VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

ALGORITMY PRO ROZPOZNÁVÁNÍ POJMENOVANÝCH ENTIT
ALGORITHMS FOR NAMED ENTITIES RECOGNITION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Luca Winter

VEDOUCÍ PRÁCE
SUPERVISOR

prof. RNDr. Ing. Jiří Šťastný, CSc.

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav automatizace a informatiky
Student:	**Bc. Luca Winter**
Studijní program:	Strojní inženýrství
Studijní obor:	Aplikovaná informatika a řízení
Vedoucí práce:	**prof. RNDr. Ing. Jiří Šťastný, CSc.**
Akademický rok:	2016/17

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Algoritmy pro rozpoznávání pojmenovaných entit

Stručná charakteristika problematiky úkolu:

Práce bude věnována extrakci informací z textu, konkrétním cílem je rozpoznání předem daných důležitých údajů v emailových zprávách. Rešeršní část práce bude věnována existujícím technikám a výběru vhodných algoritmů pro tuto práci. V praktické části student implementuje několik algoritmů a využije databázi emailových zpráv pro ověření funkčnosti a porovnání algoritmů.

Cíle diplomové práce:

1. Popište existující metody zabývající se problematikou rozpoznávání pojmenovaných entit.
2. Implementujte některé z algoritmů a porovnejte je.
3. Při porovnání výsledků z dat se zaměřte na schopnost algoritmu generalizovat, např. v případě jiné struktury vstupního textu nebo různých podobných slov.
4. Zhodnoťte dosažené výsledky a možnosti dalšího vývoje.

Seznam doporučené literatury:

MAYNARD, Diana, Valentin TABLAN, Cristian URSU, Hamish CUNNINGHAM and Yorick WILKS. Named Entity Recognition from Diverse Text Types. Tsigov Chark, 2001.

BIRD, Steven, Ewan KLEIN and Edward LOPER. Natural language processing with Python. 1st ed. Beijing: O'Reilly, c2009. ISBN 978-0-596-51649-9.

SEKINE, Satoshi and Elisabete Marques RANCHHOD. Named entities: recognition, classification, and use. Philadelphia: John Benjamins Pub. Company, 2009. Benjamins current topics, v. 19. ISBN 978-9027222497.

MANNING, Christopher D., Prabhakar. RAGHAVAN and Heinrich. SCHÜTZE. Introduction to information retrieval. New York: Cambridge University Press, 2008. ISBN 05-218-6571-9.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2016/17

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Cieľom práce je zistiť, ktorý algoritmus je schopný najlepšie rozpoznávať pomenované entity v emailových správach. V teoretickej časti práce sú popísané existujúce nástroje v tejto oblasti. Praktická časť obsahuje návrh dvoch nástrojov špeciálne určených na učenie nových modelov schopných rozpoznávať pomenované entity v emailových správach. Prvý nástroj je implementáciou neurónovej siete, druhý nástroj využíva CRF grafový model. Úspešnosť a schopnosť existujúcich i navrhnutých nástrojov generalizovať je porovnaná na časti emailových správ poskytnutých firmou Kiwi.com.

ABSTRACT

The aim of this work is to find out which algorithm is the best at recognizing named entities in e-mail messages. The theoretical part explains the existing tools in this field. The practical part describes the design of two tools specifically designed to create new models capable of recognizing named entities in e-mail messages. The first tool is based on a neural network and the second tool uses a CRF graph model. The existing and newly created tools and their ability to generalize are compared on a subset of e-mail messages provided by Kiwi.com.

KĽÚČOVÉ SLOVÁ

rozpoznávanie pomenovaných entít, spracovanie prirodzeného jazyka, rekurentné neurónové siete, podmienené náhodné polia

KEYWORDS

named entity recognition, natural language processing, recurrent neural networks, conditional random fields

BIBLIOGRAFICKÁ CITÁCIA

WINTER, L. Algoritmy pro rozpoznávání pojmenovaných entit. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2017. 69 s. Vedoucí diplomové práce prof. RNDr. Ing. Jiří Šťastný, CSc.

ČESTNÉ PREHLÁSENIE

Prehlasujem, že táto práca je mojim pôvodným dielom, spracoval som ju samostatne pod vedením prof. RNDr. Ing. Jiřího Šťastného, CSc. a s použitím literatúry uvedenej v zozname literatúry.

V Brne dňa 26. 5. 2017 .....

Bc. Luca Winter

POĎAKOVANIE

Rád by som sa týmto poďakoval vedúcemu diplomovej práce pánovi prof. RNDr. Ing. Jiřímu Šťastnému, CSc. za odbornú pomoc a rady pri spracovaní tejto práce a počas štúdia. Ďalej by som sa rád poďakoval Ing. Ondřejovi Veselému z firmy Kiwi.com za pomoc a priebežnú konzultáciu výsledkov. V neposlednej rade by som sa chcel poďakovať svojej rodine za podporu pri celom štúdiu.

OBSAH

1 ÚVOD

Dnešná doba, charakteristická prítomnosťou počítačov v každej oblasti nášho života, má za následok tvorbu veľkého množstva digitálnych informácií. Okrem stoviek miliónov priamo dostupných webových stránok [1] bežný užívateľ každodenným používaním internetu takisto vytvára veľké množstvo digitálnych dát. Sú nimi napríklad emailové správy, ďalšie formy komunikácie, textové dokumenty, prezentácie a podobne. Z obrovského množstva dostupných informácií na internete je nutné nejakým spôsobom vytiahnuť to dôležité. V prípade, že sa jedná o tabuľky, prípadne štruktúrovaný formát priamo určený pre spracovávanie počítačom je toto celkom jednoduché. Veľká časť informácií je však dostupná v článkoch, správach a iných formách. Takýto text, jednoducho čitateľný a bežne používaný ľuďmi pre komunikáciu, sa nazýva prirodzený text. Spracovanie prirodzeného textu (*NLP – natural language processing*) je oblasť umelej inteligencie, ktorá zažíva v súčasnej dobe veľký rozmach vďaka veľkému nárastu dostupnej výpočtovej sily. Na svet prichádzajú rôzni virtuálni asistenti využívajúci poznatky z tejto oblasti, ktorí nám umožňujú si uľahčiť život. Títo asistenti už teraz v mnohých oblastiach dokážu nahradiť človeka. Je tak možné si napríklad naplánovať schôdzku a virtuálny asistent sa formou otázok v prirodzenom jazyku opýta na všetky potrebné informácie a schôdzku následne zapíše do kalendára. Pred schôdzkou virtuálny asistent vyšle upozornenie v presnom čase, kedy je potrebné na schôdzku odísť, aby bol zaručený príchod na čas. Do dĺžky cesty je obvykle započítaná i aktuálna dopravná situácia.

V tejto práci je skúmaná iná časť spracovania prirodzeného jazyka, ktorou je rozpoznávanie pomenovaných entít (*NER – named entity recognition*). Jednoducho povedané sa jedná o rozpoznávanie dôležitých údajov v texte. Počítač na základe spracovaných aktuálnych správ napríklad dokáže zmysluplne odpovedať na otázky typu *„Vyhrala Barcelona včerajší futbalový zápas?“*. Základným princípom je, že počítač sa na základe učiacich dát naučí, že v blízkosti slov ako napríklad *vyhrala, prehrala, výsledok* sa často nachádza názov športového klubu. Takisto sa naučí, že slová ako napríklad *Barcelona, Amsterdam* či *Brno* často v prirodzenom texte spájame s ich príslušnými športovými klubmi. Na základe týchto poznatkov a slov *včerajší* a *futbalový* je počítač schopný dohľadať stanovené informácie a vo forme prirodzeného textu podať naspäť adekvátnu odpoveď, napríklad takúto: *„Áno, Barcelona včera vyhrala 3-2 nad klubom Real Madrid.*“. Viac informácií o pomenovaných entitách a probléme ich rozpoznávania sa nachádza v kapitole 2.

Motiváciou písania tejto práce je spolupráca s firmou Kiwi.com, ktorá má záujem využiť nástroje z tejto oblasti pre zefektívnenie procesov vo firme. Táto firma sa zaoberá vyhľadávaním a predajom leteniek. Po nakúpení letenky nasleduje obvykle potvrdzovacia emailová správa, v ktorej sa nachádzajú podrobné informácie o lete. Niekedy sa stáva, že let je aerolíniou zrušený či presunutý. V tom prípade je nutné v prijatej správe identifikovať nové údaje a čo najrýchlejšie zákazníka informovať o nových

podrobnostiach letu. Využitím oblasti rozpoznávania pomenovaných entít v takýchto emailových správach je možné tento proces zefektívniť. Ďalšie podrobnosti o probléme, ktorý je riešený, sa nachádzajú v kapitole 4.

Nástroje, ktoré v súčasnosti v tejto oblasti existujú, neboli vytvorené pre prácu s HTML kódom, ktorý obsahujú aj emailové správy. Kvôli odlišnostiam formátu HTML kódu a prirodzeného jazyka by bolo nutné upraviť zdrojový kód existujúcich riešení alebo vytvoriť nástroj, ktorý si s HTML kódom poradí. V tejto práci bola zvolená druhá možnosť – vytvorenie nových nástrojov používajúce algoritmy na základe neurónových sietí a grafových modelov. Táto možnosť bola zvolená hlavne z dôvodu obsiahlosti existujúcich nástrojov, ktorých úprava a s ňou spojené problémy by bola pravdepodobne náročnejšia ako vytvorenie nových. K nástrojom boli vytvorené i serverové časti, ktorým je možné odoslať formou HTTP žiadosti neoznačenú emailovú správu. Server vráti emailovú správu naspäť, ale rozpoznané dôležité údaje budú označené špeciálnymi HTML značkami. Tieto značky obsahujú druh rozpoznanej dôležitej informácie, ktorým môže byť napríklad číslo letu či čas odletu, a pravdepodobnosť, s ktorou sa o tento druh jedná. Vytvorené nástroje a algoritmy za nimi sú popísané v kapitolách 5, 6 a 7. Porovnanie výsledkov jednotlivých nástrojov sa nachádza v kapitole 8.

2 SPRACOVANIE INFORMÁCIÍ V TEXTE

Kapitola sa zaoberá širšou oblasťou spracovávania informácii v texte. Pojmy ako sú prirodzený text, pomenovaná entita a podoblasť rozpoznávania pomenovaných entít sú definované a vysvetlené v jednotlivých podkapitolách. Kapitola slúži ako úvod do riešenej problematiky a objasňuje pojmy, ktoré sú použité v ďalších častiach práce.

2.1 Extrakcia informácií

Extrakcia informácií je proces získavania štruktúrovaných informácií požadovaného charakteru z neštruktúrovaného alebo čiastočne štruktúrovaného textu. Môže sa jednať o vzťahy, pomenovania, čísla a podobne. Najnáročnejším v tejto oblasti býva získanie dát o udalostiach – kto, kedy, s kým, kde niečo urobil. Extrakcia informácií nachádza uplatnenie v mnohých oblastiach, od vyšetrovania teroristických útokov, firemných akvizícií, vypuknutia chorôb až po spracovanie životopisov [2]. Tab. 1 znázorňuje rozdiel medzi čiastočne štruktúrovaným a neštruktúrovaným textom.

Meno: Lukáš Zima Študijný obor: Aplikovaná informatika a riadenie Škola: Vysoké učení technické v Brne
Volám sa Lukáš Zima, študujem 2. ročník magisterského študijného oboru na Vysokom učení technickom v Brne.

Tab. 1 Čiastočne štruktúrovaný text (hore) a neštruktúrovaný text (dole).

Systémy zaoberajúce sa úlohou extrakcie informácií vyžadujú veľké množstvo informácií špecifických pre danú oblasť. Prvé systémy využívali ručne vytvorené pravidlá, ktoré spolu vytvárali zložité automaty. Tieto systémy boli veľmi efektívne, nevýhodou bola ale nutnosť vytvárania týchto pravidiel. Odhaduje sa, že vytvorenie pravidiel využívané systémom UMass MUC-4 si vyžiadalo približne 1500 človekohodín práce. Z tohoto dôvodu sa začali v tejto oblasti používať metódy strojového učenia [2].

Regulárne výrazy

Užitočným nástrojom v oblasti extrakcie informácií sú regulárne výrazy. Umožňujú vyhľadávať informácie, o ktorých vieme obmedzené množstvo informácií. Príkladom môže byť vyhľadanie všetkých časov v texte v tvare 12:34, k čomu slúži regulárny výraz *\d{2}:\d{2}*, kde *\d* predstavujú špeciálne znaky číslic a *2* značí ich hľadaný počet. Ďalej je možné vyhľadávať napríklad podľa veľkosti písmen, dĺžky slova a mnohých ďalších. V tejto práci boli využité pre urýchlenie označenia veľkého množstva textu počas tvorby

vývojového korpusu (viď kapitola 4), keďže je možné text zmysluplne nahrádzať, napríklad z vety *„Volám sa Lukáš Zima“* nahradením môžeme získať označenú vetu v tvare „*Volám sa <meno>Lukáš Zima</meno>*. Týmto spôsobom je možné efektívne označiť veľké množstvo textu, ktorý je vytvorený na základe určitej predlohy.

2.2 Základné pojmy v spracovaní textu

Tokenizácia (*tokenization*)

Pod pojmom tokenizácia sa rozumie delenie textu na menšie časti – tokeny. Väčšinou sa jedná o slovo, skratku alebo číslo, ktoré je v texte oddelené medzerami alebo interpunkciou [3]. Existujú však výnimky, kedy sa dve slová považujú za jeden token, napr. v slove „čierno-biely“. Riešenie navrhnuté v rámci tejto práce ponúka možnosť tokenizovať týmto klasickým spôsobom, na báze medzier a interpunkcie, ale i tokenizovať znak po znaku (*character by character*), tj. každý znak je braný ako samostatný token. Rozpoznávanie pomenovaných entít s tokenizáciou znak po znaku je z hľadiska výpočtovej sily omnoho náročnejšie a z tohoto dôvodu začalo dosahovať dobré výsledky až v posledných rokoch [4]. Umožňuje učenému systému lepšie pochopiť morfologicky bohatšie jazyky, kde slová môžu mať veľké množstvo rôznych predpôn a prípon podľa gramatického tvaru, alebo jazyky, kde prítomnosť jedného znaku môže ovplyvniť význam celého slova. Táto vlastnosť bola experimentálne overená napr. na čínskom jazyku, kde spôsob slovo po slove dosiahol presnosť 84,1% a spôsob znak po znaku presnosť 91,7% na úlohe rozpoznania slovného druhu (*Part-of-Speech Tagging*) [5].

Lemmatizácia (*lemmatization*) a stematizácia (*stemming*)

Proces lemmatizácie je prevod slova na jeho základný tvar (*lemma*), tj. jeho slovníkový tvar. Lemmou podstatných mien je tvar slova v prvom páde jednotného čísla, lemmou prídavných mien je prídavné meno mužského roku v prvom páde jednotného čísla a v prípade slovies sa jedná o neurčitok. Stematizácia (*stemming*) je podobný proces, pri ktorom je hľadaný základ (*stem*) slova. Stematizovaný tvar slova sa môže ale nemusí zhodovať s lingvistickým koreňom slova. Najjednoduchšie spôsoby stematizácie sú založené na definovanom odstránení predpôn a prípon slov, napr. odstránením prípony *ať* vzniknú zo slov *bývať, ležať* kmene slov *býv, lež*. Existujú aj zložitejšie algoritmy prispôsobené na konkrétnu skupinu jazykov či rôzne štatistické metódy [3].

Slovo	Lemmatizovnaný tvar	Stematizovaný tvar
Veľvyslanca	Veľvyslanec	Veľvyslan
Neurčitej	Neurčitý	Neurčit
Rozdal	Rozdať	Rozd

Tab. 2 Príklady lemmatizovaných a stematizovaných tvarov slov.

V Tab. 2 sú uvedené príklady lemmatizovaných a stematizovaných tvarov slov. Nájdenie lingvisticky správneho koreňa slova či lemmy je v morfologicky zložitých jazykoch ako slovenčina či čeština algoritmicky zložité. Naviac v reálnych aplikáciách často postačuje nájdenie časti slova, ktorá bude rovnaká pre všetky príbuzné slová. I preto je stematizácia dôležitá. Jedná sa o rýchlejší, ale menej presný proces ako lemmatizácia [6].

2.3 Prirodzený jazyk a jeho spracovanie

Prirodzený jazyk je definovaný ako jazyk bežne používaný ľuďmi na komunikáciu, akými sú napríklad anglický, český, slovenský alebo francúzsky jazyk. Od umelých jazykov, akými sú napríklad programovacie jazyky, sa líšia tým, že nie sú jednoducho definovateľné explicitnými pravidlami. Pod pojmom spracovanie prirodzeného jazyka (*NLP – natural language processing*) sa chápe akékoľvek narábanie s textom za podpory počítaču. Môže sa jednať o problémy najjednoduchšieho typu, akým je napríklad počítanie frekvencie použitých slov na rozpoznanie štýlu písania, prípadne autora textu, až po problémy zložité, pri ktorých je snaha pochopiť jazyk natoľko, aby bolo možné vytvoriť zmysluplnú odpoveď [7].

Obsahom tejto práce je spracovanie a rozpoznávanie pomenovaných entít v emailových správach, ktoré obsahujú text v jeho neprirodzenej podobe. Jedná sa o štruktúrovaný HTML kód. Prirodzený text sa nachádza len v častiach medzi značkami jazyku HTML. V správach v tomto formáte sú obsiahnuté informácie o formátovaní textu, jeho presnom rozložení a podobne. Často sa napríklad stáva, že v emailovej správe sa nachádza tabuľka, ktorá je definovaná práve značkami jazyku HTML. Tab. 3 ukazuje vizuálnu podobu ukážky HTML kódu jednoduchej tabuľky.

```
<table>
        <tr>
                <td>Meno</td>
                <td>Jozef</td>
        </tr>
        <tr>
                <td>Priezvisko</td>
                <td>Mrkvička</td>
        </tr>
        <tr>
                <td>Bydlisko</td>
                <td>Zelená 42</td>
        </tr>
        <tr>
                <td>PSČ</td>
                <td>851 10</td>
        </tr>
</table>
```

Meno	Jozef
Priezvisko	Mrkvička
Bydlisko	Zelená 42
PSČ	851 10

Tab. 3 Štruktúrovaný HTML kód (vľavo) a jeho vizuálna podoba.

Emailové správy a HTML dokumenty všeobecne je možné spracovať dvoma spôsobmi:

a) Extrakciou čistého textu spomedzi značiek jazyku HTML.
b) Považovaním jazyku HTML za prirodzený jazyk a definovaním vlastných pravidiel tokenizácie jazyku.

Výhodou spôsobu a) je, že sa z HTML kódu extrahuje text v podobe prirodzeného jazyka, tj. obyčajných slov bez HTML značiek. Je možné na takýto text nasadiť existujúce softvérové balíky, popísané v kapitole 3. Dokumenty spracované týmto spôsobom nie je jednoduché dostať späť do podoby štruktúrovaného HTML kódu. Použitím tohoto spôsobu ďalej prichádzame o užitočnú vlastnosť HTML dokumentov, ktorou je práve štruktúrovanosť. V HTML kóde sú uložené informácie o formátovaní a rozložení textu v dokumente, ktoré môžu byť užitočné pri extrakcii informácií z celku, napríklad pri extrakcii informácií z HTML tabuľky, kde navyše môžeme zvoliť sled spracovania riadok po riadku alebo stĺpec po stĺpci, viď Tab. 4. Z tabuľky je zrejmé, že pri nesprávnom zvolení sledu sa stáva úloha náročnejšou.

Pri zvolení spôsobu b) sú informácie o formátovaní a rozložení textu zachované. V dokumentoch spracovaných týmto spôsobom ostane teda väčšie množstvo informácií. Nevýhodami tohoto prístupu je nutnosť určiť vlastné pravidlá tokenizácie (viď kapitola 2.2) a dlhšie spracované dokumenty. Ďalšou nevýhodou je nevhodnosť nasadiť na takto štruktúrovaný text existujúce softvérové balíky bez ďalších úprav. Tieto balíky totiž bez úprav považujú za slová aj značky kódu HTML.

Meno Jozef Priezvisko Mrkvička Bydlisko Zelená 42 PSČ 851 10	Meno Priezvisko Bydlisko PSČ Jozef Mrkvička Zelená 42 851 10	<tr> <td> Meno </td> <td> Jozef </td> </tr> <tr> <td> Priezvisko </td> <td> Mrkvička </td> ...

Tab. 4 Spracovanie vyššie uvedeného HTML kódu spôsobom a) riadok po riadku (vľavo), stĺpec po stĺpci (v strede) a spôsobom b) (vpravo).

V tejto práci bol zvolený spôsob spracovania b). Hlavným dôvodom zvolenia tohoto prístupu je nutnosť so spracovanými emailovými správami ďalej pracovať a kontrolovať

výstupy navrhnutého riešenia, pričom je potrebné zachovať štruktúru správ. Zachovanie štruktúry je pri spôsobe a) problematické, zatiaľ čo pri spôsobe b) je zachovanie štruktúry samozrejmé z podstaty spracovania.

2.4 Rozpoznávanie pomenovaných entít

Pomenovaná entita

Termín pomenovaná entita (*NE – named entity*) a s ním spojený termín rozpoznávanie pomenovaných entít (*NER – named entity recognition*) boli zavedené v roku 1995 v súvislosti s konferenciou MUC-6 (Message Understanding Conference). Cieľom konferencií MUC bola podpora výskumu v oblasti extrakcie informácií z textu. Na týchto konferenciách sa najčastejšie vyhodnocovali finančné a vojenské správy, prípadne správy informačných služieb o teroristických útokoch. Na MUC-6 bol termín pomenovaná entita definovaný nasledovne [8]:

- ENAMEX s atribútmi ORGANIZATION pre mená organizácií, PERSON pre mená osôb a LOCATION pre názvy geografických miest.
- TIMEX s atribútmi DATE pre dátumy a roky, TIME pre časové údaje. Mohlo sa jednať o absolútne časové údaje (napr. *o 14.00*), ale i o relatívne (napr. *dva dni po*).
- NUMEX s atribútmi MONEY pre peňažné čiastky a PERCENT pre percentuálne hodnoty.

Príklad označenia vety podľa MUC-6 definície:

„Počas druhej svetovej vojny, v júni 1942, sa americká lietadlová loď USS Hornet, posledná loď triedy Yorktown, zúčastnila bitky pri Midway.“

„Počas druhej svetovej vojny, v <TIMEX TYPE=“DATE“>júni 1942</TIMEX>, sa americká lietadlová loď USS Hornet, posledná loď triedy Yorktown, zúčastnila bitky pri <ENAMEX TYPE=“LOCATION“>Midway</ENAMEX>.“

Je zrejmé, že pomocou takto definovaných entít nie je možné popísať všetky druhy dôležitých údajov nachádzajúcich sa v akomkoľvek texte, napríklad názvy artefaktov či dôležitých udalostí. Projekty IREX (*Information Retrieval and Extraction Exercise*) a CoNLL (*The Conference on Natural Language Learning*) túto definíciu ďalej rozšírili o ďalší typ ARTIFACT. S lepšími výsledkami a postupným rozširovaním úlohy NER na ďalšie domény vznikla potreba opustiť túto úzku definíciu a zovšeobecniť ju.

Podľa smerníc TEI (Text Encoding Initiative), v ktorých sú dostupné prostriedky pre označovanie textov v prirodzenom jazyku, sú pomenované entity (NE) rozšírené o adresy, čísla, časové úseky, skratky a iné. Zároveň môžu byť NE podľa konkrétnej domény a úlohy dodefinované.

Vyhodnocovanie úspešnosti rozpoznávania

Úspešnosť rozpoznávania pomenovaných entít v texte sa vyhodnocuje pomocou veličín presnosť (*precision*), pokrytie (*recall*) a ich kombináciou v podobe F-miery (*f-measure*). Pri vyhodnocovaní sa používajú skratky pre správne rozpoznanú entitu – *tp* (*true positive*), správne nerozpoznanú entitu – *tn* (*true negative*), chybne rozpoznanú entitu – *fp* (*false positive*) a chybne nerozpoznanú entitu – *fn* (*false negative*), viď Tab. 5. Vzorce pre výpočet presnosti, pokrytia a F-miery boli prevzaté z [3].

	Správne	**Chybne**
Rozpoznané	*tp (true positive)*	*fp (false positive)*
Nerozpoznané	*tn (true negative)*	*fn (false negative)*

Tab. 5 Kontingenčná tabuľka pre vyhodnocovanie NER systémov. Prevzaté z [9].

Presnosť (*precision*)

Presnosť je definovaná ako pomer správne rozpoznaných entít k celkovému počtu rozpoznaných entít. Jedná sa teda o tú časť všetkých rozpoznaných entít, ktorá bola rozpoznaná správne. Počíta sa pomocou vzorca (1).

$$\frac{tp}{tp + fp}. \tag{1}$$

Pokrytie (*recall*)

Pokrytie je definované ako pomer správne rozpoznaných entít k celkovému počtu entít. Jedná sa teda o tú časť všetkých entít v texte, ktorá bola rozpoznaná správne. Počíta sa pomocou vzorca (2).

$$\frac{tp}{tp + fn}. \tag{2}$$

F-miera (*f-measure*)

F-miera predstavuje kombináciu predošlých dvoch mier. Tradične sa používa takzvaná F_1-miera, ktorá je harmonickým priemerom presnosti a pokrytia. Počíta sa pomocou vzorca (3).

$$F_1 = 2 \cdot \frac{1}{\frac{1}{pokrytie} + \frac{1}{presnosť}} = 2 \cdot \frac{presnosť \cdot pokrytie}{presnosť + pokrytie}. \tag{3}$$

V prípade, že je kladený vyšší dôraz na presnosť alebo pokrytie, je možné použiť takzvanú F_β-mieru. Počíta sa pomocou vzorca (4), prípadne (5). Význam presnosti je možné zvýšiť znížením hodnoty β, význam pokrytia je možné zvýšiť zvýšením hodnoty β. Pri voľbe β = 1 vzorce (4) a (5) zodpovedajú vzorcu (3). Často používané sú $F_{0,5}$ a F_2-miery, kde $F_{0,5}$ miera uprednostňuje presnosť a F_2-miera uprednostňuje pokrytie.

$$F_\beta = (1 + \beta^2) \cdot \frac{1}{\frac{1}{pokrytie} + \frac{\beta^2}{presnosť}} \tag{4}$$

$$F_\beta = (1 + \beta^2) \cdot \frac{presnosť \cdot pokrytie}{(\beta^2 \cdot presnosť) + pokrytie} \tag{5}$$

3 EXISTUJÚCE NÁSTROJE

V oblasti rozpoznávania pomenovaných entít existuje mnoho komerčne dostupných nástrojov, ktoré sú schopné v prirodzenom texte rozpoznávať väčšinu bežne potrebných druhov entít, akými sú napríklad meno osoby, názov miesta, názov organizácie a podobne. Tieto nástroje obvykle obsahujú natrénované modely v niekoľkých jazykoch, ktoré toto rozpoznanie umožňujú. Niektoré nástroje však ponúkajú i spôsob, akým na základe trénovacích dát vytvoriť vlastný model, prispôsobený na mieru, umožňujúci rozpoznať akýkoľvek požadovaný druh pomenovanej entity na základe poskytnutia trénovacích dát. V tejto kapitole sú predstavené dva najznámejšie z nich, Stanford NER a Apache OpenNLP.

3.1 Stanford NER

Stanford NER, nástroj známy i pod názvom CRFClassifier, je nástrojom pre rozpoznávanie pomenovaných entít implementovaný v jazyku Java. V základnom balíku obsahuje i model pre rozpoznávanie pomenovaných entít v anglických textoch. Z internetovej stránky je možné stiahnuť modely pre ďalšie jazyky, akými sú napríklad nemčina či španielčina. Zdrojový kód pre tento nástroj je zverejnený na internetovej stránke [10]. Akademická práca citovaná v súvislosti s týmto nástrojom je [11]. V tejto práci narábame s nástrojom vo verzii 3.7.0.

Vstupným súborom pre tento nástroj je súbor obsahujúci dáta oddelené znakom tabulátoru (*tab-separated column data*). Pri obvyklom spôsobe využívania sa v prvom stĺpci nachádza token, v druhom stĺpci jeho korešpondujúca značka. Jednotlivé dokumenty sú od seba oddelené prázdnym riadkom.

Modely Conditional Random Fields (CRF)

Stanford NER využíva model na základe CRF (*conditional random fields*). Jedná sa o grafový pravdepodobnostný prístup, ktorý je podrobnejšie popísaný v kapitole 6.

Nastaviteľné parametre

Nástroj Stanford NER umožňuje nastaviť množstvo parametrov, ktoré umožňujú prispôsobiť učenie na rôznych odlišných doménach. Parametre je možné predávať cez príkazový riadok, no v prípade opakovaného spúšťania je praktickejšie vytvoriť textový súbor obsahujúci všetky parametre a cez príkazový riadok predávať len cestu k súboru vlastností (*properties file*). Možných parametrov je veľké množstvo, ich prehľad sa nachádza na internetovej stránke [10].

3.2 Apache OpenNLP

Druhým známym nástrojom v širšej oblasti spracovávania prirodzeného textu je Apache OpenNLP. Okrem úlohy rozpoznávania pomenovaných entít tento nástroj ponúka možnosť riešiť i iné časté úlohy v oblasti spracovania prirodzeného textu, akými sú napríklad segmentácia viet či rozpoznanie slovného druhu. Predtrénované modely na rozpoznávanie pomenovaných entít v niektorých jazykoch ako i zdrojový kód k tomuto nástroju sú voľne dostupné na internetovej stránke [12]. V tejto práci sa pracuje s verziou nástroju 1.7.2.

Vstupný súbor pre tento nástroj má iný formát ako v prípade Stanford NER. Jedná sa o dokumenty od seba oddelené znakom nového riadku. Pomenované entity sú označené priamo v texte (*inline*) nasledovným spôsobom:

„Počas <START:event> druhej svetovej vojny <END> , v <START:date> júni 1942 <END> , sa americká lietadlová loď <START:misc> USS Hornet <END> , posledná loď triedy <START:misc> Yorktown <END> , zúčastnila bitky pri <START:location> Midway <END> .“

3.2.1 Modely v Apache OpenNLP

V nástroji Apache OpenNLP je možné využiť niekoľko rôznych algoritmov. V tejto kapitole sú stručne uvedené princípy ich fungovania.

Model maximálnej entropie (Maximum Entropy Model)

Maximálne entropické modelovanie je rámec pre integráciu informácií z mnohých heterogénnych zdrojov informácií pre klasifikáciu. Údaje pre problém klasifikácie sú opísané ako (potenciálne veľký) počet príznakov. Tieto príznaky predstavujú napríklad dĺžku tokenu, stematizovaný tvar tokenu a podobne. Môžu byť pomerne zložité a umožňujú využiť predchádzajúce poznatky o dátach. Každý príznak predstavuje obmedzenie modelu. Potom vypočítame model maximálnej entropie, to znamená model s maximálnou entropiou zo všetkých modelov, ktoré vyhovujú obmedzeniam. Ak by sme vybrali model s menšou entropiou, pridali by sme k modelu obmedzenia, ktoré nie sú opodstatnené empirickými dôkazmi, ktoré máme k dispozícii. Výber modelu maximálnej entropie je motivovaný túžbou zachovať čo najviac neistoty.

Postup nachádzania modelu maximálnej entropie je nasledovný. Najprv pre daný zoznam príznakov vypočítame očakávanú hodnotu na základe trénovacích dát. Každý príznak následne predstavuje obmedzenie, ktoré hovorí, že táto empirická očakávaná hodnota je rovnaká ako očakávaná hodnota tejto vlastnosti vo výslednom modeli maximálnej entropie. Zo všetkých pravdepodobnostných rozdelení, ktoré spĺňajú tieto obmedzenia, sa snažíme nájsť rozdelenie s maximálnou entropiou. Je možné dokázať,

že existuje unikátne rozdelenie s touto maximálnou entropiou a existuje algoritmus, ktorým je zaručená konvergenciu k nemu. Týchto algoritmov je niekoľko, najstarším a najznámejším je algoritmus GIS (*generalized iterative scaling*). Toto rozdelenie je možné zapísať rovnicou (6):

$$p(o|h) = \frac{1}{Z(h)} \cdot \prod_{j=1}^{k} \alpha_j^{f_j(h,o)}, \tag{6}$$

kde *o* predstavuje výstup, *h* históriu (kontext), *Z(h)* normalizačnú funkciu a f_j predstavuje binárnu funkciu, ktorá rozhoduje o vlastnosti. Parameter α_j sa dá predstaviť ako váha a je iteratívne odhadovaný napríklad vyššie spomínaným algoritmom GIS [12][13].

Model Naive Bayes

Jedná sa o pravdepodobnostný model založený na Bayesovej vete. Pravdepodobnostné modely sú dôležité v problémoch, kedy nie je možné nájsť presné riešenie a je nutné stanoviť, akú pravdepodobnosť majú jednotlivé hypotézy. Ku klasifikovaniu *x* je možné formulovať model, ktorý zostavuje podmienenú pravdepodobnosť *P(y|x)* rozdielnych *y*, kde *y* predstavuje klasifikačnú triedu. Predikovaná trieda je vybraná podľa *y* s najvyššou hodnotou pravdepodobnosti, ktorá je spočítaná podľa (7).

$$P(y|x) = \frac{P(x|y) \cdot P(y)}{P(x)} \tag{7}$$

P(y) je určené početným podielom triedy v trénovacích dátach. *P(x)* nie je pre klasifikáciu relevantná, pretože je porovnávaná pre rôzne *y* na rovnakom *x*. Zaujímavým členom rovnice je člen *P(x|y)*, ktorý predstavuje pravdepodobnosť javu *x* podmienenú výskytom triedy *y*. Odhadovanie *P(x|y)* nie je triviálne, pretože spočíva v nájdení exponenciálneho množstva združených pravdepodobností jednotlivých atribútov. Za určitých predpokladov je možné tento problém zjednodušiť. V prípade Naive Bayes klasifikátoru sa predpokladá, že *p* atribútov je v každej triede nezávislých. Potom je podmienená pravdepodobnosť *P(x|y)* daná (8):

$$P(x|y) = \prod_{i=1}^{p} P(x_i|y), \tag{8}$$

čo naznačuje, že je potrebné len vypočítať každý atribút v každej triede, aby bolo možné určiť podmienenú pravdepodobnosť a týmto spôsobom sa vyhnúť výpočtom združených pravdepodobností [14].

Perceptronový model

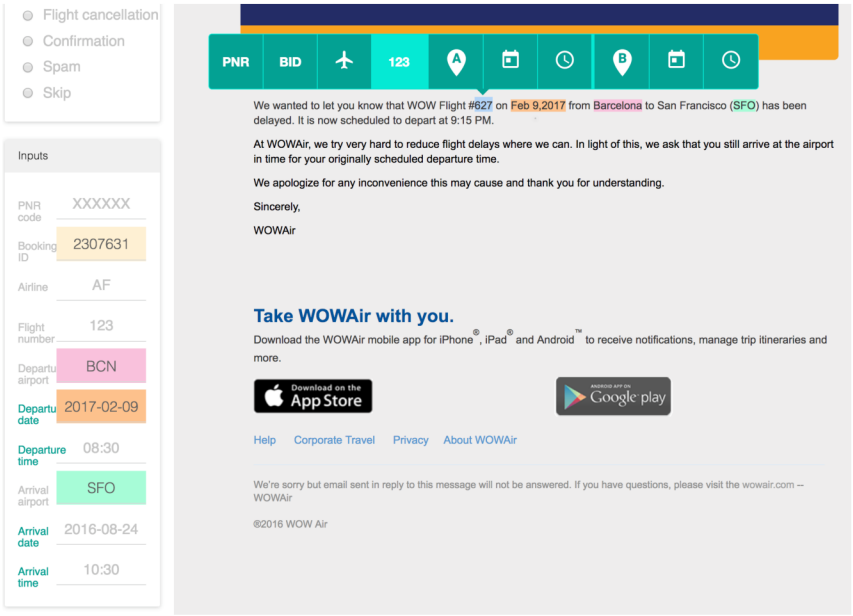
Tento model je založený na základnom stavebnom prvku najjednoduchšieho typu neurónových sietí – perceptrone. Neurónové siete a perceptron sú bližšie popísané v kapitole 5.

4 RIEŠENÝ PROBLÉM

Ako bolo spomenuté v úvode, zadanie tejto práce vzniklo v spolupráci s firmou Kiwi.com, ktorá sa zaoberá nachádzaním a predajom leteniek. Firma zbiera údaje o letoch z rôznych leteckých spoločností a ponúka ich prehľad zákazníkom na jednej internetovej stránke. Pre zákazníka hľadajúceho najlepšiu cenu letenky odpadá nutnosť prechádzať stránky jednotlivých aerolínií.

4.1 Predstavenie problému

Keď si ľudia kúpia cez stránku firmy Kiwi.com letenku, prebieha všetka komunikácia medzi aerolíniou a zákazníkom prostredníctvom firmy. V prípade, kedy je prijatá správa od leteckej spoločnosti informujúca o zmene letu, je nutné zo strany firmy informovať o zmene zákazníka. V tomto prípade je nutné z prijatej emailovej správy získať dôležité údaje – nové údaje o lete, ktorými môžu byť nový čas, miesto, dátum odletu a mnohé ďalšie. Tieto údaje v čase písania tejto práce získava tím ľudí tým, že prečítajú správu a zadajú korektné údaje do databázy o zmenách letu. Tieto emailové správy majú často všeobecný formát a je možné tieto údaje z nich získať za pomoci regulárnych výrazov, viď kapitolu 4.2. Tento spôsob vo firme už je zavedený, avšak je náchylný na akékoľvek zmeny v štruktúre HTML kódu emailovej správy. V prípade zmeny malej časti kódu, napríklad pri zmene veľkosti písma či zrušení tučného typu písma hrozí riziko, že výraz prestane fungovať a regulárny výraz bude potrebné upraviť. Z tohoto dôvodu sa javia ako vhodný kandidát algoritmy pre rozpoznávanie pomenovaných entít, i keď nejde o ich rozpoznanie v prirodzenom texte.



Obr. 1 Nástroj vyvinutý vo firme Kiwi.com. Po označení textu v emailovej správe sa objaví lišta umožňujúca označenie dôležitých údajov príslušnou značkou.

Firmou bola poskytnutá databáza emailových správ, ktoré avšak ešte neboli označené a preto nemohli poslúžiť ako trénovacia množina. V čase písania tejto práce vznikol vo firme nástroj (viď Obr. 1), vďaka ktorému bude v blízkej dobe možné získať tieto označené emaily a na týchto trénovať ďalšie modely.

Nástroj umožňuje označenie pomenovaných entít vizuálnou formou pomocou myši. Výstupom z tohoto nástroja je HTML kód obsahujúci označené pomenované entity špeciálnymi značkami HTML kódu. V rámci tejto práce však bolo potrebné vytvoriť akýsi dočasný vývojový korpus, ktorý bude obsahovať čo najväčší počet emailových správ a bude na ňom možné otestovať a porovnať jednotlivé nástroje.

4.2 Príprava vývojového korpusu

Poskytnutá databáza emailových správ obsahuje približne sto tisíc emailových správ obsahujúce rôzne údaje od rôznych leteckých spoločností, ktoré nemajú pomenované entity nijak označené. Výhodou týchto správ avšak je, že sú si medzi sebou veľmi podobné, keďže sú vytvorené dynamicky na základe predlohy. Líšia sa napríklad len v oslovení zákazníka či čísle letu.

Ručné nahliadnutie do dát ukázalo, že správy od niektorých spoločností majú pomerne vysoké percentuálne zastúpenie v celkovom množstve správ. Keďže pre učenie jednotlivých modelov je nutné pripraviť čo najväčší trénovací korpus, javí sa za vhodné zistiť, ktoré predlohy sa v databáze vyskytujú najviac a tieto správy za pomoci regulárnych výrazov označiť.

Model bag-of-words

K zisteniu, ktoré predlohy sa vyskytujú v databáze najviac, poslúžil model *bag-of-words* (skrátene BOW), v preklade teda batoh slov. Jedná sa o spôsob reprezentácie textu, v ktorom sa počíta výskyt slov v texte. Vytvorí sa slovník, v ktorom kľúč predstavuje slovo a hodnota predstavuje koľkokrát sa dané slovo vyskytlo v dokumente. Tieto modely sú využívané napríklad v klasifikácii dokumentov do kategórií, kedy sa predpokladá, že dokumenty patriace do rovnakej kategórie budú obsahovať veľa podobných slov, viď Tab. 6.

	Bag-of-words model						
	Volám	sa	Marek	Ján	Bol	pozrieť	von
Volám sa Marek.	1	1	1	0	0	0	0
Volám sa Ján.	1	1	0	1	0	0	0
Bol sa pozrieť von.	0	1	0	0	1	1	1

Tab. 6 Príklad BOW modelu dvoch podobných viet a jednej, ktorá sa na prvé dve príliš nepodobá. V prípade podobných viet sa BOW modely líšia len na dvoch miestach, avšak pri rozdielnej vete je BOW model odlišný na piatich rôznych miestach.

Pre rozdelenie emailových správ do predlôh bol použitý nasledujúci algoritmus, kde každá emailová správa reprezentuje jeden dokument.

Pre každý dokument:

1. Vytvor BOW model.
2. Porovnaj ho so známymi BOW modelmi nasledovným spôsobom.
    a. Modely podobných emailov by mali mať podobný počet rôznych slov. Porovnaj teda dĺžku slovníku. Spočítaj počet slov v aktuálnom BOW modeli a počet slov v aktuálnom známom BOW modeli. Ak je v určenom rozmedzí medzi *lower threshold* · *length* a *upper threshold* · *length*, pokračuj do bodu b. Ak nie je v určenom rozsahu, pokračuj porovnaním s ďalším známym BOW modelom. Parameter *length* v tomto kroku reprezentuje dĺžku už známeho BOW.
    b. Modely podobných emailov by pri väčšine obsiahnutých slov mali mať rovnaký počet. V prípade, že aspoň *same count threshold* · *length* slov v modeli má rovnaký počet, vyhlás dokument za dokument rovnakej predlohy, priraď ho k aktuálnemu BOW modelu a pokračuj ďalším dokumentom.
3. Pokiaľ sa nenašiel model, ktorý bol dostatočne podobný, pridaj aktuálny BOW model do známych BOW modelov a priraď k tomu modelu aktuálny dokument.

V algoritme sú zavedené parametre *lower threshold, upper threshold* a *same count threshold*. Za pomoci týchto parametrov je možné ladiť presnosť rozpoznávania predlôh.

Parameter *lower threshold* predstavuje percentuálnu časť slov, ktoré sa musia nachádzať v oboch modeloch BOW. V prípade, že nastavíme príliš široké rozmedzie medzi *lower threshold* a *upper threshold*, môže sa stať, že dva dokumenty budú priradené do rovnakej predlohy, i keď nie sú vytvorené podľa rovnakej predlohy. V opačnom prípade, kedy nastavíme príliš úzky rozsah týchto hodnôt, bude výsledkom veľké množstvo predlôh, v krajnom prípade nová predloha pre každú emailovú správu.

Podobným spôsobom je možné voliť parameter *same count threshold*. Volíme pomocou neho minimálny pomer, v akom musia byť počty jednotlivých slov v známom a posudzovanom BOW modeli rovnaké.

Názorné príklady nevhodnej voľby parametrov *lower threshold* a *upper threshold* sa nachádzajú v Tab. 7, kde BOW model pre vetu 2 porovnávame so známym BOW modelom vety 1. *LT* predstavuje parameter *lower count threshold* a *UT* parameter *upper count threshold*. Na prvom riadku vidíme dôsledok príliš širokého rozmedzia <*LT, UT*>, kedy sú združené i vety, ktoré majú len jedno slovo spoločné. Dôsledok voľby príliš úzkeho rozmedzia vidíme na druhom riadku, kde združené neboli ani vety, ktoré sa naopak líšili len v jednom slove.

Kombinácia	Zdieľané BOW slová	Rozmedzie <LT, UT>	Rovnaká predloha podľa 2a.
Veta 1 a 3	1	<0,01, 2>	áno
Veta 2 a 3	2	<0,99, 1,01>	nie
Veta 1	Bol sa pozrieť von.		
Veta 2	Volám sa Marek.		
Veta 3	Volám sa Ján.		

Tab. 7 Príklady nevhodnej voľby parametrov lower threshold a upper threshold.

Aplikovaním algoritmu BOW s parametrami *lower threshold = 0,85*, *upper threshold = 1,15* a *same count threshold = 0,85* sa podarilo približne 100 tisíc emailových správ rozdeliť do 2557 predlôh. Z týchto 2557 predlôh bolo vybratých 20 predlôh obsahujúcich najvyšší počet emailových správ, v ktorých boli následne pomocou regulárnych výrazov označené pomenované entity. Týmto spôsobom bol získaný vývojový korpus obsahujúci približne 48 tisíc emailových správ. Korpus bolo ďalej nutné rozdeliť na časť použitú na učenie modelov a testovaciu časť pre overenie ich funkčnosti i na dátach, na ktorých neboli modely učené. Správy z prvých 16 predlôh boli použité ako trénovacie dáta – korpus *Train* a správy z ďalších 4 predlôh boli použité k vytvoreniu testovacích dát – korpusu *Test*. Korpus *Train* v sebe zahŕňa približne 35 tisíc správ a korpus *Test* ďalších 13 tisíc správ. Vzhľadom k tomu, že emailové správy obsahujú osobné údaje, nie je korpus zahrnutý ako príloha k tejto práci.

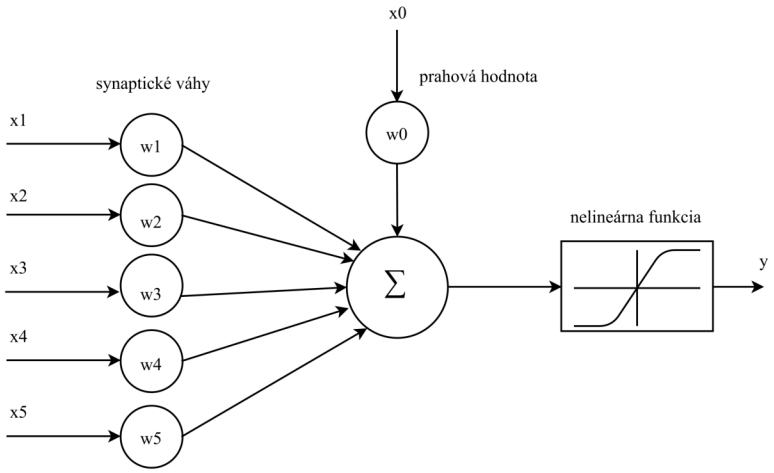
5 UMELÉ NEURÓNOVÉ SIETE

Umelé neurónové siete sa v poslednej dobe tešia veľkej popularite. I v oblasti rozpoznávania pomenovaných entít je možné ich využiť, dokonca často dosahujú podobných hodnôt úspešnosti ako iné spôsoby [4]. Umelé neurónové siete vznikli už v prvej polovici 20. storočia. Boli inšpirované biologickými neurónovými sieťami a ich podstatou je modelovanie štruktúry a činnosti biologických neurónových sietí. Základným štruktúrnym i funkčným stavebným elementom biologického informačného systému je nervová bunka – neurón. Neurónová sieť je definovaná ako súbor neurónov a spojov medzi nimi. Dôvod výberu riešenia tohoto problému i pomocou neurónových sietí je podrobnejšie popísaný v kapitole 7.

5.1 Umelý neurón

Model neurónu vytvorený v roku 1943 McCullochom a Pittsom sa dodnes používa pre bežné aplikácie. Matematický model tohoto neurónu sa skladá z troch hlavných častí. Obsahuje vstupnú, výstupnú a funkčnú časť. Vstupná časť sa skladá zo vstupov a k nim priradených synaptických váh. Na základe váhových koeficientov môžu byť jednotlivé vstupy zvýhodňované či potlačené. Nasledujúca časť je výkonná jednotka, ktorá spracuje informácie z vstupu a vygeneruje výstupnú odozvu. Tretiu časť tvorí výstupná jednotka, ktorá privádza výstupné informácie na vstup iných neurónov [15].

Na Obr. 2 je vidieť, ako pracuje jeden neurón. Vstupné hodnoty sú vynásobené príslušnými váhovými koeficientami a sčítajú sa. Na výsledok súčtu sa aplikuje nelineárna aktivačná funkcia a výsledná hodnota funkcie je privedená na vstup iných neurónov pomocou výstupnej časti. Toto sa nazýva dopredné šírenie. Na obrázku je taktiež vidieť, že neurón má jeden zvláštny vstup, ktorý nie je pripojený k výstupu žiadneho neurónu, ale privádza konštantnú hodnotu do neurónu. Táto hodnota funguje ako prahová hodnota pri aktivovaní výstupu. Keď suma váženého súčtu nepresahuje prahovú hodnotu, neurón sa neaktivuje a jeho výstup ostane nezmenený.



Obr. 2 Model umelého neurónu. Prevzaté z [16].

Matematicky môžeme výstup z neurónu popísať rovnicou (9):

$$y = F(\sum_{i=1}^{N} w_i x_i + \theta), \tag{9}$$

kde x_i ... hodnota na i-tom vstupe,
w_i ... váha i-teho vstupu,
θ ... prahová hodnota,
N ... celkový počet vstupov,
F ... transformačná (aktivačná) funkcia,
y ... hodnota výstupu neurónu.

5.2 Viacvrstvová štruktúra umelej neurónovej siete

Jediný neurón nie je schopný vykonávať príliš zložitú funkciu. Sila systému, využívajúci umelé neuróny, je v číslach, v sieti veľkého počtu neurónov. Umožňuje rôzne prepojovať vstupy a výstupy neurónov, zvýhodniť či potlačiť niektoré vstupy a minimalizovať vplyv neurónu s nesprávne nastavenými váhami na celkový výsledok.

V prípade viacvrstvovej štruktúry sú neuróny združené do vrstiev. Výstupy z vrstvy *n* sú privedené na vstup každého neurónu vo vrstve *n+1*. Prvá vrstva sa nazýva vstupná, prípadne rozdeľovacia vrstva a má za úlohu prijímať hodnoty z okolia pre spracovanie. Posledná vrstva sa nazýva výstupná a hodnoty na jej výstupe sú odozvou celého systému na vstupy z prvej vrstvy. Vrstvy medzi prvou a poslednou sa nazývajú skryté vrstvy. Ich počet závisí na zložitosti funkcie, ktorú má sieť vykonávať a na zvolenom type siete. Lineárnu funkciu AND je možné implementovať pomocou jediného neurónu [16]. Na druhej strane existujú siete obsahujúce miliardy neurónov, ktoré slúžia k rozpoznaniu objektov, asistovanému riadeniu a ďalším úlohám, ktoré sú náročné i pre ľudí.

Nutnosťou k naučeniu neurónovej siete je takzvaná trénovacia množina, ktorá obsahuje prvky popisujúce riešenú problematiku. Formálne ju môžeme definovať ako množinu prvkov (vzorov), ktoré sú definované ako usporiadané dvojice nasledujúcim spôsobom:

$$T = \left\{\{I_1, O_1\}\{I_2, O_2\} \dots \{I_p, O_p\}\right\},$$
$$I_i = [i_1\ i_2 \dots i_k], \qquad i_j \in \langle 0, 1\rangle,$$
$$O_i = [o_1\ o_2 \dots o_l], \qquad o_j \in \langle 0, 1\rangle,$$

kde p ... počet vzorov trénovacej množiny,
I_i ... vektor excitácií vstupnej vrstvy tvorenej *k* neurónmi,
O_i ... vektor excitácií výstupnej vrstvy tvorenej *l* neurónmi,
i_j, o_j ... excitácie *j-teho* neurónu vstupnej resp. výstupnej vrstvy.

V dnešnej dobe najpoužívanejšou metódou, ktorá umožňuje adaptáciu neurónovej siete na danú trénovaciu množinu, sa nazýva *backpropagation*, čo v preklade znamená metódu spätného šírenia. Na rozdiel od dopredného chodu pri šírení signálu neurónovej siete táto metóda adaptácie spočíva v opačnom šírení informácie smerom od vrstiev vyšších k vrstvám nižším. Jedná sa o nasledovný postup:

1. Použijeme prvky vektoru I_i i-teho prvku, ktorým excitujeme neuróny vstupnej vrstvy na odpovedajúcu úroveň.
2. Prevedieme dopredné šírenie tohoto signálu až k výstupnej vrstve neurónovej siete.
3. Porovnáme požadovaný stav daný vektorom O_i i-teho prvku s odozvou neurónovej siete.
4. Rozdiel medzi skutočnou a požadovanou odozvou definuje chybu neurónovej siete. Túto chybu potom v určitom pomere – *learning rate* – vraciame späť do neurónovej siete formou úprav synaptických váh medzi jednotlivými vrstvami smerom od horných vrstiev k nižším tak, aby chyba pri nasledujúcej odozve bola menšia.
5. Po vyčerpaní celej trénovacej množiny sa vyhodnotí celková chyba cez všetky vzory trénovacej množiny a ak je vyššia než požadovaná, celý proces sa opakuje znovu.

Chybu z bodu 3 je možné definovať rôznymi spôsobmi. Nazýva sa i chybovou funkciou a často je vhodné podľa riešeného problému voliť chybovú funkciu. Podstata metódy backpropagation spočíva v hľadaní minima chybovej funkcie E definovanej napríklad podľa (10) :

$$E = \frac{1}{2}\sum_{i=1}^{p}\sum_{j=1}^{m}(y_j - o_j)^2, \qquad (10)$$

kde y_j ... skutočná odozva j-teho neurónu výstupnej vrstvy,
o_j ... požadovaná odozva j-teho neurónu výstupnej vrstvy daná vzorom,
p ... celkový počet vzorov trénovacej množiny,
m ... počet neurónov výstupnej vrstvy.

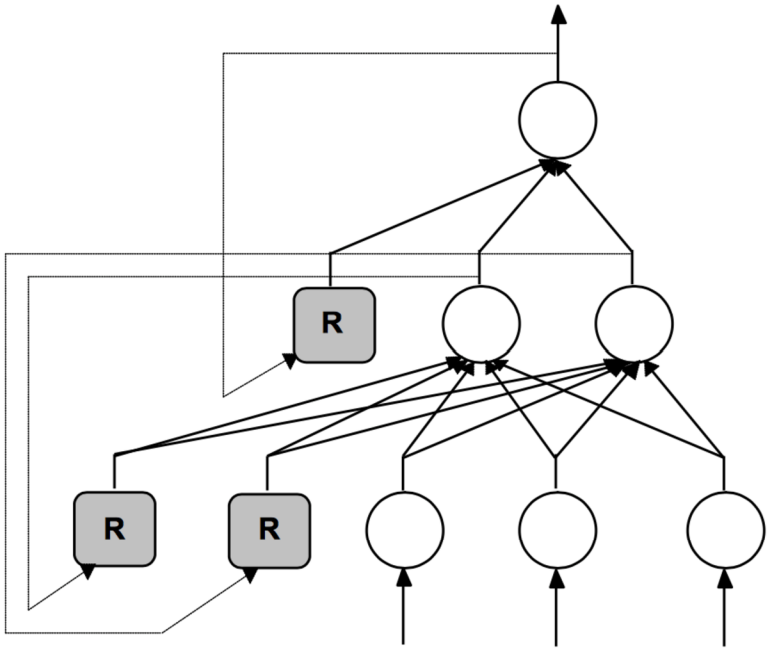
Cesta, ktorou je možné chybovú funkciu minimalizovať je práve úpravou synaptických váh medzi neurónmi *i* a *j* podľa formule (11):

$$\Delta w_i = -\eta\frac{\partial E}{\partial w_i} + \mu\Delta w_i', \qquad (11)$$

kde η ... koeficient učenia,
μ ... koeficient vplyvu zmeny váh z predchádzajúceho kroku,
$\Delta w_i'$ ... zmena synaptickej váhy z predchádzajúceho kroku.

5.3 Rekurentné neurónové siete

Neurónové siete popísane v predchádzajúcej podkapitole boli nezávislé na kontexte. To znamená, že rovnaký vstupný stimul vedie vždy na rovnakú odozvu siete. V tejto kapitole sú predstavené rekurentné neurónové siete, ktorých cieľom je do siete implementovať časový kontext. Inak povedané, odozva siete nebude daná len aktuálnym vstupným stimulom, ale bude ovplyvnená i stimulmi, ktoré aktuálnemu predchádzali. Takéto chovanie je vhodné pre úlohy predpovedania sekvencií, akými sú napríklad rozpoznávanie písma, hlasu a úlohy v oblasti spracovávania prirodzeného textu.



Obr. 3 Rekurentná viacvrstvová neurónová sieť. Prevzaté z [17].

Topológia rekurentnej siete sa líši od klasickej ďalšími rekurentnými neurónmi (viď neuróny označené písmenom R na Obr. 3). V strednej vrstve je jeden neurón odpovedajúci jednému neurónu výstupnej vrstvy a vo vstupnej vrstve sú dva rekurentné neuróny odpovedajúce dvom neurónom vnútornej vrstvy. Trénovacia množina je tvorená sekvenciami vzorov. Príkladom môžu byť sekvencie uvedené v Tab. 8.

Vstupná sekvencia					**Výstupná (cieľová) sekvencia**				
Vzor 1		Vzor 2		Vzor 3	Vzor 1		Vzor 2		Vzor 3
1 0 0	→	0 1 0	→	0 0 1	0.25	→	0.0	→	1.0
0 0 1	→	0 1 0	→	1 0 0	0.5	→	1.0	→	0.75

Tab. 8 Príklad sekvencie vzorov ako vstup do rekurentnej viacvrstvovej neurónovej siete. Prevzaté z [17].

Z Tab. 8 je zrejmé, že bez vedomosti o predchádzajúcom stave nie je možné správne určiť výstupy. Algoritmus adaptácie váh prebieha mierne iným spôsobom. Rekurentná sieť

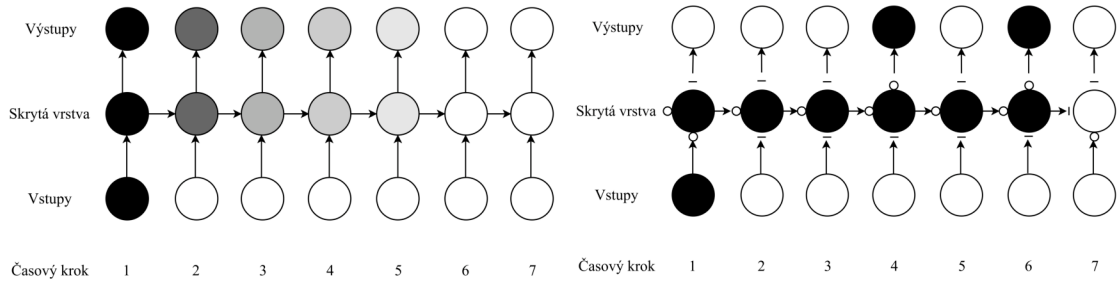
je rozložená do postupnosti obyčajných viacvrstvových sietí, kde každý „časový krok“ predstavuje novú vrstvu. Algoritmus implementujúci tento spôsob učenia sa nazýva *backpropagation through time* (BPTT) a má nasledujúce štyri kroky:

1. Prevedie sa dopredné šírenie signálov pre všetky vzory sekvencie.
2. Vypočítajú sa chyby neurónov vonkajšej a ďalej predchádzajúcich nižších vrstiev, analogicky s metódou backpropagation s tým, že najprv sa tieto chyby stanovia pre posledný časový krok a následne pre kroky predchádzajúce, až po prvý krok.
3. Vypočítané chyby z predchádzajúceho kroku algoritmu sa použijú k stanoveniu zmien váh, ktoré sa akumulujú pre každú jednotlivú väzbu a každý jednotlivý časový krok.
4. Váhy všetkých väzieb sa aktualizujú prostredníctvom hodnôt akumulovaných zmien váh vypočítaných v bode 3.

Problém rekurentných neurónových sietí vyplýva však z podstaty ich vytvorenia a spôsobu učenia – algoritmu BPTT. Rekurentné siete sú totiž len prevedené na obyčajné viacvrstvové neurónové siete, v ktorých je pre každý „časový krok“ vytvorená nová vrstva. Problém týchto sietí sa nazýva problém miznúceho gradientu (*vanishing gradient problem*) a jemu podobný problém vybuchujúceho gradientu (*exploding gradient problem*). Tradičné aktivačné funkcie jednotlivých neurónov majú gradienty v rozmedzí (-1, 1) a algoritmus backpropagation počíta gradienty pomocou retiazkového pravidla, čo znamená násobenie n čísiel v tomto rozmedzí v prípade n-vrstvovej siete. Chybové signály, ktoré „idú späť v čase“ majú tendenciu buď vybuchnúť (1) alebo zmiznúť (2) [18]. Prípad (1) môže viesť k oscilácii váh a prípad (2) spôsobí, že učenie trvá príliš dlho, prípadne vôbec nefunguje. Riešení tohoto problému je viacero, v prípade rekurentných neurónových sietí sa osvedčili varianty pokročilej architektúry LSTM a GRU.

5.4 Neurónové siete typu LSTM

V roku 1997 bola predstavená nová architektúra sietí a algoritmus pre ich učenie – takzvané *Long Short-Term Memory* (LSTM) siete. LSTM siete patria do skupiny rekurentných neurónových sietí, majú avšak ďaleko zložitejšiu štruktúru jednotlivých neurónov. Predstavujú spôsob ako sa vysporiadať s problémom miznúceho gradientu (viď Obr. 4), ich štruktúra obsahuje prvky určené k zachovaniu dlhodobej informácie.



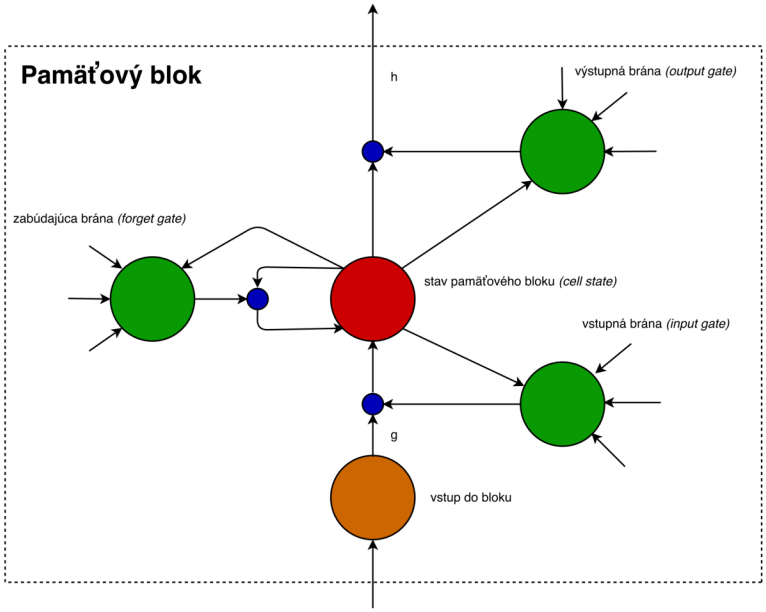
(a) (b)

Obr. 4 (a) Problém miznúceho gradientu a straty kontextu. (b) Ukážka zachovania informácie o kontexte v sieti typu LSTM. Prevzaté z [19].

LSTM vrstvy sa skladajú z pamäťových blokov, štruktúra pamäťového bloku je zobrazená na Obr. 5. Jednotlivé prvky pamäťového bloku plnia nasledujúce funkcie:

- vstupná brána (*input gate*)
  - kontroluje priechod informácie do pamäťového bloku,
- stav pamäťového bloku (*cell state*)
  - uchováva dlhodobú informáciu,
- zabúdajúca brána (*forget gate*)
  - určuje množstvo uchovávaného signálu,
- výstupná brána (*output gate*)
  - určuje množstvo propagovaného signálu.

Obr. 5 znázorňuje pamäťový blok LSTM neurónu a jeho jednotlivé časti.



Obr. 5 Pamäťový blok LSTM neurónu. Jednotky označené modrou farbou sú multiplikatívne jednotky. Písmená g, h reprezentujú aplikácie nelineárnej funkcie. Prevzaté z [20].

Dopredná propagácia signálu a spätná propagácia chyby

Rovnice (12) až (20) sú prevzaté z [20] a predstavujú propagáciu signálu v rámci jedného pamäťového bloku. Ako i v prípade ostatných typov rekurentných neurónových sietí sa v prípade, že informácie z predošlého kroku nie sú k dispozícii, nahrádzajú vektorom núl. Horný index jednotlivých premenných predstavuje časový úsek, dolné indexy ι, ϕ, ω, c po rade reprezentujú príslušnosť váh spojenia prípadne aktivácií k vstupnej bráne, zabúdajúcej bráne, výstupnej bráne a stavom pamäťového bloku. Premenné I, H, C reprezentujú po rade celkové množstvo vstupov danej vrstvy, výstupov danej vrstvy a množstvo stavových buniek v popisovanom pamäťovom bloku. Premenná w_{ij} predstavuje váhy spojenia jednotky i s jednotkou j, x_i^t je vstup pamäťového bloku na pozícii i v čase t, vstupná suma modulu j pamäťového bloku v čase t sa označuje ako a_j^t, niekedy sa tiež označuje ako preaktivácia, po aktivácii b_j^t. b_c^t reprezentuje výstup pamäťového bloku, žiadnym iným spôsobom sa signál nepropaguje do nasledujúcich vrstiev. Funkcia f je aktivačnou funkciou brán, funkcie g a h sú vstupne výstupnými aktiváciami pamäťového bloku.

LSTM siete náležia do skupiny rekurentných neurónových sietí a preto sa pre ich učenie často používa algoritmus *backpropagation through time* (BPTT), kde dochádza k spätnému šíreniu chyby do nasledujúcej vrstvy i do predchádzajúceho časového kroku, ako je popísané v predchádzajúcej podkapitole. LSTM siete sa však vyhýbajú problému miznúceho a vybuchujúceho gradientu vďaka svojim bránam, ktoré je možné podľa potreby zavrieť či otvoriť.

Vstupná brána (*input gate*)

$$a_\iota^t = \sum_{i=1}^{I} w_{i\iota} x_i^t + \sum_{h=1}^{H} w_{h\iota} b_h^{t-1} + \sum_{c=1}^{C} w_{c\iota} s_c^{t-1} \tag{12}$$

$$b_\iota^t = f(a_\iota^t) \tag{13}$$

Zabúdajúca brána (*forget gate*)

$$a_\phi^t = \sum_{i=1}^{I} w_{i\phi} x_i^t + \sum_{h=1}^{H} w_{h\phi} b_h^{t-1} + \sum_{c=1}^{C} w_{c\phi} s_c^{t-1} \tag{14}$$

$$b_\phi^t = f(a_\phi^t) \tag{15}$$

Stav pamäťového bloku (*cell state*)

$$a_c^t = \sum_{i=1}^{I} w_{ic} x_i^t + \sum_{h=1}^{H} w_{hc} b_h^{t-1} \tag{16}$$

$$s_c^t = b_\phi^t s_c^{t-1} + b_\iota^t g(a_c^t) \tag{17}$$

Výstupná brána (*output gate*)

$$a_\omega^t = \sum_{i=1}^{I} w_{i\omega} x_i^t + \sum_{h=1}^{H} w_{h\omega} b_h^{t-1} + \sum_{c=1}^{C} w_{c\omega} s_c^t \tag{18}$$

$$b_c^t = f(a_\omega^t) \tag{19}$$

Výstup pamäťového bloku

$$b_c^t = b_\omega^t h(s_c^t) \tag{20}$$

5.5 Neurónové siete typu GRU

V roku 2014 bol predstavený nový typ siete podobný LSTM [21]. Jeho úlohou je znížiť počet parametrov, ktoré obsahuje LSTM blok. Tento nový typ je nazývaný podľa jeho štruktúry *Gated Recurrent Unit* (GRU). Jeden blok GRU obsahuje dve brány, *r* (resetovaciu) a *z* (aktualizujúcu). Jednotlivé časti bloku GRU plnia nasledujúce funkcie:

- aktuálna aktivácia *h*
  - je lineárnou interpoláciou medzi predchádzajúcou aktiváciou a kandidátskou aktiváciou,
- kandidátska aktivácia $\tilde{h}$
  - je kandidátom na ďalšiu aktiváciu *h*,
- resetovacia brána (*reset gate*) *r*
  - plní podobnú úlohu ako zabúdajúca brána v LSTM bloku,
- aktualizujúca brána (*update gate*) *z*
  - určuje ako veľmi jednotka zmení svoju aktiváciu.

K výpočtu hodnôt uvedených častí slúžia rovnice (21) až (24).

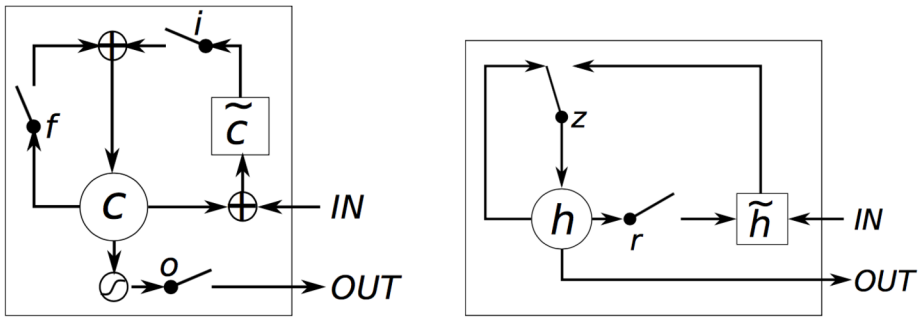
$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{21}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{22}$$

$$\tilde{h}_t = \tanh(W x_t + U(r_t \cdot n_{t-1})) \tag{23}$$

$$h_t = (1 - z_t) \cdot n_{t-1} + z_t \tilde{h}_t \tag{24}$$

Bloky GRU neobsahujú pamäťové bunky a počet parametrov, ktoré obsahujú jednotlivé bloky, je oproti LSTM výrazne nižší, viď Obr. 6.



(a) Long Short-Term Memory (b) Gated Recurrent Unit

Obr. 6 Rozdiel medzi LSTM a GRU. (a) i, f a o predstavujú vstupnú (input), zabúdaciu (forget) a výstupnú (output) bránu. c a c̃ reprezentujú aktuálny stav pamäťového bloku a nový stav pamäťového bloku. (b) r a z reprezentujú resetovaciu (reset) a aktualizujúcu (update) bránu. h a h̃ je aktivácia a kandidátska aktivácia. Prevzaté z [21].

6 MODELY CONDITIONAL RANDOM FIELDS (CRF)

CRF (*conditional random fields*) predstavujú pravdepodobnostný rámec pre označovanie a segmentáciu štruktúrovaných dát, akými sú napríklad sekvencie. Základná myšlienka spočíva v definovaní podmieneného rozdelenia pravdepodobnosti nad sekvenciami značiek. Hlavnou výhodou, ktorú majú CRF voči skrytým Markovovým modelom je ich podmienená povaha, čo má za následok uvoľnenie predpokladov nezávislosti vyžadovanými skrytým Markovovým modelom.

V prípade spracovania prirodzeného jazyka je dokument rozdelený na sekvenciu tokenov a k nim príslušných značiek. Do modelu však nevstupujú tokeny samotné, ale pre každý token sa vytvorí takzvaný zoznam príznakov. Takýmito príznakmi môžu byť napríklad dĺžka slova, predchádzajúce slovo či informácia o tom, či dané slovo obsahuje číslice. Do modelu tak vstupuje sekvencia zoznamov príznakov, z ktorých model určí sekvenciu značiek.

CRF modely sú používané aj pre iné úlohy ako rozpoznávanie pomenovaných entít, akými sú napríklad počítačové videnie [22] či spracovanie biologických sekvencií [23]. Dôvod použitia CRF modelov pre túto konkrétnu úlohu je bližšie popísaný v kapitole 7.

6.1 Definícia

Ako je popísané v [24], CRF je možné popísať ako grafový pravdepodobnostný model, v ktorom každý uzol (v našom prípade slovo, prípadne token) popisuje náhodnú premennú. Nech $x_{1:N}$ sú pozorovania (tj. slová či tokeny v dokumente) a $z_{1:N}$ sú značkami. Lineárny CRF model definuje podmienenú pravdepodobnosť podľa vzorca (25) [25].

$$p(z_{1:N}|x_{1:N}) = \frac{1}{Z}\exp(\sum_{n=1}^{N}\sum_{i=1}^{F}\lambda_i f_i(z_{n-1}, z_n, x_{1:N}, n)) \tag{25}$$

Skalár Z je normalizačný faktor, ktorý je definovaný pomocou vzorca (26) [25].

$$Z = \sum_{z_{1:N}}\exp(\sum_{n=1}^{N}\sum_{i=1}^{F}\lambda_i f_i(z_{n-1}, z_n, x_{1:N}, n)) \tag{26}$$

λ a $f()$ môžu nadobúdať ľubovoľných reálnych hodnôt a celá exponenciálna funkcia bude mať nezápornú hodnotu. V rámci exponenciálnej funkcie sčítavame cez hodnoty $n=1, \ldots N$, ktoré predstavujú pozície slov v dokumente. Pre každú pozíciu sčítavame cez hodnoty $i = 1, \ldots, F$, ktoré predstavujú vážené príznakové funkcie (*feature functions*) (viď kapitola 6.2). Skalár λ_i je váhou pre príznak $f_i()$. Parametrami, ktoré sa musí model naučiť, sú skaláre λ_i [25].

6.2 Príznakové funkcie

Príznakové funkcie sú najdôležitejšou časťou CRF modelov. V prípade lineárneho CRF modelu má príznaková funkcia všeobecný tvar $f_i(z_{n-1}, z_n, x_{1:N}, n)$, kde je braný ohľad na pár susediacich stavov z_{n-1}, z_n, celú sekvenciu vstupov $x_{1:N}$ a pozíciu n, na ktorej sa v dokumente nachádzame. Môžeme si napríklad definovať takúto jednoduchú príznakovú funkciu (27):

$$f_1(z_{n-1}, z_n, x_{1:N}, n) = \begin{cases} 1 \; ak \; z_n = MESTO \; a \; x_{n-1} = Nove \\ 0 \;\; v \; ostatných \; prípadoch \end{cases} \tag{27}$$

Táto príznaková funkcia v prípade kladnej nastavenej váhy λ_1 a splnených podmienok $z_n = MESTO$ a $x_{n-1} = Nove$ zvyšuje pravdepodobnosť značkovej sekvencie $z_{1:n}$. Inak povedané, model bude preferovať značku *MESTO* pre slovo, ktoré nasleduje po slove *Nove*. V prípade, že $\lambda_1 < 0$, CRF model sa bude snažiť vyhnúť značke *MESTO* pre slovo, ktoré nasleduje po slove *Nove*. Nastavenie váh λ_i sa model naučí z trénovacích dát, ako je popísané v ďalšej podkapitole. Ďalším príkladom môže byť nasledujúca príznaková funkcia (28).

$$f_2(z_{n-1}, z_n, x_{1:N}, n) = \begin{cases} 1 \; ak \; z_n = MESTO \; a \; x_{n+1} = nad \\ 0 \;\; v \; ostatných \; prípadoch \end{cases} \tag{28}$$

Táto príznaková funkcia je aktívna, keď aktuálna značka je *MESTO* a ďalšie slovo je *nad*, ako napríklad v prípade mesta *Moldava nad Bodvou*. V prípade vety obsahujúca „*Nové Mesto nad Váhom*“, kde môžu byť aktívne obe príznakové funkcie f_1 i f_2, hovoríme o prekrývaných príznakoch (*overlapping features*). Toto vylepší dôveru v $z_2 = MESTO$ na $\lambda_1 + \lambda_2$ [25].

6.3 Učenie CRF modelu

Pojmom učenie CRF modelu je myslené nájdenie vhodných parametrov λ_i. Pre toto potrebujeme označené sekvencie $\{(\boldsymbol{x}^{(1)}, \boldsymbol{z}^{(1)}), \ldots (\boldsymbol{x}^{(m)}, \boldsymbol{z}^{(m)})\}$, kde $\boldsymbol{x}^{(1)} = \boldsymbol{x}^{(1)}_{1:N_1}$. Keďže CRF definujú podmienenú pravdepodobnosť $p(\boldsymbol{z}|\boldsymbol{x})$, maximalizujeme podmienenú pravdepodobnosť trénovacích dát určenú vzorcom (29):

$$\sum_{j=1}^{m} \log p(\boldsymbol{z}^{(j)}|\boldsymbol{x}^{(j)}). \tag{29}$$

Táto účelová funkcia je konkávna, a preto štandardným postupom pre učenie je spočítanie gradientu a jeho využitie v optimizačnom algoritme, akým je napríklad L-BFGS [25].

7 VLASTNÉ RIEŠENIE

Keďže sa táto práca zaoberá rozpoznávaním pomenovaných entít v emailových správach a existujúce nástroje sú určené k rozpoznávaniu entít v prirodzenom texte, vznikla potreba overiť funkčnosť existujúcich nástrojov i mimo ich predurčenú doménu. Problémy, ktoré môžu nastať pri používaní nástrojov na inú činnosť ako nimi predpokladanú sú rôzne.

Najviac očividným je veľká odlišnosť štruktúry prirodzeného textu a HTML kódu. Existujúce nástroje dokumenty prevedú na postupnosť tokenov. K tomu používajú tokenizéry (viď kapitola 2.2), ktoré sú špeciálne navrhnuté tak, aby tokenizovali dokument na úrovni slov. Väčšinou sa tak deje na základe bielych znakov a interpunkčných znamienok. V prípade HTML kódu, kde je čitateľný text prepletený s (často dlhými) časťami kódu HTML, by toto mohlo viesť k drastickému zníženiu úspešnosti.

Ďalším problémom je skutočnosť, že emailové správy, na ktorých budú nástroje testované, obsahujú rôzne identifikačné údaje, čísla letov, časové údaje a podobne. Z hľadiska prirodzeného jazyka je teda nutné i textovú časť správ považovať za prirodzený jazyk o vysokej (morfologickej) zložitosti, keďže tieto kódy obvykle nemajú spoločnú lemmu. Pri takýchto morfologicky zložitých jazykoch sa osvedčilo analyzovať text znak po znaku pomocou rekurentnej neurónovej siete [4]. Takýto spôsob by teoreticky mohol viesť k tomu, že sa model naučí napríklad to, že v prípade niekoľkých čísel a veľkých písmen pri sebe je veľká pravdepodobnosť, že sa jedná o číslo letu a podobne. Ďalšou technikou, ktorá bola použitá v rámci tejto práce je algoritmus na základe CRF (*conditional random fields*). Tento spôsob bol zvolený kvôli výborným dosahovaným výsledkom v oblasti všeobecného rozpoznávania pomenovaných entít [26][27]. V prípade CRF je ale nutné vytvoriť sadu príznakov, viď kapitola 6. V prípade neurónových sietí predpokladáme, že sa jednotlivé príznaky potrebné k správnej klasifikácii naučí sama.

Prvý nástroj na báze rekurentnej neurónovej siete je implementovaný za pomoci knižnice *Keras* v jazyku Python 3 a druhý nástroj na báze CRF taktiež v jazyku Python 3, za pomoci knižnice *pycrfsuite*. K týmto nástrojom bolo vzhľadom k odlišnej povahe HTML kódu od prirodzeného textu nutné vytvoriť špeciálny tokenizér HTML kódu, ktorý je zdieľaný oboma nástrojmi.

7.1 Tokenizér HTML kódu

Tokenizéry sú nástroje, ktoré delia dokument na menšie časti - tokeny, väčšinou sa jedná o slová v texte. Tak napríklad vetu „*Idem do školy*“ je možné rozdeliť na slová *Idem, do* a *školy*. Najjednoduchší spôsob tokenizácie je oddeľovanie slov na základe bielych znakov, tj. na základe medzier, nových riadkov a podobne. Niekedy však toto nestačí a je potrebné oddeľovať slová i na základe spojovníkov, interpunkčných znamienok a iných znakov.

Vstupnými dátami pre algoritmy v tejto práci sú emailové správy obsahujúce označené pomenované entity. V prípade použitia obyčajných tokenizérov dochádza k nesprávnemu chovaniu, keďže tieto tokenizéry spracovávajú na základe svojich pravidiel i časti správy, ktoré nie sú prirodzeným textom – časti HTML kódu. Z tohoto dôvodu bolo nutné implementovať tokenizér, ktorý by si s takouto kombináciou HTML kódu a prirodzeného textu vedel poradiť. V označených emailových správach sa okrem bežných značiek HTML kódu nachádzajú i značky označujúce pomenované entity. Toto robí algoritmus mierne zložitejším, keďže je nutné skontrolovať nielen, či sa jedná o značku HTML kódu, ale v prípade že áno, je nutné skontrolovať, či sa nejedná o jednu zo značiek pomenovaných entít. Výstupom tohoto HTML tokenizéru je zoznam dvojíc, kde prvú časť tvorí token a druhú časť tvorí označenie pomenovanej entity. V prípade, že token nepredstavuje pomenovanú entitu, označí sa špeciálnym označením pre neentitu, napríklad písmenom O (*other*).

Za jeden token tento algoritmus považuje nasledovné celky.

- Znak v časti kódu, ktorý nie je súčasťou HTML značky, kde výnimku tvoria znaky, ktoré sú súčasťou značiek *script* a *style*.
- Jedna HTML značka, ktorý začína znakom < a končí znakom >.

Ďalšie kroky, ktoré algoritmus prevádza vznikli počas jeho prvých chodov. Ukázalo sa, že je potrebné zo správ odstrániť URL adresy, keďže často boli pre každú správu unikátne a toto by viedlo k zbytočne veľkému množstvu rôznych tokenov.

Základný algoritmus HTML tokenizéru je možné zhrnúť do nasledujúceho zjednodušeného pseudokódu, pričom sa v jednotlivých emailových správach postupuje znak po znaku. Ozajstný algoritmus je o niečo zložitejší a je dostupný v prílohe A.

Algoritmus HTML tokenizéru:

```
chars_to_write = ‘’
for char in message:
    if char == ‘<’:
        môže sa jednať o značku HTML kódu alebo označenie pomenovanej
        entity

        if chars_to_write obsahuje znaky
            zapíš ich ako neentitu, aby reťazec bol znova prázdny

        identifikuj aktuálnu značku
        start = ‘’

        while char != ‘>’:
            start = start + char
            char = next char

        identifikátor aktuálnej značky predstavuje prvé slovo reťazca start
        tag_name = start.split()[0]

        if je tag_name pomenovanou entitou
            prechádzaj znaky, kým nepríde koniec značky pomenovanej
            entity v tvare </tag_name> a zapíš obsah ako priradenú
            pomenovanú entitu

        else if je tag_name rovný script alebo je tag_name rovný style
            prechádzaj znaky,  kým nepríde koniec tejto značky v
            tvare </script> či </style> a následne zapíš celú značku ako
            neentitu

        else
            pridaj celý reťazec start ako neentitu do výstupu
    else:
        nejedná sa o súčasť značky HTML kódu ani označenie pomenovanej
        entity,
        chars_to_write = chars_to_write + char

if reťazec chars_to_write nie je na konci algoritmu prázdny
    zapíš ich ako neentitu
```

7.2 Implementácia neurónovej siete

Neurónová sieť bola implementovaná v jazyku Python 3 za pomoci nástroja *Keras*. *Keras* je nadstavbou pre knižnice zaoberajúce sa neurónovými sieťami *TensorFlow* a *Theano*. Táto podkapitola ponúka prehľad hlavných funkčných častí riešenia. Riešenie je rozdelené do nasledujúcich hlavných súborov:

- *html_tokenizer.py*
  Obsahuje HTML tokenizér popísaný v predchádzajúcej podkapitole potrebný pre prevod emailových správ na postupnosť tokenov.
- *config.ini*
  Jedná a o konfiguračný súbor, v ktorom je možné nastaviť všetky parametre neurónovej siete.
- *neuralner.py*
  Obsahuje implementáciu neurónovej siete i s ďalšími pomocnými metódami.
- *requirements.txt*
  Obsahuje zoznam balíkov jazyku Python, ktoré sú potrebné nainštalovať pre správnu funkciu programu.
- *server.py*
  Serverová časť, ktorá umožňuje použiť naučenú neurónovú sieť v serverovom režime.

Konfiguračný súbor config.ini

Implementovaná neurónová sieť obsahuje pomerne veľké množstvo parametrov, ktoré je možné nastaviť. Aby nebolo nutné pri každom spúšťaní zadávať všetky parametre, sú všetky parametre načítané z jediného konfiguračného súboru. Môže sa zdať, že parametrov v nasledujúcej tabuľke je veľký počet, no niektoré parametre sú odvodzované automaticky a nie je nutné ich upravovať. Takýmto parametrom je napríklad *DATA_DIR*, ktorý automaticky nastaví adresár obsahujúci trénovacie dáta na adresár *data* v adresári *ROOT_DIR*. Parametre sú zoskupené podľa typu a ich významy sú uvedené v Tab. 9.

Konfiguračný súbor je načítaný pomocou funkcie *parse_config*. Táto funkcia zo súboru vytvorí slovník kompatibilný s funkciami neurónovej siete. Používa k tomu Python modul ConfigParser s pokročilou interpoláciou ExtendedInterpolation, ktorá podporuje predávanie informácií medzi časťami konfiguračného súboru. Ako už bolo spomenuté, niektoré parametre v konfiguračnom súbore sú vytvorené automaticky kombináciou iných. Tieto sa v Tab. 9 pre prehľadnosť nenachádzajú.

Názov	Význam
ROOT_DIR	Cesta k adresáru obsahujúci celý projekt.
DATA_FILENAME	Názov súboru obsahujúci trénovacie dáta.
ENTITIES	Názvy rozpoznávaných pomenovaných entít, oddelené čiarkami.
TRAIN_SPLIT	Číslo medzi 0 a 1 udávajúce pomer správ, aký má byť použitý pre učenie.
VAL_SPLIT	Číslo medzi 0 a 1 udávajúce pomer správ, aký má byť použitý pre validáciu počas učenia.
EVAL_SPLIT	Číslo medzi 0 a 1 udávajúce pomer správ, aký má byť použitý pre overenie výsledkov po učení.
RANDOMIZE_ORDER	Hodnota typu boolean. Udáva, či majú byť dáta pred učením náhodne pomiešané.
EVALUATE	Hodnota typu boolean. Udáva, či po učení má prebehnúť vyhodnotenie naučeného modelu.
CHAR_BY_CHAR	Hodnota typu boolean. Udáva, či má HTML tokenizér spracovávať viditeľný text znak po znaku alebo slovo po slove (oddelené medzerami).
MIN_TOKEN_COUNT	V prípade, že počet výskytov tokenu je nižší ako tento parameter, bude každý takýto token považovaný za rovnaký špeciálny neznámy token.
HOST	IP adresa hostu v tvare x.x.x.x., na ktorom má server bežať.
PORT	Port, na ktorom má server bežať.
FIRST_LAYER	Typ vstupnej vrstvy neurónovej siete. Má dve možnosti, Embedding a Dense.
DENSE_SIZE	Počet neurónov vo vstupnej vrstve v prípade voľby vstupnej vrstvy typu Dense.
EMBED_SIZE	Počet neurónov vo vstupnej vrstve v prípade voľby vstupnej vrstvy typu Embedding.
MAX_LOOKOUT	Počet neurónov v skrytej, obojsmernej rekurentnej vrstve.
REC_LAYER_TYPE	Typ obojsmernej rekurentnej vrstvy. Na výber je SimpleRNN, LSTM a GRU.
NO_OF_REC_LAYERS	Počet rekurentných vrstiev.
BATCH_SIZE	Veľkosť dávky, ktorú spracuje neurónová sieť. V prípade veľkej siete a malého množstva dostupnej pamäti je možné, že učenia nebude mať dostatok pamäti. V tom prípade je vhodné túto hodnotu zmenšovať.
EPOCHS	Počet epoch, ktorý má učenie previesť.
TB_LOGGING	Hodnota typu boolean. Udáva, či sa majú niektoré hodnoty počas učenia zapisovať do logov nástroju TensorBoard.

Tab. 9 Popis parametrov v konfiguračnom súbore pre nástroj NeuralNER.

Prevod tokenov a názvov entít na číselné vstupy a výstupy neurónovej siete

Tokeny i názvy pomenovaných entít sami o sebe sú reťazce znakov. Tieto reťazce je nutné nejakým spôsobom previesť na čísla, aby bolo možné ich použiť ako vstupy či výstupy neurónovej siete. Často používaným spôsobom je kódovanie nazývané *one-hot encoding*.

Začne sa prevodom tokenov. Na začiatku učenia sa vytvorí slovník tokenov vyskytujúcich sa v trénovacom súbore, ktorý obsahuje tokeny ako kľúče a indexy k nim ako hodnoty. Tento slovník budeme nazývať *token2ind*. Vytvorí sa k nemu i opačný slovník na dekódovanie číselných hodnôt späť na tokeny, tento budeme nazývať *ind2token*. Ako vstup do neurónovej siete bude slúžiť matica *max_len* x *vocab_size*, kde *max_len* predstavuje najvyšší počet tokenov, aký bol v akomkoľvek dokumente zaznamenaný a *vocab_size* predstavuje veľkosť vytvoreného slovníka, teda počet rôznych tokenov. Matica bude obsahovať nuly. Následne pre každý riadok nastavíme hodnotu na 1 pre stĺpec, ktorý reprezentuje reťazec v riadku v slovníku *token2ind*.

Analogicky vytvoríme slovníky i pre názvy pomenovaných entít, ktoré pomenujeme *label2ind* a *ind2label*. Výstupom z neurónovej siete bude matica o rozmeroch *max_len* · *no_of_labels*. Premenná *no_of_labels* predstavuje veľkosť vytvoreného slovníka, teda počet rôznych pomenovaných entít + jednu neentitu. Celý proces tvorby týchto slovníkov sa nachádza v triede NeuralNER vo funkcii *_build_encodings*.

Trieda MetricsCalculator

Na vyhodnotenie úspešnosti bola vytvorená trieda *MetricsCalculator*, ktorá je napojená prostredníctvom takzvaných *callbacks* do procesu učenia. Po každom kroku učenia spočíta metriku F1 pre trénovací a validačný dataset a výsledky zobrazí v konzole. Po celkovom učení je možné túto triedu využiť taktiež na spočítanie metrík pre vyhodnocovaciu sadu, definovanú pomerom *EVAL_SPLIT* v konfiguračnom súbore. Toto sa udeje automaticky pokiaľ je hodnota *EVALUTE* nastavená na *True*. Táto metóda využíva knižnicu *scikit-learn*.

Funkcia generate_tuples_from_file

Táto funkcia bola vytvorená z praktického dôvodu – veľké množstvo vstupov zaberá veľké množstvo pamäte. Nástroj *Keras* umožňuje počas procesu učenia použiť takzvané generátory. Tieto generátory sú funkcie, ktoré postupne vracajú výsledky a teda nemusia vrátiť všetky naraz. Je teda možné otvoriť súbor, načítať prvý riadok do pamäte, spracovať prvý riadok a po skončení spracovania zabudnúť na prvý riadok a pokračovať druhým a takýmto spôsobom až po koniec súboru, kedy v jednom momente máme v pamäti načítaný len jeden riadok. Miesto *return* sa v generátore používa príkaz *yield*.

Táto funkcia načítava emailové správy jednu po druhej, vytvorí z nich vstupy pre neurónovú sieť a postupne ich predáva. Nie je nutné predávať správy len po jednej, je

možné nastaviť počet správ v dávke pomocou parametru *BATCH_SIZE*, ktorý sa dá nastaviť v konfiguračnom súbore. Tento prístup má nevýhodu v tom, že prístup je pomalší ako v prípade prístupu k správam priamo v pamäti. Táto nevýhoda je ale z veľkej časti vykompenzovaná tým, že *Keras* paralelne s trénovaním siete žiada o nové vstupy a tým pádom sú k dispozícii hneď, keď ich potrebuje. Veľkou výhodou tohoto spôsobu je možnosť pracovať s ľubovoľne veľkým trénovacím korpusom, ktorý by sa do pamäte nezmestil.

Trieda NeuralNER

Trieda NeuralNER v sebe obsahuje niekoľko metód, ktoré spolu vytvárajú celok umožňujúci rozpoznávanie pomenovaných entít. Pri vytváraní inštancie tejto triedy je potrebné predať jediný parameter, ktorým je slovník parametrov, viď časť o konfiguračnom súbore. Každá metóda je podrobne zdokumentovaná v zdrojovom kóde, preto tu len krátko zhrniem význam najdôležitejších metód.

Zvonku prístupné metódy má trieda NeuralNER len štyri. Sú nimi metódy *train, eval, prepare_for_tagging* a *tag_documents*. Metóda *train* slúži k samotnému učeniu neurónovej siete. Nie je potrebné jej predávať žiadne parametre, keďže parametre už sú predané pri vytváraní inštancie vo forme slovníku. Táto metóda tento slovník spracuje a naučí podľa špecifikovaných parametrov neurónovú sieť. Metóda *eval* využije triedu *MetricsCalculator* k vypočítaniu vyhodnocovacích metrík modelu. Metóda *prepare_for_tagging* je volaná pre načítanie modelu a korešpondujúcich kódovaní, aby tieto súbory nebolo nutné načítavať pri každom volaní metódy *tag_documents*. Metóda *tag_documents* slúži k využitiu naučeného modelu pre označenie pomenovaných entít v texte. Tejto metóde sa predáva jeden argument, ktorým je zoznam dokumentov na označenie. Výstupom je zoznam rovnakej dĺžky obsahujúci dokumenty doplnené o značky a ich pravdepodobnosti. Príklad výstupu je znázornený na Obr. 8.

Metóda *_tokenize_split* vstupný súbor dát rozdelí na trénovaciu, validačnú a overovaciu množinu a zároveň dokumenty v nich tokenizuje pomocou HTML tokenizéru.

Metóda *_build_encodings* sa stará o prevod tokenov a názvov entít na číselné vstupy a výstupy pre neurónovú sieť. Vytvorené slovníky ukladá do súboru na disku. Metóda *_load_encodings* slúži k načítaniu existujúcich slovníkov zo súboru na disku. Tieto slovníky sú potrebné, bez nich nie je možné naučený model používať, pretože by nebolo možné určiť, ktorý výstup patrí ktorej pomenovanej entite, analogicky by nebolo možné určiť, ktorý vstup patrí ktorému tokenu.

Metóda *_load_model* slúži k načítaniu modelu neurónovej siete z disku. Modely sú ukladané vo formáte HDF5 do adresáru *MODELS_DIR*, ktorý je špecifikovaný v konfiguračnom súbore. Metóda *_compile_model* slúži k vytvoreniu požadovanej architektúry neurónovej siete. Všetky parametre ohľadom architektúry sa nachádzajú v konfiguračnom súbore v časti *Neural Network*.

7.3 Implementácia CRF modelu

CRF model bol implementovaný taktiež v jazyku Python, za pomoci nástrojov *pycrfsuite* a *sklearn*. Časť programu pre CRF model obsahuje nasledujúce súbory:

- *html_tokenizer.py*
  Obsahuje HTML tokenizér popísaný v podkapitole vyššie potrebný pre prevod emailových správ na postupnosť tokenov.
- *config.ini*
  Jedná a o konfiguračný súbor, v ktorom je možné nastaviť všetky parametre CRF modelu.
- *crfner.py*
  Obsahuje implementáciu CRF modelu s ďalšími pomocnými metódami.
- *requirements.txt*
  Obsahuje zoznam balíkov jazyku Python, ktoré sú potrebné nainštalovať pre správnu funkciu programu.
- *server.py*
  Serverová časť, ktorá umožňuje použiť naučený CRF model v serverovom režime.

Konfiguračný súbor

Konfiguračný súbor je načítaný pomocou funkcie *parse_config*. Táto funkcia zo súboru vytvorí slovník kompatibilný s funkciami implementovanej CRF. Používa k tomu Python modul ConfigParser s pokročilou interpoláciou ExtendedInterpolation, ktorá podporuje predávanie informácií medzi časťami konfiguračného súboru. Niektoré parametre v konfiguračnom súbore sú vytvorené automaticky kombináciou iných. Tieto v Tab. 10 pre prehľadnosť nie sú uvedené.

Príznaky

Ako je popísané v kapitole 6, vstupy do CRF modelu sú tvorené sekvenciou príznakov jednotlivých tokenov. Konkrétne príznaky, ktoré boli použité v programe a vyhodnotené v ďalšej kapitole sú :

- *samotný token*
- *dĺžka tokenu*
  V prípade spracovania znak po znaku ide o dĺžku slova, v ktorom sa znak nachádza.

- *isalpha*
  Značí, či sa token skladá len z alfabetických znakov.
- *isnumeric*
  Značí, či sa token skladá len z číslic.
- *isupper*
  Značí, či sú všetky alfabetické znaky v tokene veľkými písmenami.
- *isdigit*
  Značí, či sa token skladá len z číslic. Zahŕňa však okrem bežných číslic i Unicode znaky, ktoré sú klasifikované ako číslice, akým je napríklad znak *00bc* = ¼ .
- *vyššie popísané parametre pre MAX_LOOKOUT počet tokenov v oboch smeroch, okrem dĺžky slova*
  Dĺžka samotného tokenu je vzhľadom k veľkej dĺžke HTML značiek okresaná na prvých 10 znakov. Takisto samotný token je prevedený na malé písmená, aby sa predišlo rôznym hodnotám napríklad pre tokeny *flight* a *Flight*. Tieto dopredné a spätné parametre sú identifikované svojim číslom a smerom, napríklad pre parameter *isnumeric* tretieho slova po aktuálnom je *+3:word.isnumeric*.

Typ	Názov	Význam
Project	ROOT_DIR	Cesta k adresáru obsahujúci celý projekt.
	DATA_DIR	Cesta k adresáru obsahujúci trénovacie dáta, vytvorená automaticky.
	TRAIN_FILENAME	Názov súboru obsahujúci trénovacie dáta.
	TEST_FILENAME	Názov súboru obsahujúci dáta určené na overenie výsledkov po učení.
	ENTITIES	Názvy rozpoznávaných pomenovaných entít, oddelené čiarkami.
	EVALUATE	Hodnota typu boolean. Udáva, či po učení má prebehnúť evaluácia naučeného modelu.
	CHAR_BY_CHAR	Hodnota typu boolean. Udáva, či má HTML tokenizér spracovávať viditeľný text znak po znaku alebo slovo po slove (oddelené medzerami).
Server	HOST	IP adresa hostu v tvare x.x.x.x., na ktorom má server bežať.
	PORT	Port, na ktorom má server bežať.
CRF	MAX_LOOKOUT	Maximálny počet slov v jednom smere, pre ktorý sú generované príznaky pre aktuálne slovo.
	ITERATIONS	Maximálny počet iterácii optimizačného algoritmu.

Tab. 10 Popis parametrov v konfiguračnom súbore pre nástroj CRFNER.

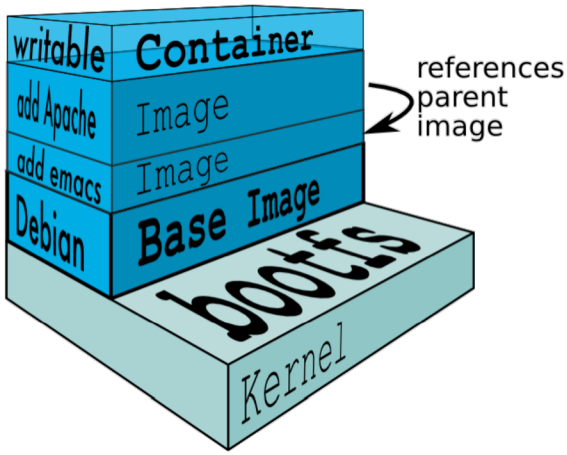
7.4 Server a kontajner Docker

Nástroje na označovanie pomenovaných entít je možné využiť i v serverovom režime. K tomuto sú navrhnuté triedy NeuralNERServer a CRFNERServer. Obe sú navrhnuté tak, že prijímajú dáta vo forme JSON odoslané na server metódou POST.

Pri inicializácii stačí predať cestu ku konfiguračnému súboru. Po načítaní potrebných modelov a pomocných súborov do pamäte je server pripravený k použitiu. Prístupový bod, cez ktorý je možné sa so serverom spojiť sa nachádza na adrese *http://host/ner*, kde *host* je IP adresa, ktorá je načítaná z konfiguračného súboru.

Server je možné tiež použiť v režime Docker kontajneru. Docker je pomerne nový projekt, ktorý vznikol v roku 2013. Umožňuje zabaliť aplikáciu a súbory, na ktorých je závislá do virtuálneho kontajneru, ktorý môže bežať na ľubovoľnom Linuxovom serveri. Toto napomáha flexibilite a prenosnosti aplikácií. Kontajnery je možné zabaliť do obrazov (*Docker image*), za pomoci ktorých je možné prenášať celé aplikácie. V prípade veľkej záťaže na jednu aplikáciu je možné z týchto obrazov veľmi rýchlo vytvoriť nové inštancie aplikácií, naopak v prípade menšej záťaže je možné pár inštancií vypnúť.

Kontajnery sú vytvorené nabaľovaním vrstiev na seba, viď Obr. 7. Toto nabaľovanie na seba je nutné špecifikovať v špeciálnom súbore zvaný Dockerfile.



Obr. 7 Vrstvy Docker kontajnerov. Prevzaté z [28].

Server je uspôsobený prijímať zoznam reťazcov, prípadne i reťazce samostatné. Funkciou serveru je po prijatí reťazcov vrátiť reťazce doplnené o značky pomenovaných entít spolu s ich predpovedanou pravdepodobnosťou. Príklad časti vráteného reťazca je možné vidieť na Obr. 8.

```
▼<td style="font-family:Arial, sans-serif;font-size:14px;padding:10px 5px;|
family:Arial, Helvetica, sans-serif !important;" align="Left">
    <dep_date prob="0.97">26/Sep/2016</dep_date>
    <dep_time prob="0.99">15:25</dep_time>
```

Obr. 8 Príklad reťazcu vráteného serverovou časťou aplikácie.

8 POROVNANIE A ZHODNOTENIE

Nástroje spomenuté v tejto práci sú porovnané na 2 rôznych častiach vývojového korpusu obsahujúcich emailové správy. Spôsobom uvedeným v kapitole 4.2 bol za pomoci regulárnych výrazov vytvorený vývojový korpus obsahujúci správy z 20 najčastejšie sa nachádzajúcich predlôh v celej databáze správ. Z týchto 20 predlôh bol vytvorený trénovací korpus, obsahujúci správy zo 16 predlôh, a testovací korpus obsahujúci správy z ďalších 4 predlôh.

Vytvorený trénovací korpus obsahuje približne 35 tisíc správ. Keďže sa jedná o správy takmer rovnakého charakteru s veľmi podobným obsahom, bol kvôli veľkej časovej náročnosti učiaceho procesu trénovací korpus *Train* zmenšený na prvých 10%, presne 3482 správ.

Testovací korpus *Test* je tvorený emailovými správami zo 4 predlôh, na ktorých nástroje neboli naučené. Jedná sa teda o správy obsahujúce dôležité údaje vo forme, s ktorou sa učené modely zatiaľ nestretli. Na tomto korpuse je overovaná schopnosť nástrojov generalizovať.

V tejto kapitole sa viackrát porovnávajú modely, ktoré boli učené na dátach tokenizovaných spôsobom slovo po slove s modelmi, ktoré boli učené na dátach tokenizovaných spôsobom znak po znaku. Z dôvodu prehľadnosti sú tieto modely ďalej označované kratším pomenovaním – modely *slovo po slove* a modely *znak po znaku*.

V prípade CRF modelov sú ďalej porovnané modely na základe rôznych hodnôt parametru *MAX_LOOKOUT* – konkrétne 3, 5 a 7 – po počtoch iterácií 30, 100, 500 a 1000.

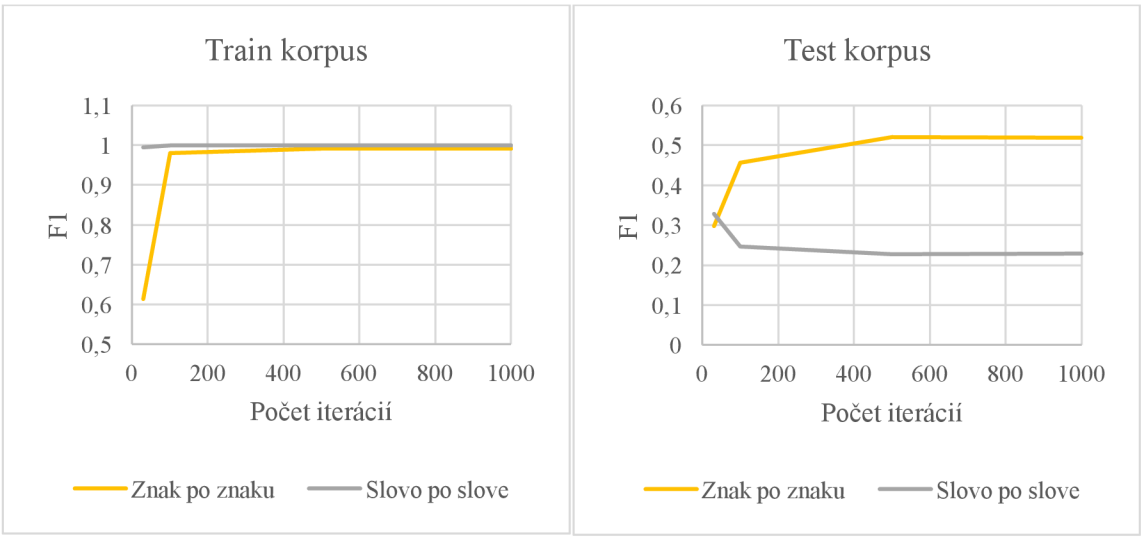
Ďalej sú porovnané modely neurónových sietí s architektúrou GRU a LSTM na základe rovnakých hodnôt parametru *MAX_LOOKOUT* po počtoch iterácií (v prípade neurónových sietí sa jednému prechodu cez všetky dáta hovorí *epochy*) 10, 20 a 30. Ďalšie parametre sú nastavené podľa hodnôt v konfiguračnom súbore v prílohe A.

Kapitola 8.1 ponúka zhodnotenie rôznych CRF modelov, kapitola 8.2 ponúka prehľad výsledkov modelov z kategórie neurónových sietí a v kapitole 8.3 je prehľad výsledkov všetkých nástrojov použitých v tejto práci, vrátane existujúcich nástrojov, a zhodnotenie získaných výsledkov.

8.1 Porovnanie CRF modelov

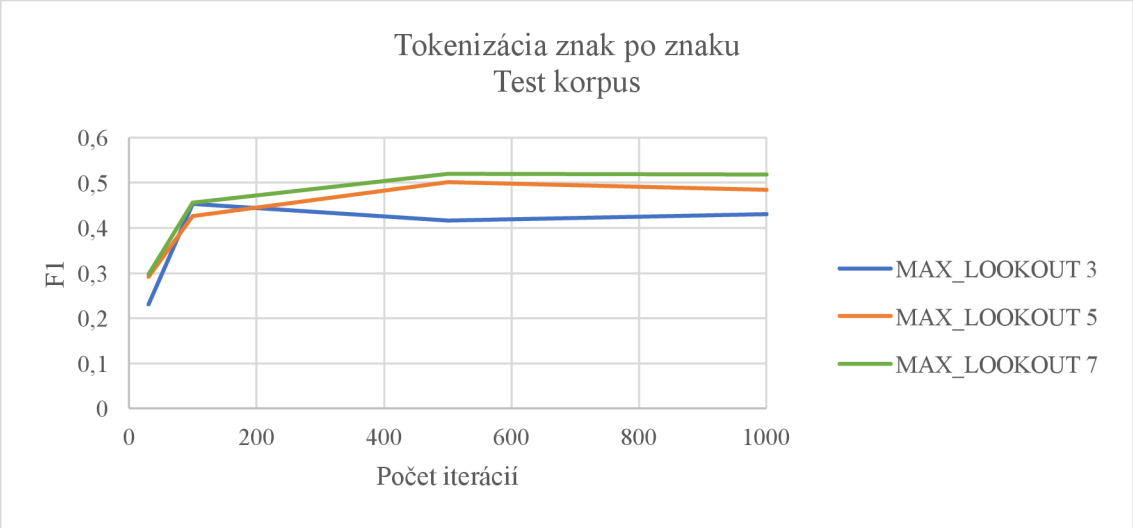
V priebehu písania tejto práce boli otestované viaceré varianty parametrov pre CRF modely. V tejto časti je zhrnuté, aký majú niektoré z parametrov vplyv na výsledky, pričom pozornosť je pre prehľadnosť venovaná len metrike F1. Kompletná tabuľka výsledkov je dostupná v prílohe A.

Na Obr. 9 je vidieť, že model *slovo po slove* sa naučil rýchlejšie spoľahlivo identifikovať pomenované entity v správach podľa predlôh, na základe ktorých sa učil. V prípade správ, ktorých predlohy neboli známe počas učenia, má horšiu úspešnosť ako model *znak po znaku*. Model *znak po znaku* potrebuje väčší počet iterácií na získanie podobnej úspešnosti na trénovacích dátach ako model *slovo po slove*. Úspešnosť modelu *znak po znaku* na testovacích dátach je však viac než dvakrát vyššia.

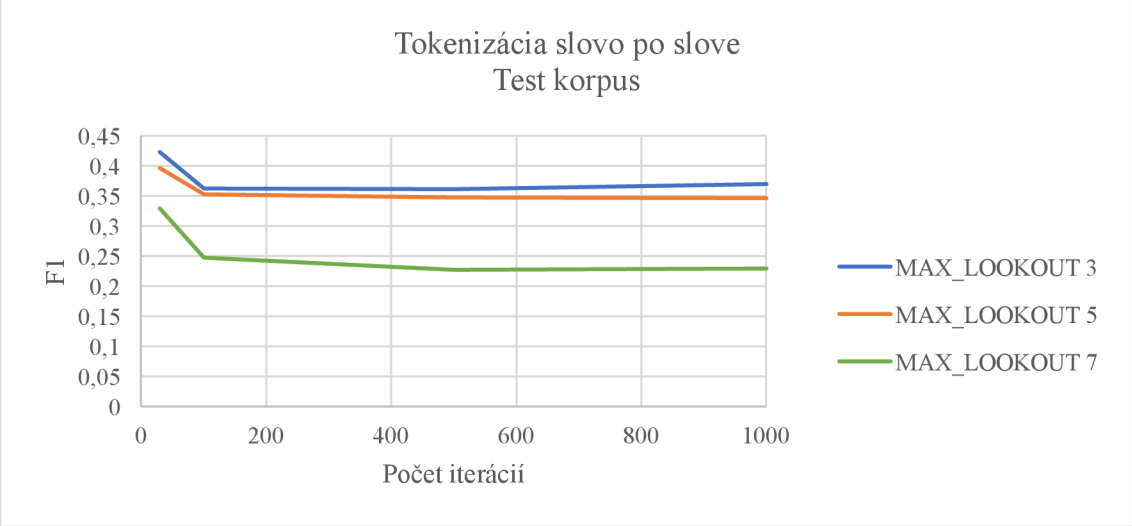


Obr. 9 Vplyv počtu iterácií a spôsobu tokenizácie.

Ďalším parametrom, ktorý má veľký vplyv na úspešnosť modelu, je parameter *MAX_LOOKOUT*. Definuje, pre koľko tokenov obidvoma smermi v sekvencii sú generované príznaky. Z Obr. 10 vidíme, že v prípade modelu *znak po znaku* sa s rastúcim parametrom *MAX_LOOKOUT* zvyšuje úspešnosť. Je to následkom toho, že modelu je umožnené sa pozerať na viac znakov okolo seba a rozhodovať na základe väčšieho množstva informácií z okolia. Na Obr. 11 je vidieť, že modelu *slovo po slove* s rastúcim počtom iterácii a rastúcim parametrom *MAX_LOOKOUT* úspešnosť postupne klesá. Je to pravdepodobne spôsobené tým, že model sa stane príliš závislým na okolitých tokenoch a pravdepodobnosť klasifikácie tokenu za pomenovanú entitu príliš klesne, keď sa v okolí nenachádza token, ktorý je očakávaný z trénovacích dát. Naučí sa teda v príliš veľkej miere spoliehať na okolité tokeny.



Obr. 10 Vplyv parametru MAX_LOOKOUT na model znak po znaku.



Obr. 11 Vplyv parametru MAX_LOOKOUT na model slovo po slove.

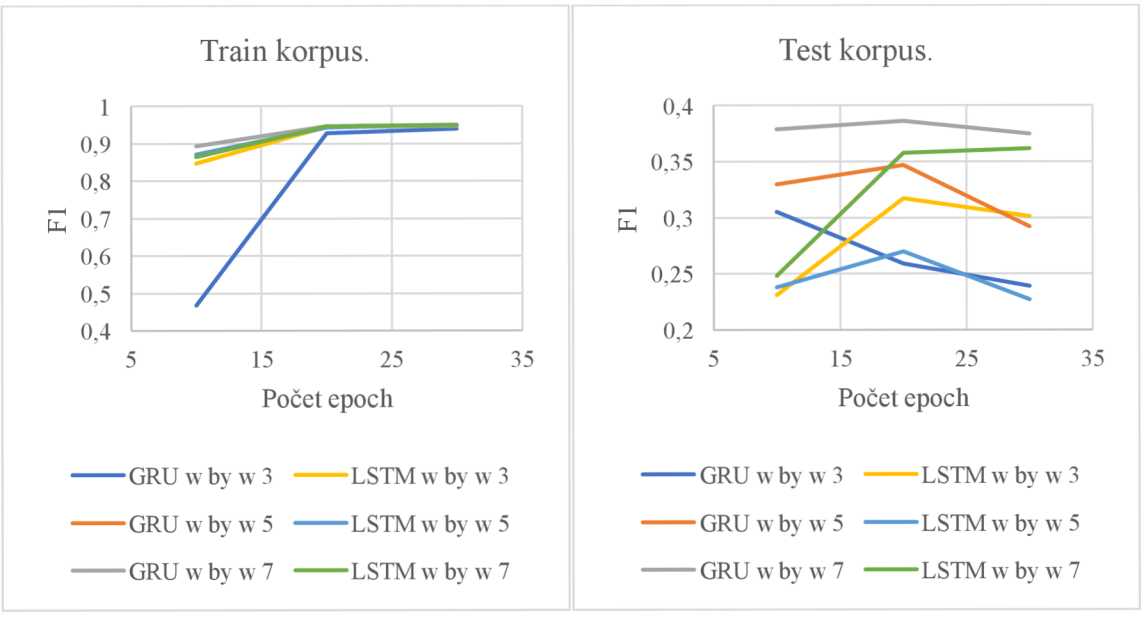
8.2 Porovnanie modelov neurónových sietí

V priebehu písania tejto práce boli otestované viaceré varianty parametrov pre modely na základe neurónových sietí. V tejto časti je zhrnuté, aký majú niektoré z parametrov vplyv na výsledky, pričom pozornosť je pre prehľadnosť venovaná len metrike F1. Kompletná tabuľka výsledkov je dostupná v prílohe A.

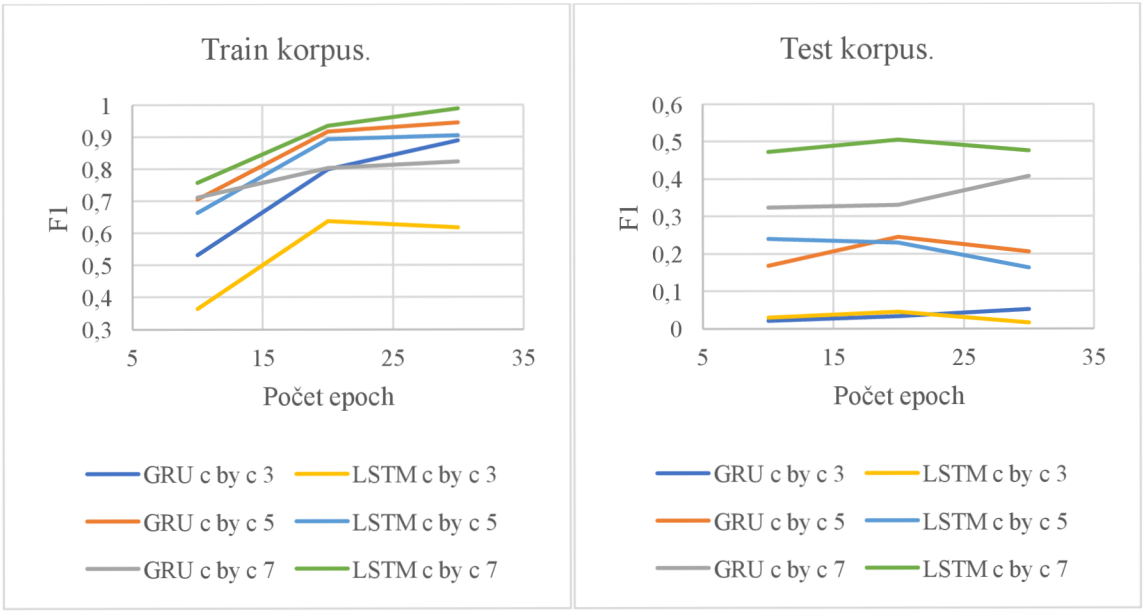
Obr. 12 ukazuje úspešnosť modelov slovo po slove (označené *w by w*) s rôznymi hodnotami parametru *MAX_LOOKOUT,* konkrétne 3, 5 a 7. Vidíme, že všetky modely sa po počte 20 epoch naučili rozpoznávať entity v korpuse Train s hodnotou metriky F1 nad 90%. Na testovacom korpuse ani jeden nepresiahol hodnotu 40%, najlepšie výsledky dosahujú modely s parametrom *MAX_LOOKOUT* rovný 7.

Rovnaké hodnoty boli otestované na modeloch znak po znaku (označené *c by c*), viď Obr. 13. Je možné vidieť, že na trénovacom korpuse dosahujú dobré výsledky – okrem

modelu *LSTM c by c 3* presiahli všetky hodnotu metriky F1 80%. Na testovacom korpuse je možné vidieť vzťah medzi hodnotou parametru *MAX_LOOKOUT* a hodnotou metriky F1, kde vyššia hodnota parametru má za následok značný nárast výslednej hodnoty F1. Najlepšie výsledky dosahujú opäť modely s parametrom *MAX_LOOKOUT* rovný 7.



Obr. 12 Porovnanie úspešnosti modelov slovo po slove na Train a Test korpuse.



Obr. 13 Porovnanie úspešnosti modelov znak po znaku na Train a Test korpuse.

8.3 Celkové porovnanie

Táto kapitola je venovaná porovnaniu úspešnosti všetkých vytvorených nástrojov spolu s existujúcimi nástrojmi spomenutými v kapitole 3. Najlepšie výsledky sú v každej tabuľke vyznačené hrubým písmom.

Z nástrojov vytvorených v rámci tejto práce sú uvedené vždy po dva modely, prvý vytvorený na základe dát tokenizovaných slovo po slove označený skratkou *w by w* a druhý vytvorený na základe dát tokenizovaných znak po znaku označený skratkou *c by c*. Uvedené modely z navrhnutých nástrojov majú nastavený parameter *MAX_LOOKOUT* na hodnotu 7, pretože ten často prinášal najlepšie výsledky, ako bolo možné vidieť v predchádzajúcich podkapitolách. Kompletná tabuľka výsledkov všetkých 20 otestovaných modelov v rôznych krokoch učenia je dostupná v prílohe A.

Ďalej sú uvedené 2 modely z nástroja Apache OpenNLP – model maximálnej entropie označený skratkou MAXENT a perceptronový model označený skratkou PERC. Model Naive Bayes nebolo možné kvôli chybe v tomto softvéri otestovať.

Nástroj Stanford NER kvôli nevhodne navrhnutému tokenizéru nie je možné priamo porovnať. Tokenizér tohoto nástroju nie je schopný spracovať tokeny obsahujúce medzery, ktoré značky HTML kódu často obsahujú.

Nástroj	Presnosť	Pokrytie	F1
CRF w by w	**1**	**1**	**1**
CRF c by c	0,992	0,992	0,992
LSTM w by w	0,999	0,933	0,95
LSTM c by c	0,993	0,983	0,988
GRU w by w	0,996	0,93	0,947
GRU c by c	0,791	0,988	0,824
OpenNLP MAXENT	0,964	0,962	0,963
OpenNLP PERC	0,953	0,953	0,953

Tab. 11 Porovnanie metrík dosiahnutých na korpuse Train.

Nástroj	Presnosť	Pokrytie	F1
CRF w by w	0,214	0,26	0,229
CRF c by c	0,609	**0,509**	**0,519**
LSTM w by w	0,328	0,42	0,362
LSTM c by c	0,626	0,414	0,475
GRU w by w	0,354	0,411	0,375
GRU c by c	0,495	0,386	0,407
OpenNLP MAXENT	**0,760**	0,050	0,094
OpenNLP PERC	0,440	0,190	0,265

Tab. 12 Porovnanie metrík dosiahnutých na korpuse Test.

Tab. 11 ukazuje, že takmer všetky nástroje boli schopné dosiahnuť úspešnosť blížiacu sa 100% na úlohe rozpoznania entít v správach podľa predlôh, na ktorých boli naučené. Toto však môže byť i znakom preučenia, kedy si modely priamo zapamätajú trénovací formát a nie sú schopné generalizovať. Výsledky z Tab. 12, ponúkajúcej prehľad metrík dosiahnutých na korpuse Test, v niektorých prípadoch potvrdzujú, v iných však vyvracajú túto skutočnosť.

Z Tab. 12 je zrejmé, že všetky modely sú schopné rozpoznať isté množstvo pomenovaných entít i v správach, ktorých predlohy im nie sú známe. Lepšie výsledky dosahujú všeobecne modely, ktoré boli učené na dátach tokenizovaných spôsobom znak po znaku. Modely CRF sa pravdepodobne naučili pre postupnosť znakov číslic uprednostňovať značky ako sú číslo letu, čas odletu a podobne. Toto im bolo z veľkej časti uľahčené definovaním pomocných príznakov, viď kapitolu 6.2. Modely neurónových sietí sa však i bez takýchto ručne definovaných pomocných príznakov naučili rozpoznávať pomenované entity s porovnateľne dobrou úspešnosťou, napríklad model *LSTM c by c* s parametrom *MAX_LOOKOUT* nastavený na hodnotu 7 dosiahol na testovacom korpuse druhý najlepší výsledok F1 metriky.

Na základe týchto výsledkov je možné pre ďalší vývoj a využitie v praxi odporučiť modely slovo po slove. V prípade trénovacích dát síce nedosahujú úplne najlepšie výsledky zo všetkých, sú však stále porovnateľne dobré. Sila v týchto modeloch však spočíva v ich lepšej schopnosti generalizovať – dosahujú najlepšie výsledky na testovacích dátach, čo im umožňuje rozpoznávať entity i v neznámych predlohách. Z modelov slovo po slove nie je jednoduché vybrať najlepší, pretože nie je možné určiť, aké úspešné budú po naučení na väčšom počte vzájomne odlišných správ.

9 ZÁVER

V práci boli analyzované existujúce riešenia v oblasti rozpoznávania pomenovaných entít a vhodnosť ich použitia v rámci konkrétnych cieľov tejto práce, teda pre rozpoznávanie pomenovaných entít v emailových správach. Pre splnenie tohoto cieľa bolo nutné, aby nástroje ponúkali možnosť naučenia úplne nových modelov. Tieto požiadavky spĺňali v čase písania práce dva známe nástroje, Apache OpenNLP a Stanford NER popísané v kapitole 3.

V priebehu písania práce nebol dostupný korpus emailových správ, na ktorých by bolo možné algoritmy testovať. Z tohoto dôvodu bola databáza emailových správ, poskytnutá firmou Kiwi.com, spracovaná a bol z nej za pomoci regulárnych výrazov vytvorený vývojový korpus obsahujúci približne 48 tisíc správ s označenými pomenovanými entitami. Tieto správy boli rozdelené na trénovací a testovací korpus, ako je popísané v kapitole 4.

Počas riešenia problému vznikla potreba vyvinúť tokenizér prispôsobený na spracovanie HTML kódu, keďže existujúce riešenia a tokenizéry sú prispôsobené na spracovanie prirodzeného textu. Tento tokenizér umožňuje rozdelenie dokumentu na menšie časti – tokeny. Keďže i ostatné časti existujúcich nástrojov sú prispôsobené na spracovanie prirodzeného textu, boli v rámci práce kompletne vyvinuté i dva nástroje. Prvý je implementáciou neurónovej siete a druhý je postavený na základe grafového modelu CRF. O týchto spôsoboch a ich implementácii pojednávajú kapitoly 5, 6 a 7.

Po implementácii a vytvorení korpusov boli v jednotlivých nástrojoch vytvorené modely s rôznymi hodnotami parametrov, ktoré boli naučené na trénovacom korpuse. Táto časť práce bola pomerne časovo náročná. Doby učenia jednotlivých modelov boli v rozmedzí niekoľkých desiatok minút až niekoľkých dní. Úspešnosť naučených modelov bola overená i na testovacom korpuse a prehľad výsledkov je dostupný v kapitole 8.

Všetky modely boli schopné sa naučiť dobre rozpoznávať entity v trénovacom korpuse. Toto bolo očakávané vzhľadom na to, že trénovací korpus obsahuje veľké množstvo správ vytvorených na základe malého počtu rôznych predlôh. Vyššiu výpovednú hodnotu o úspešnosti a nasaditeľnosti v praxi ponúkajú výsledky na testovacom korpuse. Z týchto výsledkov je zrejmé, že modely znak po znaku boli schopné lepšie generalizovať a dokážu rozpoznávať väčšiu časť pomenovaných entít i v neznámych predlohách. Z modelov znak po znaku porovnateľné výsledky dosiahli modely LSTM, GRU a CRF. Najlepšie výsledky boli dosiahnuté modelom znak po znaku CRF s hodnotou parametru *MAX_LOOKOUT* rovný 7. Model bol schopný v správach podľa neznámych predlôh schopný dosiahnuť hodnotu 51,9% F1 metriky, ktorá do seba spája metriky presnosť a pokrytie.

Pred riešením tejto práce nebolo zrejmé, či je vhodné nasadiť existujúce algoritmy na rozpoznávanie pomenovaných entít v prirodzenom texte na takýto odlišný formát vstupného textu. Z výsledkov tejto práce vyplýva, že tieto algoritmy sa ukázali byť vhodnou voľbou. Ďalším postupom by bolo naučenie modelov na reálnych dátach,

tj. dátach ponúkajúcich správy z väčšieho počtu rozdielnych predlôh. Na týchto reálnych dátach by bolo vhodné ďalej skúmať vplyv nastaviteľných parametrov a tieto optimalizovať.

Vo firme Kiwi.com bude možné vytvorené nástroje v budúcnosti použiť nasledovne. Pred prečítaním emailovej správy obsahujúcou dôležité informácie bude správa spracovaná najlepším naučeným modelom a prebehne v nej rozpoznanie pomenovaných entít. Človeku, ktorý má za úlohu túto správu prečítať, bude teda predstavená správa, ktorá obsahuje pôvodný obsah a ďalej rozpoznané pomenované entity, farebne či iným spôsobom označené. Toto urýchli spracovanie správ a v konečnom dôsledku môže viesť až k plne automatizovanému spracovávaniu prijatých emailových správ.

ZOZNAM POUŽITEJ LITERATÚRY

[1] April 2017 Web Server Survey. In: *Netcraft | Internet Security and Data Mining* [online]. Bath, United Kingdom: Netcraft, 2017 [cit. 2017-05-03]. Dostupné z: https://news.netcraft.com/archives/category/web-server-survey/

[2] *Handbook of natural language processing*. Boca Raton, FL: Chapman & Hall/CRC, c2010. Chapman & Hall/CRC machine learning & pattern recognition series. ISBN 9781420085921.

[3] NAGGY, Marek. *Systém pro extrakci informací z kriminalistických textů*. Plzeň, 2016. Diplomová práca. Západočeská univerzita v Plzni. Vedúci práce Karel Ježek.

[4] KURU, Onur, Ozan Arkan CAN a Deniz YURET. CharNER: Character-Level Named Entity Recognition. In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Japonsko: COLING, 2016, s. 911-921. ISBN 978-4-87974-702-0.

[5] SHEN, Mo, Hongxiao LIU, Daisuke KAWAHARA a Sadao KUROHASHI. Chinese Morphological Analysis with Character-level POS Tagging. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers)*. Baltimore, MD, USA: Association for Computational Linguistics, 2014, s. 253-258. ISBN 978-1-937284-73-2.

[6] HELLEBRAND, David. *Nalezení slovních kořenů v češtině*. Brno, 2010. Diplomová práca. Vysoké učení technické v Brně. Fakulta informačních technologií. Ústav informačních systémů. Vedúci práce Petr Chmelař.

[7] BIRD, Steven, Ewan KLEIN a Edward LOPER. *Natural language processing with Python*. Cambridge [Mass.]: O'Reilly, c2009. ISBN 9780596516499.

[8] RYLKO, Vojtěch. *Rozpoznávání pojmenovaných entit*. Brno, 2014. Diplomová práca. Vysoké učení technické v Brně. Fakulta informačních technologií. Vedúci práce Pavel Smrž.

[9] MANNING, Christopher, Prabhakar RAGHAVAN a Hinrich SCHÜTZE. *Introduction to information retrieval*. New York: Cambridge University Press, 2008. ISBN 978-052-1865-715.

[10] *The Stanford Natural Language Processing Group* [online]. Stanford: Stanford University Press, 2005 [cit. 2017-04-17]. Dostupné z: https://nlp.stanford.edu/software/CRF-NER.html

[11] FINKEL, Jenny Rose, Trond GRENAGER a Christopher MANNING. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In: *Proceedings of the 43rd Annual Meeting of the ACL*. Ann Arbor, MI, USA: Association for Computational Linguistics, 2005, s. 363-370.

[12] *Apache OpenNLP* [online]. The Apache Software Foundation, 2013 [cit. 2017-04-19]. Dostupné z: https://opennlp.apache.org/

[13] RATNAPARKHI, Adwait. A Simple Introduction to Maximum Entropy Models for Natural Language Processing. *IRCS Technical Reports Series* [online]. 1997, **6**(1), 1-11 [cit. 2017-05-03]. Dostupné z: http://repository.upenn.edu/ircs_reports/81/

[14] ZAPLETAL, Ondřej. *Srovnání správnosti klasifikace pomocí tradičních modelů a meta-modelů*. Brno, 2014. Bakalárska práca. Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Vedúci práce Petr Honzík.

[15] MACHÁČ, Martin. *Toolbox pro neuronové sítě pro prostředí Mathematica*. Zlín, 2009. Diplomová práca. Univerzita Tomáše Bati ve Zlíně. Fakulta aplikované informatiky. Vedúci práce Zuzana Oplatková.

[16] GÖRLICHOVÁ, Lucie. *Umělé neuronové sítě v lékařské diagnostice*. Brno, 2006. Diplomová práca. Masarykova univerzita v Brně. Prírodovedecká fakulta. Vedúci práce Marta Farková.

[17] VONDRÁK, Ivo. *Umělá inteligence a neuronové sítě*. 3. vyd. Ostrava: VŠB - Technická univerzita Ostrava, 2009. ISBN 978-80-248-1981-5.

[18] BILÍK, David. *Hluboké neuronové sítě a algoritmy posilovaného učení pro hraní videoher*. Praha, 2015. Diplomová práca. České vysoké učení technické v Praze. Fakulta informačních technologií. Vedúci práce Jan Drchal.

[19] GRAVES, Alex, Marcus LIWICKI, Santiago FERNANDEZ, Roman BERTOLAMI, Horst BUNKE a Jurgen SCHMIDHUBER. A Novel Connectionist System for Unconstrained Handwriting Recognition. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Chicago, IL, USA: IEEE Computer Society, 2009, **31**(5), s. 855-868. DOI: 10.1109/TPAMI.2008.137. ISSN 0162-8828. Dostupné tiež z: http://ieeexplore.ieee.org/document/4531750/

[20] NOVÁČIK, Tomáš. *Rekurentní neuronové sítě pro rozpoznávání řeči*. Brno, 2016. Diplomová práca. Vysoké učení technické v Brně. Fakulta informačních technologií. Vedúci práce Karel Veselý.

[21] CHUNG, Junyoung, Caglar GULCEHRE, KyungHyun CHO a Yoshua BENGIO. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. In: *NIPS 2014 Workshop on Deep Learning*. Montreal, Kanada: Neural Information Processing Systems Foundation, 2014, s. 1-9.

[22] HE, Xuming, Richard ZEMEL a Miguel CARREIRA-PERPINAN. Multiscale conditional random fields for image labeling. In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Washington, DC, USA: IEEE, 2004, s. 695-702. DOI: 10.1109/CVPR.2004.1315232. ISBN 0-7695-2158-4. Dostupné tiež z: http://ieeexplore.ieee.org/document/1315232/

[23] SHA, Fei a Fernando PEREIRA. Shallow Parsing with Conditional Random Fields. In: *Proceedings of HLT-NAACL 2003*. Edmonton, Kanada: HLT-NAACL, 2003, s. 134-141.

[24] VANTUCH, Marek. *Metody strojového učení ve zpracování přirozeného jazyka*. Brno, 2011. Bakalárska práca. Vysoké učení technické v Brně. Fakulta informačních technologií. Vedúci práce Lubomír Otrusina.

[25] ZHU, Xiaojin. Conditional Random Fields. In: *CS838-1 Advanced NLP* [online]. Madison, WI, USA: University of Wisconsin-Madison, 2007 [cit. 2017-05-03]. Dostupné z: http://pages.cs.wisc.edu/~jerryzhu/cs838/

[26] SEKER, Gökhan Akin a Gülsen ERYIGIT. Initial explorations on using CRFs for Turkish Named Entity Recognition. In: *Proceedings of COLING 2012: Technical Papers*. Mumbai, India: Association for Computational Linguistics, 2012, s. 2459–2474.

[27] GLAVAŠ, Goran, Mladen KARAN, Frane ŠARIĆ, Jan ŠNAJDER, Jure MIJIĆ, Artur ŠILIĆ a Bojana Dalbelo BAŠIĆ. CroNER: A State-of-the-Art Named Entity Recognition and Classification for Croatian Language. In: *Proceedings of the 15th International Multiconference: Information Society - IS 2012*. Ljubljana, Slovinsko: Narodna in univerzitetna knjižnica, Ljubljana, 2012, s. 73-78. ISSN 1581-9973.

[28] Docker Documentation. *Docker Docs* [online]. Docker, 2016 [cit. 2017-05-08]. Dostupné z: http://docs.master.dockerproject.org/terms/layer/

ZOZNAM POUŽITÝCH SKRATIEK A SYMBOLOV

BOW	Bag of words
BPTT	Backpropagation Through Time
CRF	Conditional Random Fields
GIS	Generalized Iterative Scaling
GRU	Gated Recurrent Unit
HDF5	Hierarchical Data Format 5
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
L-BFGS	Limited-memory Broyden–Fletcher–Goldfarb–Shanno algoritmus
LSTM	Long Short-Term Memory
MUC	Message Understanding Conference
NE	Named Entity
NER	Named Entity Recognition
NLP	Natural Language Processing
URL	Uniform Resource Locator

ZOZNAM POUŽITÝCH OBRÁZKOV

ZOZNAM POUŽITÝCH TABULIEK

ZOZNAM PRÍLOH

Príloha A – CD
CD obsahuje zdrojový kód oboch navrhnutých nástrojov. Ďalej obsahuje súbor programu Microsoft Excel, ktorý obsahuje výsledky všetkých variant nástrojov testovaných v rámci tejto práce.