VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ZVYŠOVÁNÍ KONZISTENCE V DATOVÝCH SADÁCH PRO ROZPOZNÁVÁNÍ TEXTU

IMPROVING CONSISTENCY IN TEXT RECOGNITION DATASETS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MATÚŠ TVAROŽNÝ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MARTIN KIŠŠ**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Tvarožný Matúš**
Program: Informační technologie
Název: **Zvyšování konzistence v datových sadách pro rozpoznávání textu**
Improving Consistency in Text Recognition Datasets
Kategorie: Zpracování obrazu

Zadání:

1. Prostudujte základy konvolučních sítí a rozpoznávání textu.
2. Vytvořte si přehled o současných metodách rozpoznávání textu pomocí konvolučních sítí a o problémech konzistence datových sad určených pro jejich trénování.
3. Vyberte nebo navrhněte metodu pro zvýšení konzistence datových sad pro rozpoznávání textu.
4. Obstarejte si datovou sadu vhodnou pro experimenty.
5. Implementujte navrženou metodu a proveďte experimenty nad datovou sadou.
6. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
7. Vytvořte stručné video prezentující vaši práci, její cíle a výsledky.

Literatura:

- SHI, Baoguang; BAI, Xiang; YAO, Cong. 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*, 2016, 39.11: 2298-2304.
- KANG, Lei, J. Ignacio TOLEDO, Pau RIBA, Mauricio VILLEGAS, Alicia FORNÉS a Marçal RUSIOL. Convolve, Attend and Spell: An Attention-based Sequence-to-Sequence Model for Handwritten Word Recognition. In: *Pattern Recognition*. Stuttgart, Germany: Springer International Publishing, 2019, s. 459-472. ISBN 978-3-030-12938-5.
- SHENG, Fenfen; CHEN, Zhineng; XU, Bo. NRTR: A no-recurrence sequence-to-sequence model for scene text recognition. In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2019. p. 781-786.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

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

Vedoucí práce: **Kišš Martin, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 2. listopadu 2021

Abstrakt

Táto práca sa zaoberá zvyšovaním konzistencie dátových sád pre rozpoznávanie textu. V tejto práci sú popísane problémy, ktoré nekonzistenciu spôsobujú a následne sú predstavené riešenia na jej odstránenie. Skúmaný je vplyv vlastností polygónov definujúcich ohraničenie riadkov a teda to ako upravená verzia dátovej sady, ktorá je zložená z ideálnych variant riadkov ovplyvnila presnosť modelu. Ďalej sa práca zameriava na detekciu a následné odstránenie alebo upravenie riadkov, ktorých prepis ground truth nekorešponduje so skutočným textom, ktorý sa na nich nachádza. Experimentovaním sa ukázalo, že odstránenie vizuálnej nekonzistencie na trénovacej sade nemá zásadný vplyv na natrénovanosť modelu, za to poupravením testovacej sady sa presnosť OCR modelu zlepšila o 1.1% CER. Upravením dátovej sady tak, aby neobsahovala navzájom nekonzistentné dvojice rozpoznávaného textu a príslušnej ground truth, sa model po opätovnom natrénovaní zlepšil maximálne len o 0.2% CER. Hlavným zistením tejto práce je predovšetkým preukázaný priaznivý účinok odstránenia nekonzistencie na testovacích sadách, vďaka ktorému je možné zistiť reálnejšiu chybovosť OCR modelu.

Abstract

This work is concerned with increasing the consistency of datasets for text recognition. This paper describes the problems that cause the inconsistency and then presents solutions to eliminate it. The effect of the properties of the polygons defining the text line boundaries and hence how the modified version of the dataset, which is composed of ideal text line variants, affected the accuracy of the model is investigated. Further, the work focuses on detecting and then removing or modifying text lines whose ground truth transcription does not match the actual text they contain. Experimentation showed that removing the visual inconsistency on the training set did not have a significant effect on the trained model, but modifying the test set improved the OCR accuracy of the model by 1.1% CER. By modifying the dataset so that it did not contain mutually inconsistent pairs of recognized text and the corresponding ground truth, the model improved by a maximum of only 0.2% CER after re-training. The main finding of this work is, above all, the proven beneficial effect of removing inconsistencies on test suites, thanks to which it is possible to determine a more realistic error rate of the OCR model.

Kľúčové slová

rozpoznávanie textu, OCR, HTR, neurónové siete, NN, konvolučné neurónové siete, CNN, rekurentné neurónové siete, RNN, sequence to sequence, se2seq, CTC, konzistencia, dátové sady

Keywords

text recognition, OCR, HTR, neural networks, NN, convolutional neural networks, CNN, recurrent neural networks, RNN, sequence to sequence, se2seq, CTC, consistency, datasets

Citácia

TVAROŽNÝ, Matúš. *Zvyšování konzistence v datových sadách pro rozpoznávání textu.* Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Martin Kišš

Zvyšování konzistence v datových sadách pro rozpoznávání textu

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Martina Kišša. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Matúš Tvarožný
10. mája 2022

Poďakovanie

Chcel by som poďakovať môjmu vedúcemu, pánovi Ing. Martinovi Kiššovi za odbornú pomoc, trpezlivosť, ochotu, cenné rady a vedenie tejto bakalárskej prace.

Obsah

Kapitola 1

Úvod

Rozpoznávanie textu alebo OCR z anglického spojenie *Optical Character Recognition* je metóda, ktorej hlavnou úlohou je preklad tlačeného alebo písaného textu do digitálnej podoby, typicky ASCII znakov, ktorú je ďalej možné upravovať, vyhľadávať v nej alebo s ňou prevádzať iné potrebné úkony. Je to zložitý proces, hlavne kvôli rôznorodému rozpoznávanému textu, ktorý môže byť v mnohých podobách, jazykoch, z rôznych období a písaný rozličnými štýlmi alebo autormi.

Má širokú škálu uplatnenia a prínos nachádza v podobe automatickej digitalizácie dokumentov a textov, ktoré boli vytvorené v písaných alebo tlačených formách. Tak isto sa využíva pri rozpoznávaní osobných dokladov, poštových smerovacích čísel či čítaní dopravných značiek v autonómne riadených autách.

Dnes sa na riešenie tejto problematiky používajú neurónové siete, ktoré dosahujú veľmi dobré výsledky. Existuje viacero *state-of-the-art* modelov, ktoré sa dnes na rozpoznávanie písma používajú. OCR je možno riešiť architektúrou založenou na konvolučne rekurentných sieťach alebo prístupom *sequence to sequence*. Tieto metódy budú neskôr bližšie popísané v samotnej práci.

Na prácu s nimi sú potrebné dátové sady, ktoré sa konkrétne využívajú hlavne pri ich trénovaní. Táto práca sa zaoberá práve zvyšovaním ich konzistencie a skúmaním, aký vplyv má samotná konzistencia na natrénovanosť modelu. Cieľom je dosiahnuť zlepšenie presnosti použitého OCR modelu po jeho opätovnom natrénovaní na trénovacej sade, ktorej konzistencia bola zvýšená. Pre tieto účely boli navrhnuté a neskôr skúmané metódy, ktoré upravujú alebo odstraňujú nekonzistentné riadky použitej dátovej sady.

Práca je členená do niekoľkých kapitol. Kapitola 2 približuje aktuálne prístupy k rozpoznávaniu textu a je v nej opísaná trojica rozličných architektúr neurónových sietí, ktoré sa dnes v tejto problematike používajú. Kapitola 3 obsahuje prehľad dátových sád určených na rozpoznávanie textu. V kapitole 4 sú opísané problémy, ktoré súvisia s nekonzistenciou v dátových sadách spoločne s ich návrhmi riešenia. V kapitole 5 sú zhrnuté implementačné detaily skriptov, ktoré boli vytvorené pre účely potlačenia danej nekonzistencie. V kapitole 6 sú zhrnuté výsledky experimentov a vplyv upravených dátových sád na úspešnosť testovaného modelu. V závere práce, v kapitole 7, sa nachádza zhodnotenie dosiahnutých výsledkov a možnosti budúcej práce.

Kapitola 2

Rozpoznávanie textu

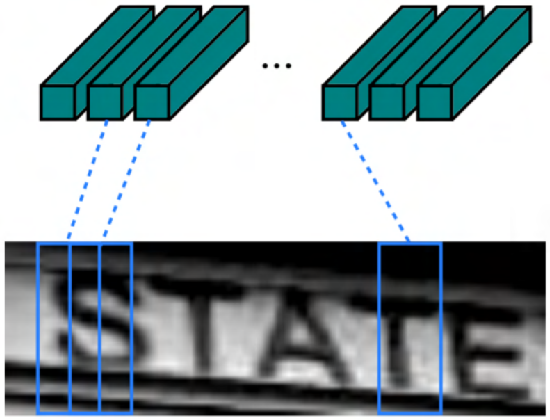
Rozpoznávanie textu alebo OCR je metóda, ktorá sa stará o prevod písaného alebo tlačeného textu do digitálnej podoby. Prvý OCR systém, nazývaný *Optophone* predstavil *Edmund Edward Fournier d'Albe* už pred vyše 100 rokmi. Jeho hlavnou motiváciou bolo umožniť prístup k písaným informáciám slepým ľuďom. Prístroj pri svojej činnosti využíval foto senzory, s pomocou ktorých detekoval čierne písmo z pozadia a prevádzal ho na zvuk[9]. Ďalší významný pokrok nastal v roku 1977, kedy *Ray Kurzweil* predstavil *Kurzweil reading machine*[23]. Tento prístroj opäť prevádzal tlačený text na zvuk. Zo stránky textu rozpoznal písmená a tie neskôr spájal do slov. Každé identifikované písmeno predstavovalo maticu, z ktorej boli postupne kontrolované geometrické črty, ktorých sumarizácia určovala výslednú triedu rozpoznávaného písmena. Dnes sa k riešeniu tohto problému pristupuje pomocou umelých neurónových sietí.

V nasledujúcej kapitole budú predstavené najpoužívanejšie metódy založené na strojovom učení, ktoré sa dnes v tejto oblasti používajú. Priblížené budú zaujímavé časti a fungovanie jednotlivých architektúr. Prvý z možných prístupov je postavený na konvolučne rekurentnej neurónovej sieti[34]. Jej rekurentná časť býva implementovaná pomocou *LSTM*[18][12] alebo *GRU*[7] jednotiek. Následne bude priblížený pristúp *sequence to sequence*[22] s využitím mechanizmu *attention*[4]. Na tomto mechanizme je postavená aj posledná diskutovaná architektúra neurónových sietí, *transformer*[36]. Vychádza z prístupu *sequence to sequence* a pracuje s rozšírenou implementáciou mechanizmu nazývaným *multi-head attention*[36].

Na konci kapitoly je spomenutá Levenshteinova vzdialenosť[27], z ktorej vychádza Levenshteinove zarovnanie, na základe ktorého je možné spočítať metriky ako *Character Error Rate* alebo *Word Error Rate*. Tie sú zakladaným porovnávacím ukazovateľom pri vyhodnocovaní presnosti jednotlivých modelov.

2.1 Konvolučne rekurentné neuronové siete

Neurónová sieť tohto typu[34] sa delí na tri hlavné časti, komponenty. Na úplnom začiatku sa nachádza konvolučná časť, ktorá extrahuje črty z pôvodného obrázka. Rekurentné vrstvy udržujú kontext pre celý vstupný obrázok a vytvárajú predikcie pre každý vektor, ktorý dostali na vstupe. Na samom konci sa nachádza transkripčná časť, ktorá vytvára výslednú predikciu.



Obr. 2.1: Sekvencia vektorov, ktoré sú pomocou konvolučnej časti extrahované zo vstupného obrázka. Každý z nich opisuje črty daného regiónu, ku ktorému prislúcha. Prevzaté z [34]

Konvolučná časť

Konvolučná časť siete pozostáva z konvolučných a *max-pooling* vrstiev, ktorých úlohou je zo vstupného obrázku extrahovať jednotlivé črty. Tieto vrstvy sú prevzaté zo štandartných CNN modelov, jedinou zmenou je odstránenie plne previazaných vrstiev.

Obrázok vstupujúci do siete musí najprv prejsť vhodným *pre-processingom*, ktorého hlavnou a nevyhnutnou častou je daný obrázok upraviť tak, aby mal požadovanú výšku, no pomer jeho strán ostal zachovaný. Vstupný obrázok, predstavujúci riadok textu je rozdelený na jednotlivé regióny, každý z nich je jednotnej fixnej šírky. Z nich konvolučná časť siete postupne sprava doľava vytvorí sekvenciu vektorov, ktoré sú vstupom do rekurentnej časti siete. Tieto vektory nezávisle popisujú jednotlivé časti vstupného obrázku. Čiže každý i-tý vektor prislúcha i-tému výrezu zo vstupu, a teda je jeho deskriptorom[34]. Zobrazené na obrázku 2.1.

Rekurentná časť

Úlohou rekurentnej časti siete je hlavne zachytávanie kontextu zo sekvencie vektorov. Pri rozpoznávaní textu pomocou opisovanej metódy je táto skutočnosť užitočná hlavne pri rozličnej šírke jednotlivých písmen, pretože niektoré úzke písmena môžu byť jednoznačne definované už pri rozbore jedného vektora, obdĺžnika zo vstupného obrázku, zatiaľ čo na úplnú identifikáciu širokých písmen je potrebných týchto vektorov niekoľko. Kontext tak isto napomáha rozpoznávaniu písmen na základe ich výšky voči okolitým písmenám, čo zjednodušuje predikovať triedu v danom okne. Napríklad pri písmenách ako sú „i“ a „l“.

Ďalšou výhodou rekurentnej časti siete je spätná propagácia chyby na vstup, a teda späť na konvolučné vrstvy. To umožňuje spoločné trénovanie ako rekurentných, tak konvolučných vrstiev naraz. Prínosom rekurentnej časti pre daný model je aj schopnosť operovať s ľubovoľnými dĺžkami sekvencií. To znamená, že dĺžka vstupného obrázka pre tento typ sietí nie je limitujúca a celá sekvencia vektorov je prechádzaná od začiatku po koniec.

Bežná rekurentná sieť svoj aktuálny interný stav počíta aj na základe minulého, už prepočítaného vektora, ktorý pri výpočte toho aktuálneho berie na vedomie. Takto postupuje pri práci s každým vektorom. V možných dlhých sekvenciách táto skutočnosť limituje infor-

mácie, ktoré sa v sieti ďalej prenášajú, dokonca môže podstatná informácia úplne zaniknúť. Tento problém je známy pod názvom *vanishing gradient problem*.

Problém úzko súvisiaci s predchádzajúcim je *exploding gradient problem*, pri ktorom jednotlivé hodnoty váh naprieč sieťou nekontrolovateľne rastú a prehlušujú kontext ostatných informácií. Kvôli týmto problémom nebolo do roku 2006, kedy *Hinton*[16] problémy prvýkrát popísal a vyriešil, úplne možné trénovať hlboké rekurentné neuronové siete a tak nedosahovali svoj skutočný potenciál hlavne pri riešení problémov s rozsiahlym kontextom. Dnes sa tieto problémy riešia pomocou *LSTM* a *GRU*.

LSTM

Long Short-Term Memory[18][12] je typ rekurentnej neurónovej siete, ktorá má schopnosť uchovávať kontext v dlhých sekvenciách, a teda potrebnú vlastnosť pri rozpoznávaní ručne písaného textu v celých slovách alebo riadkoch. Jej hlavný mechanizmus tvoria brány. Tie dokážu regulovať tok informácií a učiť sa, ktoré informácie nie sú potrebné a je možné ich zahodiť a ktoré informácie sú pre vytvorenie predikcie nevyhnutné. Nákres popisujúci konštrukciu jednotky je možné vidieť na obrázku 2.2.

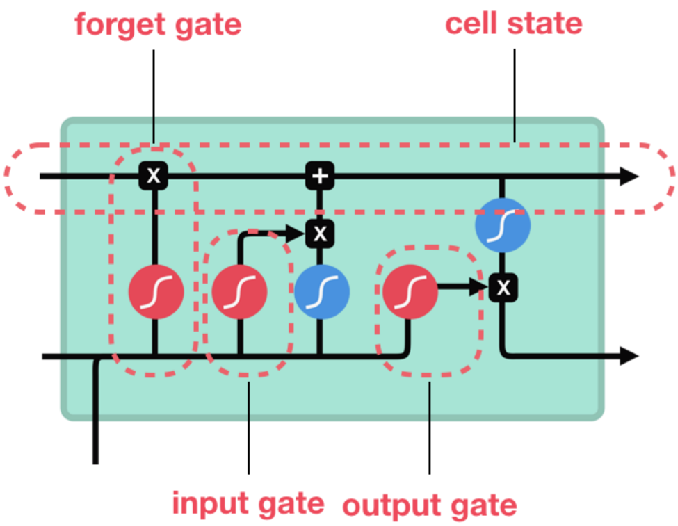
Braný tvoria aktivačné σ funkcie, ktoré vstupné hodnoty pretransformujú na čísla v intervale $\langle 0, 1\rangle$, čím bližšie je výsledné číslo k 0, tým je jeho hodnota menej podstatná, keďže číslo násobené 0 zanikne. Naopak, čím je číslo bližšie k 1, tým má jeho hodnota väčší význam.

Forget gate rozhoduje o tom či bude informácia z predchádzajúceho časového kroku zahodená alebo ponechaná. Určuje tak na základe predchádzajúceho skrytého stavu a aktuálnom vektore na vstupe. Úlohou *input gate* je vyjadriť dôležitosť novej informácie, ktorá sa nachádza na vstupe. Opäť je aplikovaná σ funkcia, čím je zabezpečené rozmedzie výslednej hodnoty v intervale <0,1>. Následne je možné vypočítať *cell state*, ktorý nesie podstatné informácie z celého vstupu po celý čas výpočtu. Učiní tak postupným násobením predchádzajúceho *cell state* s výsledkom *forget gate* a následným pričítaním výstupu z *input gate*. Výsledkom týchto operácii je nový *cell state*. Posledný na rad prichádza *output gate*, ktorý vytvára nový skrytý stav, ktorý je obohatený o informácie z aktuálneho vstupu. Táto operácia opäť spočíva v znásobení, tentokrát výstupu z *output gate* s novým, pravé vytvoreným *cell state*.

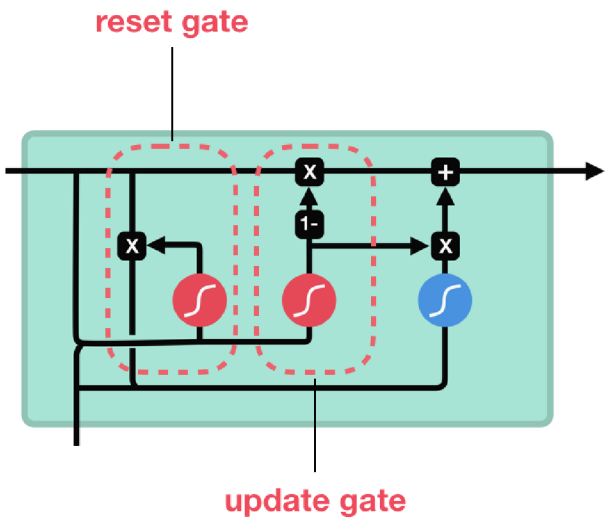
GRU

Gated Recurrent Unit[7] je obnovou *LSTM* a funguje veľmi podobne. Namiesto používania *cell state* ako dlhodobej pamäte využíva svoj skrytý stav, v ktorom dokáže držať kontext dlhú dobu. Keďže vychádza z *LSTM* tiež funguje na princípe spomínaných brán, ktoré sú vo svojej podstate implementované rovnako, teda ako σ funkcie. Tieto brány obsahuje iba dve *reset gate* a *update gate*. *Update gate* funguje podobne ako *forget gate* a *input gate* z *LSTM*, rozhoduje o tom, ktoré informácie zo vstupu majú byť zahodené a ktoré pridané do skrytého stavu. *Reset gate* sa stará o správu skrytého stavu, a teda ktoré informácie z neho majú byť pri prechode jednotkou zahodené a ktoré ponechané do ďalšej iterácie.

GRU je o niečo jednoduchšia ako *LSTM* a pri svojej činnosti s vektormi si vystačí s menej operáciami. Pri trénovaní väčšiny rekurentných neurónových sietí to tak z nej robí rýchlejšiu jednotku. Konštrukciu danej jednotky je možné vidieť na obrázku 2.3.



Obr. 2.2: Obrázok popisuje jednotlivé komponenty *LSTM*, naznačuje spojenia medzi nimi a to, akými matematickými operáciami sú implementované. Tak isto zobrazuje usporiadanie jednotlivých brán a to, v akom poradí si predávajú informácie. Prevzaté z [33].



Obr. 2.3: Obrázok popisuje jednotlivé komponenty *GRU*. Usporiadanie brán a operácie, ktorými sú ich výstupy modifikované. Už z náhľadu je možné vidieť jednoduchšiu konštrukciu celej jednotky v porovnaní s jednotkou *LSTM*. Prevzaté z [33].

Transkripčná časť

Proces transkripcie vytvára finálnu predikciu celého slova alebo riadku v závislosti od požiadavkov na model. Robí tak pretváraním výstupov rekurentnej časti siete, a teda jej predikcií pre každý vektor, inak povedané pre každú vysegmentovanú časť obrázka. Transkripcia môže výslednú sekvenciu produkovať čisto na základe predikcií vychádzajúcich z rekurentnej časti siete, bez ohľadu na výsledne slovo, ktoré vytvorí. Takýto prístup sa nazýva *lexicon-free*. Druhý prístup sa nazýva *lexicon-based*, ktorý je rozšírený o takzvaný slovník slovnej zásoby. Predikcia konkrétneho slova je vybraná z tohto slovníka na základe najväčšej podobnosti sekvencie získaného neurónovou sieťou zo vstupu s daným slovom. Avšak pri slovníkoch veľkých rozmerov by porovnávanie každého slova bolo časovo príliš náročné, preto sa od prvotnej sekvencie hľadajú takzvaný najbližší susedia. Množina týchto slov je určená na základe maximálnej Levenshteinovej vzdialenosti[27] každého z nich od pôvodne predikovaného slova. Toto je umožnené vďaka spôsobu uloženia slovníku v dátovej štruktúre BK stromu, z ktorého sa následne vyberie finálny kandidát, ktorého vzdialenosť od pôvodnej predikcie je spomedzi všetkých najmenšia.

CTC

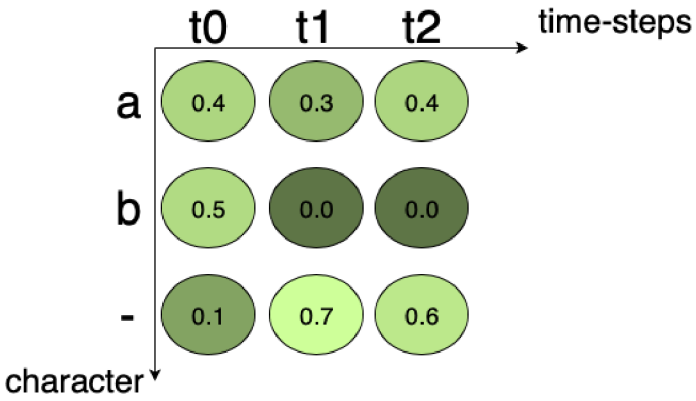
Connectionist temporal classification[13] je objektívna funkcia, ktorá umožňuje trénovanie rekurentných neuronových sietí bez nutnosti predom segmentovať vstupné dáta. Tak isto rieši dekodovanie výsledku rekurentnej neurónovej siete. Nie je tak potrebne mat klasifikovanu triedu, *ground truth*, pre kazdy vektor, pretoze CTC tento vystup patricne zarovnava. Základnou myšlienkou je interpretovať výstup z rekurentnej časti siete ako rozdelenie pravdepodobnosti všetkých možných sekvencií, ktoré zodpovedajú vstupnej sekvencii znakov na obrázku. Práve na základe tohto rozdelenia sa dá odvodiť potrebná objektívna funkcia, ktorá maximalizuje pravdepodobnosť určenia správneho označenia. Táto funkcia je diferenciovateľná, a teda sieť môže byť trénovaná štandardnou spätnou propagáciou. Formálne je zapísaná nasledovne:

$$p(\pi|\mathbf{x}) = \prod_{t=1}^{T} y_{\pi_t}^t, \forall \pi \in L'^T \tag{2.1}$$

Vstupnou sekvenciou je $\mathbf{x}$, jej dĺžku udáva parameter T. y_k^t je pravdepodobnosť výskytu znaku k v čase t. Znak k je znakom používanej abecedy L'^T, ktorá je zjednotením všetkých znakov, ktoré sa môžu na vstupe objaviť a zároveň prázdnym znakom, takzvaným *blank*. Matematicky zapísané ako $L' = L \cup \{blank\}$. Jednotlivé elementy abecedy L'^T sa nazývajú cesty a su značené sú ako π.

V ďalšom kroku je potrebné definovať mapu typu *many-to-one* $\mathcal{B} : L'^T \mapsto L^{\leq T}$, kde $L^{\leq T}$ je množinou možných značení pre danú vstupnú sekvenciu. V jednoduchosti je tento proces vykonávaný odstránením všetkých prázdnych znakov *blank* a opakujúcich sa znakov. Názorná ukážka, $\mathcal{B}$(a–ab–) = $\mathcal{B}$(–aa—abb) = aab. Nakoniec je $\mathcal{B}$ použité na definovanie podmienenej pravdepodobnosti daného značenia $\mathbf{l} \in L^{\leq T}$ ako súčet pravdepodobností všetkých ciest, ktoré mu zodpovedajú. Príklad výpočtu je znázornený na obrázku 2.4. Formálne zapísané nasledovným spôsobom:

$$p(\mathbf{l}|\mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} p(\pi|\mathbf{x}) \tag{2.2}$$



Obr. 2.4: Výpočet $p(\mathbf{l}|\mathbf{x})$ by vyzeral nasledovne. Uvedený príklad je znázornený nad vstupnou abecedou $L'^T = \{a, b, -\}$ s 3 časovými krokmi. Reťazec na vstupe bude „a“, ktorý je zároveň aj takzvaným *ground truth*. Všetky možné cesty pre „a“ v tomto príklade sú „aaa“, „a–“, „-a-“, „–a“, „aa-“, „-aa“. Sčítaním skóre jednotlivých ciest, ktoré dostaneme, je 0.048 + 0.168 + 0.018 + 0.072 + 0.012 + 0.028 = 0.346. 0.346 je pravdepodobnosť výskytu *ground truth*. Z tejto hodnoty je vypočítaný logaritmus, ktorý je následne negovaný a výsledkom je takzvaná hodnota *loss*, ktorá môže byť spätne propagovaná a neurónová sieť môže byť trénovaná. Prevzaté z [5].

Z danej formulácie vyplýva, že výstup klasifikátora by mal byť najpravdepodobnejším značením vstupnej sekvencie. Zapísané nasledovne:

$$h(\mathbf{x}) = \arg\max_{\mathbf{l} \in L^{\leq T}} p(\mathbf{l}|\mathbf{x}) \tag{2.3}$$

2.2 Sequence to sequence

Je špeciálnym typom rekurentných neurónových sietí, ktoré dokážu mapovať vstupnú sekvenciu na výstupnú, ktorej dĺžka je vopred neznáma. Pravé kvôli tejto vlastnosti býva jej architektúra aplikovaná hlavne na riešenie zložitých jazykových problémov, akým je napríklad *machine translation*, čiže preklad sekvencie textu z jedného jazyka do sekvencie textu druhého jazyka. Keďže práve pri preklade dlhej sekvencie znakov, vety v slovenskom jazyku, môže byť výstupom úplne iná sekvencia, a teda aj iný počet znakov v inom, cudzom jazyku. Okrem iného je využívaná tiež na *text summarization*, čo znamená, že dokáže sumarizovať podstatu textu. Z toho vyplýva široká škála uplatnenia ako je napríklad vytváranie četovacích robotov, ktorý dokážu napríklad automaticky odpovedať na otázky. Tiež býva používaná na rozpoznávanie reči[35]. Dnes je tento pristúp vo veľkom využívaný aj v OCR modeloch na rozpoznávanie textu[22].

Attention

Kľúčovú úlohu v *sequence to sequence* modeloch zohráva takzvaný *attention* mechanizmus, ktorý predstavil v roku 2014 *Bahdanau a spol*[4]. Vyriešil problém použitia kódovacieho vektora fixnej dĺžky, kde mal *encoder* len obmedzený pristúp k informáciám zo vstupu[8].

Attention mechanizmus využíva tri hlavné komponenty, ktoré sa nazývajú *queries* Q, *keys* K, *values* V a krok po kroku vykonáva nasledovné výpočty:

Každý *query* vektor $q = s_{t-1}$ sa porovnáva s databázou *keys* pre výpočet daného skóre. Táto operácia zhody sa vypočíta ako skalárny súčin konkrétneho uvažovaného *query* s každým *key* vektorom k_i

$$e_{q,k_i} = q \cdot k_i \tag{2.4}$$

Výsledky sú spracovane operáciou *softmax*, ktorá vygeneruje váhy

$$\alpha_{q,k_i} = \text{softmax}(e_{q,k_i}) \tag{2.5}$$

Attention je následne vypočítaný váženým súčtom *value* vektorov v_{k_i}, kde každý z nich je spárovaný s príslušným key

$$\text{attention}(q, K, V) = \sum_i \alpha_{q,k_i} v_{k_i} \tag{2.6}$$

V podstate, keď *attention* mechanizmus pracuje so sekvenciou slov, vezme *query* vektor priradený konkrétnemu slovu v sekvencii a porovná ho s každým *key* v databáze. Pritom zachytáva, ako uvažované slovo súvisí s ostatnými vo vstupnej sekvencii. Potom zoraďuje *values* podľa váh vypočítaných zo skóre tak, aby sa zachovalo zameranie na tie slová, ktoré sú relevantné pre danú *query*. Pritom vytvára *attention* výstup pre uvažované slovo[8].

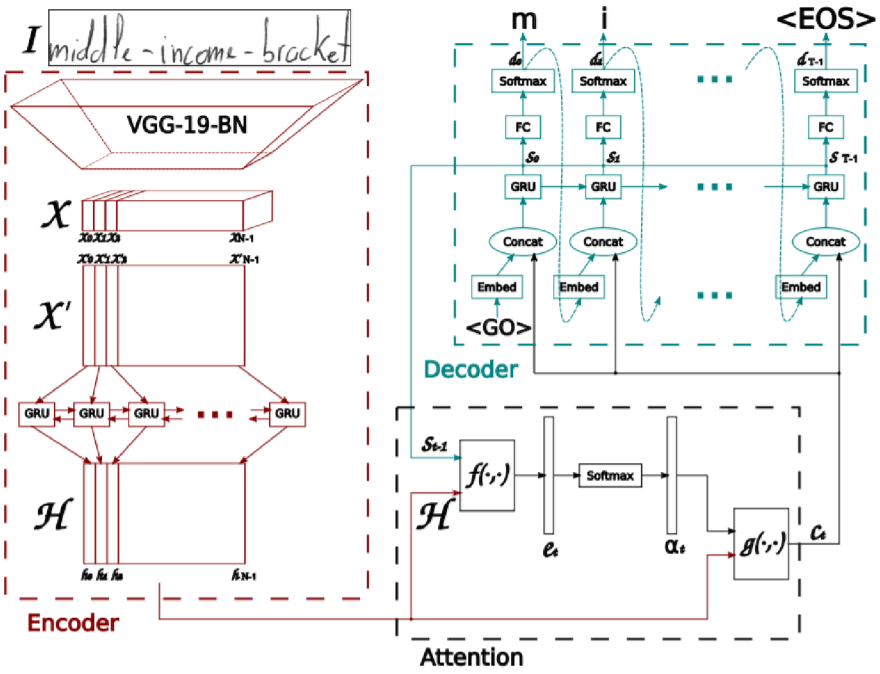
Metóda, ktorú predstavil *Kang a spol* v roku 2019 v práci [22], je postavená práve na modeli *sequence to sequence* s využitím *attention* mechanizmu. Ich model sa skladá z troch hlavných častí a to *encoder*, *attention* mechanizmus a *decoder*. Architektúru daného modelu je možné vidieť na obrázku 2.5.

Encoder

Na začiatku kodéra sa nachádza konvolučná neurónová sieť, ktorej hlavnou úlohou je extrakcia črtou zo vstupného obrázka. Vyextrahované dáta sú vstupom do *Bi-directional Gated Recurrent Unit*, ktorá je viacvrstvová, slúži na zachytávanie celkového kontextu a pridáva informácie o pozícii každého okna. Tak isto ako je možné vidieť aj na obrázku 2.5 vytvára výstup kodéra a zakóduje výslednú sekvenciu. Funkcionalita kodéra, a teda kombinácie CNN a BGRU spočíva v zakódovaní textu zo vstupného obrázku $\mathcal{I}$ do mapy vektorov $\mathcal{X}$. Táto je následne transformovaná do dvojrozmerného poľa $\mathcal{X}'$, ktoré obsahuje vyextrahované črty pre jednotlivé časové kroky. Vektory tohto poľa vstupujú do BGRU, ktorého výstupom je opäť dvojrozmerné pole vektorov $\mathcal{H}$ s rovnakou dĺžkou, ktoré obsahuje pridané informácie a z ktorého bude počítaný *attention*.

Content-based Attention

Implementácia mechanizmu *Content-based Attention* slúži modelu na hľadanie podobnosti medzi aktuálnym skrytým stavom dekodéru a črtami vstupného obrázka, ktoré sú zároveň výstupmi kodéra. Na základe toho, ktoré s vektorom najviac korelujú je mechanizmus schopný určiť príslušný znak pre daný časový krok vstupného obrázka. Formálne zapísané nasledovne:



Obr. 2.5: Architektúra modelu, ktorý predstavil *Kang a spol*, zobrazuje hlavné použité komponenty a ich vzájomné prepojenie. Prevzaté z [22].

$$\alpha_t = \text{Softmax}(e_t) \tag{2.7}$$

$$e_{t,i} = f(h_i, s_{t-1}) = w^T tanh(Wh_i + Vs_{t-1} + b) \tag{2.8}$$

Definovaná je *attention mask* α_t vektora v časovom kroku t. h_i je skrytý stav kodéra v časovom kroku $i \in \{0, 1, ..., N-1\}$. s_t je skrytý stav dekodéra v časovom kroku $t \in \{0, 1, ..., T-1\}$, kde T je maximálna dĺžka dekódovaných znakov. w, W, V a b sú natrénovateľné parametre. Po vypočítaní *attention mask* vektora, je možné vypočítať vektor s najdôležitejším kontextom nasledovne:

$$c_t = g(\alpha_t, H) = \sum_{i=0}^{N-1} \alpha_{ti} h_i \tag{2.9}$$

Location-based Attention

Location-based Attention mechanizmus je do modelu zakomponovaný pre odstránenia nevýhody, ktorou trpí vyššie uvedený *Content-based Attention* mechanizmus. Ten na detekovanie odlišností medzi viacerými reprezentáciami toho istého črtu zo vstupného obrázka, v tomto prípade písmena, potrebuje mať zakódované pozičné informácii o týchto prvkoch od kodéra. Predchádzajúci mechanizmus bol preto rozšírený o zohľadňovanie pozície na základe zarovnania vytvoreného v predchádzajúcom kroku a implementovaný je nasledovne:

$$l_t = F * \alpha_{t-1} \tag{2.10}$$

Na začiatku sa vyextrahuje počet k vektorov $l_{t,i} \in \mathbb{R}^k$ pre každú pozíciu i z predchádzajúceho zarovnania α_{t-1} a to konvolúciou s maticou $F \in \mathbb{R}^{k \times r}$. Následne je rovnica číslo 2.8 nahradená nasledujúcou rovnicou:

$$e_{t,i} = f'(h_i, s_{t-1}, l_t) = w^T tanh(Wh_i + Vs_{t-1} + Ul_{t,i} + b) \tag{2.11}$$

Kde sú w, W, V, U a b opäť trénovateľné parametre.

Attention Smoothing

Vo svojej práci ešte *Kang a spol* odporúča nahradiť *Softmax* funkciu 2.7 za σ funkciu 2.12, ktorá má za následok zachytávanie o trochu širšieho okolia okolo písmena na vstupnom obrázku. Konečný výsledok rozpoznávania daného znaku je síce rovnaký, ale z ľudského pohľadu je takto segmentovaný znak lepšie rozpoznateľný a v budúcnosti by sa pravé z tejto skutočnosti mohlo benefitovať.

$$\alpha_{t,i} = \frac{\sigma(e_{t,i})}{\sum_{i=0}^{N} \sigma(e_{t,i})} \tag{2.12}$$

Decoder

Dekodér je posledným komponentom celého modelu, ktorý je vo svojej podstate ďalšou implementáciou rekurentnej neurónovej siete. Dekóduje znaky pre každý vektor, ktorých spojenie vedie k vytvoreniu výslednej sekvencie, prepisu textu zo vstupného obrázku. Implementovaný je pomocou viacvrstvových jednosmerných GRU. Vstupom do týchto jednotiek je vždy príslušný vektor v časovom kroku t, ktorý vznikol zreťazením *embedding* vektora predchádzajúceho časového kroku $\tilde{y}_{t-1}$ a vektora c_t nesúceho kontextové informácie. Predikcia pre každý časový krok teda vyzerá nasledovne:

$$y_t = \arg\max(\omega(s_t)) \tag{2.13}$$

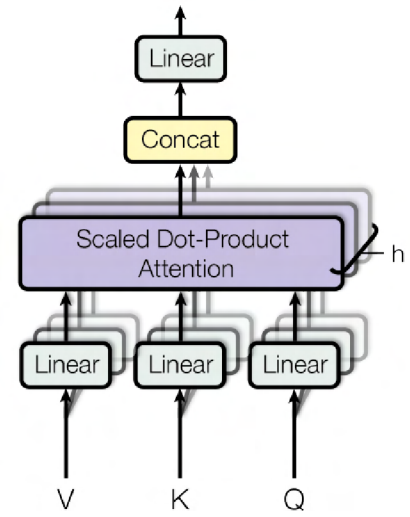
Kde $\omega(.)$ je lineárnou vrstvou. *Embedding* vektor vracia matica, takzvaná *look-up table*, ktorá je na začiatku inicializovaná náhodnými hodnotami a počas procesu trénovania sa aktualizuje.

$$\tilde{y}_t = \text{Embedding}(y_t) \tag{2.14}$$

Dekodér vždy začne po načítaní štartovacieho signálu, ktorým je znak $\langle GO \rangle$. Proces dekódovania končí po načítaní znaku $\langle EOS \rangle$ alebo pokiaľ dekodér nenačíta maximálny počet časových krokov T.

Skrytý stav dekodéra v aktuálnom časovom kroku s_t je výsledkom konkatenácie aktuálneho kontextového vektora a predchádzajúceho vstupného vektora. GRU z tohto ťaží a v každom časovom kroku čerpá z oboch informácii, čo jej pomáha vytvoriť správnu predikciu. Formálne je to zapísané nasledovne:

$$s_t = \text{Decoder}([\tilde{y}_{t-1}, c_t], s_{t-1}) \tag{2.15}$$



Obr. 2.6: Architektúra mechanizmu *Multi-Head Attention*, ktorá sa skladá z niekoľkých *attention* vrstiev pracujúcich paralelne. Prevzaté z [36]

kde $[\,\cdot\,,\,\cdot\,]$ značí konkatenáciu.

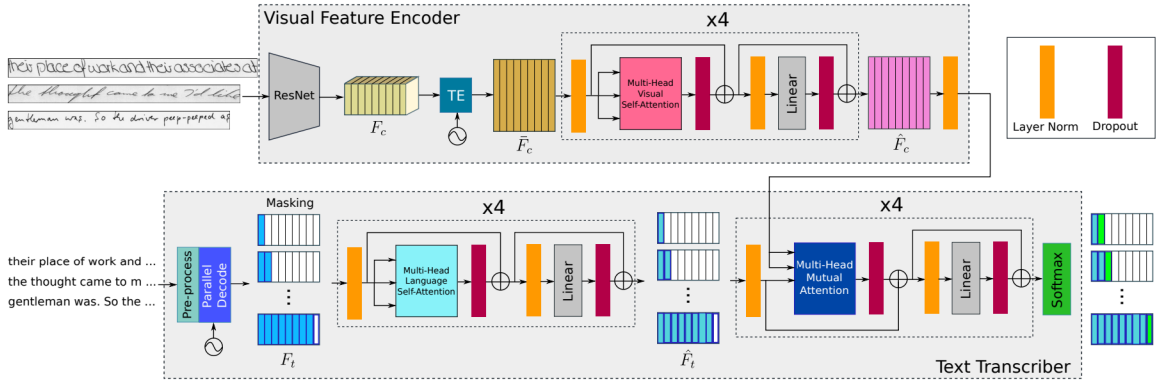
2.3 Transformer

Architektúra *transformer* je modelom neurónových sietí, ktorá si ku svojej činnosti adaptovala mechanizmus *attention* a všeobecne funguje bez použitia konvolučných alebo rekurentných častí. Svoje uplatnenie našla predovšetkým na poli *natural language processing* a *computer vision*, čo bolo pri vývoji danej architektúry aj jej hlavným cieľom. Kľúčovou vlastnosťou, práve pri riešení týchto problémov, je schopnosť zakódovať dlhodobé závislosti vo veľkých sekvenciách[32] a to aj kvôli tomu, že oproti rekurentným neurónovým sieťam netrpí problémom explodujúceho a strácajúceho sa gradientu.

V roku 2017 *Vaswani a spol* predstavili novú jednoduchú architektúru siete *transformer*[36], ktorá funguje výhradne s použitím *attention* mechanizmu bez zahrnutia rekurentných alebo konvolučných častí siete. Rekurentné modely si pre uchovanie kontextu zo vstupnej sekvencie po dobu svojej činnosti prenášajú informáciu v skrytom stave, ktorý je prepočítavaný pre každý časový krok, čo vylučuje možnosť paralelizácie. Táto architektúra však je jednoducho paralelizovateľná, a teda časová náročnosť pri trénovaní je značne znížená[36].

Multi-Head Attention

Pôvodný návrh architektúry *transformer*, ktorý uviedli *Vaswani a spol*[36] namiesto práce s jednou *attention* funkciou, ktorá operuje len s jednou maticou skladajúcou sa z *keys*, *values* a *queries*, použili týchto matíc hneď niekoľko. Všetky sú spracovávané paralelne a výsledné matice sú konkatenované do finálnej matice, ktorá obsahuje konečné hodnoty.



Obr. 2.7: Architektúra modelu, založeného na metóde transformer, ktorý predstavil *Kang a spol*, zobrazuje hlavné časti architektúry a ich vzájomné prepojenie. Prevzaté z [21].

Na problematiku rozpoznávania ručne písaného písma ako prvý metódu *transformer* použil *Kang a spol*[21]. Ich model sa skladá z dvoch hlavných komponentov, tak ako je vidieť na obrázku 2.7. *Visual Feature Encoder*, tak ako vyplýva z jeho názvu, funguje ako *encoder* a zo vstupného riadku v podobe obrázku extrahuje podstatné črty a pomocou *attention* mechanizmu sa zameriava na rozličné pozície vo vstupe. *Text Transcriber* je implementáciou *decoderu* a zameriava sa na vytváranie výslednej sekvencie znakov. Tú vytvára na základe ako vizuálnych, tak aj jazykových vlastností vstupu a pri trénovaní sa zároveň učí tiež komplexne dešifrovať písaný text spolu s modelovacím jazykom[21].

Visual Feature Encoder

Samotný kodér sa ešte člení na tri jednotlivé časti. Na úplnom začiatku sa nachádza *CNN*, ktorá zo vstupného riadku ľubovolnej dĺžky vytvorí sekvenciu vektorov F_c o veľkosti f, v ktorých sú zahrnuté vizuálne črty spolu s kontextovou informáciou. Tá poskytuje globálny pohľad na celý vstupný obrázok.

Výstup konvolučnej siete je vstupom do takzvaného *Temporal Encoding*, ktorého úlohou je zvýrazniť dôležité informácie bez použitia akejkoľvek rekurencie. V prirodzenom texte sa nachádzajú rovnaké znaky na viacerých miestach, no pre správne fungovanie *attention* mechanizmu je potrebne aby sa jednotlive vektory, ktoré ich popisujú líšili, to sa dá dosiahnuť stratou invariancie horizontálneho posunu. Preto je na základe návrhu v práci *Attention Is All You Need*[36] aplikované jednorozmerné pozičné kódovanie pomocou sínusových a kosínusových funkcií[21].

V poslednom kroku, ktorý je obsiahnutý v *Visual Self-Attention Module* je kladený ešte väčší dôraz na vyzdvihnutie najpodstatnejších vizuálnych čŕt. To je dosiahnuté použitím *multi-head attention* mechanizmu z práce *Attention Is All You Need*[36]. Súhrn najdôležitejších vizuálnych čŕt je na konci vytvorený funkciou *softmax*.

Text Transcriber

Na rozdiel od bežného dekodéru, ktorý sa používa napríklad v modeloch na prevod reči do textu, ktorý funguje na úrovni slov, tento je implementovaný na báze predikcie jednotlivých

znakov. Predikcia je teda vytváraná na základe už detekovaných písmen pre každý znak osobitne. Tak isto ako kodér, sa aj dekodér člení na tri po sebe nasledujúce logické časti.

V prvom module *Text Encoding* je okrem znakov zo vstupnej abecedy A potrebná definícia niekoľkých špeciálnych znakov. Sú to: $\langle S \rangle$ na označenie začiatku vstupnej sekvencie, $\langle E \rangle$ na označenie konca vstupnej sekvencie a $\langle P \rangle$ ako prázdny znak. Samotné vkladanie predikovaných znakov sa vykonáva pomocou plne prepojenej vrstvy, ktorá mapuje každý znak zo vstupného reťazca na f-rozmerný vektor. Na mapovanie sa opäť využíva *Temporal Encoding* podobne ako na začiatku. V prístupoch, ktoré využívajú rekurentné neuronové siete je pri dekódovaní každý znak poslaný späť do rekurentnej časti kvôli potrebe dekódovania nasledujúceho znaku, čo znemožňuje spomínanú paralelizáciu[21]. V architektúre *transformer*, sú ale všetky možné kroky spracovávané súčasne s takzvanou maskou[36]. Pri potrebe dekódovania j-tého znaku, táto maska zabraňuje dekódovaniu znakov na pozíciach väčších ako j. Čiže dekódovanie závisí len od doteraz známych predikcií vytvorených pred dekódovaním samotného znaku na pozícií j.

Druhý modul je nazvaný *Language Self-attention Module*, opäť využíva *multi-head attention* mechanizmy a jeho cieľom je ponechať len najdôležitejšie informácie z textu a ďalej sa zamerať na učenie vlastností špecifických pre daný jazyk.

Posledný modul *Mutual-attention Module* pracuje zároveň s vizuálnymi informáciami z obrázka a s terminológiou daného jazyka. Opäť je obdobou implementácie *multi-head attention* mechanizmu a očakáva sa, že jeho výstup bude zarovnaný s transkripciou, ktorá je výstupom kodéru. Konečná kombinovaná reprezentácia je privedená do lineárneho modulu, po ktorom nasleduje aktivačná funkcia *softmax*, ktorej činnosťou sa získa finálna predpoveď.

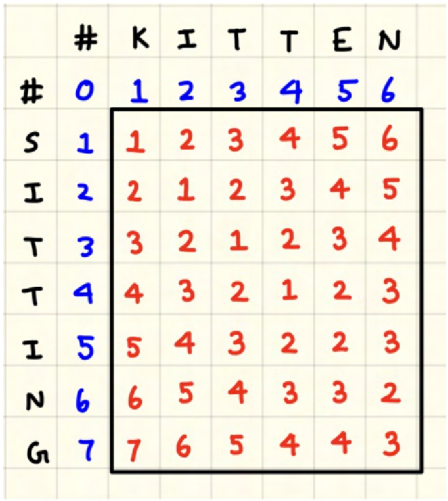
2.4 Levenshteinova vzdialenosť

V roku 1965 Vladimir Levenshtein v práci *Binary codes with correction of dropouts, insertions and substitutions of characters*[27] stanovil novú metriku na porovnávanie dvoch sekvencií. V dnešnej dobe sa využíva hlavne na porovnávanie reťazcov. Výsledkom porovnávania je číslo, ktoré sa nazýva Levenshteinova vzdialenosť a udáva ako veľmi sa dva reťazce navzájom líšia. V praxi je to minimálny počet prvkov, ktoré je potrebné zmeniť, doplniť alebo odstrániť z jednej sekvencie, aby sa rovnala druhej. Formálne zapísané nasledovne:

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{ak } \min(i,j) = 0, \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j) + 1 \\ \mathrm{lev}_{a,b}(i,j-1) + 1 \\ \mathrm{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{inak.} \end{cases} \tag{2.16}$$

kde a a b sú porovnávane reťazce, i a j sú pozície práve porovnávaných znakov v ich pôvodnom reťazci. Pokiaľ sú práve porovnávané znaky v reťazcoch totožné, Levenshteinova vzdialenosť ostáva nezmenená. Pokiaľ sa líšia, je inkrementovaná o jeden.

Najpriamočiarejšie vysvetlenie rovnice 2.16 je pomocou matíc. Znázornené na obrázku 2.8.



Obr. 2.8: Matica sa vždy začína vypĺňať od ľavého horného rohu s pokračovaním po riadkoch. Modré číslice pod jednotlivými porovnávanými písmenami značia ich indexy, pri porovnávaní dopĺňané za i a j vo východzej rovnici. Konečnú Levensthteinovu vzdialenosť udáva číslo, ktoré sa po vyplnení matice objaví na spodnom pravom rohu. V tomto prípade je Levenshteinova vzdialenosť medzi slovami **kitten** a **sitting** 3. Prevzaté z [30].

Levenshteinovo zarovnanie

Je to predovšetkým zápis vychádzajúci z rovnomennej vzdialenosti, slúži na opis atomických operácií, ktoré boli vykonané počas mapovania prvého reťazca na druhý. Pokiaľ je transformovaný reťazec dlhší a je potrebné zbaviť sa niektorého znaku, tak je operácia zapísaná špeciálnym znakom, ktorý sa zvolí na začiatku, typicky „-“. Pri mapovaní reťazca **sitting** na reťazec **kitten**, sa reťazec zarovná ako **kitten-**.

Kapitola 3

Prehľad dátových sád

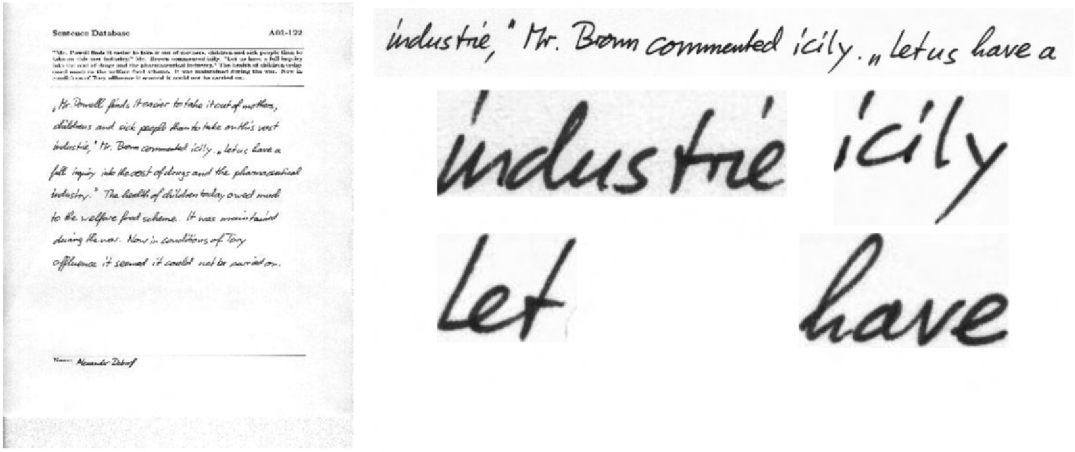
Kapitola poskytuje prierez najpoužívanejšími dátovými sadami, ktoré umožňujú tréning a následné vyhodnocovanie presnosti neurónových sietí na rozpoznávanie ručne písaného textu. Hlavná kategorizácia takýchto dátových sad je podľa typu OCR, ktoré má byť natrénované, a teda či bude jeho účelom rozpoznávať *offline* alebo *online* písaný text. V závislosti od požiadaviek na konkrétny model a jeho použitie, je potrebné vybrať vhodnú dátovú sadu. Tie sa môžu líšiť v mnohých aspektoch, od druhu použitého písma, obdobia jeho výskytu, až po jazyk, v ktorom je dané písmo zaznačené. Každá sada sa vyznačuje hlavne svojim zameraním, ktoré je dané stupňom anotácie jednotlivých prvkov. Existujú dátové sady zamerané na samostatné znaky alebo číslice, ktoré sú vo veľkom používane napríklad pri trénovaní a testovaní systémov rozpoznávajúcich poštové smerovacie čísla na poštách. Ďalšou skupinou sú dátové sady, ktorých obsah pozostáva z vyseparovaných slov, viet, riadkov alebo celých blokov textu. Podľa tohto rozdelenia sú väčšinou aj anotované. Ostatné faktory, ktoré tiež ovplyvňujú výsledný natrénovaný model je veľkosť, kvalita, rozlíšenie a konzistencia obrázkov vybraného datasetu. Prehľad spomenutých dátových sád sa nachádza v tabuľke 3.1.

IAM-DB

Dátová sada *IAM Handwriting Database*[29][20] bola vytvorená prepismi formulárov do písaného textu, ktorý bol následne naskenovaný s rozlíšením 300 dpi a uložený vo formáte PNG s 256 úrovňami šedi. Ukážka tohto formuláru sa nachádza na obrázku 3.1. Dané formuláre boli vytvorené a pochádzajú z korpusu britskej angličtiny *Lancaster-Oslo/Bergen*. Celá sada má rozsah 1 539 strán, ktoré obsahujú 5 685 viet, 13 353 riadkov textu a 115 320 slov. Svoje uplatnenie nenachádza len v rozpoznávaní textu, ale tak isto aj v identifikácií a verifikácií potencionálneho pisateľa.

IAM-OnDB

Dátová sada *IAM On-Line Handwriting Database*[28][20] bola inšpirovaná *IAM Handwriting Database*. Na tvorbe sa podieľalo 221 rôznych pisateľov, ktorí dokopy napísali 13 049 riadkov textu, pozostávajúcich z 86 272 slov. Približne rovnaké zastúpenie majú obe pohlavia a okolo 10% respondentov bolo ľavákov. Všetky dáta boli zozbierané pomocou systému *E-Beam*, ktorý dokáže premeniť na interaktívny displej akúkoľvek štandardnú tabuľu.



Obr. 3.1: Ukážka formuláru s ručným prepisom, ktoré dokopy tvoria základ dátovej sady*IAM-DB*. Zobrazené sú tiež už vysegmentované riadky a slová. Prevzaté z [1].

Zhromaždené údaje boli následne uložené vo formáte XML, vrátane ID pisateľa a použitého nastavenia nahrávania. Ku každému pisateľovi bolo okrem iného uložené aj jeho pohlavie, rodný jazyk a niektoré ďalšie faktory, ktoré by mohli byť užitočné pre potencionálnu budúcu analýzu.

IAM-HistDB

IAM Historical Document Database[2] je dátová sada, ktorá vznikla spojením troch rozličných databáz. Prvou z nich je *Saint Gall Database*[10], ktorá obsahuje latinské texty z 9. storočia, ktoré napísal jediný autor a to s použitím atramentu na pergamen. Pozostáva zo 49 listov, ktoré sa nachádzajú na 60 stranách, dokopy tvoria 1 410 riadkov a napísaných je celkovo 11 597 slov. Originál týchto textov je uschovaný v *Abbey Library of Saint Gall* vo Švajčiarsku. Druhou z nich je *Parzival Database*[11], jej náplň tvorí gotické písmo z 13. storočia, opäť je napísaná atramentom na pergamen a to v stredovekej nemčine. Má rozsah 47 strán, ktorých obsahom je rovnomenná epická báseň z toho istého obdobia. Dielo spísala trojica pisateľov. Databázu dokopy tvorí 4 477 riadkov textu a 23 478 slov, ktoré sú značené 93 rôznymi písmenami. Rovnako ako prvé spomínané dielo aj tento originálny prepis je uložený v *Abbey Library of Saint Gall* vo Švajčiarsku. Treťou a poslednou z nich je *Washington Database*[11], ktorá je vytvorená z 82 dopisov z 18. storočia, ktorých autormi sú George Washington a Thomas Jefferson. Tieto listy sú písane atramentom na papier v anglickom jazyku a ich rozsah je približne 20 strán, 656 riadkov textu, 4 894 slov a obsahujú 82 rôznych písmen.

HKR

Dátová sada *Handwritten Kazakh and Russian Database for Text Recognition*[31] je vytvorená z azbuky, 95% z nej tvorí ruština a ostávajúcich 5% tvoria slová a vety z kazaštiny. Pozostáva z formulárov generovaných *LATEX*-om, ktoré vyplnilo približne 200 rôznych autorov. Každý z nich vyplnil týchto formulárov 5 až 10, čiže celkovo ich bolo zozbieraných

vyše 1400. Nachádzajú sa v nich ručné prepisy kľúčových slov kazaštiny a ruštiny, názvy miest a obcí, ručne písané básničky a prepisy azbuky. Celá sada tak obsahuje viac ako 63 000 viet a viac ako 715 000 symbolov, ktoré sú utvorené z 33 rozličných písmen azbuky a doplnkových 9 špecifických znakov typických pre kazaštinu.

HIT-OR3C

Dátová sada *Harbin Institute of Technology Opening Recognition Corpus for Chinese Characters*[38] je zbierkou čínskych rukopisov, ktoré boli zozbierané pomocou tabletu a následne automaticky anotované softwarom *OR3C Toolkit*. Z tohto dôsledku je sada vhodná aj na trénovanie *on-line* OCR modelov. Celá sada sa člení do 5 častí: *GB1*, *GB2*, *Letter*, *Digit* a *Document*. Na prvých 4 častiach sa podieľalo 122 pisateľov a spolu vytvorili 832 650 vzoriek písaných čínskych znakov a číslic, ktoré spadajú do 6 825 rôznych tried. Piata časť s názvom *Document* je vytvorená 20 pisateľmi, pozostáva z prepísaných novinových článkov a jej rozsah je 77 168 vzoriek, spadajúcich do 2 442 rôznych tried.

NIST

Dátová sada *NIST Special Database 19*[15] je zložená z vyše 810 000 obrázkov znakov, ktoré vytvorilo viac ako 3600 pisateľov. Obsahuje separované číslice a písmená malej a veľkej latinskej abecedy. Každý obrázok ma veľkosť 128 x 128 pixelov a je zaradený do jednej zo 62 tried, a teda “0”-“9”, “A”-“Z”, “a”-“z”. Skutočnosť, že celá sada a zaradenie jednotlivých obrázkov do tried bolo ručne skontrolované, dáva výslednú chybovosť sady okolo 0.1% CER.

RIMES

Dátová sada *Recognition & Indexing of handwritten documents & faxes*[14] bola vytvorená dobrovoľníkmi, ktorí boli požiadaní, aby napísali ručne písané listy vo francúzštine výmenou za darčekové poukážky. Dobrovoľníkom bola pridelená fiktívna identita z 5 rôznych kategórií a scenár listu bol vybraný spomedzi tém, ako sú napríklad: zmena osobných údajov, žiadosť o nejaké informácie, otvorenie a zrušenie napríklad zákazníckeho účtu, úprava zmluvy alebo objednávky, sťažnosť, ťažkosti s platbou, upomienka, prehlásenie o škode a podobne. Dobrovoľníci mali list napísať vlastnými slovami a jeho celková štylizácia bola tiež ponechaná na nich, jedinou úlohou bolo držať sa dopredu vybranej témy. Na zbierke sa podieľalo viac ako 1 300 ľudí, niektorí z nich napísali až 5 listov. Dátová sada tým pádom dosiahla obsah až 12 723 strán, čo zodpovedá 5 605 listov s dvomi až tromi stranami. V celej databáze je napísaných viac ako 300 000 slov.

PERO Handwriting Dataset

Komplexná dátová sada *PERO Handwriting Dataset* je zložená z riadkov ručne písaného písma, z ktorých je 629 558 v trénovacej sade a 5762 v testovacej sade. Skladá sa dokopy zo siedmich nezávislých častí, ktoré sú z rôzneho obdobia a zároveň písane v rôznych jazykoch. Prvou je dataset *Saint Gall*[10], z ktorého dokumentov je celkovo použitých 1 324

Dátová sada	Jazyk/Písmo	Anotácia	Počet prvkov
IAM-DB	anglický	slová	15 320
IAM-OnDB	anglický	riadky	13 049
IAM-HistDB	viacjazyčný	riadky	6 543
HKR	azbuka	znaky	715 000
HIT-OR3C	čínske	znaky	832 650
NIST	latinka	znaky	810 000
RIMES	francúzsky	slová	300 000
PERO HWR	viacjazyčný	riadky	635 320

Tabuľka 3.1: Tabuľka zobrazuje porovnanie spomenutých dátových sád. Porovnáva ich zameranie podľa úrovne anotácie prvkov, ktoré tvoria ich obsah, podľa jazyka alebo druhu písma v ktorom sú prvky napísané. Tak isto porovnáva aj ich rozsah, ktorý je udávaný v počte príslušných anotovaných prvkov.

riadkov. Ďalšou častou je dataset *Parzival*[11], ktorý celej dátovej sade poskytuje 4 330 riadkov. Použité sú tiež dokumenty z datasetu *Bentham*[6], ktorého pôvodná verzia sa skladá z viac ako 25 000 strán textu, ktoré napísal anglický filozof a reformátor *Jeremy Bentham*. Anotované sú dobrovoľníkmi, ktorý sa zúčastnili iniciatívy smerovanej na preloženie týchto spisov. Texty pochádzajú z prelomu 17. a 18. storočia a celkovo je z nich použitých 11 070 riadkov. Ďalej sa v nej nachádza 1 381 riadkov starého nemeckého rukopisu, ktoré sú prevzaté z databázy *Herzog August Bibliothek Wolfenbüttel Manuscript database*[3]. Ďalších 88 376 riadkov vychádza z dát, ktoré sú zhromaždené v zbierke *111 let českého dopisu v korpusovém zpracování*[17], vytvorenou *Zdeňkou Hladkou*. Jedná sa o 2 000 ručne písaných osobných dopisov, ktoré boli napísane v rozmedzí rokov 1902 až 2012. Z českých kroník je v datasete zastúpených 25 062 riadkov. Z datasetu *READ Dataset* sa v danej dátovej sade nachádza 178 756 riadkov, ktoré sú napísane v nemeckom jazyku. V spolupráci s *Moravsko zemskou knihovnou* bolo anotovaných niekoľko strán textu, tieto dáta vytvárajú dataset s názvom *Brno HWR* a je z neho použitých 4 307 riadkov. Ostávajúcich 320 714 riadkov textu bolo zozbieraných výskumnou skupinou projektu PERO[1].

[1] https://pero.fit.vutbr.cz/

Kapitola 4

Konzistencia dátových sád

Dátové sady sú nevyhnutnou súčasťou práce s neurónovými sieťami. Hlavnú rolu zohrávajú pri ich trénovaní. Dá sa povedať, že kvalita daného natrénovaného modelu je priamo závislá na kvalite dátovej sady a tak isto je možné povedať, že kvalita dátovej sady zas súvisí s jej konzistenciou, a teda plne korešpondujúcimi hodnotami prepisov *ground thruth* k súvisiacim dátam. Dátové sady pre rozpoznávanie textu nemusia byť plne konzistentné a to vedie k zmäteniu modelu pri jeho trénovaní, z čoho vyplýva strata potencionálne lepších výsledkov[37].

Keďže sa táto práca zaoberá konkrétnou problematikou *OCR*, a teda rozpoznávaním textu, cieľom mojej práce je vhodne upraviť alebo detektovať a eliminovať práve tie položky, v tomto prípade dvojice riadkov textu a ich prepisov *ground thruth* dátovej sady, ktoré by mohli mať na trénovanie výsledného modelu negatívny vplyv. V nasledujúcej kapitole rozoberiem jednotlivé problémy a situácie pri ktorých nekonzistencia nastáva. Tak isto načrtnem riešenia, s ktorými budem na potlačenie týchto problémov experimentovať.

4.1 Problémy v dátových sadách

Hlavný problém, ktorý v dátových sadách nastáva, vzniká pri segmentovaní, a teda vyrezávaní riadkov zo súvislého textu, ktorý sa nachádza v dokumente, ku ktorému bol prepis *ground truth* vytváraný. Takéto dátové sady sú, tak ako bolo spomínané aj v kapitole 3, zväčša vytvárané prepisom celého súvislého textu, pričom je dodržané jeho formátovanie a počet slov v jednotlivých riadkoch. Pri neskoršom segmentovaní riadkov z pôvodného dokumentu, ktoré prebieha automaticky, môžu byť z riadkov odstránené celé slová alebo podstatné črty písmen, ktoré ich definujú. Vypadnuté časti textu môžu byť zabrané inými riadkami, ku ktorým naopak nepatria, čo vytvára druhý problém. V riadku sa nachádza text, ktorý k jeho prepisu neprislúcha. Takéto časti textu alebo opäť v niektorých prípadoch celé písmená či slová pôsobia rušivo a trénovaný model zbytočne mätú. Tieto útržky textu môžu byť v krajných prípadoch také veľké, že môžu byť rozpoznávané ako písmená alebo celé slová. V týchto situáciách ani nezáleží či sú rozpoznané korektne alebo nie, pretože prepis *ground truth*, ktorý k nim patrí je priradený inému riadku. Z tohto dôvodu je pri následnom vyhodnocovaní predikcie alebo výpočte chyby pre tieto riadky zaznamenané množstvo zbytočných chýb. Príčin tohto problému môže byť hneď niekoľko.

Môže ho spôsobovať všeobecne nesprávne nastavenie segmentovania riadkov, ktoré písmená, buď zbavuje významných čŕt alebo naopak zbytočne zachytáva priveľké časti neprislúchajúcich písmen okolitých riadkov.

Ďalší dôvod sa nachádza v samotnom texte, ktorý nie je úplne vhodný na trénovanie OCR modelov. V ručne písaných listoch sa môžu vyskytovať rôzne vsuvky, a teda slová umiestnené nad zvyčajnou líniou ostatných slov, pričom sú do vety vsunuté rôznymi pomocnými znakmi napríklad v tvare písmena *V*. Ľudia vytvárajúci *ground truth* prepis takéhoto riadku môžu slovo do vety zakomponovať alebo ho z vety vypustiť. Určiť, ktorá z možností je správna, je obtiažne a nejednoznačné. Zakomponované slovo môže byť pri segmentácií z riadku tak povediac odstrihnuté a v prepise *ground truth* sa tým pádom nachádza slovo, ktoré nebude nikdy z obrázku riadka rozpoznané. Opakom tohto je slovo alebo písmeno, ktoré sa pri vyrezávaní riadka objavuje v jeho spodnej časti. V pôvodnom texte patrilo, na základe vsuvky, riadku o úroveň nižšie, kde bolo pri vytváraní jeho prepisu *ground truth* aj priradené, v aktuálnom prepise však chýba. Model ho ale rozpoznáva, čo opäť vedie k nekonzistencií na danom riadku.

Ďalším príkladom sú rôzne šípky a čiary ukazujúce zo slova niekam do jeho okolia, ktoré majú signalizovať prehodený slovosled. Opäť nastáva situácia, kedy môže byť pri vytváraní prepisu slovosled prehodený a *ground truth* hodnota bude voči obrázku riadku nevalidná.

V texte sa zvyknú nachádzať aj preškrtnuté písmena, celé slová alebo časti viet, ktoré ale OCR model dokáže plne rozpoznať. Do prepisu ale zakomponované byť nemusia. Opäť tak sú do dátovej sady pridávané nekonzistentné riadky, ktorých *ground truth* prepis nezodpovedá textu na obrázku. V texte sa mimo vyššie uvedené príklady môžu nachádzať tiež rôzne machule, škrtance a znaky, ktoré nenesú podobu alebo vyznám žiadneho z písmen. Ich určenie ako pri vytváraní prepisu *ground truth*, tak pri rozpoznávaní OCR modelom je nejasné a pre trénovanie OCR modelov môže byť nevhodné.

4.2 Návrhy riešenia

Riešenia navrhnuté na zvýšenie konzistencie použitej dátovej sady budú priamo zamerané na problémy uvedené vyššie. Ich vplyv bude následne otestovaný na experimentoch, ktoré sú bližšie popísané v kapitole 6.

Riešením problému, či sú riadky z ich pôvodného dokumentu segmentované správne bude vyžadovať vytvorenie rôznych úprav ich ohraničujúcich polygónov, na základe ktorých sú riadky neskôr z dokumentu vyrezávané. Z rôznych transformácií sa následne vytvoria rôzne varianty pre každý riadok. Potom sa pre každú vygenerovanú variantu daného riadku vypočíta hodnota CTC chybovej funkcie, na základe ktorej bude jasne poznať, ktorý riadok danému modelu vyhovoval najviac. Varianty riadkov s najnižšou hodnotou budú zoskupené do novej, takpovediac upravenej dátovej sady, na ktorej sa natrénuje nový model. Dvojicu starého a novonatrénovaného modelu bude možné porovnať na totožnej testovacej sade a tým zistiť, či takto upravená trénovacia dátová sada zaručí modelu nižšiu chybovosť.

Pri riešení ďalšieho problému sa zameriam na detekciu nekonzistentných položiek, riadkov z použitého datasetu. Sú to hlavne riadky, ktorým neprislúcha správny prepis *ground truth*. Tieto riadku môžu byt nasledovne z datasetu odstraňované alebo opravované, na základe toho či sa nachádzajú v trénovacej alebo testovacej sade. Chýbajúce znaky alebo celé slová či už v transkripcií alebo v prepise *ground truth* by mali byť jasne rozpoznateľné na základe Levenshteinovho zarovnania[27]. Ako príklad uvediem riadok textu, ktorého príslušná *ground truth* obsahuje slovo navyše, toto slovo bolo z riadka pri jeho vyrezávaní odstrihnuté, pretože sa nachádzalo vysoko nad jeho úrovňou. Pri mapovaní sekvencie *ground*

truth na prečítanú sekvenciu znakov sa vo výslednom zarovnaní objaví istá postupnosť znakov *blank*, ktorá reprezentuje znaky slova v prepise *ground truth*, ktoré sa nenamapovali na žiadny znak z výstupu OCR modelu, pretože znaky tohto slova v ňom chýbali. Keďže zarovnávanie prebieha v oboch smeroch, to isté bude platiť pokiaľ sa vo výstupe OCR bude nachádzať nejaké slovo alebo časť vety navyše voči prepisu *ground truth*. Tentokrát bude táto postupnosť znakov *blank* na opačnej strane, a teda v zarovnanom prepise *ground truth*. Pokiaľ sa budú nachádzať tieto zhluky znakov *blank* v oboch zarovnaniach, ale ich umiestnenie bude navzájom nezávislé, a teda v každom zarovnaní iné, dá sa uvažovať, že prečítaný text má voči prepisu *ground truth* prehodený slovosled.

Kapitola 5

Implementácia

V nasledujúcej kapitole budú zhrnuté implementované skripty, ktoré bolo potrebné vytvoriť pre účely prevedenia experimentov spomenutých v sekcií 4.2. Skripty sú implementované v jazyku *Python 3.7* a je možné ich rozdeliť do dvoch častí podľa toho, ku ktorému experimentu patria, keďže na sebe nie sú závislé. Na prevedenie prvého experimentu bolo týchto skriptov vytvorených niekoľko, pracujú chronologicky a dajú sa rozdeliť do jednotlivých krokov. V druhom experimente bude potrebné trénovaciu sadu zbaviť nekonzistentných riadkov a v testovacej sade previesť korekciu na chybných prepisoch *ground truth*, pre tieto účely som vytvoril skript na ich detekciu.

Niektoré mnou implementované skripty využívajú aj funkcionalitu implementovanú vo verejnom *GitHub* repozitáry *pero-ocr*[1]. Na úplné prevedenie experimentov som využíval aj skripty zo súkromného *GitHub* repozitáru *pero*, ktorý patrí výskumnej skupine projektu PERO.

5.1 Dataset s upravenými variantami riadkov

Pre účely prvého experimentu som vytvoril viacero skriptov, keďže celý experiment sa skladá z niekoľkých dielčích krokov, tak ku každému z nich prislúcha jeden skript. Ich vstupmi sú výstupy z predchádzajúceho kroku. V prvom kroku bolo potrebné stanoviť počiatočné transformácie polygónov, ktoré definujú ohraničenie riadkov v pôvodnom dokumente. Následne bol každý riadok vyrezaný v každej variante, ktorá bola definovaná v prvom kroku. Dokopy tak vzniklo 27 variánt každého riadku, čo sa dá interpretovať aj ako 27 rôznych variánt celého datasetu. Tieto riadky podliehali transformáciám ako predĺženie, upráva výšok alebo posun ich ohraničujúceho okna. Následne bol spomedzi všetkých variánt riadka vybraný pravé ten, ktorého výsledná hodnota CTC chybovej funkcie[13] bola spomedzi všetkých ostatných najnižšia. Všetky riadky v ich najlepšej verzií tak vytvorili novú dátovú sadu, na ktorej si bol predom natrénovaný OCR model najistejší.

Stanovenie jednotlivých transformácií riadka

Na začiatku bolo potrebné stanoviť, aké sú možnosti úprav daného riadka a ktoré z nich sú vôbec zmysluplné a mohli by viesť k zlepšeniu výsledkov. Riadok je teda možné meniť hneď v niekoľkých smeroch, a teda ho predlžovať alebo skracovať, zväčšovať alebo zmenšovať každú z jeho dvoch výšok, ktorými je polygón daný alebo celé okno posúvať na oboch

[1] https://github.com/DCGM/pero-ocr

```
<PcGts xmlns="http://schema.primaresearch.org/PAGE/gts/pagecontent/2013-07-15">
  <Page imageFilename="Mistni_narodni_vybor_Hladke_Zivotice_inv_c_15_008.jpg"
        imageWidth="4000" imageHeight="2820">
    <TextRegion id="6a07acd2-a58b-43b7-b7ad-2298a6fac04a">
      <Coords points="452,256 384,256 384,336 524,336 524,256 452,256"/>
      <TextLine id="acc82e8b-abcf-4fa0-97b4-e30774147501"
                custom="heights_v2:[59.8,19.7]" conf="1.000">
        <Coords points="384,256 384,336 452,336 524,336 524,256 452,256 384,256"/>
        <Baseline points="384,316 452,316 524,316"/>
        <TextEquiv>
          <Unicode>Str. 10.</Unicode>
        </TextEquiv>
      </TextLine>
    </TextRegion>
  </Page>
</PcGts>
```

Obr. 5.1: Na obrázku sa nachádza ukážka *Page XML* súboru. Jeden takýto súbor prislúcha ku každému dokumentu, ktorý obsahuje riadky tvoriace dátovú sadu. Element *TextRegion* udáva textový región na stránke a v jeho elemente *Coords* pod atribútom *points* je dané jeho ohraničenie. Element *TextLine* definuje daný riadok, ktorý sa nachádza vo vnútri textového regiónu, ktorý týchto riadkov môže obsahovať viacero. V jeho atribútoch sú špecifikované potrebné informácie, ako jeho ID v atribúte *id* a jeho hornú a spodnú výšku v atribúte *custom*. Jeho ohraničujúci polygón vychádza zo spodnej dotyčnice písmen v danom riadku, ktorej záchytné body sú uložené v elemente *Baseline* a oboch výšok. Tieto body sa nachádzajú v elemente *Coords*. Posledná podstatná časť je *ground truth*, ktorej prepis je uložený v elemente *Unicode* v príslušnom zanorení.

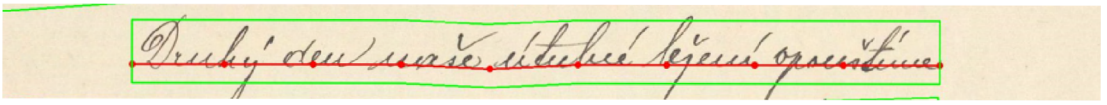
osách. Transformácie pracujú s už danými hodnotami, ktoré jednotlivé riadky v ich pôvodnom dokumente definujú. Tie boli stanovené dedikovaným modelom[24], ktorý tieto riadky rozpoznáva a je schopný ich z pôvodného dokumentu identifikovať. Na začiatku stanoví *baseline*, ktorá je dotyčnicou spodných častí písmen. Z nej následne odhadne výšky a z týchto záchytných bodov zostaví stávajúci polygon, inak povedané nájde takzvané najmenšie ohraničenia riadka.

Tieto dáta sú uložené v XML súboroch, ktorých formát sa nazýva *Page XML*, ich štruktúru je možné vidieť na obrázku 5.1.

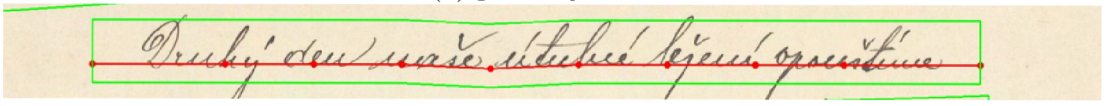
Na prvotnej testovacej sade, ktorá pozostávala z 10 náhodne zvolených dokumentov, pričom na riadkoch každého z nich bola prevedená každá dielčia transformácia, ktorých ukážky sú zobrazené na obrázku 5.2, s kladnými aj zápornými 10%, som sledoval či je ohraničenie riadku stále korektné, a teda či je zachytený celý jeho text alebo naopak či sa za jeho hranice nedostáva príliš veľká časť okolitých riadkov. Úpravy, ktoré boli príliš drastické a už z vyrenderovaných vzoriek bolo jasne vidieť nekonzistenciu, boli vyradené. Z tohto dôvodu boli odstránené možnosti horizontálneho posunu, a teda posunu okna pôvodných rozmerov na osi x spolu s akýmkoľvek skracovaním riadka.

Finálne transformácie, s ktorými sa v experimente ďalej pracovalo boli aplikované ako samostatne, tak aj vo všetkých možných kombináciách bez toho aby sa navzájom vylučovali. Dielčie transformácie teda boli nasledovné:

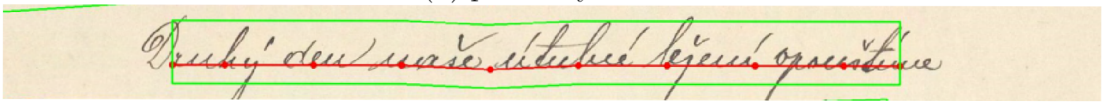
- predlžovanie riadka o 5% a 10%
- upráva celkovej výšky riadka o -10% a +10%



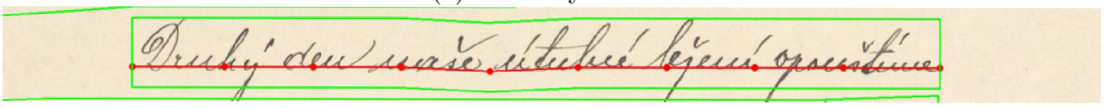
(a) pôvodný riadok



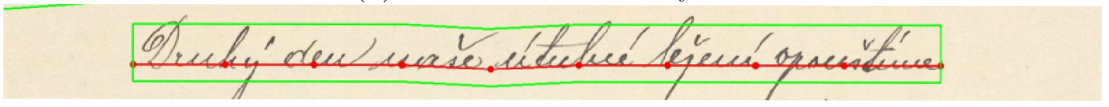
(b) predlžený riadok



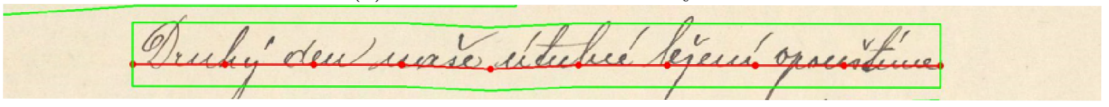
(c) skrátený riadok



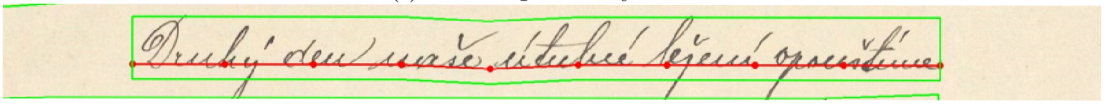
(d) riadok so zväčšenou výškou



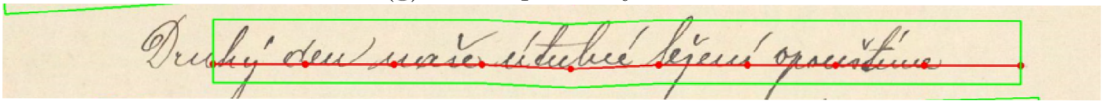
(e) riadok so zmenšenou výškou



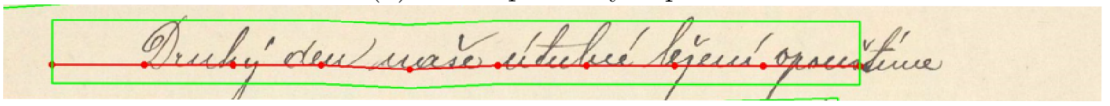
(f) riadok posunutý nadol



(g) riadok posunutý nahor



(h) riadok posunutý doprava



(j) riadok posunutý doľava

Obr. 5.2: Na obrázkoch sa nachádza znázornená každá z jednotlivých transformácií, každá z nich bola prevedená s koeficientom 10%. Na obrázkoch (c), (h) a (j) je jasne vidieť, že dané transformácie zbavujú pôvodný riadok značných kusov pôvodného textu.

- vertikálny posun riadka na osi y o 10% smerom nadol a 10% smerom nahor

Po skombinovaní všetkých nevylučujúcich sa možností vzniklo 27 rôznych variánt každého riadka. Medzi nimi sa nachádzajú ako samostatne stojace transformácie, tak ich kombinácie a tak isto pôvodný neupravený riadok.

Vytvorenie transformovaných riadkov

Pre vytvorenie transformovaných riadkov bolo potrebné vytvoriť transformované *Page XML* súbory. Všetky úpravy sa prevádzali v elementoch *TextLine*. V nich sa upravovali predovšetkým elementy *Coords*, presnejšie ich atribúty *points*, ktoré prislúchajú k jednotlivým riadkom. Skript iteruje cez priečinok so vstupnými *Page XML* súbormi a podľa vstupných parametrov upravuje potrebné atribúty.

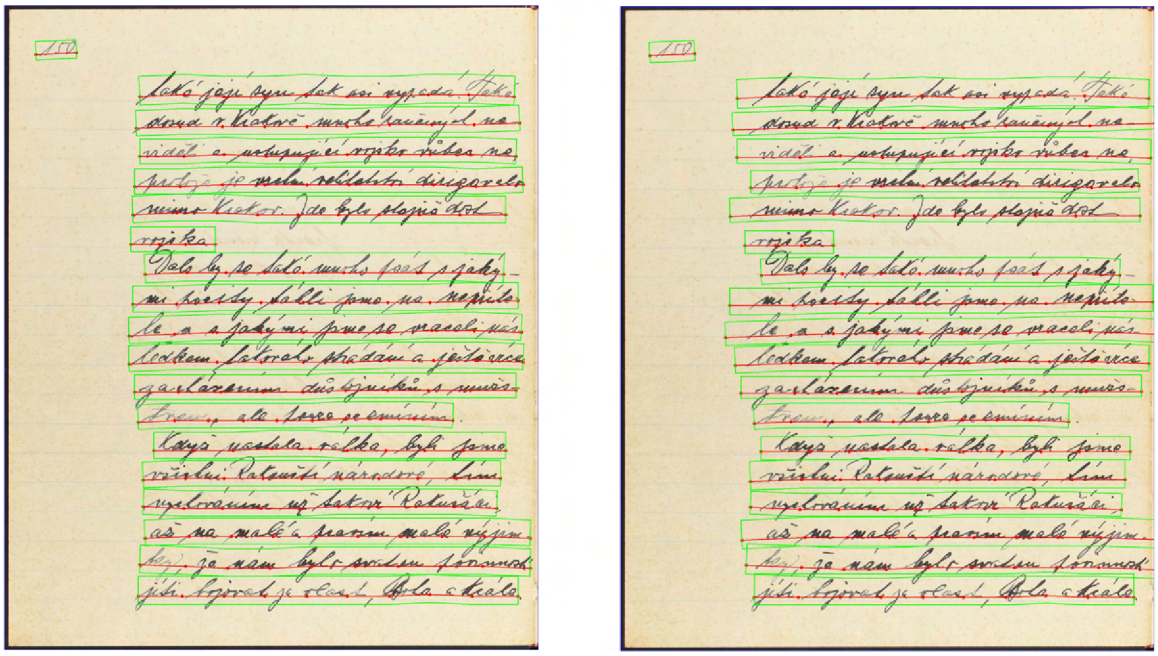
Pokiaľ je cieľom transformácie upraviť dĺžku riadka, stane sa tak upravením *points* v prislúchajúcom *Baseline* elemente. Toto predĺženie prebieha posunutím krajných bodov pod rovnakým uhlom, v akom sa nachádza jej posledný segment. Odvodenie, kam až je krajné body potrebné posunúť, aby výsledná dĺžka odpovedala požadovaným percentám, je dosiahnuté spočítaním dĺžok jednotlivých segmentov *Baseline*. Ak má transformácia v sebe aj zmenu výšky pôvodného riadka, tak je prepísaná hodnota *custom* atribútu v prislúchajúcom riadku na prepočítané hodnoty. Ak sa nachádza v transformácií aj vertikálny posun celého okna, tak sa opäť pozmenia obe výšky. Typicky je z nich jedna zväčšená a jedna skrátená, podľa toho, do ktorého smeru je požadovaný riadok potrebné posunúť. To má za následok posunutie záchytných bodov celého polygónu pri ich odvodzovaní. Nakoniec sú všetky body definujúce ohraničenie riadka prepočítané. Ich poloha vychádza zo súradníc jednotlivých bodov *Baseline* a výšok. Stanovené sú podla kolmíc postupne na každý zo segmentov *Baseline*, ich body dotykov sú jednotlivé záchytné body a konce pomyselnej dotyčnice sú stanovené na základe výšok.

Výber najlepších variánt riadkov

Každá varianta každého riadka bola spracovaná natrénovaným OCR modelom. Z výsledných *logit*-ov, bola na jednotlivých výstupoch voči *ground truth* spočítaná CTC chybová funkcia[13]. Hodnoty každého riadku boli spolu s ich ID uložené do súborov, podľa ich pôvodného dokumentu. Na základe týchto hodnôt bola pre každý riadok vybraná varianta, ktorá mala hodnotu použitej chybovej funkcie najnižšiu, a teda OCR model pri určovaní transkripcie si bol na vybranej variante najistejší. Z týchto riadkov boli následne pre každý dokument dátovej sady poskladané *Page XML* súbory definujúce riadky, ktoré sa na nich nachádzajú. Názorná ukážka ako sa dané XML súbory líšia a ako menia vlastnosti riadkov na príslušnej stránke textu je možné vidieť na vyrenderovaných dokumentoch, ktoré sa nachádzajú na obrázku 5.3. Najlepšie varianty riadkov boli následne z dokumentov vyrezané a spolu tvoria novú verziu datasetu, na ktorej model, ktorý varianty riadkov porovnával, dosahuje najlepšie výsledky.

5.2 Detekcia chybných riadkov

Pre účely druhého experimentu bolo potrebné vhodne detekovať riadky, ktorých prislúchajúci prepis *ground truth* neodpovedá textu nachádzajúcemu sa na obrázku. Jedná sa hlavne o prípady slov, ktoré boli do viet vsunuté neskôr pomocou rôznych symbolov v tvare písmena V alebo predstavujú opravu zaškrtaného slova. Vo všeobecnosti sú takto doplnené

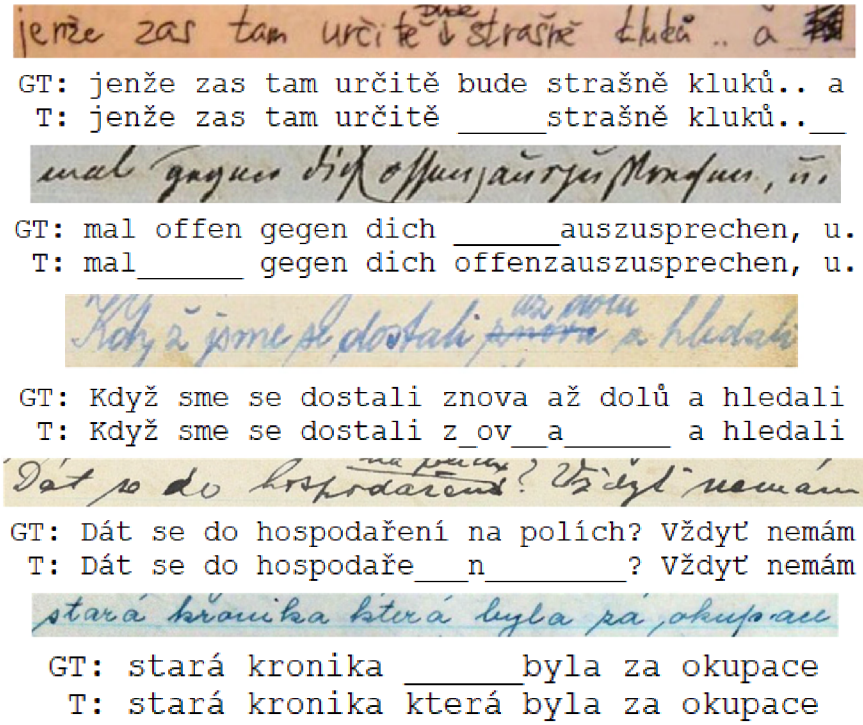


Obr. 5.3: Na obrázku vľavo sú riadky segmentované na základe pôvodných *Page XML* súborov, čiže na nich nie je prevedená žiadna transformácia. Na obrázku vpravo sa nachádzajú riadky s najlepším možným ohraničením spomedzi ich všetkých možných variánt.

slová umiestnené niekde nad písmenami pôvodného riadka a pri jeho vyrezávaní tak boli zahodené. Ide teda hlavne o detekciu riadkov, ktorým v ich prepise chýba slovo. Ukážky podozrivých riadkov, ktoré sú cieľom detekcie sú zobrazené na obrázku 5.4.

Slovo, ktoré je v prepise *ground truth* navyše alebo je v ňom naopak vynechané, nie je nejak závislé na celkovej dĺžke textu, ktorý je obsiahnutý či už v samotnom prepise alebo v danom riadku. Preto bolo cieľom detekovať podozrivý riadok tak isto nezávisle na jeho odpovedajúcej dĺžke. Keďže vypadnuté slovo o dĺžke napríklad 3 znakov, v celom prepise o dĺžke 30 znakov, nie je detekovateľné na základe CER, ktorá by v prípade, že ostatné znaky sú prečítané správne bola 10%. Takýto riadok je jednoducho zameniteľný s riadkom, kde boli len nesprávne prečítané 3 znaky.

Algoritmus pre detekciu podozrivých riadkov rešpektuje tvrdenie uvedené vyššie. Základom jeho činnosti je detekcia zvýšenej hustoty *None* na jednom mieste v zarovnanom prepise *ground truth* alebo samotnej transkripcií. Skript iteruje naraz cez obe zarovnané sekvencie znakov a v prípade, že nájde *None* v jednej zo sekvencií, aktivuje príslušné počítadlo. Celkovo skript pracuje s dvomi počítadlami. Jedno pre zarovnaný prepis *ground truth* a jedno pre zarovnanú transkripciu. Každé z nich je inkrementované nezávisle zakaždým, keď je nájdený ďalší *None* v príslušnej sekvencií a dekrementované ak znak v aktuálnej iterácií nie je *None*. Akonáhle je hodnota jedného z počítadiel rovná hodnote parametru, riadok je vyhodnotený ako podozrivý a jeho ID je zapísané do výstupného súboru. Parameter je možné meniť a udáva dolnú hranicu dĺžky sekvencie *None*, a teda potencionálne chýbajúceho slova, ktoré keď riadok obsahuje, je označený ako podozrivý. Na základe toho, ktoré z počítadiel riadok označilo, je tiež možné v prípade potreby určiť, o akú chybu v danom riadku ide. Ako je vidieť na obrázku 5.4, zarovnanie neprebehne vždy bezchybne a správne rozpoznané písmená sa môžu zarovnať aj na písmená vypadnutého slova. Preto ak nastane situácia, že pri prechode sekvenciami aktuálny znak nie je *None*, ale zároveň



Obr. 5.4: Ukážky nekorešpondujúcich dvojíc obrázkov textu s ich prepisom *ground truth*, ktoré sa nachádzajú v použitej dátovej sade. Na transkripciách a *ground truth* prepisoch, ktoré sú zarovnané podľa Levenshteinovho zarovnania, je jasne vidieť vynechané, prevyšujúce alebo nesprávne umiestnené slovo. Je reprezentované sekvenciou podtržítok, ktoré predstavujú *None* mapovaný na miestach písmen vynechaného slova. Pokiaľ sa táto sekvencia nachádza v prepise *ground truth*, znamená to, že dané slovo je v nej vynechané. Ak sa naopak táto sekvencia nachádza v transkripcií, tak dané slovo je v prepise *ground truth* navyše. Ak je sekvencia podtržítok ako v prepise *ground truth*, tak aj v prislúchajúcej transkripcií, je možné, že sa jedná o prehodený slovosled. Túto situáciu je možné vidieť na druhom obrázku zhora.

ten predchádzajúci v danej sekvencií ním bol, príslušné počítadlo nie je dekrementované a ostane nezmenené. Táto podmienka by mala vykrývať chyby použitého Levenshteinovho zarovnania.

Pri navrhovaní algoritmu som pracoval s viacerými verziami základného mechanizmu detektoru. Ten je ďalej označovaný ako *K*. Prvý z nich označil riadok ako podozrivý, pokiaľ hodnota jedného z počítadiel dosiahla hodnotu *X-1* a zároveň sekvencia *None* končila medzerou. Táto verzia je ďalej označovaná ako *M*. Cieľom bolo detekovať aspoň časť riadkov, ktoré obsahovali nekorektné slová o jeden znak kratšie ako bola hodnota nastaveného parametru. Ako je vidieť aj na obrázku 5.4, pokiaľ je slovo z daného riadku vypadnuté, tak často k nemu prislúcha ešte jedna medzera. Táto je neskôr spoločne so znakmi slova namapovaná na *None*. Na rovnakom princípe je založená aj ďalšia verzia, označovaná ako *S*, ktorá k medzere pridáva ostatné interpunkčné znamienka a špeciálne znaky.

V ďalšom skúmaní riadkov a toho ako sa zarovnáva transkripcia s prepisom *ground truth* som si všimol, že pomerne často nastáva chyba pri čítaní špeciálnych znakov, hlavne − a =. Bežne sa tieto znaky ohraničujú medzerami a tak je tomu aj v niektorých prepisoch *ground truth* aj keď text na obrázku tomu neodpovedá. Pri čítaní týchto sekvencií je pre OCR model

ťažko rozpoznateľné či tam dané medzery patria alebo nie, hlavne keď sa jedná o ručne písaný text. Preto ak algoritmus detektoru narazí na postupnosť *None*−*None* a na mieste pomlčky sa nachádza ľubovolný špeciálny znak, počítadlo nie je inkrementované a v cykle sa rovno preskočí odpovedajúci počet iterácií. Verzie, ktoré obsahovali aj implementáciu týchto špeciálnych podmienok, sú označované + (*K+*, *S+*).

Kapitola 6

Experimenty

V tejto kapitole sú popísané jednotlivé experimenty, ktorých cieľom je zvýšiť konzistenciu dátovej sady, a teda zlepšiť natrénovanosť modelu. Prvý experiment pracuje s dátovou sadou a možnosťami rôznych transformácií jednotlivých riadkov pri ich segmentovaní z celistvého dokumentu. Upravuje ich parametre ako sú napríklad dĺžka, výška či posun okna na jeho vertikálnej ose. Následne je pre každý riadok z dátovej sady vybraná jeho najlepšia varianta spomedzi všetkých vytvorených transformácií. Tieto riadky spolu vytvárajú novú verziu použitej dátovej sady, na ktorej je model natrénovaný a následne porovnaný s modelom natrénovaným na pôvodnej sade. V druhom experimente sú detekované riadky, ktoré by mali mať nesprávny prepis *ground truth* voči textu, ktorý sa skutočne nachádza na obrázku. Pokiaľ sa takto detekované riadky nachádzajú v trénovacej sade, sú z nej pre účely trénovania vyradené. Detekovaným riadkom, z testovacej sady je následne ručne opravený ich prepis *ground truth* tak aby odpovedal textu nachádzajúcemu sa na obrázku. Všetky experimenty sú prevádzané na dátovej sade *PERO Handwriting Dataset*, ktorú mi pre účely tejto práce spolu s OCR modelom neurónovej siete poskytla výskumná skupina projektu PERO. Každý z modelov bol trénovaný 400 000 iteráciami.

6.1 Vyhodnocovanie

V nasledovnej sekcií zhrniem metriky, ktoré boli pri vyhodnocovaní jednotlivých experimentov používané a zároveň patria medzi najpoužívanejšie metriky vo svojich oblastiach použitia. Na objektívnu evaluáciu, či už výstupu z OCR, alebo na čo najobjektívnejšie porovnávanie rozličných OCR modelov a ich natrénovanosti sa používajú metriky, ktoré sa nazývajú CER a WER. Ide o metriky nezávislé na dĺžke porovnávaných reťazcov, čo je pre zachovanie objektivity nevyhnutné. Obe z nich pracujú s Levensthteinovou vzdialenostou2.4, čiže porovnávajú vystúp z OCR s príslušným *ground truth* prepisom. Počas testovania algoritmu na detekciu podozrivých riadkov som používal metriku *F1 score*, ktorá má uplatnenie aj na poli strojového učenia. Používa sa najmä pri vyhodnocovaní výkonnosti klasifikačných modelov.

Character Error Rate (CER)

Výpočet CER vyplýva z Levensthteinovej vzdialenosti, a teda súčtu minimálneho počtu operácií, ktoré je potrebné s jednotlivými znakmi prepisu *ground truth* spraviť aby sa rovnal výstupu OCR[26]. Formálne zapísané nasledovne,

$$CER = \frac{S + D + I}{N} \tag{6.1}$$

kde S je počet substitúcií znakov, D je počet odstránených znakov, I je počet vložených znakov a N je celková dĺžka alebo inak povedané počet všetkých znakov v pôvodnom prepise *ground truth*.

Metrika je udávaná v percentách, ktoré značia koľko znakov z výstupu bolo predikovaných nesprávne. To znamená, čím je výsledné percento nižšie, tým je výkon modelu lepší. Model, ktorý vyprodukuje bezchybný prepis má teda hodnotu CER rovnú 0%.

Pokiaľ je výstup z OCR dlhší ako samotný prepis *ground truth*, môže hodnota CER nadobúdať hodnoty väčšie ako 100%. Na potlačenie tohto javu je hodnota normalizovaná na základe nasledovnej rovnice:

$$CER_{normalized} = \frac{S + D + I}{S + D + I + C} \tag{6.2}$$

kde C je počet správne predikovaných znakov.

Určiť na základe CER či je model dostatočne dobrý, nie je úplne jednoznačné. Treba brať do úvahy aj konkrétny prípad použitia a komplexnosť celého problému, v prípade OCR zložitosť vstupných obrázkov. Na aspoň približné určenie kvality daného modelu, bol v roku 2009 publikovaný článok s názvom *How Good Can It Get? Analysing and Improving OCR Accuracy in Large Scale Historic Newspaper Digitisation Programs.*[19], ktorý rozdeľuje OCR modely podľa hodnoty CER do troch kategórii, 1-2% znamená, že výkonnosť modelu je dobrá. Pokiaľ je hodnota v rozmedzí 2-10%, model je priemerný a ak hodnota prekračuje 10% výkonnosť modelu je podpriemerná až slabá. V prípade rozpoznávania veľmi zložitého rukopisu, ktorý obsahuje mnoho slovo z okraja slovnej zásoby, je prípustná aj hodnota pohybujúca sa okolo 20%.

Word Error Rate (WER)

Druhou významnou metrikou na meranie výkonu a porovnávanie rozličných OCR modelov je WER[26]. Jej hodnota naberá vyznám v prípade rozpoznávania viet či súvislého textu. Tak isto je pomerne dôležitá, pokiaľ rozpoznávaný text obsahuje napríklad rodné alebo telefónne čísla, keďže v ich prípade je dôležitá správnosť každého jedného znaku. Formálne je zapísaná nasledovne:

$$WER = \frac{S_w + D_w + I_w}{N_w} \tag{6.3}$$

Od metriky CER sa líši len v najmenšej porovnávanej položke, ktorou je v tomto prípade celé slovo w.

Typicky je hodnota WER vždy značne vyššia od hodnoty CER.

F1 score

F1 score je metrikou používanou v oblasti strojového učenia, používa sa predovšetkým na odhad výkonnosti klasifikačných modelov a jej výhodou je nezávislosť na počte prvkov v jednotlivých triedach[25]. Je spojením dvoch jednoduchších výkonnostných metrík, *Precision* a *Recall*. Všetky z nich sú udávane v percentách.

Pri počítaní ako *F1 score*, tak *Precision* a *Recall* je potrebné okrem počtu prvkov v jednotlivých triedach ešte poznať nasledujúce hodnoty:

True positives - pozitívne prvky, ktoré boli predikované ako pozitívne

False positives - negatívne prvky, ktoré boli predikované ako pozitívne

True negatives - negatívne prvky, ktoré boli predikované ako negatívne

False negatives - pozitívne prvky, ktoré boli predikované ako negatívne

Metrika *Precision* vyjadruje, koľko prvkov z celej množiny pozitívnych prvkov bolo predikovaných ako pozitívne. Matematicky zapísané nasledovne:

$$Precision = \frac{True\ positives}{True\ positives + False\ positives} \tag{6.4}$$

Model s nízkou hodnotou *Precision* síce nenájde veľa pozitívnych prvkov, ale tie ktoré vyberie, sú s vysokou pravdepodobnosťou správne. Naopak, model, ktorého hodnota *Precision* je vysoká, nájde väčšinu alebo všetky pozitívne prvky, ale okrem nich vyberie aj veľké množstvo nesprávnych, negatívnych prvkov.

Metrika *Recall* vyjadruje, koľko prvkov z celej množiny prvkov predikovaných ako pozitívne je skutočne pozitívnych. Matematicky zapísané nasledovne,

$$Recall = \frac{True\ positives}{True\ positives + False\ negatives} \tag{6.5}$$

Spojením dvoch vyššie uvedených metrík vzniká metrika *F1 score*, ktorá je definovaná ako ich harmonický priemer zapísaný nasledovne,

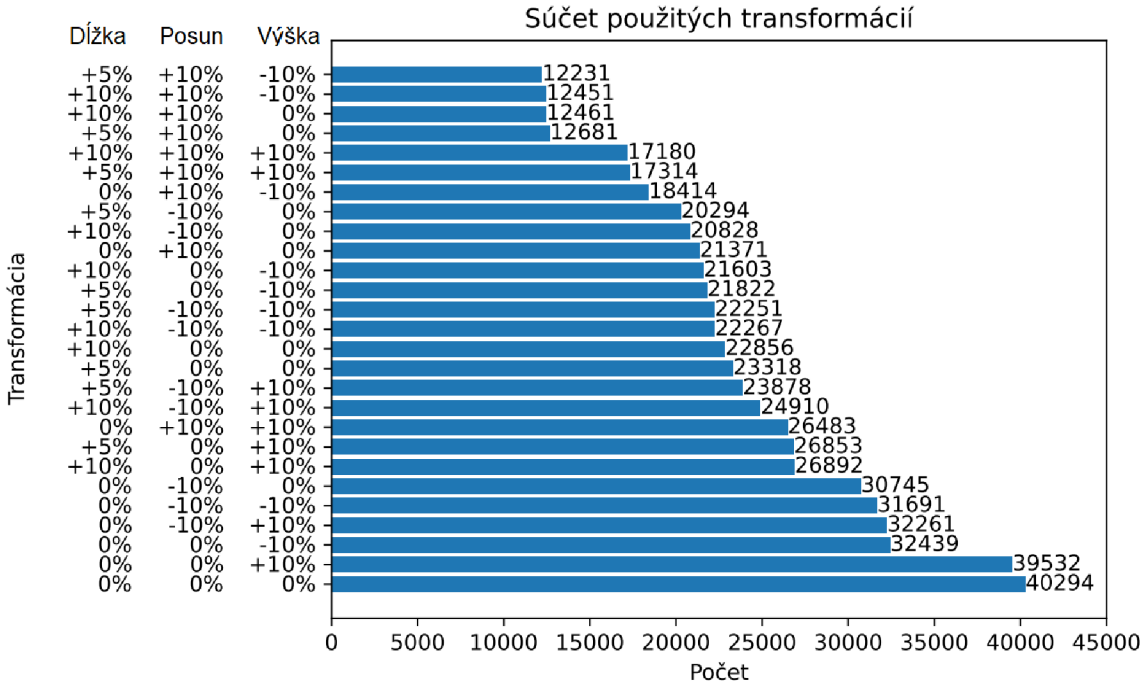
$$F1\ score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{6.6}$$

Pokiaľ je potrebné na vyhodnotenie klasifikačného modelu použiť len jedno číslo, uprednostňuje sa práve *F1 score*.

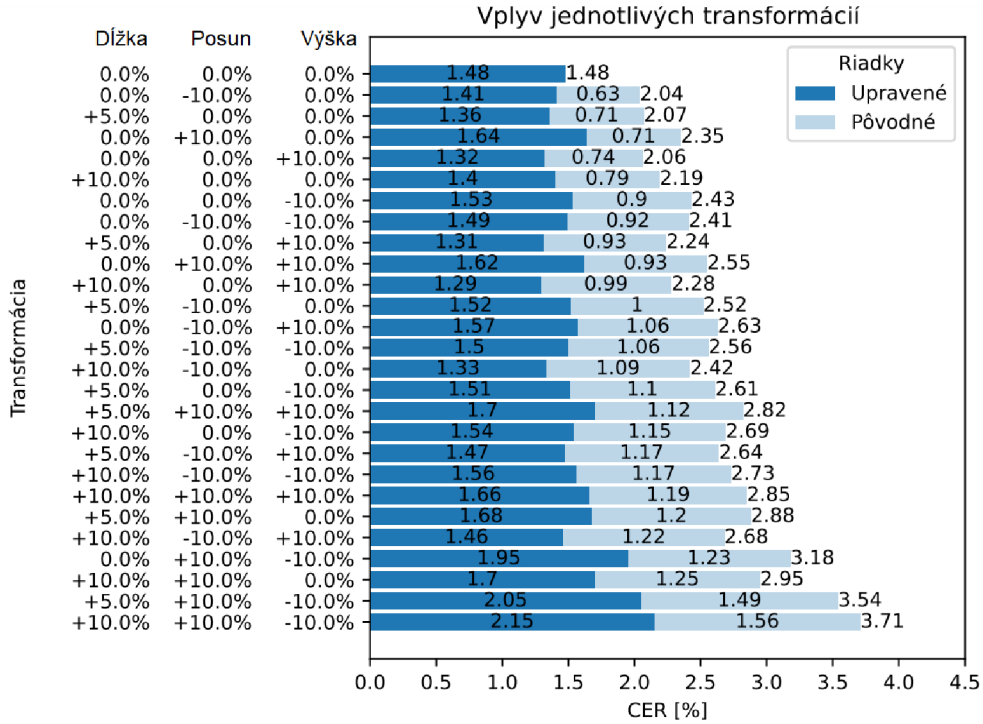
6.2 Vplyv vlastností okna definujúceho vyrezaný riadok na úspešnosť modelu

V nasledujúcom experimente som prevádzal niekoľko transformácií a úprav na polygónoch, ktoré ohraničujú jednotlivé riadky a na základe ktorých sú neskôr jednotlivé riadky textu vyrezávané z pôvodného dokumentu. Cieľom bolo vytvoriť verziu použitého datasetu, ktorá bude obsahovať riadky segmentované vo formáte, na ktorom si je predom natrénovaný OCR model najistejší. Ide hlavne o prípadné poupravenie riadkov, ktorým boli pri ich segmentácií odrezané podstatné črty písmen alebo príslušné diakritické znamienka, ktoré sa nachádzajú v pôvodnom prepise *ground truth*. Bez ich výskytu v danom riadku tak OCR model nemá šancu prepis určiť správne. To isté platí aj v opačnom prípade, keď do riadku preniká priveľa šumu z okolia. Zároveň bude z týchto dát možné zistiť ako pracuje detektor riadkov[24] a či sa naskytajú nejaké možnosti zlepšenia v samotnej detekcií.

Zastúpenie jednotlivých transformácií v novovytvorenej dátovej sade je vidieť na obrázku 6.1. Z výsledkov grafu vyplýva, že drvivá väčšina riadkov mala tendenciu meniť sa, na základe čoho je možné zvažovať poupravenie nastavenia alebo opätovné natrénovanie dedikovaného modelu na detekciu riadkov. Ten sa stará o prvotnú segmentáciu riadkov v pôvodnom dokumente. Síce ani jedna z konkrétnych použitých transformácií nebola zastúpená v značnom počte, ktorý by prevyšoval ostatné, no je možné vidieť, že množstvo z nich



Obr. 6.1: Graf zobrazuje počty výskytov transformácií riadkov v upravenej dátovej sade.



Obr. 6.2: Graf zobrazuje o koľko jednotlivé transformácie zlepšili hodnoty CER oproti pôvodným riadkom.

	Pôvodný model		Nový model	
	CER	WER	CER	WER
Pôvodný dataset	3.81%	16.22%	3.73%	15.82%
Nový dataset	2.68%	11.78%	3.45%	14.72%

Tabuľka 6.1: Porovnanie pôvodného modelu, ktorý bol natrénovaný na pôvodnej dátovej sade s modelom natrénovaným na novovzniknutej dátovej sade. Porovnanie prebehlo opäť na pôvodnej a upravenej verzií testovacej sady.

dosahovalo nižšiu hodnotu CTC chybovej funkcie keď mali zväčšenú výšku alebo bolo ich celé ohraničenie posunuté smerom nahor. Toto chovanie je možné pripisovať potrebe lepšieho zachytávania horných častí riadkov, ktoré zväčša zahŕňa diakritické znamienka alebo črty na vrcholoch vysokých písmen.

Vyhodnotenie, aký vplyv na výslednej transkripcií mali dané transformácie na k nim prislúchajúcich riadkoch, a teda tých, ktoré mali ako svoju najlepšiu variantu zvolenú práve danú transformáciu sa nachádza na obrázku 6.2. Na grafe je vidieť, ze najväčší rozdiel na výslednej transkripcií mali na svojich riadkoch práve transformácie, ktoré riadky predlžovali a posúvali ich ohraničenie smerom nadol. Z grafu tiež vychádza, že práve najdrastickejšia úprava akou je predĺženie najviac hýbala s pôvodnou hodnotou CER. To môže byť spôsobené tým, že predĺžený riadok lepšie zachytával okrajové slová alebo ich časti, ktoré neboli pri pôvodnej segmentácií správne zachytené.

Porovnanie modelu, ktorý bol natrénovaný na pôvodnej a novovytvorenej dátovej sade sa nachádza v tabuľke 6.1.

Natrénovanie modelu na novej dátovej sade neprinieslo nejak značne signifikantný rozdiel v jeho presnosti. To môže byť spôsobené odstránením variability jednotlivých riadkov, ktorá má pri trénovaní OCR modelov tak isto svoj význam. Ak je model trénovaný na vizuálne nekonzistentných riadkoch, naučí sa, ako sa s istou mierou vizuálnej nekonzistencie vysporiadať aj na testovacej sade. Na potlačenie zníženej variability trénovacích riadkov, ktoré by ale mali vo svojej vysegmentovanej forme lepšie zachytávať okrajové časti písmen a interpunkčné znamienka, som model natrénoval na novej trénovacej sade s rôznymi stupňami zosilnenej geometrickej augmentácie.

Výsledky tohto testovania sa nachádzajú v tabuľke 6.2. Vyplýva z nich, že zosilnená geometrická augmentácia pri trénovaní čiastočne pomohla presnosti modelu, pretože z celkového zlepšenia hodnoty CER na pôvodnej testovacej sade z 3.81%, ktorú dosahoval model natrénovaný na pôvodných variantách riadkov, voči hodnote 3.54%, ktorú dosahoval model natrénovaný na lepšie vyrezaných riadkoch je z väčšej časti, približne o 0.2%, dosiahnutých práve vďaka nej.

Z týchto zistení vyplýva aj záver celého experimentu. Variabilita v trénovacej sade spôsobená nedokonalosťou pri segmentácií riadkov má na výslednú natrénovanosť modelu väčší vplyv ako umelo vyprodukovaná, pretože pri trénovaní až tak nevadí mierne zvýšená chybovosť. Za to pri použití predstaveného postupu na testovacej sade je pri evaluácií modelov vyhodnotená ich reálnejšia chybovosť, keďže budú mať na vstupe lepšie detekované riadky. Respektíve, detektor riadkov nemusí byt vždy plne kompatibilný s tým, čo očakáva na vstupe OCR. To vyplýva z porovnania pôvodného modelu na pôvodných a lepšie vyrezaných riadkoch testovacej sady, kde dosiahol rozdiel v hodnotách CER približne 1.1%.

g vyjadruje konštantu, ktorou sa násobili parametre geometrických augmentácií, pri **g = 1.0** sa jedná o pôvodné hodnoty

		Testovacia sada	
		pôvodná	**nová**
Úroveň geometrických augmentácií	**g = 1.0**	3.73%	3.45%
	g = 1.5	3.58%	3.32%
	g = 2.0	3.56%	3.27%
	g = 2.5	3.54%	3.26%

Tabuľka 6.2: Tabuľka zobrazuje vplyv rôznych stupňov geometrickej augmentácie na úspešnosť modelu natrénovaného na upravených riadkoch trénovacej sady. Porovnanie prebehlo na pôvodnej a upravenej testovacej sade.

6.3 Odstránenie nekonzistentných riadkov

V tomto experimente bolo potrebné vhodne detektovať nekonzistentné riadky v dátovej sade. Do tejto skupiny spadajú predovšetkým riadky, ktorých prepis *ground-truth* nie je plne korešpondujúci s textom, ktorý sa na danom obrázku nachádza. Detekované riadky, ktoré sa nachádzali v trénovacej sade, z nej boli odstránené a tie, ktoré patrili do testovacej sady som ručne skontroloval a upravil ich prepis *ground truth* tak, aby odpovedal textu na vyrezanom obrázku. Cieľom experimentu je vytvoriť trénovaciu sadu, ktorá bude zbavená nekonzistentných riadkov a na ktorých nie je možné dosiahnuť hodnotu CER rovnú nule ani s ideálnym OCR modelom. Model, ktorý bude na tejto sade natrénovaný, bude neskôr porovnaný s modelom natrénovaným na pôvodnej trénovacej sade.

Vyhodnotenie rôznych verzií detektoru podozrivých riadkov sa nachádza v tabuľke 6.3. Porovnávanie prebiehalo na časti používanej dátovej sady, ktorú som ručne skontroloval. Pozostávala z 5 009 riadkov rozdelených do dvoch skupín, podľa toho či príslušný prepis odpovedal ich skutočnému textu alebo nie. Kontroloval som predovšetkým riadky s vysokou pravdepodobnosťou chýbajúceho alebo prevyšujúceho slova, ktoré boli vybrané, ak sa v ich zarovnaní objavil vyšší počet *None*, ktorý ale nemusel byt v sekvencií na základe ktorej ich neskor detekoval samotný detektor. Nakoniec sa v skontrolovanej dátovej sade nachádzalo 3 670 riadkov s nevalidným prepisom a 1 339, ktorých prepis *ground truth* bol správny. Ak riadky obsahovali iba drobnú nedokonalosť, vyhodnotil som ich prepis ako korektný.

Na základe výsledkov porovnávania bol pre účely experimentu zvolený detektor, ktorý berie do úvahy špeciálne znaky na konci *None* sekvencií a zároveň nepočíta s chybami v podobe medzier v tesnom okolí špeciálnych znakov (*S+*).

Zvolený algoritmus na detekciu podozrivých riadkov bol nezávisle spustený na použitej dátovej sade trikrát, vždy s iným parametrom. Ten udáva dolnú hranicu dĺžky sekvencie *None*, na základe ktorej je riadok vyhodnotený ako podozrivý. Výsledkom teda boli 3 upravené dvojice trénovacej a testovacej sady. Prvotný parameter, s ktorým je možné začať detekciu a vytvoriť prvú dvojicu sád, bol určený na základe mediánu a priemeru dĺžky všetkých slov, ktoré sa v dátovej sade nachádzajú. Priemerná dĺžka slova bola vypočítaná na hodnotu 5.029 a medián dĺžky slov sa v použitej dátovej sade rovnal 5. Vychádzal som práve z týchto hodnôt, keďže použitá dátová sada obsahuje texty v rôznych jazykoch a z rôzneho obdobia. Pri použití parametru 5 bolo celkovo detekovaných 3 276 riadkov textu, z

c predstavuje parameter algoritmu pre detekciu chybných riadkov

		TP	**FP**	**TN**	**FN**	**F1 score**
c = 3	K	3 627	734	605	43	0.90324
	M	**3 646**	923	416	**24**	0.88505
	S	**3 646**	939	400	**24**	0.88334
	K+	3 586	**660**	**679**	84	**0.90601**
	S+	3 619	879	460	51	0.88614
c = 4	K	3 093	323	1 016	577	0.87298
	M	3 295	487	852	375	0.88432
	S	**3 324**	508	831	**346**	0.88616
	K+	3 030	**290**	**1 049**	640	0.86695
	S+	3 285	450	889	385	**0.88723**
c = 5	K	2 654	164	1 175	1 016	0.81812
	M	2 891	232	1 107	779	0.85117
	S	**2 906**	239	1 100	**764**	**0.85282**
	K+	2 590	**153**	**1 186**	1 080	0.80773
	S+	2 852	216	1 123	818	0.84654

Tabuľka 6.3: Tabuľka zobrazuje porovnanie predstavených verzií detektoru s použitím rôznych parametrov. Toto testovanie prebiehalo na plne skontrolovanej časti použitej dátovej sady.

c predstavuje parameter algoritmu pre detekciu chybných riadkov,
c = 0 je ekvivalentné s pôvodnou dátovou sadou

		Testovacia sada			
		c = 0	**c = 5**	**c = 4**	**c = 3**
Trénovacia sada	**c = 0**	3.81%	3.75%	3.73%	3.71%
	c = 5	3.62%	3.60%	3.57%	3.55%
	c = 4	3.77%	3.72%	3.71%	3.69%
	c = 3	3.83%	3.78%	3.76%	3.75%

Tabuľka 6.4: Tabuľka zobrazuje vplyv výsledných poupravených testovacích a prefiltrovaných trénovacích sád na úspešnosť modelu.

nich 22 spadalo pod testovaciu sadu. Tie boli ručne skontrolované a v prípade potreby som previedol požadovanú korekciu v ich prepisoch *ground truth*. Z príslušnej trénovacej sady tak bolo odobratých 3254 riadkov. Druhá dvojica bola vytvorená s použitím parametru 4, s ktorým skript detekoval dokopy 4 944 podozrivých riadkov, z nich 42 sa nachádzalo v testovacej sade a 4 902 riadkov bolo z trénovacej sady odobratých. V treťom behu boli v dátovej sade detekované riadky, ktorých sekvencia *None* dovŕšila dĺžku 3. Takto bolo detekovaných 10 398 riadkov, z čoho 98 patrilo do testovacej a 10 300 do trénovacej sady.

Porovnanie jednotlivých modelov, ktoré boli natrénované na trénovacích sadách vytvorených na základe výsledkov z detektoru nekonzistentných riadkov pri použití rôznych parametrov sa nachádza v tabuľke 6.4. Spolu s trénovacími sadami boli vytvorené aj príslušné poupravené testovacie sady, na ktorých vzájomné porovnanie modelov prebehlo.

Z výsledného porovnania je vidieť, ze najväčší rozdiel v zlepšení na pôvodnej testovacej sade dosiahol model natrénovaný na trénovacej sade vytvorenej s pomocou detektoru s parametrom 5. Jedná sa o rozdiel 0.19% na hodnotách CER, ktorý nie je nejak zásadne väčší hlavne z dôvodu, že celkový počet nekonzistentných riadkov, ktoré sa v trénovacej sade nachádzajú je malý. S použitím tohto parametru ich bolo identifikovaných iba 3 254, čo tvorí približne 0.5%. Na výsledkoch je tiež možné pozorovať, že odstránenie náročnejších riadkov, takých, ktorých zarovnanie výslednej transkripcie obsahovalo kratšie postupnosti *None*, z trénovacej sady pôsobí kontraproduktívne. Model pri vyhodnocovaní podobných riadkov v testovacej sade potom dosahoval horšie výsledky ako model, ktorého trénovacia sada bola zbavená len riadkov s vysokou pravdepodobnosťou chyby v príslušnom prepise.

Na základe experimentu vyplýva, že z trénovacej sady má zmysel odstrániť riadky, ktoré majú vysokú pravdepodobnosť, že v ich prepise slovo chýba alebo sa naopak nachádza navyše. Naopak tým, že na testovacej sade sa riadky neodstraňujú, ale opravuje sa ich prepis, dáva najväčší zmysel nechať detektovať aj riadky s nižšou pravdepodobnosťou chýbajúceho alebo prevyšujúceho slova. Celkom konzistentne je totiž vidieť zlepšenie všetkých modelov pri dôkladnejšie opravených testovacích sadách.

Kapitola 7

Záver

Cieľom tejto práce bolo preskúmať konzistenciu dátových sád pre rozpoznávanie textu a následne sledovať aký vplyv má prípadná nekonzistencia pri trénovaní a vyhodnocovaní OCR modelov. Následne stanoviť aké problémy nekonzistenciu v dátových sadách spôsobujú a definovať postup na ich potlačenie.

Nekonzisteciu je možné rozdeliť podľa problémov, ktoré ju spôsobujú. Vizuálna nekonzsitencia je spôsobená hlavne nedokonalosťami v automatickej segmentácií riadkov. Prejavuje sa odstránením čŕt písmen, ktoré pomáhajú k ich následnému správnemu rozpoznaniu. Ďalšia nekonzistencia nastáva medzi textami, ktoré sa skutočne nachádzajú na obrázku a k nim príslušným prepisom *ground truth*, ktoré nie sú správne a nedefinujú pôvodný text v jeho skutočnej forme.

Na potlačenie vizuálnej nekonzistencie bolo vytvorených 27 variant každého riadku z použitej dátovej sady. Následne boli pre každý z riadkov vybraté ich najlepšie varianty, ktoré boli určené na základe výsledku CTC chybovej funkcie. Tieto riadky spolu tvorili novú verziu dátovej sady. Pre odstránenie nekonzistentných riadkov z pôvodnej dátovej sady, bol navrhnutý detektor riadkov, ktorý označil riadky, ktorým v ich prepise chýbalo alebo prevyšovalo slovo alebo bol prehodený slovosled. V závislosti na vstupnom parametre detektoru, boli následne chybné riadky z trénovacej sady odstránené a v testovacej sade bol ručne opravený ich prepis.

Na výsledkoch z experimentov bol preukázaný vplyv konzistentnosti dátovej sady na úspešnosť OCR modelu. Potlačenie vizuálnej nekonzistencie malo najväčší dopad na testovaciu sadu, na nej dosahovali všetky testované OCR modely konzistentne lepšie výsledky. Pôvodný model sa na lepšie vyrezanej testovacej sade zlepšil o 1.1%. Naopak model, ktorý bol natrénovaný na takto vytvorenej trénovacej sade nedosahoval nejak zásadne lepších výsledkov. Výsledky nezlepšila ani dodatočne zosilnená geometrická augmentácia pri trénovaní na upravených riadkoch. Z čoho vyplýva vyznám variability riadkov, spôsobenou ich nedokonalou segmentáciou pri trénovaní. V druhom experimente bol dokázaný nepriaznivý vplyv nekonzistencie riadkov s ich prepismi. Opäť bolo vidieť zlepšujúcu sa presnosť všetkých testovaných modelov na testovacej sade v závislosti od miery jej skontrolovania a následného opravenia. Najlepšie výsledky dosahoval model, ktorému boli z trénovacej sady odobraté len riadky s vysokou pravdepodobnosťou chyby. Oproti pôvodnému modelu dosahoval približné zlepšenie 0.2% CER naprieč všetkými testovacími sadami.

Možnosti budúceho vývoja práce by pozostávali z natrénovania modelu, ktorý sa stará o segmentáciu riadkov na dátach vytvorených pri prvom experimente. Následne skúmať výsledky OCR modelu pri natrénovaní a testovaní na variantách riadkov, ktoré by vypro-

dukoval. Tak isto je tu možnosť experimentovania s inými, menšími dátovými sadami, kde by mohla nekonzistencia zohrávať väčšiu rolu na celkových výsledkoch.

Literatúra

[1] *FKI: Research Group on Computer Vision and Artificial Intelligence: IAM Handwriting Database* [https://fki.tic.heia-fr.ch/databases/iam-handwriting-database]. Navštívená: 10.12.2021.

[2] *FKI: Research Group on Computer Vision and Artificial Intelligence: IAM Historical Document Database* [https://fki.tic.heia-fr.ch/databases/iam-historical-document-database]. Navštívená: 10.12.2021.

[3] *Herzog August Bibliothek Wolfenbüttel Manuscript database* [http://diglib.hab.de/?db=mss&lang=en]. Navštívená: 26.4.2022.

[4] Bahdanau, D., Cho, K. a Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. 2014. arXiv:1409.0473. Dostupné z: https://arxiv.org/pdf/1409.0473v1.pdf.

[5] Bansal, S. *Explanation of Connectionist Temporal Classification*. 2019. Dostupné z: https://sid2697.github.io/Blog_Sid/algorithm/2019/10/19/CTC-Loss.html.

[6] Causer, T., Grint, K., Sichani, A.-M. a Terras, M. 'Making such bargain': Transcribe Bentham and the quality and cost-effectiveness of crowdsourced transcription1. *Digital Scholarship in the Humanities*. Oxford University Press (OUP). január 2018, zv. 33, č. 3, s. 467–487. DOI: 10.1093/llc/fqx064. Dostupné z: https://doi.org/10.1093/llc/fqx064.

[7] Cho, K., Merriënboer, B. van, Bahdanau, D. a Bengio, Y. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In: *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar: Association for Computational Linguistics, Október 2014, s. 103–111. DOI: 10.3115/v1/W14-4012. Dostupné z: https://aclanthology.org/W14-4012.

[8] Cristina, S. *The Attention Mechanism from Scratch*. 2021. Dostupné z: https://machinelearningmastery.com/the-attention-mechanism-from-scratch/.

[9] D'ALBE, E. E. F. The Optophone: An Instrument for Reading by Ear. *Nature*. Springer Science and Business Media LLC. jún 1920, zv. 105, č. 2636, s. 295–296. DOI: 10.1038/105295a0. Dostupné z: https://doi.org/10.1038/105295a0.

[10] 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 - HIP '11. ACM Press, 2011. DOI: 10.1145/2037342.2037348. Dostupné z: https://doi.org/10.1145/2037342.2037348.

[11] Fischer, A., Keller, A., Frinken, V. a Bunke, H. Lexicon-free handwritten word spotting using character HMMs. *Pattern Recognition Letters*. Elsevier BV. máj 2012, zv. 33, č. 7, s. 934–942. DOI: 10.1016/j.patrec.2011.09.009. Dostupné z: https://doi.org/10.1016/j.patrec.2011.09.009.

[12] Graves, A., Liwicki, M., Fernandez, S., Bertolami, R., Bunke, H. et al. A Novel Connectionist System for Unconstrained Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Institute of Electrical and Electronics Engineers (IEEE). máj 2009, zv. 31, č. 5, s. 855–868. DOI: 10.1109/tpami.2008.137. Dostupné z: https://doi.org/10.1109/tpami.2008.137.

[13] Graves, A., Fernández, S., Gomez, F. a Schmidhuber, J. Connectionist temporal classification. In: *Proceedings of the 23rd international conference on Machine learning - ICML '06*. ACM Press, 2006. DOI: 10.1145/1143844.1143891. Dostupné z: https://doi.org/10.1145/1143844.1143891.

[14] Grosicki, E., Carré, M., Brodin, J.-M. a Geoffrois, E. Results of the RIMES Evaluation Campaign for Handwritten Mail Processing. In: *2009 10th International Conference on Document Analysis and Recognition*. IEEE, 2009. DOI: 10.1109/icdar.2009.224. Dostupné z: https://doi.org/10.1109/icdar.2009.224.

[15] Grother, P. J. a Flanagan, P. A. *NIST Handprinted Forms and Characters, NIST Special Database 19*. National Institute of Standards and Technology, 1995. DOI: 10.18434/T4H01C. Dostupné z: http://www.nist.gov/srd/nistsd19.cfm.

[16] Hinton, G. E., Osindero, S. a Teh, Y.-W. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*. MIT Press - Journals. júl 2006, zv. 18, č. 7, s. 1527–1554. DOI: 10.1162/neco.2006.18.7.1527. Dostupné z: https://doi.org/10.1162/neco.2006.18.7.1527.

[17] Hladká, Z. *111 let českého dopisu v korpusovém zpracování*. Masarykova univerzita, 2013. ISBN 978-80-210-6141-5. Dostupné z: https://www.muni.cz/vyzkum/publikace/1084855.

[18] Hochreiter, S. a Schmidhuber, J. Long Short-Term Memory. *Neural Computation*. MIT Press - Journals. november 1997, zv. 9, č. 8, s. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735. Dostupné z: https://doi.org/10.1162/neco.1997.9.8.1735.

[19] Holley, R. How good can it get? Analysing and improving OCR accuracy in large scale historic newspaper digitisation programs. D-Lib Magazine. 30.3.2009. ISSN 1082-9873. Dostupné z: http://www.nla.gov.au/ndp/project_details/documents/ANDP_HowGoodCanitGet.pdf.

[20] Hussain, R., Raza, A., Siddiqi, I., Khurshid, K. a Djeddi, C. A comprehensive survey of handwritten document benchmarks: structure, usage and evaluation. *EURASIP Journal on Image and Video Processing*. Springer Science and Business Media LLC. december 2015. DOI: 10.1186/s13640-015-0102-5. Dostupné z: https://doi.org/10.1186/s13640-015-0102-5.

[21] Kang, L., Riba, P., Rusiñol, M., Fornés, A. a Villegas, M. Pay Attention to What You Read: Non-recurrent Handwritten Text-Line Recognition. 2020. arXiv:2005.13044. Dostupné z: https://arxiv.org/pdf/2005.13044.pdf.

[22] Kang, L., Toledo, J. I., Riba, P., Villegas, M., Fornés, A. et al. Convolve, Attend and Spell: An Attention-based Sequence-to-Sequence Model for Handwritten Word Recognition. In: *Lecture Notes in Computer Science*. Springer International Publishing, 2019, s. 459–472. DOI: 10.1007/978-3-030-12939-2_32. Dostupné z: https://doi.org/10.1007/978-3-030-12939-2_32.

[23] Kleiner, A. a Kurzweil, R. A description of the Kurzweil reading machine and a status report on its testing and dissemination. *Bulletin of prosthetics research*. 1977, 10 27, s. 72–81.

[24] Kodym, O. a Hradiš, M. Page Layout Analysis System for Unconstrained Historic Documents. 23.2.2021. arXiv:2102.11838. Dostupné z: https://arxiv.org/pdf/2102.11838.pdf.

[25] Korstanje, J. *The F1 score*. 2021. Dostupné z: https://towardsdatascience.com/the-f1-score-bec2bbc38aa6.

[26] Leung, K. *Evaluate OCR Output Quality with Character Error Rate (CER) and Word Error Rate (WER)*. 2021. Dostupné z: https://towardsdatascience.com/evaluating-ocr-output-quality-with-character-error-rate-cer-and-word-error-rate-wer-853175297510#1db9.

[27] Levenshtein, V. I. Binary codes with correction of dropouts, insertions and substitutions of characters. 2.1.1965. Dostupné z: http://www.mathnet.ru/links/29f4c16d7880a9ffee37eea9e373f1c0/dan31411.pdf.

[28] Liwicki, M. a Bunke, H. IAM-OnDB - an on-line English sentence database acquired from handwritten text on a whiteboard. In: *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*. IEEE, 2005. DOI: 10.1109/icdar.2005.132. Dostupné z: https://doi.org/10.1109/icdar.2005.132.

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

[30] Nam, E. *Understanding the Levenshtein Distance Equation for Beginners*. 2019. Dostupné z: https://medium.com/@ethannam/understanding-the-levenshtein-distance-equation-for-beginners-c4285a5604f0.

[31] 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 Science and Business Media LLC. august 2021, zv. 80, 21-23, s. 33075–33097. DOI: 10.1007/s11042-021-11399-6. Dostupné z: https://doi.org/10.1007/s11042-021-11399-6.

[32] Perez, V. *Transformers in Computer Vision: Farewell Convolutions!* 2020. Dostupné z: https://towardsdatascience.com/transformers-in-computer-vision-farewell-convolutions-f083da6ef8ab.

[33] Phi, M. *Illustrated Guide to LSTM's and GRU's: A step by step explanation.* 2018. Dostupné z: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21.

[34] 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.* Institute of Electrical and Electronics Engineers (IEEE). november 2017, zv. 39, č. 11, s. 2298–2304. DOI: 10.1109/tpami.2016.2646371. Dostupné z: https://doi.org/10.1109/tpami.2016.2646371.

[35] Singh, P. *A Simple Introduction to Sequence to Sequence Models.* 2020. Dostupné z: https://www.analyticsvidhya.com/blog/2020/08/a-simple-introduction-to-sequence-to-sequence-models/.

[36] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L. et al. Attention Is All You Need. 2017. arXiv:1706.03762. Dostupné z: https://arxiv.org/pdf/1706.03762.pdf.

[37] Zhang, B., Li, Z., Gan, Z., Chen, Y., Wan, J. et al. CroAno : A Crowd Annotation Platform for Improving Label Consistency of Chinese NER Dataset. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations.* Association for Computational Linguistics, 2021. DOI: 10.18653/v1/2021.emnlp-demo.32. Dostupné z: https://doi.org/10.18653/v1/2021.emnlp-demo.32.

[38] Zhou, S., Chen, Q. a Wang, X. HIT-OR3C. In: *Proceedings of the 8th IAPR International Workshop on Document Analysis Systems - DAS '10.* ACM Press, 2010. DOI: 10.1145/1815330.1815359. Dostupné z: https://doi.org/10.1145/1815330.1815359.

Príloha A

Obsah priloženého pamäťového média

- *bp.pdf* - text bakalárskej práce v elektronickej podobe
- *video.mp4* - video k bakalárskej práci
- *doc/* - priečinok so zdrojovými súbormi práce v latex-u
- *src/* - priečinok so zdrojovými súbormi skriptov k experimentom