VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

ACTIVE LEARNING PRO ZPRACOVÁNÍ ARCHIVNÍCH PRAMENŮ

ACTIVE LEARNING FOR WORK WITH ARCHIVE MATERIALS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. ALŽBETA ŠTAJEROVÁ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JAROSLAV ROZMAN, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání diplomové práce

147861

Ústav:	Ústav inteligentních systémů (UITS)
Studentka:	**Štajerová Alžbeta, Bc.**
Program:	Informační technologie a umělá inteligence
Specializace:	Strojové učení
Název:	**Active Learning pro zpracování archivních pramenů**
Kategorie:	Zpracování obrazu
Akademický rok:	2022/23

Zadání:

1. Nastudujte různé přístupy pro rozpoznávání ručně psaného textu a metodu Active Learning. Zaměřte se na použití pro rozpoznávání historických pramenů.
2. Navrhněte systém, který bude schopen používat různé implementace OCR, tzn. uživatel si bude moci vybrat buď mezi už existujícím OCR nebo vámi vlastnoručně vytvořeným OCR. Existující OCR volte s ohledem na možnost do nich doplňovat další softwarovou funkcionalitu. Navrhněte, jak implementovat doučování ze záznamů opravených uživateli nejen u převzatých OCR, ale i u vámi vytvořeného OCR.
3. Vytvořte kompletní navržený systém, implementujte vlastní OCR a dále vytvořte automatický systém doučování, který bude systém pravidelně doučovat podle nově opravených záznamů. Umožněte také full textové vyhledávání v rozpoznaných datech.
4. Systém otestujte a vyhodnoťte výpočetní náročnost pro enormní množství skenů a úspěšnost rozpoznávání.

Literatura:

- V. Romero, J. A. Sánchez a A. H. Toselli, "Active Learning in Handwritten Text Recognition using the Derivational Entropy," *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, Niagara Falls, NY, 2018, pp. 291-296
- D. Hříbek. Active Learning pro zpracování archivních pramenů. Brno, 2020. Semestrální projekt. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jaroslav Rozman, Ph.D.

Při obhajobě semestrální části projektu je požadováno:

- První dva body zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Rozman Jaroslav, Ing., Ph.D.**
Vedoucí ústavu:	Hanáček Petr, doc. Dr. Ing.
Datum zadání:	1.11.2022
Termín pro odevzdání:	17.5.2023
Datum schválení:	3.11.2022

Abstrakt

Cieľom tejto diplomovej práce je navrhnúť a implementovať systém na optické rozpoznávanie znakov v archívnych historických dokumentov. V prvej časti práce sa zaoberám štúdiom problematiky optického rozpoznávania znakov a procesom spracovania vstupných dát. Ďalej je opísaná téma aktívneho učenia a využívaných metód. Prehľadávam dostupné riešenia systémov pre rozpoznávanie ručne písaných historických dokumentov. Popisujem architektúry neurónových sietí využívaných na rozpoznávanie textu. Výsledkom práce návrh a následná implementácia systému pre rozpoznávanie textu, ktorý umožňuje anotáciu, doučovanie modelov a vyhľadávanie v záznamoch.

Abstract

The aim of this Master's thesis is to design and implement an OCR system for archival historical documents containing handwriting text. The first part of the thesis deals with the study of optical character recognition, the process of OCR pipepline. Then the topic of active learning and its methods is described. The thesis reviews the available solutions for recognition of handwritten historical documents. I further describe the neural network architectures used for text detection. The thesis results in the design and subsequent implementation of system for text recognition of historical documents, enabling user annotation, full-text search in annotation records.

Kľúčové slová

OCR, optické rozpoznávanie znakov, aktívne učenie, rozpoznávanie ručne písaného textu, HTR, strojové učenie

Keywords

OCR, optical character recognition, active learning, handwriting text recognition, HTR, machine learning

Citácia

ŠTAJEROVÁ, Alžbeta. *Active Learning pro zpracování archivních pramenů*. Brno, 2023. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jaroslav Rozman, Ph.D.

Active Learning pro zpracování archivních pramenů

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracovala samostatne pod vedením pána Ing. Jaroslava Rozmana, Ph.D. Uviedla som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpala.

.....
Alžbeta Štajerová
16. mája 2023

Poďakovanie

Ďakujem vedúcemu mojej diplomovej práce Ing. Jaroslavovi Rozmanovi, Ph.D. za konzultácie pri vypracovaní tejto práce.

Rada by som poďakovala svojím rodičom za ich podporu, nie len počas môjho celého štúdia.

Veľmi by som chcela poďakovať môjmu priateľovi Matúšovi za oporu a motiváciu vytrvať aj v náročných chvíľach.

Nechcem zabudnúť ani na mojích priateľov, ktorí pre mňa vždy mali útechu.

~~Dobre už bolo~~. Bude ako nebolo.

Obsah

Úvod

Žijeme v dobe rozmachu digitalizácie. Informácie sú ľahko prístupné cez internet. Avšak spracovanie ručne písaného textu je zložitý proces a to predovšetkým u historických dokumentov. Vo všeobecnosti platí, že čím starší dokument je, tým je horšie čitateľnejší. Dôvody sa líšia. V priebehu času sa jazyk a písmo vyvíja. Archívne dokumenty sú horšie prístupné pre bežné nahliadnutie. Preto sú mnohé archiválie spracované aj do elektronickej podoby. Objem takýchto dát nie je možné spracovať manuálne. Optické rozpoznávanie znakov (*optical character recognition*, OCR) a rozpoznávanie ručne písaného textu (*handwritten text recognition*, HTR) umožňuje automaticky prevádzať naskenové dokumenty na strojovo čitateľný digitálny text. Z dôvodu zložitejšieho a finančne náročnejšieho procesu anotovania sa rozširuje využívanie aktívneho učenia (*active learning*) k dosiahnutiu najoptimálnejších výsledkov a zníženiu nárokov na anotáciu.

Cieľom tejto práce je vytvorenie systému pre digitalizáciu historických archívnych dokumentov, umožňujúci detekciu a rozpoznanie písma v skenoch dokumentov pomocou rôznych OCR systémov a poskytujúci vyhľadávanie v záznamoch rozpoznaných prepisov. Užívatelia budú mať k dispozícii anotačné rozhranie pre úpravu a pridanie detekovaných oblastí písma a korekciu prepisov rozpoznaného textu. Systém bude moduly vykonávajúce OCR s opravenými anotáciami doučovať.

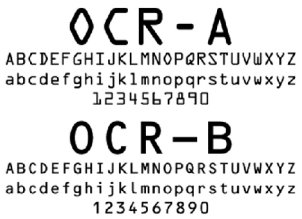
Táto práca je rozdelená do 6 kapitol. V kapitole 1 je popísaný proces optického rozpoznávania znakov, predstavené súčasné architektúry neurónových sietí využívane pre rozpoznavanie textu, analyzované dostupné systémy vykonávajúce OCR a predstavené dostupné dátové sady historických dokumentov obsahujúce ručne písane písmo. Kapitola 2 sa venuje štúdiu aktívneho učenia, variant a aplikácie aktívneho učenia pre úlohu OCR. Ďalej kapitola 3 obsahuje návrh modulového systému a opis jednotlivých častí pre správu, anotáciu a rozpoznávanie textu v historických archívnych dokumentoch. Kapitola 4 popisuje implementáciu systému, vyhodnotenie úspešnosti OCR a zhodnotenie výpočtovej náročnosti. V kapitole 5 sumarizujem výsledky a navrhujem námety na ďalší vývoj práce.

Kapitola 1

Optické rozpoznávanie znakov

V tejto kapitole sa opisuje proces optického rozpoznávania znakov, konkrétne oblasť umelej inteligencie, strojového učenia, ktorého úlohou je rozpoznať znaky. Vstupné dáta sú obrázky ručne písaného alebo tlačeného textu, ktoré sú získané vytvorením skenu dokumentu alebo fotografiou scény kde sa text nachádza. Výstupom je digitálna textová reprezentácia rozpoznaného textu zo vstupných dát. Ďalej v kapitole je uvedená klasifikácia OCR systémov. Následne kapitola zahŕňa rôzne typy architektúr využívané pre rozpoznávanie textu a analýzu dostupných systémov. Kapitola uvádza zhrnutie dátových sád využívaných pre úlohy rozpoznania ručne písaného textu.

Snaha o optické rozpoznanie znakov začala v polovici 20. storočia. V nasledujúcich desaťročiach sa OCR rozšírilo do širšej verejnosti. Obchody, banky, nemocnice a iné služby mali záujem o urýchlenie procesov a digitalizáciu dokumentov. Spočiatku pri rozpoznávaní nastávalo veľa chýb a proces bol zdĺhavý. Taktiež kvalita vytlačeného textu bola nižšia, čo bolo spôsobené rôznymi typmi fontov, kvalitou papiera, tlačiarňami a pod. Tieto dôvody viedli k vytvoreniu niekoľkých štandardov. Štandardizovaný bol papier, atrament a aj fonty OCRA OCRB (viď obrázok 1.1). To prispelo k zlepšeniu automatizácie v procese optického rozpoznania znakov textových dokumentov.



Obr. 1.1: Štandardizované fonty OCR-A a OCR-B [44]

Zo začiatku boli metódy založené na porovnávaní znakov s predlohami vzorových znakov (*template matching*). Tento spôsob bol nepoužiteľný pre využitie rozpoznávania ručne písaného textu. Nasledovali modely, ktoré boli schopné rozpoznávať ručne písané znaky z formulárov. Tie mali špecifikované políčka určené pre osamostatnené ručne písané znaky. Formuláre boli sprevádzané pravidlami, ako ich vyplňovať, napr. veľkým tlačeným písmom a pod.

Výskumu v oblasti optického rozpoznania znakov a ručne písaného textu venujú napríklad konferencie ICFHR[1] a ICDAR[2]. Súčasný stav optického rozpoznania znakov umožňuje rozpoznávať tlačený text s výbornými výsledkami jednou chybou na sto znakov. Oblasť sa zameriava na ručne písaný text, historické dokumenty a rozpoznávanie textu v rôznom prostredí (ŠPZ, nápisy na budovách), ktorých výsledky sú v porovnaní horšie.

1.1 OCR systémy

OCR systémy klasifikujeme rôznymi spôsobmi. Základný rozdiel je založený na type písma, ktoré sa má rozpoznávať - písané písmo rukou alebo vytlačený text tlačiarňou. Rozpoznávanie ručne písaného písma je zložitejšia úloha. Jazyky majú rôznu abecedu. Taktiež rukopis sa mení v závislosti od autora. Špecifické je zapisovanie jednotlivých znakov, sklon písma, dodržiavanie riadkov a pod. Štýl ručne písaného písma sa vyvíja s dobou. Dnešný každodenný rukopis je jednoduchší, ako rukopis vytvorený s atramentovým perom v minulých storočiach. Vytlačený text má v zásade jednotný font. OCR systémy môžeme rozdeliť aj na základe toho, či rozpoznávajú len jeden druh písma alebo umožňujú rozpoznanie znakov z rôznych fontov.

Ručne písaný text pomocou stylusu na obrazovku tabletu sa zaraduje do online rukopisu. V tomto prípade je zvyčajne dostupných viac informácii o ťahoch - rýchlosti, tlaku, pozícii. Typicky je dosahovaná väčšia miera úspešnosti pri rozpoznávaní a rozpoznávanie prebieha už v procese písania. Offline rukopis sa považuje rozpoznanie už napísaného textu na papieri a následne oskenovaného skenerom. Takéto pozadie a spôsob získania dát pridáva šum.

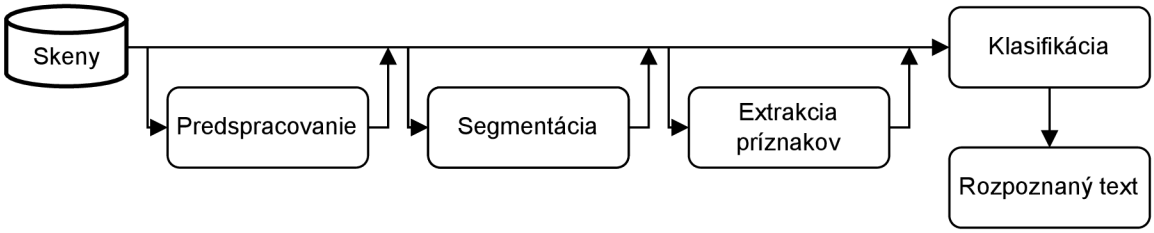
Väčšina state-of-the-art OCR HRT modelov sa spolieha na predspracovanie vstupného obrázka, kedy sa segmentujú relevantné textové oblasti na paragrafy, riadky, slová či znaky. Problémomje segmentácia rukopisu. Písaný text je častokrát nerovnomerný a nedodržiava rovný riadok. Súčasťou vstupných dát môžu byť tiež netextové oblasti obsahujúce artefakty vytvorené šumom, alebo iné netextové znaky. Spájanie segmentovaných oblastí môže priniesť chybu do systému hlavne u jazykov, ktoré sú písané sprava doľava. Formátovanie a odsadenie jednotlivých textových oblastí sa v niektorých prípadoch zanedbáva.

1.2 Proces rozpoznávania znakov

Optické rozpoznávanie textu pozostáva z niekoľkých krokov (viď obrázok 1.2):

1. zber vstupných dát - skeny dokumentov.
2. predspracovanie,
3. segmentácia,
4. extrakcia príznakov,
5. klasifikácia a rozpoznanie znakov,
6. vyhodnotenie úspešnosti optického rozpoznávania znakov,
7. následné spracovanie.

[1] https://icfhr2022.org/
[2] https://icdar2023.org/



Obr. 1.2: Proces optického rozpoznania znakov

Zber vstupných dát

Zber vstupných dát - skeny dokumentov (*image acquisition*) je prvý krok pri úlohe optického rozpoznávania znakov. Je to proces získavania dokumentov v ich digitálnej obrazovej forme. Dokumenty sú vyfotené kamerou, prípadne mobilom, alebo naskenované skenerom. Fotografie vytvorené mobilom sú odlišné v porovnaní s profesionálnym skenerom. Typicky je prítomné rozmazanie a nerovnomerné nasvietenie. V prípade historických archiválií je možné vytvoriť snímky pomocou multispektrálnej kamery s cieľom zlepšiť kvalitu vytvorených snímkov [12]. V tomto kroku nenastáva žiadna úprava dát. Snahou je vytvorenie konzistentnej sady jednotných vstupných dát. Dôležité je dodržať jednotný postup aj pri samotnom použití v praxy. Nesprávny postup pri získavaní dát môže poskytnúť nízku kvalitu vstupných dát. Je vhodné použiť jednotné zariadenie získavania fotografii, dodržať jednotný sklon, nasvietenie, vzdialenosť od dokumentu a pod.

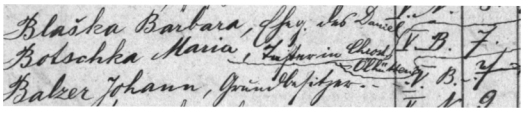
Predspracovanie

Predspracovanie (*preprocessing*) je ďalším krokom v spracovaní dát pre úlohu optického rozpoznávania znakov získaných v predošlom kroku zberu vstupných dát. Vytvorené dáta môžu obsahovať informácie, ktoré nie sú prínosné pri optickom rozpoznávaní znakov. Vo fáze predspracovania nastáva čistenie dát, čím sa zredukuje zložitosť a náročnosť ich ďalšieho spracovania v nasledujúcom procese. Cieľom úprav je zlepšenie čitateľnosti a odstránenie chýb vzniknutých v predošlom kroku získania dát - oprava natočenia textu, odstránenie rozmazania, nerovnomerného nasvietenia a šumu, normalizácia ťahov pri písaní rukou. Rozpoznanie textu s vhodne predspracovanými dátami môže fungovať lepšie, ale v prípade vytvorených kvalitných obrazových dokumentov môžu zbytočné úpravy viesť k zhoršeným výsledkom.

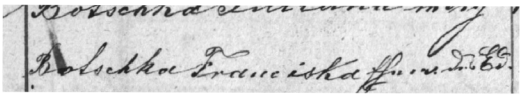
Historické dokumenty majú často problém so degradáciou kvality. Medzi hlavné problémy (viď obrázok 1.3) patrí prítomnosť šmúh, rôzne pozadie, nerovnomerné osvetlenie, škvrny spôsobené zlým archivovaním (poškodenie vlhkosťou), stopy od atramentu, rozpitie atramentu na ďalšiu stranu (*bleed-through*) [8].

Pri optickom rozpoznávaní znakov v predspracovanie pozostáva z nasledovných procesov:

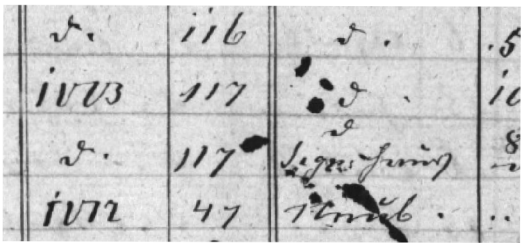
- **Korekcia sklonu písma a otočenie** (*slant and skew correction*) - v závislosti od kvality vstupných dát môžu byť dokumenty rôzne natočené. Taktiež pri rukopise nemusia byť nutne dodržiavané riadky. Konzistentnosť v jednotnom otočení, zlepšuje výsledky počas procesu rozpoznávania textu. Uhol otočenia dokumentu je odhadnutý medzi horizontálnym smerom stránky a smerom riadkov. Následne je dokument oto-



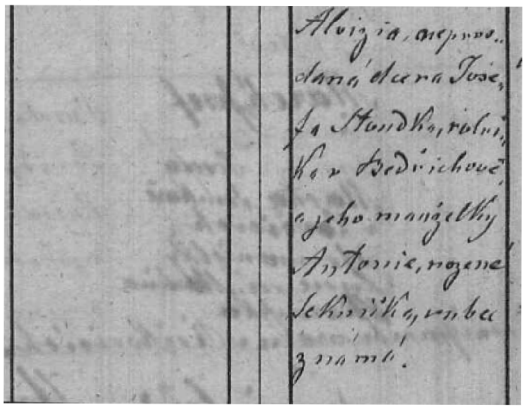
(a) Heterogénna výška znakov



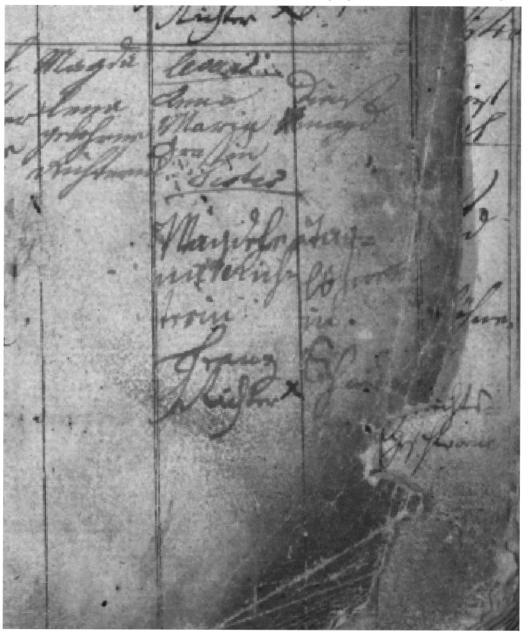
(b) Šmuha vzniknutá pri písaní



(c) Atramentové škvrny



(d) *Bleed-through* atramentu cez list papiera



(e) Fyzické poškodenie listu dokumentu

Obr. 1.3: Problémy historických dokumentov; Matričná kniha sig. M-16 7889, Matričná kniha ACTAPUBLICA č.k. 5005, Matričná kniha SOA Litomerice sig. 1/3

čený o tento uhol. Využívanými metódami sú najčastejšie Houghova transformácia[3] alebo analýza hlavných komponent (PCA).

- **Prahovanie** (*thresholding*, *binarization*) - snahou tohto kroku je oddeliť znaky od pozadia. Skeny sú najčastejšie vyhotovené farebne. Pre spracovanie OCR nie je potrebné uchovať všetky farebné kanály, pretože tieto informácie prispievajú ku komplexnosti a veľkosti vstupných dát. Obrázky sa prevedú do odtieňov sivej. Obrázok následne tvorí len jeden kanál, kedy pixely nadobúdajú hodnoty 0 až 255 (0 čierna, 255 biela). Predpokladom je, že odtiene šedej patriace znakom a pozadiu sa značne odlišujú [35]. Metódy môžme rozdeliť na lokálne a globálne. Globálne prístupy hľadajú jednu hodnotu úrovne, ktorá je vhodná pre aplikáciu prahovania celej stránky dokumentu. Tento postup sa využíva hlavne pri predspracovaní tlačených dokumentov z kancelárskeho prostredia. Lokálne metódy hľadajú vhodnú hodnotu pre každý pixel vstupného obrázka osobitne na základe susediacich pixelov. Táto metóda je komputačne náročnejšia, ale dosahuje lepšie výsledky pri nekonzistnom skene či poškodenej stránke. Typicky využívanými prahovacími metódamy sú Sauvola prahovanie [32] a Otsu metóda [26]. Parametre metód sú nastavované v závislosti od spracovávanej dátovej sady. Zvyčajne vylaďovanie závisí od experimentálneho zhodnotenia. Výsledkom prahovania je priradenie hodnoty 0 pixelom prislúchajúcim pozadiu na základe hodnôt pod zvoleným prahom. Naopak hodnoty 255 sú priradené pixelom obsahujúcim text. Hodnoty pixelov tak nadobúdajú len dve hodnoty. Efektom je redukovanie informácií pozadia a zvýraznenie textu.

- **Odstránenie šumu** (*noise removal*) - naskenované dokumenty prirodzene obsahujú určitú mieru šumu. V binárnych obrázkoch má formu osamostatnených zhlukov začiernených pixelov, ktoré nie sú súčasťou textu. Tieto pixely by mali mať bielu farbu, rovnako ako pozadie. Bežnou technikou odstraňovania šumu je median filter[43], ktorý sa aplikuje na obrázok v odtieňoch šedej.

- **Morfologické úpravy** pracujú s binárnymi vstupnými dátami. Pixely s hodnotou 255 vstupného obrázka predstavujú popredie (text). Pixely s hodnotou 0 predstavujú pozadie. Úpravy vedú k odstráneniu šumu a zjednodušeniu tvaru rukopisu. Základnými morfologickými transformáciami sú dilatácia, erózia a z nich sú odvodené otvorenie, uzavretie [6] a ďalšie. Operácia otvorenia, využíva kombináciu erózie nasledovanú dilatáciou. Umožňuje odstrániť osamostatnené zhluky šumu z popredia obrázka. Uzavretie je operácia pozostávajúca dilatáciou nasledovanou eróziou. Objekty (text) na vstupnom obrázku, ktoré sú blízko seba zacelí. Výsledkom sú objekty so zaplnenými dierami a vyhladenými obrysmi. Pomocou týchto techník dokážeme vstupné dáta zbaviť znečistenia v podobe šmúh, machúľ a obnoviť poškodené časti.

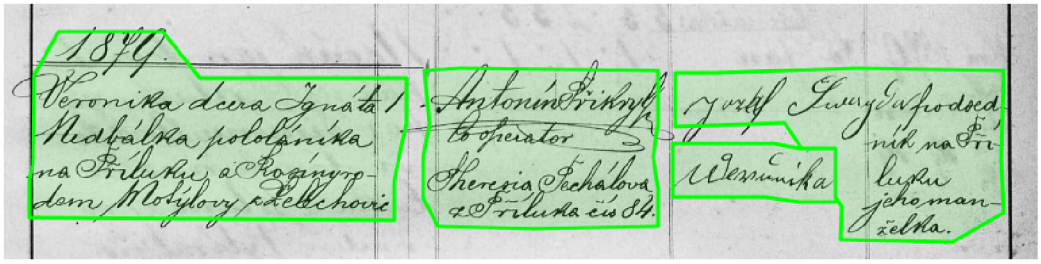
Segmentácia

Segmentácia (viď obrázok 1.4) je rozdelenie dokumentu na menšie časti, regióny, ktoré sú spracovávané v ďalších krokoch OCR. Cieľom je detekovať relevantné textové oblasti dokumentu a oddeliť tak netextové oblasti, ktoré nie sú zaujímavé pre proces rozpoznávania. Segmentácia prebieha na viacerých úrovniach. Prvou úrovňou je extrakcia textových oblastí. Obrázkové časti a šum sú ignorované. Výsledkom je pole súradníc určujúce pozíciu

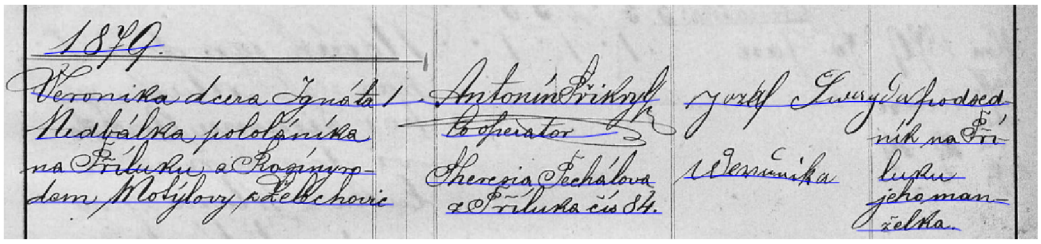
[3] https://sk.wikipedia.org/wiki/Houghova_transform%C3%A1cia

textových oblastí vrámci stránky. Následne textové odstavce sú dekompozitované na jednotlivé riadky. Pozícia riadkov sa uvádza pomocou linky (*baseline*), na ktorej riadok leží. Následne sa určuje výška riadku. Kombináciou informácii o linke a výšky riadku sa určí polygón v ktorom je riadok obsiahnutý. V závislosti od zvoleného prístupu OCR, segmentované riadky môžu byť ďalej rozdelené na slová a tie na osobitné znaky. Niektoré metódy OCR pristupujú k dokumentom s preskočením kroku segmentácie (*segmentation-free approach*). Avšak typicky rozpoznávanie prebieha na úrovni segmentovaných riadkov, slov či znakov.

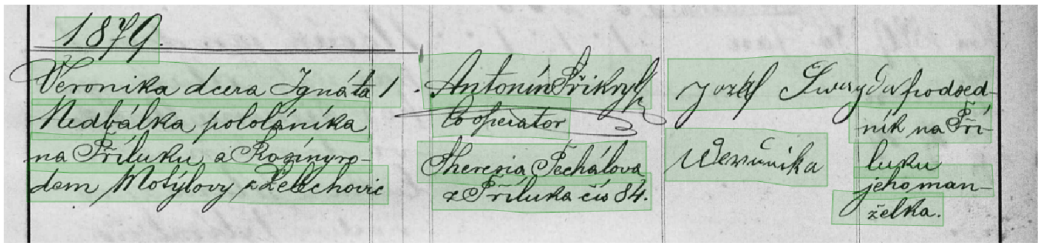
Pri segmentácii ručne písaného textu nastávajú problémy z dôvodu kurzívy a prelínajúcich sa znakov. Nutnosťou je krok predspracovania a úprava otočenia segmentovanej oblasti [7]. Pre segmentáciu je možné využiť prístup s využitím histogramu, stochastický prístup HMM[4] a iné.



(a) Rozpoznanie textových oblasti z obrázka (Pero OCR)



(b) Detekcia riadku (Transkribus)



(c) Výrez polygónov pre jednotlivé riadky (Pero OCR)

Obr. 1.4: Segmentácia dokumentu archívnej knihy ACTAPUBLICA č.k. 5005

Klasifikácia

Klasifikácia a rozpoznanie znakov (*clasification and recognition*) sa považuje za najzložitejší krok. Cieľom klasifikátoru obrazu textu je prevedenie obrazového vstupu na výstupnú textovú sekvenciu. Využiť môžeme klasifikátory založené na učení s učiteľom (*supervised learning*). Učenie prebieha pomocou dátovej sady, obsahujúcej anotované trénovacie dáta

[4]Skrytý Markovov model

vo forme vstupu a očakávaného výstupu (*target*, *ground truth*). Ďalšími klasifikátormi sú prístupy založené na učení bez učiteľa (*unsupervised learning*). Rozdiel v trénovacích dátach predstavujú neanotované vzorky dát. V prípade OCR klasifikované triedy predstavujú jednotlivé alfanumerické znaky vopred danej abecedy spolu so symbolmi, ktoré je OCR schopné rozpoznať.

Vyhodnotenie úspešnosti optického rozpoznávania znakov

- Pre objektívne vyhodnotenie úspešnosti sa používajú prevažne metriky *Character Error Rate* - *CER* [22] a *Word Error Rate* - WER [45], ktoré vychádzaju z Levenštejnovej vzdialenosti (*Levenshtein distance*) [19]. Táto metrika meria editačnú vzdialenosť pre textové sekvencie. Uvádza tri možné jednoznakové editačné úpravy - pridanie znaku (*insertions*), vypustenie znaku (*deletions*) alebo zámenu znaku za iný (*substitutions*). Najmenší počet takýchto operácií k prevedeniu jedného reťazca na druhý udáva Levenštejnovu vzdialenosť medzi týmito dvoma reťazcami. V závislosti od úpravy tak nastáva chyba nahrádzania, chyba vymazania a chyba vkladania. Kvocient medzi počtom chýb a počtom odhadov sa nazýva chybovosť (*error rate*).

(a) pridanie znaku (b) vypustenie znaku (c) zámena znaku za iný

Obr. 1.5: Úpravy Levenštejnovej vzdialenosti [18]

Miera chybovosti znakov (*Character error rate*) metrikou využívanou pri rozpoznávaní znakov zo vstupných dát. Indikuje chybovosť predikcií modelu na úrovni znakov, ktoré model nerozpoznal správne. Výpočet popisuje rovnica 1.1, kde S značí počet nahradení iným znakom, I značí počet operácii vloženia znaku , D je počet operácii vynechania znaku a N je počet všetkých znakov v referenčnom texte. Čím vyššia je hodnota CER, tým viac sa líši rozpoznaný text od skutočného textu. Perfektné skóre je nula. V prípade, že je počet chýb väčší ako dĺžka textu, sa v praxi udáva normalizovaná CER. Počet operácií je vydelený súčtom všetkých operácií a počtom správne rozpoznaných znakov. Výsledkom je hodnota v intervale 0-100%. Indikuje percentá znakov, ktoré OCR systém rozpoznal nesprávne. Hodnoty 1-2% normalizovaného CER dosahujú SOTA modely. Metrika CER avšak neumožňuje efektívne posúdiť, v akých konkrétnych prípadoch je OCR systém nedostatočný.

$$CER = \frac{(S + D + I)}{N} \tag{1.1}$$

Metrika miery chybovosti slov (WER) je vhodnejšia pre dlhšie nadväzujúce sekvencie. Indikuje počet slov textu, ktoré model nerozpoznal správne. Vychádza z predpokladu, že rozpoznaná sekvencia slov má inú dĺžku ako referenčný prepis textu. Levenštejnova vzdialenosť sa aplikuje na úrovni slov namiesto znakov. Výpočet popisuje rovnica 1.2, kde S_w udáva počet nahradení slova iným slovom, či chyba spôsobená zmenou písmen, D_w počet vynechaní slova, I_w počet vložených, nadbytočných slov a N_w celkový počet slov. Avšak rovnako ako CER neumožňuje bližšie detaily o pôvode chýb.

Pri výpočte metrík sa zámena veľkých a malých písmen typicky nepovažuje za chybu.

$$WER = \frac{(S_w + D_w + I_w)}{N_w} \tag{1.2}$$

V niektorých prípadoch sa využíva aj metrika Sentence error rate (SER). Táto určuje pomer chybovosti na úrovni sekvencií viet.

Následné spracovanie

Posledným krokom po rozpoznaní textu môže byť následná úprava (*postprocessing*) rozpoznaných slov na základe kontroly a detekcie chýb s cieľom zlepšiť presnosť výsledkov a zlepšiť čitateľnosť. Príkladom je úprava pravopisu alebo kontrola slov na základe dostupného slovníka (*lexicon driven correction*) [41]. Ak sa rozpoznané slovo nenachádza v slovníku je to považované za chybu a je nutná oprava. Ďalším prístupom je úprava na základe kontextu v akom sa slovo vyskytlo (*context-based correction*) [21], ktorý spája sémantickú aj gramatickú opravu. V prípade anglického jazyka, ak by OCR systém rozpoznal slovo "teh", kontext výskytu by mohol napovedať, že správne jedná o určitý člen "the".

1.3 Existujúce OCR architektúry

Optické rozpoznávanie znakov je oblasť strojového učenia. Rozpoznávanie ručne písaného textu je podoblasťou OCR. Vzhľadom na zložitosť úlohy rozpoznávania rukopisu sa výskum rozvíja samostatne oproti vývoju OCR.

V začiatkoch OCR sa využívali Hidden Markov Models (HMM) [1, 2]. Jedna z prvých architektúr využívajúca neurónové siete [17] rozpoznávala len jednoznakové čísla na dátovej sade MNIST. Využívala konvolučné neurónové siete a dramaticky zlepšila výsledky predikcie.

V oblasti strojového učenia dnes dominujú hlboké neurónové siete (*Deep neural network*, DNN). V minulosti bola ich nevýhoda vo výpočetnej náročnosti kvôli veľkosti architektúry z dôvodu počtu trénovaných parametrov. Ďalšou nevýhodou bola nutnosť fixnej dĺžky vstupného aj výstupného vektoru. Priame použitie konvolučných neurónový sietí tak neumožňuje spracovanie sekvenčných dát. Využitím rekurentných neurónových sietí [28] je možný prístup *sequence-to-sequence*, resp. seq2seq, kedy sa vstupná sekvencia mapuje na variabilnú výstupné sekvenciu. Prvé využitie rekurentných neurónových sietí bolo aplikované v architektúre modelu MDLSTM s využitím CTC chybovej funkcie [11] na rozpoznávanie jednoriadkového horizontálneho textu v arabštine.

Skrytý Markovov model

Skrytý Markovov model (Hidden Markov Model, HMM) vychádza z Markovov modelu. Obsahuje množinu stavov, z ktorých sa skladá, množiny symbolov, ktoré generuje, a množiny prechodov medzi stavmi modelu.

Neurónové siete

Neurónová sieť je algoritmus v strojovom učení, vytvorený na základe abstrakcie biologických neurónových buniek (neurónov) ľudského mozgu. Model je reprezentovaný výpočtovým grafom. Jeho jednotlivé uzly, perceptrony, predstavujú operátory. Neurónová sieť spracuje vstup podľa jej výpočetného grafu a vypočíta výstup, ktorý následne porovná so vstupným

vzorom. Porovnaným je získaná chyba, na základe gradientného zostupu sa aktualizujú váhy siete. Trénovaním neurónovej siete dochádza k vhodnej zmene váh a tak aj priblíženiu sa k vstupným vzorom. Najčastejšie využívaným prístupom je využitie algoritmu spätného šírenia chyby (*backpropagation*).

Trénovanie neurónových sieti je experimentálny proces založený na skúsenostiach vzhľadom na vykonávanú úlohu. Je potrebné zabezpečiť, aby funkcia, ktorú sieť aproximuje, obsahovala dostatok učiacich paramatrov (váh) a operátorov. Kľúčová je taktiež trénovacia dátová sada s dostatočným množstvom vzorov (dvojice vstup, vzor).

Konvolučné neurónové siete

Konvolučné neurónové siete (CNN) sú neurónové siete odvodené od matematickej lineárnej operácie konvolúcie, ktorá sa využíva v ich architektúre. Predpokladajú spracovanie dát, ktoré majú mriežkový vzor. Sú navrhnuté s cieľom extrakcie príznakov od najnižších úrovní až po vyššie úrovne. Skladá sa z:

- konvolučnej vrstvy,
- nelineárnej aktivačnej funkcie,
- združovacie vrstvy (*pooling layers*),
- plne prepojenej vrstvy.

Konvolučná vrstva a združovacia vrstva vykonávajú extrakciu príznakov. Konvolučný filter je postupne aplikovaný na celom vstupe. Posúva sa podľa určenej veľkosti a kroku. Typicky prvé vrstvy konvolučných neurónových sietí obsahujú jednoduchšie filtre reagujúce na hrany. Vyššie vrstvy získavajú zo vstupných dát detailnejšie informácie. Výsledkom je aktivačná mapa. Príkladmi operácií využívaných vrámci pooling vrstiev sú *max pooling*, nachádza lokálne maximum, a *average pooling*, nachádza hodnoty lokálnych priemerov. Aplikáciou pooling vrstvy sa zmenšuje dimenzia vstupu. Poslednou vrstvou je plne prepojená vrstva, ktorá mapuje príznaky vo forme vektorov dimenzie jedna na výstup v závislosti od danej úlohy.

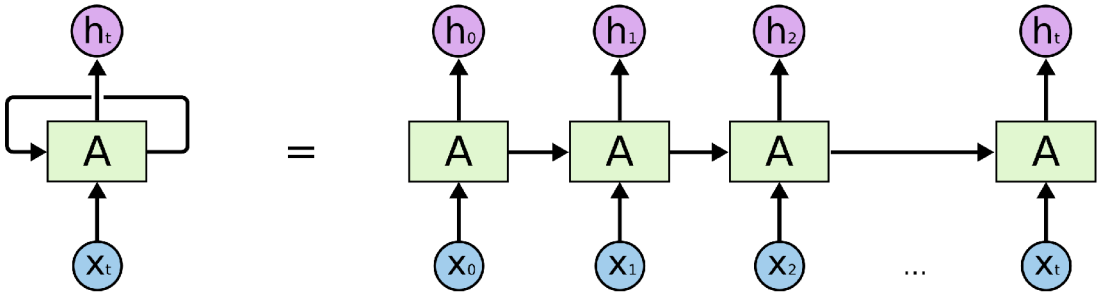
Rekurentné neurónové siete

Rekurentné neurónové siete (RNN) sú využívané predovšetkým v oblasti spracovania prirodzeného jazyka (*Natural Language Processing*, NLP) a pri spracovaní časových radov. Výhodou ich architektúry je spracovanie ľubovolnej dĺžky vstupu bez nutnosti zväčšovania veľkosti modelu a teda bez zvyšovania počtu trénovaných parametrov siete. Hlavnou myšlienkou rekurentných neurónových sietí je propagácia relevantnej informácie vzhľadom na kontext výskytu. Informácie o kontexte sú podmienené smerom spracovania. Smer zľava doprava využíva informáciu o predchádzajúcich vstupoch a ich kontexte, smer zprava doľava naopak.

Architektúra RNN (viď obrázok 1.6) pozostáva z udržiavania skrytého stavu (*hidden state*), ktorý reprezentuje pamäť predošlých vstupov. Ten je aktualizovaný v každom kroku výpočtu. Výstup každej rekurentnej bunky je na vstupe ďalšej rekurentnej bunky spolu so vstupom.

Variantami vylepšujúcimi základné *vanilla* RNN sú Long Short-Term memory (LSTM) [13] a Gated Recurrent Unit (GRU) [5]. Upravujú modul rekurentnej bunky s cieľom lepšieho naučenia sa dlhodobých súvislostí v dátach. Rekurentné neuronové siete často trpia

problémom miznúceho gradientu (*vanishing gradient*), kedy sa váhy siete stávajú veľmi malé. Tento problém vedie k pomalému a neefektívnemu trénovaniu. Opačný problém pri trénovaní je explodujúci gradientu (*exploding gradient*). Oba problémy kompromitujú proces trénovania.



Obr. 1.6: Rekurentná neurónová sieť; Výstup h_t je vypočítaný v rekurentnej buňke A na základe vstupu x_t a predchádzajúceho skrytého stavu h_{t-1} [25]

Transformery

Architektúra transformérov pozostáva z dvoch častí - enkodér, dekodér. Enkodér kóduje vstupnú sekvencie do vnútornej reprezentácie príznakových vektorov, ktorá je následne dekódovaná dekodérom na výstupnú sekvenciu. Trénovací proces je riadený dátami. Jedná sa o *end-to-end* prístup. Výhodou využitia transformerov namiesto samotných RNN je zvyčajne dosiahnutie lepšej presnosti a umožnenie spracovania aj dlhších reťazcov.

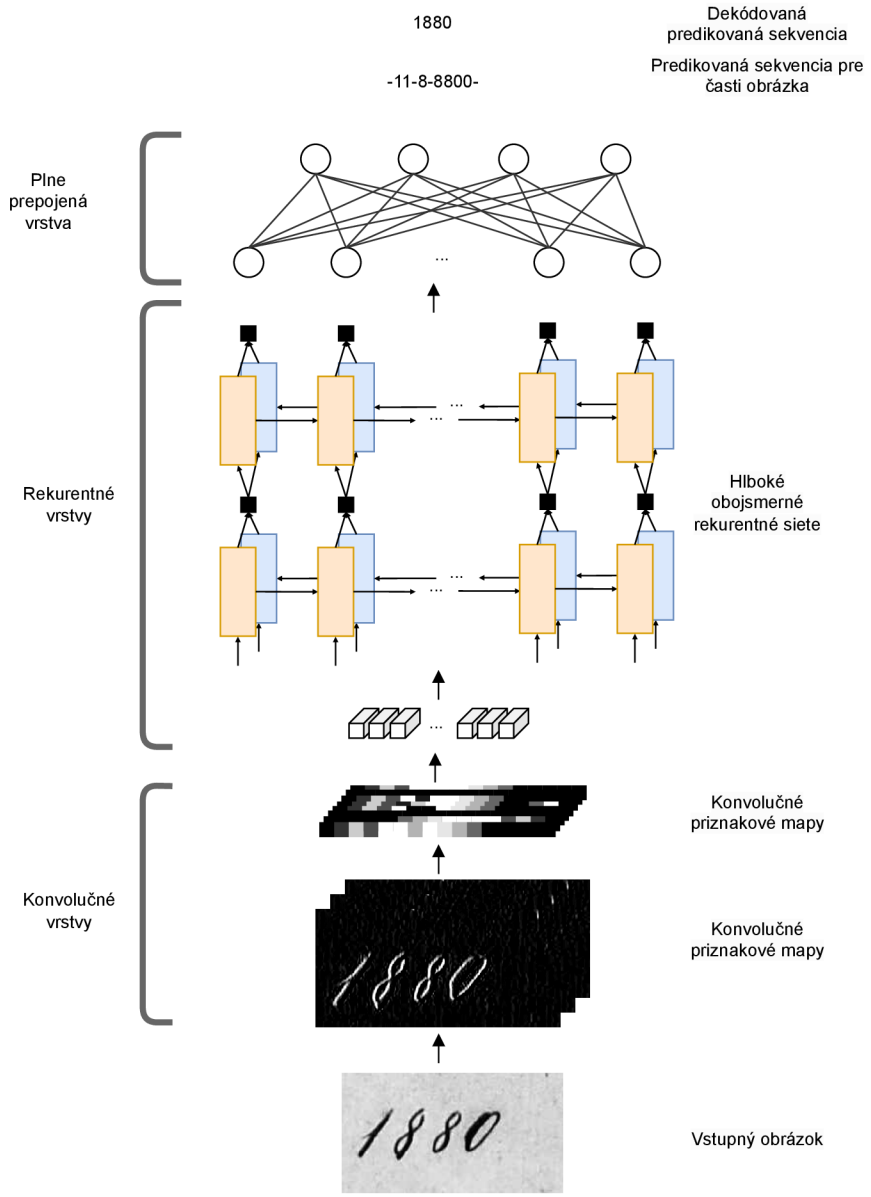
Transformery sú často dopĺňané mechanizmom pozornosti (*attention*) [42]. Model prikladá väčšiu váhu relevantným častiam vo vnútorných reprezentáciách vstupných dátach.

Konvolučné rekurentné neurónové siete

CRNN (*Convolutional Recurrent Neural Network*) [37] - Architektúra kombinuje jedny z dvoch najrozšírenejších neurónových sietí - konvolučné neurónové siete (*convolutional neural network*, CNN) nasledované rekurentnou neurónovou sieťou (*recurrent neural network*, RNN). Skladá sa z troch hlavných častí (viď obrázok 1.7):

- konvolučné vrstiev,
- rekurentné vrstiev,
- transkripčnej vrstvy.

Na vstupe modelu je obrázok segmentovaného textu (výsek slova, prípadne riadku) upravený na fixnú výšku. Ďalej je spracovaný konvolučnými vrstvami. Tie sú založené na *Very Deep Convolutional Networks* (VGG) [39]. Výstupom je konvolučná mapa príznakov. Nasledujúcou časťou modelu sú rekurentné vrstvy reprezentované zväčša obojsmernými rekurentnými neurónovými sieťami. Úlohou transkripčnej vrstvy je klasifikácia príznakov na výstupnú sekvenciu. Určí pravdepodobnostné rozloženie tried (znakov určenej abecedy) pre vektory príznakov. Časť pozostáva z plne prepojených vrstiev s aplikovanými nelineárnymi aktivačnými funkciami. Výstupom je vektor predikcií pre daný vstup s dimenziami o dĺžke predikovanej sekvencie a vstupnej abecedy.



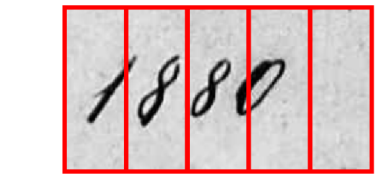
Obr. 1.7: Architektúra CRNN [37]

Chybová funkcia CTC

CTC (*Connectionist Temporal Classification*) [10] je objektívna funkcia aplikovaná pre klasifikáciu sekvenčných dát, kde je zarovnanie medzi sekvenciami náročné. Typicky vstupné dáta neobsahujú správne zarovnanie jednotlivých znakov na horizontálnej pozícii na obrázku. Ďalším problémom môže byť aj rozdielna dĺžka vstupných a výstupných sekvencií. Využitie CTC umožňuje trénovanie bez informácie o pozícii znakov na obrázku. Cieľom funkcie je maximalizovať ohodnotenie postupnosti výstupnej sekvencie pre vstupnú sekvenciu.

Pre trénovanie CRNN je využitá CTC stratová funkcia (*CTC loss function*). Na vstupe je matica dimenzie $T \times C$, kde T je počet buniek, na ktorých bol rozdelený vstupný výrez a C je počet znakov abecedy rozšírenej o prázdny znak (*blank*). Abeceda určuje množinu znakov, ktorú dokáže model rozpoznať. Na maticu je aplikovaná funkcia *softmax*. Každá časť výrezu obsahuje hodnoty v intervale $0 - 1$. Suma hodnôt pre každý výrez je rovná 1. Pre všetky bunky výrezu náleží jeden vektor hodnôt z matice. Jednotlivé hodnoty vektoru predstavujú pravdepodobnosť, že daná časť výrezu obsahuje jednotlivý znak abecedy. Pre hodnotu 1 vyplýva istota výskytu, pre hodnotu 0 naopak neistota.

Pri inferencii sa matica výstupu dekóduje. Najjednoduchším spôsobom dekódovania matice je zvolenie najpravdepodobnejšieho prepisu. Najlepšia cesta sa vyhodnocuje na základe najväčšej pravdepodobnosti znaku. Znaky s triedou *blank* a duplicitné znaky sú odstránené z výslednej sekvencie. Ak sa znak *blank* nachádza medzi dvoma zhodnými znakmi, tieto dva znaky sú dekódované ako dva rovnaké znaky vedľa seba. Ak nasledujú dva identické znaky za sebou, sú dekódované len ako jeden náležiaci znak.



Vstupný obrázok rozdelený na časti

0	0.01	0.04	0.05	0.16	0.01
1	0.40	0.00	0.00	0.00	0.00
...	...	...	...	...	...
7	0.29	0.00	0.02	0.02	0.00
8	0.00	0.35	0.24	0.1	0.00
blank	0.02	0.01	0.16	0.1	0.3

Pravdepodobnostná matica CTC

"1880 "

Obr. 1.8: CTC dekódovanie sekvencie

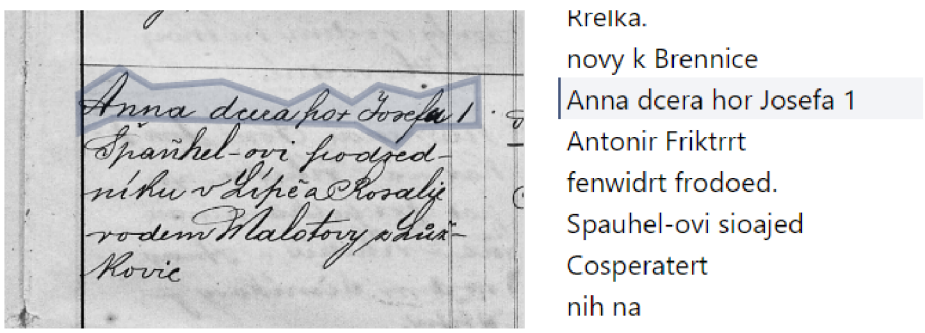
1.4 Existujúce nástroje riešenia OCR

Najrozšírenejším nástrojom využívaným na riešenie OCR je **TesseractOCR**[5]. Tento nástroj dokáže rozpoznať 116 jazykov. Zameriava sa hlavne na rozpoznávanie tlačeného písma. Umožňuje dotrénovanie modelu. V začiatkoch bol vyvíjaný firmou Hewlett-Packard, neskôr firmou Google [40]. Tá jeho vývoj ukončila. Dnes je projekt Tesseract OCR vedený univerzitou Mannheim. Tesseract OCR využíva architektúru s LSTM rekurentnými neurónovými sieťami, ktoré zlepšili presnosť rozpoznávaného textu. Hlavnou výhodou tohto nástroja je jeho voľné využitie a podpora množstva jazykov.

Menej robustnými nástrojmi sú **paddleOCR**[6], **EasyOCR**[7]. Využitie PaddleOCR je možné aj na mobilných zariadeniach.

Nástroj **Kraken**[8], vychádzajúci z knižnice **OCRopus**[9], priamo ponúka modely určené na spracovanie a rozpoznávanie ručne písaných historických dokumentov.

Komerčne poskytovanou platformou na rozpoznávanie textu je **Transkribus**[10] [23]. Platforma vznikla pod projektom READ (*Recognition and Enrichment of Archival Documents*). Zameriava sa predovšetkým na historické dokumenty s tlačeným ale aj ručne písaným textom. Umožňuje trénovanie vlastných inštancií modelov a vyhľadávanie v dokumentoch s podporou 30 jazykov. Poskytované verejné modely sú vytvorené pomocou open-source nástroja PyLaia[11]. Na obrázku 1.9 je ukážka detekcie a rozpoznania textu v ručne písanom dokumente pomocou platformy Transkribus.



Obr. 1.9: Platforma Transkribus; Ukážka detekcie a rozpoznania textu kroniky archívu MZA, č. knihy 5005

Nástroj **Cloud Vision** API od Google ponúka REST API OCR systém. Dosahuje vysokú presnosť, avšak nemá voľnú licenciu. Každý dotaz je platený a preto je jeho využite pre enormné množstvo archívnych dokumentov nevhodné.

1.5 Dátové sady

V nasledujúcej sekcii sú popísané využívané dátové sady využívané pri úlohe optického rozpoznávania znakov. Uvádzané dátové sady sú anotované ručne písané dokumenty. Nie-

[5] https://github.com/tesseract-ocr/tesseract
[6] https://github.com/PaddlePaddle/PaddleOCR
[7] https://github.com/JaidedAI/EasyOCR
[8] https://kraken.re/
[9] https://github.com/ocropus/ocropy
[10] https://readcoop.eu/transkribus/
[11] https://github.com/jpuigcerver/PyLaia

ktoré predstavujú historické dokumenty, ktoré sú vhodné hlavne pri využití v konkrétnom riešení archívnych dokumentov. Kvalitných dátových sád je v porovnaní s inými problematikami strojového učenia pomerne málo. Platí, že čím starší dokument, tým menej ľudí je schopných ho čítať a tak aj anotovať. Transkript je typicky uložený v textových súboroch s príponou `txt`, ktoré obsahujú len text alebo uložený vo formáte XML, ktorý obsahuje text a aj segmentovanú oblasť, v ktorej sa prepis nachádza.

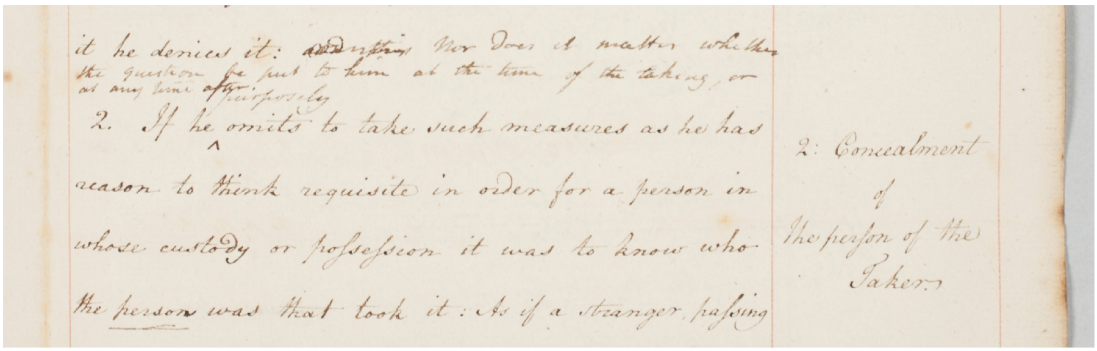
Dátové sady typicky obsahujú obrázkové súbory a textové súbory. Každý obrázok má anotované polygony segmentovaných oblastí spolu s prislúchajúcim prepisom (*ground truth*).

Dátová sada Bentham R0

Bentham R0 [31] - dátová sada vytvorená projektom Transcribe Bentham. Kolekcia obrázkov je vytvorená z diel filozofa Jeremy Bentham (1748-1832) pojednávajúc o práve a morálnej filozofii v anglickom jazyku. Tvorí ju 443 fotografií stránok, ktoré dokopy obsahujú 11473 anotovaných regiónov textových oblastí riadkov spolu s ich prepisom textu. Obsahuje rozdelenie na trénovaciu, validáciu a testovaciu sadu. Tabuľka 1.1 sumarizuje informácie o dátovej sade.

Počet	Trénovacia sada	Validačná sada	Testovacia sada	Celkom
Strany	350	50	33	433
Riadky	9198	1415	860	11473
Znaky	86	86	86	86

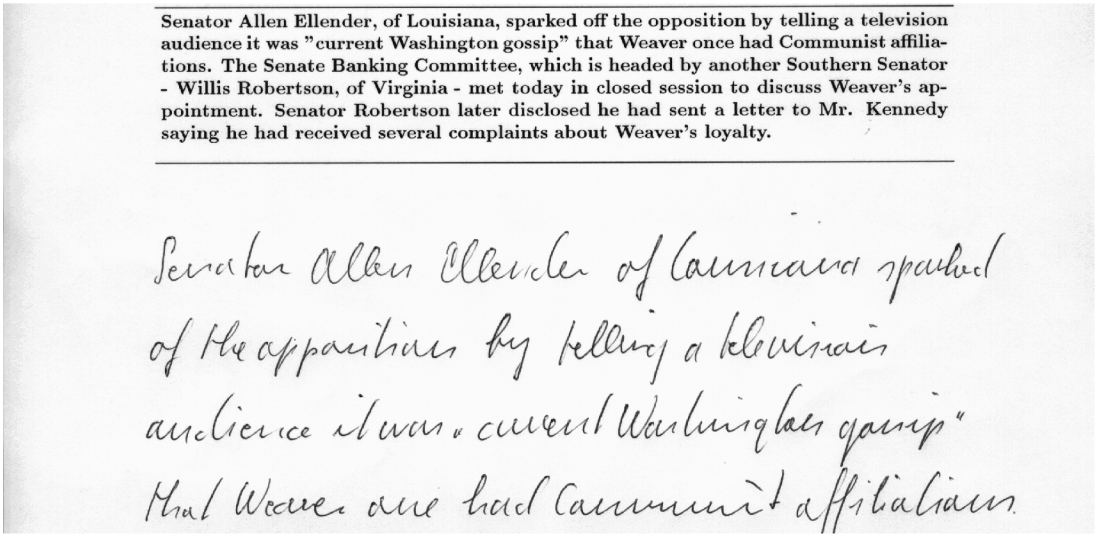
Tabuľka 1.1: Prehľad rozdelenia dát dátovej sady Bentham R0



Obr. 1.10: Ukážka dátovej sady Bentham R0

Dátová sada IAM

IAM [20] - korpus obsahuje 1539 oskenovaných stránok ručne písaného textu od 657 rôznych pisateľov v anglickom jazyku. K dispozícii je 5685 viet na 13353 segmentovaných riadkoch. Súčasťou dát je aj prepis textu. Jazykom textu je angličtina. Dátová sada neobsahuje odporúčané rozdelenie na trénovaciu a testovaciu dátovú sadu.



Obr. 1.11: Ukážka dátovej sady IAM

Dátová sada Digital Peter

Digital Peter [27] je dátová sada tvorená 9694 obrázkami vytvorených oskenovaním rukopisu (1709-1713) Peter the Great v ruštine. Oblasť riadkov spolu s jednotlivými prepismi je k dispozícii spolu s rozdelením dátovej sady na trénovaciu, testovaciu a validačnú.

Dátová sada Saint gall

Saint gall [9] obsahuje 60 strán, 1410 textových riadkov. Text je písaný v latinčine a datuje sa do 9. storočia. Rukopis je vytvorený len jediným pisateľom.

Dátová sada HKR

HKR [24] je databáza ručne písaného kazašského a ruského jazyka. Väčšina prepisov je v ruskom jazyku. Dáta majú formu ručne vypísaných 1400 formulárov.

Dátová sada Parzival

Parzival Database je zbierka dokumentov z 13. storočia. Obsahuje 47 ručne písaných stránok od 3 rôznych pisateľov. Jazykom rukopisu je stredoveká nemčina.

Dátová sada HJDataset

HJDataset [36] - dátová sada tvorená japonskými historickými dokumentami obsahujúca 2100 oskenovaných stránok a 250000 anotovaných segmentovaných textových oblastí.

1.6 Výpočtová náročnosť modelov strojového učenia

Posudzovanie výpočtovej náročnosti je dôležitým aspektom pri trénovaní modelov strojového učenia. Vo všeobecnosti posúdenie výpočtovej náročnosti vyžaduje komplexné zohľadnenie viacerých kritérií. Množstvo potrebných výpočtov závisí od zložitosti modelu, veľkosti

spracovávaných údajov a hardvérových možností prostredia použitého na trénovanie a inferenciu. Uvedené faktory ovplyvňujú informované rozhodnutia o optimalizácii modelov a využitie hardvéru na dosiahnutie optimálneho výkonu.

Výpočtová náročnosť trénovania (viď obrázok 1.12a) a inferencie modelu (viď obrázok 1.12b) je čiastočne rozdielna. Inferencia nastáva po až po natrénovaní modelu. V oboch prípadoch je nutné, aby bol model načítaný v pamäti zariadenia. Rovnako v oboch prípadoch nastáva dopredný prechod výpočetným grafom modelu. Ďalej sa pri trénovaní porovná výstup modelu s očakávaným vzorom. Chyba medzi predpovedaným a skutočným výstupom sa spätne šíri modelom (*Backpropagation*) s použitím reťazového pravidla na výpočet gradientov.

Pri trénovaní modelu je bežnou metrikou na posúdenie počet trénovateľných parametrov v modely. Čím viac parametrov model má, tým viac výpočtov je nutné vykonávať pri jeho trénovaní. Vrstvy predstavujúce aktivačné funkcie, Dropout, Batch normalizácia a združovacie vsrtvy nezväčšujú model, pretože nemajú trénovateľné parametre. Rôznorodosť architektúr a algoritmov učenia poskytuje rôzne kompromisy medzi výpočtovou zložitosťou, množstvom dát potrebných na natrénovanie modelu a výkonom. Architektúra konvolučnej neurónovej siete na klasifikáciu obrázkov typicky obsahuje viac parametrov ako jednoduchší model logistickej regresie. Časová zložitosť logistickej regresie počas trénovania je úmerná počtu vstupných príznakov m a počtu trénovaných vzoriek n - $O(n*m)$. Priestorová zložitosť logistickej regresie je $O(m)$. Časová zložitosť logistickej regresie počas predikcie je $O(m)$ [16]. Trénovanie a aj inferencia CNN je náročnejšia. Populárny jazykový model GPT-3, ktorý je súčasťou nástroja ChatGPT[12], má 175 miliard parametrov. Na jeho uloženie je potrebných 800GB úložiska [3].

Ďalším údajom pri určovaní náročnosti zohráva veľkosť údajov. Väčšie dáta vyžadujú väčšie pamäťové nároky, viac výpočtov. Čas trénovania sa tak priamo úmerne zvyšuje.

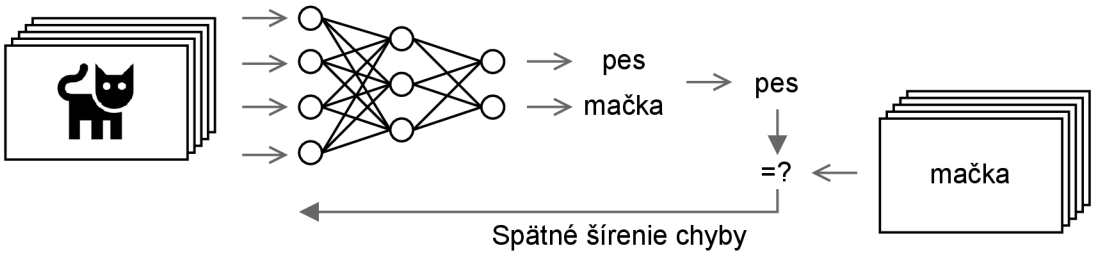
Trénovací čas jednej epochy resp. celkový čas potrebný na natrénovanie modelu slúži ako ďalší indikátor náročnosti. Dlhší čas naznačuje väčšiu výpočtovú náročnosť. Na natrénovanie modelu je typicky nutný opakovaný prechod trénovacou dátovou sadou. Počet epoch určuje počet prechodov. Kratší čas môže naznačovať, že model je menej zložitý, alebo že využívaný hardvér je výkonný.

Rozdelenie dátovej sady na dávky vzoriek ovplyvňuje počet iterácii na jednu epochu. Veľkosť dávky určuje koľko vzoriek model spracúva naraz. Počet vzoriek použitých v každej dávke počas trénovania ovplyvňuje trénovací čas. Menšie veľkosti dávok znižujú požiadavky na pamäť hardvéru, ale môžu prispieť k predĺženiu trénovania. Zatiaľ čo väčšie veľkosti dávok môžu urýchliť trénovanie, ale vyžadujú viac pamäte.

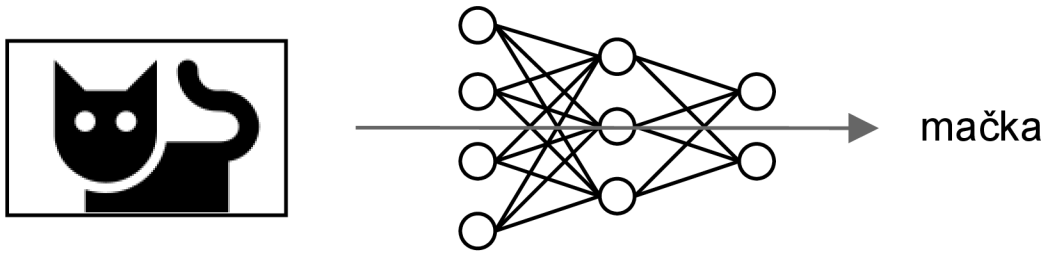
Metrika FLOPs (*Floating-Point Operations Per Second*) meria počet matematických operácii s pohyblivou rádovou čiarkou ako sčítanie a násobenie, ktorú výpočtový stroj môže vykonať za jednu sekundu. Používa sa na porovnanie výkonu rôznych platforiem. Počet nutných FLOP pre model počas trénovania je určený počtom vrstiev v modely, veľkosťou každej vrstvy, počtom a veľkosťou tréningových vzoriek.

Typ hardvéru využitého na trénovanie ovplyvňuje rýchlosť vykonávania výpočtov. Špecializovaný hardvér, ako jednotky TPU (*Tensor Processing Units*) môžu ďalej akcelerovať trénovanie deep modelov. Architektúra TPU je navrhnutá na efektívne vykonávanie operácii násobenia matíc a ďalších operácii lineárnej algebry a trénovanie je oveľa rýchlejšie ako tradičné CPU alebo GPU.

[12] https://chat.openai.com/



(a) Trénovanie; Na začiatku je nenatrénovaný model, ktorý sa iteratívne učí z poskytovaných dát. Vstupné dáta sú spracované dopredným prechodom cez model. Výstupom modelu je predikcia, ktorá sa porovná s očakávaným výsledkom použitím stratovej funkcie. Chyba medzi predpovedaným a skutočným výstupom sa spätne šíri model, pričom sa upravujú váhy modelu tak, aby sa chyba minimalizovala. Výsledkom procesu trénovania je optimalizovaný model na danej úlohe.



(b) Inferencia; Optimalizovaný model na úlohu vykonáva naučenú schopnosť predikcie na nových dátach dopredným prechodom modelu.

Obr. 1.12: Proces trénovania a inferencie

Inferencia modelu je menej výpočtovo náročná a rýchlejšia hlavne z dôvodu, že nedochádza k žiadnemu spätnému šíreniu chyby, čo sa vyžaduje počas trénovania. Váhy modelu pri predikovaní na vzorkách sú fixné (zamrazené). Pri trénovaní sa môžu využiť augmentačné techniky na vstupných dátach. Náhodné orezávanie, otáčanie a pridávanie šumu zvyšuje náročnosť v kroku predspracovania dát. Tieto úkony sa nevykonávajú na testovacej dátovej sade. Súhrne povedané, pri inferencii je výpočtová náročnosť určená zložitosťou modelu a veľkosťou vstupných údajov.

Kapitola 2

Aktívne učenie

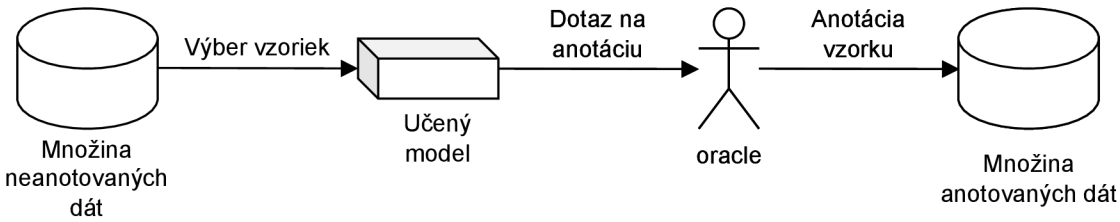
Typický proces učenia algoritmov strojového učenia s učiteľom je pasívny, bez zásahov. Učený algoritmus dostáva na vstupe postupne náhodný výber vzoriek. V situáciach, kedy je dostupné veľké množstvo neanotovaných dát a anotácia zdĺhavý proces je využitie aktívneho učenia (*active learning*, *query learning*, *optimal experimental design*) výhodné. Jedná sa o formu interaktívneho učenia s učiteľom. Snahou je anotácia dát optimálnym spôsobom a využitie ľudského úsilia na maximum. Užívateľovi sú ponúkané vzorky, ktoré má anotovať. Priorizuje sa predkladanie dát, ktoré majú najväčší vplyv na trénovanie modelu. Užívateľ, človek, ktorý anotuje poskytnuté dáta sa nazýva učiteľ alebo *oracle*. Dáta sú ponúkané jednotlivo alebo po dávkach (*batches*). Cieľom takéhoto procesu je, aby učený algoritmus dosiahol rovnaké až lepšie výsledky s menším úsilím, pripadne kratšou dobou učenia. Skrátenie fázy získavania anotácií a učenia šetrí zdroje a aj čas [29, 33].

Táto podoblasť strojového učenia umelej inteligencie je využívaná v mnohých podoblastiach spracovania jazyka (rozpoznávania reči), extrakcie informácií a ďalších klasifikačných úlohách.

2.1 Varianty aktívneho učenia

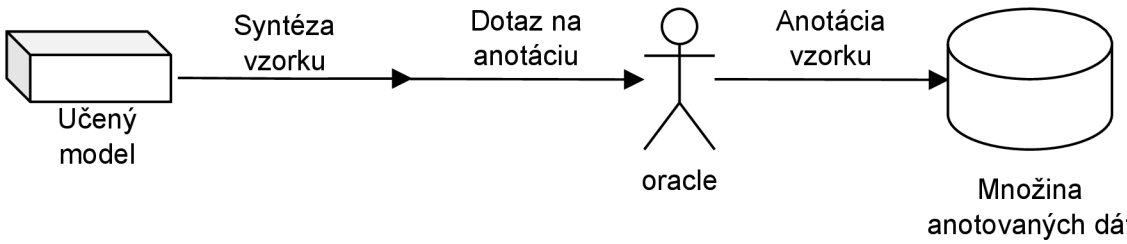
Pri aktívnom učení nastávajú 3 scenáre:

Pool-Based sampling (viď obrázok 2.1) je najrozšírenejším prípadom aktívneho učenia. Vstupná množina dát $\mathcal{U}$ obsahuje veľké množstvo neanotovaných vzoriek a malé množstvo anotovaných dát. Model je natrénovaný s touto množinou anotovaných dát $\mathcal{L}$. Celá množina neanotovaných dát je ohodnotená na základe stratégie. Užívateľovi tak môžu byť predkladané vzory, kde si je učiaci sa model najmenej istý (greedy prístup). Tieto dáta majú najväčšiu mieru informatívnosti pre učený model.



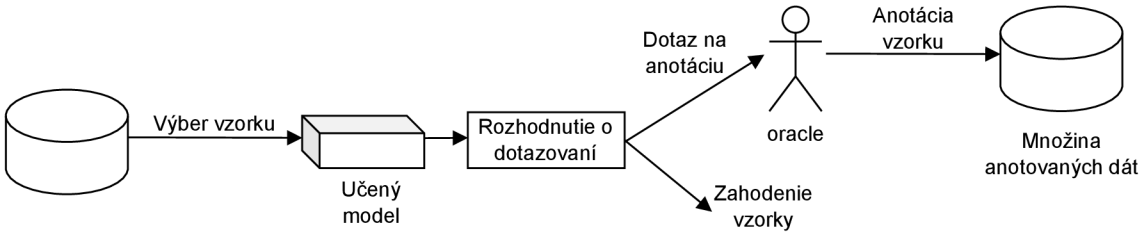
Obr. 2.1: Pool-Based sampling

Syntéza dotazov o príslušnosti (*Membership Query Synthesis*) je variantou aktívneho učenia, kedy model vygeneruje (syntetizuje) vzorku z danej distribúcie vstupnej množiny dát (viď obrázok 2.2). Model sa dotazuje učiteľovi o akú príslušnosť triedy vzorku sa jedná. Príkladom môže byť snímka číslice, ktorá bola získaná vyrezaním časti obrázka. Úlohou užívateľa je určiť aká to je číslica. Tento scenár nastáva pri úlohách regresie. Pri vygenerovaných nových dátach modelom, nie človekom, môže nastať tvorenie nezmyslov. Napríklad ak by boli vytvárané obrázky, ktoré neobsahujú číslicu, ale len umelé hybridné znaky bez sémantického významu. Pre anotátora sú takéto vzorky problematicky rozpoznateľné.



Obr. 2.2: Membership Query Synthesis

Poslednou rozlišovanou variantou aktívneho učenia je selektívny výber vzoriek z toku dát (*Stream-Based Selective Sampling*, *sequential active learning*). Predpokladá jednoduché získavanie nových neanotovaných dát. Model vygeneruje nový dátový vzorok a ohodnotí ho (viď obrázok 2.3). Ak by bola anotácia tohto vzorku prínosná pre učenie modelu, nasleduje výzva užívateľa k anotácii. Inak určí príslušnosť triedy sám model alebo sa vzorok zahodí. Rozhodnutie o zahodení určuje zvolená stratégia. Rozdiel medzi selektívnym výberom dát od pool-based vzorkovania je to, že sa pracuje len s jednou vzorkou v danom čase, ktorá sa ohodnotí. Tento scenár je vhodnejší využiť v situáciach, kedy výpočetné zdroje alebo pamäť na ohodnotenie celej dátovej sady je obmedzená (napr. pri mobilných zariadeniach).



Obr. 2.3: Stream-Based Selective Sampling

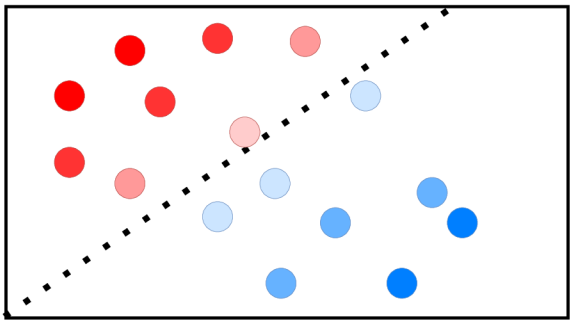
2.2 Stratégie skórovania pri aktívnom učení

Všetky stratégie zahrňujú hodnotenie informovanosti funkciu $\phi(x)$ neoznačených dát x. Čím vyššiu hodnotu nadobúda, tým je vstup vhodnejší pre výber na anotáciu. Existuje mnoho navrhovaných spôsobov stratégie:

- **Neistota** (*Least Confidence - LC, Uncerain Sampling*) - Jedná sa o najjednoduchšiu stratégiu. Stratégiu je možné využiť pre modely, ktoré sú schopné ohodnotiť istotu rozhodnutia výstupu. Dátam z neanotovanej množiny model odhadne náležiace triedy.

Pravdepodobnosť, o akú triedu sa jedná, určuje skóre pre túto stratégiu. Dáta sú zoradené od najmenšieho skóre po najväčšie. Anotátorovi sú prioritne predkladané dáta, kde si je model najmenej istý.

Jednoduchou aplikáciou stratégie je model binárnej klasifikácie. Logistický model rozhoduje o lineárnom priestore oddelujúcom dve triedy, typicky priamka, či n-rozmerná plocha. Výstupom klasifikátoru s aplikovanou aktivačnou funkciou Sigmoid je číslo v rozmedzí 0-1. V prípade, že výstup $F(x)$ pre vzor x je menší ako 0.5 sa jedná o jednu triedu. Pre hodnoty väčšie a rovné ako 0.5 sa vstup x považuje za druhú triedu. Na základe stratégie neistoty vyberáme také vzorky z neanotovaných vstupných dát, ktoré sú najbližšie od rozhodovacej hranice. Model je si najmenej istý pre hodnoty blížiace sa k 0.5. Pre zložitejšie modely stratégia funguje obdobne.



Obr. 2.4: Priestor vstupných dát dvoch tried (červená a modrá) s deliacou hranicou určenou binárnym klasifikátorom; Priehľadnosť určuje neistotu rozhodnutia triedy podľa modelu

$$\phi(x)^{LC} = 1 - P(y = c* | x) \tag{2.1}$$

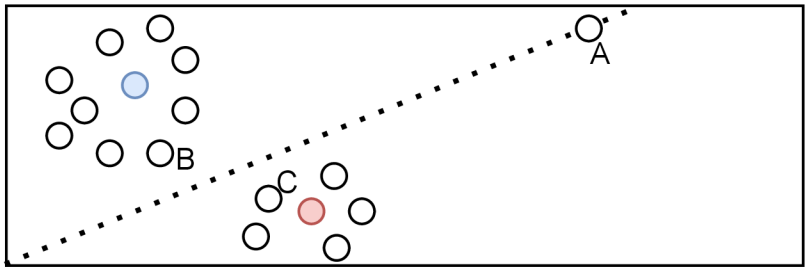
- **Nezhoda** (*Query-By-Committee*) [34] - Stratégia pozostáva zo zahrnutia komisie do výberu dát. Komisia je tvorená z viacerých natrénovaných modelov z dostupnej trénovacej anotovanej dátovej sady. Každý člen komisie vyberá vhodného kandidáta nezávisle - najinformovanejší vzorok. Dáta, na ktorých sa komisia zhodne najmenej, sú zvolené na anotáciu. Veľkosť typickej komisie začína na 2 až 3 členoch. Typy modelov využité v komisii nemusia byť rovnaké. Ak je komisia tvorená jedným typom modelu tak sa členovia líšia inými konfigurovateľnými hypermarametrami. Najčastejšie využívanými metódami sú Support Vector Machines (SVM), logistická regresia, náhodné lesy a Naive Bayes klasifikátor. Takýto spôsob trénovania sa nazýva ansámblové učenie (*ensemble learning*) využívajúce metódy *boosting* a *bagging*. Ďalším spôsobom učenia komisie je trénovanie jednotlivých členov na rozdielnej podmnožine trénovacích dát.

- **Odhad zmeny modelu** (*Expected Model Change*, *Expected Gradient Length*) Pridanie nového vzorku do trénovacej množiny a následné trénovanie modelu ovplyvňuje zmenu modelu. Miera tejto zmeny očakáva zlepšenie výkonnosti modulu. Preto je nežiadúci výber dát k anotácii, ktoré neovplyvnia učený model. Prístup vyberá takú vzorku z trénovacej množiny dát, ktorá by priniesla najväčšiu zmenu do aktuálneho modelu, ak by sme poznali jej označenie (*label*). Ako názov označuje, stratégia rozhodovania výberu dát je využívaná pri modeloch optimalizovaných na základe gradientu. Vybraná vzorka by mala viesť k najväčšiemu gradientu. Tento prístup skórovania je

výpočetne zložitejší. Ohodnotenie dátovej sady sa vždy vzťahuje ku aktuálnemu stavu modelu.

$$\phi^{EGL}(\mathbf{x}) = \sum_{\hat{\mathbf{y}} \in \mathcal{N}} P(\hat{\mathbf{y}} \mid \mathbf{x}; \theta) \|\nabla \ell(\mathcal{L} \cup \langle \mathbf{x}, \hat{\mathbf{y}} \rangle; \theta)\| \tag{2.2}$$

- **Odhad zníženia chyby** (*Expected Error Reduction*) - Odhad zníženia chyby je ešte výpočetne náročnejší ako predošle zmienený odhad zmeny modelu. Zameriava sa na meranie toho, o koľko sa pravdepodobne zníži jeho chyba generalizácie.
- **Znižovanie odchýlky** Vzhľadom na nákladnosť odhadu zmeny modelu a odhadu zníženia chyby môžeme chybu zovšeobecňovania znížiť nepriamo minimalizáciou výstupného rozptylu. Vhodným využitím sú prípady kedy sú dáta zašumené a rozptyl odhadu je vysoký.
- **Prehľadávanie priestoru vstupu** Oproti predchádzajúcim prístupom sa metóda prehľadávania priestoru vstupu zameriava na celý vstupný priestor, a nie len na jednotlivé prípady. V prípade využitia neistoty, vyberáme vzorky blízko rozhodovacej hranice. Prístup odhadu zmeny modelu môže vyberať podobné vzorky a dopytovať sa tak na anotáciou odľahlých hodnôt, pretože sa očakáva významná zmena v modely. Avšak nevyberáme nutne vzorky, ktoré sú reprezentatívne pre ostatné dáta distribúcie. Znalosť distribúcie vstupných dát môže viesť k presnosti odhadov údajov ako celku (viď obrázok 2.5). Podobným prístupom je explicitné modelovanie rozdelenia vstupov počas výberu.



Obr. 2.5: Binárny klasifikátor s rozhodovacou hranicou; Zafarbené kruhy sú anotované vzorky a biele kruhy sú neanotované vzorky; Pri ohodnotení na základe neistoty by bol zvolený vzorok A kvôli pozícii na rozhodovacej hranici. Avšak výber vzoriek B alebo C by viedlo k viac informáciám o distribúcii dát.

- **Pseudo-anotácie.** Pseudoanotácia je praktika používaná na generovanie ďalších anotovaných dát z neanotovanej dátovej sady. Model produkované predikcie považuje ako za zaručené anotované dáta, ktoré sa používajú ďalej v iteratívnom dotrénovaní modelu. Tento prístup môže byť účinný pri nedostatku anotovaných dát. Kvalita pseudoanotácií závisí od aktuálne trénovaného modelu.
- **Cena anotácie.** Pri každom vytváraní dátovej sady je potrebné zohľadniť zdroje na jej tvorbu. Tieto náklady na cenu anotácie môžu zahŕňať čas, vynaložené úsilie, ktoré sa líši v závislosti od zložitosti úlohy, veľkosti dát a požiadavky odbornej znalosti na spoľahlivú anotáciu. Napríklad, anotácia horšie čitateľného riadku bude pravdepodobne náročnejšia aj pre model aj z pohľadu človeka.

2.3 Aplikácia aktívneho učenia vrámci HTR

Snaha využiť aktívne učenie prebieha aj vrámci rozpoznávania ručne písaného textu práve z dôvodu obtiažnejšieho získavania prepisov dokumentov. Konkrétnym prípadom využitia je preferencia anotácie riadkov, ktoré sú najmenej podobné riadkom využitých počas trénovacieho procesu modelu rozpoznávajúceho text. Predikcia riadkov veľmi podobná tým, ktoré model už videl, bude prebiehať lepšie ako na neznámych riadkoch.

Navrhujú sa rôzne prístupy ako redukovať množstvo potrebných trénovacích dát alebo dosiahnuť lepšie výsledky s dostupnými anotovanými dátami pomocou aktívneho učenia [30], transfer learning a aj kombináciou oboch prístupov [4]. Predošlé snahy využívali HMM miesto rozšírenejších, modernejších hlbokých neurónových sietí. Ich výsledky naznačili zníženie anotačného úsilia len v menšej miere. Výber vzoriek podľa neurčitosti prebieha pomocou Shannonovej entropie. Ďalším navrhnutým spôsobom je prístup meriania informatívnosti na základe TF-IDF.

Kapitola 3

Návrh riešenia

V kapitole návrhu riešenia systému OCR je stanovený cieľ vypracovania riešenia systému pre rozpoznávanie textu z obrazových súborov archívnych matričných kníh. Následne je na základe tohto cieľa navrhnutá architektúra modulového systému. Každý z týchto modulov architektúry je jednotlivo popísaný.

3.1 Požiadavky riešenia

Cieľom tejto práce je navrhnúť, implementovať a otestovať systém rozpoznávajúci ručne písaný text v historických dokumentoch s podporou aktívneho učenia. Systém bude podporovať rôzne prístupy OCR. Pre úlohu rozpoznávania znakov bude možné zvoliť existujúci nástroj OCR alebo vlastný OCR model. Aplikácia bude poskytovať užívateľom rozhranie pre anotáciu dát - korekciu rozpoznaných textov, úpravu segmentovaných polygónov. Dostupné OCR riešenia bude systém doučovať užívateľmi anotovanými dátami. Užívateľ si bude môcť v rozpoznaných textových záznamoch vyhľadávať.

Vrámci fulltextového vyhľadávania bude umožnené užívateľovi nastaviť filtračné parametre - archív, potvrdený alebo rozpoznaný záznam. Nájdené záznamy by mali obsahovať znenie textu a presmerovanie na konkrétny sken daného dokumentu spolu s vyznačením relevantného segmentu.

Systém bude mať prístup k lokálnemu úložisku dokumentov, ktoré obsahujú naskenované archívne dáta. Aplikácia bude mať možnosť prezerať si obrázky jednotlivých dokumentov. Systém bude umožňovať vykonávať úlohy dotrénovania s novými získanými anotovanými záznamami. Užívatelia sa v systéme budú môcť zaregistrovať a získať tak prístup do systému.

3.2 Analýza predchádzajúceho riešenia

Predchádzajúce riešenie [14] bolo vypracované vo forme webovej aplikácie. Obsahovalo anotačnú komponentu, pomocou ktorej užívatelia upravovali segmentované oblasti dokumentov a rozpoznaný prepis. Užívateľom umožňovalo vkladanie vlastných dátových sád. Poskytovala rozpoznanie textu pomocou implementovaného modelu CRNN v rámci Pytorch a nástroja Tesseract OCR.

Práca bola implementovaná s využitím viacero technológii (Python, PHP, Javascript, a iné). Z tohto dôvodu by bolo nutné zvýšené úsilie na rozšírenie systému.

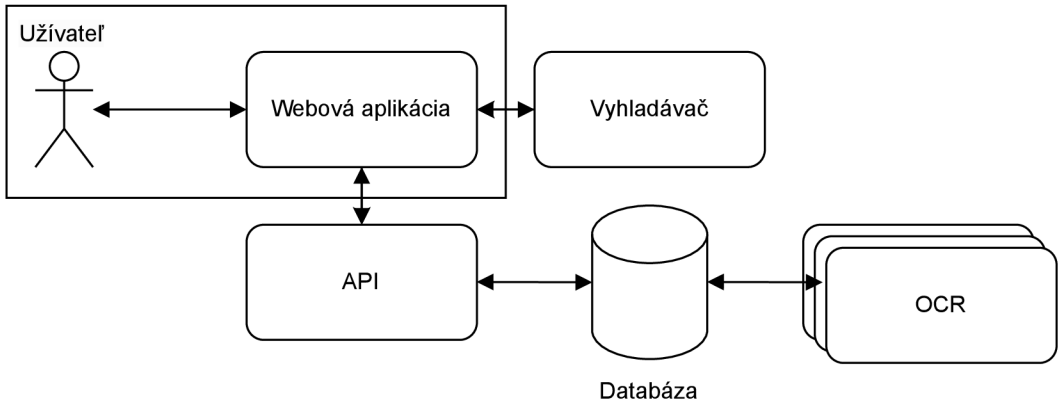
Riešenie je ťažko prenositeľné aj napriek využitiu virtualizačnej platformy Docker, pretože pri každom nasadení sa systém musí zostaviť znovu. Taktiež v riešení často chýba použitá verzia závislostí, kvôli tomu dochádza ku chybe pri preklade z dôvodu nekompatibility verzií jednotlivých balíčkov. S odstupom času je pátranie po využívaných verziách komplikovanejšie.

Predchádzajúce riešenie bolo nasadené a zverejnené. Využilo ho niekoľko genealógov, ktorí vložili do systému dátové sady obsahujúce skeny dokumentov archívnych kníh. Vzniklo tak niekoľko tisíc anotácii. Avšak vzhľadom na enormné množstvo archívnych kníh je ich spätné dohľadávanie obtiažnejšie a kategorizácia týchto anotácii je obtiažnejšia. Užívatelia taktiež nedodržiavali konsenzus, že jeden dataset obsahoval obrázky z jednej archívnej knihy. Z tohto dôvodu vypracované riešenie uvažuje nahraté archívne knihy, ktoré sú pripravené na anotáciu. Systém neumožňoval export dát a anotácie boli uložené len v databázi. Pre tento účel vznikol skript `prev_solution_convert/db_data_parse.py`, ktorý potvrdené anotácie daného datasetu uloží do súbora JSON so súradnicami polygónu textovej oblasti, textovým prepisom a cestou k obrázku, ktorému náleží.

3.3 Návrh architektúry systému

Architektúra systému je delená do samostatných modulov (viď obrázok 3.1) s dôrazom na návrh architektúry orientovanej na služby (*Service Oriented Architecture*, SOA). Výhodami modulového návrhu sú:

- izolovanosť,
- prenositeľnosť,
- škálovateľnosť,
- nahraditeľnosť.



Obr. 3.1: Moduly systému

Webová aplikácia

Rozhranie pre správu systému, anotáciu dát bude poskytovať webová aplikácia.

API

Jednotlivé moduly medzi sebou komunikujú prostredníctvom správ zasielaných vo formáte JSON[1]. Každý modul poskytuje verejné aplikačné rozhranie (API).

Vyhladávač

Užívateľovi poskytuje možnosť fulltextového vyhľadávania v rozpoznaných textoch. Podporuje vyhľadávanie s drobnými chybami (preklepy). Na vstupe je očakávaný hľadaný textový výraz. Výstupom hľadania sú výsledky textových prepisov obsahujúce zhodu s hľadaným výrazom.

OCR

Modulov OCR bude v systéme dostupných niekoľko. Každý druh OCR je osamostatnený modul.

Anotátor

Komponenta anotácie umožňuje užívateľovi v uživateľskom rozhraní anotovať dáta v obrázkovom aj textovom formáte. Užívateľ zadá prepis, či označí oblasť paragrafu, riadku alebo slova.

Návrh spôsobu segmentácie dokumentov

Segmentácia stránok dokumentov prebieha pomocou nástroja PERO OCR[2] [15]. V rámci rozšírenia riešenia je nahradiť tento nástroj iným, ktorý by umožňoval doučovanie na základe opravených regiónov. Spomínaný nástroj doučovanie neumožňuje. V niektorých prípadoch segmentácie stránok archívnych dokumentov dochádza ku chybám členenia - ukončenie riadku pred jeho koncom, odrezané vyššie písmo a pod.

Návrh doučovania modelov

Systém bude pravidelne vykonávať doučovanie OCR modulov pomocou potvrdených anotovaných dát od užívateľov. Modely OCR po dokončení procesu dotrénovania následne vykonajú inferenciu na dátach bez potvrdených transkripcií.

[1] https://www.rfc-editor.org/rfc/rfc8259
[2] https://github.com/DCGM/pero-ocr

Kapitola 4

Implementácia

Táto kapitola pojednáva o použitých technológiách a nástrojoch využitých pre implementáciu systému.

4.1 Použité technológie

Systém je členený do modulov, kde každý modul má samostatné prostredie vytvorené kontajnerom Docker. API Komunikácia prebieha pomocou mikroframeworku Flask[1] v jazyku Python. Vlastný CRNN model bol implementovaný pomocou knižnice Pytorch Lightning. Webová aplikácia používa open-source JavaScript framework Vue.Js[2] pre vytváranie užívateľských rozhraní.

Pytorch Lightning

Pytorch Ligtning[3] je open-source knižnica pre strojové učenie, vychádzajúca z PyTorch, nad ktorým vytvára Wrapper. Umožňuje vyššiu flexibilitu experimentácie a škálovateľnosť. Implementácia procesov je pomocou Pytorch Ligtning abstraktnejšia ako pomocou samotnej knižnice Pytorch. Kladie si dôraz na cieľ ľahšej čitateľnosti pre využitie na výskumné a aj komerčné použitie strojového učenia. Udržiava kód štrukturovaný a modulárny aj s rastúcou zložitosťou projektu. Uľahčuje prácu nad distribuovanými GPU a TPU výpočtami.

Docker

Docker[4] je platforma využívajúca virtualizáciu na úrovni operačného systému, poskytujúca jednotné rozhranie a izoláciu v ľubovolnom prostredí Windows, Linux či macOC. Slúži k automatizácii nasadzovania aplikácií pomocou kontajnerov. Aplikácie sú tak jednoduchšie prenositeľné. Docker Compose je nástroj na špecifikáciu a spúšťanie aplikácií Docker tvorených viacerými službami (*services*). Súbor `docker-compose.yml` definuje využívané služby, siete a adresáre (*volumes*). Každá služba má samostatnú konfiguráciu. Vymedzené sú premenné prostredia, používané porty a pod. Následne je celý systém aplikácie ovládateľný pomocou príkazu `docker compose`. Využitie Docker Compose zjednodušuje proces

[1] https://flask.palletsprojects.com/en/2.2.x/
[2] https://vuejs.org/
[3] https://www.pytorchlightning.ai/
[4] https://www.docker.com/

spúšťania aplikácií Docker s viacerými kontajnermi a uľahčuje správu a škálovanie zdrojov aplikácie podľa potreby.

Flask

Pre vývoj aplikačného rozhrania (API) bola zvolená knižnica Flask v jazyku Python. Je to odľahčený mikro rámec využívaný pre svoju jednoduchosť. Spolu s SQLAlchemy umožňuje objektovo relačné mapovanie (ORM) databázových dotazov.

Vue.JS

Aplikáciu tvoria znovupoužiteľné časti používateľského rozhrania (komponenty). Šablóna komponentu je tvorená časťou definujúcu štruktúru HTML, súborom údajov a definíciou správania skriptom a voliteľnou časťou štýlu pravidiel CSS, upravujúcou vizuálny vzhľad. Komponenty možno používať modulárne a organizovať.

Základné komponenty - tlačidlá, formuláre a pod, boli využité z knižnice komponentov Vuetify[5], ktorá obsahuje sadu vopred navrhnutých komponentov v štýle Material Design[6], ktoré je možné integrovať do Vue.Js projektov.

MLflow

MLflow[7] je open-source platforma na správu životného cyklu modelov strojového učenia s cieľom sledovať, reprodukovať, zdieľať modely a výsledky experimentov. MLflow sa skladá z niekoľkých komponentov, ktoré spolupracujú a poskytujú komplexné riešenie. MLflow Tracking sa používa na zaznamenávanie a vyhľadávanie experimentov. Umožňuje zaznamenávať parametre, metriky a artefakty (súbory) spojené s experimentmi, vyhľadávať a porovnávať rôzne behy. MLflow možno používať so širokou škálou knižníc a rámcov strojového učenia vrátane TensorFlow, PyTorch, Scikit-Learn a XGBoost. Podporuje rôzne programovacie jazyky vrátane Pythonu, R a Javy.

Elasticsearch

Elasticsearch[8] je populárny open-source fulltextový vyhľadávač využívaný na indexovanie, vyhľadávanie a analýzu veľkých objemov údajov. Pre využívanie Elasticsearch je nutné vytvoriť index a mapovanie dát do šablóny pre údaje vyhľadávania. Mapovanie opisuje štruktúru vyhľadávaných dát. Pre prácu s Elasticsearch je možné využiť oficiálne webové rozhranie Kibana[9] aj knižnice v jazykoch Python, .NET, Java a ďalšie. Vyhľadávanie pomocou Elasticsearchu je škálovateľné vzhľadom na dostupné možnosti zdrojov. Elasticsearch sa automaticky stará o balancovanie indexu dát, ktoré spravuje. Výhodou využitia je rýchle odbavovanie vyhľadávacích dotazov aj nad obrovskými objemy dát. Nevýhodou sú zvýšené nároky na hardware - nutnosť priestoru na disku, kvôli indexácii.

[5] https://vuetifyjs.com/en/
[6] https://m3.material.io/
[7] https://mlflow.org/
[8] https://www.elastic.co/
[9] https://www.elastic.co/kibana/

Nginx

Webový server Nginx vytvára reverzný proxy server. Spracováva požiadavky od klientských zariadení zasielané na server webovej lokality. Následne reverzný proxy server preposiela požiadavky na ďalšie serveri (alebo mikroslužby Docker v projekte) podľa požadovaného URI a prijíma od nich odpovede. Zabezpečuje nepriamu komunikáciu. Vytvorením takéhoto prepojenia umožňuje nasadenie systému, ktoré poskytuje klientom jednotné rozhranie prostredníctvom jediného portu.

4.2 Spracovanie archívnych kníh

České archívne matičné dokumenty v digitalizovanej podobe spravujú nasledovné archívy:

- Archiv hlavního města Prahy[10]
- Moravský zemský archiv v Brně[11]
- Státní oblastní archiv v Hradci Králové[12]
- Státní oblastní archiv v Litoměřicích[13]
- Státní oblastní archiv v Plzni[14]
- Státní oblastní archiv v Praze[15]
- Státní oblastní archiv v Třeboni[16]
- Zemský archiv v Opavě[17]

Jednotlivé matričné knihy majú identifikáciu na základe signatúry, prípade inventárneho čísla. Obsahujú informácie o narodeniach, úmrtiach, svadbách a farnosti alebo úrade, ktorý ich vytváral. Neobsahujú prepisy textu, ale v posledných rokoch sa postupne knihy sprístupňujú na internete v podobe vytvorených obrázkových dokumentov týchto kníh. Každá z uvedených matrík spravuje tisíce archívnych kníh. V začiatkoch digitalizácie bola digitalizácia dokumentov vykonaná pomocou mikrofilmu. Horšie mikrofilmy sa postupne naskenuvávaju.

Pre spracovanie archívnych kníh do internej reprezentácie bola vytvorená abstraktná trieda `AbstractArchive` umiestnená v súbore `abstract_archive.py`. Zdrojové kódy súvisiace so spracovaním archívov, archívnych kníh a skenov kníh sú uložené v adresári `backend/modules/seed`. Adresár obsahuje štyri triedy dedené z predošle spomenutej abstraktnej triedy špecifikujúce spracovanie archívov, ktoré mi boli poskytnuté na indexáciu do systému (viď tabuľka 4.1):

- Moravský zemský archiv v Brně:

[10] http://katalog.ahmp.cz/pragapublica/permalink?xid=7EF18906B65E11DF820F00166F1163D4
[11] http://actapublica.eu
[12] https://aron.vychodoceskearchivy.cz
[13] http://matriky.soalitomerice.cz
[14] http://www.portafontium.eu
[15] https://ebadatelna.soapraha.cz
[16] http://digi.ceskearchivy.cz
[17] http://digi.archives.cz

Archív	Počet kníh	Počet skenov
Moravský zemský archiv v Brně	11958	1914416
Státní oblastní archiv v Litoměřicích	13704	2113550
Zemský archiv v Opavě	13419	1902973
Státní oblastní archiv v Hradci Králové	10954	1856157

Tabuľka 4.1: Spracované matričné knihy archívov

Trieda: `ActapublicaArchive`

Súbor: `actapublica_archive.py`

- Zemský archiv v Opavě:

  Trieda: `OpavaArchive`

  Súbor: `opava_archive.py`

- Státní oblastní archiv v Litoměřicích:

  Trieda: `LitomericeArchive`

  Súbor: `litomerice_archive.py`

- Státní oblastní archiv v Hradci Králové:

  Trieda: `ZamrskArchive`

  Súbor: zamrsk_archive.py

Triedy archívov očakávajú na vstupe konektoru cestu adresára, obsahujúceho adresáre archívnych kníh obsahujúce obrázky. Pre spracovanie archívu je nutná implementácia funkcie `_process_archive_book` - určuje ako sa má adresár spracovať a uložiť do databázy informácie o spracovávanej knihe (signatúra, inventárne číslo) a jej obrázkov. Dodatočné informácie o pôvodcovi, type pôvodcu, rokoch a typov záznamov je možné doplniť špecifikovaním súboru JSON. Informácie sa spárujú na základe identifikácie podľa signatúry knihy.

4.3 Predspracovanie

Predspracovanie obrázkov je v aplikáciách počítačového videnia kľúčovou úlohou a zahŕňa celý rad operácií, ktoré sa musia vykonať na prípravu obrázkov na ďalšiu analýzu a použitie. Na uľahčenie tohto procesu sú v adresári `common/preprocessing` zahrnuté súbory zdrojových kódov obsahujúce metódy predspracovania obrazu. Tento balík obsahuje niekoľko pomocných metód, ktoré sú vhodné na spracovanie obrázkov. Tieto metódy sú nevyhnutné na dosiahnutie presných výsledkov v aplikáciách počítačového videnia a môžu výrazne zlepšiť presnosť konečného výstupu.

Medzi tieto metódy patrí odstraňovanie šumu (dostupné v súbore `noise_removal.py`), binarizácia (dostupná v súbore `binarization.py`), normalizácia a štandardizácia (dostupné v súbore `normalization.py`), morfologické transformácie a korekcia natočenia obrazu (dostupná v súbore `deskew.py`) na zabezpečenie vodorovnej polohy textu.

Metóda odstraňovania šumu môže napríklad odstrániť náhodné odchýlky, ktoré sa môžu objaviť v obraze, zatiaľ čo binarizácia môže pomôcť oddeliť popredie textu od pozadia.

Normalizácia a štandardizácia sú užitočné na úpravu hodnôt pixelov obrazu na vhodnejší rozsah kladných hodnôt (napr. z 0-255 na 0-1). Štandardizácia sa vykonáva tak, že sa od každého pixelu odčíta priemer a výsledok sa vydelí štandardnou odchýlkou. Rozdelenie takýchto údajov by sa podobalo Gaussovej krivke so stredom v nule.

4.4 Moduly systému

Jednotlivé modely systému obsahujú izolované prostredie vytvorené pomocou Docker kontajnera. Obsahujú nainštalované balíčky závislostí pre vykonávanie úloh jednotlivých modulov. Súbor `docker-compose.yml` obsahuje definíciu prostredia aplikácie jednotlivých kontajnerov. Systém tvoria moduly: `nginx`, `db`, `backend`, `frontend`, `elasticsearch`, `mlflow`, `crnn_ocr`, `pero_ocr`, `paddle_ocr`. Súbor obsahuje popis služieb pre nasadenie produkčnej verzie systému. Jednotlivé služby znova nezostavujú, ale ich najnovšia verzia sa stiahne z repozitárov Docker Hub[18], kde sú nahraté. Upload zostavených balíčkov na cloud umožní použitie služieb v ich zakonzervovanom stave a tak zjednoduší ich použitie aj s odstupom času hlavne v prípade, kedy by niektoré závislosti už neboli dostupné. Pre vytvorenie vývojového prostredia je vhodnejší súbor `docker-compose.build.yml`.

Konfiguračný súbor `.env` obsahuje premenné prostredia, ktoré možno upraviť na konfiguráciu správania aplikácie. Obsahuje dvojice hodnôt kľúč-hodnota nastavených portov a citlivé informácie o účte do databázy.

4.4.1 Vytvorený OCR model CRNN

V tejto časti sú opísané implementačné detaily vytvoreného OCR modelu pomocou knižnice PyTorch Lightning. Zdrojové kódy neurónovej siete sú v adresári */ocr/crnn_ocr*. Jupyter notebook *crnn_model.ipynb* slúži interaktívna ako demonštrácia trénovania a inferencie siete. Pre klasické trenovanie pomocou skriptu je vhodné využiť *crnn-demo.py*. Konfigurácia parametrov je upraviteľná v súbore `config.py`.

Vytvorený OCR modul bol implementovaný podľa vzoru architektúry CRNN bližšie popísanej v sekcii 1.3. Táto neurónová sieť je využívaná v state of the art (SOTA) riešeniach úlohy pre rozpoznávanie optických znakov ručne písaného textu a to aj v prípadoch historických dokumentov archívov.

Na vstupe očakáva výrez riadku s fixnou výškou. Výstupom je sekvencia, ktorú hladný algoritmus (*greedy search*) dekóduje na text podľa vopred určenej abecedy. Pri vytváraní výstupnej sekvencie sa vyberie vždy len jeden najpravdepodobnejší znak.

Model CRNN

Trieda CRNN obsahuje implementáciu neurónovej siete - definíciu topológie, účelovej funkcie CTC, konfiguráciu metódy optimalizácie parametrov siete - Adam, dopredný prechod a pod. Zdrojové kódy sú uložené v priečinku *model*. Jedná sa o modul `LightningModule`, ktorý definuje ako jednotlivé časti neurónovej siete spolu interagujú. Bloky neurónovej siete sa inicializujú vo funkcii `__init__` Pre trénovanie je potrebné definovať funkciu `training_step`. Vo funkcii `configure_optimizers` sa definuje optimalizátor pre model spolu s rýchlosťou učenia (*learning rate*).

Architektúru siete tvoria konvolučné neurónové siete, tzv. *backbones*. Následne je na výstup konvolučných vrstiev aplikovaná lineárna plne prepojená vrstva `map2seq` a potom

[18] https://hub.docker.com/

rekurentná neurónová sieť GRU. Poslednou časťou je transkripčná vrstva tvorená plne prepojenou lineárnou vrstvou, na ktorú je aplikovaná varianta aktivačnej funkcie Softmax - LogSoftmax, ktorú je nutné použiť z dôvodu využívania CTC chybovej funkcie. Konfiguráciu CRNN je možné ovplyvniť úpravou parametrov v súbore `crnn_ocr/config.py` alebo pri inicializácii v konštruktore. Dôležitým parametrom je určenie `VOCABULARY`, ktorý stanovuje modelu jeho výstupnú abecedu spolu so znakom reprezentujúcim *blank* na začiatku reťazca. Tieto prvky množiny podmieňujú možné klasifikované znaky zo vstupu.

Validačné metriky pre výpočet Levenshtein vzdialenosti CER a WER sú využité z balíčka TorchMetrics.

Dataset a Dataloader

Trieda `Dataset` slúži na načítanie dátových sád do internej reprezentácie využívaných pre trénovanie, validáciu a testovanie modelu. Počas načítania prebehne predspracovanie. Dataset uchováva vzorky a ich zodpovedajúce štítky.

Pre iteratívny proces trénovania, či inferencie modelu trieda DataLoader zapúzdruje Dataset a vytvára tak generátor dát z načítaného datasetu. Prístup ku vzorkám je tak jednoduchší. Vrámci DataLoader inicializácie sa v konštruktori určí veľkosť dávky dát `batch_size`. Pre urýchlenie načítavania je vhodné zvýšiť parameter `num_workers`, ak to dovoľujú pamäťové možnosti, a parameter `pin_memory`, ak pracujeme na zariadeniach CUDA architektúry[19]. Parameter `shuffle` je vhodné vypnúť pri práci s testovacou dátovou sadou.

Trainer

Trieda `Trainer` obstaráva konfiguráciu, vykonanie trénovacieho, validačného a testovacieho cyklu spolu s načítavaním dát z príslušných `dataloaders`. Automaticky zapína a vypína gradienty v závislosti od fázy, ktorú model vykonáva. Taktiež nahráva dáta a vykonáva operácie na správnych zariadeniach. Vykonáva volanie spätných volaní (*callbacks*) vo vhodných časoch, napríklad ukladanie checkpointu modela, a pod.

Pri volaní funkcie `fit` triedy Trainer je nutné definovať objekty Dataloader pre trénovacie dáta a validačné dáta, ak chceme aby prebiehala validácia.

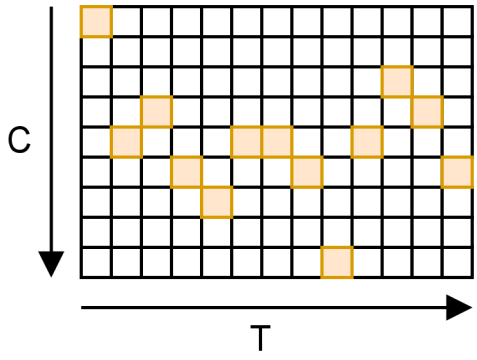
Hodnotenie predikcií

Metóda Mean Max (MM) alebo aj Mean Probability (MP) pracuje s maticou pravdepodobností. Ohodnotí skóre výstupnej sekvencie ako priemer maximálnych všetkých pravdepodobností 4.1. Skóre výsledku predikcie, kde si je model menej istý bude nižšie a hodnoty sú radené vzostupne. Metóda skórovania zachytáva mieru neistoty modelu. Implementácia sa nachádza v súbore `ocr/crnn_ocr/utils/score.py`.

4.4.2 Prevzatý OCR nástroj PaddleOCR

Služba `paddle_ocr` zaisťuje trénovanie a inferenciu s nástrojom PaddleOCR. Trénovanie instancie PaddleOCR prebieha s potvrdenými anotáciami zo systému. PaddleOCR podporuje detekciu aj rozpoznanie textu zo vstupného obrázka. Nástroje je možné trénovať alebo použiť predtrénované modely.

[19] https://developer.nvidia.com/cuda-toolkit



Obr. 4.1: Metóda MM ohodnotenia predikovanej výstupnej sekvencie pomocou pravdepodobnostnej matice; Vrámci jedného časového kroku je vybraná jedna pravdepodobnosť. Celkové ohodnotenie sekvencie je priemerom hodnot naprieč časom.

Trénovanie

PaddleOCR požaduje pre trénovanie modulov detekcie a rozpoznania dáta predspracované. Pre detektor je nutné spracovať anotácie regiónu do zoznamu uloženého v textovom súbore CSV[20], kde hodnoty sú oddelené tabulátorom `\t`, obsahujúcim cestu k obrázku a polom slovníkov anotácii. Každý slovník má dva kľúče, `points` - štyri koordináty súradníc (x,y) usporiadaných v smere hodinových ručičiek od bodu v ľavom hornom rohu a kľuč `transcription` s hodnotou textového reťazca anotácie. Súbor `PP-OCRv3_det.yml` konfiguruje proces trénovania.

Podobne pre rozpoznávač sú spracované potvrdené anotácie regiónov. Oblasť regiónu je vyrezaná z obrázka a uložená do samostatného súboru. Textový súbor CSV obsahuje zoznam dvoch hodnôt - cestu k vyrezanému obrázku regiónu `scan-id-region-idx.jpg` a textový prepis anotácie. Súbor `PP-OCRv3_rec.yml` konfiguruje proces trénovania.

Po natrénovaní je nutná konverzia a export modelov pre použitie na inferenciu. Po každom natrénovaní modelov, prípadne chybe pri trénovaní, sa spracované zoznamy a obrázky zmažú, aby sa vždy trénovalo s aktuálnymi anotovanými dátami.

Inferencia

Vykonávanie inferencie prebieha pomocou triedy `PaddleOCR` a jej funkcie `ocr()`. Pre konštruktor PaddleOCR je nutné uviesť adresáre obsahujúce predtrénované modely detekcie a rozpoznávania. Parametrami funkcie je obrázok, načítaný knižnicou `python-opencv` a prepínače, ktoré vypínajú alebo zapínajú detekciu a rozpoznanie textu zo vstupu.

4.4.3 Plánovač úloh

Jednotlivé moduly sú synchronizované cez databázu. Systém predpokladá nasadenie na jednom zariadení. Preto sú úlohy vykonávané jednotlivo, aby nedochádzalo k zabratiu zdrojov iným modulom.

[20] https://en.wikipedia.org/wiki/Comma-separated_value

4.4.4 Webová aplikácia

Vstup do webovej aplikácie je umožnený pre zaregistrovaných a prihlásených užívateľov. Používatelia si môžu vyhľadať archívne matričné knihy, anotovať textové oblasti, zadať ich prepis a potvrdiť ich správnosť.

Zdrojové kódy implementácie webovej aplikácie sú uložené v adresári `/frontend`. Komponenty využité pre uživateľské rozhranie systému sú v adresári `/frontend/src/components`.

API backend

Endpointy API sú implementované v `backend/endpoints`. Jednotlivé súbory obsahujú kód na spracovanie požiadaviek na jednotlivé endpointy. Modely v adresári `common/api_models` sú využívané na reprezentáciu dátových štruktúr pri požiadavkách a odpovediach API. Súbor `request_argument_parsers.py` obsahuje špecifikáciu parserov vstupných dát pre API endpoint. Pre očakávanie vstupu sa použije dekorátor Flask-RESTX[21] `api.expect`. Využítim tejto konvencii dokchádza aj k validácii vstupných dát. Súbor `response_models.py.py` obsahuje výstupné modely, ktoré je možné využiť pri použití `api.marshal_with` dekorátora na API endpoint. Zoznam dustupných end-point adries:

- `/archive/list`: Zoznam dostupných matrík
- `/archive/<archive_name>/signature/list?signarure`: Vyhľadávanie signatúr v archíve
- `/archive/<archive_name>/signature?signarure` Vyhľadávanie matričných kníh
- `/<archive>/signature/<signature>/image/<idx>/transcripts`: Zoznam regiónov v obrázku archívnej knihy
- `/task/new_task`: Vytvorenie novej úlohy
- `/task/`: Zoznam úloh
- `/user/register`: Registrácia užívateľa do systému
- `/user/login`: Prihlásenie užívateľa do systému
- `/user/me`: Validácia tokenu užívateľa

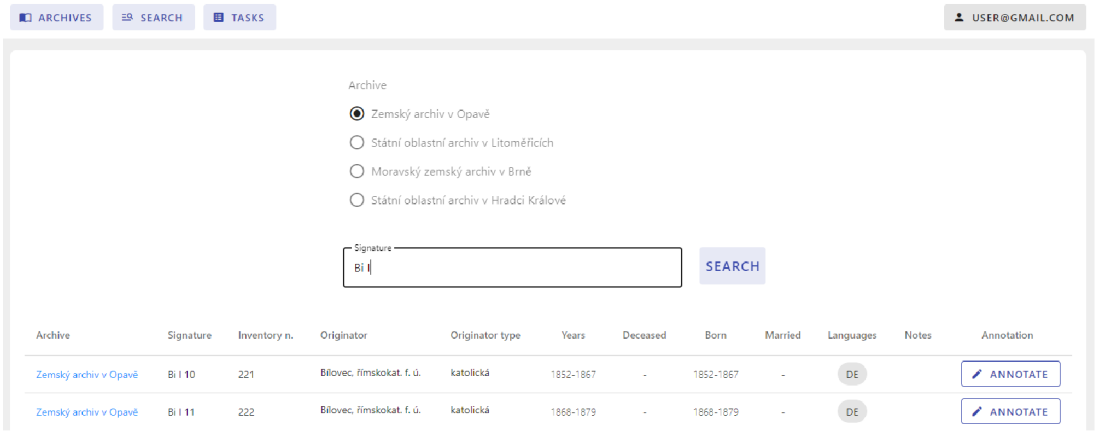
Prehľad archívov

Stránka „Archives“ umožňuje užívateľovi filtrovať archívne knihy jednotlivých archívov podľa ich signatúry (viď obrázok 4.2). Všetky knihy spĺňajúce kritérium vyhľadania sú zobrazené v zozname s dostupnými informáciami o pôvodcovi, rokoch a jazyku knihy. Výsledky obsahujú preklik na anotačné rozhranie webovej aplikácie.

Vytváranie úloh

Užívateľ môže vo webovej aplikácii vytvoriť tri typy úloh - segmentáciu, inferenciu OCR a trénovanie OCR. Segmentáciu vykonáva modul `pero_ocr` a `paddle_ocr`. Inferenciu OCR realizuje `paddle_ocr` a `crnn_ocr`. Rovnako ako aj doučovanie. Pravidelné vykonávanie úloh je možné zabezpečiť úpravou `routine_scheduler.py`, ktorý v určených intervaloch vytvorí úlohu. Moduly systému následne automaticky spracujú príslušnú úlohu.

[21] https://flask-restx.readthedocs.io/en/latest/



Obr. 4.2: Výpis filtrovaných archívnych matričných kníh podľa signatúry

Fulltextové vyhľadávanie prepisov

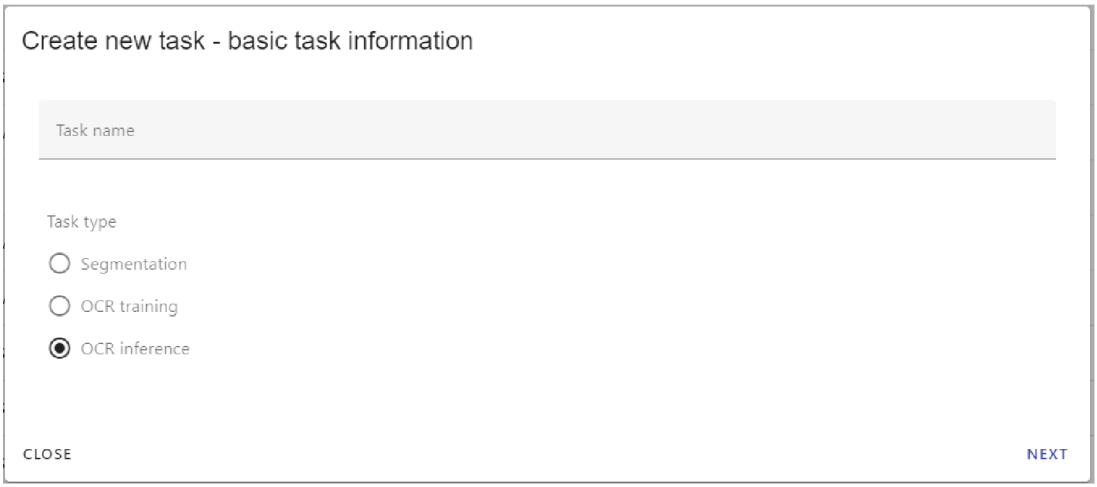
Vyhľadávanie v prepisoch dokumentov tvorí ďalšiu časť webovej aplikácie (viď obrázok 4.4). Užívateľ si môže obmedziť vyhľadávanie na základe filtrov. Je možné zvoliť filter výsledkov na základe potvrdenia správnosti prepisu. Rovnako si môže navoliť zdroj archívnych dokumentov. Užívateľ sa môže zobraziť konkrétny dokument, z ktorého prepis pochádza.

Realizácia fulltextového vyhľadávania je pomocou Elasticsearch (ES). Na správu indexu sa využíva klient Python Elasticsearch v súbore `common/elastic.py`. Definuje slovník mapovania `transcript_mapping`, ktorý opisuje schému indexu Elasticsearch. Schéma je zložená z textového poľa pre text transkriptu a polí kľúčových slov pre archív a signatúru, identifikáciu obrázka a potvrdenie správnosti. Trieda `EsManagement` poskytuje rozhranie na správu indexu. Obsahuje metódy na vytvorenie indexu, vkladanie, aktualizáciu, vymazanie dokumentov v indexe a hromadné vkladanie a aktualizáciu dokumentov v indexe. Metóda `process_transcript` sa používa na spracovanie databázového objektu SQLAlchemy ORM Transcript na dokument, ktorý je možné vložiť do indexu ES. Metóda `upsert` vloží alebo aktualizuje jeden dokument do indexu pomocou indexovej metódy, ktorú poskytuje klient Elasticsearch. Metóda `upsert_bulk` vkladá alebo aktualizuje viacero dokumentov do indexu. Metóda `delete_index` odstráni celý index. Metódy `delete_by_id` a `delete_by_ids` odstránia dokument alebo dokumenty z indexu na základe ID.

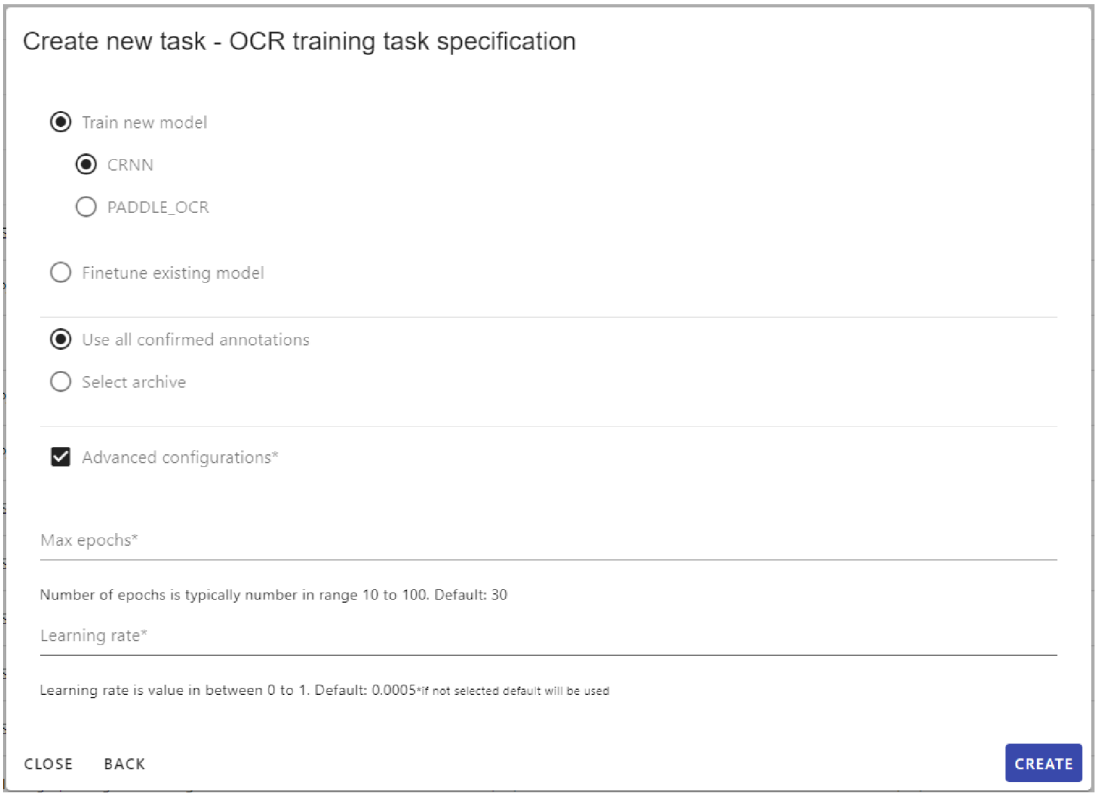
Komponenty implementujúce webové rozhranie vyhľadávania sú umiestnené v priečinku `components/search`. Konfiguračný skript `searchConfig.js` vytvára konektor na Elasticsearch API a vymedzuje správanie vyhľadávania v indexe. Komponenty vytvárajú interaktívne vyhľadávacie prostredie pre používateľov. Objekt `searchQuery` špecifikuje vyhľadávací dotaz, ktorý sa odosiela. Komponent `SearchFacet.vue` definuje fazetový filter s viacerými možnosťami zaškrtávacích políčok a logiku filtrácie vyhľadávaných výsledkov. Komponent `SearchResult.vue` určuje štruktúru a vizualizáciu výsledkov vyhľadania.

Anotácia dokumentov

Webová aplikácia umožňuje anotáciu dostupných dokumentov. Po načítaní stránka obsahuje obrázok skenu dokumentu s predspracovanými predikciami - segmentované oblasťami (*bounding-box*), rozpoznaný text príslušných oblastí.

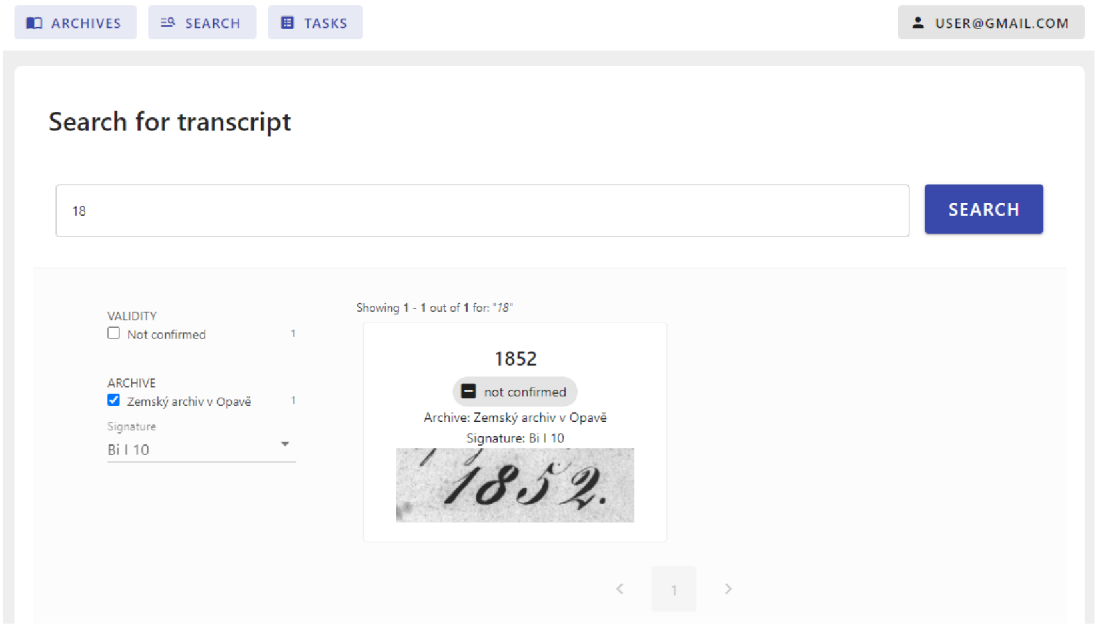


(a) Voľba typu úlohy



(b) Špecifikácia parametrov úlohy trénovania

Obr. 4.3: Vytváranie úloh pomocou webovej aplikácie



Obr. 4.4: Ukážka stránky z webovej aplikácie fulltextového vyhľadávania v prepisoch riadkov

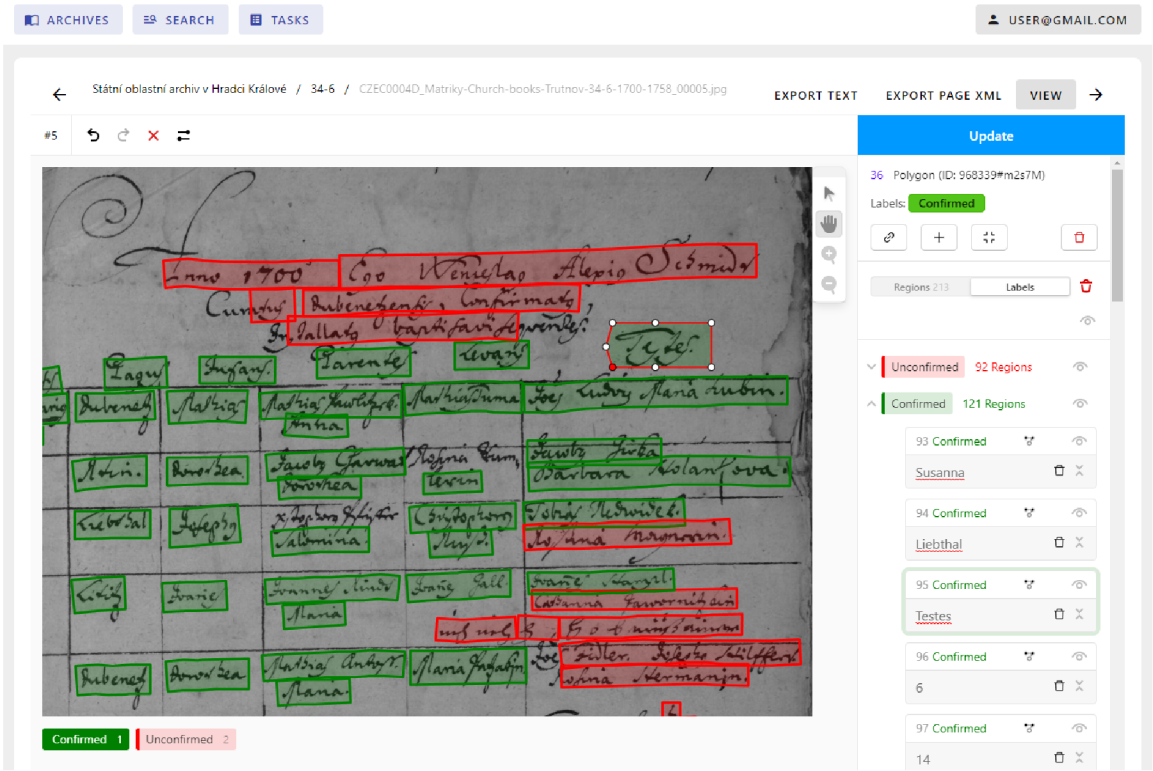
Implementácia komponenty (viď obrázok 4.5) je vytvorená pomocou open-source nástroja Label Studio[22]. Podporuje anotáciu dát rôznych druhov - obrázkové, audio, a pod. Komponenta `AnnotationDetail.vue` obsahuje konfiguráciu Label Studio Frontend.

Export anotácii Implementácia exportu údajov vo formáte Page XML alebo TXT anotácií je užitočná funkcia pre užívateľa na analýzu a zdieľanie anotácii. Page XML je formát založený na XML. Bežne sa používa na reprezentáciu rozloženia a obsahu digitalizovaných naskenovaných dokumentov, ako sú knihy, noviny a rukopisy. Tento formát umožňuje jednoduchú manipuláciu s textovými, obrazovými a metadátovými prvkami. Obsahuje hierarchickú štruktúru oblastí strany, textových oblastí, riadkov a slov a ich obsah. Formát TXT je jednoduchý formát textových súborov, ktorý sa ľahko číta a spracováva, ale neobsahuje polygóny oblastí riadkov. Export anotácii v oboch formátoch umožňuje používateľom zdieľať svoje údaje alebo ich ďalej spracovávať pre svoju potrebu.

```
<PcGts xmlns="http://schema.primaresearch.org/PAGE/gts/pagecontent/2019-07-15" xmlns:xsi="
    http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://schema.
    primaresearch.org/PAGE/gts/pagecontent/2019-07-15/pagecontent.xsd">
  <Page imageFilename="hello_word.jpg" imageWidth="1000" imageHeight="5000"">
      <TextRegion id="r1" type="paragraph">
          <Coords points="10,20 300,20 300,50 10,50"/>
          <TextLine id="l1">
              <Coords points="10,20 300,20 300,30 10,30"/>
              <TextEquiv><Unicode>Hello World!</Unicode></TextEquiv>
          </TextLine>
      </TextRegion>
  </Page>
</PcGts>
```

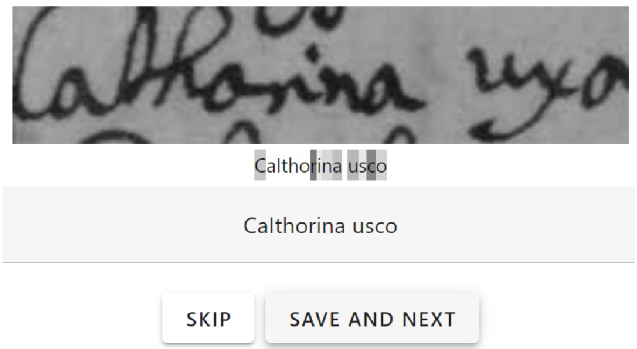
Výpis 4.1: Ukážka formátu Page XML

[22] labelstud.io



Obr. 4.5: Ukážka anotačnej komponenty LabelStudio; Zelené regióny sú potvrdené oblasti, červéne regióny nepotvrdené riadky s prediovanými prepismy

Navrhovanie anotácii Webová aplikácia umožňuje odporučovanie anotácie textového prepisu s najnižším skóre istoty. Užívateľovi sa zobrazí výsek polygónu textovej oblasti a predikovaná sekvencia so zvýraznenými charaktermi, ktoré majú nízku pravdepodobnosť výskytu.



Obr. 4.6: Ukážka odporučenia anotácie textového prepisu - obsahuje výrez textovej oblasti regiónu, predikovaný prepis, so zvýraznenými charaktermi, u ktorých si model nie je istý.

4.5 Testovanie systému

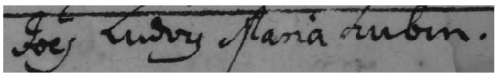
Táto časť sa popisuje postup testovanie systému a vyhodnotenia úspešnosti modelu CRNN optického rozpoznávania znakov. Systém bol nasadený a otestovaný na stroji *pcrozman2*. Je dostupný verejnosti na adrese. Stroj obsahuje dve grafické karty NVIDIA GeForce RTX 2080 Ti 11 GB a NVIDIA GeForce GTX 1080 Ti 11 GB.

4.5.1 Vyhodnotenie úspešnosti modelu CRNN

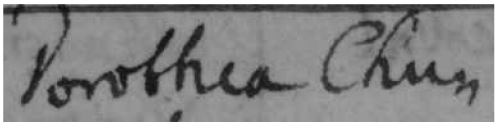
Typická miera úspešnosti OCR sa líši v závislosti od dát. OCR systémy tlačených dokumentov dosahujú CER 1-2%. Rukopisný text je ťažšie čitateľný sám o sebe a závisí od kvality a typu pôvodného dokumentu. HTR systémy poskytujú presnosť od CER 50%.

Pred trénovaním boli vstupné dáta z dátových sád predspracované. Pri spracovávaní bola určená výška obrázku riadku. Šírku obrázku ovplyvňuje dávka, v ktorej sa daná vzorka predloží modelu. V závislosti od najširšieho obrázku sa ostatné vstupné obrázky doplnia sprava na najširšiu šírku, tak aby text začínal z ľavého okraja obrázka. Následne boli obrázky prevedené do tensorovej reprezentácie a každá hodnota pixelu bola normalizovaná s cieľom rýchlejšej konvergencie. Popis výpočtu metrík WER a CER bol bližšie opísaný v časti 1.2. Tabuľka 4.2 zhrňuje dosiahnuté výsledky úspešnosti implementovaného modelu CRNN.

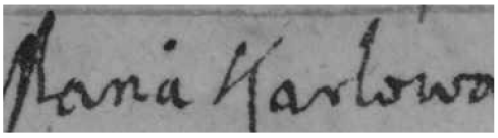
Porovnanie úspešnosti rozpoznania znakov s predošlým riešením je približné z dôvodu neznámeho rozdelenia dát do trénovacej a testovacej dátovej sady. Trénovanie a vyhodnotenie modelov bolo vykonané na náhodnom rozdelení na trénovaciu a testovaciu dátovú sadu v pomere 90:10. V prípade dátovej sady Bentham R0 bolo využité náhodné rozdelenie napriek dostupnému odporúčanému zoznamu rozdelenia dát do jednotlivých dátových sád. Najlepšími dosiahnutými výsledkami CER metriky na dátovej sade Bentham R0 podľa článku [27] boli v rozmedzí intervalu 4-8% a matrika WER v intervale 33-40%. S využitím augmentácie dát bolo dosiahnuté podľa článku [38] CER 2-3% a WER 8-13%. Predošlé riešenie [14] dosiahlo CER 5% a WER metrika nebola uvedená. Výsledky úspešnosti rozpoznávania sú uvedené v tabuľke 4.2.



(a) Potvrdený región s prepisom *Joes Ludvig Maria Kubin*



(b) Región s predikovaným prepisom CRNN *Porothea Chiars*



(c) Región s predikovaným prepisom CRNN *Mana Karlowa*

Obr. 4.7: Ukážky regiónov z matričnej knihy sig. 34-6 Státní oblastní archiv v Hradci Králové

ZAMRSK sig. 34-6 je dátová sada vytvorená amatérskymi genealógmi. Obsahuje 20 stránok a približne 4000 anotácii z prvých 20 obrázkov matričnej knihy sig. 34-6 z Státního oblastního archivu v Hradci Králové. Kniha bola vytvorená v rokoch 1700 až 1758 a jej rukopis je ťažšie čitateľnejší. Preto pri trénovaní som použila predtrénovaný model na dátovej sade Bentham. Model tak konvergoval rýchlejšie. Výsledky úspešnosti rozpoznávania sú uvedené v tabuľke 4.2.

Rozpoznanie regiónov riadkov PaddleOCR je v porovnaní s PeroOCR horšie. Funguje na regiónoch tlačeného písma a na novšie vytvorených matričných knihách. Na starších rukopisoch, kde je väčšia tendencia horšieho rukopisu, nedokáže detekovať všetky riadky.

Dátová sada	CER	WER
Bentham R0	0.07	0.38
ZAMRSK sig. 34-6	0.04	0.2

Tabuľka 4.2: Výsledky úspešnosti modelom CRNN

4.5.2 Vyhodnotenie výpočtovej náročnosti

Porovnanie doby trvania trénovania a inferencie na dátovej sade Bentham R0 na grafickej karte NVIDIA GeForce GTX 1660 Ti zhrňuje tabuľka 4.4 a tabuľka 4.3 trvanie na procesore AMD Ryzen 5 3600. Merania nezahrňujú čas segmentácie, ukladania dát do databázy a do indexu Elasticsearch, ani načítanie vstupných dát do tensoru výseku obrázka riadku a tensoru transkripcie. Nástroj PERO OCR priemerne spracoval segmentáciu 46 stránok za minútu. Spracovanie aktuálne dostupných 8 miliónov snímkov v systéme by zabralo približne 120 dní.

Pre trénovanie a inferenciu CRNN OCR je nutné orezať textové oblasti zo vstupného obrázka. Priemerný počet spracovaných regiónov bol 8000 za minútu. Stránky archívnych matričných dokumentov obsahujú typicky 100 až 300 regiónov, priemerne 200. Archívne knihy majú priemerne 150 strán. Ak by sme predpokladali pre jednu 150 listovú knihu 150 regiónov na stránku, spracovanie pred trénovaním alebo inferenciou by vyžadovalo približne

Dátová sada	Rozdelenie	Počet vzoriek	Veľkosť dávky	Počet iterácii	Trvanie [s]
Bentham R0	trénovanie	10613	8	1327	∼1320
			16	664	∼1300
			32	332	∼1560
	inferencia	860	1	860	∼90
			8	108	∼50
			16	54	∼50
			32	27	∼50

Tabuľka 4.3: Porovnanie doby trvania trénovania a inferencie na dátovej sade Bentham R0 na procesore AMD Ryzen 5 3600

3 minúty. Pre zrýchlenie procesu inferencie bol model CRNN ukladaný v ScriptMode vo formáte JIT[23].

Počas nasadenia systému bola meraná celková výpočtová náročnosť úlohy inferencie - načítanie dát, inferencia, uloženie dát do databáze. Pre 386 obrázkových dokumentov obsahujúcich približne 60 000 regiónov. Predspracovanie oblastí trvalo približne 8 min, inferencia 90 sekúnd a následne uloženie do databázy a Elasticsearch indexu 4 minúty.

Dátová sada	Rozdelenie	Počet vzoriek	Veľkosť dávky	Počet iterácii	Trvanie [s]
Bentham R0	trénovanie	10613	8	1327	∼100
			16	575	∼60
	inferencia	860	1	860	∼45
			8	108	∼12
			16	54	∼6
			32	27	∼5

Tabuľka 4.4: Porovnanie doby trvania trénovania a inferencie na dátovej sade Bentham R0 na grafickej karte NVIDIA GeForce GTX 1660 Ti

Dátová sada	Rozdelenie	Počet vzoriek	Veľkosť dávky	Počet iterácii	Trvanie [s]
Bentham R0	trénovanie	10613	8	1327	∼34
			16	664	∼24
			32	332	∼24
			64	166	∼32
	inferencia	860	1	860	∼10
			8	108	∼2
			16	54	∼1
			32	27	∼1
			256	4	∼3

Tabuľka 4.5: Porovnanie doby trvania trénovania a inferencie na dátovej sade Bentham R0 na grafickej karte NVIDIA GeForce GTX 2080 Ti

[23] https://pytorch.org/docs/stable/jit.html

Kapitola 5

Záver

Cieľom tejto diplomovej práce bolo naštudovať prístupy optického rozpoznávania znakov so zameraním na ručne písaný text a metódy aktívneho učenia. Okrem toho bolo potrebné navrhnúť systém, ktorý by využíval rôzne OCR moduly pre rozpoznávanie textu - dostupný prebratý a vytvorený OCR model. Navrhnutý systém bol následne implementovaný a otestovaný. Systém umožňuje úpravy detekcií segmentovaných oblastí a rozpoznaného textu a z potvrdených anotácií doučovať modely vykonávajúce OCR. Systém podporuje fulltextové vyhľadávanie vrámci rozpoznaných prepisov textov. Úspešnosť rozpoznávania bola vyhodnotená. Práca sa tiež zameriava zhodnotenie výpočtovej zložitosť pre veľké množstvo skenov dokumentov. Riešenie obsahuje indexáciu 50 tisíc matričných kníh zo štyroch archívov s údajmi o ich signatúre, inventárnom čísle, rokmi vzniku a jazykom písma spolu s ich skenmi. Dokopy systém obsahuje približne osem miliónov skenov, ktoré sú pripravené na postupné spracovanie OCR. Anotácie, vytvorené počas sprístupnenia predošlého riešenia boli importované aj do stávajúceho riešenia. Systém bol nasadený a webová aplikácia je používateľom dostupná z verejného internetu. OCR moduly je možné pravidelne doučovať z potvrdených anotácií prostredníctvom automatizovaných úloh.

V porovnaní s predchádzajúcim vypracovaným riešením došlo ku zakomponovaniu niekoľkých návrhov rozšírení práce. Bol pridaný export prepisov v textovom formáte a PAGE XML. Užívatelia majú archívne knihy sprístupnené k prezeraniu a anotovaniu. Predošlé databázové vyhľadávanie bolo vylepšené využitím pokročilejšieho vyhľadávača Elasticsearch. Zakomponovaním komponenty MLflow umožňuje jednoduchšiu správu modelov a sledovanie výsledkov trénovania a validácie. Platformu je možné využiť aj pre ďalšie rozšírenia systému o modely strojového učenia. Systém navyše umožňuje doučovanie modulu detektoru PaddleOCR textových oblastí, rovnako aj modulov rozpoznávača textu PaddleOCR a CRNN.

Ďalšie možnosti rozšírenia systému sa vyskytujú v pridaní iných OCR, dekódovaním výstupu modelu CRNN pomocou Word Beam Search namiesto Greedy Search, zvyšuje však časovú náročnosť. Rozšírenie systému o rozpoznávanie pomenovaných entít (NER) by uľahčilo vyhľadávanie v rozpoznaných prepisoch matrík, ktoré sa v budúcnu plánuje zakomponovať s bakalárskou prácou ďalšieho študenta. S ohľadom na veľké množstvo archiválnych dokumentov, by jednou z možností rozšírení mohlo byť distribuované rozdelenie úloh učenia a inferencie na viacero zariadení. Pre malé množstvo anotovaných archiválii by augmentácia dát mohla rozšíriť trénovacie dáta a zlepšiť výsledky OCR. V rámci aktívneho učenia by mohlo byť experimentované s ďalšími prístupmi ohodnotenia predikcii. Napokon, úspešnosť OCR by taktiež mohol vylepšiť krok postprocessingu.

Literatúra

[1] Bianne Bernard, A.-L., Menasri, F., Mohamad, R. A.-H., Mokbel, C., Kermorvant, C. et al. Dynamic and contextual information in HMM modeling for handwritten word recognition. *IEEE transactions on pattern analysis and machine intelligence*. IEEE. 2011, zv. 33, č. 10, s. 2066–2080.

[2] Bluche, T., Ney, H. a Kermorvant, C. Tandem HMM with convolutional neural network for handwritten word recognition. In: IEEE. *2013 IEEE international conference on acoustics, speech and signal processing*. 2013, s. 2390–2394.

[3] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D. et al. Language models are few-shot learners. *Advances in neural information processing systems*. 2020, zv. 33, s. 1877–1901.

[4] Burdett, E., Fujimoto, S., Brown, T., Shurtz, A., Segrera, D. et al. Active Transfer Learning for Handwriting Recognition. In: Springer. *International Conference on Frontiers in Handwriting Recognition*. 2022, s. 245–258.

[5] Cho, K., Van Merriënboer, B., Bahdanau, D. a Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. *ArXiv preprint arXiv:1409.1259*. 2014.

[6] Comer, M. L. a Delp III, E. J. Morphological operations for color image processing. *Journal of electronic imaging*. SPIE. 1999, zv. 8, č. 3, s. 279–289.

[7] Dave, N. Segmentation methods for hand written character recognition. *International journal of signal processing, image processing and pattern recognition*. 2015, zv. 8, č. 4, s. 155–164.

[8] Drira, F. Towards restoring historic documents degraded over time. In: IEEE. *Second International Conference on Document Image Analysis for Libraries (DIAL'06)*. 2006, s. 8–pp.

[9] Fischer, A., Frinken, V., Fornés, A. a Bunke, H. Transcription alignment of Latin manuscripts using hidden Markov models. In: *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*. 2011, s. 29–36.

[10] Graves, A., Fernández, S., Gomez, F. a Schmidhuber, J. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: *Proceedings of the 23rd international conference on Machine learning*. 2006, s. 369–376.

[11] Graves, A. a Schmidhuber, J. Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks. In: Koller, D., Schuurmans, D., Bengio, Y. a Bottou, L., ed. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2008, sv. 21. Dostupné z: https://proceedings.neurips.cc/paper/2008/file/66368270ffd51418ec58bd793f2d9b1b-Paper.pdf.

[12] Hedjam, R. a Cheriet, M. Historical document image restoration using multispectral imaging system. *Pattern Recognition*. 2013, zv. 46, č. 8, s. 2297–2312. DOI: https://doi.org/10.1016/j.patcog.2012.12.015. ISSN 0031-3203. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0031320313000642.

[13] Hochreiter, S. a Schmidhuber, J. Long short-term memory. *Neural computation*. MIT Press. 1997, zv. 9, č. 8, s. 1735–1780.

[14] Hříbek, D. *Active Learning pro zpracování archivních pramenů*. Brno, CZ, 2021. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/23784/.

[15] Kodym, O. a Hradiš, M. Page layout analysis system for unconstrained historic documents. In: Springer. *International Conference on Document Analysis and Recognition*. 2021, s. 492–506.

[16] Kumar, P. *Time Complexity of ML Models* [https://medium.com/analytics-vidhya/time-complexity-of-ml-models-4ec39fad2770]. 2019 [cit. 27.04.2022].

[17] Lecun, Y., Bottou, L., Bengio, Y. a Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. 1998, zv. 86, č. 11, s. 2278–2324. DOI: 10.1109/5.726791.

[18] Leung, K. *Evaluate OCR output quality with character error rate (CER) and word error rate (WER)*. Towards Data Science, Sep 2021 [cit. 27.12.2022]. Dostupné z: https://towardsdatascience.com/evaluating-ocr-output-quality-with-character-error-rate-cer-and-word-error-rate-wer-853175297510#1db9.

[19] Levenshtein, V. I. et al. Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet Union. *Soviet physics doklady*. 1966, sv. 10, č. 8, s. 707–710.

[20] Marti, U.-V. a Bunke, H. The IAM-database: an English sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*. Nov 2002, zv. 5, č. 1, s. 39–46. DOI: 10.1007/s100320200071. ISSN 1433-2833. Dostupné z: https://doi.org/10.1007/s100320200071.

[21] Mays, E., Damerau, F. J. a Mercer, R. L. Context based spelling correction. *Information Processing & Management*. Elsevier. 1991, zv. 27, č. 5, s. 517–522.

[22] Morris, A., Maier, V. a Green, P. From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition. In:. Január 2004.

[23] Muehlberger, G., Seaward, L., Terras, M., Oliveira, S. A., Bosch, V. et al. Transforming scholarship in the archives through handwritten text recognition: Transkribus as a case study. *Journal of documentation*. Emerald Publishing Limited. 2019.

[24] Nurseitov, D., Bostanbekov, K., Kurmankhojayev, D., Alimova, A., Abdallah, A. et al. Handwritten Kazakh and Russian (HKR) database for text recognition. *Multimedia Tools and Applications*. Springer. 2021, zv. 80, č. 21, s. 33075–33097.

[25] Olah, C. *Understanding LSTM Networks*. 2015 [cit. 20.12.2022]. Dostupné z: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[26] Otsu, N. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*. IEEE. 1979, zv. 9, č. 1, s. 62–66.

[27] Potanin, M., Dimitrov, D., Shonenkov, A., Bataev, V., Karachev, D. et al. Digital Peter: New Dataset, Competition and Handwriting Recognition Methods. In: *The 6th International Workshop on Historical Document Imaging and Processing*. 2021, s. 43–48.

[28] Puigcerver, J. Are multidimensional recurrent layers really necessary for handwritten text recognition? In: IEEE. *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. 2017, sv. 1, s. 67–72.

[29] Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z. et al. A survey of deep active learning. *ACM computing surveys (CSUR)*. ACM New York, NY. 2021, zv. 54, č. 9, s. 1–40.

[30] Romero, V., Sánchez, J. A. a Toselli, A. H. Active learning in handwritten text recognition using the derivational entropy. In: IEEE. *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2018, s. 291–296.

[31] Sánchez, J. A., Romero, V., Toselli, A. H. a Vidal, E. ICFHR2014 competition on handwritten text recognition on transcriptorium datasets (HTRtS). In: IEEE. *2014 14th International Conference on Frontiers in Handwriting Recognition*. 2014, s. 785–790.

[32] Sauvola, J. a Pietikäinen, M. Adaptive document image binarization. *Pattern recognition*. Elsevier. 2000, zv. 33, č. 2, s. 225–236.

[33] Settles, B. *Active Learning Literature Survey*. Computer Sciences Technical Report 1648. University of Wisconsin–Madison, 2009.

[34] Seung, H. S., Opper, M. a Sompolinsky, H. Query by committee. In: *Proceedings of the fifth annual workshop on Computational learning theory*. 1992, s. 287–294.

[35] Sezgin, M. a Sankur, B. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic imaging*. SPIE. 2004, zv. 13, č. 1, s. 146–165.

[36] Shen, Z., Zhang, K. a Dell, M. A large dataset of historical japanese documents with complex layouts. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, s. 548–549.

[37] Shi, B., Bai, X. a Yao, C. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*. IEEE. 2016, zv. 39, č. 11, s. 2298–2304.

[38] Shonenkov, A., Karachev, D., Novopoltsev, M., Potanin, M. a Dimitrov, D. StackMix and Blot Augmentations for Handwritten Text Recognition. *ArXiv preprint arXiv:2108.11667*. 2021.

[39] Simonyan, K. a Zisserman, A. Very deep convolutional networks for large-scale image recognition. *ArXiv preprint arXiv:1409.1556*. 2014.

[40] Smith, R. An overview of the Tesseract OCR engine. In: IEEE. *Ninth international conference on document analysis and recognition (ICDAR 2007)*. 2007, sv. 2, s. 629–633.

[41] Tschichold, C. Lexically driven error detection and correction. *Calico Journal*. JSTOR. 2003, s. 549–559.

[42] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L. et al. Attention is all you need. *Advances in neural information processing systems*. 2017, zv. 30.

[43] Villar, S., Torcida, S. a Acosta, G. Median Filtering: A New Insight. *Journal of Mathematical Imaging and Vision*. Máj 2017, zv. 58, s. 1–17. DOI: 10.1007/s10851-016-0694-0.

[44] White, D. *Comparison-of-ocr-a-and-ocr-b*. 2018 [cit. 14.12.2022]. Dostupné z: https://www.jaguarsoftware.com/?attachment_id=2210.

[45] Woodard, J. a Nelson, J. An information theoretic measure of speech recognition performance. In:. 1982.

Príloha A

Obsah priloženého pamäťového média

- `/latex` - LaTeXprojekt technickej správy diplomovej práce
- `/src` - zdrojové kódy aplikácie

Docker image systému sú dostupné na cloudovej platforme Docker hub - https://hub.docker.com/u/betsst

Príloha B

Spustenie riešenia

Pre vývoj aplikácie je vhodné používať `docker-compose.build.yml`. Obsahuje pomocné služby vhodné k debugu. Pre nasadenie zas `docker-compose.yml`. Systémové premenné sú definované v `.env.example`.

- `docker compose [-f docker-compose.build.yml] up [-d]` - znovu vytvorenie systému a spustenie služieb [na pozadí]
- `docker compose -f docker-compose.build.yml down` - zastavenie a odstránenie služieb
- `docker compose [-f docker-compose.build.yml] build` - vytvorenie Docker image služieb v projekte
- `docker compose [-f docker-compose.build.yml] start` - spustenie systému
- `docker compose [-f docker-compose.build.yml] stop` - zastavenie systému
- `docker compose [-f docker-compose.build.yml] logs {service}` - zobrazenie logov služby
- `docker stats` - sledovanie vytaženia zdrojov
- `docker ps` - zoznam aktívnych služieb