FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Zpracování výpisu z účtu pomocí metod strojového učení
Student:	Michal Lebeda
Vedoucí:	Mgr. Adam Szabó
Studijní program:	Informatika
Obor / specializace:	Znalostní inženýrství
Katedra:	Katedra aplikované matematiky
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

V dnešní době je stále mnoho míst, která si data vyměňují nestrukturovanými a strukturovanými formáty. Jedním z příkladů jsou výpisy z bankovních účtů.

Cílem práce je umožnit komplexní zpracování výpisu z účtu ve formátu pdf. Aplikace musí podporovat alespoň 5 různých druhů výpisů z různých bank.

Postup řešení:
- Proveďte rešerši stávajících metod včetně metod s využitím strojového učení pro vytěžování textu z pdf souborů.
- Na základě rešerše navrhněte a implementujte aplikaci, která data transformuje do jednotného strojově čitelného formátu. Zaměřte se na extrakci majitele, jednotlivých plateb, banky, typu účtu a na rozlišení jednotlivých transakcí.
- Vyhodnoťte přesnost navržené aplikace a, pokud je to možné, srovnejte její funkčnost s jinými aplikacemi.

Data budou stažena z veřejně dostupných zdrojů.

Doporučené reference na publikace:
- Business Document Information Extraction: Towards Practical Benchmarks, Skalický Matyáš, Šimsa Štěpán, Uřičář Michal, Šulc Milan, 2022 https://arxiv.org/abs/2206.11229?context=cs
- TableFormer: Table Structure Understanding with Transformers, Nassar Ahmed,

---

Elektronicky schválil/a Ing. Magda Friedjungová, Ph.D. dne 3. prosince 2022 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Livathinos Nikolaos, Lysak Maksym, Staar Peter, 2022, https://arxiv.org/abs/2203.01017
- GriTS: Grid table similarity metric for table structure recognition, Smock Brandon, Pesala Rohith, Abraham Robin, 2022, https://arxiv.org/abs/2203.12555
- Chargrid: Towards Understanding 2D Documents, Katti Anoop Raveendra, Reisswig Christian, Guder Cordula, Brarda Sebastian, Bickel Steffen, Höhne Johannes, Faddoul Jean Baptiste, 2018, https://arxiv.org/abs/1809.08799

Bakalářská práce

ZPRACOVÁNÍ VÝPISU Z ÚČTU POMOCÍ METOD STROJOVÉHO UČENÍ

Michal Lebeda

Fakulta informačních technologií
Katedra aplikované matematiky
Vedoucí: Mgr. Adam Szabó
11. května 2023

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěl bych poděkovat především Adamu Szabó, za cenné rady a časté konzultace, dále své rodině za poskytnuté zázemí při psaní práce a Marku Sušickému, za to, že se mě před touto problematikou pokoušel varovat, ale zároveň respektoval mou volbu tématu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 11. května 2023 .....

Abstrakt

Práce se zabývá strojovým zpracováním výpisu z účtu ve formátu PDF. Jelikož se jedná o nestrukturovaný dokument, nelze z něj přímočaře získat informace o účtu a jednotlivých transakcích pro další využití. Cílem práce je převod zmíněného dokumentu do strukturovaného formátu, a proto je nejdříve provedena rešerše formátu PDF, existujících metod strojového čtení nestrukturovaných dokumentů spolu s analýzou veřejně dostupných datasetů, které jsou mezi sebou porovnány a je diskutována jejich využitelnost při řešení dané úlohy.

V praktické části je popsán částečně automatizovaný postup tvorby vlastního datasetu zakládajícím na veřejně dostupných výpisech z účtu, vysvětlena problematika této činnosti a následná tvorba modelu strojového učení. Jsou zhodnoceny dosažené výsledky modelu, další možnosti jejich zlepšení a nástin rozšíření stávajícího datasetu efektivní cestou. Výsledkem práce je aplikace pro zpracování výpisu z účtu do strukturovaného formátu.

Klíčová slova výpis z účtu, nestrukturovaný formát,umělá neuronová síť, PyTorch, KIE, KILE, LIR

Abstract

This thesis deals with machine processing of bank statement in PDF format. Since it is an unstructured document, it is not possible to directly extract account and individual transaction information for further use. The goal is to convert the said document into a structured format, therefore, a research on PDF format, existing methods for machine reading of unstructured documents along with analysis of publicly available datasets is first carried out and their applicability in solving the given task is discussed.

The practical part describes a partially automated procedure of creation of a custom dataset based on publicly available bank statements, explains the problems of this activity and the subsequent creation of a machine learning model. The achieved results of the model, further possibilities of their improvement and an outline of the extension of the existing dataset in an effective way are presented. The result of the work is an application for processing account statements into a structured format.

Keywords bank statement, unstrucutred format, artificial neural network, PyTorch, KIE, KILE, LIR

Seznam zkratek

HTML	HyperText Markup Language
XML	Extensible Markup Language
PDF	Portable Document Format
XREF	Cross-reference table
BIC	Business Identification Code
IBAN	International Bank Account Number
DPI	Dots Per Inch
LI	Line Item
FNN	Feedforward Neural Network
TD	Table Detection
TSR	Table Structure Recognition
FA	Functional Analysis
TE	Table Extraction
TATR	Table Transformer
LIR	Line Item Recognition
KIE	Key Information Extraction
KILE	Key Information Localization and Extraction
BBox	Bounding Box
CSV	Comma Separated Values
CSS	Cascading Style Sheets
API	Application Programming Interface
RTF	Rich Text Format

Úvod

Ačkoliv se naše společnost nachází v době rozkvětu znalostního inženýrství, stále jsou součástí našich každodenních životů zastaralé postupy jako je výměna dat v nestrukturovaných formátech. Jedním z těchto zdrojů dat je výpis z bankovního účtu ve formátu PDF. Fakt, že PDF je nestrukturovaný formát v praxi znamená, že nelze strojově rozpoznat, zda sekvence znaků s jistotou náleží jedné větě či odstavci. V tabulkách není zřejmé, k jakému „typu informace“ z hlavičky se váže obsah buňky, kde začíná, končí nebo zda je nevyplněná. Dokonce nelze ani s jistotou detekovat tabulku samotnou.

Rozpoznání těchto informací z výpisu, a především rychlost, přitom může mít vliv na řadu běžných životních situací. Při žádosti o úvěr je žadatel bankou kontrolován a musí výpis z účtu předložit, pokud má v daný moment účet u jiné banky. Následně probíhá kontrola výpisu, řeší se výše příjmů, potenciální finanční problémy, odesílání odchozích plateb sázkovým společnostem a vhodnost žadatele na základě rizikového profilu. Firmy kvůli vedení účetnictví párují faktury s transakcemi typicky pomocí variabilního symbolu. Neziskové společnosti a politické strany mají povinnost zveřejňovat své výpisy z transparentních účtů jako prevenci střetu zájmů. Policie České republiky při vyšetřování finančních podvodů analyzuje výpisy z účtů. V roce 2019 byl v souvislosti s tímto tématem vypsán tendr od Ministerstva vnitra [1].

Aktuálně banky nemají povinnost umožnit stahování výpisu ve strukturované formě. Pokud tato možnost existuje, bývá soubor výpisu dostupný ve formátu CSV, který nepodporuje elektronický podpis zaručující integritu výpisu. Tím pádem by kdokoli mohl tento soubor dále volně upravovat, a proto není zvykem, aby byl tento formát akceptován. Nejběžnějším formátem výpisu z účtu je stále nestrukturovaný formát PDF, jelikož zaručuje shodný vzhled na všech zařízeních včetně tisku.

Jelikož výpis z účtu je citlivý dokument, není příliš obvyklé, že by ho někdo sdílel veřejně. Mezi výpisy různých bank jsou značné rozdíly, zejména umístění informací v tabulce transakcí, což znesnadňuje strojové čtení. V současné době neexistuje veřejný dataset, který by obsahoval výpisy z účtu českých bank. Nedostupnost testovacích dat značně komplikuje vývoj jakéhokoli řešení pro automatizaci čtení informací z výpisu, protože neexistuje způsob, jak ověřit jeho kvalitu.

V teoretické části práce je provedena rešerše existujících datasetů a s tím souvisejících modelů zaměřující se na zpracování dokumentů. Dále jsou prozkoumána dostupná komerční řešení a popsáno jedno konkrétní řešení z bankovního prostředí. Z důvodu neexistence datasetu zaměřeného přímo na výpisy z účtu ani takového, který by obsahoval alespoň zdánlivě podobná data, je vytvořen dataset vlastní. Je popsán sběr podkladových dat, jejich správa a samotná tvorba datasetu poloautomatizovanou cestou.

V rámci praktické části byly otestovány vybrané modely z teoretické části, zvolena architektura modelu vyhovující požadavkům pro naplnění cílů práce a následně implementován model, pro který byla předzpracována data do jeho vstupního formátu a diskutovány jeho výsledky.

Výsledkem je vytvořený program, umožňující převod výpis z účtu ve formátu PDF do struk-

turované reprezentace formátu JSON.

Cíle

Cílem práce je analýza používaných metod čtení informací z nestrukturovaných dokumentů a rešerše stávajících řešení. Dále bude v rámci práce vytvořena aplikace, která elektronicky generovaný soubor bankovního výpisu z účtu ve formátu PDF převede na strukturovaný soubor jednotného formátu. Program by měl být schopen provést tento proces pro alespoň 5 českých bank, zaměřovat se jak na obecné informace o účtu, tak na rozeznání jednotlivých plateb, extrakci těchto informací a následný převod do strukturovaného formátu. Nakonec bude vyhodnocena přesnost vytvořeného řešení a provedena diskuse.

Kapitola 1

Výpis z bankovního účtu

Kapitola se zabývá výpisy z bankovních účtů, jejich strukturou a typy informací, které se v nich nachází. Zvlášť je popsána tabulka operací proběhlých za dané období na účtu.

Výpis z bankovního účtu je přehledné shrnutí všech operací, které na účtu proběhly za dané období [2]. Banka ho uživateli obvykle dává ke stažení ve formátu PDF s elektronickým podpisem, který zaručuje integritu. Výpis účtu může být tvořen jednou nebo více stranami.

Na obrázku 1.1 lze vidět ukázku první strany výpisu z bankovního účtu Fio banky. Oranžově jsou vyznačeny obecné informace, které se týkají účtu nebo výpisu samotného, tedy například jméno majitele a období výpisu. Tyto informace se obvykle vyskytují na první straně výpisu. Červeně je pak vyznačena hlavička tabulky operací obsahující názvy polí jednotlivých operací, následována samotnými operacemi vyznačenými střídavě zeleně a modře pro lepši přehlednost.

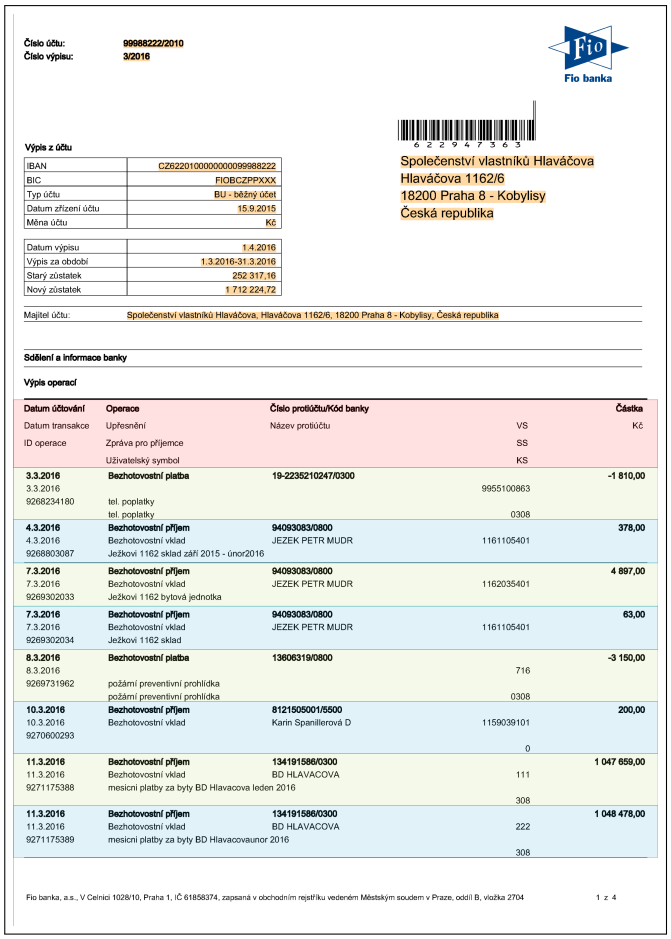
1.1 Obecné informace

Z obecných informací byla věnována pozornost zejména následujícím polím, které byly rozděleny do skupin dle jejich charakteru:

- informace o účtu:
  - BIC - identifikuje banku, u které je účet veden [3],
  - IBAN účtu - standardizovaný formát zápisu čísla účtu dle mezinárodní normy ISO 13616 [3],
  - název účtu,
  - číslo účtu,
  - jméno majitele účtu,
  - adresa majitele účtu,
  - měna účtu,
- informace o konkrétním výpisu:
  - frekvence výpisu (roční, měsíční atd.),
  - číslo výpisu,
  - datum začátku období výpisu,
  - datum konce období výpisu,

- datum vytvoření výpisu,

- agregované informace o finančních transakcích:
  - počáteční zůstatek,
  - konečný zůstatek,
  - debetní obrat (výdaje),
  - kreditní obrat (příjmy).



Číslo účtu: 99988222/2010
Číslo výpisu: 3/2016

Fio banka

Výpis z účtu

IBAN	CZ6220100000000099988222
BIC	FIOBCZPPXXX
Typ účtu	BU - běžný účet
Datum zřízení účtu	15.9.2015
Měna účtu	Kč

Datum výpisu	1.4.2016
Výpis za období	1.3.2016-31.3.2016
Starý zůstatek	252 317,16
Nový zůstatek	1 712 224,72

Společenství vlastníků Hlaváčova
Hlaváčova 1162/6
18200 Praha 8 - Kobylisy
Česká republika

Majitel účtu: Společenství vlastníků Hlaváčova, Hlaváčova 1162/6, 18200 Praha 8 - Kobylisy, Česká republika

Sdělení a informace banky

Výpis operací

Datum účtování	Operace	Číslo protiúčtu/Kód banky		Částka
Datum transakce	Upřesnění	Název protiúčtu	VS	Kč
ID operace	Zpráva pro příjemce		SS	
	Uživatelský symbol		KS	
3.3.2016	Bezhotovostní platba	19-2235210247/0300		-1 810,00
3.3.2016			9955100863	
9268234180	tel. poplatky			
	tel. poplatky		0308	
4.3.2016	Bezhotovostní příjem	94093083/0800		378,00
4.3.2016	Bezhotovostní vklad	JEZEK PETR MUDR	1161105401	
9268803087	Ježkovi 1162 sklad září 2015 - únor2016			
7.3.2016	Bezhotovostní příjem	94093083/0800		4 897,00
7.3.2016	Bezhotovostní vklad	JEZEK PETR MUDR	1162035401	
9269302033	Ježkovi 1162 bytová jednotka			
7.3.2016	Bezhotovostní příjem	94093083/0800		63,00
7.3.2016	Bezhotovostní vklad	JEZEK PETR MUDR	1161105401	
9269302034	Ježkovi 1162 sklad			
8.3.2016	Bezhotovostní platba	13806319/0800		-3 150,00
8.3.2016			716	
9269731962	požární preventivní prohlídka			
	požární preventivní prohlídka		0308	
10.3.2016	Bezhotovostní příjem	8121505001/5500		200,00
10.3.2016	Bezhotovostní vklad	Karin Spanillerová D	1159039101	
9270600293				
			0	
11.3.2016	Bezhotovostní příjem	134191586/0300		1 047 659,00
11.3.2016	Bezhotovostní vklad	BD HLAVACOVA	111	
9271175388	mesicni platby za byty BD Hlavacova leden 2016			
			308	
11.3.2016	Bezhotovostní příjem	134191586/0300		1 048 478,00
11.3.2016	Bezhotovostní vklad	BD HLAVACOVA	222	
9271175389	mesicni platby za byty BD Hlavacovaunor 2016			
			308	

Fio banka, a.s., V Celnici 1028/10, Praha 1, IČ 61858374, zapsaná v obchodním rejstříku vedeném Městským soudem v Praze, oddíl B, vložka 2704 1 z 4

Obrázek 1.1 Struktura první strany výpisu Fio banky (zdroj: [4])

Je důležité poznamenat, že není zaručen výskyt všech zmíněných polí, případně stejného pořadí, polohy a pojmenování. Některé informace se mohou vyskytovat na jiné stránce (např. „konečný zůstatek“ na konci výpisu) nebo se mohou vyskytovat vícekrát (obvykle pak na každé stránce).

Ve výpisech se také vyskytují další méně časté informace, které pro mírné zjednodušení a jejich nízkou míru užitku budou ignorovány. Jedná se například o nabídky bankovních produktů, bonusy za platby kartou a další. Tyto informace se natolik liší mezi bankami, že je nelze generalizovat. Samotná informační hodnota není příliš velká, protože tyto informace slouží spíše marketingovým účelům.

1.2 Tabulka operací

Samotný přehled operací na účtu je zapsán tabulkou, která obvykle začíná hlavičkou, obsahující názvy polí, jejichž hodnoty se vyskytují v LI pod ní. Množství polí v rámci jednoho sloupce LI může být více a hlavička u některých bank může chybět.

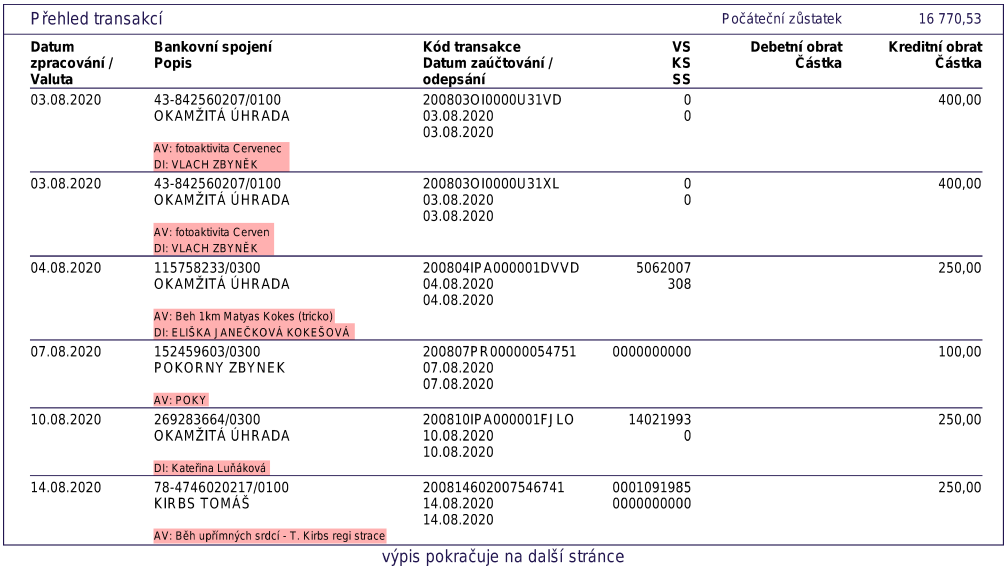
▶ **Definice 1.1** (LI). *Line Item je n-tice hodnot polí popisující jednu instanci objektu k extrakci [5].*

Na obrázku 1.1 se pod červeně vyznačenou hlavičkou nacházejí zeleně a modře vyznačené jednotlivé LI1.1. Uvnitř každého LI jsou hodnoty uspořádány způsobem vizuálně odpovídajícímu hlavičce. Mezi obvyklá pole vyskytující se v tabulce operací patří:

- kalendářní data týkající se operace:
  - datum zpracování,
  - datum valuty,
  - datum zaúčtování,
  - datum odepsání,
- číslo protiúčtu,
- DI - debetní identifikace,
- AV - zpráva pro příjemce,
- symboly:
  - VS - variabilní symbol,
  - KS - konstantní symbol,
  - SS - specifický symbol,
- obrat (také uváděno jako „částka“),
  - v některých případech se dělí na dvě samostatná pole „debetní obrat“ a „kreditní obrat“,
- identifikace platby.

U polí tabulky operací existuje mezi bankami ještě větší nekonzistence než u obecných informací. Výpisy se liší jak počtem a pojmenováním polí, tak jejich umístěním.

Ve výpisech se často vyskytují pole, pro která v hlavičce chybí název. Název pole pak bývá nalevo od hodnoty nebo nad hodnotou. Jako příklad jsou na obrázku 1.2 zvýrazněna pole `AV` a `DI`. Podobným způsobem se často vyskytují i pole se symboly `VS`, `KS` a `SS`. Tato pole jsou různá dle konkrétní banky.



Přehled transakcí — Počáteční zůstatek 16 770,53

Datum zpracování / Valuta	Bankovní spojení Popis	Kód transakce Datum zaúčtování / odepsání	VS KS SS	Debetní obrat Částka	Kreditní obrat Částka
03.08.2020	43-842560207/0100 OKAMŽITÁ ÚHRADA AV: fotoaktivita Cervenec DI: VLACH ZBYNĚK	200803OI0000U31VD 03.08.2020 03.08.2020	0 0		400,00
03.08.2020	43-842560207/0100 OKAMŽITÁ ÚHRADA AV: fotoaktivita Cerven DI: VLACH ZBYNĚK	200803OI0000U31XL 03.08.2020 03.08.2020	0 0		400,00
04.08.2020	115758233/0300 OKAMŽITÁ ÚHRADA AV: Beh 1km Matyas Kokes (tricko) DI: ELIŠKA JANEČKOVÁ KOKEŠOVÁ	200804IPA000001DVVD 04.08.2020 04.08.2020	5062007 308		250,00
07.08.2020	152459603/0300 POKORNY ZBYNEK AV: POKY	200807PR00000054751 07.08.2020 07.08.2020	0000000000		100,00
10.08.2020	269283664/0300 OKAMŽITÁ ÚHRADA DI: Kateřina Luňáková	200810IPA000001FJLO 10.08.2020 10.08.2020	14021993 0		250,00
14.08.2020	78-4746020217/0100 KIRBS TOMÁŠ AV: Běh upřímných srdcí - T. Kirbs regi strace	200814602007546741 14.08.2020 14.08.2020	0001091985 0000000000		250,00

výpis pokračuje na další stránce

Obrázek 1.2 Část výpisu banky Moneta se zvýrazněnými názvy polí, které jsou na stejném řádku následovány hodnotou v rámci LI (zdroj: [6])

Kapitola 2
Formát PDF

Cílem kapitoly je popsat formát PDF a vysvětlit důvody, proč je obtížné tento formát strojově zpracovávat.

Strukturované dokumenty obsahují veškeré potřebné informace [1] pro pochopení logických vztahů mezi jednotlivými objekty. Jako příklad lze uvést formát HTML, kde element `<p>` označuje odstavec a veškerý obsah náležící odstavci pak ve stromové struktuře leží v tomto elementu jakožto potomek, případně vícero potomků. Samotný odstavec může být ve stromové struktuře opět potomkem jiného elementu, například `<table>` nebo `<li>`. Nestrukturované dokumenty žádné z těchto informací neobsahují [7].

„*PDF je nestrukturovaný formát, tím pádem při extrakci textu z PDF vyvstávají (mimo jiné) následující problémy:*

- *instrukce pro vykreslení textu neumožňují jednoznačně určit, kde začíná odstavec,*
- *mezera může být kódována jako znak nebo se kurzor jednoduše posune na nové místo za mezerou,*
- *instrukce pro vykreslování textu mohou mít různé pořadí, na které se nedá spolehnout.*

“[8].

Samotný formát PDF tvoří již zmíněné instrukce. Při pohledu na PDF optikou programovacích jazyků pracuje se základními datovými typy:

- řetězce,
- celá čísla,
- pravdivostní hodnoty,
- „Name objects“ (rezervované řetězce).

Z těchto primitivních typů se skládají komplexnější:

- slovníky mapující jednu hodnotu na druhou,
- pole hodnot,
- streamy (obvykle binárních dat),

[1] Nebo se dají algoritmicky odvodit.

Výpis kódu 2.1 Zkrácená ukázka streamu instrukcí pro vykreslení stránky (zdroj: [9])

```
6 0 obj
<</Length 52185>>
stream

            q
            BT
            0.521569 0.780392 0.870588 rg
            /F1 1.000000 Tf
            24.000000 0 0 24.000000 59.500000 725.760000 Tm
            (Hello World!) Tj
            ET
            Q

            q
            ... atd. ...
```

- odkazy.

Soubor obsahuje „cross-reference table“ nebo také „XREF“ tabulku, která představuje mapování všech PDF objektů a jejich bytový offset. Nachází se v ní například bytová pozice metadat stejně jako pozice kořene dokumentu, ve kterém je dále odkaz na pole se samotnými odkazy na jednotlivé stránky [9]. Pokud je tato tabulka poškozena, většinou není možné správně zobrazit dokument.

V kontextu této práce je nejvíce zajímavý stream dat jednotlivých stránek. Ty se skládají z instrukcí pro vykreslení. Ve výpisů kódu 2.1 lze vidět část streamu instrukcí potřebných k vykreslení stránky. Operátory jsou prefixové, což znamená, že argumenty jsou před jejich názvy. Popis instrukcí [9] je následující:

- `q` - vloží nové grafické prostředí na zásobník,
- `BT` - započne text,
- `0.521569 0.780392 0.870588 rg` - nastaví barvu v RGB,
- `/F1 1.000000 Tf` - použije font specifikovaný v `/Resources/Font/F1`, ve velikosti 1,
- `24.000000 0 0 24.000000 59.500000 725.760000 Tm` - nastaví transformační matici fontu (zde nastavení velikosti fontu na 24 a určení pozice),
- `(Hello World!) Tj` - vykreslí text „Hello World!“,
- `ET` - ukončí text,
- `Q` - zahodí poslední prvek ze zásobníku grafických prostředí.

Z výše uvedených informací vyplývá, že strojové čtení PDF dokumentů, obzvlášť pak s různorodými tabulkami, je netriviální úkol. Vzhledem k tomu, že práce se zabývá čtením výpisů za použití metod strojového učení, nebude nízko úrovňové struktuře formátu PDF věnována další pozornost. Pro samotnou práci s PDF soubory bude použita knihovna Poppler [10], která čtení dokumentu řeší za uživatele.

Kapitola 3
Umělé neuronové sítě

Tato kapitola se zabývá umělými neuronovými sítěmi, které jsou v současné době čím dál více využívány při řešení netriviálních problémů. Pokrok v této oblasti je způsoben přelomovými objevy na poli návrhu architektury těchto sítí, algoritmů umožňujících rychlejší učení a čím dál větší dostupnosti výkonného hardwaru. Výhodou těchto sítí je jejich schopnost práce s širokou škálou vstupních dat, kdy vstupem mohou být obrázky, časové řady či sekvence znaků. Řešené úlohy mohou být regresního či klasifikačního charakteru, kde výstup může být jedna či více proměnných.

3.1 Učení s učitelem

Učení s učitelem nebo také supervised learning je založeno na využití datasetu vzorků, kde každý vzorek obsahuje vstupní data a požadovaný výstup. Vstupními daty mohou být například parametry nemovitosti, jako je plocha, vzdálenost od stanice tramvaje a výstupní hodnotou může být cena. Učení má za cíl nastavit parametry modelu tak, aby jeho výstupy spočítané na základě vstupních dat co nejvíce odpovídaly skutečným hodnotám.

Výstupní data jsou obvykle zatížena šumem a naprosto přesných výsledků tedy téměř nikdy nelze dosáhnout. Při vysoké přesnosti modelu je velice pravděpodobné, že si síť nějakým způsobem uložila správné hodnoty pro veškeré vzorky[1] a poté tyto hodnoty po identifikaci vzorku na vstupu předala jako řešení. Tomuto jevu se říká přeučení a typicky vzniká příliš velkou kapacitou modelu. Model poté již negeneralizuje problém, ale hledá ve svých interních datech konkrétní nebo velmi podobné vzorky. Na vzorcích, které nebyly použity při trénování pak většinou selhává.

Z tohoto důvodu se data dělí na tři množiny, kde první slouží k trénování modelu čili přímo ovlivňuje jeho parametry. Tento proces má za cíl minimalizovat takzvanou ztrátovou funkci, která udává chybu modelu.

Validační množina slouží k ověření, že nedochází k přeučení modelu během procesu učení, ale vnitřní parametry jí nejsou přímo ovlivněny. Při tvorbě modelu často dochází k experimentům a změnám hyperparametrů architektury sítě, které jsou obvykle měněny na základě výsledků, kterých je modelem dosaženo na trénovací a validační množině. Tímto způsobem se model může nepřímo naučit šum a přibližnou charakteristiku vzorků z validační množiny, což vede k horším výsledkům v produkčním prostředí na dosud neznámých datech. Pro zlepšení představy o očekávaných výsledcích je model před použitím v produkci vyhodnocen na testovací množině, která do té doby nesmí být žádným způsobem použita pro tvorbu modelu. V momentě vyhodnocení pak již model nelze na základě těchto výsledků žádným způsobem upravovat.

[1] Nebo alespoň velké množství vzorků.

3.2 Dopředná neuronová síť

„Dopředná neuronová síť, anglicky Feedforward neural network (FNN), je typem takzvané umělé neuronové sítě[2]*, jejíž snahou je na základě vstupu odhadnout hodnotu nějaké funkce g^* na svém výstupu. Například pro klasifikátor, $y = g^*(x)$ zobrazuje vstup x na kategorii y. FNN definuje zobrazení $y = g(x; \theta)$ a učí se hodnoty parametrů θ, které vedou k nejlepší aproximaci.“* [11, str. 164]

„Slovo ‚network‘ v názvu označuje, že jsou obvykle reprezentovány jako spojení několika různých funkcí. Model reprezentuje acyklický orientovaný graf popisující způsob spojení jednotlivých funkcí. Například spojení funkcí $g^{(1)}$, $g^{(2)}$ a $g^{(3)}$ je výsledná funkce $g(x) = g^{(3)}(g^{(2)}(g^{(1)}(x)))$. Jednotlivým funkcím se říká vrstvy sítě. Název hluboké strojové učení vychází z počtu zřetězených funkcí neboli hloubky sítě. Poslední vrstva sítě je nazývána výstupní.“ [11, str. 164-165]

„Během trénování sítě je cílem přiblížit výstup funkce $g(x)$ výstupu $g^*(x)$. Trénovací data udávají šumem zatíženou aproximaci případů vyhodnocení funkce $g^*(x)$ v různých bodech. Výstupní vrstva tedy musí pro každý vstup x produkovat hodnotu co nejblíže $g^*(x)$. Vrstvám, které nejsou výstupní se nazývají skryté, protože trénovací data neurčují jejich požadované hodnoty výstupů.“ [11, str. 165]

Neuronová v názvu znamená, že tento model je volně inspirovaný biologickými neurony. Každá skrytá vrstva sítě je typicky vektor. V praxi jsou vrstvy reprezentovány vektory. Jednotlivé vrstvy si lze představit tvořené menšími jednotkami, které z vektoru produkují skalár. Tato jednotka pak připomíná neuron, kdy z více vstupů spočítá výstup za pomocí aktivační funkce. [11, str. 165]

Pro představu je zde uvedeno několik aktivačních funkcí. Pro skryté vrstvy se často využívá funkce ReLU[3], která svůj výstup jednoduše omezuje na hodnoty větší nebo rovné nule. Tohoto chování je docíleno pomocí funkce *max* s argumenty 0 a vstupním parametrem x jak je uvedeno ve vzorci 3.1.

$$\mathrm{ReLU}(x) = \max(0, x) \tag{3.1}$$

Funkce sigmoid s předpisem 3.2 se využívá pro binární klasifikaci a jejím výstupem je hodnota v rozsahu $(0, 1)$. 3.2.

$$\mathrm{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

Pro výpočet distribuce pravděpodobností v rámci klasifikace do K tříd se využívá funkce softmax, která ze vstupního vektoru $x = (x_1, x_2, \ldots, x_K)$ vytvoří vektor, jehož součet prvků je rovný jedné a dá se interpretovat jako distribuce pravděpodobností. Její předpis pro jednotlivé prvky vektoru je zapsán vzorcem 3.3 níže.

$$\mathrm{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{3.3}$$

Nosnou myšlenkou těchto sítí je vytváření nových příznaků ze vstupních dat, čehož je docíleno v již zmíněných skrytých vrstvách. Každý příznak je obvykle lineární kombinací výstupů z předchozí vrstvy s přidaným posunem[4] a následnou aplikací nelineární aktivační funkce, které musí být diferencovatelná[5].

Proces učení sítě má za cíl nalezení optimálních hodnot posunu a vah jednotlivých vrstev. V případě nevyužití nelinearity by bylo možné síť zjednodušit na jednu operaci maticového násobení a její výsledky by tím pádem byly značně omezeny [11, str. 168].

[2] Artificial Neural Network (ANN).
[3] Rectified linear unit.
[4] Někdy také nazývaný intercept či bias.
[5] Nebo alespoň skoro všude diferencovatelná.

3.3 Učení na základě gradientu

Na rozdíl od lineárních modelů nelinearita uvnitř FNN způsobí, že ztrátová funkce se stává nekonvexní. Učení tedy probíhá iterativním procesem na základě gradientu. Pro FNN je třeba nejdříve inicializovat váhy malými náhodnými hodnotami a posuny mohou být vynulovány. [11, str. 173]

Náhodné hodnoty způsobí, že se každá váha podílí na výsledku mírně odlišným způsobem, což později vede k postupnému vytvoření reprezentací nových příznaků uvnitř sítě.

Důležitá je volba ztrátové funkce, která závisí na typu řešené úlohy. V případě regrese je to střední kvadratická chyba[6] jejíž vzorec 3.4 je prostý rozdíl hodnot umocněný dvěma.

$$L(Y, \hat{Y}) = (Y - \hat{Y})^2 \tag{3.4}$$

Pro binární klasifikaci se používá binární relativní entropie[7], kde $\hat{p} = \hat{P}(Y = 1|X = 1)$:

$$L(Y, \hat{p}) = -Y \log \hat{p} - (1 - Y) \log(1 - \hat{p}) \tag{3.5}$$

V případě klasifikace do více tříd je využito kategorické relativní entropie[8], kde $\hat{p}_i = \hat{P}(Y = i|X = X)$, $\hat{p} = (\hat{p}_1, \ldots, \hat{p}_c)^T$ a $\mathbb{1}_{Y=j} = 1$ když $Y = j$ v opačném případě $\mathbb{1}_{Y=J} = 0$. Vzorec pro kategorickou entropii je pak:

$$L(Y, \hat{p}) = -\sum_{j=1}^{c} \mathbb{1}_{Y=j} \log \hat{p}_j = -\log \hat{p}_Y \tag{3.6}$$

Při učení je cílem minimalizovat chybu predikce měřené touto ztrátovou funkcí L na trénovací množině velikosti N, kde w jsou parametry sítě, které se vyskytují ve funkci g:

$$J(w) = \frac{1}{N} \sum_{i=1}^{N} L(Y_i, g(x_i; w)) \tag{3.7}$$

Pro nalezení minima za pomocí vylepšené varianty gradientního sestupu je využito pravidla derivace složené vícehodnotové funkce, které je přímím zobecněním vztahu pro výpočet derivace složené funkce. Pro $f : \mathbb{R}^n \longrightarrow \mathbb{R}$, $g : \mathbb{R} \longrightarrow \mathbb{R}^n$, kde $g = g(g_1, \ldots, g_n)^T$:

$$\frac{\partial f \circ g}{\partial x}(x) = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(g(x)) \frac{\partial g_i}{\partial x}(x) \tag{3.8}$$

Směrem od poslední vrstvy se postupně pronásobují a sčítají derivace, které se dále propagují směrem ke vstupní vrstvě. Následně jsou dle směru gradientu upraveny váhy sítě, ve snaze celkovou chybu minimalizovat. [12]

Celé trénovací množině se říká batch, česky dávka, a postupu, kdy je síť optimalizována až po průchodu všemi vzorky z trénovací množiny se říká batch gradient descent. Obvykle se ale změna vah aplikuje na základě průchodu menšího množství vzorků trénovací množiny, takzvané mini-batch. V praxi se nicméně často mini-batch označuje pouze výrazem batch, protože se změna vah na základě celé trénovací množiny příliš nepoužívá, z důvodu pomalé konvergence. Z tohoto důvodu bude v rámci práce používán pojem dávka jakožto označení pro mini-batch. Pro úplnost: případ, kdy ke změně vah dochází na základě jednoho vzorku se nazývá stochastic gradient descent.

[6] Mean squared error.
[7] Binary cross-entropy.
[8] Categorical cross-entropy.

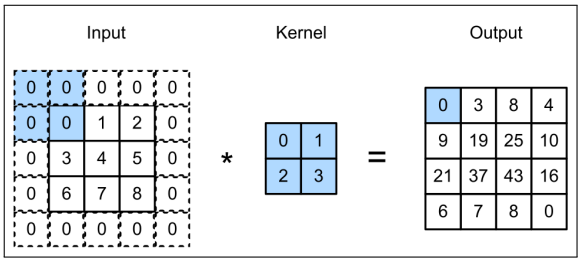
3.4 Konvoluční neuronová síť

Konvoluční neuronová síť, anglicky convolutional neural network (CNN), je speciálním případem ANN, kde je snížen počet vah mezi vrstvami a slouží pro zpracování dat v mřížkovém uspořádání [11, str. 327]. Může se jednat například o časové řady, kde je mřížka jednorozměrná nebo obrázky, které jsou dvojrozměrné v případě černobílých obrázků a trojrozměrné, pokud jsou obrázky barevné.

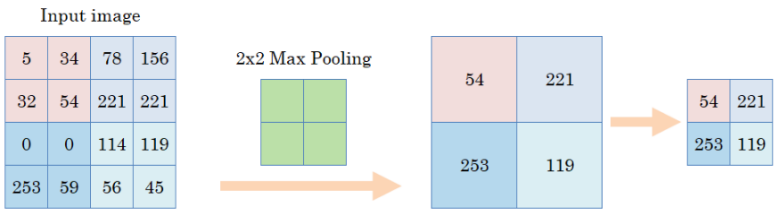
Uspořádání vah pro tuto lineární kombinaci je nazýváno jádro (kernel) a jeho obvyklé rozměry jsou 3×3 nebo 5×5. Jádro mění svou pozici na vstupní mřížce a na každém místě posunu je proveden skalární součin. Výstup je zaznamenán do výstupního tensoru, takovým způsobem, že pozice jádra na vstupní mřížce je analogicky promítána na výstupní. Váhy jsou po celou dobu tohoto procesu pro všechny pozice stejné. V rámci pohybu po vstupní mřížce se mohou uvažovat pouze takové pozice, kdy jádro nepřesahuje okraje mřížky, případně může být mřížka vhodně rozšířena[9]. Na obrázku 3.1 lze vidět vizualizace konvoluce, kde je znázorněna na dvojrozměrných datech. Pro RGB obrázky a další typy dat se postupuje analogicky, s rozdílem, že filtr má zvlášť váhy pro každý kanál. Jader je obvykle více a obvykle se počet zvyšuje s hloubkou vrstvy v síti. Existují také různé úpravy, kde jsou filtry rozděleny do skupin, které ignorují některé kanály.

Z důvodu malých rozměrů má jádro schopnost detekovat pouze lokální příznaky. Proto se dále používá takzvaný pooling, kdy se data rozdělí na malé čtverce a z každého z nich se spočítá jedna výstupní hodnota. Tu lze vyjádřit buďto jako průměr nebo největší z původních. Vstup pro další vrstvu se tím pádem zmenší a další filtr již pracuje s méně detailním obrázkem.

Intuice pro CNN by mohla být, že první vrstva detekuje detaily jako hrany, ale hlubší vrstvy již nepracují s původním obrázkem a barvami, nýbrž s příznaky, které jsou pro člověka často neuchopitelné. Počet kanálů se zvyšuje s hloubkou a v poslední vrstvě může mít mřížka rozměr například $1 \times 1 \times$ počet kanálů.



Obrázek 3.1 Vizualizace vstupu, jádra a výstupu konvoluce. Hodnoty výstupu jsou produkovány skalárními součiny čtveřic vstupů a jádra (zdroj: [13])



Obrázek 3.2 Ukázka max poolingu, při kterém je pro výstup vybrána nejvyšší hodnota ze zdrojové matice a ostatní jsou ignorovány (zdroj: [14])

[9] Například nejbližším prvkem z mřížky či nulovou hodnotou.

3.5 Dávková normalizace

Dávková normalizace, anglicky batch normalization (BN), má za cíl zabránit nevhodné vstupní distribuci hodnot vstupů aktivačních funkcí uvnitř sítě.

V praxi je implementována jako vrstva, která má snahu transformovat své vstupy způsobem, který zaručí, aby průměrná hodnota jednotlivých příznaků vstupu aktivační funkce nebyla příliš vzdálená od 0 a rozptyl nebyl neúměrně velký vůči vahám sítě, které by musely tyto hodnoty kompenzovat, což značně znemožňuje učení.

Při využití s konvolučními vrstvami se toto provádí v rámci skupin skalárů, představujících jednotlivé kanály. Například při aplikaci BN na obrázek je zvlášť normalizován kanál červené, modré a zelené barvy[10].

Jako naivní řešení tohoto problému se nabízí přímé využití normalizace, které zařídí průměrnou hodnotu 0 a rozptyl 1. Problémem je nutnost znalosti distribuce výstupních hodnot vrstvy, která se v síti nenachází před BN vrstvou, ta se ale během trénování může neustále měnit. Také rozptyl rovný jedné by omezil možnosti sítě reprezentovat některé problémy. [15]

Vstupy jsou nejdříve znormalizovány dle odhadu průměru a rozptylu na základě hodnot dávky. Výsledné hodnoty jsou dále za využití trénovatelných parametrů β pro posun a γ pro škálování upraveny. V případě, že se jedná o BN pro konvoluční vrstvu, používá se jiný pár parametrů pro každý kanál. Celý výpočet je popsán vzorcem 3.9, kde x jsou hodnoty vstupů v rámci dávky případně dále omezeny na jeden konkrétní kanál. [15]

$$y_i = \gamma \frac{x_i - E[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} + \beta \tag{3.9}$$

Parametry β a γ se během trénování mohou měnit podle potřeb sítě, nicméně by se nemělo stát, že rozptyl bude neúměrně velký. Při inferenci jsou místo výpočtu odhadů rozptylu a průměru použity hodnoty z trénovacích dat. [15]

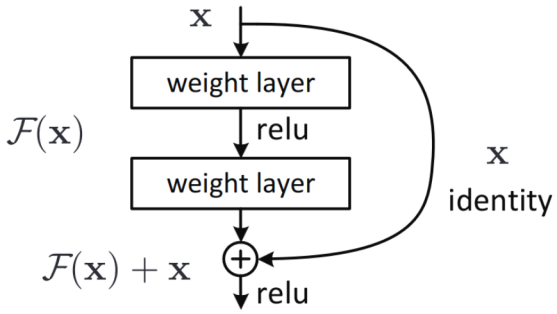
3.6 Zbytkové spoje

Zbytkové spoje, anglicky residual connections, napomáhají toku gradientu při zpětné propagaci a schopnosti vrstev kopírovat vstup bez vynucených změn způsobených filtry konvolučních vrstev [16].

Na obrázku 3.3 lze vidět blok sítě ResNet, v jehož rámci byla tato technika představena. ResNet obsahuje desítky konvolučních vrstev, což značně ztěžuje proces trénování.

Tato technika je elegantní z pohledu nízké výpočetní náročnosti, nicméně dochází ke zvýšení využití paměti, protože po dobu do přičtení identity musí být její hodnota uložena v paměti [16].

[10] Jedná se pouze o příklad, protože BN vrstvy se nenachází přímo za vstupem sítě.

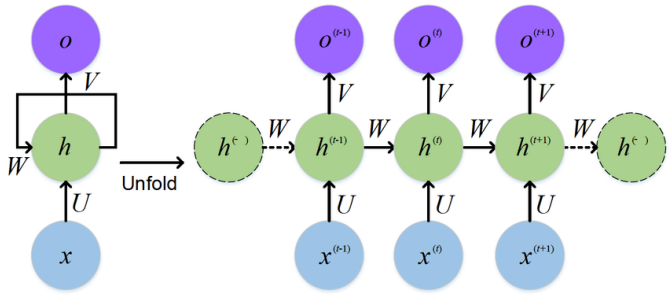


Obrázek 3.3 Ukázka bloku sítě ResNet, využívajícího zbytková spojení. Před aplikací vrstvy vah je vstup beze změn uložen do paměti a následně přičten k výstupu poslední vrstvy před aktivační funkcí (zdroj: [16])

3.7 Rekurentní neuronová síť

Rekurentní neuronové sítě (zkráceně RNN) jsou využívány pro práci se sekvenčními daty s variabilní délkou. RNN obsahuje paměť reprezentující interní stav, ve kterém se nacházejí zakódované informace o dosud zpracované části sekvence, která je na vstup vkládána po částech. Při zpracování každé části vstupu síť dokáže produkovat výstup a novou hodnotu tohoto stavu, kterou použije jako vstupní skrytý stav do dalšího kroku. V prvním kroku je stav inicializovaný na hodnoty v závislosti na konkrétní síti. Schématický popis RNN lze vidět na obrázku 3.4.

Problémem RNN je takzvaný vanishing gradient, který je způsoben postupným snížením podílu informace z předešlých iterací na hodnotě ztrátové funkce a s tím spojeným nízkým podílem na hodnotě gradientu. Toto vede k omezeným možnostem sítě adekvátně upravit váhy a možnosti efektivně využít rané hodnoty vstupu. V praxi to znamená, že model může v dlouhém souvětí či odstavci ztratit možnost držet informaci, která byla součástí počátku sekvence. Tuto omezenou paměť předešlých hodnot částečně řeší Long short-term memory (LSTM) architektura, nicméně nutnost rekurentního průchodu zůstává.



Obrázek 3.4 Nalevo se nachází reprezentace rekurentní buňky. Napravo je naznačen proces rekurentního průchodu sítí, kdy dochází k předávání výstupu a propagace nového skrytého stavu do dalšího kroku (zdroj: [17]).

3.8 Transformer

Transformer je architektura poprvé popsána v článku Attention is All You Need [18]. Místo rekurentního přístupu jsou vstupy zpracovány naráz a informace o pozici je číselně zakódována. To eliminuje nutnost zpracovávání vstupu po jednotlivých krocích, což umožňuje rychlejší trénování sítě. Tento přístup způsobil rapidní vývoj dalších metod, na jejichž základech staví dnes

populární modely typu ChatGPT [19].

Architekturu rozdělenou na část enkodéru a část dekodéru, uvedenou v původním článku, lze vidět na obrázku 3.5.

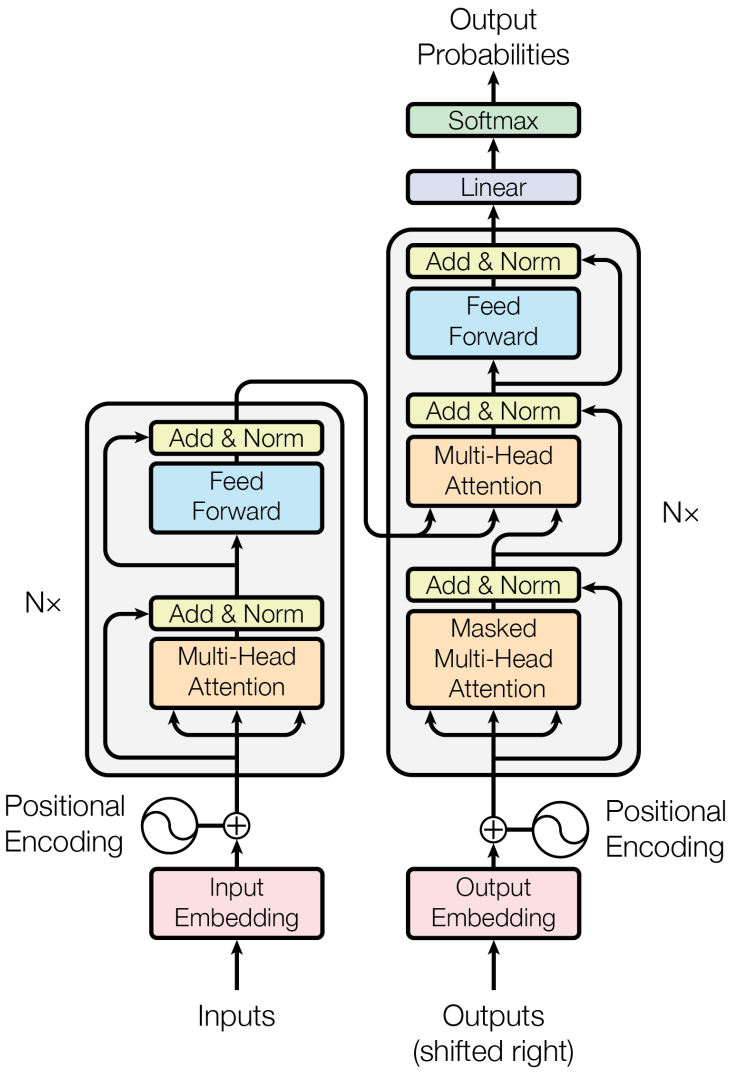
Vstupem enkodéru je sekvence tokenů, které jsou za pomocí embeddingu převedeny do interní vektorové reprezentace sítě. Dále je přidáno poziční kódování, nahrazující dosud používaný rekurentní přístup, v rámci kterého byly části vstupu zpracovávány jeden po druhém. Poziční kódování využívá vlastností goniometrických funkcí, přesněji rozdílných hodnot těchto funkcí na základě různých velikostí periody, aby model měl možnost odvodit, v jaké vzdálenosti jsou tokeny umístěny ve vstupní sekvenci.

Samotné tokeny jsou ze vstupní sekvence vytvořeny v rámci předzpracování. Obvykle se nejedná o celá slova vstupního jazyka, jelikož by takový slovník mohl dosáhnout příliš velkých rozměrů a při využití pro překlad by bylo třeba dvou samostatných slovníků zvlášť pro vstupní i výstupní jazyk. Také je sníženo riziko, že daný token nebude obsažený v trénovacích datech a v případech, kdy se na vstupu nachází jména osob či firem by měl model v ideálním případě tyto tokeny pouze překopírovat na výstup bez dalších úprav. Tokeny lze vytvořit více způsoby, ale obecně se jedná o krátké řetězce, které se v uvnitř zpracovávaných slov často vyskytují. Celkově tokeny neobsahují mezery, aby nevynucovaly návaznost určitých slov.

V rámci dekodéru se pro identifikaci vzájemných spojitostí mezi tokeny využívá takzvané metody self-attention, která funguje na principu vyhledávání pravděpodobně vzájemně provázaných tokenů. Tento proces se odehrává ve vnitřní reprezentaci dat uvnitř modelu, ale k pochopení principu lze využít člověkem snáze uchopitelné interpretace, kdy například tokeny podmětu omezují svou pozornost na takové, které se tváří jako přísudek či větný předmět. To samé platí i opačně a přísudek naopak zvyšuje váhu tokenů, které jeví znaky podmětu. Následně jsou hodnoty tokenů, které se zdají relevantní překopírovány k původnímu a nově vzniklý kontextu umožní spojit informace z dvou odlišných částí věty, což je nutné pro korektní překlad do jazyka, který má opačný slovosled než jazyk vstupní. Této technice se říká se anglický říká self-attention[11], protože v uvnitř konkrétní sekvence dochází k rozpoznání vazeb mezi tokeny. Obvykle model vybírá z více možností propojení tokenů, tedy využívá takzvanou multi-head attention. Inference končí v momentě, kdy je na vstup předán speciální token pro ukončení.

Tvorba embeddingu je na rozdíl od deterministického procesu tokenizace závislá na schopnostech generalizace sítě, tedy jejímu stupni natrénování. V případě kvalitně natrénovaného modelu pak embeddingy dobře reprezentují význam takto zpracované sekvence a lze jej využít například pro vyhledávání textů v indexové databázi, kde je využito porovnání obvykle na základě kosinové podobnosti těchto vektorů.

[11] Attention česky znamená pozornost.



Obrázek 3.5 Architektura sítě transformer. Vlevo se nachází enkodér, který vstupní sekvenci zakóduje do interní reprezentace a následně předá do dekodéru, který začne produkovat výstup.

Kapitola 4

Analýza stávajících metod

V této kapitole bude nejprve provedena analýza datasetů zabývajících se strukturou dokumentu, což umožní jasnější představu o rozsahu a typu dostupných dat pro budoucí model. Dále bude věnován prostor samotným modelům, které často byly vytvořeny v rámci práce na konkrétním datasetu [20, 21, 22]. Na základě těchto informací bude nastíněno další řešení.

4.1 Existující Datasety

Tato sekce se zabývá již existujícími datasety, určenými k tvorbě modelů pro zpracování dokumentů. Jelikož je velká část výpisu tvořena převážně tabulkou, byl prostor věnován hlavně takovým, které se tabulkami zabývají.

Již ze samotné podstaty byznysových dokumentů plyne nedostatek veřejně dostupných datasetů obsahující tento typ dokumentů [22]. Publikace zabývající se zpracováním byznysových dokumentů často využívají soukromé datasety, což vede k nemožné reprodukci experimentů a ověření funkčnosti daného řešení. Toto je způsobeno absencí dostatečně velkého a veřejně dostupného datasetu dané domény [5].

Anotace je proces, při kterém se vyznačí jednotlivé objekty v obvykle nestrukturovaných datech, aby model měl možnost „pochopit“, co je žádaný výsledek [23]. V kontextu této práce anotace může znamenat například vyznačení čísla účtu na výpisu, či označení jednotlivých transakcí.

Tento proces je obecně velice časově náročný [24, 25] a často tvoří většinu doby strávené na řešení úlohy strojovým učením. Některé zdroje [25] uvádějí, že anotování dat obvykle zabere 25 % času stráveném na projektu, zaměřeného na strojové učení. Samotná tvorba modelu oproti tomu „pouze“ 20 % [25]. Celý proces je také náchylný k lidským chybám a nekonzistentnímu rozhodování během procesu [26].

Kvůli vysoké náročnosti tvorby vlastního datasetu byly prozkoumány již existující, volně dostupné datasety, které se sice často zabývají jinými doménami, ale přesto mohou pomoci ujasnit celkovou představu o existujících metodách.

4.1.1 PubTabNet

▶ **Definice 4.1** (BBox)**.** *Bounding box objektu (zkráceně BBox) je obdélník, který celý objekt svými rozměry obklopuje [27].*

Jedná se o volně dostupný dataset vytvořený na základě odborných medicínských článků

z PubMed Central Open Access Subset[1]. Při tvorbě bylo využito skutečnosti, že články jsou k dispozici jak v PDF formátu, tak formátu XML, který je strukturovaný. Části PDF byly propojeny s částmi v XML reprezentaci, čímž byla rozpoznána hierarchická struktura uvnitř PDF. Dataset obsahuje 510 tisíc obrázků tabulek spolu s texty uvnitř buněk ve formátu HTML [21]. V datasetu nejsou dodatečné informace týkající se dané domény. Později byly přidány BBoxy neprázdných buněk [28, 22]. V praxi se omezení na neprázdné buňky ukázalo jako nevýhoda [20], případně dokonce chyba [29]. Ze zkoumaných vzorků byl vyvozen závěr, že z HTML struktury lze u jednotlivých vzorků odvodit, které buňky tvoří hlavičku a které hodnotu pole.

Způsob vytvoření tohoto datasetu dokládá, jak moc je důležitá archivace i původních strukturovaných dat, která se případným převodem na nestrukturovaná stávají mnohem méně použitelná. Pozitivní skutečností je, že se články a další výstupy vědecké práce postupně čím dál více zveřejňují i ve strukturovaných formátech [30].

▶ **Definice 4.2** (TD). *Table Detection je proces s cílem detekce a lokalizace tabulky v dokumentu nebo obrázku [29].*

▶ **Definice 4.3** (TSR). *Table Structure Recognition je úloha představující rozdělení tabulky na buňky, sloupce, řádky a identifikaci buněk, které náleží do více sloupců nebo řádků [29].*

▶ **Definice 4.4** (FA). *Functional Analysis je úloha rozpoznání klíčů a hodnot v rámci tabulky [29].*

▶ **Definice 4.5** (TE). *Table Extraction je kombinace TD, TSR a FA [29].*

Protože zdrojem pro vytvoření tohoto datasetu byla medicínská data, není žádný doménový překryv s tématem práce, kromě faktu, že se v obou případech jedná o data zapsaná do tabulek. Při vizuální kontrole menšího vzorku tabulek tak nemohlo být překvapením, že se ani vzdáleně nepodobají výpisům. V datasetu není kompletně zohledněna úloha TE, pouze její dílčí část TD a částečě TSR, v rámci které chybí souřadnice a rozměry buněk.

4.1.2 FinTabNet

FinTabNet vnikl v rámci práce na modelu Global Table Extractor [22]. Kromě toho došlo také k rozšíření PubTabNet datasetu o vyznačení rozměrů jednotlivých buněk. Opět bylo využito propojení částí nestrukturovaného dokumentu se strukturovaným.

Samotný FinTabNet obsahuje finanční zprávy společností z amerického indexu S&P 500. Vzhledově se tabulky od těch v PubTabNet liší různými způsoby, mají méně čárových oddělovačů, větší rozestupy uvnitř tabulky a větší barevnou různorodost[2]. Dataset obsahuje přes 70 tisíc stran s anotacemi tabulky a její vnitřní struktury. Dále obsahuje 110 tisíc tabulek s BBoxy textů[3] buněk [22].

Stejně jako v případě PubTabNet platí, že v datasetu nejsou zohledněny významy textů uvnitř buněk, stejně tak nejsou přiřazeny alespoň obecné třídy jako například datum, částka atd. Navíc není rozlišováno mezi buňkami hodnoty a hlavičky, což lze vypozorovat i z online ukázky [4].

Dataset obsahuje velké množství dat, které se více blíží tématu této práce. Tabulky se také o trochu více podobají výpisům, protože jednotlivé LI zahrnují větší množství řádků, mezi kterými nejsou vizuální oddělovače. Při exploraci bylo bohužel zjištěno, že velká část tabulek neobsahuje jednotlivé LI, kde by se opakoval stejný typ objektu, ale spíše celá tabulka představovala výčet několika unikátních informací. Velkým negativem datasetu je, že neumožňuje explicitní určení, zda se jedná o název nebo hodnotu pole, tedy nezohledňuje úlohu FA, což velmi snižuje využitelnost. Nakonec je nutné připomenout, že formáty zápisů částek a kalendářních dat se samozřejmě neshodují s českými.

[1] https://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/

[2] Při zběžné ruční exploraci PubTabNet nebyla dokonce tabulka využívající barvy vůbec nalezena.

[3] Tedy neberou v potaz prázdné místo uvnitř buňky, stejně jako v případě PubTabNet.

[4] `https://dax-cdn.cdn.appdomain.cloud/dax-fintabnet/1.0.0/data_preview/index.html`

4.1.3 TableBank

Zdrojem podkladových dat byly dokumenty z internetu. Opět zde bylo využito strukturovaných formátů, konkrétně `.docx`, který interně využívá XML reprezentaci struktury a v dalších případech bylo využito zdrojových souborů LaTeX [31].

Dataset obsahuje přes 145 tisíc instancí tabulky. Na rozdíl od předchozích zmíněných neobsahuje ani BBoxy textů buněk, což je spolu s nízkou komplexností tabulek datasetu vytýkáno [20].

Protože dataset není přímo doménově či tematicky zaměřený, obsahuje více různorodé dokumenty jak strukturou, tak užitým jazykem. Kromě angličtiny, se v nich vyskytuje i čínština, japonština, arabština a dalších jazyky. LaTeX zdrojové kódy pocházejí z arXiv[5] a jsou psány převážně anglicky. [31]

4.1.4 PubTables-1M

Jak již název napovídá dataset obsahuje téměř milion vzorků a je po vzoru PubTabNet založen na datech PubMed Central Open Access [6].

Autoři jako motivaci vzniku uvádějí problémy, které se váží na použití zmíněné metody tvorby kombinace strukturovaných a nestrukturovaných dat u předchozích datasetů [29]. Hlavním cílem je odvození maximálního množství informací, které v předchozích datasetech nejsou zohledněny, aby výsledný dataset mohl být použit pro tvorbu většího množství úloh než předchozí řešení. Druhým cílem je zlepšit přesnost anotací za použití deterministických algoritmů předzpracování. V datasetu nejsou vyznačeny žádné třídy informací či relace.

Velkým problémem jsou nekorektní anotace v případech, kdy určitá buňka zabírá oblast velikosti několika běžných buněk (viz obrázek 4.3a). Tento problém je v předchozích datasetech obcházen vyznačováním BBoxu pouze pro texty buňky, což vede k možnosti nesprávného rozpoznání struktury, kdy rozsah platnosti názvu sloupce či řádku je chybně interpretován [29].

Při anotaci sloupců tabulky člověkem nebývá sledován BBox textu názvu sloupce, ale spíše se přihlíží k volnému místu a zarovnání textů v hodnotách polí pod ním.

Tyto nesprávné anotace narušují předpoklad, že model se učí na datech, která jsou naprosto pravdivá [29]. Důležitým poznatkem je, že podstata těchto chyb nemusí být náhodným šumem, ale chyby mohou být způsobeny povahou domény či zkreslením v rozhodování jednotlivců, v případě ručních anotací. Tyto chybné informace se model naučí na základě trénovací množiny a dále je zohledňuje při inferenci [32]. Jelikož se chyby stejné povahy nacházejí i ve validační a testovací množině, je zkreslené i vyhodnocení výsledků modelu [29, 32].

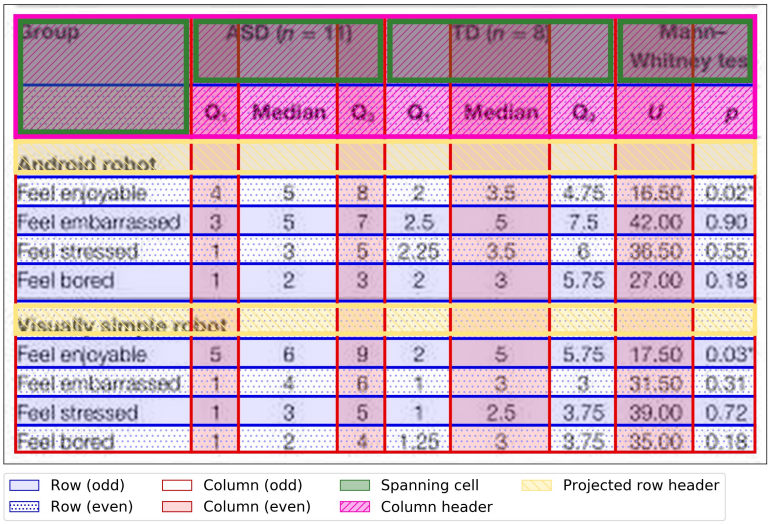
V případech, kdy jsou prázdné buňky nesprávně děleny je nazýván „oversegmentation“ [29]. Jako řešení tohoto problému je v rámci PubTables-1M vytvořen algoritmus pro slučování buněk tabulky. Tento algoritmus byl vytvořen na základě předpokladů z Wang[7] modelu tabulky [33] a neobsahuje žádné metody strojového učení. Dle autora tento model dodržuje 97 % jím zkoumaných tabulek [33]. Na obrázku 4.3a lze vidět příklad přílišného dělení buněk tabulky, na obrázku 4.3b pak vizualizaci anotace stejné tabulky po úpravě zmíněným algoritmem.

Z důvodů velikosti tohoto datasetu nebylo možné zkontrolovat vzorky ručně, proto byla vytvořena automatizovaná kontrola kvality. V případech jasného porušení požadavků na kvalitu je tabulka ignorována, jedná se o případ, kdy například anotace obsahuje navzájem překrývající se buňky. Dále je zkoumána editační vzdálenost řetězců uvnitř buněk anotací a řetězců z PDF, kde je určena míra tolerance, pro případy dělení slov na konci řádku atd. Po jednotlivých kontrolách jsou odstraněny neobvyklé tabulky, které vybočují z distribuce vzorků [29].

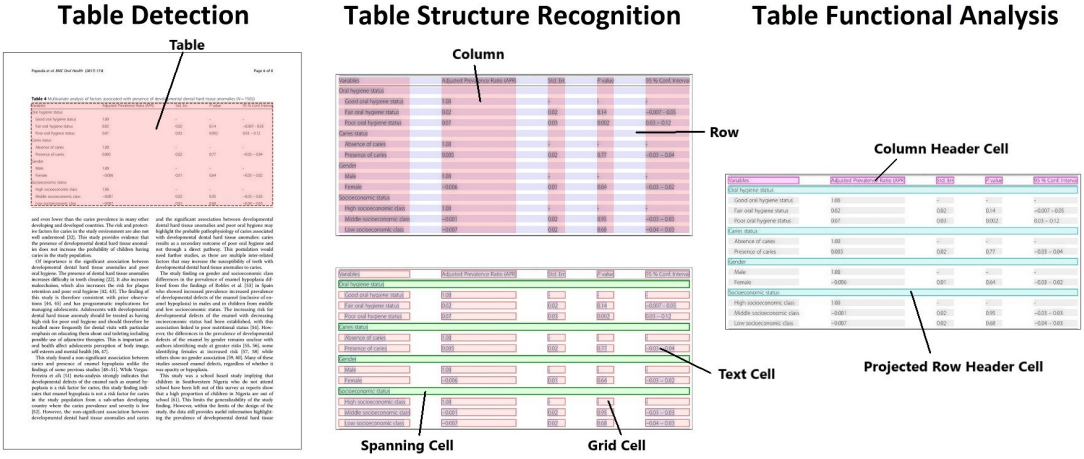
Oproti již zmíněným datasetům PubTables-1M obsahuje přomo BBoxy všech buněk a neomezuje se tak pouze na texty. Dále obsahuje již zmíněnou očištěnou tabulku a explicitní informace

[5] `https://arxiv.org/`
[6] https://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/
[7] Z dostupných zdrojů se nepodařilo zjistit pohlaví potřebné pro správné skloňování.

o poloze jednotlivých řádků a sloupců, včetně BBoxů. Je rozlišováno mezi buňkami, které zabírají obvyklý prostor a těmi, které zabírají například celý řádek. Velkým přínosem je identifikace takzvaných „projected rows“ [34] nebo také „super rows“ [35], které zabírají celý horizontální prostor a určují společnou závislost řádků pod nimi. Ukázku všech tříd buněk lze vidět na obrázku 4.1. Celkově tyto informace umožňují lépe pokrýt jednotlivé dílčí cíle TE, než existující datasety. Všechny kroky nutné ke dosažení TE jsou znázorněny na obrázku 4.2. Ani v tomto případě nejsou informace v buňkách děleny do tříd, takže není možné použít dataset například pro tvorbu detektoru částek či kalendářních dat.



Obrázek 4.1 Ukázka výskytů veškerých tříd v rámci PubTables-1M (zdroj: [29])



Obrázek 4.2 Ukázka rozdělení jednotlivých dílčích úloh TE (zdroj: [29])

		ΔSDM			
		better	equal	Worse	Sum
ΔSCA	better	19457 (28.9)	12 (0.02)	14654 (21.8)	**34,123 (50.8)**
	equal	1158 (1.7)	21989 (32.7)	1024 (1.5)	**24,171 (36.0)**
	worse	3755 (5.6)	2 (0.003)	5183 (7.7)	**8,940 (13.2)**
	Sum	**24370 (36.2)**	**22003 (32.7)**	**20861 (31.0)**	**67,234 (100.0)**

(a) Původní tabulka před použitím algoritmu na slučování buněk (zdroj: [29])

		ΔSDM			
		better	equal	Worse	Sum
ΔSCA	better	19457 (28.9)	12 (0.02)	14654 (21.8)	**34,123 (50.8)**
	equal	1158 (1.7)	21989 (32.7)	1024 (1.5)	**24,171 (36.0)**
	worse	3755 (5.6)	2 (0.003)	5183 (7.7)	**8,940 (13.2)**
	Sum	**24370 (36.2)**	**22003 (32.7)**	**20861 (31.0)**	**67,234 (100.0)**

(b) Tabulka po úpravě algoritmem pro sloučení prázdných buněk (zdroj: [29])

Obrázek 4.3 Porovnání tabulky před a po sloučení buněk algoritmem

4.1.5 SynthTabNet

Dataset vznikal pro účely tvorby modelu TableFormer [20], a jak již název napovídá, vznikl z vygenerovaných syntetických dat. Autoři se do velké míry inspirovali z nesyntetických datasetů PubTabNet, FinTabNet a TableBank, u kterých identifikovali různé nedostatky, které nejsou v žádném z nich vyřešeny. V rámci analýzy již existujících datasetů bylo například zjištěno, že distribuce komplexity tabulek je značně nevyvážená, protože silně převládají jednodušší tabulky [20].

V rámci tvorby syntetického datasetu a modelu je zmíněn problém nejednotného formátu existujících datasetů. Z důvodu velikosti dat byly ostatní datasety převedeny do formátu shodného s PubTabNet. Ze všech byly odstraněny příliš velké tabulky a hustoty pixelů podkladových obrázků byly sjednoceny na 72 DPI [8]. Nassar a spol. dále zmiňuje problematiku datasetu TableBank, který v naprosté většině případů postrádá BBoxy textů buněk a jeho tabulky jsou autory považovány za jednoduché, protože kromě jejich nízké velikosti navíc neobsahují řádky a sloupce rozšířených rozměrů. [20]

Výhodou syntetických dat je možnost různých změn stylu vzhledu, velikosti tabulky, a tedy i komplexity struktury. Texty pro výplň buněk jsou generovány na základě statistické analýzy slov původních datasetů. Hlavní výhodou generování je nesporně možnost vytvořit opravdu velký dataset, v tomto případě se jedná o 600 tisíc vzorků [20]. Využití samotných syntetických informací v buňkách je nicméně velice omezené až téměř nulové.

4.1.6 DocLayNet

Dataset obsahuje 80 tisíc ručně anotovaných stránek PDF dokumentů, z kterých 7 tisíc obsahuje 2 anotace (od dvou osob) a tisíc a půl obsahuje tři anotace. Třídami v dokumentech jsou popis, poznámka pod čarou, vzorec, prvek výčtu, hlavička a patička stránky, obrázek, nadpis sekce, tabulka, text a nadpis. Data jsou vytvořena z veřejně dostupných dat jako jsou finanční zprávy, manuály, vědecké články, zákony a regulace, patenty a vládní tendry. Dataset není přímo zaměřen na určitý jazyk, ale z 95 % je tvořen angličtinou. Skeny dokumentů, které často nejsou dokonale rovné, byly v co největším možném množství případů ignorovány, aby pro usnadnění anotace bylo možné používat obdélníky bez rotace. [36].

Pro samotnou tvorbu byly nejdříve dokumenty předzpracovány, určeny třídy, pravidla a postupy, pro zvýšení úrovně konzistence mezi anotujícími zaměstnanci, kteří se také podrobili zkouškám, pro kontrolu kvality jejich anotací [36].

Dataset se bohužel pro účely práce ukázal jako nedostatečný, protože není zaměřen na třídy textových informací ale pouze na objekty dokumentu.

[8] Tedy standardní hustotu pixelů knihovny Poppler.

4.1.7 Shrnutí analýzy existujících datasetů

Z volně dostupných datasetů byly prozkoumáni kandidáti v oblasti práce s tabulkami, případně zaměřující se na strukturu obecných dokumentů. Datasety byly navzájem porovnány z pohledu velikosti, druhů informací uvnitř vzorků, způsobu tvorby a kvality dat.

Hlavním identifikovaným problémem je podezření, že některé strojově vytvořené anotace jsou nesprávné. Ve strojově generovaných datasetech sice dochází k výraznému usnadnění práce a velkému nárůstu počtu vzorků, ale již nedochází k tak pečlivé kontrole expertem [20]. V rámci některých [29][20] datasetů bylo za účelem kontroly kvality použito různých automatizovaných filtrů a měření. U jiných je kontrola založena na manuálním zkoumání menšího množství vzorků [22, 28], či porovnáním anotací pocházejících od více různých lidí [36].

Vzhledem k tomu, že většina datasetů obsahuje pouze základní informace vytvořené ze zdrojových souborů, se jako jediný dostatečně podrobný dataset jeví PubTables-1M, který původní informace ještě dále algoritmicky obohacuje. Bohužel je založený na medicínských článcích, které jsou bankovním výpisům značně vzdálené.

Z pohledu domény je tedy k tématu této práce nejblíže FinTabNet, případně podmnožina SynthTabNet. Oba tyto datasety ovšem postrádají detailnější informace pro plné splnění cíle TE.

Důležité je také zmínit, že v rámci průzkumu datasetů nebyl nalezen žádný, který zohledňuje problém názvu pole vedle hodnoty v rámci jednoho LI (viz obrázek 1.2). Vzhledem k poměrně vysoké četnosti výskytu tohoto jevu ve výpisech se již existující datasety pro toto využití zdají nevhodné.

4.2 Stávající metody strojového čtení dokumentů

Sekce zkoumá existující řešení pro úlohy podobné strojovému čtení bankovních výpisů z účtu. Krom samotných modelů jsou zmíněny i některé architektury, techniky a obecné principy, kterých je v modelech a jejich trénování využito. Závěrem je zhodnocení použitelnosti jednotlivých modelů pro účely práce.

Přístupů k řešení úloh podobných čtení dokumentů strojovým učením je celá řada a při volbě modelu je tak nutné zvážit hned několik faktorů.

Při výběru modelu je nejdůležitější formulace cíle úlohy a následná specifikace požadovaného výstupu, tím se upřesní, zda se jedná o úlohu regresní či klasifikační. V některých případech lze problém interpretovat a následně řešit oběma způsoby [37, 38].

Pro strojové učení je nesmírně důležitá kvalita datasetu, která v praxi znamená konzistentní anotace, rovnoměrné zastoupení tříd a dostatečnou rozmanitost vzorků. Naopak je nežádoucí výskyt anomálií, které mohou být např. způsobené chybným měřením či lidskou chybou při manuální anotaci. Dostatečně velký a kvalitní dataset by pak měl umožnit natrénovat model s dobrou schopností generalizovat [11] úlohu a tím pádem dosáhnout podobných výsledků jak na trénovací, tak na validační a testovací množině.

Je třeba zvolit takovou architekturu modelu, pro kterou již existují datasety nebo je nutné dataset vytvořit. V případě použití již existujícího datasetu, který naprosto přesně nereprezentuje doménu problému, je pak nutné vytvořit alespoň menší dataset pro validaci a testování. Výhodou takového přístupu je, že v případě úspěchu na testovacích datech model pravděpodobně opravdu generalizuje. V některých případech se ale trénovací data liší natolik, že nelze očekávat dobré finální výsledky.

Další možností je použít předtrénované řešení, případně ho do finálního modelu zakomponovat jako modul. Úspěšnost předtrénovaných řešení na nových datech závisí na velikosti a různorodosti jejich trénovacích dat, případně použití augmentace [39], dropout vrstev [40] a vrstev s batch normalization [15] pro zlepšení generalizace a zabránění přeučení. Při dodatečném trénování modelu s předtrénovanou částí hrají roli další faktory, a ne vždy musí být přímočaré [11]. Opět záleží, jak moc původní trénovací data podobná s daty úlohy.

Další možností je využití více modelů pro jednotlivé podúlohy celého problému. Zde vzniká komplikace, že ne vždy tyto modely mají stejné vstupy, případně vyžadují informaci, kterou není možné z dostupných dat získat.

▶ **Definice 4.6** (KIE)**.** *Key Information Extraction (dále KIE) je úloha extrakce hodnot vybraných klíčových informací z textu či obrázku [41]. V rámci úlohy nemusí být explicitně brán ohled na lokalizaci výskytu informace, která tedy ani nemusí být součástí výstupu modelu.*

▶ **Definice 4.7** (KILE)**.** *Key Information Localization and Extraction (dále KILE) v kontextu dokumentu označuje proces nalezení (například určením BBoxů) polí jednotlivých předdefinovaných kategorií*[9] *(klíčů) za účelem čtení a agregaci jejich hodnot pro extrakci klíčových informací každé kategorie. [5]*

▶ **Definice 4.8** (LIR)**.** *V kontextu obrázku stránky dokumentu nebo tabulky je cílem Line Item Recognition detekce všech přítomných LI, jejich klasifikace do tříd (například, objednávka, sleva atd.) a pro každý detekovaný LI lokalizace a extrakce jemu náležících klíčových informací. [5]*

V zadání práce je uvedeno, že dokumenty výpisů mají být převedeny ze své nestrukturované formy do jednotného strukturovaného formátu. Pro tento převod je nutné identifikovat jednotlivé klíčové informace z první strany dokumentu (případně dalších stran) (KIE 4.6). Dále je nutné detekovat tabulku operací a rozdělit ji na individualní LI z kterých jsou extrahovány jednotlivé hodnoty polí (LIR 4.8).

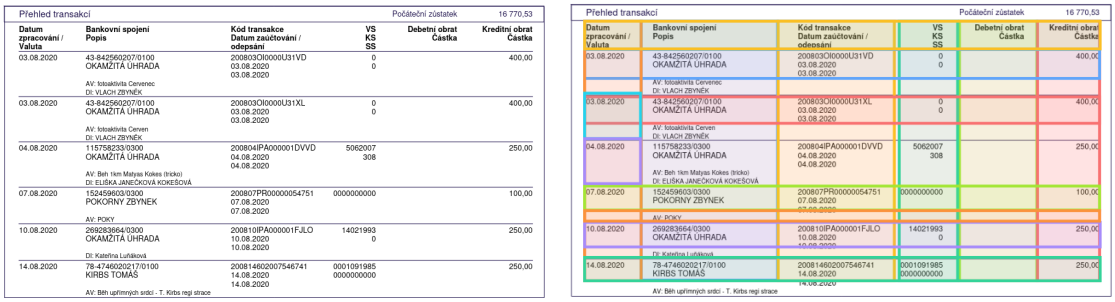
4.2.1 Table Transformer

Table Transformer (TATR) byl vytvořen společností Microsoft Corporation [29] a pro trénování využívá dataset PubTables-1M. Jak napovídá sám název modelu, využívá architekturu transformeru pro kompletní extrakci tabulky. Staví na již existujícím modelu transformeru pro detekci objektů Detection Transformer (DETR) vyvinutým společností Facebook (dnes Meta Platforms) a je schopný kompletně splnit úlohu TE. V rámci úlohy TD jsou nalezené tabulky klasifikovány do tříd `table` a `table rotated` v případech, kdy jsou pro lepší čitelnost v dokumentu otočeny o 90 stupňů. Pro TSR a FA jsou detekovány třídy `table`, `table columne`, `table row`, `table column header`, `table projected row header` a `spanning cell`, přičemž průnik tříd `table row` a `table column` představuje pomyslnou třídu `table grid cell`. První část sítě je již existující architekturou sítě ResNet-18 spolu s šesti vrstvami enkodéru a shodné množství vrstev pro dekodér.

Dle článku [29], v rámci kterého byl model poprvé představen vyplývá, že z počátku sloužil pouze pro detekci a klasifikaci tříd, bez převodu do HTML či CSV. V současné době je již možné výslednou tabulku převést do HTML či CSV.

Pro ověření využitelnosti byl model použit pro rozpoznání struktury tabulky v online prostředí Hugging Face [42]. Na obrázku 4.4a lze vidět vstupní tabulk a na obrázku 4.4b pak výstup modelu. Při bližším prozkoumání je zjevné, že model nedokáže rozpoznat strukturu jednotlivých LI, dokonce ani LI samotné. V případě sloupců se zdá, že odpovídají struktuře.

[9] V rámci práce se používá pojem třída, ponecháno pro přesnější překlad.



(a) Vstupní tabulka transakcí (zdroj: [6]) **(b)** Výstup TATR (původní zdroj: [6])

Obrázek 4.4 Ukázka použití modelu TATR pro rozpoznání struktury tabulku transakcí

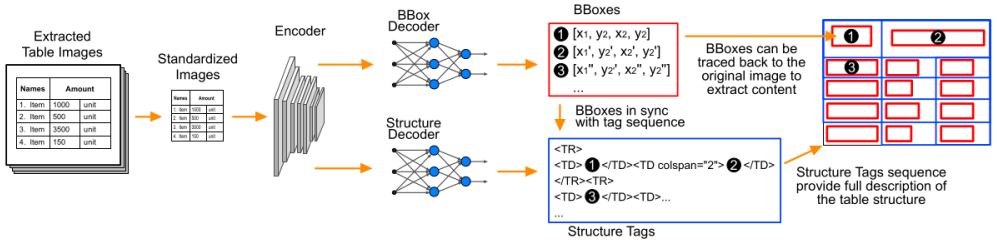
4.2.2 TableFormer

Model vznik v rámci datasetu SyntTabNet, který umožnil generovat velké množství syntetických tabulek. Architektura modelu sestává z konvoluční sítě, která jako vstup má standardizovaný obrázek dané tabulky a výstupem je vektor vytvořených příznaků fixní délky. Následuje dekodér struktury tabulky, který příznaky převede na sekvenci HTML tagů představujících tabulku. Při každém vygenerování datové buňky tabulky tagem `<td>` je skrytý stav předán také do BBox dekodéru, který dané buňce vytvoří BBox. [20]

Enkodér je tvořen již existující sítí ResNet-18, která obvykle slouží pro klasifikaci obrázků do tříd. Z tohoto důvodu byla její poslední lineární vrstva odstraněna a výstupem nejsou detekované třídy ale příznaky ve vnitřní reprezentaci sítě [20].

Dekodér struktury tabulky je implementován jako transformer s vlastním enkodérem výstupních příznaků konvoluční části tvořený dvěma vrstvami a vlastním dekodérem o čtyřech vrstvách, které využívají multi-head attention. Výstupem je sekvnce HTML tagů reprezentující buňky tabulky. Každé vygenerování tagu buňky `<td>` způsobí předání skrytého reprezentace buňky do BBox dekodéru, pro speciální případy kdy buňky jsou horizontálně či vertikálně roztaženy je předán tag `<` následovaný `rowspan=` nebo `colspan=` s číslem označujícím počet standardních buněk, které zpracovávaná nahrazuje. [20]

Dekodér BBoxů produkuje rozměry buňky na základě výstupu z CNN části sítě a skrytého stavu buňky vytvořené v dekodéru struktury. Opět je využito pozornosti transformeru a následně je výstup předán do FNN s třemi vrstvami s aktivační funkcí ReLU a predikuje normalizované souřadnice BBoxu každé buňky tabulky, které jsou následně klasifikovány jako prázdné nebo vyplněné textem.



Obrázek 4.5 Schéma architektury modelu TableFormer

4.2.3 Chargrid

Model [43] je založen na metodách strojového vidění za použití konvolučních vrstev. Vstupy nejsou pouze vykreslené obrázky stránek, ale navíc je vstup obohacen o BBoxy znaků z původního

dokumentu. Pro zakódování těchto informací je využito one-hot encodingu vybraných znaků vyskytujících se ve zpracovávaných dokumentech. Díky tomu není nutné použití modelu pro zpracování textu, protože jednotlivé znaky jsou uloženy přímo v bitmapě vstupu a model by měl být schopný rozpoznat typická uskupení znaků reprezentující třídy k extrakci. Částky, kalendářní data mají obvykle formát, který obsahuje speciální znaky jako tečky, čárky a obsahují čísla, textové informace naopak obvykle čísla neobsahují.

Toto řešení je více univerzální, protože vstupní data lze generovat prakticky z libovolného dokumentu bez znalosti BBoxů buněk tabulky, nicméně je nutné cílové třídy vyznačit.

Znaky jsou zpracované zobrazením z prostoru znaků do prostoru celočíselných indexů $\{a, b, c, d, \dots\} \rightarrow \{1, 2, 3, 4, \dots\}$. Následně jsou BBoxy jednotlivých znaků zapsány do matice v místech jejich výskytu pomocí těchto indexů a aplikován one-hot encoding, který z matice $m \times n$, kde m je výška a n šířka matice vytvoří matici o rozměrech $m \times n \times c$, kde c představuje počet znaků. Tímto je docíleno, že v místě výskytu BBoxu znaku je na příslušné c souřadnici hodnota 1 a v místech bez výskytu hodnota 0. Ukázku stránky zakódované pomocí tohoto postupu lze vidět na obrázku 4.6.

Katti a spol. poukazuje [43], že znak může představovat oblast o velikosti mnoha pixelů, což je odlišné od tradičních úloh zpracování přirozeného jazyka, kde každý znak představuje jeden token. V této reprezentaci jsou větší znaky reprezentovány větším počtem pixelů. Není to považováno za problém, naopak tato vlastnost může pomoci implicitně zakódovat další informace [10], které by jinak nebyly modelu dostupné.

Architekturu sítě se skládá z enkodéru a dvou dekodérů. Při enkódování jsou použity konvoluční vrstvy se s parametrem `stride`, který způsobí snížení šířky a výšky. Naopak počet kanálů se krom poslední vrstvy vždy zdvojnásobuje. Konvoluční vrstvy provádějí takzvanou dilatovanou konvoluci, při které se filtr neaplikuje na přímo sousedící prvky vstupu, ale je mezi nimi mezera, kterou lze vidět na obrázku 4.7.

První dekodér je určen pro segmentaci a jeho výstupem je pravděpodobnostní rozdělení jednotlivých tříd pro daný pixel obrázku. Je využito obrácené konvoluce, která naopak snižuje počet kanálů, ale zvyšuje šířku a výšku. Jako aktivační funkce je použit softmax, který pro jednotlivé pixely vytvoří distribuci pravděpodobností výskytu tříd. Pro určení třídy pak stačí nalézt souřadnici v c, která má tuto hodnotu nejvyšší a zpětně zjistit třídu, která jí náleží. V rámci tříd existuje i speciální třída označující místa bez výskytu.

Druhý dekodér má za cíl nalézt BBoxy jednotlivých LI a opět využívá stejných vrstev pro obnovení původní šířky a výšky. K tomuto účelu je využito takzvané anchor reprezentace BBoxů kde je využito klasifikační a regresní hlavy, která vychází z článku popisující Faster R-CNN [44]. Při tvorbě trénovacího datasetu je zvoleno několik obvyklých tvarů BBoxů, které vyskytujících se v daném datasetu vyskytují. Těmto zvoleným rozměrům se říká anchory a jsou následně použity pro tvorbu trénovacích dat.

Pro každý pixel dokumentu a každý anchor je uvažován pomyslný BBox s rozměry anchoru, který má střed v daném pixelu. Následně je spočítána největší možná hodnota takzvané intersection over union (IoU) dle vzorce 4.1, kde A je BBox se středem ve zkoumaném pixelu s rozměry definovanými konkrétním anchorem a L je LI BBox z dokumentu. Toto se provede pro všechny BBoxy LI v daném dokumentu. Je vybrána nejvyšší hodnota IoU a testováno, zda přesahuje nejnižší hodnotu [11] nutnou pro klasifikaci jako výskyt LI. V opačném případě je dále testována nejvyšší možná hranice pro klasifikaci jako pozadí [12]. V intervalu mezi těmito dvěma hodnotami není přiřazena žádná třída.

$$\text{IoU} = \frac{\text{plocha}(A \cap L)}{\text{plocha}(A \cup L)} \tag{4.1}$$

Cílem regresní úlohy je na jednotlivých pixelech klasifikovaných jako výskyt LI odhad čtveřice

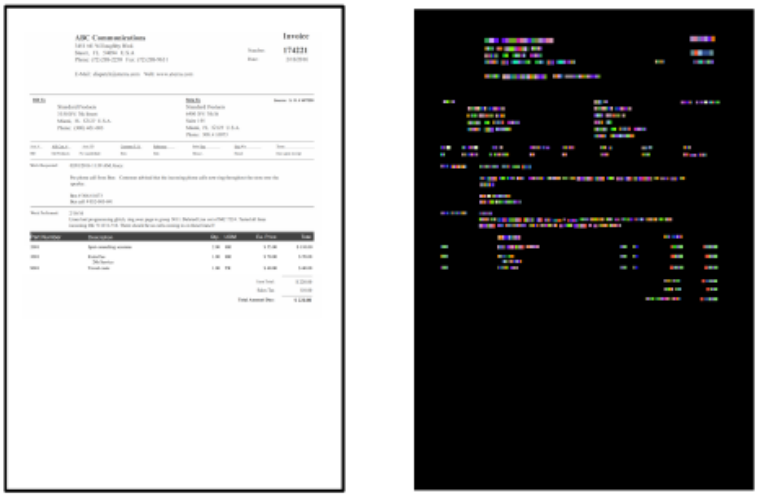
[10] Například velikost fontu.
[11] Obvykle 0,7.
[12] Obvykle hodnoty 0,3.

hodnot x_T, y_T, w_T a h_T. Pro každý BBox A definovaný anchorem a souřadnicí pixelu, a k němu nejlépe ohodnoceným BBoxem L z podkladového dokumentu, je výpočet popsán čtveřicí rovnic 4.2 kde x, y označuje středové souřadnice a w a h šířka a výšku BBoxu.

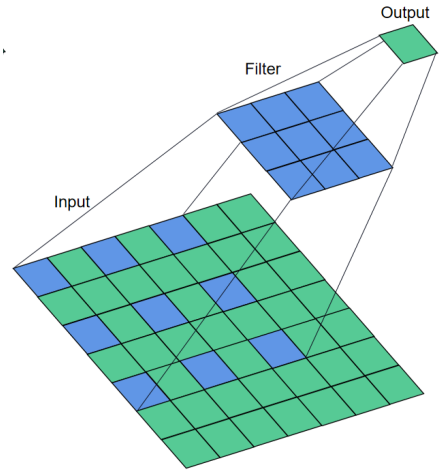
$$\begin{aligned} x_T &= (x_L - x_A)/w_A \\ y_T &= (y_L - y_L)/h_A \\ w_T &= \log(w_L/w_A) \\ h_T &= \log(h_L/h_A) \end{aligned} \tag{4.2}$$

Enkodér a oba dekodéry jsou propojeny pomocí zbytkových spojů. Výstupy vrstev enkodéru jsou přičteny k jednotlivým vstupům vrstev obou dekodérů a pro kombinaci těchto kanálů využívá 1×1 konvoluce, která kombinuje příznaky vyskytující se na stejném místě.

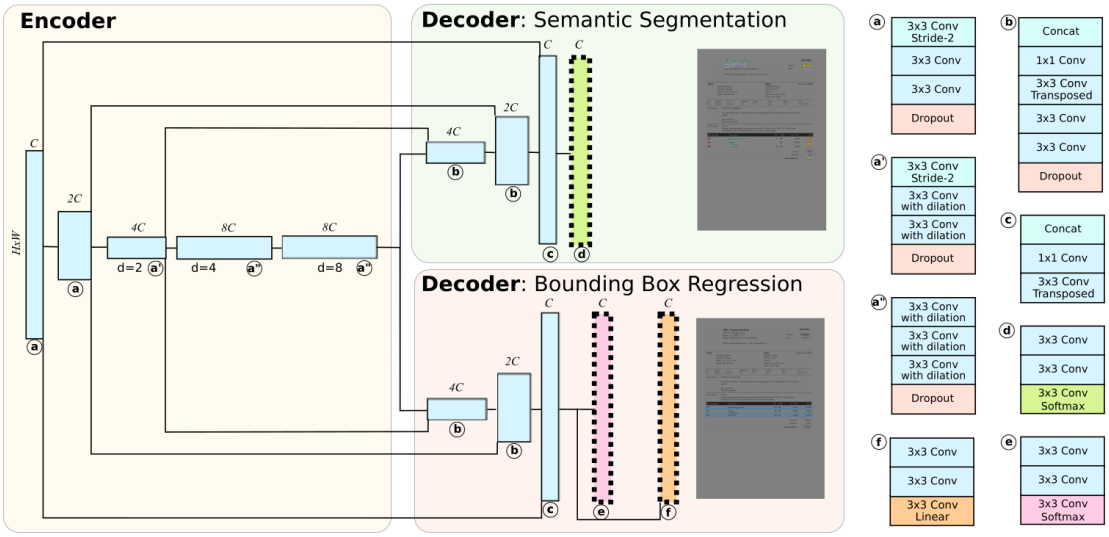
Celá architekturu včetně enkodéru a obou dekodérů je schématicky zakreslena na obrázku 4.8.



Obrázek 4.6 Vlevo se nachází stránka původního dokumentu a vpravo její zakódovaná varianta kde jsou barvami rozlišeny jednotlivé znaky (zdroj: [43])



Obrázek 4.7 Dilatovaná konvoluce, při které není pro výpočet výstupní hodnoty použito přímo sousedících prvků vstupu (zdroj: [45])



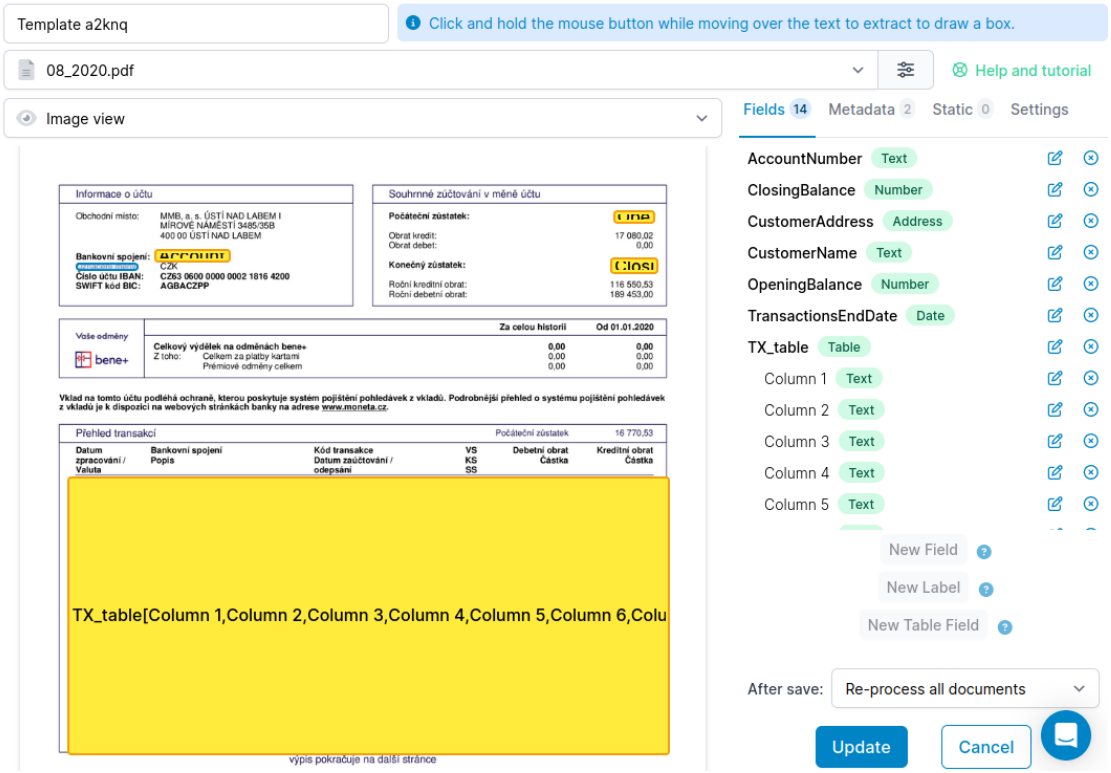
Obrázek 4.8 Architektura sítě Chargrid (zdroj: [43])

4.3 Online služby

Pro průzkum existujících řešení bylo prozkoumáno několik online nástrojů. Jedná se zpravidla o placené služby, které ale umožňují jejich produkt zdarma vyzkoušet. Nástroje, které tuto možnost neumožňovaly nebyly testovány, ale zdá se, že cílí spíše na firmy než jednotlivce.

4.3.1 Parseur

Parseur [46] je online služba pro extrakci informací z nestrukturovaných dokumentů. Hned po prvním pokusu o přečtení testovacího souboru se objevila hláška, že je třeba vytvořit šablonu. Tu bylo nutné vytvořit manuálně na základě pár ukázkových dokumentů. Nástroj nejdříve detekoval 3 slovní spojení ukončené dvojtečkou, které korektně považoval z názvy typu informací za nimi. S informacemi mimo tabulku transakcí nebyl takový problém a detekce většinou fungovala, ale pro tabulku se nepodařilo najít funkční nastavení pro úspěšnou extrakci. Nástroj pravděpodobně počítá s tím, že v rámci jednoho LI není více řádků textu. Také nebyl nalezen způsob, jak v tabulce vyznačovat třídy informací, veškerá detekce probíhala na základě odhadu řádků a manuálně definovaných sloupců. Pro účely výpisů z účtu se nástroj zdá být nepoužitelný, nicméně je nutné zmínit, že vzhledem k obtížnosti tvorby šablony, kdy se dokument nepříjemně posouval při najetí kurzorem na definovanou hodnotu v seznamu nebylo testování věnováno více jak hodina času.



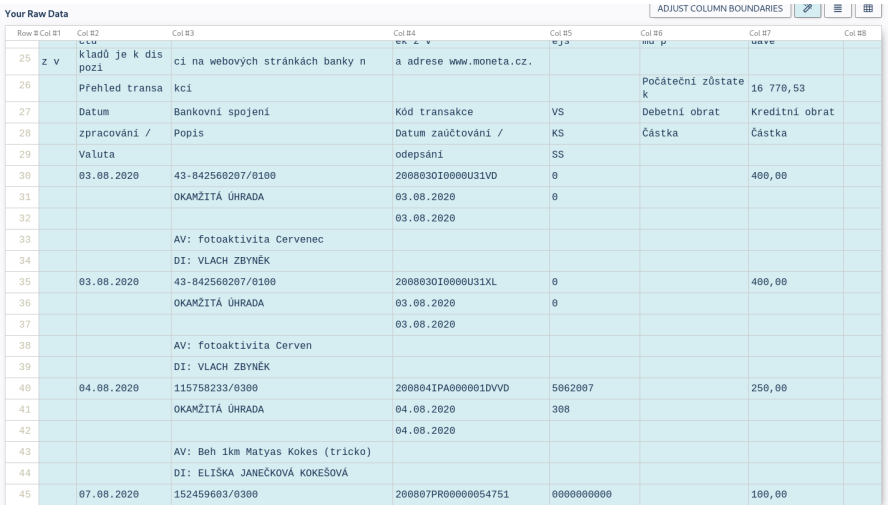
Obrázek 4.9 Ukázka stránky tvorby šablony služby Parseur [46] (zdroj výpisu: [6])

4.3.2 GroupDocs

Tento nástroj [47] místo nahraného PDF zobrazil jen šedé stránky nebo chybovou hlášku. Na stánce Trustpilot [48] má aplikace v současné době 5 hodnocení kde 2 uživatelé hodnotí službu 5 hvězdičkami a 3 uživatelé pouze 1 hvězdičkou se stížností na design aplikace. Jeden z pozitivně hodnotících uživatelů zmiňuje, že službu používá k převodu souboru RTF do formátu PDF a druhá recenze viditelně není od koncového uživatele, ale firmy, která provozuje komerční API na převod dokumentů a má na této stránce právě tuto jedinou recenzi. Z toho lze vyvodit závěr, že služba skutečně nebude plně funkční pro použití s PDF dokumentem.

4.3.3 Docparser

Tato služba [49] podobně jako Parseur využívá užiatelem dodaného vzorku dokumentů. Bohužel také zpracovává tabulku velice podobným způsobem, kdy uživatel vyznačí hranice jednotlivých sloupců. Navíc se nepodařilo omezit tabulku rozměry na svislé ose, což způsobilo, že ve výstupu jsou informace ze začátku výpisu rozdělené dle sloupců tabulky. Ukázku výstupu nástroje lze vidět na obrázku 4.10.



Your Raw Data — ADJUST COLUMN BOUNDARIES

Row #	Col #1	Col #2	Col #3	Col #4	Col #5	Col #6	Col #7	Col #8
25	z v	kladů je k dis pozi	ci na webových stránkách banky n	a adrese www.moneta.cz.				
26		Přehled transa	kcí			Počáteční zůstate k	16 770,53	
27		Datum	Bankovní spojení	Kód transakce	VS	Debetní obrat	Kreditní obrat	
28		zpracování /	Popis	Datum zaúčtování /	KS	Částka	Částka	
29		Valuta		odepsání	SS			
30		03.08.2020	43-842560207/0100	200803OI0000U31VD	0		400,00	
31			OKAMŽITÁ ÚHRADA	03.08.2020	0			
32				03.08.2020				
33			AV: fotoaktivita Cervenec					
34			DI: VLACH ZBYNĚK					
35		03.08.2020	43-842560207/0100	200803OI0000U31XL	0		400,00	
36			OKAMŽITÁ ÚHRADA	03.08.2020	0			
37				03.08.2020				
38			AV: fotoaktivita Cerven					
39			DI: VLACH ZBYNĚK					
40		04.08.2020	115758233/0300	200804IPA000001DVVD	5062007		250,00	
41			OKAMŽITÁ ÚHRADA	04.08.2020	308			
42				04.08.2020				
43			AV: Beh 1km Matyas Kokes (tricko)					
44			DI: ELIŠKA JANEČKOVÁ KOKEŠOVÁ					
45		07.08.2020	152459603/0300	200807PR00000054751	0000000000		100,00	

Obrázek 4.10 Ukázka výstupu služby Docparser [49] s nesprávně zahrnutými informacemi mimo tabulku transakcí (zdroj výpisu: [6])

4.4 Řešení z bankovního prostředí

V pokročilé fázi psaní práce došlo k rozhovoru se zaměstnancem jedné z největších bank v České republice, ve které je podobný problém také řešen. Motivací je identifikace výpisů s odchozími úhradami sázkovým společnostem. Řešení, které je pro tuto činnost využíváno je spojení proprietárního modelu třetí strany, který nejdříve extrahuje tabulku. Dále je aplikována sada pravidel podmíněných detekovanou bankou. Řešení není naprosto dokonalé, ale dokáže zpracovat i naskenované dokumenty. V problematice kalendářních dat[13] se tímto rozhovorem nedosáhlo nových zjištění, protože dotázaný se zabýval čistě implementační stránkou a toto téma spadá píše do účetnictví.

V případech, kdy člověk žádá o půjčku se často stává, že již má účet u jiné banky a dochází k této kontrole výpisu. Během rozhovoru padla, zpočátku neseriózně znějící myšlenka, že banky výpisy z účtu dělají složité a strojově nečitelné cíleně. Toto tvrzení není možné ověřit, ale jisté je, že si rozhodně nebudou dobrovolně sdílet informace, pokud si člověk chce vzít úvěr u jiné společnosti[14]. Ať je skutečnost jakákoli, pravdou je, že tento problém banky intenzivně trápí a dosud na něj vynakládají lidské zdroje[15].

[13] Blíže popsána v sekci 5.7.
[14] To jim nejspíše ani nedovoluje zákon.
[15] V případě této banky nadále dochází k manuálním kontrolám.

Kapitola 5
Tvorba vlastního datasetu

Tato kapitola se zabývá celým procesem tvorby datasetu. Budou zmíněny předpoklady, které byly pro tvorbu použity, použité nástroje a samotný postup tvorby. Nakonec jsou zmíněné problémy a případná řešení.

Protože se nepodařilo nalézt veřejný dataset obsahující bankovní výpisy z účtu a veškeré nalezené datasety byly zaměřené převážně na strukturu tabulky odlišných oblastí [31, 22, 21], byl vytvořen dataset vlastní. Jako podklad pro dataset byla, jak navrhuje Skalický a spol. [5], použita veřejně dostupná data z internetu.

Možnost zhotovení syntetických dat by byla za určitých okolností také možná, ale přináší několik požadavků. Nejdříve je nutné porozumět dané doméně a relevantním podkladovým datům. Porozumění znamená nejen schopnost správně interpretovat informace a znát termíny domény, ale především znát různé neobvyklé anomálie jednotlivých výpisů, rozdíly mezi bankami i speciální případy v rámci jednoho typu výpisu. Dataset SynthTabNet použitý pro TableFormer sice texty generuje náhodně podle četnosti [20], ale také pak dále nepracuje s třídami informací uvnitř jednotlivých buněk. Z důvodu náročnějšího zpracování skutečných dat pro pochopení obecných pravidel, bylo od ambiciózní možnosti vytvoření syntetického datasetu upuštěno.

5.1 Předpoklady

V průběhu práce se pracovalo s několika předpoklady. Kvůli značné různorodosti, a to i v rámci jedné banky, byla většina předpokladů vyvrácena opravdu rychle. V naprosto ideálním případě je nejlepší se v této oblasti nespoléhat vůbec na nic. Pro ilustraci tohoto tvrzení, případně výpomoc budoucímu čtenáři, který by řešil podobný problém, jsou níže uvedeny předpoklady, které jistě nejsou pravdivé:

- číslo účtu není nikdy zalomeno[1],
- všechny banky mají jednotlivé LI vizuálně odděleny čarou nebo pozadím,
- LI není nikdy rozdělen mezi stránkami[2],
- LI nebude obsahovat duplicitní informace,
- pro člověka bude tabulka jasně interpretována[3],

[1] Většinou opravdu není, tím spíše nebezpečné.
[2] Týká se PDF generovaných z HTML.
[3] Asi nejvíce naivní předpoklad.

- hodnoty stejné třídy se budou v rámci výpisu vyskytovat na stejném místě, bez ohledu na banku protistrany.

Pro kompletnost jsou níže uvedeny předpoklady, které nebyly dosud vyvráceny[4]. Předpoklady jsou založeny na nepraktičnosti a nejasnosti výpisu v případě jejich porušení.

- tabulka operací nikdy horizontálně nesousedí s jinou tabulkou operací, či klíčovými informacemi,
- LI operace vždy obsahuje alespoň jedno datum.

5.2 Sběr dat

Zdrojem výpisů byly stránky neziskových organizací, spolků, sbírek a obcí. Prvotní nalezení webových stránek obsahující výpisy bylo realizováno přes vyhledávač Google, za použití dotazu `filetype:pdf výpis z účtu`. Dále byly stránky ručně prohledány pro nalezení ostatních výpisů za jiná období. V případě většího množství výpisů byl pro rychlejší extrakci použit regulární výraz na zdrojovém kódu stránky se seznamem výpisů.

Z počátku byl chybně zvolen postup ručního pojmenovávání souborů dle banky, například `csob_001.pdf` a zápis URL spolu s názvem do CSV souboru. Tento postup se ukázal jako neudržitelný, protože bylo nutné kontrolovat, zda již neexistuje záznam pro každou stahovanou URL.

Jako mnohem efektivnější se ukázal postup, kdy veškeré nalezené PDF soubory výpisů byly uloženy s originálním názvem a metadata zaznamenána do XLXS souboru. Metadaty se v tomto případě rozumí:

- URL, ze které byl výpis stažen (pokud možno přímý odkaz),
- název souboru, tak jak byl uložen na disk (viz níže),
- název banky výpisu,
- poznámka.

XLSX tabulka usnadnila práci s udržováním metadat, obzvláště při stahování většího množství výpisů z jedné stránky a změnách struktury při vývoji. Kontrola s tabulkou se prováděla pouze při kolizi s existujícím souborem, který se pak uložil (spolu s připojenou unikátní příponou) jen za podmínky, že byl z nového zdroje. Do pole poznámky bylo zaznamenáno cokoli zajímavého nebo neobvyklého k danému výpisu. Jako příklad lze uvést, že některá PDF obsahovala několik výpisů sloučených do jednoho souboru nebo různé výpisy od stejné banky prošly v průběhu let aktualizacemi vzhledu. V případě České spořitelny pak došlo ke změně služby internetového bankovnictví Servis24 na službu George.

Některé výpisy byly pro účely práce nepoužitelné, protože obsahovaly nekvalitní skeny. Tyto výpisy byly přesto zaznamenány, aby se případně tyto zdroje znovu neprozkoumávaly. Soubory těchto výpisů byly taktéž ponechány, aby se při potenciálním pokusu o opakované stažení okamžitě zjistilo, že webová stránka s výpisy byla již prozkoumána. Držení nepoužitých dat lze v tomto případě tolerovat, protože takových souborů nebylo mnoho a značně to ulehčilo práci. Nebylo tak třeba manuálně vyhledávat soubory v rámci tabulky, pouze přidávat nové záznamy, pokud soubor na disku již neexistoval.

[4] K jejich vyvrácení ale pravděpodobně dříve či později dojde.

5.3 Nástroj Label Studio

Pro anotaci byl použit nástroj Label Studio[5]. Jedná se o open source nástroj s webovým rozhraním. Při tvorbě datasetů této práce byl nástroj používán lokálně jedním uživatelem, ale je možné ho použít i v rámci teamu. Výhodou Label Studia je možnost užití zdarma. Dále má možnost spustit vlastní lokální instanci a lze poměrně slušně konfigurovat některé aspekty, obzvláště pak přímo anotační obrazovku. Anotace je obecně monotónní činnost, při které se rychle projeví drobná nepříjemná zpomalení, či frustrace ze zbytečných repetitivních kroků. Label Studio umožňuje konfiguraci klávesových zkratek a upravení rozhraní dle specifických potřeb. Je dobré zmínit, že tento nástroj neslouží pouze k anotaci jednotlivých obrázků, ale i textu či videa.

Veškeré použitelné výpisy byly zpracovány do vzorků pro tento anotační nástroj. Label Studio umožňuje více metod importu vzorků. Přímočará možnost je vygenerovat obrázky jednotlivých stránek do složky, ze které se pak jednotlivé obrázky importují jako vzorky. Optimální se ukázala možnost, kdy pro každou stránku použitého dokumentu existuje ve zvolené složce JSON soubor představující jeden vzorek datasetu. Obrázek se v uživatelském rozhraní načítá z cesty, specifikované v tomto souboru. Tato možnost má několik výhod.

První výhodou je možnost přidání uživatelem definovaných metadat. Ta posléze lze použít v rámci uživatelského rozhraní Label Studia pro filtrování či vyhledávání jednotlivých vzorků. V této práci byla metadata použita pro přidání názvu zdrojového dokumentu, čísla stránky, názvu banky a dalších informací dle konkrétního datasetu (viz dále).

Další nespornou výhodou je možnost do souboru vzorku přidat „předanotace“. Ty slouží např. jako pomocné vodítko v případech, kdy je známá poloha objektu ale neznámá třída. Uživatel poté „pouze“ přiřazuje třídy k předpřipraveným BBoxům, případně je upravuje či maže. Do předanotace lze také vložit přímo informaci o predikované třídě (např. z dosavadního modelu). Tvary oblastí výskytu tříd jsou dodávány v procentech, takže při přegenerování obrázků do jiného rozlišení, např. pro lepší čitelnosti fontů, zůstanou na svém místě.

5.4 Dataset pro Line Item Recognition

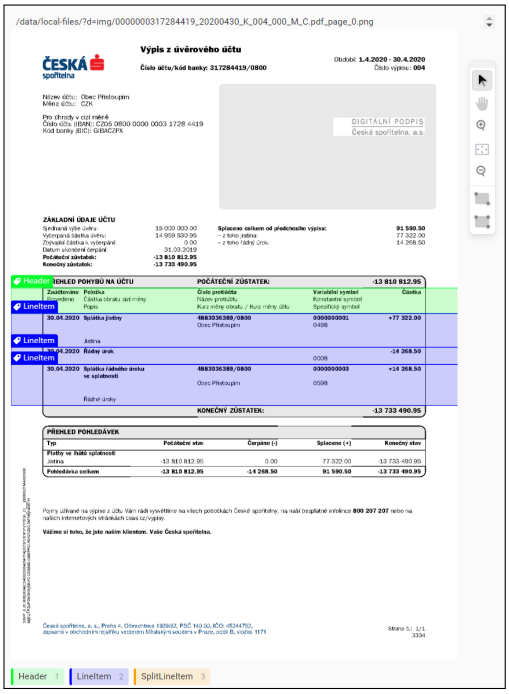
Line Item Recognition dataset je primárně zaměřený na tabulku operací. Z implementačních důvodů vysvětlených dále vytvořený LIR dataset přesně neodpovídá definici 4.8, protože přímo nezahrnuje informace o třídách hodnot uvnitř jednotlivých LI. Obsahuje ovšem informace o polohách samotných LI a také polohu hlavičky (pokud existuje). Z těchto informací lze poté odvodit, které texty jsou uvnitř tabulky a které jsou mimo ni. V rámci LIR datasetu se rozlišují 3 třídy:

- `Header`,
- `LineItem`,
- `SplitLineItem`.

Třída `SplitLineItem` měla označovat LI, který byl rozdělen na konci řádku a pokračoval na další. Později se ukázalo, že ve výpisech, které byly generovány přímo bankou, se tato třída nevyskytuje. Výskyty této třídy existovaly pouze v případech, kdy byl výpis vytvořený z webové stránky v prohlížeči virtuálním tiskem do PDF. Není vyloučeno, že se tato třída vyskytuje ve výpisech banky, která není zahrnuta v datasetu této práce, speciálně pak zahraniční.

Na obrázku 5.1 lze vidět ukázku hotové anotace LIR v Label Studiu. Ve spodní části je výběr zmíněných tříd, vpravo pak výběr nástrojů pro práci s anotací. Na první pohled by se mohlo zdát zvláštní, že anotace přesahují šířku tabulky. Jedná se o drobné zjednodušení práce při vyznačování obdélníků. Během tvorby datasetu se pracovalo s předpokladem 5.1, že nenastane situace, kdy

[5] `https://labelstud.io/`



Obrázek 5.1 Ukázka obrazovky anotace datasetu LIR v Label Studiu s načtenou první stranou výpisu od České Spořitelny (zdroj: [50])

tabulka sousedí s klíčovou informací či jinou tabulkou. Tím pádem horizontální prostor patří buďto tabulce nebo je prázdný. Toto téma bude dále znovu vysvětleno v širším kontextu.

5.5 Dataset pro Key Information Localization and Extraction

KILE dataset obsahuje třídy informací jednotlivých textů v rámci celého výpisu, tedy jak tabulky operací, tak obecných informací, což v praxi znamená mnohem větší počet tříd. V tabulce 5.1 je spolu s významy vypsaný seznam tříd, které se nacházejí mimo tabulku operací. Převážně je jejich výskyt limitován na první stranu výpisu. V tabulce 5.2 je pak naopak seznam tříd, které se objevují uvnitř tabulky operací. Předpona `Tx`, naznačující slovo „transakce“, není naprosto přesná. Ve výpisech se nacházejí i řádky, kde se prakticky o žádnou transakci nejedná.

Na obrázku 5.2 lze vidět ukázku hotové anotace KILE v Label Studiu. Obrázek dobře ilustruje, jak může být anotace obtížná. I přes velkou volnost Label Studio nedovoluje konfigurovat veškeré aspekty zobrazení. Konkrétně zde vyvstal problém s velikostí, případně umístěním, popisků bounding boxů. Ty totiž bohužel často překrývají ostatní texty.

5.6 Předanotace heuristickým řešením

Tvorba KILE datasetu obsahující takové množství tříd se ukázala velmi obtížná, jak z otázky času, tak z pohledu udržení konzistence a přesnosti. Z tohoto důvodu byl zvolen postup pomocné předanotace heuristickým řešením.

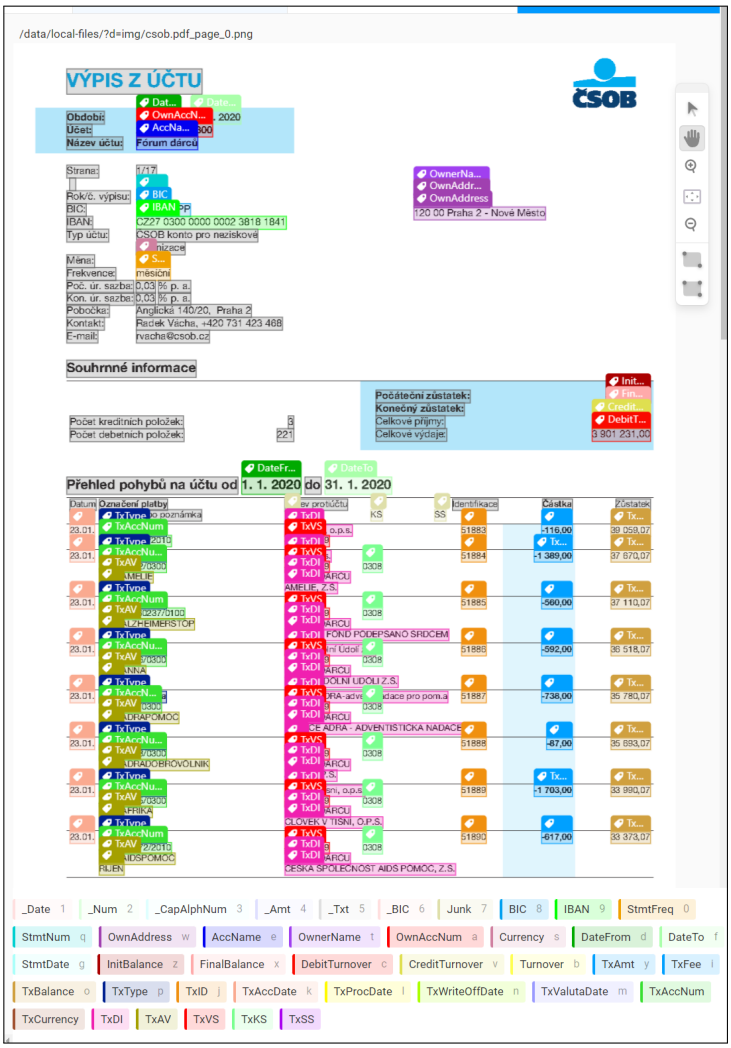
Heuristické řešení funguje ve dvou fázích, mezi kterými jsou výstupy první fáze manuálně zkontrolovány, případně upraveny. Tím se zajistí úspora práce a nepropagují se chyby v takové míře.

název třídy	význam
BIC	BIC banky výpisu
IBAN	IBAN účtu výpisu
StmtFreq	frekvence výpisu (měsíční, roční, …)
StmtNum	číslo výpisu
AccName	název účtu
OwnAddress	adresa majitele účtu
OwnerName	název majitele účtu
OwnAccNum	číslo účtu daného výpisu
Currency	název měny
DateFrom	datum počátku období výpisu
DateTo	datum konce období výpisu
StmtDate	datum vytvoření výpisu
InitBalance	počáteční zůstatek
FinalBalance	konečný zůstatek
DebitTurnover	debetní obrat
CreditTurnover	kreditní obrat

Tabulka 5.1 Třídy mimo tabulku operací, obvykle vyskytující se na první straně

název třídy	význam
TxAmt	částka operace, zahrnuje sloupce „kreditní/debetní obrat“
TxFee	poplatek
TxBalance	zástatek po vykonání operace
TxType	typ operace
TxID	identifikace transakce (z pohledu banky)
TxAccDate	datum zaúčtování
TxProcDate	datum zpracování
TxWriteOffDate	datum odepsání
TxValutaDate	datum valuty
TxAccNum	číslo protiúčtu
TxCurrency	měna operace
TxDI	debetní identifikace
TxAV	zpráva pro příjemce
TxVS	variabilní symbol
TxKS	konstantní symbol
TxSS	specifický symbol

Tabulka 5.2 Třídy uvnitř tabulky operací



Obrázek 5.2 Ukázka obrazovky datasetu KILE v Label Studiu (zdroj: [51])

V jazyce Python byla vytvořena kódová reprezentace některých objektů dokumentu. Jedná se o reprezentaci stránky dokumentu, která dále obsahuje textové objekty a případně LI.

Textové objekty obsahují, krom samotného textového řetězce i BBox, který byl získán z výstupu knihovny, potenciální třídu a informaci, zda text vznikl sloučením z dvou jiných nebo je původní. Pro BBoxy byly implementovány metody vykonávající rutinní operace. Samotné textové objekty obsahují znakové objekty, opět s vlastními BBoxy.

Tato reprezentace vedla na značné usnadnění práce s dokumentem a jeho objekty. V případě textových objektů bylo možné například oddělit podřetězec a rozdělit tak text na dva [6] nebo tři menší [7]. Opačným směrem bylo možné texty slučovat do větších celků, na základě vzdáleností a mezer. Při celém procesu bylo nutné hlídat správnou práci se znakovými objekty a BBoxy uvnitř textových objektů.

5.6.1 Heuristické řešení LIR

Předanotace se pro LIR dataset vytvoří následujícím postupem:

1. dokument se vykreslí do bitmapy ve stupních šedi,
2. na bitmapě se provede detekce hran,
3. pro každý řádek pixelů je na základě detekce rozhodnuto, zda funguje jako oddělovač,
4. rozpoznají se LI splňující kontrolní podmínky.

Detekce hran je provedena za použití funkce `adaptiveThreshold`[52] z knihovny OpenCV[8]. „*V případě, kdy obrázek obsahuje různě osvětlené oblasti, může pomoci adaptnivní mez. Algoritmus určí mezní hodnotu pro každý pixel na základě malé oblasti v okolí. Tímto je získána rozdílná mezní hodnota pro různé oblasti ve stejném obrázku, což dává lepší výsledky pro obrázky s různorodým osvětlením.*“ [53].

Thresholding se obvykle používá pro rozdělení hodnot na dvě skupiny podle určité meze. Jako příklad lze uvést úpravu čitelnosti skenů dokumentů, kdy se pro stránku ve stupních šedi zvolí hranice a pro každý pixel se na základě této hranice rozhodne, zda má být bílý nebo černý [38]. Díky algoritmu výpočtu adaptivní meze a volbou malého okolí lze detekovat také hrany. Použité volání detekce hran využívá čtvercovou oblast 3×3 (nejmenší možnou) a hraniční hodnota je vypočítána jako průměr této oblasti. Prostřední pixel oblasti je tedy považován za hranu, pouze pokud je v rámci této oblasti nadprůměrně tmavý. Pokud tato oblast má celá stejnou barvu, rovná se průměru a tím pádem hranou není.

Pro každý řádek pixelů se poté spočítá, v jakém poměru je tvořen hranou. Ignorují se takové řádky, kde je toto procento pod 50 %. Mezi dvojice po sobě jdoucích řádků, které nebyly ignorovány, se vytvoří potenciální LI. První podmínkou pro ponechání LI je zahrnutí textového řetězce, který vyhovuje regulárnímu výrazu pro datum. Druhou je, že výška nepřesáhne 10 % dokumentu. To se ukázalo jako velmi užitečné např. v případně, kdy pod tabulkou byl půlstránkový text obsahující datum.

Úplně posledním krokem je detekce hlavičky, při které se v rámci každého LI spočítá počet textů, obsahující klíčová slova, která byla vypozorována z hlaviček několika výpisů. Zároveň součet nesmí být příliš nízký, protože v datasetu existují výpisy bez hlavičky.

Je dobré zmínit, že LI v mnohých výpisech nejsou odděleny rovnou čarou. Toto řešení umožňuje detekovat např. LI vyznačené šedými obdélníky (viz 5.3d). Nicméně v případně úplné absence vizuálního oddělení pak pochopitelně tato strategie selhává.

Na obrázku 5.3a lze vidět vizualizaci hodnot detekce hran. Dále na obrázku 5.3b vidíme relativní porovnání hodnot detekce hran pro jednotlivé řádky a poté jsou tyto řádky na obrázku

[6] Pokud podřetězec obsahoval první nebo poslední znak.
[7] Pokud podřetězec neobsahoval krajní znaky.
[8] `https://opencv.org/`

5.3c již očištěny od šumu. Oddělovače, které neohraničovaly LI jsou zde ale stále, protože se zatím neprovedla kontrola přítomnosti kalendářního data. Na posledním obrázku 5.3d jsou již rozpoznané LI.

Nakonec se každý vzorek zkontroluje a případně upraví manuálně v Label Studiu. Finální výstup se vygeneruje pouze tehdy, pokud je uživatelem explicitně řečeno, že předanotace nebo její úprava byla zkontrolována. Ve výstupní složce se poté nachází JSON anotace, obsahující BBoxy a metadata jednotlivých stránek dokumentů.

5.6.2 Heuristické řešení KILE

Postup pro předanotace pro KILE je mnohem složitější než v případě LIR. Cílem je jednotlivým textům (nebo jejich částem) přiřadit třídu určující typ informace.

▶ **Definice 5.1** (metatřída)**.** *Metatřída je uměle vytvořená třída, která slouží pouze pro implementaci heuristického algoritmu, kde indikuje, že text může náležet pouze do určité podmnožiny tříd.*

Meta třídy (definice: 5.1) jsou v rámci algoritmu přiřazeny textům, za účelem postupné redukce počtu možností validních tříd pro jednotlivé texty. Možnými metatřídami jsou:

- `_BIC` - text s podezřelý jako kód BIC ,
- `_Date` - datum libovolného významu,
- `_Amt` - částka,
- `_Num` - číslo,
- `_CapAlphNum` - jakýkoli text obsahující pouze velká písmena a čísla.

Třída `_BIC` je zahrnuta, protože specifikaci BIC splňují často i jiné texty. K finálnímu přiřazení dochází tedy až později. Lze si povšimnout, že veškeré meta třídy začínají podtržítkem, což je užitečné při vizuální kontrole nebo algoritmizaci.

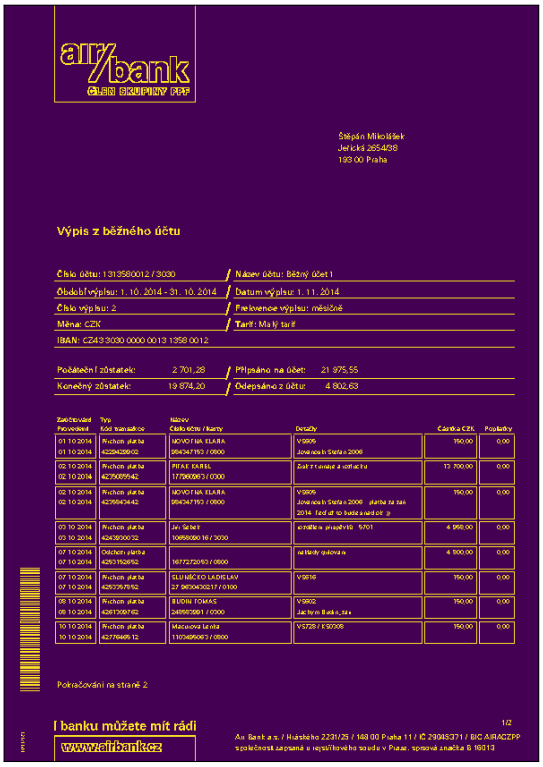
Algoritmus zvažuje více důvodů, proč daný text klasifikovat jako určitou třídu. Tyto důvody jsou v implementaci uloženy jako jednotlivé konfigurační objekty. Každý konfigurační objekt určitým způsobem definuje pravidla, která text (nebo jeho část) musí splňovat, aby mu byla přiřazena třída. Těchto pravidel může existovat pro jednu třídu více, ale nikdy nejsou testována v konjunkci. Každé funguje víceméně nezávisle na ostatních a může mít samo o sobě za následek označení textu danou třídou. Výjimkou jsou pravidla, která pro své splnění vyžadují, aby text již byl označen určitou metatřídou. Priorita je obecně udávána pouze pořadím, ve kterém se tato pravidla testují. V rámci ověřování se pravidlo vždy vyhodnotí na všech textech, kde dosud není přiřazená finální třída[9].

Některá pravidla mohou vyžadovat, aby byl text součástí tabulky operací. S informacemi z LIR datasetu (vytvořených dříve) lze dokument rozdělit na tabulku operací a oblast mimo tabulku.

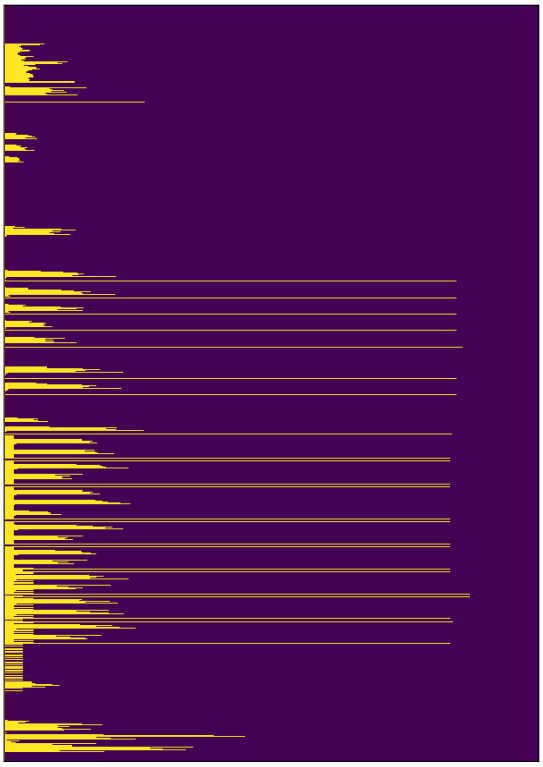
Algoritmus nerozlišuje mezi výpisy různých bank a tím pádem ani nevolí speciální pravidla. Tato možnost je ale díky flexibilní konfiguraci otevřena pro případné budoucí úpravy. Jako reálný se jeví scénář, kdy algoritmus na výpis určité banky nefunguje tak dobře, jak by mohl fungovat, pokud by pravidla nebyla sdílená. V takovém případě by se po detekci výpisu dané banky použila odlišná sada pravidel. Z důvodu poměrně malého množství dat se občasné chyby vyřešily ruční korekcí anotace při kontrole. Při větším množství dat by ale tento postup nemusel být realizovatelný.

Pro větší přehlednost a jednodušší implementaci byla pravidla rozdělena na několik typů. Každý typ má odlišné parametry a tím pádem se pravidla různých typů liší postupy, kterými jsou vyhodnocovány. Typy pravidel jsou následující:

[9] Tedy libovolné třídy kromě metatříd.

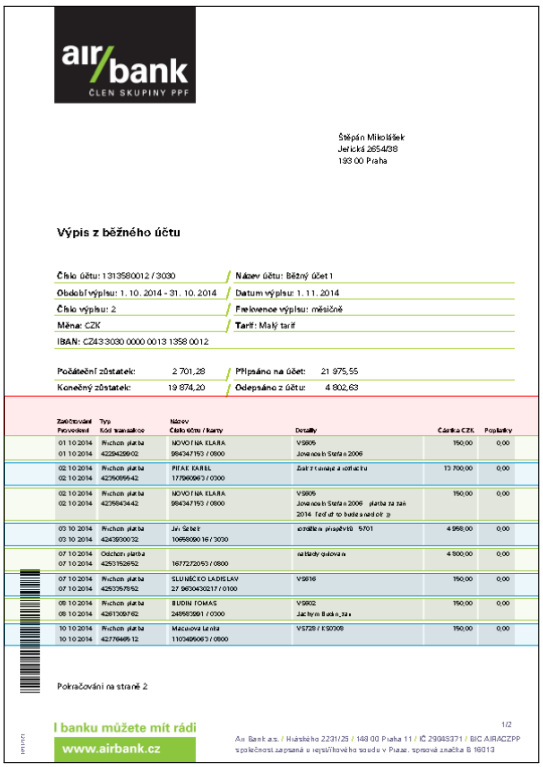


(a) LIR Heuristika: detekce hran (původní zdroj: [54])



(b) LIR Heuristika: porovnání hodnot řádků

(c) LIR Heuristika: očištěné hodnoty řádků



(d) LIR Heuristika: rozpoznání LI (původní zdroj: [54])

1. regulární výraz - u každého textu se pouze ověří, zda vyhovuje regulárnímu výrazu. Tento typ pravidla slouží spíše k detekci metatříd nebo vybraných tříd, které jsou mimo tabulku informací a má následující konfigurovatelné atributy:
   - třída - přiřazovaná třída,
   - regulární výraz - výraz kterému musí text (ne vždy nutně celý) vyhovovat,
   - počáteční kotva [10] - zda první písmeno textu musí být součástí regulárního výrazu,
   - koncová kotva [11]- zda poslední písmeno textu musí být součástí regulárního výrazu,
   - pozice na stránce - v tabulce operací, mimo tabulku operací nebo na pozici nezáleží,
2. kontext v okolí - text musí být označen metatřídou a v okolí se vyskytuje text, který splňuje regulární výraz. Tento typ pravidla je použit zejména v případech, kdy se nalevo od zkoumaného textu vyskytuje název hodnoty (např. „Datum výpisu:“ před „1. 1. 2023“), Atributy jsou:
   - třída - přiřazovaná třída,
   - vyžadovaná třída - třída, kterou musí text mít pro otestování pravidla,
   - jeden z regulárních výrazů:
     - pro text nalevo od zkoumaného textu,
     - pro text nad zkoumaným textem,
   - pozice na stránce - v tabulce operací, mimo tabulku operací nebo na pozici nezáleží,
3. kontext LI - vyhodnocuje se v rámci jednotlivých LI. Pozice na stránce je tedy implicitně vždy uvnitř tabulky operací, atributy jsou:
   - třída - přiřazovaná třída,
   - vyžadovaná třída - třída, kterou musí text mít pro otestování pravidla,
   - regulární výraz pro nejbližší text z hlavičky,
   - pouze jedna instance - zda se může vyskytovat vícekrát v rámci LI. Slouží k vybrání pouze jedné nejlepší možnosti,
   - pravděpodobně alespoň jedna instance - zda se třída pravděpodobně vyskytuje v LI (např. částka a datum). Slouží ke zvýšení priority.

Samotný postup pro vytvoření anotací je následující:

1. texty se rozdělí na dvě skupiny, texty z tabulky operací a texty mimo ní,
2. sousední texty se spojí, pokud navazují čísly nebo vybranými speciálními znaky,
3. textům se pokusí přiřadit vybrané třídy[12], na základě pravidel typu 1,
4. téměř všechny sousední texty se sloučí[13],
5. za pomoci dalších pravidel typu 1 vyžadujících spojení většiny sousedních textů se provedou detekce tříd,
6. aplikují se pravidla typu 2,

[10] Některé konfigurační objekty sdílejí konstantu regulárního výrazu, ale liší se v přístupu k začátku a konci textu. V rámci přehlednosti implementace pak byla preferována možnost definice konstanty na jednom místě. Z tohoto důvodu byly obě kotvy parametrizovány.
[11] Viz 10.
[12] Včetně metatříd.
[13] Krom textů, které navazují některým z vybraných znaků.

7. aplikují se pravidla typu 3.

V kroku 2 je cílem spojit jednotlivé texty náležící jednomu zápisu data, částky nebo čísla účtu. V zápisu data a částek se totiž dle typografických pravidel vyskytují mezery. Každý jednotlivý text získaný z knihovny Poppler je vždy bez mezer. Například při výskytu data „20. 2. 2013“ by byly výstupem tři texty, konkrétně „20.“, „2.“ a „2023“.

Dále se v kroku 3 přiřazují takové třídy, které jsou s velice vysokou pravděpodobností správně určené. Jedná se například o třídu `Currency`, `IBAN` a `OwnAccNumber`. Také se zde vyskytují již zmíněné metatřídy, které pouze zvyšují znalost informace v textu.

Následuje spojení veškeré sousedních textů (krok 4, kromě takových, kde se neshoduje již přiřazená třída [14]. Výjimku představuje nespojení dvojice textů, kde první z nich končí dvojtečkou. Ta totiž obvykle znamená, že po ní následuje hodnota, která nebude obsahovat její samotný název před dvojtečkou. Proběhne detekce tříd, které vyžadují spojení textů navzájem navazujících znaků. Mezi tyto třídy patří například `_CapAlphNum`.

Do této chvíle veškeré přiřazování tříd pouze testovalo texty na regulární výraz. V dalších krocích se používají dodatečné informace v okolí textu či hlavičky v tabulce operací. V kroku 6 se pro každé pravidlo kontroluje regulární výraz buďto u textu nalevo od zkoumaného nebo u textu nad ním. Tato fáze se týká převážně informací mimo tabulku operací. Jako příklad, kde třída záleží na kontextu v okolí v rámci LI lze opět zmínit obrázek 1.2, na kterém je vidět příklad názvu hodnoty pole zakončeného dvojtečkou, následovaný samotnou hodnotou pole.

Posledním krokem je přiřazení tříd v rámci LI. Pro každé pravidlo se detekují relevantní texty hlavičky. Například pro třídu `TxAccDate` (datum zaúčtování) se tyto texty hledají pomocí regulárního výrazu `Datum zaúčtování|Datum účtování|Zaúčtováno|Zaúčtování`. Pro každý text (kompatibilní metatřídy) v rámci LI se spočítá vzdálenost vůči jednotlivým relevantním textům. Ta se počítá na základě dosaženého procenta maximálního překryvu s relevantním textem v hlavičce (vzorec 5.1) a vzájemného rozdílu odhadnutých čísel řádků v rámci LI (vzorec 5.2). Výsledná vzdálenost je pak spočítána kombinací těchto dvou hodnot ve vzorci 5.3.

$$\text{překryv} = \frac{\text{šířka}_{A \cap B}}{\min(\text{šířka}_A, \text{šířka}_B)} \tag{5.1}$$

$$d_{\text{řádků}} = \text{abs}(\text{řádek}_A - \text{řádek}_B) \tag{5.2}$$

$$d = (1 - \text{překryv}) + 0.2 * d_{\text{řádků}} \tag{5.3}$$

Jednotlivé výsledky obsahují samotný zkoumaný text z LI, potenciální třídu a již zmíněnou vzdálenost. Výsledky jsou seřazeny od nejmenší vzdálenosti, tedy největší pravděpodobnosti správného odhadu. Třídy se postupně v tomto pořadí přiřadí. Jakmile má text přiřazenou třídu, je dále ignorován. V případě, že třída může být dle konfigurace přiřazena pouze jednou, jsou její další pokusy o přiřazení rovněž ignorovány.

Nakonec se zkoumají dosud nepoužité konfigurace tříd, u kterých se očekává existence alespoň jedné instance v rámci LI. Vzdálenost v rozhodování již nehraje roli. Tyto třídy se přiřadí libovolnému textu, který má přiřazenou požadovanou metatřídu. Priorita je udána pořadím v konfiguraci detekce tříd.

5.7 Třídy KILE datasetu

Z důvodu vysoké rozmanitosti výpisů, převážně pak užívání odlišné terminologie, bylo občas obtížné určit některé třídy.

Během tvorby datasetu vzniklo podezření, že někteří uživatelé přidávají variabilní (případně také konstantní) symbol ručně do pole poznámky, i když se symbol zároveň nachází v určeném poli. Na druhou stranu konzistence ve formátování (včetně „VS:“) a vysoká četnost tohoto jevu

[14] Včetně meta tříd.

Tabulka 5.3 Přehled názvů polí s kalendářními daty u různých bank

název pole data	Moneta	KB	ČSOB	AirBank	Fio	Raiff. SK	Raiff. CZ	ČSAS
zaúčtování	×	×		×	×			×
odepsání	×							
zpracování	×							
valuta	×		×				×	
datum			×				×	
transakce		×			×			
provedení				×				×
bez hlavičky						×		

naznačují, že se může jednat o chybu na straně banky. Obecně v rámci manuální kontroly několikanásobně vyvstala otázka, zda hodnotu vyhovující určitému poli, brát jako hodnotu pole i v případě, kdy byla napsána ručně uživatelem do poznámky.

Jedním z nejobtížnějších problémů byla samotná identifikace a počet tříd obsahující kalendářní data v tabulce operací. Bohužel se data jmenují různě napříč bankami a přesné významy se dohledávají jen velmi obtížně. Tabulka 5.3 ukazuje, jaká pole s datem se vyskytovala v jednotlivých výpisech. Bylo vytvořeno mapování podle významové podobnosti slov. V případech, kdy je ve výpisu napsáno pouze `datum` nebo chybí hlavička, se s největší pravděpodobností jedná o pole `datum zaúčtování`. U ostatních rozhodování nebylo tak jednoznačné a s vysokou pravděpodobností vneslo do datasetu šum. Při zpětné kontrole dokonce byly zjištěny časté nekonzistence v manuálním postupu, které musely být dodatečně opraveny. Příčinou bylo pořadí zpracovaných dokumentů, kdy se až po delší době objevil dokument, který vyvrátil dosud platící domněnku o některých třídách ve výpisu dané banky.

5.7.1 Zjednodušení KILE datasetu

Kvůli výskytu tříd, kde i pro člověka bylo obtížné rozeznat rozdíly, byl počet tříd snížen.

Ze všech typů kalendářních dat v LI se stalo jedno `TxDate`. Texty, které nebylo možné klasifikovat se označily jako `TxOther`. Do této třídy se také dostávají `TxAV` a `TxDI`, protože se příliš často vyskytovalo jméno uživatele uvnitř zprávy pro příjemce. Obecně se jednalo o texty tvořící věty nebo nějaký význam.

Podobným způsobem se třída `OwnAddress` sloučila do `OwnerName`. Tyto dvě třídy prakticky nelze rozeznat jinak než manuálně.

5.8 Vztahy mezi jednotlivými datasety

Jak již bylo zmíněno, předanotace pro KILE vyžaduje jako vstup informace o poloze jednotlivých LI z LIR datasetu. Samotné heuristické řešení LIR není dokonalé, a proto byl každý vzorek manuálně zkontrolován, případně upraven. Úpravy, mimo změny třídy, také často obnášely úpravu rozměrů BBoxu. To bohužel vedlo k tomu, že BBox pak mohl přesahovat dělící čáru (strojově nalezenou detekcí) a podobné nekonzistentní výstupy. Proto byl zvolen následující postup standardizace všech anotací.

Pro každý BBox LI se nalezenou veškeré texty, které svými středy náleží do tohoto LI. Ten se poté upraví tak, aby jeho vrchní a spodní okraje byly shodné s nejvyšším, respektive nejnižším okrajem náležících textů. Vzniklý dataset se opět ručně zkontroluje, aby se omylem nestalo, že manuálně upravený obdélník nezahrnul nějaký text LI. Případné opravy se provádí na prvku původního LIR datasetu následované přegenerováním standardizované verze.

Za pomocí vzniklého datsetu se, dle již uvedeného postupu, vytvoří KILE dataset, který je po manuální kontrole převeden na zjednodušený a následně je opět ručně zkontrolován. Zde je vhodné podotknout, že tato kontrola je prakticky vynucená Label Studiem, protože dataset je již zkontrolován v předchozím kroku, kdy obsahuje více tříd. Nicméně tímto způsobem bylo odhaleno pár chyb, které omylem prošly první kontrolou, kdy byl dokument méně přehledný. Celé schéma tohoto postupu je vyznačeno na obrázku 5.4. Dataset nelze ani v poslední fázi považovat za zcela finální, jelikož nikdy neexistuje záruka naprosté správnosti. Nasvědčuje tomu i fakt, že některé chyby byly nalezeny velice pozdě a pouze náhodou.

Každý vzorek z předanotace nebo transformace musel být v Label Studiu vždy ověřen, aby se dostal do výstupní složky. V případě nalezení chyby v datasetu, který byl algoritmicky vytvořen úpravou z jiného[15], byla v prostředí Label Studia zvolena možnost odmítnutí předanotace. Zdroje odmítnutých vzorků byly poté ručně opraveny.

Pro umožnění automatizace některých úloh byla vytvořena databáze, díky které se daly snáze hlídat návaznosti jednotlivých vzorků. Tato databáze také umožnila nalezení dokumentu nebo zdrojového vzorku [16] a vyřešila tím problém nejasného pojmenování vzorků ve výstupní složce Label Studia, kdy k němu z názvu nebylo možné přiřadit původní předanotaci, případně dokument samotný. Databázové schéma pro správu vzorků a zdrojových dokumentů lze vidět na obrázku 5.5. Tento přístup nesmírně ušetřil práci v případech, kdy se chyba v anotaci bez povšimnutí propagovala dále. V ideálním případě by se dalo velkému množství chyb vyhnout při kontrole více lidmi, ale to bohužel při psaní této práce nebylo možné.

Pro správu datasetů byly vytvořeny, kromě již zmíněné databáze, také Python scripty a interaktivní Jupyter notebooky. Scripty sloužily pro dávkové operace, jako je smazání odmítnutých anotací z výstupní složky či přegenerování jejich souborů. Také napomáhaly s udržováním databáze samotné, kontrolovaly existenci záznamu pro každý uložený dokument, či přidávaly do databáze nové dokumenty z již zmíněného XLSX souboru s metadaty. V neposlední řadě také dávkově označovaly vybrané dokumenty jako ignorované na základě textového souboru, do kterého byly během kontroly přidávány například skeny. Jupyter notebooky oproti tomu sloužily spíše k vizuální kontrole jednotlivých stránek, spouštění specifických dotazů databáze při vyhledávání dokumentů konkrétních bank nebo kontrole návaznosti a ladění kódu.

5.9 Problémy

Tato část se zabývá problémy, které vyvstaly během tvorby datasetu.

5.9.1 Podkladová data

Celý postup tvorby datasetů byl velice náročný, jak z časového hlediska, tak z hlediska dodržování konzistentních postupů. Na vině je hned několik následujících faktorů.

Předně nebylo jasné, jaké třídy informací se obecně vyskytují ve zdrojových dokumentech. Třídy byly několikrát zaktualizovány a staré anotace zahozeny, aby se docílilo větší konzistence a stejných možností výběru tříd v rámci všech vzorků.

Druhým důvodem byly nejasné případy, kdy ani člověk nedokázal jednoznačně určit o jakou třídu se jedná. V některých případech z počátku docházelo k chybnému označení, dokud nebyl znám případ, který dokázal, že se jedná spíše o jinou třídu. Typickým příkladem tohoto jevu jsou již zmíněné uživatelské poznámky, kam uživatelé pravděpodobně psali některé klíčové informace ručně.

Různorodost výpisů je mezi bankami opravdu značná a samotný fakt, že u některé banky jsou ve výpisu čtyři pole kalendářních dat a u některých jsou pouze dvě (nebo dokonce jedno)

[15] Standardizovaný LIR a zjednodušený KILE.
[16] V případě, že vzorek je vytvořený za pomocí jiného.

je zarážející. Obecně zde platí, že se není téměř na co spolehnout, protože ani systém pro generování výpisů jednotlivých bank nepočítal se všemi případy. Jako příklad lze uvést zalomení dvou posledních znaků čísla účtu, kdy se na řádek jednoduše nevešlo. Dále formát zápisu různých poplatků, či případná podrobnější rozepsání jejich struktury, jsou naprosto odlišná mezi jednotlivými bankami.

5.9.2 Správa vzorků a datasetu

Vytvoření podpůrného kódu pro správu jednotlivých datasetů bylo tím pádem více problematické, než se předpokládalo. Většina problémů pramenila z omezených zkušeností s tvorbou datasetů a nově nalezených problémů v datech. Do databázového schématu bylo tak občas třeba dodatečně doplnit sloupec či relaci. Podobný problém nastal i u metadat anotací jednotlivých stránek, protože dokud se nezačaly kontrolovat, nebyl důvod k jednotlivým vzorkům dohledávat další informace, případně zdrojový soubor. Během práce se při ruční anotaci KILE datasetu ukázalo, že je velice užitečné mít název banky uložený v rámci metadat jednotlivých vzorků, pro porovnávání a dodržování konzistentního přístupu u nových anotací. Bohužel některé vzorky byly v té době již zpracované, a tím pádem nešlo tuto informaci přidat do všech[17].

5.9.3 Omezení anotačního nástroje

Nástroj Label Studio byl většinu času nesmírně užitečný, ale i tak má své nevýhody. Přestože umožňuje poměrně vysokou míru úprav, v době psaní práce nebylo možné měnit styl, a hlavně velikost popisků u jednotlivých instancí tříd v dokumentu. To vedlo k nízké přehlednosti, kdy se některé instance skrývaly za popisky. Jednoduché řešení, kdy se dokument přiblížil, také vedlo k dalším problémům. Předně stisknutí prostředního tlačítka myši, pro inicializaci posunu stránky, se zrušil veškerý dosavadní výběr. Program sice nabízí možnost použít pro posun klávesovou zkratku, kde se toto neděje, ale měnit zažité návyky je velice obtížné. Jako řešení tohoto problému se ukázalo do jisté míry úsměvné otočení monitoru na výšku, díky čemuž bylo možné vidět celý dokument najednou.

V případě vzorků s velkým počtu BBoxů byla znatelně zhoršená odezva a bylo třeba dokument opravdu značně přiblížit, aby se snížil počet vykreslovaných obdélníků. To občas způsobilo chybu, projevující se vizuálním odtržením vrstvy s anotacemi od vrstvy podkladového obrázku. Při prvním setkání s touto chybou nebylo jasné, zda se vrstva skutečně neposouvá. Naštěstí se ukázalo, že chyba je čistě zobrazovacího charakteru a k žádné skutečné změně souřadnic nedochází.

Při postupném selektivním výběru BBoxů se v Label Studiu zobrazí překryvný obdélník reprezentující výběr a jeho rozměry pokrývají všechny dosud vytvořené BBoxy. V situaci, kdy uživatel nejdříve vybere například BBox vlevo nahoře a následně druhý vpravo dole, tento obdélník pokryje téměř celý obrázek a pokus o výběr BBoxu uprostřed stránky celý výběr následně zruší. Jelikož BBoxy obsahovaly vždy jedno slovo, bylo při hromadném přiřazení třeba dopředu plánovat pořadí výběru. V některých situacích bylo jistější přiřazovat po menších skupinách nebo v rámci jednoho LI, aby se v případě chyby nedošlo k zbytečné ztrátě dosavadního výběru.

Některé problémy se podařilo vyřešit úpravou konfigurace zobrazení, přesněji úpravou CSS, kdy se podařilo panel s nabídkou tříd přesunout, aby nepřekážel a nezabíral místo pro pracovní plochu s obrázkem. Podobným způsobem se podařilo opravit nezvyklé chování kolečka myši při změně přiblížení, které přibližovalo a oddalovalo přesně opačným směrem než u všech ostatních programů, včetně prohlížeče samotného. Docházelo pak k nečekanému chování, při kterém se dokument při pokusu o oddálení sice zpočátku zmenšoval, ale v momentě, kdy se pod kurzorem uvolnilo místo a nacházel se nad nově odkrytým pozadím stránky, se celá stránka naopak začala

[17] Bez smazání nebo manuálních úprav již vytvořených výstupních souborů.

přibližovat (tentokrát v režii prohlížeče). Některé chyby byly v rámci práce reportovány v GitHub repozitáři Label Studia[18], některé již byly nahlášeny jinými uživateli, ale dosud nevyřešeny [19].

Může se zdát, že tyto chyby jsou ve skutečnosti banalita a dá se alespoň na některé z nich zvyknout, ale bohužel při střídavé práci s více různými programy si prakticky není možné dávat pozor na nestandardní chování jednoho z nich. Také větší množství monotónní práce si za velmi krátkou dobu začne vybírat svou daň. Drtivá většina chyb byla způsobena překrytím menšího částí textu názvem třídy z popisku již anotovaného BBoxu.

5.9.4 Formát a obsah datasetu

Obsahem obou datasetů jsou vyznačené BBoxy, které v případě LIR označují instance LI, v případě KILE pak třídy informací ze stránky výpisu. Dataset obsahující strukturu tabulky, které využívají jiné datasety [22, 28] nebylo možné vytvořit. Tyto datasety byly totiž vygenerovány strojově ze strukturovaných podkladových dat, které pro výpisy nejsou dostupné. Manuální cestou by nebylo možné vytvořit konzistentní anotace buněk, které jsou často prázdné nebo z dostupného textu nelze určit okraj, dokonce v některých případech nelze určit ani počet buněk. Vzhledem k tomu, že cílem práce je převod do strukturovaného formátu, je nutné extrahovat informace s ohledem na jejich třídu. K tomu by formát tabulky nijak nepomohl, protože ze zjištěné struktury buněk by tato informace stejně nešla odvodit. Některé hodnoty se nachází volně v tabulce, bez jejich indikace v hlavičce, nebo je název před nebo nad hodnotou.

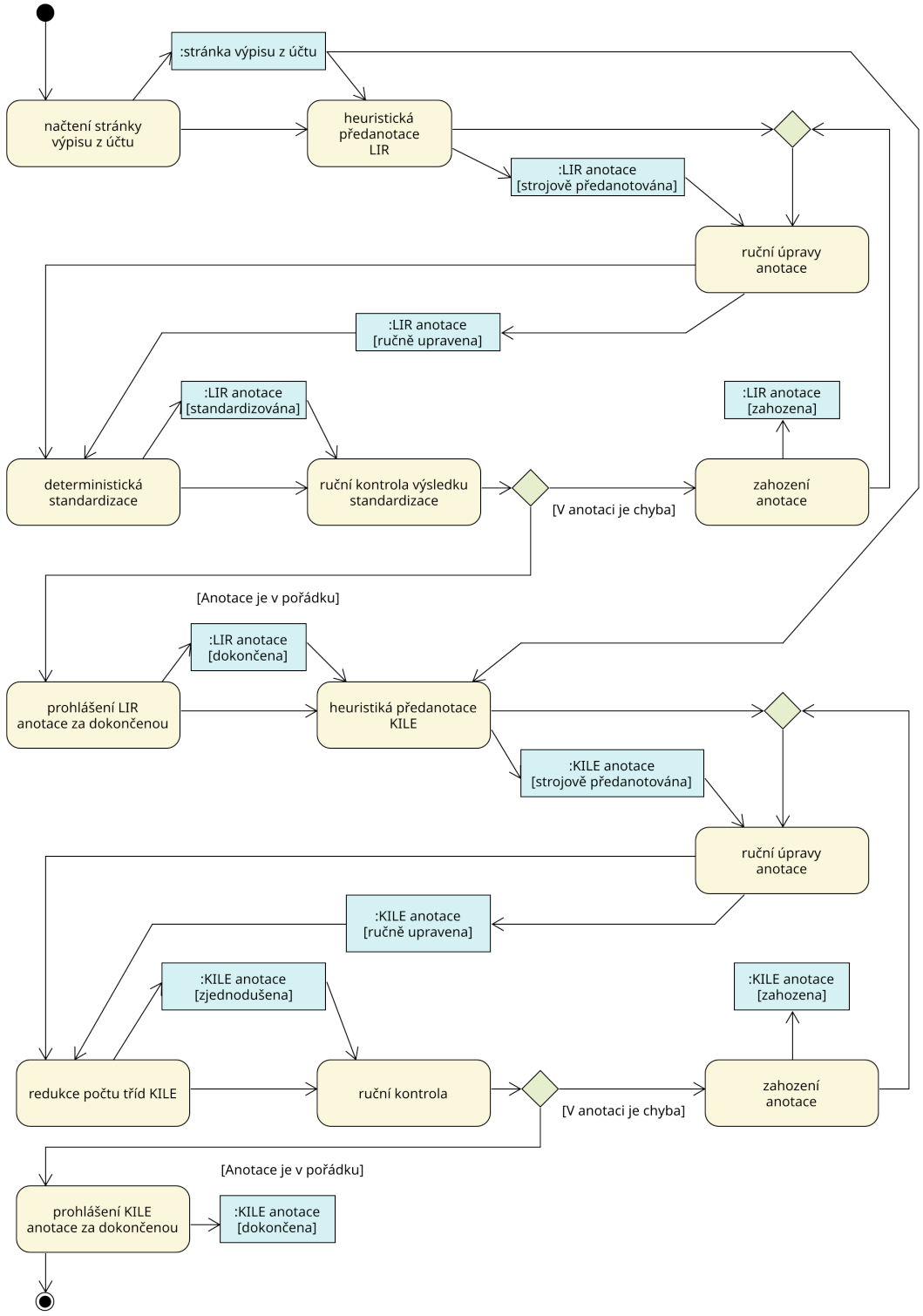
5.10 Shrnutí procesu tvorby datasetů

V rámci práce byly vytvořeny dva datasety se stejnými podkladovými daty. První z nich se zaměřuje úlohu LIR, konkrétně pouze na lokalizaci LI. Druhý je zaměřen čistě na KILE, ale ve spojení s prvním společně plně pokryjí cíle LIR a KILE. Oba datasety byly vytvořeny poloautomatizovanou cestou, kdy byla data nejdříve předanotována heuristickým řešením a následně ručně zkontrolována, případně opravena.

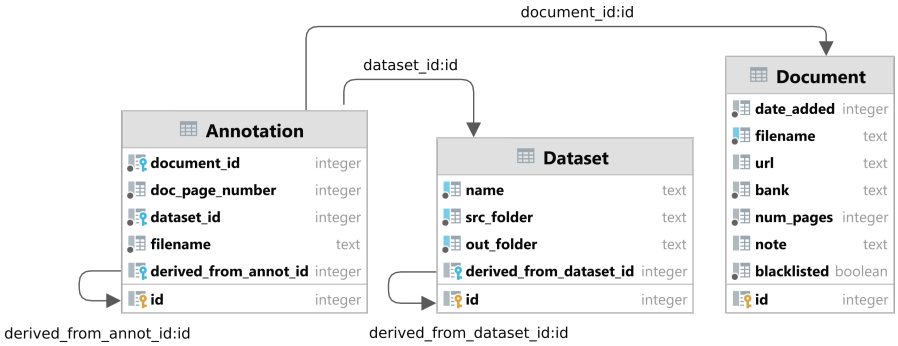
Z důvodu problematiky rozklíčování struktury buněk a také nesnadné reprezentace této struktury v anotačním nástroji, dataset tuto strukturu neobsahuje, což omezuje množinu použitelných modelů.

Dataset obsahuje 878 anotací stránek dokumentů výpisů z účtů, které byly získány z veřejných zdrojů, které jsou zaznamenány v databázi.

[18] https://github.com/heartexlabs/label-studio/issues/3839
[19] https://github.com/heartexlabs/label-studio/issues/2947



Obrázek 5.4 Schéma postupu tvorby datasetu



Obrázek 5.5 Schéma databáze pro správu datasetů, anotací a zdrojových dokumentů

Kapitola 6

Implementace převodu do strukturovaného formátu

Kapitola se zabývá tvorbou finální aplikace pro převod dokumentu do strukturovaného formátu za využití vytvořeného modelu strojového učení, jehož architektura byla zvolena na základě kapitoly analýzy existujících řešení 4.2, vedle toho je využito heuristické řešení, které je spojení dvou algoritmů použitých v rámci předanotace datasetu a následně jsou obě řešení porovnána. Výsledkem jsou dva Python skripty pro příkazovou řádku, umožňující převod dokumentu do formátu JSON.

6.1 Tvorba model strojového učení

Z prozkoumaných modelů v sekci 4.2 byl vyhodnocen jako vhodný pouze model Chargrid 4.2.3, protože jako jediný umožňuje plně řešit úlohy LIR a KILE, na rozdíl od modelů TableFormer 4.2.2 a Table Transformer 4.2.1. V případě použití obou těchto modelů by bylo nutné další zpracování buněk tabulky, kterou se nepodařilo v rámci tvorby vlastního datasetu přesně určit, protože nebylo možné rozpoznat, které buňky jsou nevyplněné, případně získat rozměry buněk pokrývající větší prostor, než jsou rozměry vloženého textu.

Model Chargrid byl implementován na základě článku Chargrid: Towards Understanding 2D Documents [43], kde je popsána jeho struktura 4.8. Implementace byla uskutečněna v jazyce Python za použití knihovny PyTorch [55], která obsahuje předem předpřipravené komponenty pro tvorbu ANN a implementuje jejich chování i pro spuštění na grafické kartě, která značně urychluje výpočty.

Velkou část implementace představoval průzkum různých vedlejších zdrojů, především článku věnující se modelu Faster R-CNN [44], protože některé informace nebyly jednoznačně interpretovatelné, naštěstí téměř všechny nejasnosti byly vyřešeny prozkoumáním citovaných materiálů. Schéma architektury sítě Chargrid 4.8 se ukázalo pro kompletní pochopení nedostatečné, protože neobsahuje úplný seznam prvků jednotlivých bloků, přesněji vrstvy dávkové normalizace a aktivační funkce ReLU pro konvoluční bloky. V textu jsou bloky podrobněji popsány a síť bylo možné implementovat přímočaře. Kritickým místem se ukázaly transponované konvoluce, které v druhé části sítě zvětšují rozměry matic příznaků faktorem 2. Jelikož jsou veškeré bloky v síti citlivé na rozměry vstupních a výstupních dat, při chybně zvolených parametrech těchto bloků docházelo k vypisování chyb, kvůli nekompatibilním rozměrům výstupů vrstev se vstupy dalších. Konkrétně při průchodu příznaků touto vrstvou byla na výstupu produkován tensor[1] téměř správných rozměrů, ale šířka a výška byly přesně o jeden prvek menší, než bylo požadováno. Po

[1] Tensor je označení pro zobecněnou matici libovolných rozměru.

bližším prozkoumání oficiální dokumentace této vrstvy [55] vyšlo najevo, že parametry, jako jsou `padding` a `stride` jsou sice pojmenováním shodné s parametry vrstev pro klasickou konvoluci, ale výstup ovlivňují jiným způsobem. Dle vzorce z dokumentace byl pečlivě prozkoumán vliv jednotlivých parametrů na rozměry výstupu, a nakonec identifikován parametr `output_padding`, který tento problém vyřešil.

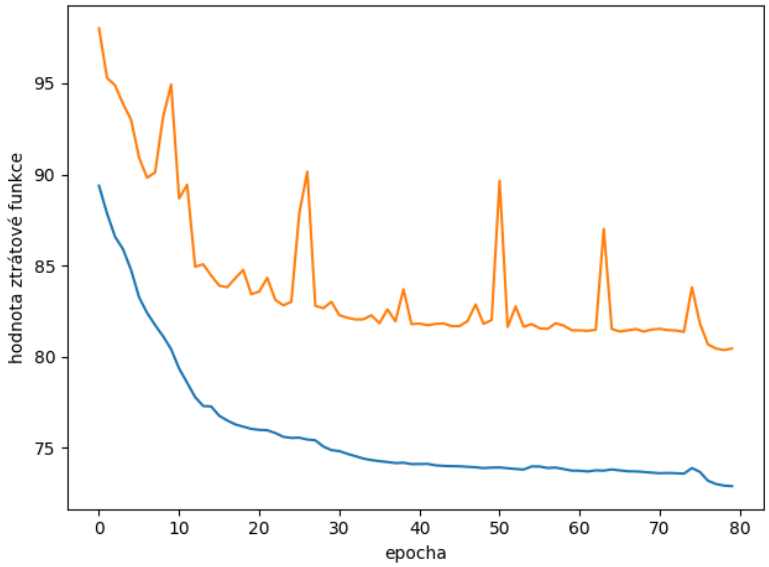
Pro přípravu trénovacích dat bylo vytvořeno zobrazení jednotlivých znaků či skupin znaků, na číselné indexy. Sloučení znaků bylo použito za účelem snížení velikosti vstupních dat modelu, protože je využito one-hot kódování, které se zvyšujícím se počtem tříd úměrně zvyšuje rozměry vstupních dat modelu. Použité zobrazení pro znaky lze vidět v příloze A.1. Háčky a čárky byly ze vstupních textů odstraněny a v rámci zobrazení byla vytvořena univerzální třída `ClassMapping`, která zobrazuje libovolné znaky na číselné indexy. Pro nalezení cílového indexu interně využívá slovník, který při zobrazení neznámého vstupu nahrazuje vrácením zmíněné univerzální třídy, zdrojový kód si lze prohlédnout v příloze A.3. Dále byla vytvořena zobrazení pro třídy úlohy KILE a LIR, které lze obě rovněž nalézt v příloze A.2 a A.3.

Následovala implementace PyTorch třídy datasetu [56], která se používá jako základ dynamického vytváření vzorků modelu. Zpočátku zvolený postup, kdy při každém požadavku na vzorek byla v databázi nalezen záznam o jeho anotacích a následný převod do správné reprezentace vstupu sítě se ukázal jako velice neefektivní, protože se každým průchodem datasetem vytvářely ty samé vzorky jako v předchozí. Bohužel z důvodu velikosti dat nebylo možné využívat ukládání dat do operační paměti mezi iteracemi. Také bylo vhodnější síť trénovat na rychlé grafické kartě v cloudové službě Google Colaboratory [57], která uživatelům zpřístupňuje možnost využití speciálního hardwaru pro trénování neuronových sítí, který je pro tyto účely mnohonásobně rychlejší než běžný počítače. Z tohoto důvodu bylo vyvinuta třída, která jako vstup dostane objekt původní implementace PyTorch datasetu, zanalyzuje rozměry produkovaných tensorů a metadat, které dataset pro každý vzorek produkuje a následně je postupně uloží na disk pro další použití. Vstupní data pro neuronovou síť jsou uložena jako takzvaný memmmap [2], který umožní, že celý tensor nemusí být načtený v operační paměti, ale systém plynule načítá data z disku podle potřeby. Práce s daty probíhá obvyklým způsobem, jako by byla celá uložena v operační paměti. Díky tomu bylo také méně komplikované celý dataset nahrát do již zmíněné služby bez podpůrného kódu pro databázi, zpracování anotací a zdrojových PDF souborů. Pro čtení předem generovaného datasetu byla opět vytvořena třída implementující PyTorch dataset, která je schopna na základě uložených metadat z výše uvedeného postupu věrně napodobit chování a výstupy původního implementace ovšem tentokrát bez dynamického vytváření vzorků z podkladových dat. Pro implementaci uložení dat na disk a následného načtení ze souboru memmap bylo dbáno na udržení nezávislosti na zbytku práce a také co největší všeobecnosti implementace, což umožňuje další využití v jiných projektech, protože vstupní dataset pro tento převod je omezen pouze existencí implementovaných metod rodičovské třídy datasetu PyTorch [56].

Výstup sítě pro segmentaci na KILE třídy využívá ztrátové funkce křížové entropie pro klasifikaci do více tříd. Pro detekci BBoxu jednotlivých LI je využito binární křížové entropie a pro regresi velikosti BBoxu se jedná o Huber loss. V případě KILE bylo nutné zvolit váhy pro ztrátovou funkci, aby se zamezilo snaze modelu vytvořit snadné řešení a predikovat jen pozadí dokumentu.

Výstupy segmentace KILE tříd lze přímočaře interpretovat nalezením indexu kanálu třídy s největší hodnotou výstupu pro danou souřadnici. Oproti tomu výstupy dekodéru BBoxů, který je používán pro detekci jednotlivých LI přímo interpretovat nelze a je třeba využít inverze výpočtu již popsaného postupu v 4.2.3. Jelikož tento proces vytvoří mnoho kandidátů oblastí výskytu objektu, je třeba nejdříve snížit jejich počet volbou hraniční hodnoty pro připuštění do dalšího kroku. Tato hodnota byla po určité době experimentování určena jako 0,54. Dále je třeba vybrat takové oblasti, které jsou s největší pravděpodobností opravdu místa výskytu objektu a odfiltrovat ty, které s dobře hodnocenými mají významný překryv s IoU přesahující mezní hodnotu. Jako hraniční byla v tomto případě zvolena hodnota 10 % a aplikován algoritmus Non-maximum

[2] Memmory-mapped file.



Obrázek 6.1 Graf změny hodnoty ztrátové funkce pro jednotlivé epochy trénování.

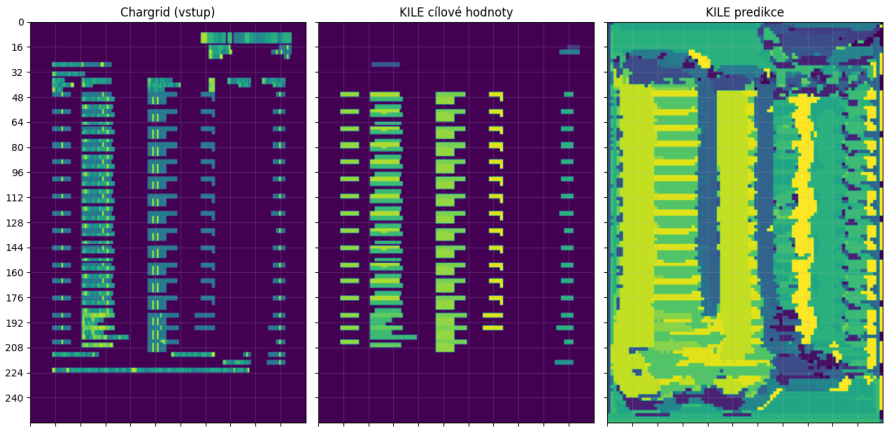
suppression [58], který zmíněný postup provádí. Protože jednotlivé LI vždy představují celou horizontální oblast, pro jejich reprezentaci bylo použito pouze výšky a pozice na vertikální ose, díky čemuž je výsledný model o něco menší.

V článku [43] je popsáno trénování sítě pomocí optimizéru SGD [59], ale při pokusech o trénování sítě tímto způsobem bylo velice obtížné nalézt hyperparametry pro viditelnou konvergenci, a proto byl zvolen optimizér AdamW [60], který by měl umožnit rychlejší konvergenci [61] a zdál se být daleko méně citlivý na změny hyperparametrů jako je rychlost učení. Model byl trénován 80 epoch na množině 658 vzorků s validační množinou 132 a testovací množinou velikosti 88 vzorků, tedy 85, 15 a 10 % z celkových 878 vzorků datasetu a rychlostí učení 0,005. Na obrázku 6.1 lze vidět graf hodnot ztrátové funkce na trénovací a validační množině v závislosti na iteraci průchodu datasetem. Z grafu jasně vyplývá, že model nepřekvapivě dosahuje na validační množině horších výsledků, protože při vyhodnocení na datech, která nebyla přímo použita pro trénování je nutné generalizace. Hodnoty ztrátové funkce v obou případech klesají, ale na validačních datech dochází k občasným výkyvům chyby. Průměrná hodnota ztrátové funkce na testovací množině byla 72,09 což je méně než pro validační množinu v poslední iteraci.

6.2 Vyhodnocení modelu strojového učení

Ukázalo se, že model strojového učení není zatím dostatečně přesný. Výstup modelu pro jeden vzorek z testovací množiny lze vidět na obrázku 6.2, dále na obrázku 6.3 lze vidět matici záměn tříd na úrovni znaků, aby se předešlo sporným vyhodnocením případů původně souvislých řetězců výstupu knihovny Poppler, které byly při anotaci rozděleny, například odstraněním částí názvů hodnot jako „AV:“. Shrnutí všech výsledků představuje tabulka 6.1 kde N je počet prvků výskytu třídy v anotaci a následně jsou uvedeny hodnoty přesnosti daného řešení.

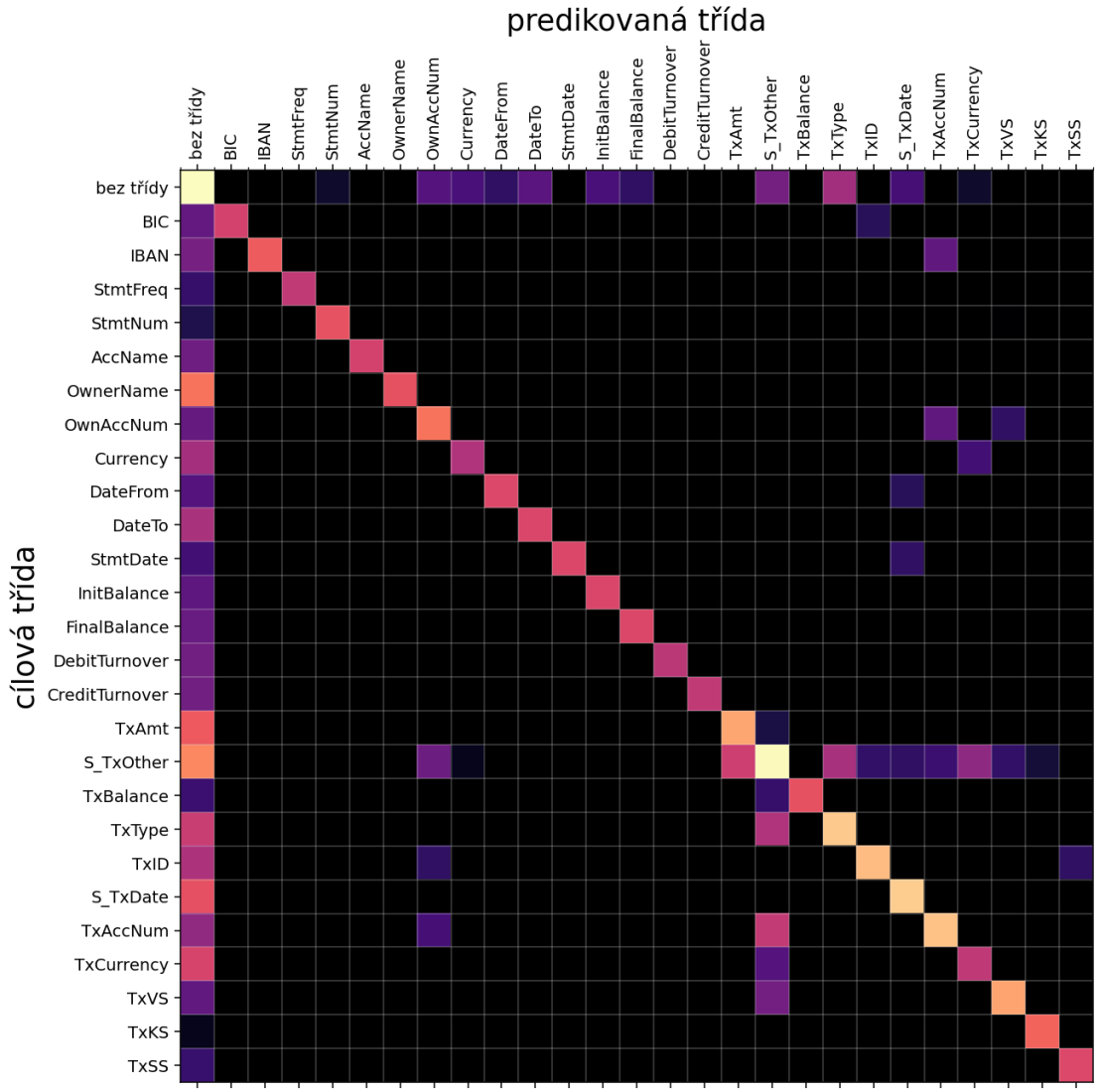
V případě rozpoznávání transakcí má model průměrnou chybu 8,68 na stránku. V některých případech také predikuje LI vzdálené od skutečné tabulky transakcí, jak lze vidět na obrázku výstupu modelu pro stránku výpisu 6.4.



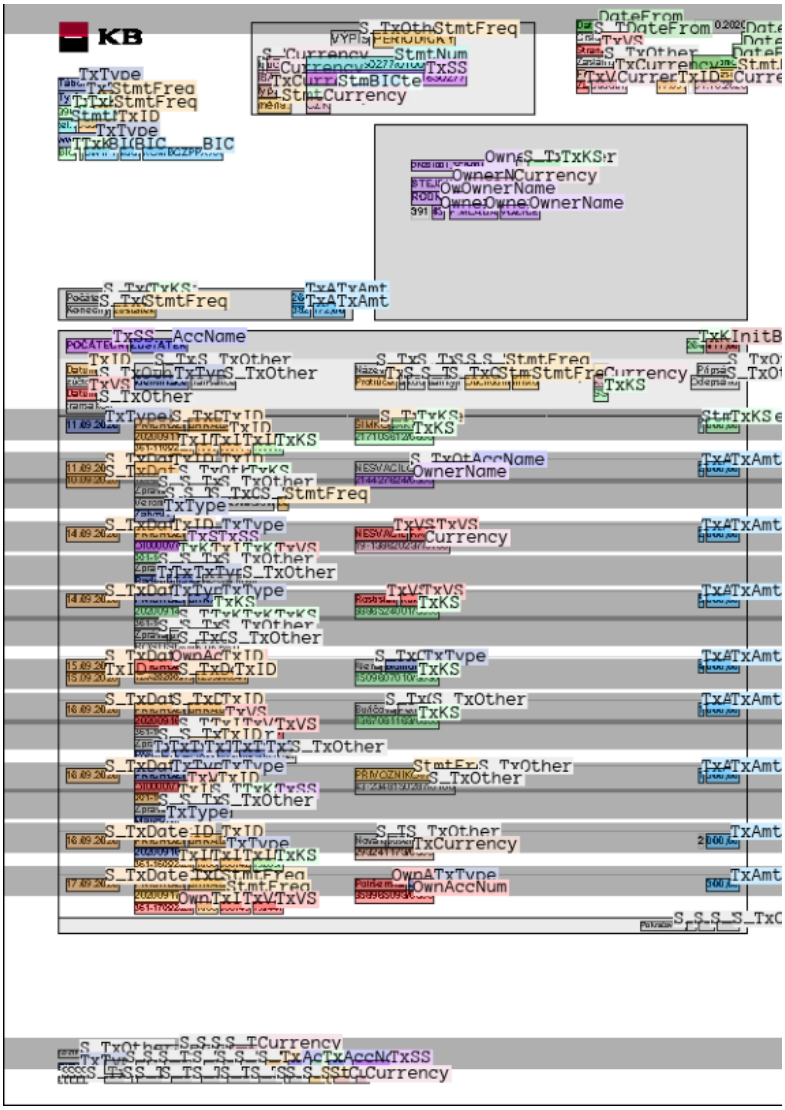
Obrázek 6.2 Ukázka zpracování vzorku z testovací množiny modelem. Vlevo se nachází vstupní mřížka, tedy jednotlivé znaky které jsou ve one-hot kódované formě předány modelu, dále je vykreslen cílový výstup detekce KILE tříd následovaný skutečným výstupem modelu

třída	N	TP	FN	FP	TN	přesnost	výtěžnost	F1
bez třídy	54484	8522	45962	7857	103804	0.520	0.156	0.241
BIC	170	165	5	196	165779	0.457	0.971	0.621
IBAN	528	0	528	0	165617	-	-	-
StmtFreq	91	87	4	2339	163715	0.036	0.956	0.069
StmtNum	383	277	106	135	165627	0.672	0.723	0.697
AccName	163	158	5	1442	164540	0.099	0.969	0.179
OwnerName	1299	1218	81	1018	163828	0.545	0.938	0.689
OwnAccNum	1134	979	155	3599	161412	0.214	0.863	0.343
Currency	158	78	80	535	165452	0.127	0.494	0.202
DateFrom	250	178	72	1415	164480	0.112	0.712	0.193
DateTo	279	0	279	0	165866	-	-	-
StmtDate	229	207	22	486	165430	0.299	0.904	0.449
InitBalance	222	196	26	1424	164499	0.121	0.883	0.213
FinalBalance	244	0	244	0	165901	-	-	-
DebitTurnover	104	0	104	0	166041	-	-	-
CreditTurnover	110	0	110	0	166035	-	-	-
TxAmt	5044	3877	1167	401	160700	0.906	0.769	0.832
S_TxOther	53999	45004	8995	22665	89481	0.665	0.833	0.740
TxBalance	175	0	175	0	165970	-	-	-
TxType	10987	9632	1355	10642	144516	0.475	0.877	0.616
TxID	7717	7373	344	3123	155305	0.702	0.955	0.810
S_TxDate	12994	11757	1237	1910	151241	0.860	0.905	0.882
TxAccNum	9521	0	9521	0	156624	-	-	-
TxCurrency	297	294	3	307	165541	0.489	0.990	0.655
TxVS	4470	4091	379	6155	155520	0.399	0.915	0.556
TxKS	717	688	29	4624	160804	0.130	0.960	0.228
TxSS	376	305	71	786	164983	0.280	0.811	0.416
							mikro F1	0.572

Tabulka 6.1 Tabulka vyhodnocení modelu strojového učení



Obrázek 6.3 Matice záměn tříd pro model strojového učení vypočtená na bázi jednotlivých znaků. Vertikální osa představuje cílovou třídu a horizontální predikovanou, hodnoty matice jsou pro lepší čitelnost přeškálovány do logaritmické škály $\langle 0, 1 \rangle$.



Obrázek 6.4 Výsledek inference modelu s šedě vyznačenými predikcemi LI a popisky tříd (původní zdroj: [62])

6.3 Heuristický model

Heuristický model původně nebyl určen k inferenci, ale většina potřebné implementace již byla vytvořena v rámci předanotace datasetu 5.6. Je zde využito obou částí algoritmu, kdy první z nich detekuje jednotlivé LI a druhé, která detekuje třídy dokumentu, s rozdílem, že proces v tomto případě probíhá bez manuálních oprav mezi dílčími částmi. Zbývalo vytvořit pouze vhodnou interpretaci dílčích výsledků, které jsou výstupy jednotlivých částí algoritmu a jejich následné předání, protože v původním užití se nepracuje přímo s výstupy, ale s nově vzniklými soubory anotací vzniklých manuální kontrolou za využití nástroje Label Studio. K tomuto účelu byly použity již hotové objektové reprezentace prvků dokumentu.

6.4 Vyhodnocení heuristického řešení

Heuristické řešení nebylo původně plánováno pro přímé použití, ale již vykazuje poměrně dobrý potenciál oproti modelu strojového učení, pro který je nutné mít velké množství trénovacích data. Na obrázku 6.5 je vykreslena matice záměn odhadů tříd dokumentů po sekvenčním spuštění obou částí[3] řešení bez manuálních zásahů. Hodnoty jsou počítány rovněž na základě jednotlivých znaků, ale s tím rozdílem, že pro vyhodnocení byl použit celý dataset, dosažené výsledky pak shrnuje tabulka 6.2. Průměrná chyba počtu detekovaných LI, tedy absolutní hodnoty rozdílu správného počtu od detekovaného je 0,052.

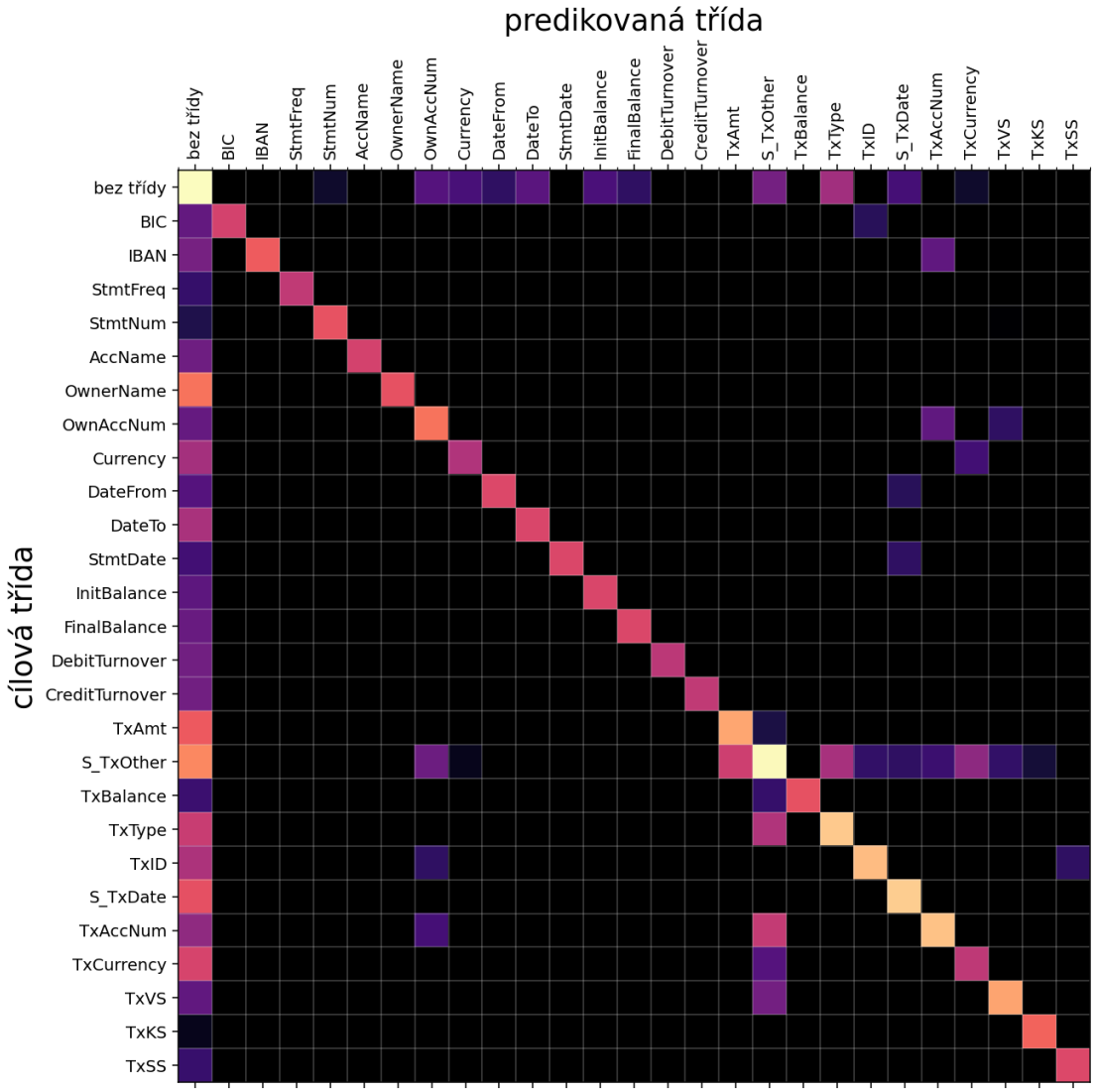
6.5 Diskuse výsledků heuristického řešení

Z matice záměn lze vyčíst několik informací, předně diagonála je celá poměrně dobře pokryta, zjednodušená třída `S_TxOther`, která vznikla sloučením tříd `TxAV` a `TxDI` ze své podstaty pokrývá velké množství znaků[4], ale sloučením došlo také k tomu, že obdobným způsobem přesunuly chyby z obou tříd původních. Dále v některých případech dochází k chybám části algoritmu, která má za cíl detekci jednotlivých LI, což je vidět u záměn tříd tabulky transakcí s prefixem `Tx` se třídami vyskytující se pouze mimo tabulku. Algoritmus chybně přijme jako LI i oblasti, které se nacházejí mimo tabulku transakcí v případech, kdy na stránce skutečná tabulka transakcí chybí a nalezená tabulka s naprosto odlišným významem je kvůli přítomnosti kalendářního data chybně interpretována. Tento jev byl pozorován při ručních kontrolách anotace datasetu LIR a chybné vzorky byly následně ručně upraveny pro omezení dalších chyb předanotace KILE datasetu.

Další vlastností je větší množství znaků, kterým nebyla přiřazena třída, přestože v cílové anotaci ji přiřazenou mají. Z pohledu tohoto řešení to znamená, že v některých případech nedochází ke korektní detekci, naopak z pohledu kvality datasetu tento fakt dokazuje, že při manuální kontrole datasetu bylo prováděno velké množství manuálních úprav, což zvyšuje kvalitu výsledného datasetu a větší pravděpodobnost využití pro model strojového učení. Konfigurace heuristického řešení nebyla od chvíle vytvoření datasetu nijak měněna.

Jelikož při tvorbě předanotací, pro které toto řešení původně vzniklo, docházelo zejména v rané fázi ke změnám konfigurace na základě nových typů vzorků a nepočítalo se s dalším využitím tohoto řešení, nebylo vyhodnocení omezeno pouze na testovací množinu, jako v případě modelu. Nebylo by možné totiž zaručit separaci vzorků, které byly použity v procesu tvorby pravidel, protože bohužel nebyly nikam poznamenávány. Z tohoto důvodu se nezdálo přínosné dělit dataset na části, protože by se snížil počet vzorků a potenciálnímu zkreslení by se stejně nebylo možné vyhnout. Riziko „přeučení“ by mělo být menší, protože pravidla jsou velmi obecná a za celou dobu tvorby nedošlo k tomu, že by se iterativně měnila jakákoli konstanta s cílem vyřešit konkrétní dokument. Změny byly provedeny zejména případech, kdy dosud neznámý typ výpisu používal jiný název pole hodnoty než v případě již zkoumaných. Veškerá pravidla jsou v aktuálním stavu aplikována bez rozdílu chování a ve stejném pořadí na všech výpisech bez ohledu na banku, což znamená, že přidání pravidla zaměřeného na specifickou banku může výsledky pro ostatní banky naopak zhoršit a nelze tak dosáhnout „přeučení“ na téměř všech vzorcích současně.

[3] Heuristických řešení pro LIR a KILE.
[4] Viz souřadnice na diagonále.



Obrázek 6.5 Matice záměn tříd pro heuristické řešení vypočtená na bázi jednotlivých znaků. Vertikální osa představuje cílovou třídu a horizontální predikovanou, hodnoty matice jsou pro lepší čitelnost přeškálovány do logaritmické škály $\langle 0, 1 \rangle$

třída	N	TP	FN	FP	TN	přesnost	výtěžnost	F1
bez třídy	546757	546118	639	47483	1102414	0.920	0.999	0.958
BIC	1962	1903	59	0	1694692	1.000	0.970	0.985
IBAN	5736	5592	144	0	1690918	1.000	0.975	0.987
StmtFreq	1046	1034	12	0	1695608	1.000	0.989	0.994
StmtNum	4111	4104	7	3	1692540	0.999	0.998	0.999
AccName	1953	1877	76	0	1694701	1.000	0.961	0.980
OwnerName	15407	3929	11478	0	1681247	1.000	0.255	0.406
OwnAccNum	11841	11729	112	132	1684681	0.989	0.991	0.990
Currency	1085	631	454	24	1695545	0.963	0.582	0.725
DateFrom	2633	2593	40	10	1694011	0.996	0.985	0.990
DateTo	2876	2377	499	39	1693739	0.984	0.826	0.898
StmtDate	2498	2470	28	0	1694156	1.000	0.989	0.994
InitBalance	2443	2400	43	23	1694188	0.991	0.982	0.986
FinalBalance	2505	2443	62	10	1694139	0.996	0.975	0.985
DebitTurnover	975	892	83	0	1695679	1.000	0.915	0.956
CreditTurnover	1099	1016	83	0	1695555	1.000	0.924	0.961
TxAmt	52578	47060	5518	1591	1642485	0.967	0.895	0.930
S_TxOther	514997	491870	23127	1994	1179663	0.996	0.955	0.975
TxBalance	3919	3893	26	0	1692735	1.000	0.993	0.997
TxType	126123	124156	1967	872	1569659	0.993	0.984	0.989
TxID	86163	85579	584	19	1610472	1.000	0.993	0.996
S_TxDate	143543	139696	3847	47	1553064	1.000	0.973	0.986
TxAccNum	105188	103814	1374	110	1591356	0.999	0.987	0.993
TxCurrency	3196	986	2210	231	1693227	0.810	0.309	0.447
TxVS	45899	45762	137	22	1650733	1.000	0.997	0.998
TxKS	7454	7452	2	4	1689196	0.999	1.000	1.000
TxSS	2667	2654	13	10	1693977	0.996	0.995	0.996
							mikro F1	0.969

Tabulka 6.2 Tabulka vyhodnocení heuristického modelu

6.6 Výsledná implementace

Výsledné řešení se skládá z dvojice spustitelných Python skriptů, které převedou daný dokument ve formátu PDF do strukturovaného formátu JSON. První z nich využívá heuristické řešení a je viditelně přesnější, druhý pak umělou neuronovou síť architektury Chargrid. Ukázky výstupních dokumentů se nacházejí v příloze spolu se zmíněnými Python skripty a instrukcemi pro spuštění.

Kapitola 7
Diskuse

Hlavním rizikem vytvořeného datasetu je možnost, že některé případy výskytů tříd mohly být v rámci manuálních úprav vzorků označovány nekonzistentně. Ve skutečnosti také mohlo dojít k chybné interpretaci výpisu a zavedení konzistentní chyby do celého procesu. Pro úplné ověření kvality anotací je třeba konzultovat výsledky s odborníky v oboru a případně provést nutné úpravy. V ideálním případě by bylo zanalyzováno a následně vytvořeno alespoň menší množství zaručeně správných vzorků pro různé různé banky, které by plnily funkci regresních testů. Tento postup by alespoň částečně eliminoval riziko, že bude na rozšíření datasetu vynaloženo mnoho dalšího úsilí a času, ale nakonec dojde ke zjištění, že celý proces byl chybný, a tím pádem je dataset méně kvalitní. V takovém případě by chyba šla částečně opravit zpětným dohledáním vzorků na základě data jejich vzniku, které by bylo porovnáno s datem změny způsobu anotace. Nejlepší volbou by bylo vytvoření více anotací stejných dokumentů nezávislými experty, kteří se v této doméně orientují.

Model zatím nedosahuje příliš využitelných výsledků, ale zdá se, že jistý potenciál přeci jen představuje. Na obrázku 6.4 výstupu modelu lze vidět, že na místech skutečných výskytů tříd `S_TxDate` a `TxAmt` je téměř ve všech případech správně rozpozná.

Možnost vytvoření syntetického datasetu se nezdá jako pravděpodobná, je ale možnost vytvořit polosyntetický dataset na základě již existujících vzorků. Například by mělo být možné dosud anotované vzorky rozdělit na jednotlivé transakce a náhodně je vkládat do výpisu s odstraněnou tabulkou. Také se nabízí vytvořit generátory některých hodnot tříd a původní data nahrazovat generovanými, problémem je v tomto případě docílení přesné kopie rozložení znaků a volby fontu.

Program je poměrně rychlý, a to i přesto, že v některých případech provádí velké množství iterativních operací, kterým je zvykem se při práci v Pythonu co nejvíce vyhýbat a delegovat tyto činnosti k nativně implementovaným knihovnám. Jediný případ, kdy se tato zpomalení projevují je vyhodnocení, která je, jak již bylo zmíněno, prováděno na jednotlivých znacích dokumentu, což znatelně zvyšuje komplexitu výpočtu. Řešením tohoto problému by mohlo být nalezení vhodné reprezentace dokumentu například v rámci nativně implementované knihovny NumPy [63], či přepsání náročných operací do nativního kódu.

V případě navázání na tuto práci se nabízí prozkoumat modely určené pro klasickou segmentaci obrázku se zachováním stávající reprezentace vstupních dat Chargrid. Od doby vzniku Chargrid došlo k mnoha posunům v oblasti segmentace, což dokazuje například síť Detectron2 [64] a je tedy velice pravděpodobné, že některé z nich bude dosahovat mnohem lepších výsledků. Další možností je vytvořit 2 řešení, jedno pro úlohu detekce transakcí a druhou pro extrakci klíčových informací. Jako reálná se jeví i možnost docílení výsledků jedné z těchto činností heuristickým řešením. Také by mohlo být možným postupem rozdělení dokumentu na transakce a následně provádět extrakci informací z řádků transakcí samostatně, případně s kontextem dete-

kované hlavičky.

Otázkou pro další výzkum je snížení velikosti reprezentace dokumentu, které Chargrid využívá, jelikož zvyšování počtu příznaků, tedy kanálů „obrázku“ se stoupající hloubkou sítě je poměrně častým jevem hlubokého učení a velikost zpracovaných dat pak velice rychle narůstá. Nabízí se tak možnost snížení počtu tříd znaků jejich sloučením nebo zvolením jiné reprezentace jako v případě BERTgrid [65], kde jsou texty rozděleny na tokeny a následně reprezentovány pomocí embeddingů pro úspornější zakódování většího množství informací, které lépe vyhovují interní reprezentaci příznaků sítě.

Kapitola 8
Závěr

V rámci práce byla provedena rešerše oblasti modelů strojového učení, jejich trénovacích datasetů, veřejně dostupných služeb komerčního charakteru a také jedno neveřejné řešení z domény bankovnictví.

Při analýze modelů strojového učení byl dbán důraz na možnost co nejpřímočařejšího využití pro extrakci klíčových informací z dokumentu, bohužel na tuto úlohu většinu modelů nelze přímo aplikovat, protože jsou zaměřeny na mřížkovou reprezentaci buněk tabulky. Datasety, které napomáhaly vývoji těchto modelů často neobsahují dostatek informací pro řešení cílů práce nebo je zohledňují pouze částečně. Dále bylo zjištěno, že tvorba datasetu významné velikosti je manuální cestou prakticky nemožná, což dosvědčuje zvolený automatizovaný způsob tvorby téměř všech zkoumaných datasetů využívající původní strukturované reprezentace nestrukturovaných dokumentů. Žádný veřejně dostupný dataset obsahující výpisy z bankovních účtů se nepodařilo nalézt, dokonce ani v surové formě bez vyznačených tříd.

V oblasti komerčních řešení je kladen důraz na možnost zpracování co největší škály dokumentů, což pro uživatele znamená nutnost ruční přípravy menšího množství trénovacích vzorků přímo uvnitř uživatelského rozhraní aplikace. Zjevnou nevýhodou těchto řešení je povinnost za službu platit při zpracování většího množství dokumentů. Také některým uživatelům nemusí být příjemné sdílet svá citlivá data se komerčními společnostmi, a to přesto, že v Evropské unii platí GDPR a uživatel je z právního hlediska poměrně dobře chráněn. Celkově testovaná řešení nedosahovala příliš dobrých výsledků.

V rámci práce byl z veřejně dostupných dat vytvořen dataset s 878 vzorky stránek výpisů z účtů, který umožňuje řešit úlohu extrakce 26 tříd klíčových informací z dokumentu. Pro tvorbu bylo vyvinuto částečně automatizované řešení, které tvorbu dělí do dvou fází a v rámci každé jsou vzorky manuálně kontrolovány. Díky tomu je dosaženo značné úspory času, usnadnění práce, zabránění vzniku chyb a zvýšení úrovně konzistence než při čistě manuálním postupu. V rámci práce byly zmíněny problémy, které nastaly během této činnosti a jsou pravděpodobně překážkou tvorby velkých datasetů pro extrakci klíčových informací z nestrukturovaných dokumentů i v jiných doménách.

Pro správu vzorků a podkladových dokumentů bylo naprogramováno úložiště, které umožňuje vyhledat informace o původu jednotlivých vzorků a chybné dávkově odstraňovat či přegenerovat. Díky tomu je v budoucnu možné efektivně spravovat i větší množství vzorků.

Vytvořený model dosahuje nedostatečných výsledků v porovnání s heuristickým řešením, což je pravděpodobně způsobeno relativně malou velikostí datasetu pohybující se v řádu vyšších stovek vzorků oproti desetitisícům až statisícům vzorků obvyklých pro oblast strojového učení umělých neuronových sítí, také jistou roli hraje rozmanitost vzorků, jelikož dataset obsahuje výpisy různých bank. Heuristické řešení dokáže s dostatečnou přesností extrahovat důležité informace o jednotlivých transakcích, především variabilní symbol pro párování faktur. Samotný

fakt, že jednotlivé transakce lze rozlišit otevírá další možnosti pro zpracování klíčových informací.

Dalším pozitivním faktem je, že při tvorbě byla věnována pozornost flexibilitě konfigurace automatizovaného řešení, kterou by bylo možné volit podle konkrétní banky výpisu a tím pravděpodobně dosáhnout opravdu vysoké přesnosti. Dosud této možnosti nebylo využito, ale vzhledem k zjištění, že tento postup je používaný v produkčním prostředí nejmenované banky se jeví jako velice perspektivní. V případě nasazení systému do produkce by mohlo toto řešení zároveň sloužit k vytváření datasetu z nových dat, což by následně umožnilo model učit na větším množství vzorků a zlepšit tak jeho výsledky.

Výsledný program je dodán jako jednoduchá aplikace pro příkazovou řádku a umožňuje využití jak modelu strojového učení, tak heuristického řešení, které již ve stávající podobě funguje dostatečně dobře.

Příloha A

Zobrazení tříd na celočíselné indexy

Níže se nachází zobrazení, použitá pro převod třídy na numerický index pro vytvoření reprezentace dat modelu strojového učení.

A.1 Zobrazení znaků na indexy

```
[None] -> 0
['0'] -> 1
['1'] -> 2
['2'] -> 3
['3', '4', '5', '6', '7', '8', '9'] -> 4
['a'] -> 5
['b'] -> 6
['c'] -> 7
['d'] -> 8
['e'] -> 9
['f'] -> 10
['g'] -> 11
['h'] -> 12
['i'] -> 13
['j'] -> 14
['k'] -> 15
['l'] -> 16
['m'] -> 17
['n'] -> 18
['o'] -> 19
['p'] -> 20
['q'] -> 21
['r'] -> 22
['s'] -> 23
['t'] -> 24
['u'] -> 25
['v'] -> 26
['w'] -> 27
```

```
['x'] -> 28
['y'] -> 29
['z'] -> 30
['A'] -> 31
['B'] -> 32
['C'] -> 33
['D'] -> 34
['E'] -> 35
['F'] -> 36
['G'] -> 37
['H'] -> 38
['I'] -> 39
['J'] -> 40
['K'] -> 41
['L'] -> 42
['M'] -> 43
['N'] -> 44
['O'] -> 45
['P'] -> 46
['Q'] -> 47
['R'] -> 48
['S'] -> 49
['T'] -> 50
['U'] -> 51
['V'] -> 52
['W'] -> 53
['X'] -> 54
['Y'] -> 55
['Z'] -> 56
['-'] -> 57
[':'] -> 58
[','] -> 59
['.'] -> 60
['/'] -> 61
['€'] -> 62
ANY_OTHER -> 63
class count: 64
```

A.2 Zobrazení tříd KILE

```
[None] -> 0
['BIC'] -> 1
['IBAN'] -> 2
['StmtFreq'] -> 3
['StmtNum'] -> 4
['AccName'] -> 5
['OwnerName'] -> 6
['OwnAccNum'] -> 7
['Currency'] -> 8
['DateFrom'] -> 9
['DateTo'] -> 10
['StmtDate'] -> 11
```

```
['InitBalance'] -> 12
['FinalBalance'] -> 13
['DebitTurnover'] -> 14
['CreditTurnover'] -> 15
['TxAmt'] -> 16
['S_TxOther'] -> 17
['TxBalance'] -> 18
['TxType'] -> 19
['TxID'] -> 20
['S_TxDate'] -> 21
['TxAccNum'] -> 22
['TxCurrency'] -> 23
['TxVS'] -> 24
['TxKS'] -> 25
['TxSS'] -> 26
class count: 27
```

A.3 Zobrazení tříd LIR

```
[None, 'Header'] -> 0
['LineItem'] -> 1
class count: 2
```

Bibliografie

1. NOVÝ, Josef. Aplikace pro standardizaci hromadného zpracování, analýzu a vizualizaci finančních toků z definovaných bankovních informací zpracovaných u PČR v rámci odhalování trestné činnosti | National electronic tool [online]. 2023 [cit. 2023-03-08]. Dostupné z: https://nen.nipez.cz/en/verejne-zakazky/detail-zakazky/N006-19-V00030390.
2. *Jak rozumět bankovním výpisům? | Banky.cz* [online]. [B.r.]. [cit. 2023-04-15]. Dostupné z: https://www.banky.cz/clanky/jak-rozumet-bankovnim-vypisum/.
3. *IBAN a BIC - využití v platebním styku - Česká národní banka* [online]. [B.r.]. [cit. 2023-04-15]. Dostupné z: https://www.cnb.cz/cs/platebni-styk/iban/iban-a-bic-vyuziti-v-platebnim-styku/.
4. *Výpis z účtu* [online]. [B.r.]. [cit. 2023-04-18]. Dostupné z: http://svhlavacova.jamrtal.cz/documents/2016/04/vypis-z-uctu-3-2016.pdf.
5. SKALICKÝ, Matyáš; ŠIMSA, Štěpán; UŘIČÁŘ, Michal; ŠULC, Milan. *Business Document Information Extraction: Towards Practical Benchmarks*. 2022. Dostupné z arXiv: 2206.11229 [cs.IR].
6. *08_2020.pdf* [online]. [B.r.]. [cit. 2023-04-18]. Dostupné z: https://www.uprimnesrdce.cz/wp-content/uploads/2020/09/08_2020.pdf.
7. SCHELLEKENS, Joris. *Borb Examples: About structured vs. unstructured document formats* [online]. [B.r.]. [cit. 2023-04-16]. Dostupné z: https://github.com/MichalLebeda/borb-examples%5C#87-about-structured-vs-unstructured-document-formats.
8. SCHELLEKENS, Joris. *Borb Examples: About structured vs. unstructured document formats* [online]. [B.r.]. [cit. 2023-04-16]. Dostupné z: https://github.com/MichalLebeda/borb-examples%5C#87-about-structured-vs-unstructured-document-formats. (vlastní překlad, fork originálního repozitáře kvůli opravě chyby).
9. SCHELLEKENS, Joris. *Borb Examples: Deep Dive* [online]. [B.r.]. [cit. 2023-04-16]. Dostupné z: https://github.com/MichalLebeda/borb-examples%5C#8-deep-dive-into-borb. (vlastní překlad, fork originálního repozitáře kvůli opravě chyby).
10. CID, Albert Astals. Poppler [online]. 2023 [cit. 2023-04-30]. Dostupné z: https://poppler.freedesktop.org/.
11. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep Learning* [online]. MIT Press, 2016 [cit. 2023-05-01]. Dostupné z: http://www.deeplearningbook.org. (vlastní překlad, matematické značení upraveno).
12. KAREL KLOUDA Juan Pablo Maldonado Lopez, Daniel Vašata. BI-VZD-11-cs-slides.pdf [online]. 2023 [cit. 2023-05-08]. Dostupné z: https://courses.fit.cvut.cz/BI-VZD/lectures/files/BI-VZD-11-cs-slides.pdf.

13. ZHANG, Aston; LIPTON, Zachary C.; LI, Mu; SMOLA, Alexander J. Dive into Deep Learning. *arXiv preprint arXiv:2106.11342*. 2021.

14. AUNG, Robert. Do convolutional neural networks mimic the human visual system? | MSAIL [online]. 2023 [cit. 2023-04-30]. Dostupné z: `https://msail.github.io/post/cnn_human_visual/`.

15. IOFFE, Sergey; SZEGEDY, Christian. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR*. 2015, roč. abs/1502.03167. Dostupné z arXiv: `1502.03167`.

16. HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. *Deep Residual Learning for Image Recognition*. 2015. Dostupné z arXiv: `1512.03385 [cs.CV]`.

17. FENG, Weijiang; GUAN, Naiyang; LI, Yuan; ZHANG, Xiang; LUO, Zhigang. Audio visual speech recognition with multimodal recurrent neural networks. In: 2017, s. 681–688. Dostupné z DOI: `10.1109/IJCNN.2017.7965918`.

18. VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N.; KAISER, Lukasz; POLOSUKHIN, Illia. *Attention Is All You Need*. 2017. Dostupné z arXiv: `1706.03762 [cs.CL]`.

19. PRAKASH, Amit. Transformer architecture: The engine behind ChatGPT [online]. 2023 [cit. 2023-04-30]. Dostupné z: `https://www.thoughtspot.com/data-trends/ai/what-is-transformer-architecture-chatgpt`.

20. NASSAR, Ahmed; LIVATHINOS, Nikolaos; LYSAK, Maksym; STAAR, Peter. *TableFormer: Table Structure Understanding with Transformers*. 2022. Dostupné z arXiv: `2203.01017 [cs.CV]`.

21. ZHONG, Xu; SHAFIEIBAVANI, Elaheh; YEPES, Antonio Jimeno. *Image-based table recognition: data, model, and evaluation*. 2020. Dostupné z arXiv: `1911.10683 [cs.CV]`.

22. ZHENG, Xinyi; BURDICK, Doug; POPA, Lucian; ZHONG, Xu; WANG, Nancy Xin Ru. *Global Table Extractor (GTE): A Framework for Joint Table Identification and Cell Structure Recognition Using Visual Context*. 2020. Dostupné z arXiv: `2005.00589 [cs.CV]`.

23. *Machine Learning Essentials: What is Data Annotation?* [online]. [B.r.]. [cit. 2023-04-17]. Dostupné z: `https://www.defined.ai/blog/machine-learning-essentials-what-is-data-annotation/`.

24. What is Data Labeling? | IBM [online]. [B.r.] [cit. 2023-04-15]. Dostupné z: `https://www.ibm.com/topics/data-labeling`.

25. *The Ultimate Guide to Data Labeling for Machine Learning* [online]. [B.r.]. [cit. 2023-04-17]. Dostupné z: `https://www.cloudfactory.com/data-labeling-guide#Lets-Get-Started-brLabeled-data-and-ground-truth2`.

26. GEIGER, R. Stuart; COPE, Dominique; IP, Jamie; LOTOSH, Marsha; SHAH, Aayush; WENG, Jenny; TANG, Rebekah. "Garbage in, garbage out" revisited: What do machine learning application papers report about human-labeled training data? *Quantitative Science Studies*. 2021, roč. 2, č. 3, s. 795–827. Dostupné z DOI: `10.1162/qss_a_00144`.

27. Bounding boxes - IBM Documentation [online]. [B.r.] [cit. 2023-04-06]. Dostupné z: `https://www.ibm.com/docs/en/informix-servers/12.10?topic=structure-bounding-boxes`.

28. ibm-aur-nlp/PubTabNet [online]. [B.r.] [cit. 2023-04-15]. Dostupné z: `https://github.com/ibm-aur-nlp/PubTabNet`.

29. SMOCK, Brandon; PESALA, Rohith; ABRAHAM, Robin. *PubTables-1M: Towards comprehensive table extraction from unstructured documents*. 2021. Dostupné z arXiv: `2110.00061 [cs.LG]`.

30. LEMIRE, Daniel; VELLINO, Andre. *Extracting, Transforming and Archiving Scientific Data*. 2011. Dostupné z arXiv: 1108.4041 [cs.DL].

31. LI, Minghao; CUI, Lei; HUANG, Shaohan; WEI, Furu; ZHOU, Ming; LI, Zhoujun. *TableBank: A Benchmark Dataset for Table Detection and Recognition*. 2020. Dostupné z arXiv: 1903.01949 [cs.CV].

32. WEI, Jiaheng; ZHU, Zhaowei; CHENG, Hao; LIU, Tongliang; NIU, Gang; LIU, Yang. *Learning with Noisy Labels Revisited: A Study Using Real-World Human Annotations*. 2022. Dostupné z arXiv: 2110.12088 [cs.LG].

33. WANG, Xinxin. *Tabular Abstraction, Editing, and Formatting*. 1996. Dostupné také z: https://hdl.handle.net/10012/10962. Dis. pr. University of Waterloo, Ontario, Canada.

34. HU, Jianying; KASHI, Ramanujan; LOPRESTI, Daniel; WILFONG, G. Table structure recognition and its evaluation. 2001.

35. TENGLI, Ashwin; YANG, Yiming; MA, Nian Li. Learning Table Extraction from Examples. In: *Proceedings of the 20th International Conference on Computational Linguistics*. Geneva, Switzerland: Association for Computational Linguistics, 2004, 987–es. COLING '04. Dostupné z DOI: 10.3115/1220355.1220497.

36. PFITZMANN, Birgit; AUER, Christoph; DOLFI, Michele; NASSAR, Ahmed S.; STAAR, Peter. DocLayNet: A Large Human-Annotated Dataset for Document-Layout Segmentation. In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, 2022. Dostupné z DOI: 10.1145/3534678.3539043.

37. MEMON, Shahan Ali; ZHAO, Wenbo; RAJ, Bhiksha; SINGH, Rita. Neural Regression Trees. In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019. Dostupné z DOI: 10.1109/ijcnn.2019.8852133.

38. PRECIOSO, Daniel; GÓMEZ-ULLATE, David. *NILM as a regression versus classification problem: the importance of thresholding*. 2020. Dostupné z arXiv: 2010.16050 [eess.SP].

39. YANG, Suorong; XIAO, Weikang; ZHANG, Mengcheng; GUO, Suhan; ZHAO, Jian; SHEN, Furao. *Image Data Augmentation for Deep Learning: A Survey*. 2022. Dostupné z arXiv: 2204.08610 [cs.CV].

40. SRIVASTAVA, Nitish; HINTON, Geoffrey; KRIZHEVSKY, Alex; SUTSKEVER, Ilya; SALAKHUTDINOV, Ruslan. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*. 2014, roč. 15, č. 56, s. 1929–1958. Dostupné také z: http://jmlr.org/papers/v15/srivastava14a.html.

41. HUANG, Zheng; CHEN, Kai; HE, Jianhua; BAI, Xiang; KARATZAS, Dimosthenis; LU, Shijian; JAWAHAR, C. V. ICDAR2019 Competition on Scanned Receipt OCR and Information Extraction. In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. 2019, s. 1516–1520. Dostupné z DOI: 10.1109/ICDAR.2019.00244.

42. AL, Smock Et. microsoft/table-transformer-structure-recognition · Hugging Face [online]. 2023 [cit. 2023-05-08]. Dostupné z: https://huggingface.co/microsoft/table-transformer-structure-recognition.

43. KATTI, Anoop Raveendra; REISSWIG, Christian; GUDER, Cordula; BRARDA, Sebastian; BICKEL, Steffen; HÖHNE, Johannes; FADDOUL, Jean Baptiste. *Chargrid: Towards Understanding 2D Documents*. 2018. Dostupné z arXiv: 1809.08799 [cs.CL].

44. REN, Shaoqing; HE, Kaiming; GIRSHICK, Ross; SUN, Jian. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. Dostupné z arXiv: 1506.01497 [cs.CV].

45. HE, Jiajun; WU, Ping; TONG, Yizhi; ZHANG, Xujie; LEI, Meizhen; GAO, Jinfeng. Bearing Fault Diagnosis via Improved One-Dimensional Multi-Scale Dilated CNN. *Sensors*. 2021, roč. 21, č. 21. ISSN 1424-8220. Dostupné z DOI: 10.3390/s21217319.

46. Home | parseur.com [online]. 2023 [cit. 2023-05-08]. Dostupné z: https://app.parseur.com/.
47. Online Document Parser - GroupDocs.Parser App [online]. 2023 [cit. 2023-05-08]. Dostupné z: https://products.groupdocs.app/parser/total?fileName=08_2020.pdf&folderName=f01257b4-2484-437a-973c-0646ae7b8b0f.
48. *GroupDocs App Reviews | Read Customer Service Reviews of groupdocs.app* [online]. [B.r.]. [cit. 2023-05-09]. Dostupné z: https://www.trustpilot.com/review/groupdocs.app.
49. Document Parsers [online]. 2023 [cit. 2023-05-08]. Dostupné z: https://app.docparser.com/.
50. *evt_file.php* [online]. [B.r.]. [cit. 2023-04-18]. Dostupné z: https://www.pristoupim.cz/evt_file.php?file=3843.
51. *aktual.pdf* [online]. [B.r.]. [cit. 2023-04-18]. Dostupné z: https://www.darcovskasms.cz/banka/aktual.pdf?page=dms-bankovni-ucet3.
52. OpenCV: Miscellaneous Image Transformations. 2023. Dostupné také z: https://docs.opencv.org/4.x/d7/d1b/group_imgproc_misc.html#ga72b913f352e4a1b1b397736707afcde3.
53. *OpenCV: Image Thresholding* [online]. OpenCV, [b.r.] [cit. 2023-04-06]. Dostupné z: https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html. (vlastní překlad).
54. *Výpis* [online]. [B.r.]. [cit. 2023-04-18]. Dostupné z: https://www.scxaverov.cz/download/42/10-2014.pdf.
55. *PyTorch* [online]. [B.r.]. [cit. 2023-05-09]. Dostupné z: https://pytorch.org/.
56. *torch.utils.data — PyTorch 2.0 documentation* [online]. [B.r.]. [cit. 2023-05-09]. Dostupné z: https://pytorch.org/docs/stable/data.html#torch.utils.data.Dataset.
57. *Google Colaboratory* [online]. [B.r.]. [cit. 2023-05-09]. Dostupné z: https://colab.research.google.com/.
58. K, Sambasivarao. Non-maximum Suppression (NMS). A Technique to remove duplicates and… | by Sambasivarao. K | Towards Data Science [online]. [B.r.] [cit. 2023-05-09]. Dostupné z: https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c.
59. *SGD — PyTorch 2.0 documentation* [online]. [B.r.]. [cit. 2023-05-09]. Dostupné z: https://pytorch.org/docs/stable/generated/torch.optim.SGD.html.
60. *AdamW — PyTorch 2.0 documentation* [online]. [B.r.]. [cit. 2023-05-09]. Dostupné z: https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html.
61. GUGGER, Sylvain; HOWARD, Jeremy. *fast.ai - AdamW and Super-convergence is now the fastest way to train neural nets*. [B.r.]. Dostupné také z: https://www.fast.ai/posts/2018-07-02-adam-weight-decay.html. (Accessed on 05/10/2023).
62. 1232301830277_3_1132_20201001.pdf [online]. [B.r.] [cit. 2023-04-09]. Dostupné z: https://rodna.eu/attachments/article/220/1232301830277%5C_3%5C_1132_20201001.pdf.
63. HARRIS, Charles R.; MILLMAN, K. Jarrod; WALT, Stéfan J. van der; GOMMERS, Ralf; VIRTANEN, Pauli; COURNAPEAU, David; WIESER, Eric; TAYLOR, Julian; BERG, Sebastian; SMITH, Nathaniel J.; KERN, Robert; PICUS, Matti; HOYER, Stephan; KERKWIJK, Marten H. van; BRETT, Matthew; HALDANE, Allan; RÍO, Jaime Fernández del; WIEBE, Mark; PETERSON, Pearu; GÉRARD-MARCHANT, Pierre; SHEPPARD, Kevin; REDDY, Tyler; WECKESSER, Warren; ABBASI, Hameer; GOHLKE, Christoph; OLIPHANT, Travis E. Array programming with NumPy. *Nature*. 2020, roč. 585, č. 7825, s. 357–362. Dostupné z DOI: 10.1038/s41586-020-2649-2.
64. WU, Yuxin; KIRILLOV, Alexander; MASSA, Francisco; LO, Wan-Yen; GIRSHICK, Ross. *Detectron2*. 2019. Dostupné také z: https://github.com/facebookresearch/detectron2.

65. DENK, Timo I.; REISSWIG, Christian. *BERTgrid: Contextualized Embedding for 2D Document Representation and Understanding*. 2019. Dostupné z arXiv: 1909.04948 [cs.CL].

Obsah přiloženého média

- `README.md` ... instrukce ke spuštění
- `config` ... modul konfigurace aplikace
- `dataset` ... modul implementace správy databáze vzorků
- `fonts` ... fonty pro vizualizaci
- `heuristic` ... modul implementace heuristického řešení
- `memmapize` ... modul pro uložení datasetu na disk
- `ml_model` ... modul modelu strojového učení s Jupyter notebooky
  - `checkpoints` ... adresář s uloženými váhami modelu
  - `memmaped_dataset` ... složka pro uložený datasetu modelu
  - `chargrid.py` ... implementace převodu dokumentu do reprezentace modelu
  - `chargrid_dataset.py` ... implementace datasetu vytvářející vzorky pro model
  - `chargrid_network.py` ... implementace sítě
  - `dataset_creation.ipynb` ... Jupyter notebook pro vytvoření a uložení datasetu na disk
  - `eval.ipynb` ... Jupyter notebook pro evaluaci modelu
  - `train.ipynb` ... Jupyter notebook pro trénování modelu
  - `util.py` ... Python soubor s obecnými funkcemi pro tvorbu a testování modelu
- `notebooks` ... adresář s Jupyter notebooky
- `scripts` ... adresář obsahující scripty pro správu dat a inferenci
- `constants.py` ... Python soubor s konstantami
- `document.py` ... Python objektová reprezentace prvků dokumentu
- `lib.py` ... Python soubor obsahující funkce pro rutinní operace
- `eval.py` ... Python soubor obsahující funkce pro evaluaci
- `visualize.py` ... Python soubor s funkcemi pro vizualizaci
- `requirements.txt` ... soubor s Python závislostmi
- `output_examples` ... adresář s ukázkami zpracovaných výpisů
  - `airbank.json`
  - `airbank_ml.json`
  - `csas.json`
  - `csas_ml.json`
  - `csob.json`
  - `csob_ml.json`
  - `ge.json`
  - `ge_ml.json`
  - `kb.json`
  - `kb_ml.json`
  - `moneta.json`
  - `moneta_ml.json`