Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Vizuálne spracovanie informácii z verejných registrov

Bakalárska práca

2021
Miloš Uriga

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Vizuálne spracovanie informácii z verejných registrov

Bakalárska práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	Ing. Viktor Kocur
Konzultant:	Mgr. Martin Turček

Bratislava, 2021
Miloš Uriga

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Miloš Uriga
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Vizuálne spracovanie informácii z verejných registrov
Visual processing of data from public registries

Anotácia: Informatizácia spoločnosti umožnila rôznym orgánom štátnej správy ako aj iným subjektom zverejňovať veľké množstvo dát podliehajúcich verejnemu záujmu. Takto zverejnené dáta sú často významným podkladom pre investigatívnu žurnalistiku. Mnohé z týchto dát sú však zverejnené vo forme, ktorá neumožňuje efektívnu analýzu väčšieho množstva týchto dát. Nástroje na spracovanie takýchto dát tak môžu byť užitočné pre prácu novinárov ako aj širokej verejnosti.

Cieľ: Cieľom práce je navrhnúť, implementovať a otestovať softvér, ktorý bude automaticky vizuálne spracovávať informácie z vybraného verejného registra. Softvér bude navrhnutý a otestovaný v kontexte využitia pri investigatívnej práci novinárov ako aj bežnou verejnosťou. Zadanie práce bude bližšie špecifikované po vzájomnej konzultácii.

Vedúci: Ing. Viktor Kocur
Konzultant: Mgr. Martin Turček
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 30.09.2020

Dátum schválenia: 06.10.2020 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie: Tu môžete poďakovať školiteľovi, prípadne ďalším osobám, ktoré vám s prácou nejako pomohli, poradili, poskytli dáta a podobne.

Abstrakt

V registri partnerov verejného sektora by mal byť ako konečný užívateľ výhod zapísaný majiteľ, no v niektorých prípadoch je zapísaný iba štatutár firmy. To môže byť úplne legitímne. Avšak účelom registra je okrem iného zverejnenie majiteľa firmy obchodujúcej so štátom. Pre verejnosť by bolo preto prospešné vedieť, u ktorých firiem napriek zápisu v registri nie sú známi majitelia. Vo všetkých prípadoch je v registri zapísaný konečný uživateľ výhod. Z pohľadu vyhľadávania na webe registra nie je ale nijako rozlišiteľné, na základe čoho bol zapísaný konečný užívateľ výhod a teda či sa jedná o majiteľa alebo štatutára. Táto skutočnosť je ale vyjadrená vo verifikačnom dokumente každého subjektu. Cieľom tejto práce bolo vytvorenie aplikácie, ktorá je schopná rozdeľovať subjekty z registra podľa ich verifikačných dokumentov do kategórie majiteľov a štatutárov. Pre dosiahnutie tohto cieľa bolo nevyhnutné vytvoriť dataset pozostávajúci zo spomínaných verifikačných dokumentov a malého množstva štruktúrovaných dát prislúchajúcim ku každému dokumentu. Aplikácia prehľadáva webovú stránku registra odkiaľ získava štruktúrované dáta a dokumenty. Väčšina týchto dokumentov je naskenovaná a preto ich prevádza pomocou OCR na text. Text je klasifikovaný pomocou klasifikátora založenom na viacvrstvovom perceptróne. F1-skóre výsledného modelu tohto klasifikátora je pri menej zastúpenej triede štatutárov 0,89.

Kľúčové slová: klasifikácia dokumentov, spracovanie prirodzeného jazyka, optické rozpoznávanie znakov

Abstract

As end-user of benefits should be entered in the Slovak register of public sector partners the owner of company. However, in some cases only members of statutory authorities of company are entered. It could be absolutely legitimate. However, the purpose of register is among other things to publicize owner of company which does business with the state. Therefore, it would be beneficial for citizens to know, which companies owners are not known, despite of record in register. End-user of benefits have to be recorded in register in all cases. However there is no way to recognize on what basis was end-user entered in register and therefore whether he is owner or member of statutory authorities by searching on web of register. But this fact is expressed in verification document of every subject. The goal of this thesis was developing an application, which could classify subjects of register according their verification documents as owners or statutory authorities. To achieve this goal was necessary to create data set. It contains verification documents and small piece of structured data belonging to each of them. The application browse the web page of register and acquires structured data and documents from it. For as much as most of these documents are scanned, it performs OCR on them to create text. That text is then classified by classifier based on multi-layer perceptron. F1-score of final model of this classifier is 0,89 on less represented class of statutory authorities.

Keywords: document classification, natural language processing, optical character recognition

Obsah

Úvod

Známym problémom verejného obstarávania nie len na Slovensku bývajú schránkové firmy u ktorých nevieme, kto je ich majiteľom. Za účelom vysporiadania sa s týmto problémom vznikol v roku 2017 register partnerov verejného sektora, ktorý býva tiež označovaný ako protischránkvý register. Tento register je podľa Transparency International Slovensko „popri centrálnom registri zmlúv najlepším nástrojom na overenie, kto podniká s verejnými inštitúciami“ [46].

Prínos registra je, že verejnosť môže jednoducho vyhľadať, kto je tzv. konečným užívateľom výhod firiem, ktoré obchodujú so štátom. V registri totiž musí byť pri každej firme zapísaný aspoň jeden konečný užívateľ výhod. Toto ale má úskalia, keďže za istých podmienok môže byť v registri ako jediný konečný užívateľ výhod danej firmy zapísana fyzická osoba, ktorá z nej nevlastní vôbec nič. Niekedy sa totiž môže do registra zapísať namiesto majiteľa štatutár firmy.

To ešte nemusí znamenať nič nelegitímne, naopak, v niektorých prípadoch takýto postup prikazuje zákon. Avšak takéto firmy si zaslúžia zvýšenú pozornosť verejnosti a najmä investigatívy, keďže v takýchto prípadoch register nezaznamenáva skutočných majiteľov a teda neplní svoj účel.

Problémom ale je, že register nijako nerozlišuje medzi konečnými užívateľmi výhod, ktorí boli zapísaní ako skutoční majitelia a tými, ktorí boli zapísaní ako štatutári. Verejnosť teda nemôže vyhľadať firmy, ktoré obchodujú so štátom, ale v skutočnosti nezverejňujú mená svojich majiteľov.

Cieľom našej práce je vysporiadať sa s týmto problémom a priniesť nástroj, vďaka ktorému bude možné zobraziť iba tie firmy, u ktorých je v registri zapísaný ako konečný užívateľ výhod ich štatutár, resp. štatutári.

Na dosiahnutie tohto cieľa používame existujúce nástroje a metódy z oblasti extrakcie dát z webu, optického rozoznávania znakov (OCR) a spracovania prirodzeného jazyka. V práci využívame, že informácie o tom, z akého dôvodu bola daná osoba zapísaná

ako konečný užívateľ výhod sa nachádzajú v naskenovanom verifikačnom dokumente, ktorý je súčasťou každého platného záznamu na webe registra. Vďaka tomu by sa náš problém dal označiť ako problém klasifikovania dokumentov.

V prvej kapitole priblížime problematiku registra partnerov verjného sektora. V druhej kapitole popíšeme fungovanie OCR. V nasledujúcej kapitole rozoberieme teóriu klasifikácie dokumentov. V štvrtej kapitole predstavíme návrh nášho riešenia. Implementáciu tohto riešenia popíšeme v nasledujúcej kapitole. Nakoniec v šiestej kapitole zhodnotíme výsledky klasifikátorov a predstavíme možnosti zlešpenia implementácie.

Kapitola 1

Register partnerov verejného sektora

V tejto kapitole bližšie predstavíme register, s ktorým budeme pracovať. Najprv popíšeme jeho fungovanie a prístup k nemu. Ďalej vysvetlíme kto sú koneční užívatelia výhod, resp. do akých kategórii sa rozdeľujú. Najprv z pohľadu práva a potom trochu zjednodušene na úroveň potrebnú pre našu prácu. V závere zdôvodníme potrebu zjednodušeného rozdelenia a ukážeme na čo je potrebný program, ktorý dokáže jednotlivé záznamy z registra takto rozdeliť.

1.1 Fungovanie registra

Register partnerov verjného sektora (RPVS) vznikol na základe zákona č. 315/2016 Z, z, o registri partnerov verejného sektora a o zmene a doplnení niektorých zákonov [4]. Cieľom fungovania tohoto registra je väčšia kontrola verejnosti nad tým, kam tečú peniaze zo štátneho rozpočtu. Do tohto registra sa musí zapísať každá fyzická alebo právnicka osoba, ktorá nie je subjektom verjnej správy a zároveň prijíma finančné prostriedky zo štátneho rozpočtu alebo, zjednodušene, akýmkoľvek spôsobom obchoduje so štátom. Takáto osoba sa nazýva partner verejného sektora (PVS).

Register je dostupný webovej adrese *rpvs.gov.sk*. Na tejto webstránke je možné prezerať všetkých PVS, ktorý boli zapísaný do registra. Ako je vidno na Obr. 1.1, v registri je možné vyhľadávať podľa rôznych údajov, napr. podľa mena konečného užívateľa výhod, či názvu spoločnosti. Každý PVS má zároveň samostantú podstránku, jej výzor ukazuje Obr. 5.1. Fungovanie webstránky bližšie popisujeme v sekcii 5.3.

Čo je pre nás dôležité, na webstránke registra môžeme nájsť tzv. verifikačné dokumenty. tie vznikajú na základe už spomínaného zákona [4], ktorý v §11 hovorí o identifikácii KUV. Tento zákon ukladá oprávnenej osobe - ktorou je väčšinou notár alebo advokát - povinnosť vyhotoviť verifikačný dokument s predpísanou štruktúrou:

(5) Identifikácia konečného užívateľa výhod a overenie identifikácie konečného užívateľa výhod sa preukazuje verifikačným dokumentom, v ktorom oprávnená osoba

a) odôvodní, na základe akých informácií postupom podľa odseku 4 identifikovala konečného užívateľa výhod alebo overila identifikáciu konečného užívateľa výhod,

b) uvedie vlastnícku štruktúru a riadiacu štruktúru partnera verejného sektora, ak je ním právnická osoba,

c) uvedie údaje podľa § 4 ods. 3 písm. f), ak o nich má alebo mohla mať vedomosť vrátane označenia verejnej funkcie,

d) v prípade partnera verejného sektora podľa § 4 ods. 4 preukáže, že podmienky na zápis vrcholného manažmentu do registra sú splnené,

e) vyhlási, že skutočnosti uvedené vo verifikačnom dokumente zodpovedajú ňou skutočne zistenému stavu.

V našej práci sa budeme zaoberať práve týmito dokumentami.

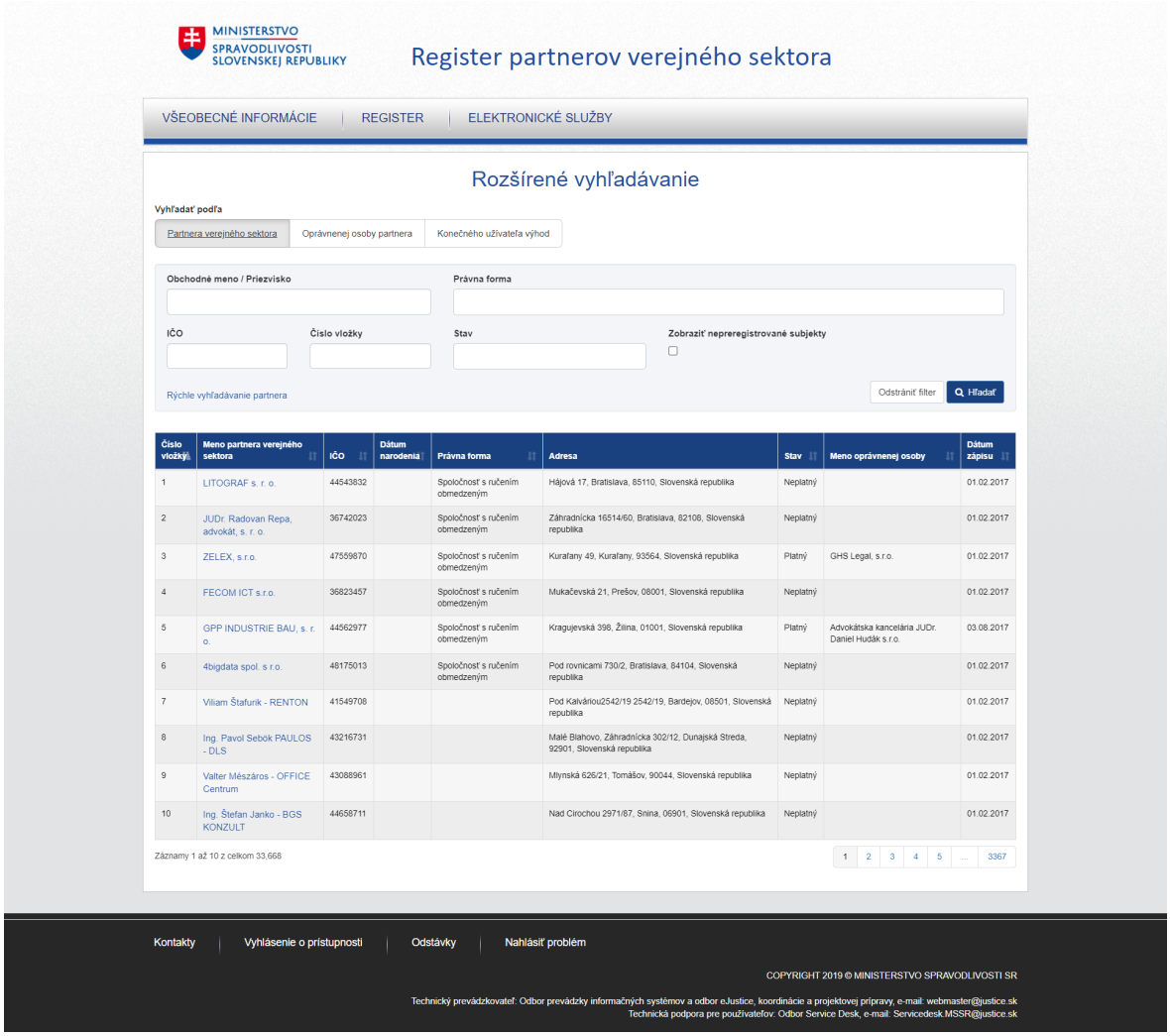
1.2 Konečný užívateľ výhod z pohľadu práva

O tom, kto je konečný užívateľ výhod hovorí zákon č. 297/2008 Z. z. o ochrane pred legalizáciou príjmov z trestnej činnosti a o ochrane pred financovaním terorizmu a o zmene a doplnení niektorých zákonov. V § 6a definuje KUV v zásade troch typov, aj keď prvý typ KUV môže mať viacero foriem [5]:

(1) Konečným užívateľom výhod je každá fyzická osoba, ktorá skutočne ovláda alebo kontroluje právnickú osobu, fyzickú osobu – podnikateľa alebo združenie majetku, a každá fyzická osoba, v prospech ktorej tieto subjekty vykonávajú svoju činnosť alebo obchod; medzi konečných užívateľov výhod patrí najmä,

a) ak ide o právnickú osobu, ktorá nie je združením majetku ani emitentom cenných papierov prijatých na obchodovanie na regulovanom trhu, ktorý podlieha požiadavkám na uverejňovanie informácií podľa osobitného predpisu, rovnocenného právneho predpisu členského štátu alebo rovnocenných medzinárodných noriem, fyzická osoba, ktorá

1. má priamy alebo nepriamy podiel alebo ich súčet najmenej 25 % na hlasovacích právach v právnickej osobe alebo na jej základnom imaní vrátane akcií na doručiteľa,



Obr. 1.1: Webstránka RPVS

2. má právo vymenovať, inak ustanoviť alebo odvolať štatutárny orgán, riadiaci orgán, dozorný orgán alebo kontrolný orgán v právnickej osobe alebo akéhokoľvek ich člena,
3. ovláda právnickú osobu iným spôsobom, ako je uvedené v prvom a druhom bode,
4. má právo na hospodársky prospech najmenej 25 % z podnikania právnickej osoby alebo z inej jej činnosti,

b) ak ide o fyzickú osobu – podnikateľa, fyzická osoba, ktorá má právo na hospodársky prospech najmenej 25 % z podnikania fyzickej osoby – podnikateľa alebo z inej jej činnosti,

c) ak ide o združenie majetku, fyzická osoba, ktorá

1. je zakladateľom alebo zriaďovateľom združenia majetku; ak je zakladateľom alebo zriaďovateľom právnická osoba, fyzická osoba podľa písmena a),

2. má právo vymenovať, inak ustanoviť alebo odvolať štatutárny orgán, riadiaci orgán, dozorný orgán alebo kontrolný orgán združenia majetku alebo ich člena alebo je členom orgánu, ktorý má právo vymenovať, inak ustanoviť alebo odvolať tieto orgány alebo ich člena,
3. je štatutárnym orgánom, riadiacim orgánom, dozorným orgánom, kontrolným orgánom alebo členom týchto orgánov,
4. je príjemcom najmenej 25 % prostriedkov, ktoré poskytuje združenie majetku, ak boli určení budúci príjemcovia týchto prostriedkov; ak neboli určení budúci príjemcovia prostriedkov združenia majetku, za konečného užívateľa výhod sa považuje okruh osôb, ktoré majú významný prospech zo založenia alebo pôsobenia združenia majetku.

(2) Ak žiadna fyzická osoba nespĺňa kritériá uvedené v odseku 1 písm. a), za konečných užívateľov výhod u tejto osoby sa považujú členovia jej vrcholového manažmentu; za člena vrcholového manažmentu sa považuje štatutárny orgán, člen štatutárneho orgánu, prokurista a vedúci zamestnanec v priamej riadiacej pôsobnosti štatutárneho orgánu.

(3) Konečným užívateľom výhod je aj fyzická osoba, ktorá sama nespĺňa kritériá podľa odseku 1 písm. a), b) alebo písm. c) druhého a štvrtého bodu, avšak spoločne s inou osobou konajúcou s ňou v zhode alebo spoločným postupom spĺňa aspoň niektoré z týchto kritérií.

1.3 Konečný užívateľ výhod v našej práci

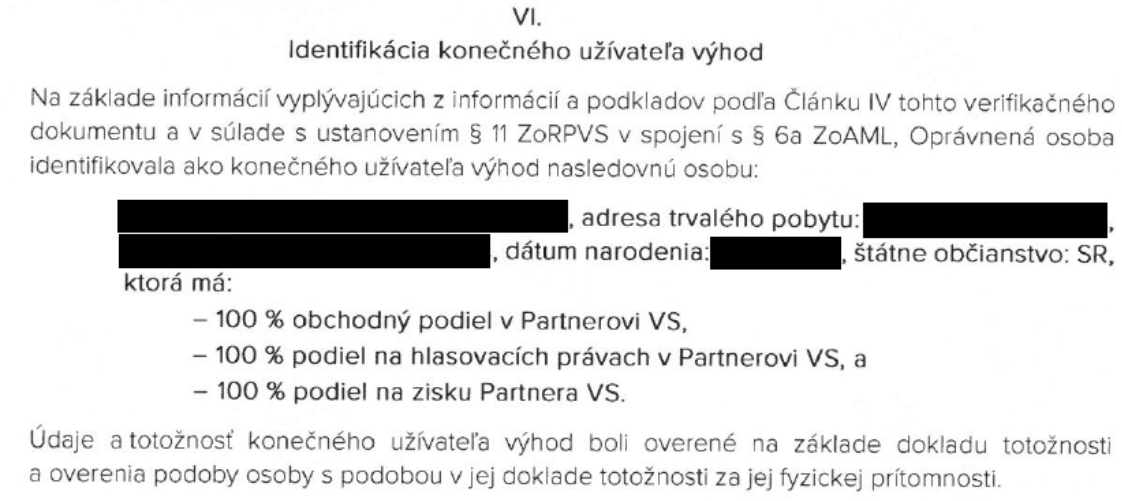
Právnicke rozdelenie toho, kto je KUV je pomerne komplikované. V našej práci používame zjednodušené delenie na majiteľov a štatutárov. V tomto rozdelení vychádzame z ukotveného právne rozdelenia. Preto KUV delíme do dvoch kategórii:

1. takých, ktorí spĺňajú podmienky podľa odseku č. 1) alebo č. 3) a teda by sa zjednodušene dal, resp. dali ak ich je viac, označiť za skutočného/ých majiteľa/ov PVS.

2. takých, ktorí boli identifikovaní podľa odseku č. 2) a teda je/sú štatutárom/mi PVS.

Obr. 1.2 zobrazuje časť z verifikačného dokumentu PVS, ktorého KUV zapísaný v registri je typu majiteľ. Môžeme to vidieť podľa toho, že daná fyzická osoba vlastní 100% obchodný podiel, podiel na hlasovacích právach aj podiel na zisku v PVS. Obr. 1.3 zobrazuje naopak časť z verifikačného dokumentu PVS, ktorého KUV je typu štatutár.

To vieme povedať na základe toho, že oprávnená osoba v dokumente tvrdí, že uvedené osoby spĺňajú definíciu KUV ako členovia vrcholového manažmentu. Samozrejme, v daných dokumentoch sa môže nachádzať viacero indícii, ktoré by naznačoval zaradenie do jednej alebo druhej kategórii. V iných dokumentoch dokumentoch sa naopak nemusia vyskytovať úplne rovnaké formulácie. Tieto obrázky ale zobrazujú príklad takých častí dokumentu, ktoré jednoznačne určujú typ dokumentu, resp. PVS.



VI.

Identifikácia konečného užívateľa výhod

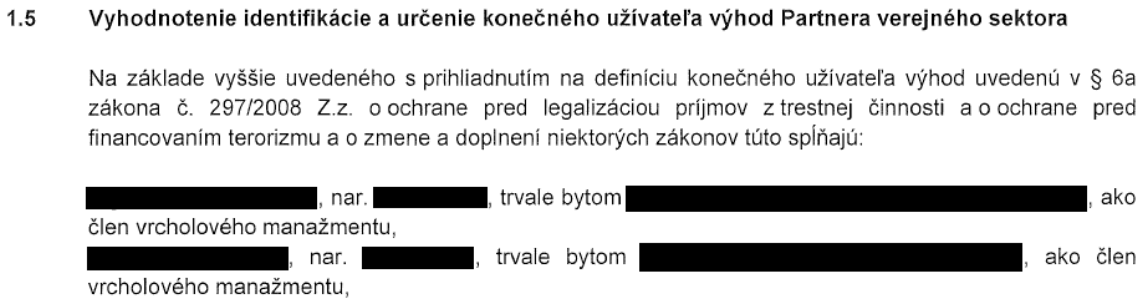
Na základe informácií vyplývajúcich z informácií a podkladov podľa Článku IV tohto verifikačného dokumentu a v súlade s ustanovením § 11 ZoRPVS v spojení s § 6a ZoAML, Oprávnená osoba identifikovala ako konečného užívateľa výhod nasledovnú osobu:

[redacted], adresa trvalého pobytu: [redacted], [redacted], dátum narodenia: [redacted], štátne občianstvo: SR, ktorá má:

- 100 % obchodný podiel v Partnerovi VS,
- 100 % podiel na hlasovacích právach v Partnerovi VS, a
- 100 % podiel na zisku Partnera VS.

Údaje a totožnosť konečného užívateľa výhod boli overené na základe dokladu totožnosti a overenia podoby osoby s podobou v jej doklade totožnosti za jej fyzickej prítomnosti.

Obr. 1.2: Časť z verifikačného dokumentu - majiteľ



1.5 **Vyhodnotenie identifikácie a určenie konečného užívateľa výhod Partnera verejného sektora**

Na základe vyššie uvedeného s prihliadnutím na definíciu konečného užívateľa výhod uvedenú v § 6a zákona č. 297/2008 Z.z. o ochrane pred legalizáciou príjmov z trestnej činnosti a o ochrane pred financovaním terorizmu a o zmene a doplnení niektorých zákonov túto spĺňajú:

[redacted], nar. [redacted], trvale bytom [redacted], ako člen vrcholového manažmentu,
[redacted], nar. [redacted], trvale bytom [redacted], ako člen vrcholového manažmentu,

Obr. 1.3: Časť z verifikačného dokumentu - majiteľ

1.4 Motivácia klasifikovania

KUV je bez ohľadu na to, podľa ktorého spôsobu bol určený, zapísaný do registra. Takto sa stráca informácia, či KUV je majiteľ alebo štatutár. Ak sa ako KUV zapíše iba štatutár PVS, stráca sa význam registra. Môže sa totiž stať, že prostriedky zo štátneho rozpočtu čerpá firma z daňového raja, ktorú v skutočnosti vlastní úradnik či politik, ktorý čerpanie zabezpečil a zároveň register na to nijako neupozorní, lebo ako

KUV bude uvedený niekto iný.

To, že sa ako KUV zapíše vrcholoví manažment, teda štatutári, je bežné napr. pri korporátoch, ktorých akicie sú obchodované na burze. Samotný fakt, že sa nenašla žiadna fyzická osoba, ktorá by sa dala označiť ako majiteľ ešte nemusí znamenať nič nekalé. Keď ale štát obchoduje s takýmito spoločnosťami, vzbuduje to väčšie podozrenie ohľadom toho, kto je v skutočnosti za danou spoločnosťou. Preto by bolo prínosné vedieť, ktorá spoločnosť neuvádza skutočných majiteľov. Vďaka tomu by sa verejnosť i investigatíva mohla viac sústrediť na tieto spoločnosti.

V našej práci sa snažíme vylepšiť fungovanie RPVS tak, aby lepšie slúžil zámeru - aby sa verejnosť mohla ľahko dopátrať, kto vlastní spoločnosť obchodujúcu so štátom. To umožníme tak, že nebude nutné pracne prechádzať všetky záznamy v registri, ale iba tie, pri ktorých skutočný majiteľ aj napriek zápisu nie je na prvý pohľad známy.

Kapitola 2

OCR

Táto kapitola sa venuje optickému rozpoznávaniu znakov - OCR. Najprv predstavíme, prečo v našej práci potrebujeme OCR. To potom predstavíme - najprv jeho stručnú históriu a následne jeho komponenty. Nakoniec predstavíme OCR systém, ktorý budeme v našej práci používať - Tesseract.

2.1 Využitie v práci

Na to, aby sme dokumenty mohli akokoľvek analyzovať, potrebujeme ich mať uložené v takom formáte, aby boli strojom čitateľné. Musia byť teda uložené ako text. Všetky dokumenty, ktoré v našej práci spracúvame sú uložené vo formáte PDF. Avšak iba malá časť je uložená ako text. Vo väčšine prípadov sa jedna o obrázok – naskenovaný dokument. Preto prvým krokom je získanie textu z obrázku pomocou technológie OCR.

2.2 Popis OCR

OCR alebo optické rozoznávanie znakov sa zaoberá problémom klasifikácie optických vzorov v digitálnom obraze do príslušných alfanumerických či iných znakov [12]. Vďaka tomu sa uľahčí ich ukladanie, keďže namiesto množstva pixelov môžeme uložiť jeden alebo viac znakov. Ešte väčším benefitom je ale to, že takto uložený text môžeme prehľadávať, analyzovať a ľahko upravovať.

Tento problém bol na začiatku výskum rozpoznávania vzorov považovaný za jednoduchý [33]. Znakov je pomerne malé množstvo a ľahko sa s nimi pracuje. Problém

prichádza ak sa neobmedzíme iba na latinku a jeden font – v tom prípade množstvo vzorov narastá. Ešte väčší problém ale spôsobuje ručne písaný text a kvalita spracovaného obrazu. Problematika rozpoznávania znakov sa ukázala byť komplikovanejšou. V súčasnosti je už ale dostupných množstvo komerčných aplikácii vykonávajúcich OCR na pomerne dobrej úrovni [12]. Tieto aplikácie sú ale stále veľmi závisle na kvalite vstupného obrazu, preto obzvlášť pri spracovaní menej kvalitného obrazu sa stále nemôžu porovnávať v presnosti rozpoznania znakov s ľudskými schopnosťami [12]. V našej práci používame open source OCR engine Tesseract [3].

Najskoršie generácie OCR systémov sa spoliehali predovšetkým na techniky rozpoznávania obrazcov a spracovania obrazu [33], veľké zlepšenie ale prinieslo zapojenie metód umelej inteligencie. V nedávnej minulosti prinieslo ďalšie zlepšenie, podobne ako v iných oblastiach, využitie umelých neurónových sieti (ANN) [12]. Za hlavné kroky OCR by sa dali označiť [44]:

1. analýza rozloženia dokumentu
2. rozpoznanie znakov
3. post-processing

2.3 Prípravne kroky pred rozpoznávaním znakov

Shafait považuje analýzu rozloženia dokumentu (layout analysis) za prvý krok, ktorý OCR systém vykonáva [44]. Chaudhuri tento krok nazýva location segmentation, no ešte pred ním uvádza krok optical scanning [12].

Krok alebo komponent optical scanning vytvorí digitálnu verziu pôvodného dokumentu. Okrem toho je možné do tohto kroku zahrnúť techniku spracovania obrazu – prahovanie. Pomocou prahovania sa zo šedotónového obrazu vytvorí dvojúrovňový – čiernobiely obraz. Prahovanie môže byť lokálne alebo globálne. Pri globálnom sa nájde jedna konštanta slúžiaca ako prah pre celý dokument. Pri lokálnom sa naopak vyberá vytvorí viacero oblastí, ktoré majú samostatné prahové konštanty. V niektorých implementáciách môže mať každý pixel vlastnú oblasť a teda aj vlastnú prahovú konštantu [12].

Layout analysis alebo *location segmentation* sa snaží v dokumente identifikovať presne tie oblasti, ktoré obsahujú text. Takto sa odfiltruje napr. ilustračný obrázok umiestnený pri texte od samotného textu, ale aj biele oblasti, kde sa žiaden text nenachádza. Výstupom je obraz rozdelený na bloky, ktoré by mali obsahovať iba text. V závislosti od implementácie a dokumentu môže jeden blok obsahovať jedno slovo až jeden stĺp-

ček. Komplikáciou je ak súčasťou dokumentu sú aj tabuľky, čo sa týka aj časti nami analyzovaných dokumentov [12, 44].

Pre nás je najzaujímavejší krok pre-processing (predspracovanie), keďže v tomto kroku sa snažíme zlepšiť rozpoznanie textu niektorých dokumentov. Dokumentácia Tesseract enginu, ktorý v práci používame, totiž pre-processing odporúča pre zlepšenie výsledku vykonať dodatočný pre-processing ešte pred spustením Tesseractu – aj keď tento OCR systém sám používa rôzne metódy spracovania obrazu [3].

Metód pre-processingu je mnoho v závislosti od problému, či problémov daného obrazu. Jeho základným cieľom ale je aby bol obraz čitateľnejší pre ďalšie komponenty OCR. V krátkosti predstavíme niekoľko problémov, ktoré pre-processing rieši. Konkrétne riešenia ale vynecháme a v neskorších kapitolách predstavíme iba tie z nich, ktoré použijeme odchýliac sa od štandardných techník Tesseractu.

2.3.1 Odstránenie šumu

Jednou z najbežnejších metód je odstránenie šumu, ktorý každý digitálny obraz obsahuje. Časť šumu sa odstráni prahovaním, no po ňom môžu ostať napr. diery v čiarach, zaoblené rohy písmen a podobné artefakty [12, 44].

2.3.2 Normalizácia

Predovšetkým pri rozpoznávaní rukou písaného textu môže byť nápomocná normalizácia. Hoci aj súvislý rukou písaný text môže meniť otočenie, veľkosť či rovinu (riadok) na ktorej je text písaný. Okrem toho, častým problémom obzvlášť pri hrubších knihách býva zatočenie textu, ktorý bol predtým písaný v jednej rovine. Historické dokumenty zas môžu mať problém s vypúšťaním príliš veľkého množstva atramentu a teda s príliš hrubým písmom. Túto skupinu problémov rieši normalizácia [12, 44].

2.3.3 Kompresia

Bežné kompresné techniky pre obrázky nie sú vhodné pre rozpoznávanie znakov. Zároveň, určitá kompresia môže byť žiadúca pre zvýšenie rýchlosti spracovania, resp. učenia sa. Aj po kompresii musí byť OCR systém schopný rozpoznať tvar jednotlivých znakov. Preto ako kompresná technika sa používa už spomínané prahovanie alebo thinning. Vďaka prahovaniu sa dramaticky zmenší potrebné miesto na uloženie farby jedného pixelu. Cieľom však je zachytiť celý pôvodný znak, hoci v praxi sa často stáva, že niektoré šedé pixely z kraja jednotlivých znakov sa prahovaním vymažú. Thinning sa naopak nesnaží zachytiť celý pôvodný znak ale iba jeho kostru [12].

Ďalej nasledujú kroky, ktoré sa snažia po vyčistení dát v pre-processingu nájsť vhodnú reprezentáciu častí obrázku, pomocou ktorej systém ľahšie rozpozná znaky.

2.4 Rozpoznávanie znakov

Snahou je zaradiť rozpoznávanú vzorku do správnej triedy. Samotné rozpoznávanie jednotlivých znakov môže OCR systém vykonávať štyrmi základnými prístupmi, resp. ich kombináciou. Každý z týchto prístupov môže používať holistické alebo analytické stratégie. Holistické stratégie si nevyžadujú segmentáciu a pristupujú najprv k celému slovu až potom k jednotlivým znakom. Efektívne sú najmä pri ťažko segmentovateľných textoch, napr. písaných kurzívou. Analytické stratégie naopak vyžadujú segmentácie, keďže postupujú od jednotlivých znakov, či dokonca ich čŕt nahor k slovám a následne textu [12, 34].

2.4.1 Template matching

Tento prístup bol historicky prvý vo vývoji OCR systémov. Vytvoria sa pri ňom prototypy jednotlivých tried, ktoré môžu mať rôznu reprezentáciu. V závislosti od toho sa môže porovnávať miera zhody skupiny pixelov, zakrivenia či primitív. Bez ohľadu na spôsob reprezentácie, porovnávajú sa jednotlivé prototypy s obrázkom, ktorý sa má rozpoznať. V závislosti od komplexnosti systému sa môže porovnávať priamo s prototypmi alebo sa tieto prototypy môžu rôzne deformovať. Každopádne, rozpoznaný obraz sa zaradí do triedy s ktorej prototypom sa najlepšie zhoduje [12, 33].

2.4.2 Štatistický prístup

Pri tomto prístupe je nevyhnutné každý rozpoznávaný obrázok reprezentovať ako vektor príznakov. Tieto príznaky by mali byť vybrané tak, aby dovoľovali zaradenie do viacerých tried. Zároveň, podľa týchto príznakov by mali byť jednotlivé triedy separovateľné. Cieľom je naučiť sa na trénovacej množine tieto hranice. Prostriedkom môže byť klasterizácia, Bayesov či Markovov model [12, 22].

2.4.3 Syntakticko-štrukturálny prístup

Za týmto prístupom je snaha rekurzívne rozdeliť rozpoznávaný obrázok na primitíva. Komplexný obraz je tak reprezentovaný primitívami a vzťahmi medzi nimi. Z primitív sa pomocou pravidiel dajú vytvoriť inštancie jednotlivých tried [12, 22].

2.4.4 ANN

V súčasnosti najpoužívanejším prístupom v OCR sú umelé neurónové siete. ANN poskytujú predovšetkým možnosť masívneho množstvo paralelných výpočtov. Napriek rôznosti možných architektúr sa dá dokázať, že väčšina je ekvivalentná štatistickým metódam. V OCR systémoch sú najpoužívanejšími architektúrami dopredný viacvrstvový perceptrón a self-organizing map [12].

2.5 Post-processing

Posledným krokom, ktorý OCR systém vykonáva je post-processing. Využíva pritom techniky NLP na odhalenie a opravenie chýb, ktorých sa dopustil pri rozpoznávaní. Tieto chyby môžu byť také, že ako výstup rozpoznávania dostaneme slovo, ktoré

1. nie je skutočným slovom z daného jazyka
2. je slovom daného jazyka, no nie tým, ktoré bolo v rozpoznávanom dokumente

Ak sa jedná o prvý prípad, korekcia môže byť pomerne jednoduchá. Môže sa prehľadať slovník jazyka, prípadne vypočítať pravdepodobnosť, že niektoré písmena budú pri sebe. Napr. v slovenčine je nulová pravdepodobnosť, že ď a y budú vedľa seba a tak ak niečo také počas post-processingu nájdeme, s istotou bolo niektoré písmeno rozpoznané zle. Komplikovanejšia, no nie nemožná je korekcia v druhom prípade. Na to potrebujeme analyzovať kontext daného slova. Na to môžeme použiť rôzne štatistické modely ako ukazuje Tong [12, 49].

2.6 Tesseract

V našej práci používame open-source OCR systém Tesseract, ktorý bol vyvíjaný najskôr ako PhD projekt v spoločnosti HP. Táto spoločnosť neskôr prebrala vývoj systému až kým sa nestal open-source. Krátko na to nad ním prebrala záštitu spoločnosť Google, ktorá ho naďalej vyvíja ako open-source projekt [19, 36, 47].

Tesseract po pre-processingu extrahuje komponenty obrázku a ich obrysy organizuje do tzv. Blobov. Bloby sú organizované do riadkov textu. Riadky sú následné analyzované pre fixnú výšku textu. Rozdelenie riadku na slová sa vykonáva s prihliadnutím na rovnomerné, ale aj nerovnomerné medzery [47].

Samotné rozpoznávanie je dvojfázové, keďže Tesseract používa adaptívne rozpoznávanie. V prvej fáze sa rozpoznávajú rad za radom všetky slová. Tie, ktoré sú rozpoznané dostatočne dobre sa následne uložia ako dáta na trénovanie adaptívneho klasifikátora.

Ten sa používa až v druhej fáze, keď sa opäť prejde celá strana. Pri tomto druhom prechode sa už ale rozpoznávajú iba tie slová, ktoré neboli v prvej fáze rozpoznané dostatočne dobre. Nakoniec sa riešia nejasné medzery a alternatívne hypotézy pre výšku jednotlivých riadkov. Lingvistický post-processing je v Tesseracte iba minimálny [47].

Kapitola 3

Klasifikácia dokumentov

Jadrom našej práce je rozdelenie dokumentov do dvoch kategórii, podľa vzťahu aký popisujú. To je problém z oblasti spracovávania prirodzeného jazyka. Preto v tejto kapitole najprv popíšeme niektoré základné techniky NLP, ktorú sú používané naprieč rôznymi úlohami NLP. Ďalej predstavíme možnosti reprezentácie textu a nakoniec popíšeme metódy problematiky NLP, ktorá najviac dotýka našej úlohe - klasifikácia dokumentov.

3.1 Pre-processing

V tejto skcii predstavíme štandardné techniky predspracovania pri NLP systémoch. Budeme sa pridŕžať najmä pipeliny použitej v Standford CoreNLP s prihliadnutím na riešenia pre slovenský jazyk. Možnosti úprav textu a pridávania anotácii týmto nebudú vyčerpané, no predstavíme základné kroky a prípadné špecificky v našej práci využité postupy detailnejšie vysvetlíme neskôr [17, 29].

3.1.1 Tokenizácia

Štandardne sa ako prvý krok v NLP označuje tokenizácia, hoci napr. medzi nástrojmi NLP4SK môžeme nájsť aj službu na vyčistenie textu od netextových časí ako napr. hypertextových referencií. Pod tokenizáciou sa rozumie proces, ktorý text rozdelí na tokeny. Token je základná jednotka s ktorou sa v NLP pracuje. Tokenizácia je jazykovo závislá. V jazykoch, ktoré nepoužívajú žiaden oddeľovač medzi slovami, ako napr. čínština, je tokenizácia zložitejšia. V slovenskom jazyku je tokenizácia pomerne jednoduchá vďaka oddeľovačom ktorým je medzera. V kontexte slovenského jazyka je preto token väčšinou token ekvivalentný jednému slovu – postupnosti znakov medzi dvoma medzerami. Tokenizátor vyvíjaný na TUKE ale napríklad rozdeľuje zložené čísla (vyjadrené slovom) na viacero tokenov [6, 14, 32, 50].

3.1.2 Segmentácia viet

V tomto kroku sa snažíme rozdeliť postupnosť tokenov do viet. Výhodou je, že všetky vety končia interpunkčným znamienkom. Komplikáciou je, že za interpunkčným znamienkom môže veta pokračovať a to slovom s malým začiatočným písmenom ako aj veľkým. Príkladom sú skratky ako napr., po ktorých môže nasledovať aj vlastné podstatné meno [10, 29].

3.1.3 PoS značkovanie

Tento anotačný krok priradzuje k jednotlivým slovám vo vete ich gramatické kategórie. Aj táto úloha je silno jazykovo závislá a na jej náročnosť vplýva morfologické bohatstvo jazyka. Jedno slovo môže mať mnoho rôznych PoS tagov. Ktorý je v danej vete správny sa určuje na základe kontextu – okolitých slov. Problémom je, že v slovenčine, na rozdiel od angličtiny, pozícia slova vo vete neurčuje jednoznačne jeho POS tagy. Pre správne priradenie potrebujeme poznať jeho morfologický tvar a teda sufix [21].

Najbežnejším prístupom pri PoS značkovaní je štatistický. Takéto anotátory sú ale veľmi závislé na veľkosti manuálne anotovaných dát. Získavanie takýchto dát je veľmi prácne a zároveň pomerne odborne náročné. V slovenských podmienkach existuje ručne morfologicky anotovaný korpus vo veľkosti približne 1,2 milióna tokenov od Jazykovedného ústavu Ľ. Štúra Slovenskej akadémie vied [1, 21].

3.1.4 Lematizácia a stemovanie

Lematizácia aj stemovanie sa snažia o podobnú vec a totiž normalizovať slová, aby sme napr. nepovažovali malo a mala za rozličné slová, resp. tokeny. Stemovanie získava koreň slova na základe pravidiel, často používajúc aj odstránenie bežne používaných prípon v danom jazyku. Preto z *malo* aj *mala* vytiahne iba koreň *mal*. Lematizácia sa spolieha predovšetkým na gramatické pravidlá, či translačné matice medzi slovom a jeho základnou formou. Najmä pre jazyky, ktoré nemajú dostatočné manuálne anotované korpusy ale môže byť užitočná aj lematizácia s pomocou vektorových modelov. Lematizáciou dostaneme zo slov *malo* a *mala* tiež jedno slovo, no iné ako pri stemovaní – *mať* [18].

3.1.5 Rozpoznávanie pomenovaných entít

Nájdenie určitých pomenovaných entít je bežnou úlohou v NLP. Pri problémoch extrakcie informácii ako je ten náš hrá obzvlášť dôležitú úlohu. Entity, triedy do ktorých môžeme zaradiť jednotlivé vlastné podstatné mená sú obyčajne ľudia, organizácie, dátumy či miesta. Názov jednej entity sa môže skladať z viacerých tokenov. Pri hľadaní

triedy entity hrajú dôležitú úlohu štruktúrované zoznamy. Samy o sebe ale nie sú postačujúce. Unikátnych priezvisk je len v USA 1,5 milióna a aj na malom Slovensku vznikne každý deň niekoľko desiatok firiem. Vytvárať a udržiavať takéto zoznamy by bolo náročné a neefektívne. A tak hoci napr. Wikipedia môže poskytovať cenné informácie pre rozpoznávanie pomenovaných entít, používajú sa aj prístupy založené na príznakoch. Aj tieto prístupy ale využívajú tzv. gazetteer, zoznamy známych entít [15, 25, 30].

3.2 Reprezentácia textu

Keď už máme text normalizovaný a v nejakej miere anotovaný, prichádza na rad otázka reprezentácie celého dokumentu. Vo všeobecnosti sa reprezentáciou textu snažíme zachytiť početnosť slova v texte a ich blízkosť. Pričom pod blízkosťou sa myslí a) ich poradie vo vete a výskyt okolitých slov a po b) sémantická podobnosť. Jednotlivé reprezentácie nedokážu zachytiť všetky informácie a tak od výberu reprezentácie závisí aké informácie budeme mať k dispozícii. Preto sa reprezentácia textu vyberá podľa typu úlohy ktorú riešime. Pre ďalšie spôsoby pozri Zhang a Pérez-Iglesias [38, 51].

3.2.1 Bag of words

Pre klasifikovanie témy dokumentu sa ukazuje ako pomerne efektívna táto na výpočet i na uchovanie dát jednoduchá reprezentácia. Ťaží z toho, že v textoch sa vyskytujú vo veľkom množstve slová týkajúce sa danej témy. Napríklad ak nejaký text obsahuje slová ako *token*, *tokenizácia* či *PoS*, je veľmi pravdepodobné, že sa bude týkať NLP. V bag of words sa dokument reprezentuje jedným vektorom x , kde x_j je počet výskytov slova v danom dokumente. Dĺžka x je mohutnosť množiny slovníka. V niektorých prípadoch je potrebné x normalizovať pre lepšie porovnávanie naprieč dokumentmi. To môže spraviť vydelením x jeho najpočetnejším členom. Tým bude mať x_j hodnoty medzi 0 a 1. V tejto reprezentácii nemáme žiadnu informáciu o poradí slov. To znamená, že vety *A vlastní 100% podielu B* a *B vlastní 100% podielu A* budú mať úplne identickú reprezentáciu aj napriek tomu, že ich význam je opačný [15].

3.2.2 N-gramy

Jednou zo základných reprezentácii dokumentu sú n-gramy. Kým bag of words poradie slov zanedbáva, n-gramy ho uchováva, keďže zaznamenáva n-tice slov. N-gramy sú ale veľmi podobné BoW, mohli by sme ich nazvať zovšeobecnením BoW, keďže klasické BoW sa skladá z unigramov. Takto by sme ľahko odlíšili vety z predchádzajúceho príkladu. Prvá veta by pri najčastejšom type n-gramov – bi-grame bola reprezentovaná ako {*A vlastní, vlastní 100%, 100% podielu, podielu B*}. Druhá veta by bola odlišná:

{*B vlastní, vlastní 100%, 100% podielu, podielu A*} . Napriek týmto výhodám sa zdá, že používanie N-gramov pri klasifikácii dokumentu väčšinou prináša iba malé zlepšenie oproti používaniu unigramu a teda BoW. N-gramy sú ale využívané v iných problémoch, napr. tam kde potrebujeme predikovať pravdepodobné nasledujúce slovo, či zistiť, aké pravdepodobné je takéto usporiadanie slov. Všeobecne je ale problém n-gramov ich závislosť na veľkosti datasetu, keďže mohutnosť množiny možných n-gramov V^n, kde V je veľkosť slovníka, rastie s narastajúcim n exponenciálne [11, 23].

3.2.3 Sémantická vektorová reprezentácia

N-gramy dokážu v istej miere zachytiť kontext slova a vďaka tomu určiť pravdepodobnosť výskytu daného slova v špecifickom kontexte. V úlohe zachytenia významu, a to tak sémantického ako aj syntaktického, sa ale omnoho lepšie darí vektorovej reprezentácii slov, viet i celých dokumentov. Pre zachytenie sémantickej podobnosti sa ale používa vektorová reprezentácia slov. Vektory dokážu zachytiť, že napríklad slová *kráľ* a *muž* sú v podobnom vzťahu ako *kráľovná* a *žena*. Vo vektorovom zápise to znamená, že platí: kráľ – muž = kráľovná – žena [23, 24].

Táto reprezentácia je postavená na distribučnej hypotéze. Tá hovorí, že každé slovo možno poznať na základe slov, ktoré sa vyskytujú blízko neho. Ak potom máme vektor reprezentujúci slovo *kráľ*, dalo by sa povedať aj to, že máme vektor reprezentujúci slová zvyčajne sa vyskytujúce pri slove *kráľ*. Kontext tohto slova bude zrejme podobný ako pri *kráľovná*. [16, 15]

To ako veľa nám napovedá kontext slova najlepšie vidno vtedy, keď natrafíme na slovo, ktoré nepoznáme. Nech máme napríklad nasledujúce vety:

1. Vo svojej izbe mám iba jedno x.

2. Vďaka x cez deň nie je potrebné svietiť umelým svetlom.

3. Chcel som otvoriť x, aby som mohol dýchať čerstvý vzduch.

Z daného kontextu by malo byť zjavné, že x je okno. Toto okno môže ale rôzneho druhu - strešné, francúzke alebo iné. Okrem toho, v izbe mám tiež iba jedny *dvere*, ktoré tiež môžem chcieť otvoriť, aby som dýchal čerstvý vzduch. Slovo *dvere* ale nemôžme dosadiť do druhej vety. *Dvere* budú mať odlišný kontext a teda aj odlišnú reprezentáciu, aj keď v nejakej miere podobnú. Keďže je ale vektor, ktorým slová reprezentujeme mnohorozmerný, môžme zachytiť podobnosť v rôznych rovinách. Vďaka tomu dokáže vektorová reprezentácia zachytiť tak sémantiku ako aj syntax [15].

3.3 Metódy klasifikácie textu

Klasifikácia textu, resp. dokumentov je jedna z najzákladnejších vecí, ktoré môžeme s textom chcieť robiť. Aplikácii klasifikácie textu je mnoho, medzi najznámejšie patrí napr. klasifikácia mailov na *spam* a *ostatné*, či analýza sentimentu. Často využívanou ilustráciou využitia analýzy sentimentu je dataset recenzií filmov na stránke IMDB. Ten obsahuje spolu približne 50 000 silno polarizovaných recenzií, tzn. časť z nich je veľmi pozitívna kým druhá časť veľmi negatívna. Úlohou klasifikátora by v tomto prípade bolo rozdeliť recenzie na *pozitívne* a *negatívne*. Všeobecne, pri tomto probléme sa snažíme pre vstupný dokument D priradiť triedu $y \in Y$, kde Y je množina všetkých možných tried dokumentu. Takto by sme napr. k značkám jednotlivých recenzií mohli pridať napr. *neutrálne* [15, 28].

Z reprezentácii textu popísaných v sekcii 3.2 dostaneme príznaky. Tie potrebujeme na rozlišovanie medzi jednotlivými kategóriami. Prekvapivo, ak aj vezmeme za príznaky iba bag of words reprezentáciu daných textov, môžeme dosiahnúť pomerne dobrú klasifikáciu. Ak by sme si napr. vybrali recenziu, ktorá obsahuje slová ako *nudný, hrozný* či *zlý*, už len zo samotného faktu, že recenzia tieto slová obsahuje vieme povedať, že recenzia je pravdepodobne negatívna. Naopak, recenzia plná slov ako *úžasný, perfektný* či *krásny* bude pravdepodobne pozitívna. Aby sa ale dal odlíšiť význam slovných spojení ako *nebol nudný, nebol zlý*, môžme použiť alebo pridať aj ďalšie príznaky, ako napr. n-gramy [15].

3.3.1 Naivný Bayes

Najjednoduchším klasifikátorom je naivný Bayesov klasifikátor. Ten sa snaží nájsť pre zadané príznaky nájsť pravdepodobnosť triedy: $P(C_i|\mathbf{X})$ kde $C_i \in C$, pričom C je množina tried a $\mathbf{X}$ je vektor príznakov. To môžeme podľa Bayesovej vety zapísať ako $P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)*P(C_i)}{P(\mathbf{X})}$.

Tento klasifikátor využíva predpoklad, že všetky príznaky z $\mathbf{X}$ sú nezávislé, preto sa označuje za naivný. Vďaka tomu je ale v porovnaní s ostatnými jednoduchý na výpočet tak pri trénovaní, ako aj pri samotnom klasifikovaní. Pri využití tohto predpokladu môžeme zapísať $P(\mathbf{X}|C_i) = \prod_{i=1}^{n} P(X_i|C_i)$ [42].

Napriek mylnému predpokladu, naivný Bayesov klasifikátor v mnohých úloh dosahuje porovnateľné a v niektorých prípadoch dokonca lepšie výsledky ako iné používané klasifikátory [48].

Pre klasifikovanie dokumentu je dôležité vedieť určiť triedu pre zadaný dokument. Naivný Bayes pri učení naopak modeluje distribúciu príznakov pre dané triedy. Následne klasifikuje podľa vytvoreného modelu. Tento typ klasifikátorov sa nazýva generatívny. [15]

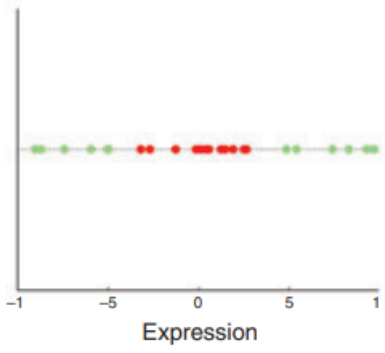
3.3.2 Stroj podporných vektorov

Stroj podporných vektorov (SVM) je diskriminatívnym klasifikátorom, teda pre jednotlivé triedy nehľadá distribúcie jednotlivých príznakov, ale iba hranice v príznakoch medzi jednotlivými triedami. Keďže príznaky reprezentujeme ako mnohorozmerný vektor, hranicou medzi triedami je oddeľujúca nadrovina. Keď SVM nájde správnu nadrovinu, určenie triedy je jednoduché, keďže nadrovina rozdelí priestor na dve časti - takú, ktorá patrí do danej triedy a takú ktorá nepatrí.

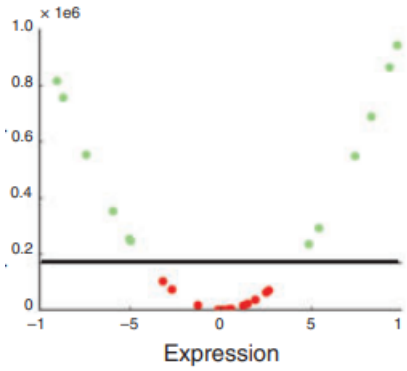
Nadrovín, ktoré rozdeľujú dve dané triedy je v spojitom priestore nekonečno. Na to, aby aby nadrovina správne rozdeľovala nie len testovaciu množinu, ale aj ďalšie dokumenty, SVM sa snaží vybrať nadrovinu s čo najväčším okrajom. To znamená takú, ktorá je v strede medzi oboma triedami.

Ďalším problém, s ktorým sa SVM vysporiadava sú odľahlé, eventuálne zašumené dokumenty. Ak by sa SVM snažilo zachytiť aj niektoré zašumené dokumenty, môže sa celková presnosť zhoršiť, preto sa môžeme rozhodnúť niekoľko odľahlých dokumentov ignorovať.

Nakoniec, môže sa stať, že dané dokumenty nie je možné v n-rozmernom priestore, kde n je počet príznakov, lineárne rozdeliť. Tento prípad ilustruje obrázok 3.1. V takom prípade môže pomôcť vhodná kernelová funkcia, pridá ďalšie rozmery, čo môže umožniť nájsť oddeľujúcu nadrovinu, ako ukazuje obrázok 3.2 [35].



Obr. 3.1: Lineárne neseparovateľne dáta v jednorozmernom priestore [35]



Obr. 3.2: Pôvodne lineárne neseparovateľné dáta v dvojrozmernom priestore [35]

3.3.3 Viacvrstvový perceptrón

Umelé neurónové siete môžu byť natrénované rozličné úlohy, medzi ktoré patrí aj klasifikácia dokumentov. Viacvrstvový perceptrón (MLP) sa skladá z troch alebo viacerých vrstiev: vstupnej, skrytej a výstupnej, pričom v skrytej vrstve môže byť viacero vrstiev, s rozličným počtom neurónov. Počet neurónov na vstupnej vrstve je rovný počtu príznakov, aby mohol každý príznak prichádzať do MLP prostredníctvom vlastného neurónu. Počet neurónov na výstupnej vrstve závisi od počtu tried, ktoré chceme klasifikovať. Väčšinou je rovný počtu tried, no ak chceme rozlišovať iba medzi dvoma triedami, postačí aj jeden neurón na výstupnej vrstve. Jednotlivé vrstvy sú prepojené tak, že každý neurón na i-tej vrstve má prepojenie s váhou $w_{i,j}$ na každý neurón z vrstvy $i+1$. Neuróny na jednej vrstve medzi sebou nie sú prepojené [39].

Na to, aby MLP mohol klasifikovať aj lineárne neseparovateľné triedy sa využíva aktivačná funkcia, ktorá je nelineárna - napr, sigmoid, tanh či ReLu. Pre učenie sa modelu je dôležitý algoritmus spätného šírenia chyby (backpropagation). Ten je založený na technikách gradientného zostupu. Vďaka nemu sa v MLP upravujú váhy tak, aby bola chyba v klasifikácii, či predikovaní čo najmenšia [39].

3.3.4 Klasifikácia s využitím predtrénovaných modelov

Sémantická vektorová reprezentácia slov radikálne zlepšila schopnosť reprezentovať slovo v počítači. Pre klasifikáciu celých dokumentov by to ale bolo málo. Avšak pomocou predtrénovaných modelov a metód ako je BERT, je možné dosiahnúť zlepšenie výsledokv aj v klasifikácii dokumentov ako ukazuje Adhikari [9] a Shaheen [45].

Kapitola 4

Návrh riešenia

V tejto kapitole si popíšeme nami navrhnuté riešenie. Pre overovanie presnosti a vhodnosti nášho postupu je nutné vytvoriť dataset preto sa v prvej sekcii tejto kapitoly budeme venovať. V nasledujúcich sekciách si popíšeme návrh automatizovaného systému, ktorý je určený na využitie uživateľom bez programátorských schopností. Následne popíšeme návrh procesu OCR a samotnej klasifikácie textu. Implementáciu narhnutého riešenia popíšeme v nasledujúcej kapitole.

4.1 Dataset

Na overenia správnosti každého klasifikačného problému je nevyhnutné mať dataset. Navyše, pre klasifikátor využívajúci učenie s učiteľom je dataset nevyhnutný na učenie. Keďže ale pre našu úlohu nebol vytvorený dataset, museli sme si vyrobiť vlastný. Okrem toho sme v snahe zlepšiť klasifikátor vytvorili aj dataset na úrovni viet. V nasledujúcich podsekciách popisujeme oba naše datasety.

4.1.1 Primárny dataset

Položkou primárneho datasetu je verifikačný dokument so štruktúrovanými dátami. Všetky štruktúrované dáta patriace k jednotlivým dokumentom sú uložené v jednom .csv súbore. Vzhľadom na kroky použité pri implementácii klasifikátora bol náš dataset vopred rozdelený na trénovaciu a testovaciu časť. Veľkosť datasetu zohľadňuje náročnosť jeho získavania. V časti určenej na trénovanie sa nachádza po 50 dokumentov z oboch tried - majiteľov aj štatutárov. Testovacia časť obsahuje rovanký počet majiteľov, avšak iba 24 štatutárov. Tento pomer sa viac približuje skutočnému pomeru jednotlivých tried v registri.

4.1.2 Dataset viet

Pre problém klasifikácie dokumentov je postačujúce mať dokumenty označené značkou, k akej ketegórii dokument ako celok patrí. Pre klasifikáciu dokumentov je ale typické jasnejšie delenie medzi dokumentmi - rieši skôr otázky ako: je tento dokument rozprávkou alebo vedeckým článkom o biológii? Avšak všetky dokumenty v RPVS sú právnickymi dokumentami. Aj keď to úplne nebráni ich variabilite, ich štruktúra je predpísaná zákonom (viď sekcia 1.2). Naše dokumenty sú rozdeľované skôr podľa vťahu medzi dvoma entitami, ktorý je v dokumente popísaný.

Pre problém extrakcie vzťahu medzi dvoma entitami je ale nedostačujúce, resp. príliš komplikované pracovať nad celým dokumentom. Vzťah štandardne extrahujeme z viet. Preto sme vytvorili tiež dataset obsahujúci vety, ktoré môžu vypovedať o tom, ako KUV bol zapísaný štatutár, prípadné že bol zapísaný majiteľ. Poslednou a najčastejšou možnosťou ale je, že veta nehovorí nič o vzťahu PVS a KUV. Tento dataset tvorí jeden .csv súbor. Jeho jednou položkou je veta, ku ktorej je priradená značka majiteľa, štatutára alebo neutrálna.

4.2 Automatizovaný systém

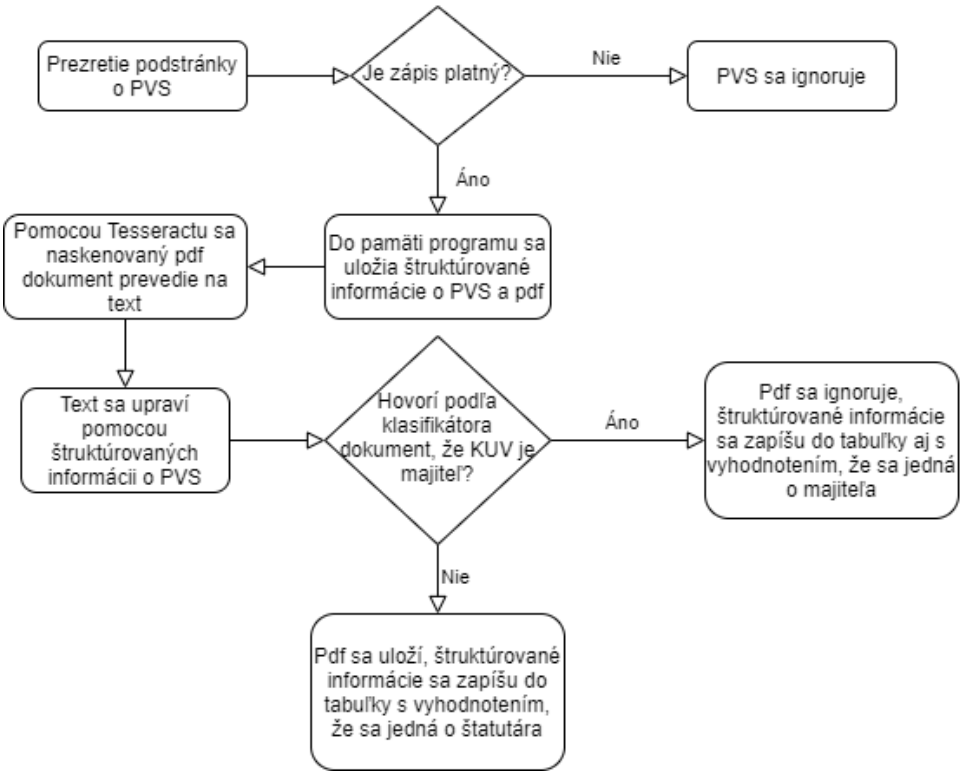
Užívateľské rozhranie systému je jednoduché, keďže dôraz v našej práci kladieme na klasifikáciu dokumentov. Užívateľ z príkazoveho riadku spustí automatizované spracovanie dát z registra. Systém nemusí prechádzať naraz všetky položky registra, ale napr. iba tie, ktoré boli pridané od posledného spracovávania.

Obr. 4.1 popisuje spracovávanie jednej položky z registra. Výsledkom spracovania je .csv súbor s údajmi a verifikačné dokumenty tých PVS, ktorých KUV sú štatutármi uložené v jednom priečinku.

4.3 Extrakcia dát z webu

Všetky dáta - štrukturované informácie o PVS, OS aj to, ktoré fyzické osoby sú zapísané KUV aj naskenovaný dokument, ktorý je v práci analyzovaný pochádza zo stránky *rpvs.gov.sk*. Tieto dáta sú pri behu programu postupne stiahnuté, vyhodnotené a v niektorých prípadoch uložené ako to ukazuje Obr. 4.1.

Každý PVS, ktorý získa zápis do RPVS má vlastnú stránku s adresou v nasledujúcom



Obr. 4.1: Pipeline aplikácie

tvare: *rpvs.gov.sk/rpvs/Partner/Partner/Detail/X*, kde X je číslo vložky. Číslo vložky znamená poradie zápisu, teda prvá spoločnosť zapísaná do RPVS ma číslo vložky 1 a každá ďalšia o jedno číslo väčšie.

Z týchto podstránok získavame dáta jednotlivých položiek registra. Po extrakcii dát z webu sa na ďalšie spracovanie postupuje slovník s informáciami spomínanými v úvode kapitoly a PDF dokument.

4.4 Prevod PDF dokumentov na text

Všetky verifikačné dokumenty sú vo formáte PDF. Iba malá časť z nich je typu vyhľadavateľné PDF - tzn. také PDF, z ktorého sa dá priamo vytiahnuť text. Väčšina dokumentov je tvorená naskenovanými dokumentami, teda obrázkami, väčšinou PNG, vloženými do PDF. Cieľom v tejto fáze je dostať z PDF dokumentu TXT súbor alebo jeden reťazec, vhodný na ďalšie spracovávanie.

Aby sme spracovali vyhľadávateľné PDF, PDF so skenmi a aj ich kombinácie, snažíme sa pri každom PDF získať text najprv z obrázkov a následne priamo z textu.

Pri extrakcii textu z obrázkov používame OCR systém Tesseract. Okrem toho používame vlastnú kontrolu toho, či Tesseract správne rozpoznal znaky z daného obrázka. V prípade, že nie, obrázky rotujeme a znova skúšame použiť Tesseract.

Následné získanie text prvej časti z vyhľadávateľného pdf je už triviálne. Spojením textu z oboch častí získavame výsledný text, s ktorým môžeme ďalej pracovať.

4.5 Klasifikátory

Jadrom našej práce je vytvorenie klasifikátora, ktorý správne rozdelí dokumenty podľa ich obsahu. Ako prvé predstavíme klasifikátor, ktorý sa využíva iba regulárne výrazy. Ďalej ukážeme o niečo modernejší typ klasifikátora, ktorý využíva učenie s učiteľom. Tento klasifikátor predstavíme v dvoch variantách. Keď budeme hovoriť o majiteľoch a štatutároch, budeme tým myslieť dokumenty ako boli definované v sekcii 1.2.

4.5.1 Klasifikácia pomocou regulárnych výrazov

Pre nájdenie vzorov charakteristických pre štatutárov i majiteľov bolo nutné zanalyzovať jednotlivé verifikačné dokumenty. Aby sme nezhnehodnotili relevantnosť výsledkov testovania, analyzovali sme iba dokumenty z testovacej množiny.

Pri návrhu klasifikátora sme vychádzali vedomosti získanej pri tvorbe datasetu, že drvivá väčšina dokumentov sú typu *majiteľ*. Preto sme sa v dokumentoch snažili nájsť také vzory, ktoré by dokazovali opak. Na to, aby bol dokument klasifikovaný ako *štatutár*, sa musí nájsť aspoň jeden reťazec spĺňajúci niektorý zo vzorov. Ak sa nenájde žiaden, dokument je klasifikovaný ako *majiteľ*.

Klasifikátor vykonáva pred samotným hľadaním niekoľko úprav textu. Tieto kroky bližšie popisujeme v sekcii 5.5:

1. Nahradenie vybraných entít ich všeobecnými značkami
2. Zjednotenie rôznych vyjadrení kľúčových fráz
3. Lematizácia
4. Odstránenie stop slov

Tabuľka 4.1: Návrh vzorov

Pred spracovaním	Po spracovaní
podmienky na zápis členov vrcholového manažmentu podľa ust. § 4 ods. ... sú splnené	podmienka zápis vmkuv ust . § 4 ods . . . . byť splnené
ako spoločnosti nepriamo ovládanej emitentom cenných papierov	spoločnosť nepriamo ovládanej emitentom cenných papier
sa zapisujú namiesto KÚV členovia vrcholového manažmentu	zapisujú namiesto kuv vmkuv
žiadna fyzická osoba nespĺňa definíciu konečného	nofo nespĺňa definícium konečného
neidentifikovala žiadne fyzické osoby ako kuv	neidentifikovala nofo kuv

Tabuľka 4.1 ukazuje, ako asi vyzerajú vzory napovedajúce, že analyzujeme štatutára. V tejto tabuľke je tiež možné vidieť, ako vyzerá text po aplikovaní vyššie uvedených krokov.

4.5.2 Klasifikácia pomocou učenia s učiteľom

Vďaka vytvorenému datasetu sme mohli vytvoriť aj klasifikátor využívajúci učenie s učiteľom. Tu uvádzame model, ktorý sa ukázal ako najúspečnejší.

Fáza pre-processingu je pri tomto klasifikátore jednoduchšia. Jediným krokom je nahradenie vybraných entít ich všeobecnými značkami ako to popisuje podsekcia 5.5.1.

Používame dva typy príznakov. Prvým typom sú n-gramy znakov v rámci jedného slova. Tzn. že jednym n-gramom môže byť len postupnosť znakov jedného slova alebo celé slovo. Druhým typom použitých príznakov sú bigramy slov. Extrakciu príznakov bližšie popisuje podsekcia TODO.
Na samotné klasifikovanie využívame viacvrstvový perceptrón (MLP) s dvoma skrytými vrstvami. Pri trénovaní používame na optimalizáciu váh modelu metódu L-BFGS [27].

4.5.3 Klasifikácia pomocou učenia s učiteľom s využitím označkovaných viet

Tento klasifikátor je veľmi podobný tomu, ktorý sme popísali v predchádzajúcej podsekcii. Jediný rozdielom je, pridanie jedného príznaku, ktorého hodnota je výsledkom ďalšieho klasifikátora, ktorý pracuje s označkovanými vetami. Spôsob získavania týchto

viet bližšie popisujú podsekcie 4.1.2 a **??**.

Na klasifikovanie viet používame rovnaký postup ako na klasifikovanie celých dokumentov. Rozdiel oproti klasifikátoru popísanemu v podsekcii 4.5.2 je iba v inak nastavených hyperparametroch.

Pre pripomenutie, klasifikátor viet rozdeľuje vety do troch kategórii: majiteľ, štatutár a neutrálne. Pre účely získania príznaku pre hlavný klasifikátor sa najprv zistí, koľko viet každého typu dokument obsahuje. Príznak potom dostaneme ako rozdiel počtu viet typu majiteľ a typu štatutár. Vety označené ako neutrálne sa ignorujú.

MLP klasifikátor teda dostáva na vstupe vektor príznakov zložený z hodnôt vyjadrujúcich n-gramy znakov, bigramy slov a rozdiel počtu viet klasifikovaných ako majiteľ a štatutár.

Kapitola 5

Implementácia

Táto kapitola opisuje implementácia riešenia navrhnutého v predchádzajúcej kapitole. Preto je jej štruktúra podobná štruktúre predchádzajúcej kapitoly. Program je implementovaný v pythone. Všetky odkazované .py moduly sú súčasťou zdrojového kodu (viď príloha 6.4).

5.1 Dataset

5.1.1 Primárny dataset

Pri tvorbe datasetu sme sa snažili čo najmenej zasahovať do výberu, ktoré dokumenty zaradiť do datasetu a ktoré nie. Tým sme chceli dosiahnuť, aby čo najviac zodpovedal dokumentom v registri. Okrem toho sme nechceli umelo zhromažďovať viac dokumentov, ktoré vyhotovala jedna OS, či dokumentov, ktoré mali rovnakého KUV. Ak by totiž všetky dokumenty vyhotovovala iba jedna OS, štruktúra by bola veľmi podobná a problém klasifikácie omnoho jednoduchší. Preto sme prechádzali od začiatku registra všetky platné záznamy, bez akýchkoľvek ďalších filtrov a priradzovali im označenia, až kým sa nenaplnil potrebný počet.

Vďaka tomuto postupu sme získali dobrú predstavu o tom, aké je rozloženie dokumentov, resp. firiem v registri. Prešli sme prvých 1100 v tom čase platných záznamov a z nich sme 74 identifikovali ako štatutárov. To znamená, že v RPVS je zapísaných 6-7% firiem, u ktorých je zapísaný ako KUV ich štatutár. Pri detailnejšej analýze dokumentov sme zistili, že asi tri dokumenty sme označili nesprávne. Tieto dokumenty sme z datasetu odstránili.

Tabuľka 5.1: Ručne označené vety

Majiteľ	Štatutár	Neutrálne	Spolu
117	121	673	911

5.1.2 Dataset viet

Vety sme získali z primárneho datasetu, avšak iba z jeho časti používanej na trénovanie. Samozrejme, neprešli sme všetky vety v 100 dokumentoch. Pomocou regulárnych výrazoch sme na označovanie vybrali pre nás zaujímavé vety. Tými boli tie, ktoré po úpravách popísaných v 5.5.2 a 5.5.1 obsahovali skratku *kuv* a zároveň *pvs*. Tieto vety boli rozdelené do troch kategórii. Ich počet, očistený o duplikáty pre prehľadnosť uvádzame v tabuľke 5.1.

5.2 Automatizovaný systém

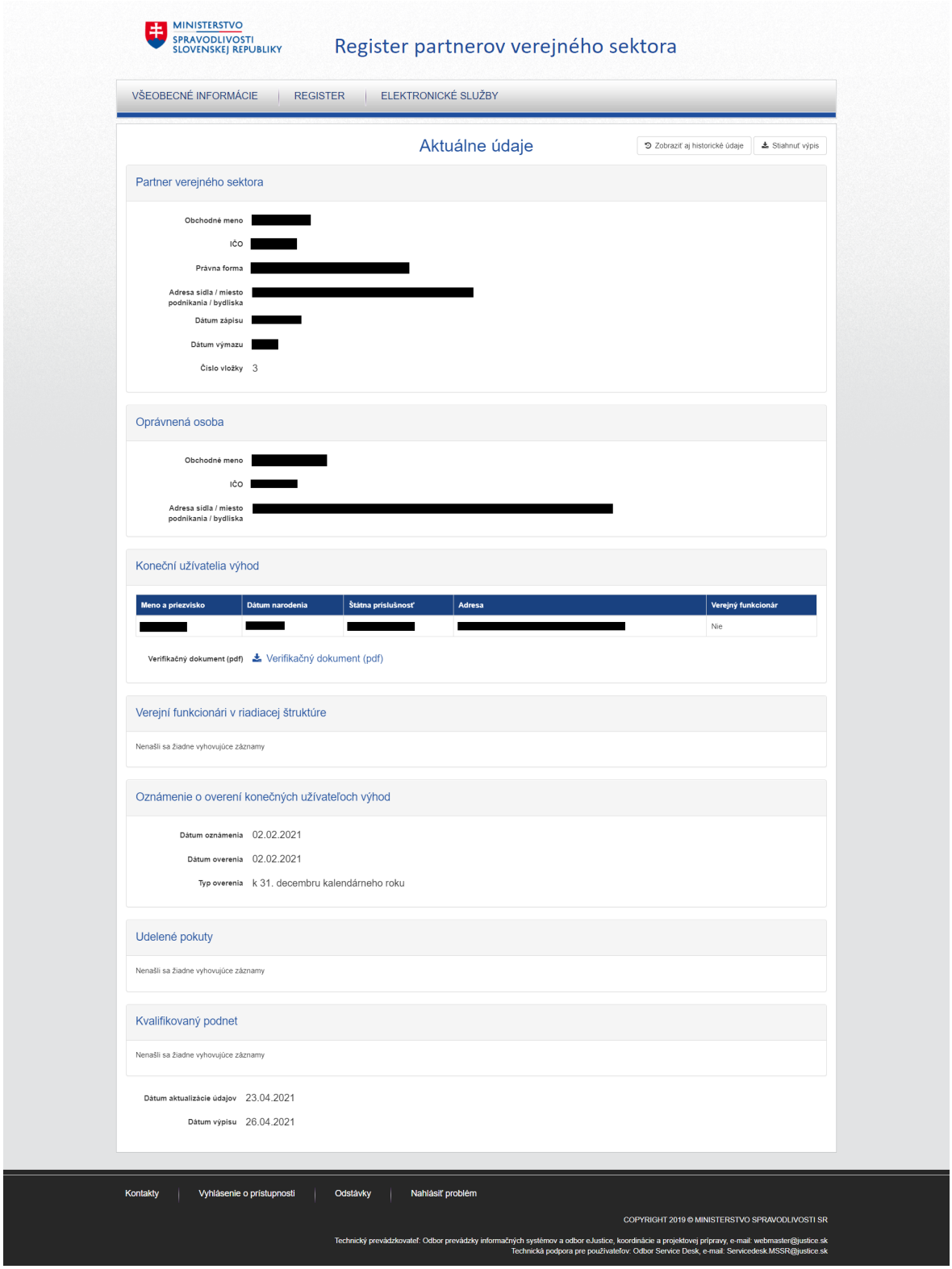
Jednoduchý modul *ui.py* spája funkcionalitu celého systému. Hlavnou funkciou ktorú poskytuje je *continue_where_stopped*. Tá najprv zistí ktorý PVS bol naposledy skontrolovaný, podľa údaju „Číslo položky“, aby mohol byť skontrolovaný nasledujúci. Použitím modulu, ktorý popisujeme v sekcii 5.3 získa štruktúrované informácie o nasledujúcom PVS a jeho verifikačný dokument. Ten následne pošle klasifikátoru popísanemu v sekcii **??**. Nakoniec, v závislosti od typu dokumentu uloží iba údaje alebo aj dokument s popisom, akej kategórie je dokument, resp. PVS.

5.3 Webscrapping

Všetok kód, ktorý súvisi so získavaním dát z RPVS možno nájsť v module *webscrapping.py*. Ten využíva na získanie HTML stránky *urllib3* [8] a na jej spracovanie *Beautiful Soup* [41]. Na uloženie PDF dokumentov využíva PyMuPDF [7].

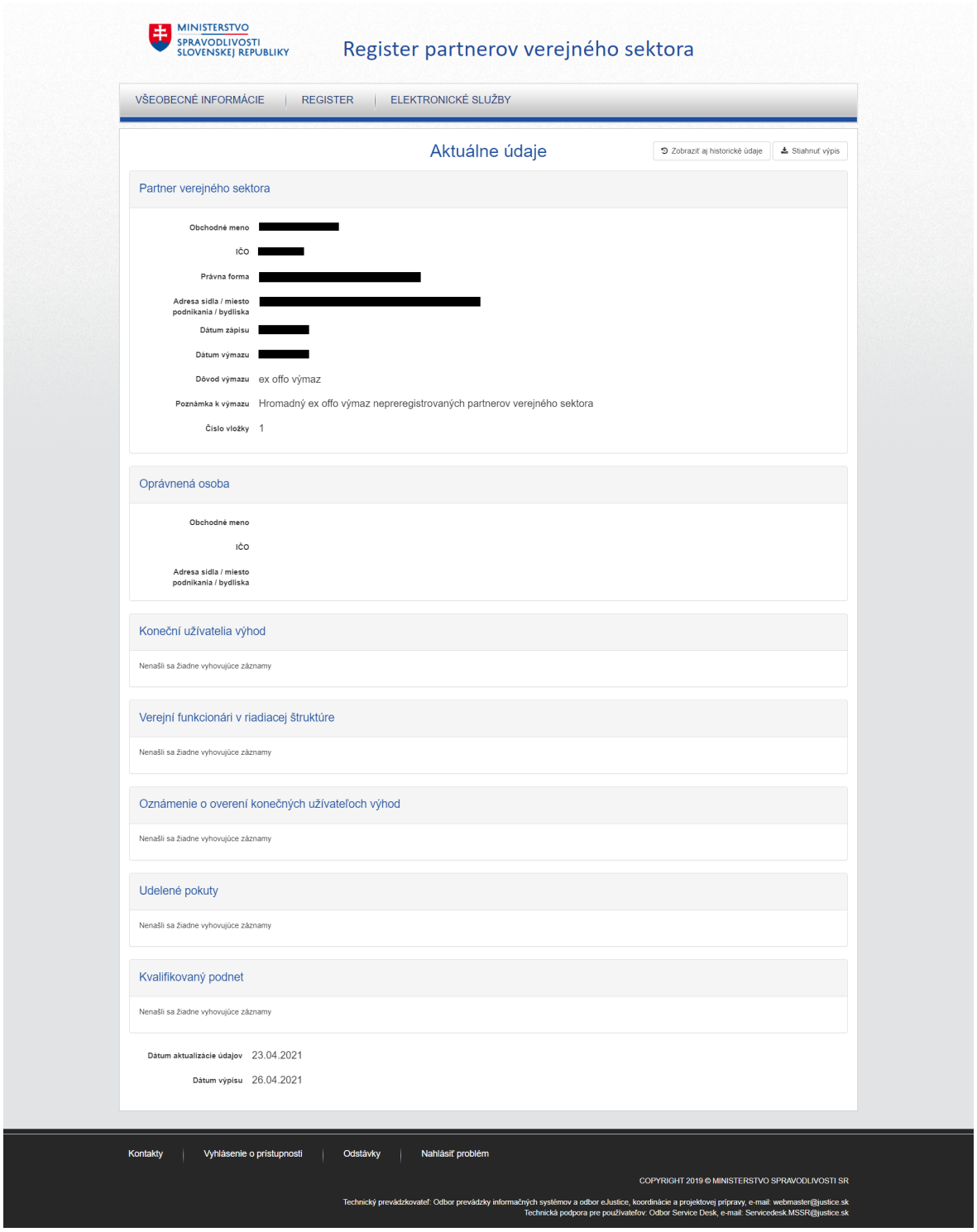
Informácie na podstránke PVS sú usporiadané do logických blokov HTML elementov *div* s triedou *panel panel-default*. K ním sa vieme jedoducho dostať vďaka *Beautiful Soup*. Pre nás sú zaujímavé bloky s názvami: *Partner verejného sektora, Oprávnená osoba* a *Koneční užívatelia výhod*. Z týchto blokov sa extrahujú dáta a vložia do slovníka.

Extrakcia informácii z blokov je pomerne jednoduchá, menšie komplikácie spôsobuje iba blok *Koneční užívatelia výhod*. Obsahuje mená konečných užívateľov výhod spoločne s ich osobnými informáciami. Okrem toho, obsahuje verifikačný pdf dokument.



Obr. 5.1: Detailná stránka PVS

Daný PVS ale môže mať neplatný zápis, v tom prípade nemá uvedeného žiadneho konečného užívateľa výhod ani verifikačný dokument. Ak ale zápis je platný, musí byť uvedený najmenej jeden KUV. Maximálny počet KUV ale nie je limitovaný. Mená



Obr. 5.2: Detailná stránka PVS, ktorého záznam je už neplatný

všetkých KUV sa preto uložia do jedného reťazca, oddelené týmito znakmi: „ | “. Bolo by bezpochyby prirodzenejšie využiť ako oddeľovač čiarku, no keďže mená niektorých KUV obsahujú pri tituloch aj čiarku, zvolili sme tento spôsob.

Posledný krokom v tejto časti je stiahnutie PDF dokumentu. To sa vykoná tak, že

najprv pomocou *Beautiful Soup* nájdeme URL daného dokumentu. Potom pomocou požiadavky typu 'GET' dokument stiahneme a uchováme ako PyMuPDF pdf dokument.

5.4 Prevod PDF dokumentov na text

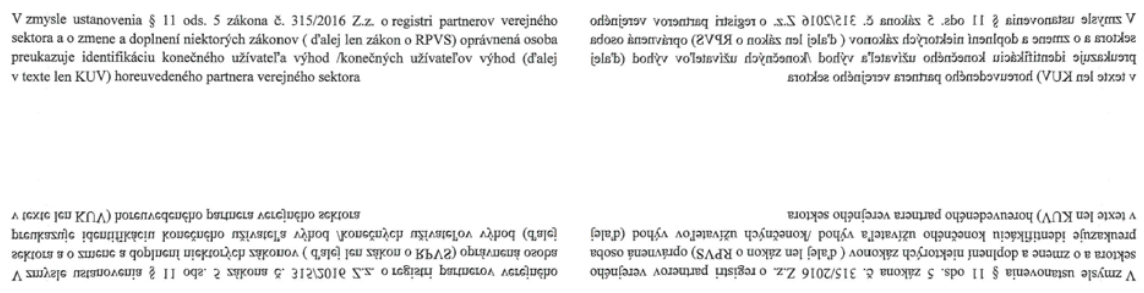
Všetok kód zabezpečujúci fukncionalitu popísanú v tejto sekcii je dostupný v module *ocr.py*. Ten je závislí predovšetkým od *Python-tesseract*, ktorý je popísaný v podkapitole 2.6. Ďalšími modulami, ktoré sú v tomto kroku využívané sú napr. *Pillow* na reprezentáciu obrázkov a už spomínaný PyMuPDF.

Pre používanie programu je pripravená funkcia *conver_to_text*, ktorá ako povinný parameter dostane cestu k PDF alebo samotný fitz dokument. Nepovinným parametrom je možné nastaviť, aby sa výsledný text uložil na disk. Toto je vhodné ak chceme testovať iba samotný klasifikátor nad testovacími dátami, keďže OCR je trvá pomerne dlho aj pri využití GPU. Na testovanie modul ocr.py poskytuje viaceré užitočné funkcie na testovanie úspešnosti klasifikátora, ktoré napr. prejdú celým priečinkom, vytiahnú text z TXT súborov a vrátia zoznam reťazcov - jeden pre každý dokument z priečinku.

Funkcia *conver_to_text* najprv zjednotí dokument prijatý ako parameter na fitz dokument. Následne z dokumentu vytiahne všetky obrázky takým spôsobom, že prejde cez všetky strany dokumentu a na každej strane využije metódu triedy *Page, getImageList()*. Tá vráti zoznam obrázkov z danej strany. Tie sú uložené ako *Pillow.Image* objekt a vložené do jedného zoznamu, ktorý je spoločný pre všetky strany daného dokumentu.

Ak je tento zoznam neprázdny, nad každou položkou, teda obrázkom sa zavolá funkcia *pytesseractu - image_to_string*, ktorá vracia reťazec. V niektorých PDF sú ale obrázky rôzne rotované, ako to ukazuje obr. 5.3. V tom prípade bude výsledný reťazec nesprávny. Preto v tomto kroku reťazec kontrolujeme. Keďže sme si všimli, že pri zle otočenom obrázku tesseract rozpoznáva veľa veľkých písmen, kontrolujeme výskyt veľkých písmen na začiatku rozpoznaného reťazca. Ak je príliš veľký, skúsime obrázok inak otočiť a nad takým obrázkom opäť zavolať *pytesseract* funkciu a skontrolovať ho. Takto skúsime viacero rotácii. Ak je v nejakej rotácii výsledok vyhovojúci, použije sa. Inak sa použije reťazec získaný z prvého, nijak nerotovaného obrázku. Toto sa vykoná postupne na každom jednom obrázku zo zoznamu a postupne sa vytvorí vytvorí jeden reťazec.

V ďalšom kroku program opäť prechádza cez všetky strany. V tomto prípade sa už ale



Obr. 5.3: Rotátice obrázkov v dokumentoch

nepozerá na obrázky, ale iba na text, ktorý je možné vytiahnuť priamo z PDF.

Nakoniec sa oba reťazce spoja tak, že k reťazcu získanému z obrázkov sa pripojí reťazec získaný priamo z PDF. To môže spôsobiť nesprávne usporiadanie textu, keďže text z obrázkov, aj keby bol v závere PDF, bude na začiatku. Tento problém sme ale vyhodnotili ako menej závažný ako keby sme text z obrázkov a z PDF pomiešali, čo by bola alternatíva k nášmu postupu.

5.5 Klasifikácia pomocou regulárnych výrazov

V podsekcii 4.5.1 sme ukázali základnu štruktúru tohoto klasifikátora. V tejto sekcii najprv predstavíme v akom module a s akými závislosťami bol implementovaný. Následne prejdeme jednotlivými vykonávanými úpravami a nakoniec ukážeme použité regulárne výrazy. Keďže implementácia klasifikátora bola jednoduchšia ako analyzovanie a hľadanie vzorov, v tejto sekcii popisujeme aj túto analýzu a postup, ako sme sa dostali práve k výsledným vzorom.

Pre klasifikátor založený na regulárnych výrazoch sme vytvorili v module pattern.py triedu *PatternExtract*. V tomto module sa využíva predovšetkým knižnica stanza. Pomocou nej sú dokumenty tokenizované a lematizované.

5.5.1 Nahradenie vybraných entít ich všeobecnými značkami

Na to, aby klasifikátor bol schopný lepšie klasifikovať nie len dokumenty z trénovacej či testovacej množiny, je vhodné zovšeobecniť výrazy špecifické iba pre jeden dokument.

Tabuľka 5.2: Entity zo štrukturovaných dát so značkami

Značka	Entita	Značka v typickom NER
KUV	Meno a priezvisko KUV	PERSON
PVS	Názov PVS	ORG
OS	Názov OS	ORG/PERSON
ADDR	Adresa sídla PVS	ADDRESS

Všeobecne sa pri spracovaní prirozdeného jazyka využíva NER [26]. Pri tomto procese sa k jednotlivým pomenovaným entitám priradia značky vyjadrujúce ich druh - napr. osoba, adresa či organizácia. Keďže je NER zložitejšou úlohou, ktorú *stanza* pre slovenský jazyk nepodporuje, implementovali sme iba veľmi zjednodušený kvázi NER. Využívame štruktúrované dáta, ktoré sme získali pri sťahovaní dokumentu, ako sme popísali v kapitole 5.3. V tabuľke 5.2 sú uvedené údaje, značky ktoré pužívame aj s ich významom. Okrem toho v nej uvádzame, aké by mali tieto entity značky v typickom všeobecnom NER.

Príkladom toho, čo je v tejto fáze nahradzované sú mená konečných užívateľov výhod a partnera verjného sektora ich značkami. Uvažujme ako príklad firmu O_1 ktorá má podľa RPVS troch KUV - M_1, M_2, M_3. V jej verifikačnom dokumente nájdeme, že 10% z nej vlastní firma O_2. Okrem toho nájdeme vetu, podľa ktorej 'M_4 je jediným spoločníkom O_2'. Z toho jednoznačne vyplýva, že M_4 je konečným užívateľom výhod firmy O_2. Ak by sme nepoužívali žiaden NER, museli by sme v tomto prípade hľadať iba vzor 'je jediným spoločníkom'. To by nás v tomto prípade priviedlo k chybnému výsledku, keďže M_4 môže byť aj inou spoločnosťou, čo nám nijako nepomôže. Ak by sme vďaka NER vedeli, že M_4 je osobou a O_2 organizáciou, boli by sme na tom už lepšie, no stále by sme potrebovali mimotextové informácie o tom, či je pre klasifikáciu daného dokumentu naozaj kľúčový vzťah medzi M_4 a O_2.

Avšak vzory, ktoré sme použili, sa zameriavajú na nájdenie potrvrdenia, že sa zapisuje ako KUV štatutár. Takéto vety sú statické a neobsahujú zmienku o konkrétnych KUV a PVS. Preto v našej implementácii táto úprava nijako nezlepšuje klasifikovanie.

Väčší potenciál využitie týchto mimotextových informácii by podľa nášho názoru bol pri hľadaní vzorov potvrdzujúcich, že dokument je typu majiteľ. Táto hypotéza zostala neoverená. Prínos využitia mimotextových informácii sa ale ukázal pri využití klasifikátorov používajúcich učenie s učiteľom, ako ukazujeme v podsekcii 5.6.1.

5.5.2 Zjednotenie rôznych vyjadrení kľúčových fráz

Niektoré nami navrhnuté vzory nezachytili hľadané reťazce iba kvôli malým štylistickým odlišnostiam alebo malým chybám pri OCR. Tieto problémy by sa dali vyriešiť aj napísaním všeobecnejších regulárnych výrazov, avšak pre lepšiu prehľadnosť sme namiesto toho tieto problémy riešili vo fáze predspracovania.

Takto sme napr. napravili častú chybu OCR, kedy bola nesprávne rozpozná skratka PVS ako PYS či PV5. Ďalej sme výrazy ako *členovia jeho vrcholového manažmentu* či *člen predstavenstva štatutárneho orgánu* zamenili za jednotnú skratku „vmkuv“ - vrcholový manažment partnera verejného sektora.

Ďalšou funkciou tu použitých regulárnych výrazov je zachytenie viacerých slov, ktoré majú rovnaký koreň. Vlastné stemovanie sme vo vybraných prípadoch použili, z toho dôvodu, že lematizácia pomocou *stanza* sa ukázala ako nedostatočne presná, ako možno vidieť aj na tabuľke 4.1.

Ako príklad uvádzame časť z použitého regulárneho výrazu: „.tatut.r[^]“. Účelom tejto časti je rozpoznať slová ako *štatutárny, statutárny, štatutárneho*. Vďaka použitému znaku bodky v regexe nezáleží, či OCR rozpozná písmená „š“ a „á“ správne. Podobne nezáleží na tom, v akom gramatickom tvare bude slovo „štatutárny“.

5.5.3 Lematizácia a odstránenie stop slov

Na lematizáciu používame knižnicu *stanza*. V tom istom kroku zároveň odstraňujeme stop slová. Prostredníctvom *stanzy* získame všetky tokeny. Následne prechádzame cez všetky tokeny. Nepozeráme sa ale na pôvodný text, ale na lemmu daného tokenu. Ak s nenachádza medzi stop slovami, pripojíme ju do nového reťazca. Ešte predtým ale skontrolujeme, či sme lematizáciou neodstránili zápor - tzn. či sme z „nebol“nespravili „byť“. Ak lematizácia odstránila záporny tvar, pridáme na začiatok slova predponu „ne“.

Základný zoznam stop slov sme prebrali z github repozitára projektu lab.SNG [2]. Tento zoznam sme následne upravili, keďže obsahoval aj slová ako je, či má, ktoré pre našu prácu sú potrebné a ich odstranením by sme stratili cenné informácie. Ďalej sme tieto slová lematizovali, vďaka čomu vznikli v zozname duplikáty, ktoré sme odstránili. Okrem toho sme zo zoznamu odstránili cudzie slová, ktoré tam boli zaradené omylom.

Tabuľka 5.3: Úspešnosť jednotlivých vzorov pri klasifikovaní štatutárov

Regulárny výraz	Trénovacia množina			Testovacia množina		
	precision	recall	f1	precision	recall	f1
ovl.d[^]* emitent[^]*	1	0,08	0,15	1	0	0
6a ods \. 2	1	0,68	0,81	0,91	0,42	0,57
namiesto kuv	0,92	0,22	0,35	0,75	0,25	0,375
nofo (pvs)*nesp..a (krit.r(^)*|podmienk(^)*)	0,94	0,34	0,5	0,9	0,375	0,53
pov[^]* vmkuv	1	0,66	0,8	0,83	0,42	0,56

5.5.4 Rozhodujúce regulárne výrazy

V dokumentoch typu štatutár z trénovacej množiny sme našli niekoľko podobných viet, ktoré s menšími obmenami opakovali vo viacerých dokumentoch. Jadro z týchto vzorov zobrazuje tabuľka 4.1. Po ich nasadení a otestovaní sme výsledky analyzovali tak, že pri všetkych nesprávne klasifikovaných majiteľoch sme sa snažili nájsť v danom dokumente tú vetu, na základe ktorej bol celý dokument klasifikovaný ako majiteľ. Pri nesprávne klasifikovaných štatutároch sme sa naopak snažili nájsť vety podobné tým, ktoré naše regulárne výrazy zachytávali. Okrem toho sme ďalej hľadali nové určujúce vety.

Výsledkom tejto analýzy bola úprava pôvodných výrazov a pridanie nových. Zistili sme, že vzory tak ako sme ich navrhli na začiatku, boli príliš dlhé a špecifické. Preto sme ich skrátili a z niektorých slov uprostred vzorov sme spravili nepovinné, čím získali väčšiu flexibilitu. Pridali sme nové vzory, napr. na základe vety: '*V takýchto prípadoch sa za konečných užívateľov výhod považujú členovia vrcholového manažmentu*', ktorá sa v rôznych obmenách opakovala, sme doplnili nasledujúci regulárny výraz: *pov[^]* vmkuv*.

Výsledné regulárne výrazy sú zobrazené v tabuľke 5.3. Tabuľka okrem toho zobrazuje ako úspešné boli jednotlivé vzory v klasifikovaní štatutárov. Porovnanie výsledkov na testovacej a trénovacej množine ukazuje, že naše regulárne výrazy pomerne dobre klasifikujú analyzované dokumenty, no nie sú tak všeobecné, aby dokázali porovnateľne úspešne klasifikovať všetky dokumenty z registra. Celkové výsledky klasifikátora aj s porovnaním s ostatnými nami implementovanými klasifikátormi uvádzame v kapitole **??**.

5.6 Klasifikácia pomocou učenia s učiteľom

V tejto sekcii si predstavíme implementáciu klasifikátora navrhnutého v podeskcii 4.5.2. Najprv predstavíme modul v ktorom bol klasifikátor implementovaný aj s jeho závislosťami. Následne popíšeme aké kroky klasifikátor vykonáva a ukážeme, aké hyperparametre sme vyhodnotili ako najoptimálnejšie.

Tento klasifikátor je implementovaný v triede *SupervisedClassifier*, ktorú možno nájsť v module *slearning.py*. Pre funkcionalitu je kľúčová knižnica *sickit-learn* [37]. V menšej miere je nápomocná aj knižnica *numpy* [20].

5.6.1 Pipeline klasifikátora

Pre prístup ku klasifikátoru sú dôležité dve funkcie: *train* a *is_owner*, ktorá je iba konkrétne formulovaná evaluačná funkcia.

Funkcia *train* zabezpečuje natrénovaný model, pričom poskytuje dve možnosti. Môžeme zadať cestu k oddeleným súborom, majiteľov a štatutárom, alebo cestu k už natrénovanému modelu, ktorý sa načíta. V prípadem že chceme model nanonovo natrénovať, *train* najprv načíta dáta z priečnikov majiteľov a štatutárov. Načítané dáta majú podobu dvoch zoznamov. Prvý je tvorený reťazcami, kde pre každý dokument je jeden reťazec, ktorý obsahuje všetok text daného dokumentu. Pri načítaní textu sa nahradzujú vybrané entity ich všeobecnými značkami, ako to popisuje podsekcia 5.5.1. Druhý zoznam obsahuje nuly alebo jednotky - ktoré označujú do akej triedy daný dokument patrí. Po načítaní sa texty prevedú dvoma rôznymi spôsobmi na vektor, pomocou *sklearn* triedy *CountVectorizer*. Prvý spôsob rozdelí slová na znaky, pričom berie do úvahy postupnosť 1 až 8 znakov, ktorá sa vyskytuje v 7-96% dokumentoch. Druhý spôsob počíta výskyt dvoch posebe idúcich slov (bigramy), ktoré sa vyskytujú najmenej v 5% dokumentov z trénovacej množiny (pozri tabuľka 5.4). Oba tieto vektory sú nakoniec spojené do jedného. Na takto reprezentovaných dokumentoch sa nakoniec trénuje *MLPClassifier* zo *sklearn*.

Pri funkcii *is_owner* je postup podobný, avšak ako parameter potrebujeme názov PDF, ak ho máme stiahnuté aj s metadátami alebo dokument a metadáta. Dokument sa pomocou úprav transformuje do reprezentácie ako pri trénovaní, aby sa mohol využiť natrénovaný model na klasifikovanie.

Tabuľka 5.4: Parametre CountVectorizer

min_df	max_df	ngram_range	analyzer
0.07	0.96	1, 8	char_wb
0.05	1	2	word

5.6.2 Hyperparametre

Pri trénovaní *MLPClassifier* používame na hľadanie optima L-BFGS, ktorý je na tak malom datasete akým disponujeme najpresnejší a zároveň pomerne rýchly. Ako aktivačnú funkciu využívame ReLu. Skrytá vrstva obsahuje 100 neurónov na prvej a 50 neurónov na druhej vrstve.

5.7 Klasifikácia pomocou učenia s učiteľom s využitím označkovaných viet

Keďže jediný rozdiel medzi týmto klasifikátorom a tým, ktorý sme popísali v predchádzajucej sekcii je využitie označkovaných viet, v tejto sekcii sa budeme sústrediť práve na tento rozdiel.

5.7.1 Pipeline klasifikátora

Vo fáze trénovania sa najprv natrénuje klasifikátor viet. Pri trénovaní sa načítavajú vety priamo z tabuľky. Aj pri tomto klasifikátore sa využíva dvakrát *CountVectorizer* a *MLPClassifier*, avšak s odlišnými hyperparametrami.

Keď je klasifikátor viet natrénovaný, nasleduje trénovanie hlavného klasifikátora. Pritom sa text prevedie na vektor ako to ukazuje sekcia 5.6. K tomuto jedna hodnota podľa výstupu klasifikátora viet. Tá sa získava tak, že v dokumente sa vhľadajú také vety, ktoré po úpravách popísaných v podsekcii 5.5.2 obsahujú zároveň skratky „KUV“ a „PVS“. Tieto vety sa vyhodnotia pomocou klasifikátora viet. Takto sa zistí počet neutrálnych viet, počet viet typu majiteľ a typu štatutár. Nakoniec sa od počtu viet typu majiteľ odčíta počet viet typu štatutár. Táto výsledná hodnota sa pridá do vektora reprezentujúceho dokument, s ktorým následne pracuje MLP model. Podobne sa dokument transformuje na vektor aj vo fáze predikovania pri funkcii *is_owner*.

5.7.2 Hyperparametre

Hyperparametre hlavného klasifikátora aj vektorizéra sú totožné s tými popísanými v podsekcii 5.6.2. Klasifikátor viet pužíva mierne odlišné hyperparametre. *CountVectorizer* ktorý spočítava n-gramy znakov berie do úvahy n-gramy v rozsahu 1-9, vyskytujúce sa v 6-95% dokumentov. Pri n-gramoch slov sa berú do úvahy n-gramy rozsahu 2-3 s výskytom v 3-75% dokumentov z trénovacej množiny. Aj v tomto prípade používame *MLPClassifier* s L-BFGS, keďže dataset nie je oveľa väčši. Malý rozdiel je v skrytej vrstve, ktorá má 90 a 45 neurónov.

Kapitola 6

Výsledky a diskusia

V tejto kapitole najprv popíšeme spôsom a metriky, ktoré využívame na vyhodnotenie úspešnosti klasifikátorov. Následne popíšeme výsledky jednotlivých klasifikátorov. Potom sa pozrieme na výsledky klasifikátora viet a nakoniec predstavíme návrhy na vylepšenie našej implementácie, ktoré sme už nestihli vykonať.

6.1 Spôsob evalauácie klasifikátorov

Keďže je náš dataset, ako aj celý register, nevyvážený v mohutnosti tried, jednoduchá evaluačná metóda ako presnosť (accuracy - viď vzorec 6.1) by nám signalizovala príliš optimistické výsledky. Pri našom dateste by sme bez námahy označením všetkých dokumentov ako „majiteľ“ by sme získali presnosť 68% [31].

$$accuracy = \frac{correct\ predictions}{all\ predictions} \tag{6.1}$$

$$F1 = \frac{2 * P * R}{P + R} \tag{6.2}$$

Preto sme pri vyhodnocovaní používali metriku F1 skóre, ktorá berie lepšie do úvahy nevyváženosť tried. F1 skóre je definované ako harmonický priemer *precision*(P) a *recall*(R) (viď 6.2). *Precision*, definovaná vo vzorci 6.3 vyjadruje pomer dokumentov správne zaradených do danej triedy (TP) so všetkými dokumentami, ktoré boli zaradené do danej triedy (TP + FP). *Recall*, definovaný vo vzorci 6.4, vyjadruje pomer správne zaradených dokumentov do danej triedy (TP) so všetkými dokumentami danej triedy (TP + FN) [43, 13].

$$precision = \frac{TP}{TP + FP} \tag{6.3}$$

Tabuľka 6.1: Prehľad úspešnosti klasifikátorov

	Majiteľ			Štatutár		
Klasifikátor	precision	recall	f1	precision	recall	f1
Regex	0.86	0.86	0.86	0.71	0.71	0.71
MLP - iba BoW	0.92	0.98	0.95	0.95	0.83	0.89
MLP - s vetami	0.91	0.96	0.93	0.9	0.79	0.84
MLP - iba vety	0.88	0.58	0.7	0.49	0.83	0.62

$$recall = \frac{TP}{TP + FN} \tag{6.4}$$

Pre účely našej práce je dôležité najmä nájsť dokumenty typu štatutár. Tzn. mať pri štatutároch čo najvyšši recall, no so zachovaním primeranej precision. Ako ale ukazuje tabuľka 6.1, pri všetkých klasifikátoroch sme dosiahli recall rovnaký alebo o niečo horší ako precision. Ak nenapíšeme inak, budeme rozoberať hodnotenie klasifikovania štatutárov, takže v zmysle vzorca 6.2 sú dokumenty označené za štatutárov *positive* (TP).

6.2 Výsledky klasifikátorov

Všetky klasifikátory sme vyhodnocovali na množine dokumentov pozostávajúcej z 50 majiteľov a 24 štatutárov.

Najúspešnejší klasifikátor, popísaný v sekcii 5.6 dosiahol f1-skóre 0,89 s recall 0,83 a precision 0,95. To znamená, že spomedzi 74 dokumentov určených na testovanie označil 21 ako štatutárov, z nich 20 správne. Ďalej 4 dokumenty označil nesprávne ako majiteľov. Samozrejme, hľadali sme najvhodnejšie hyperparametre pre náš dataset a toto sú najlepšie dosiahnuté výsledky s hyperparametrami, ktoré sme už predstavili v sekcii 5.6.

V snahe zlepšiť recall sme skúsili využiť aj klasifikovanie viet (viď tabuľka 6.1 - MLP - s vetami). To však nebolo úspešné a precision aj recall iba zhoršilo. Dôvodom je zrejme predovšetkým nedostatočná presnosť klasifikovania viet, ktorú bližšie popisujeme v nasledujúcej sekcii.

Jurafsky a Martin tvrdia, že klasifikátory založené na regulárnych výrazoch majú výhodu vysokej precision, zatiaľ čo k ich nevýhodam patrí nízky recall a pracnosť hľadania obrazcov vyjadrených regulárnymi výrazmi [23]. V našej práci sa nám potvrdila iba časová náročnosť získavania obrazcov. Precision aj recall sú pri tomto klasifikátore rov-

Tabuľka 6.2: Confusion matrix klasifikovania viet

	Predikované		
Skutočné	Neutárlne	Majiteľ	Štatutár
Neutárlne	71	6	23
Majiteľ	5	12	0
Štatutár	3	0	18

naké, pričom recall oproti očakávaniu relatívne vysoký. Úspešnosť jednotlivých vzorov zobrazuje tabuľka 5.3.

6.3 Výsledky klasifikovanie viet

Výsledky klasifikátora viet uvádzame v dvoch tabuľkách. Ako ukazuje tabuľka 6.1, tento klasifikátor má pri štatutároch pomerne slabé F1-skóre. Stále má ale pomerne zaujímavý recall, najmä ak uvážime, že jeho účelom je iba dopĺňať hlavný klasifikátor s cieľom zvýšiť jeho recall.

Keď sa pozrieme bližšie na chybovosť klasifikátora zobrazenú v tabuľke 6.2, zistíme pozitívnu informáciu, že klasifikátor nikdy nezamenil vety, z ktorých jednoznačne vyplýva, že dokument je typu majiteľ značkou štatutár a naopak. Všetka chybovosť spočíva iba v zamenení neutrálnych viet s majiteľmi, resp. štatutármi a naopak. Napriek tomu musíme skonštatovať, že náš pokus s využitím klasifikátora viet nebol úspešný, aj keď v ňom vidíme potenciál pre ďalšie zlepšovanie klasifikovania dokumentov.

6.4 Návrhy na ďalšiu prácu

Vo fáze OCR vidíme priestor na zlepšenie rozpoznania zle otočených PDF dokumentov.

V nahradzovaní vybraných entít ich všeobecnými značkami, ktoré popisujeme v podsekcii 5.5.1 tiež vidíme priestor na zlepšenie, ktoré by mohlo zlepšiť úspešnosť klasifikovania. V našej implementácii totiž nahradzujeme, až na malú výnimku, iba entity zhodné so štruktúrovanými dátami. Myslíme si ale, že by mohlo pomôcť štruktúrované dáta parsovať a tak ich hľadať v texte. Napr. ak máme meno KUV: *Ing. Meno Priezvisko*, V našej implementácii hľadáme v texte dokumentu iba *Ing. Meno Priezvisko*, hoci v niektorých prípadoch by mohlo byť prínosné hľadať aj *Meno Priezvisko*. Prínosom by tiež bolo využiť hľadanie blízkych slov pre eliminovanie chýb OCR - aby sme v texte nahradili meno KUV/PVS napr. aj pri chýbajúcom jednom písmene.

Ako už bolo spomenuté vyššie, klasifikovanie viet je v našej implementácii pomerne nepresné a teda má priestor na zlepšenie. Ten vidíme najmä v úprave dát v datasete - jednotlivé riadky častokrát nepredstavujú celé vety. Ďalej, vety sú uložené tak, že rôzne kľúčové frázy sú zjednotené do ich skratiek - tak, ako to popisuje podsekcia 5.5.2. To ale pri klasifikovaní celého dokumentu zhoršovalo výsledky a preto si myslíme, že ak by boli vety uložené bez tohto spracovania, mohlo by to zlepšiť výsledky.

Trendom v NLP je využívanie predtrénovaných modelov a transformátorov. Tie prinášajú zlepšovanie výsledkov rôznych oblastiach NLP, nevynímajúc klasifikáciu dokumentov [45]. Preto by v ďalšej práci mohlo byť nápomocné využiť napr. doc2vec [40].

Záver

Cieľom našej práce bolo vytvoriť aplikáciu, ktorá z webovej stránky RPVS vytiahne takých PVS, ktorých zapísaní KUV označujeme ako štatutárov, nie skutočných majiteľov. Teda aplikáciu, ktorá dokáže prechádzať webovú stránku RPVS, previesť naskenovaný PDF dokument na text a nakoniec tento text zaradiť do jednej z dvoch kategórii.

Keďže je označovanie majiteľov a štatutárov zjednodušením, najprv sme špecifikovali toto rozdelenie a vysvetlili jeho účel. Keďže podstatnou časťou našej práce bola analýza dokumentov, vysvetlili sme tiež spôsob získavania týchto dokumentov. Ďalej sme vysvetlili metódy, ktoré boli pre dosiahnuie tohto cieľa nevyhnutné - najmä OCR a NLP.

Pre plynulé fungovanie aplikácie s čo najmenším zaangažovaním užívateľa naša aplikácia zahŕňa postupnosť krokov od prehľadávania na webe RPVS, cez OCR dokumentov až po klasifikáciu dokumentov a uloženie relevantných dokumentov tak, aby s nimi mohol užívateľ ďalej pracovať. V našej práci sme popísali celý tento postup aplikácie.

Dôležitou časťou našej práce je klasifikovanie verifikačných dokumentov. Preto sme v predstavili tri druhy klasifikátorov, ktoré sme implementovali. Ako prvý klasifikátor využívajúci regulárne výrazy, ktorý iba vyhľadáva frázy, resp. obrazce v texte. Tento klasifikátor bol vzhľadom na dosiahnuté výsledky pomerne časovo náročný na analýzu a hľadanie správnych obrazcov. Najlepší klasifikátor, ktorý sa nám podarilo implementovať a ktorý využívam v našej aplikácii, využíva viacvrstvový perceptrón a teda učenie s učiteľom. Tento klasifikátor dosahuje pri klasifikovaní štatutárov f1-skóre 0,89. Posledným implementovaným druhom klasifikátora bolo rozšírenie predchádzajúceho o klasifikovanie viet. Toto rozšírenie ale neprinieslo očakávané zlepšenie okrem iného pre nedostatočne presné klasifikovanie jednotlivých viet.

V práci ukazujeme aj možné vylepšenia našej implementácie. Okrem toho má ale práca aj možné rozšírenie, ktoré ďaleko presahuje presahuje rozsah našej práce. Tým rozšírením je rozoznávanie vzťahov jednotlivých fyzických a právnických osôb spomínaných v dokumentoch. Takéto rozšírenie by mohlo z RPVS získať ďaleko viac informácii, ktoré

by boli prínosné aj pre investigatívu. Okrem toho, s poznaním vzťahov medzi osobami vyskytujúcimi sa v dokumentoch vidíme tiež priestor na zlepšenie klasifikovania dokumentov.

Literatúra

[1] Slovenský národný korpus – prim-8.0-public-sane. bratislava: Jazykovedný ústav Ľ. Štúra sav 2018. Dostupné na `https://korpus.juls.savba.sk/`.

[2] Stop-words. Dostupné na `https://github.com/SlovakNationalGallery/elasticsearch-slovencina`.

[3] Tesseract documentation. [prístup 2021-01-21]. Dostupné na: `http://tesseract-ocr.github.io/`.

[4] Zákon č. 315/2016 z. z. o registri partnerov verejného sektora a o zmene a doplnení niektorých zákonov.

[5] Zákon č.297/2008 z. z. o ochrane pred legalizáciou príjmov z trestnej činnosti a o ochrane pred financovaním terorizmu a o zmene a doplnení niektorých zákonov.

[6] NLP4sk - natural language processing tools for slovak language [softvér], 2021. [prístup 2021-02-10]. Dostupné na `http://arl6.library.sk/nlp4sk/`.

[7] Pymupdf, 2021. Dostupné na `https://github.com/pymupdf/PyMuPDF`.

[8] urllib3, 2021. Dostupné na `https://github.com/urllib3/urllib3`.

[9] Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. DocBERT: BERT for Document Classification. *arXiv:1904.08398 [cs]*, August 2019. arXiv: 1904.08398.

[10] Bednárik. Slovenské NLP nástroje [softvér], 2019. [prístup 2021-02-12]. Dostupné na `http://nlp.bednarik.top/`.

[11] Ron Bekkerman and James Allan. Using bigrams in text categorization. page 10.

[12] Arindam Chaudhuri, Krupa Mandaviya, Pratixa Badelia, and Soumya K. Ghosh. Optical character recognition systems. pages 9–41. Publisher: Springer, Cham.

[13] Davide Chicco and Giuseppe Jurman. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21(1):6, 2020.

[14] Douglas Nunes de Oliveira and Luiz Henrique de Campos Merschmann. Joint evaluation of preprocessing tasks with classifiers for sentiment analysis in brazilian portuguese language.

[15] Jacob Eisenstein. *Introduction to Natural Language Processing.* MIT Press, 2019.

[16] J. R. Firth. A synopsis of linguistic theory 1930-55. 1952-59:1–32, 1957.

[17] Tsu-Jui Fu, Peng-Hsuan Li, and Wei-Yun Ma. GraphRel: Modeling text as relational graphs for joint entity and relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1409–1418. Association for Computational Linguistics, 2019.

[18] Ladislav Gallay and Marián Šimko. Utilizing vector models for automatic text lemmatization. In Rūsiņš Mārtiņš Freivalds, Gregor Engels, and Barbara Catania, editors, *SOFSEM 2016: Theory and Practice of Computer Science*, Lecture Notes in Computer Science, pages 532–543. Springer.

[19] Google. Announcing tesseract OCR - the official google code blog [online]. [prístup 2021-02-01]. Dostupné na: `http://googlecode.blogspot.com/2006/08/announcing-tesseract-ocr.html`.

[20] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[21] D. Hládek, J. Staš, and J. Juhár. Dagger: The slovak morphological classifier. In *Proceedings ELMAR-2012*, pages 195–198. ISSN: 1334-2630.

[22] A. K. Jain, R. P. W. Duin, and Jianchang Mao. Statistical pattern recognition: a review. 22(1):4–37. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[23] Daniel Jurafsky and James Martin. *Speech and Language Processing, 2nd Edition.* Prentice Hall, 2nd edition edition, 2008.

[24] Scharolta Katharina Sienčnik. Adapting word2vec to named entity recognition. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*, pages 239–243. Linköping University Electronic Press, Sweden.

[25] Michal Laclavík, Štefan Dlugolinský, and Michal Blanárik. Experimenting with slovak wikipedia as a source for language technologies. page 6.

[26] Ulf Leser and Jörg Hakenberg. What makes a gene name? Named entity recognition in the biomedical literature. *Briefings in Bioinformatics*, 6(4):357–369, 12 2005.

[27] Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, August 1989.

[28] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[29] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60. Association for Computational Linguistics.

[30] Andrei Mikheev, Marc Moens, and Claire Grover. Named entity recognition without gazetteers. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics* -, page 1. Association for Computational Linguistics.

[31] Aditya Mishra. Metrics to Evaluate your Machine Learning Algorithm, May 2020.

[32] Vijayarani Mohan. Preprocessing techniques for text mining - an overview. 2015.

[33] S. Mori, C. Y. Suen, and K. Yamamoto. Historical review of OCR research and development. 80(7):1029–1058. Conference Name: Proceedings of the IEEE.

[34] Farhan M. A. Nashwan, Mohsen A. A. Rashwan, Hassanin M. Al-Barhamtoshy, Sherif M. Abdou, and Abdullah M. Moussa. A holistic technique for an arabic OCR system. 4(1):6. Number: 1 Publisher: Multidisciplinary Digital Publishing Institute.

[35] William S Noble. What is a support vector machine? *Nature Biotechnology*, 24(12):1565–1567, 2006.

[36] Chirag Patel, Atul Patel, and Dharmendra Patel. Optical character recognition by open source OCR tool tesseract: A case study. 55:50–56.

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[38] Joaquín Pérez-Iglesias, José R. Pérez-Agüera, Víctor Fresno, and Yuval Z. Feinstein. Integrating the probabilistic models BM25/BM25f into lucene.

[39] Hassan Ramchoun, Mohammed Amine Janati Idrissi, Youssef Ghanou, and Mohamed Ettaouil. Multilayer Perceptron: Architecture Optimization and Training. *International Journal of Interactive Multimedia and Artificial Intelligence*, 4:6.

[40] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, 2010. ELRA.

[41] Leonard Richardson. Beautiful soup documentation. *April*, 2007.

[42] I Rish. An empirical study of the naive Bayes classifier. page 6.

[43] Yutaka Sasaki. The truth of the F-measure. page 5, 2007.

[44] Faisal Shafait and R. Smith. Table detection in heterogeneous documents. page 9.

[45] Zein Shaheen, Gerhard Wohlgenannt, and Erwin Filtz. Large Scale Legal Text Classification Using Transformer Models. *arXiv:2010.12871 [cs]*, October 2020. arXiv: 2010.12871.

[46] Transparency International Slovensko. Ako dobrou hodnotou za peniaze je protischránkový zákon?, 2017. [prístup 2021-05-13]. Dostupné na `https://transparency.sk/sk/ako-dobrou-hodnotou-za-peniaze-je-protischrankovy-zakon/`.

[47] R. Smith. An overview of the tesseract OCR engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 2, pages 629–633. ISSN: 2379-2140.

[48] S L Ting, W H Ip, and Albert H C Tsang. Is Naïve Bayes a Good Classifier for Document Classification? *International Journal of Software Engineering and Its Applications*, 5(3):10, 2011.

[49] Xiang Tong and David A. Evans. A statistical approach to automatic OCR error correction in context. In *Fourth Workshop on Very Large Corpora*.

[50] Jonathan J. Webster and Chunyu Kit. Tokenization as the initial phase in NLP. In *Proceedings of the 14th conference on Computational linguistics* -, volume 4, page 1106. Association for Computational Linguistics.

[51] Wen Zhang, Taketoshi Yoshida, and Xijin Tang. A comparative study of TF*IDF, LSI and multi-words for text classification. 38(3):2758–2765.

Príloha A: obsah elektronickej prílohy

Zdrojový kód je zverejnený aj na stránke `https://github.com/muriga/bachelor_thesis`.