra	0.704 ± 0.023	0.730 ± 0.031	0.718 ± 0.032	0.721 ± 0.012	0.709 ± 0.018
FP+TN dataset					
Přesnost	0.718 ± 0.015	0.740 ± 0.034	0.751 ± 0.017	0.754 ± 0.014	0.760 ± 0.017
Úplnost	0.831 ± 0.025	0.825 ± 0.034	0.806 ± 0.043	0.794 ± 0.029	0.776 ± 0.024
F-míra	0.770 ± 0.014	0.779 ± 0.011	0.778 ± 0.025	0.773 ± 0.011	0.768 ± 0.009
Přesnost (accuracy)	91.834	90.912	89.048	87.626	85.994

Tabulka 6.16: Metriky modelů s dropoutem 0.4 pro různé počty epoch. Modely byly testovány s klíčovými slovy. Přesnost (accuracy) je průměr všech běhů nad trénovacím datasetem.

6.8 Zmenšení modelu

Základní model bez dropout vrstev při trénování v osmi epochách dosahoval přesností přes 95 %. Tato přesnost je s ohledem na pravděpodobnou chybovost klasifikace pomocí klíčových slov příliš vysoká. Ukázalo se, že je tento problém možné kompenzovat snížením počtu epoch při trénování. Takový přístup ale nedává stabilní výsledky, protože váhy sítí nemají dost času na to se ustálit. Přidání dropout vrstev výsledky zlepšilo, ale jako problém stále přetrvává jejich nestabilita. Tu by mohlo vyřešit zmenšení počtu trénovatelných parametrů sítě.

Pokud bude mít síť málo trénovatelných parametrů, bude schopna se soustředit pouze na některé struktury ve stránkách. Pokud síti nezakryjeme klíčová slova, je pravděpodobné, že bude nucena se na ně se svou omeze-

	dropout 0.5	dropout 0.4	dropout 0.3	dropout 0.2
FP dataset				
Přesnost	0.725 ± 0.028	0.725 ± 0.037	0.711 ± 0.027	0.701 ± 0.032
Úplnost	0.710 ± 0.019	0.735 ± 0.036	0.726 ± 0.045	0.705 ± 0.048
F-míra	0.717 ± 0.013	0.730 ± 0.031	0.718 ± 0.034	0.703 ± 0.036
FP+TN dataset				
Přesnost	0.746 ± 0.034	0.740 ± 0.034	0.728 ± 0.031	0.731 ± 0.033
Úplnost	0.807 ± 0.025	0.825 ± 0.034	0.832 ± 0.031	0.823 ± 0.015
F-míra	0.774 ± 0.016	0.779 ± 0.011	0.776 ± 0.026	0.734 ± 0.019
Přesnost (accuracy)	88.560	90.912	91.532	92.640

Tabulka 6.17: Metriky modelů s s různými dropouty pro 5 epoch. Modely byly testovány s klíčovými slovy. Přesnost (accuracy) je průměr všech běhů nad trénovacím datasetem.

nou kapacitou fixovat, protože klíčová slova budou nejjednodušší způsob, jakým dosáhnout vysoké přesnosti. Pokud síti klíčová slova zakryjeme, mohli bychom síť donutit soustředit svou omezenou kapacitu na jiné vlastnosti stránek a tím dosáhnout lepších výsledků. Tento přístup by nám navíc mohl umožnit trénování ve více epochách, což by mělo pomoct se stabilitou výsledků.

Jako výchozí bod jsme zvolili základní model a počet parametrů jsme upravovali pomocí velikosti poolů pooling vrstev, počtu filtrů konvolučních vrstev a velikosti předposlední plně propojené vrstvy. Navíc byly testovány sítě, kde byla poslední pooling vrstva nahrazena globální pooling vrstvou, což vedlo k výraznému snížení počtu trénovatelných parametrů. Sítě byly trénovány v osmi epochách bez klíčových slov.

Tento přístup výsledky nijak zásadně nezlepšil (Tabulka 6.18). I po velmi radikálním zmenšení kapacity sítě výsledky nebyly podstatně lepší než při použití základního modelu. Zmenšení sítě na síť s 5 019 trénovatelnými parametry pak vyústilo v nepoužitelnou síť, která nebyla schopna eliminovat falešnou pozitivitu a navíc ignorovala třídu podmínek. Tento jev lze vysvětlit tím, že malá kapacita sítě způsobila její soustředění na zbylé dvě početnější třídy. Stránek, klasifikovaných pomocí klíčových slov jako ochrana dat, je v datasetu přibližně čtyřikrát více než podmínek, a nerelevantních stránek je více přibližně patnáctkrát více.

Problém by mohlo vyřešit použití dropoutu po embedding vrstvě. Podobně jako v sekci 6.6 se nám síť pravděpodobně fixuje na kontext klíčových slov. Pokud při trénování náhodně zakryjeme síti některé embeddingy, mohli bychom jí zabránit ve fixaci na kontext klíčových slov.

Parametrů	258 775	127 395	79 451	37 623
FP dataset				
Přesnost	0.662 ± 0.055	0.688 ± 0.021	0.682 ± 0.040	0.629 ± 0.087
Úplnost	0.660 ± 0.017	0.686 ± 0.061	0.682 ± 0.046	0.659 ± 0.075
F-míra	0.661 ± 0.033	0.685 ± 0.029	0.682 ± 0.039	0.644 ± 0.081
FP+TN dataset				
Přesnost	0.690 ± 0.039	0.706 ± 0.021	0.707 ± 0.032	0.679 ± 0.025
Úplnost	0.798 ± 0.036	0.800 ± 0.053	0.790 ± 0.031	0.813 ± 0.019
F-míra	0.739 ± 0.025	0.749 ± 0.019	0.746 ± 0.028	0.740 ± 0.018
Přesnost (accuracy)	96.932	95.338	93.884	97.268

Parametrů	16 527	8 963	7 603	5 019
FP dataset				
Přesnost	0.715 ± 0.031	0.729 ± 0.018	0.718 ± 0.034	0.600 ± 0.167
Úplnost	0.702 ± 0.030	0.692 ± 0.037	0.669 ± 0.050	0.568 ± 0.130
F-míra	0.708 ± 0.026	0.710 ± 0.025	0.692 ± 0.033	0.583 ± 0.146
FP+TN dataset				
Přesnost	0.723 ± 0.035	0.739 ± 0.016	0.730 ± 0.034	0.612 ± 0.168
Úplnost	0.815 ± 0.021	0.793 ± 0.029	0.757 ± 0.058	0.654 ± 0.127
F-míra	0.766 ± 0.029	0.765 ± 0.017	0.742 ± 0.025	0.631 ± 0.147
Přesnost (accuracy)	94.934	92.076	90.816	88.732

Tabulka 6.18: Výsledky testování modelů s menším počtem trénovatelných parametrů. Testování proběhlo bez klíčových slov v osmi epochách. Přesnost (accuracy) je průměr všech běhů nad trénovacím datasetem.

Jako základ pro testování byla vybrána síť s 16 527 trénovatelnými parametry (Tabulka 6.19), která byla nejlepším kompromisem výsledků a stability. Do té byla přidána po embedding vrstvě dropout vrstva, která s určitou pravděpodobností nahradí embedding vektor slova nulovým vektorem. Síť pak opět byla trénována osm epoch.

Úvaha se ukázala jako správná. Modely trénované s dropoutem embedding vektorů podaly nejlepší výsledky ze všech trénovaných modelů (Tabulka 6.20). Použití vyšších pravděpodobností dropoutu se ukázalo jako nejlepší, kdy se výsledky na FP datasetu pohybovaly kolem f-míry 79 %. Ačkoliv jsou výsledky vlivem dropoutu nestabilní, s ohledem na to, jak dobré jsou, je jejich nestabilita přijatelná. Interval spolehlivosti modelu s dropoutem 0.5 je spíše anomálie, protože je menší než u modelů s nižším dropoutem, které by měly být stabilnější.

Vrstva	Parametry
Embedding	-
Dropout	dropout_prob={0.6, 0.5, 0.4, 0.3}
Conv1D	filters=12, kernel_size=4, init=he_normal
MaxPool1D	pool_size=6
Activation	activation=relu
Conv1D	fiters=16, kernel_size=8, init=he_normal
GlobalPool1D	-
Activation	activation=relu
Flatten	-
Dense	neurons=28, activation=relu, init=he_normal
Dense	neurons=3, activation=softmax, init=glorot
Parametry učení	batch_size=32, learning_rate=0.001, keywords=no epochs=8

Tabulka 6.19: Model s 16 527 trénovatelnými parametry s dropout vrstvou po embedding vrstvě.

Dropout	0.6	0.5	0.4	0.3
FP dataset				
Přesnost	0.789 ± 0.034	0.779 ± 0.017	0.760 ± 0.013	0.734 ± 0.025
Úplnost	0.795 ± 0.024	0.795 ± 0.022	0.764 ± 0.042	0.736 ± 0.027
F-míra	0.792 ± 0.027	0.787 ± 0.017	0.762 ± 0.027	0.735 ± 0.022
FP+TN dataset				
Přesnost	0.803 ± 0.041	0.790 ± 0.017	0.763 ± 0.012	0.740 ± 0.035
Úplnost	0.840 ± 0.024	0.851 ± 0.023	0.836 ± 0.033	0.825 ± 0.020
F-míra	0.821 ± 0.026	0.819 ± 0.012	0.798 ± 0.020	0.780 ± 0.020
Přesnost (accuracy)	88.144	88.682	89.462	90.200

Tabulka 6.20: Výsledky testování modelu s 16 527 trénovatelnými parametry pro různé pravděpodobnosti dropoutu embedding vektorů. Testování proběhlo bez klíčových slov v osmi epochách. Přesnost (accuracy) je průměr všech běhů nad trénovacím datasetem.

7 Výsledky

Poté, co bylo dokončeno testování modelů, byl vybrán model s nejlepšími výsledky, který byl poté porovnán na kontrolním datasetu s klasifikací pomocí klíčových slov.

Tento způsob porovnání ale nereprezentuje způsob použití ve finální aplikaci. Protože falešná negativita klasifikace pomocí klíčových slov je podle statistik z kontrolního datasetu nulová, nemá smysl používat neuronovou síť pro klasifikaci stránek, označených klasifikací pomocí klíčových slov jako nerelevantní. Místo toho bude síť použita pouze pro eliminaci falešně pozitivních stránek, kde je klasifikace pomocí klíčových slov nedostačující. Tím se navíc ušetří čas, který by aplikace strávila zpracováním všech odkazů.

Byl proto proveden další test, ve kterém síť klasifikovala pouze stránky, které klasifikace pomocí klíčových slov označila jako relevantní. Tento test lépe reprezentuje reálné použití sítě, kdy se budou kombinovat silné stránky neuronové sítě a klasifikace pomocí klíčových slov.

	Podmínky	Ochrana dat	Nerelevantní
Podmínky	19	2	18
Ochrana dat	2	87	72
Nerelevantní	0	0	400
Přesnost (precision)	0.676		
Úplnost	0.900		
F-míra	0.772		
Přesnost (accuracy)	0.843		

Tabulka 7.1: Matice záměn a metriky klasifikace pomocí klíčových slov na kontrolním datasetu.

Architektura sítě je popsána v Tabulce 6.19, kde byla použita pravděpodobnost dropoutu 0.5 a model byl trénován osm epoch. Model s pravděpodobností dropoutu 0.5 podával srovnatelné výsledky s modelem s pravděpodobností 0.6, a proto byl vybrán jeden z nich. Samotná síť přinesla značné zlepšení (Tabulka 7.2) oproti klasifikaci pomocí klíčových slov (Tabulka 7.1). Sítě dosáhly zlepšení mezi 6 a 10 % v f-míře a zlepšení mezi 6 a 9 % v přesnosti (accuracy) za cenu mírného zvýšení falešné negativity.

Kombinace sítí a klasifikace pomocí klíčových slov výsledky mírně zlepšuje. Nejlepší model z pěti trénovaných modelů v kombinaci s klasifikací

pomocí klíčových slov (Tabulka 7.3) dosahuje f-míry 88 % a přesnosti (accuracy) přes 94 %. Navíc snižuje falešnou pozitivitu o 71 % ve srovnání s klasifikací pomocí klíčových slov a počet falešně negativních vzorků je malý.

Pouze síť	
Přesnost (precision)	0.788 ± 0.028
Úplnost	0.923 ± 0.019
F-míra	0.850 ± 0.019
Přesnost (accuracy)	0.919 ± 0.016
Síť + klíčová slova	
Přesnost (precision)	0.813 ± 0.021
Úplnost	0.928 ± 0.018
F-míra	0.866 ± 0.013
Přesnost (accuracy)	0.931 ± 0.010

Tabulka 7.2: Metriky neuronové sítě a neuronové sítě v kombinaci s klasifikací pomocí klíčových slov na kontrolním datasetu.

	Podmínky	Ochrana dat	Nerelevantní
Podmínky	20	3	6
Ochrana dat	0	82	20
Nerelevantní	1	4	464
Přesnost (precision)	0.828		
Úplnost	0.940		
F-míra	0.880		
Přesnost (accuracy)	0.943		

Tabulka 7.3: Matice záměn a metriky nejlepší neuronové sítě z pěti trénovaných v kombinaci s klasifikací pomocí klíčových slov na kontrolním datasetu.

8 Závěr

V rámci práce byl implementován crawler, který dokáže efektivně procházet web. Omezení počtu stránek, přidaných do fronty, se ukázalo jako přínosné a zlepšilo rychlost shromažďování relevantních stránek.

Klasifikace na základě klíčových slov nepodala dostatečně kvalitní výsledky. Falešná pozitivita se ukázala jako zásadní problém, který nebylo možné jednoduše a spolehlivě řešit upravením klasifikace pomocí klíčových slov. Pomocí této metody byl ale vytvořen dataset, který byl použit pro trénování komplexnějších řešení.

Pro zlepšení výsledků byly použity konvoluční neuronové sítě. Jako problém se ukázala přítomnost klíčových slov, na které se sítě fixovaly. Tento problém byl vyřešen omezením kapacity sítí, zakrytím klíčových slov a náhodným nulováním sémantických vektorových reprezentací slov. Díky tomu se síť nebyla schopna fixovat na klíčová slova ani na jejich kontext.

I přes vysokou chybovost datasetu se s použitím neuronových sítí povedlo zlepšit výsledky v porovnání s klasifikací pomocích klíčových slov. Podařilo se výrazně snížit míru falešné pozitivity a to až o 71 % v porovnání s klasifikací pomocí klíčových slov. F-míra dosahovala zlepšení mezi 6 a 9 %. Falešná negativita se na nejlepším natrénovaném modelu neukázala jako problém.

Výsledky by bylo teoreticky možné zlepšit použitím kvalitnějšího datasetu, k jehož vytvoření by byla použita neuronová síť. Zároveň finální model nebyl testován s upravenými hyperparametry, což by také mohlo výsledky zlepšit. Dále v rámci práce nebyla provedena integrace neuronové sítě jako klasifikátoru do crawleru, která bude realizována později v rámci projektu.

9 Přehled zkratek

NLP - Natural Language Processing
DOM - Document Object Model
HTML - HyperText Markup Language
XPath - XML Path Language
XML - Extensible Markup Language
ReLU - Rectified Linear Unit
CBOW - Continuous Bag of Words
GloVe - Global Vectors
HTTP - HyperText Transfer Protocol
JS - JavaScript
FP - falešně pozitivní
TN - skutečně negativní
LSTM - Long Short-Term Memory
URL - Uniform Resource Locator
CSV - Comma Separated Values

Seznam tabulek

Seznam obrázků

Literatura

[1] Cho, J. – Garcia-molina, H. – Icejfix, P. Crawling The Web: Discovery And Maintenance Of Large-Scale Web Data. 12 2001.

[2] Daubner, L. Deep learning, 2015. Dostupné z: https://is.muni.cz/th/tmuko/thesis.pdf.

[3] Diaz, G. *stopwords-iso* [online]. Dostupné z: https://github.com/stopwords-iso/stopwords-cs.

[4] Fabiyi, S. D. A Review of Unsupervised Artificial Neural Networks with Applications. *International Journal of Computer Applications*. 02 2019, 181, s. 22–26. doi: 10.5120/ijca2019918425.

[5] Goodfellow, I. – Bengio, Y. – Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[6] Grave, E. et al. Learning Word Vectors for 157 Languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[7] Gómez, R. *Understanding Categorical Cross-Entropy Loss Binary Cross-Entropy Loss Softmax Loss, Logistic Loss, Focal Loss and all those confusing names* [online]. 2018. Dostupné z: https://gombru.github.io/2018/05/23/cross_entropy_loss/.

[8] Harris, Z. S. Distributional Structure. *WORD*. 1954, 10, 2-3, s. 146–162. doi: 10.1080/00437956.1954.11659520. Dostupné z: https://doi.org/10.1080/00437956.1954.11659520.

[9] Joulin, A. et al. Bag of Tricks for Efficient Text Classification. *arXiv preprint arXiv:1607.01759*. 2016.

[10] Jurafsky, M. J. H. *Speech and Language Processing* [online]. December 2020. Dostupné z: https://web.stanford.edu/~jurafsky/slp3/ed3book_dec302020.pdf.

[11] Kasche, J. – Nordström, F. *Regularization Methods in Neural Networks* [online]. 2020. Dostupné z: https://www.diva-portal.org/smash/get/diva2:1389238/FULLTEXT01.pdf.

[12] Lederer, J. Activation Functions in Artificial Neural Networks: A Systematic Overview, 2021.

[13] Liu, L. et al. *Long Length Document Classification by Local Convolutional Feature Aggregation* [online]. July 2018. Dostupné z: https://www.mdpi.com/1999-4893/11/8/109/pdf.

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

[15] Mikolov, T. et al. Distributed Representations of Words and Phrases and their Compositionality. *CoRR*. 2013, abs/1310.4546. Dostupné z: http://arxiv.org/abs/1310.4546.

[16] Oliveira, R. M. S. d. et al. A System Based on Artificial Neural Networks for Automatic Classification of Hydro-generator Stator Windings Partial Discharges. *Journal of Microwaves, Optoelectronics and Electromagnetic Applications*. 09 2017, 16, s. 628 – 645. ISSN 2179-1074. Dostupné z: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S2179-10742017000300628&nrm=iso.

[17] O'Shea, K. – Nash, R. An Introduction to Convolutional Neural Networks, 2015.

[18] 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.

[19] Phung – Rhee. A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets. *Applied Sciences*. 10 2019, 9, s. 4500. doi: 10.3390/app9214500.

[20] Srivastava, N. et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* January 2014, 15, 1, s. 1929–1958. ISSN 1532-4435.

[21] Steinberger, J. První přednáška předmětu KIV/IR.

[22] TensorFlow. *TensorFlow Keras Categorical Crossentropy Loss Function* [online]. Dostupné z: https://www.tensorflow.org/api_docs/python/tf/keras/losses/CategoricalCrossentropy.

[23] Xu, B. et al. Empirical Evaluation of Rectified Activations in Convolutional Network, 2015.

[24] Zhai, Y. – Liu, B. Extracting Web Data Using Instance-Based Learning. 10, s. 113–132, 05 2007. doi: 10.1007/11581062_24. Dostupné z: https://

www.cs.uic.edu/~liub/publications/WISE05-instance-extract.pdf.
ISBN 978-3-540-30017-5.

A Klíčová slova

Klíčová slova pro stránky o ochraně dat jsou následující:

gdpr, cookie, privacy, ochrana, soukromi, soukromí, zpracovani, zpracování, zásady, zasady, udaju, udaje, údajů, údaje, protection, policy.

Klíčová slova pro stránky s podmínkami užití jsou následující:

podminky, podmínky, podmínkami, smluvni, smluvní, uzivani, užívání, terms, conditions.

B Uživatelská dokumentace

Na přiloženém DVD jsou zdrojové kódy aplikace, použité datasety a konfigurace neuronových sítí ve složce **src**. Program je napsán v jazyce Python. Použité knihovny nejsou přiloženy, ale je přiložen soubor *requirements.txt*, takže je možné knihovny doinstalovat přes pip.

Program byl testován na OS Windows 10 PRO 1909 a Ubuntu 20.04.1 s kernelem 5.8.0-48-generic, v obou případech s Pythonem ve verzi 3.8.1. Program byl testován na systémech s 16 GB operační paměti. Jako problém se ukázalo vytváření sémantických reprezentací pro OOV slova. Je totiž potřeba načíst do paměti binární soubor potřebný pro jejich vytvoření, který je poměrné velký (6.8 GB). Na systému s 16 GB operační paměti to při testování nebyl problém, ale u systému s 8 GB paměti a 2 GB swap oddílem, na kterém běžel OS Linux, paměť nestačila a program byl předčasně ukončen.

Všechny skripty využívají relativní cesty a je nutné je spouštět bez přesouvání, jinak nebudou fungovat.

B.1 Crawler

Ke spuštění crawleru slouží skript *run_crawler.py*. Konfiguraci crawleru lze upravit v souboru *config.py* ve složce **crawler**. Výchozí nastavení je takové, jaké bylo použito v práci.

Crawler podporuje pouze prohlížeč Google Chrome, který je potřeba stáhnout. Dále je potřeba pro něj stáhnout ovladač chromedriver. Program očekává, že ovladač se bude jmenovat *chromedriver* v případě, že bude použit OS Linux, nebo *chromedriver.exe* v případě, že bude použit OS Windows, a že bude umístěn ve složce **src**. V souboru *config.py* lze upravit proměnnou `chromedriver_path` tak, aby ukazovala stažený ovladač.

Výstupem crawleru jsou soubory *nolinks.txt*, *nocookies.txt* a *noterms.txt*, které obsahují stránky, ze kterých nebyly staženy žádné relevantní stránky, žádné stránky o ochraně dat a žádné stránky s podmínkami užití. Crawler stažené stránky ukládá do složky **pages**, kde každá doména má svou složku, která obsahuje složky **cookies** a **terms**, které obsahují stažené stránky o ochraně dat a stažené stránky s podmínkami. Crawler lze zastavit a znovu spustit, ale pokud bude v průběhu zastavování program zapisovat do některého z perzistentních seznamů, může být perzistentní seznam poškozen, což znamená, že je pak nutné crawler znovu spustit od nuly. Průběh se zaznamenává do souboru *log.txt*.

B.2 Neuronové sítě

Složka **src** mimo složku **crawler** obsahuje také následující složky:

- **model_configurations** - obsahuje konfigurace neuronových sítí pro možnost ověření výsledků,
- **neural_net** - obsahuje zdrojové kódy předzpracování textu, neuronových sítí a validace modelů,
- **raw_datasets** - obsahuje crawlerem shromážděné relevantní a nerelevantní stránky, použité v práci jako dataset,
- **split_datasets** - obsahuje předzpracovaný dataset rozdělený na trénovací a testovací části,
- **validation_datasets** - obsahuje vývojové a kontrolní datasety s jejich anotacemi,
- **final_models** - obsahuje modely, použité v sekci Výsledky.

Pro reprodukci výsledků jsou relevantní pouze složky **final_models** a **model_configurations**.

B.2.1 Vytvoření datasetu

Pro vytvoření datasetu a jeho rozdělení na trénovací a testovací části slouží skript *create_dataset.py* ve složce **src**. Ten nevyžaduje žádné argumenty a je tedy potřeba konfiguraci provést úpravou skriptu *create_dataset.py*. Pro správné fungování je nutné stáhnout FastText vektory pro český jazyk. K tomu slouží skript *download_fasttext.py*, který stačí spustit bez argumentů.

Je potřeba tento skript spustit na systému s dostatkem operační paměti. V rámci testování se ukázalo 16 GB operační paměti jako dostačující, ale menší velikosti mohou být problematické. Jako nedostačující se ukázalo 8 GB operační paměti a 2 GB swap.

Ve výchozím nastavení se jako dataset použijí relevantní a nerelevantní stránky ze složky **raw_datasets** a vytvoří se složky **train_created** a **test_created**, které obsahují trénovací a testovací datasety, připravené k použití v neuronové síti. Navíc se vytvoří soubor *preprocessing-config*, který obsahuje data nutná pro předzpracování a word embedding.

B.2.2 Trénování modelů

Skripty *train_saved_model.py* a *train_created_model.py* ve složce **src** slouží pro trénování modelů. První z nich trénuje modely použité v práci nad daty použitými v práci. Druhý skript modely trénuje nad daty, které byly vytvořeny pomocí skriptu *create_dataset.py* (viz. výše). Skripty jinak fungují obdobně.

Skripty vyžadují jeden argument, a to konfiguraci modelu. Konfigurace modelů jsou ve složce **model_configurations**, která obsahuje následující relevantní složky:

- **zakladni_model** - obsahuje konfigurace základního modelu (viz. Sekce 6.6),
- **dropout_model** - obsahuje konfigurace modelu s dropoutem (viz. Sekce 6.7),
- **zmensene_modely** - obsahuje konfigurace zmenšených modelů (viz. Sekce 6.8).

V těchto složkách se nacházejí konfigurace modelů, které jsou pojmenované tak, aby bylo z názvu zřejmé, o který model v práci se jedná.

Skripty jako argument vyžadují cestu k modelu ve tvaru <složka modelu/konfigurace>, tedy například

```
zakladni_model/s-klicovymi-slovy-8-epoch
```

Protože skripty využívají relativní cesty, je nutné je nepřesouvat. Skript poté načte konfiguraci a spustí trénování a validaci modelu. Průběh trénování lze sledovat v konzoli. Výsledky validace jsou pak také vypsány do konzole. Validace se provádí nad všemi ověřovacími datasety, a jako výsledek je do konzole pro každý dataset vypsána matice záměn a spočtené metriky.

Skript *train_created_model.py* je nastaven tak, aby automaticky pracoval s daty, vytvořenými pomocí skriptu *create_dataset.py*. Jediné, co očekává, jsou složky **train_created** a **test_created**. Skript slouží k ověření chování modelů nad jiným rozdělením datasetu. Pro jednoduchost skript používá konfigurace předzpracování použité v práci. To kvůli tomu, že konfigurace modelů mohou používat i ignorovat klíčová slova. Tímto způsobem lze ověřit nové rozdělení datasetu bez nutnosti řešit, jestli používat konfiguraci s nebo bez klíčových slov.

B.2.3 Finální modely

Modely ze sekce Výsledky jsou uloženy ve složce **final_models**. Ta obsahuje celkem pět modelů. Model z Tabulky 7.3 je pojmenován **model-best**. Pro ověření výsledků u těchto modelů slouží skript *validate_final_model.py* ve složce **src**. Ten vyžaduje jeden argument, a to název modelu ze složky **final_models**. Skript poté provede validaci modelu nad všemi datasety a navíc pro kontrolní dataset provede validaci v kombinaci s klasifikací pomocí klíčových slov.

C Obsah DVD

Přiložené DVD obsahuje následující složky:

- **src**
- **text**

Složka **src** obsahuje zdrojové kódy a použité datasety. Relevantní obsah je popsán výše v uživatelské dokumentaci.

Složka **text** obsahuje zdrojové soubory pro LaTeX, pomocí kterých je možné zkompilovat PDF soubor s textem bakalářské práce. Zdrojový text práce je v souboru *prace.tex* a použitá literatura v souboru *literatura.bib*. Složka **crawler_csv** obsahuje CSV soubory se statistikami běhů crawleru a složka **img** obsahuje vektorovou grafiku použitou v práci.