VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AUTOMATICKÉ ZÍSKÁNÍ BIBLIOGRAFICKÝCH ÚDAJŮ Z DOKUMENTU

AUTOMATED EXTRACTION OF BIBLIOGRAPHIC INFORMATION FROM DOCUMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MARIE PAŘILOVÁ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JAN KOHÚT**
SUPERVISOR

BRNO 2025

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

164604

Ústav: Ústav počítačové grafiky a multimédií (UPGM)
Studentka: **Pařilová Marie**
Program: Informační technologie
Název: **Automatické získání bibliografických údajů z dokumentu**
Kategorie: Zpracování obrazu
Akademický rok: 2024/25

Zadání:

1. Vytvořte si přehled o současných přístupech pro extrakci bibliografických údajů dokumentů.
2. Navrhněte vhodnou metodu pro extrakci bibliografických údajů dokumentů.
3. Vytvořte vhodnou datovou sadu nebo rozšiřte některou z existujících datových sad.
4. Vyhodnoťte navrženou metodu na datové sadě.
5. Zhodnoťte dosažené výsledky.
6. Vytvořte krátké video prezentující výsledky vaši práce.

Literatura:

- Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).
- Sido, J., Pražák, O., Přibáň, P., Pašek, J., Seják, M. and Konopík, M., 2021. Czert--Czech BERT-like Model for Language Representation. *arXiv preprint arXiv:2103.13031*.
- Shi, B., Bai, X. and Yao, C., 2016. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, *39* (11), pp.2298-2304.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3.

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

Vedoucí práce: **Kohút Jan, Ing.**
Vedoucí ústavu: Černocký Jan, prof. Dr. Ing.
Datum zadání: 1.11.2024
Termín pro odevzdání: 14.5.2025
Datum schválení: 12.11.2024

Abstrakt

Tato práce se zabývá automatickou extrakcí bibliografických údajů z naskenovaných titulních stran historických knih s cílem usnadnit katalogizaci v knihovnách a archivech. Porovnávány jsou dva přístupy: detekční model YOLO v kombinaci s OCR a multimodální model LayoutLM, který integruje text, vizuální prvky a rozvržení stránky. Součástí řešení je také poloautomatický nástroj pro zarovnání anotací, jenž zrychluje přípravu trénovacích dat. Výstupem je strukturovaný JSON soubor s klíčovými bibliografickými prvky. Práce hodnotí přesnost a praktičnost obou přístupů.

Abstract

This thesis focuses on the automatic extraction of bibliographic data from scanned title pages of mostly historical books, aiming to facilitate cataloging in libraries and archives. Two approaches are compared: a detection model based on YOLO combined with OCR, and a multimodal LayoutLM model that integrates text, visual features, and page layout. The work also includes a semi-automatic alignment tool designed to speed up the annotation of training data. The output is a structured JSON file containing key bibliographic elements. The thesis evaluates the accuracy and practical usability of both approaches.

Klíčová slova

automatická extrakce, bibliografická metadata, digitalizace, YOLO, OCR, LayoutLM, historické dokumenty, titulní strana, zpracování obrazu, strojové učení

Keywords

automatic extraction, bibliographic metadata, digitization, YOLO, OCR, LayoutLM, historical documents, title page, image processing, machine learning

Citace

PAŘILOVÁ, Marie. *Automatické získání bibliografických údajů z dokumentu.* Brno, 2025. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jan Kohút

Automatické získání bibliografických údajů z dokumentu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana Ing. Jana Kohúta. Veškeré použité literární prameny, publikace a další zdroje, ze kterých jsem čerpala, jsou v práci citovány a uvedeny v seznamu použité literatury.

.....
Marie Pařilová
13. května 2025

Poděkování

Děkuji vedoucímu své bakalářské práce Ing. Janu Kohútovi za odborné vedení, trpělivost, cenné rady a podporu během zpracování této práce. Zároveň bych ráda poděkovala své rodině a blízkým za trpělivost, podporu a motivaci, kterou mi po celou dobu studia poskytovali.

Obsah

Seznam obrázků

Kapitola 1

Úvod

V době rozsáhlé digitalizace knihoven a archivních sbírek nabývá automatická extrakce textových informací zásadního významu pro efektivní zpřístupnění tištěných dokumentů v digitální podobě. Tradiční manuální katalogizace bibliografických údajů — jako jsou názvy publikací, autoři, nakladatelé, místo a rok vydání — je nejen zdlouhavá, ale rovněž náchylná k chybám, zejména při zpracování historických tiskovin s různorodou typografií a rozvržením.

Tato práce porovnává dva přístupy ke zpracování titulních stran knih s cílem navrhnout systém, který automatizuje extrakci bibliografických údajů a minimalizuje lidský zásah. První přístup využívá detekční model YOLO (You Only Look Once) pro lokalizaci předem definovaných typů textových prvků na obrázku — například „titulek“, „autor“ nebo „rok vydání“. Každá detekovaná oblast (bounding box) je poté zpracována pomocí OCR, které převádí vizuální informace na strojově čitelný text.

Druhý přístup, založený na jazykovém modelu LayoutLM, zpracovává celý dokument najednou jako multimodální vstup — tedy současně text, jeho pozici a vizuální kontext stránky. Tento model umožňuje vnímat dokument jako strukturovaný celek, což výrazně zlepšuje schopnost rozpoznat kontext a odlišit podobné typy údajů. Výhodou tohoto přístupu je schopnost zachytit jemné vizuálně-jazykové vzory, které jsou klíčové například pro rozlišení podtitulků, edičních poznámek nebo místa vydání.

Součástí této práce je také návrh a implementace poloautomatického zarovnávacího nástroje, který slouží ke zrychlení a zpřesnění procesu anotace trénovacích dat — zejména při práci s rozsáhlými historickými kolekcemi, kde ruční označování představuje výraznou časovou zátěž.

Bez ohledu na zvolený přístup následuje fáze postprocessingu, která slouží k volbě nejvhodnějších predikcí a sjednocení výstupu do konzistentní podoby. Výsledkem celého procesu je strukturovaný výstupní soubor ve formátu JSON, který obsahuje extrahované bibliografické údaje a pravděpodobnostní skóre.

Oba přístupy — detekční metoda kombinující YOLO a OCR, i model LayoutLM využívající multimodální porozumění — mají své silné a slabé stránky a v rámci této práce jsou porovnávány z hlediska přesnosti a robustnosti.

Kapitola 2

Automatické zpracování bibliografických údajů

Tato kapitola se zaměřuje na objasnění základních pojmů a technologií, které tvoří teoretický základ práce. Nejprve budou popsány bibliografické údaje a jejich struktura, následuje přehled metod používaných pro analýzu dokumentů. Dále bude představena architektura modelu YOLO pro detekci objektů a modelu LayoutLM, který v sobě kombinuje jazykové, prostorové a vizuální informace. Cílem této kapitoly je poskytnout čtenáři nezbytné znalosti pro pochopení praktické části práce.

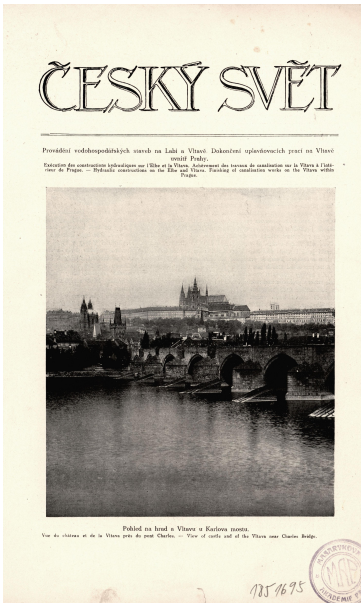
2.1 Bibliografické údaje a jejich struktura

Bibliografické údaje představují soubor informací, které slouží k jednoznačné identifikaci a popisu dokumentu, jako je kniha, článek nebo disertace. Jejich hlavní účel spočívá ve správném citování zdrojů, usnadnění vyhledávání dokumentů a jejich zařazení do knihovních katalogů či vědeckých databází.

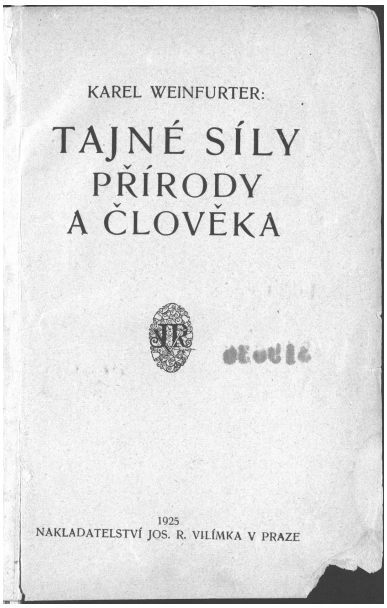
Bibliografický popis slouží jako nástroj k poskytnutí dostatku údajů k tomu, aby byl kdokoli schopen nalézt dokument, z něhož autor čerpal a pochopil návaznost na předchozí poznání a získal širší kontext dané tématiky [31]. Standardizovaná forma bibliografických údajů zajišťuje konzistenci napříč různými systémy a citačními styly (např. APA, MLA, ISO 690, Chicago). I přesto se konkrétní podoba může lišit v závislosti na zvolené normě nebo typu dokumentu.

Mezi základní prvky bibliografického popisu patří především autor nebo autoři, tedy osoby odpovědné za vznik publikace. V některých případech může být uveden editor, který je zpravidla označen výrazem „upravil“. Dále jsou důležité vydavatelské údaje, které zahrnují místo vydání (obvykle město), název nakladatele (instituce nebo vydavatelství) a rok vydání. Výjimečně může být uveden i tiskař nebo místo tisku. Pokud je dokument součástí série, uvádí se rovněž název této série a případně její číslo. V případě, že nejde o první vydání, bývá specifikováno číslo vydání, například „2. přepracované vydání“.

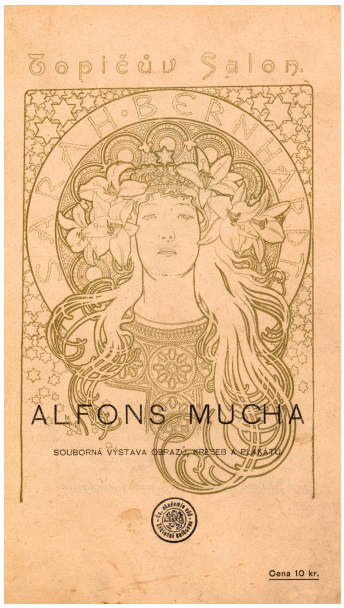
Na titulních stranách knih jsou tyto údaje často prezentovány ve volné a graficky rozmanité podobě. Nebývají explicitně označené názvem atributu (např. „autor“ nebo „vydavatel“), ale vycházejí z kontextu a typografického zvýraznění (např. větší písmo, tučný text). To může ztěžovat jejich automatické rozpoznání, a proto je analýza titulních stran výzvou pro systémy pro strojové zpracování dokumentů. Jak píše Svenonius: „vizuální pre-



(a) Český svět



(b) Tajné síly přírody a člověka



(c) Alfons Mucha

Obrázek 2.1: Příklady titulních stran knih.

zentace informací v bibliografických záznamech není vždy standardizována, což znesnadňuje automatizované zpracování“ [28].

Ve většině případů platí, že jméno autora bývá umístěno v horní nebo střední části titulní strany a je typograficky zdůrazněno. Název knihy zaujímá nejvýraznější pozici, často ve středu stránky. Vydavatelské údaje se obvykle nacházejí ve spodní části stránky a mohou obsahovat město vydání, název nakladatele a rok. Není to však pravidlem, neexistuje přesně stanovený formát titulních stran. Tato různorodost ztěžuje klasifikaci jednotlivých údajů a vyžaduje přístupy kombinující vizuální informace (rozložení stránky) s textovým obsahem.

Příklady titulní strany knihy. Pro ilustraci jsou na obrázcích 2.1 příklady titulních stran knih, které představují různorodé rozmístění bibliografických údajů. Na uvedených příkladech lze identifikovat některé bibliografické údaje. Ukázka 2.1b prezentuje nejběžnější rozmístění bibliografických údajů, které je zmíněno výše a zde je demonstrováno. Název publikace, zvýrazněný největší velikostí písma, obsahuje každá strana, i když na ukázce 2.1c je součástí obrázku, ze kterého není jasně čitelný a současně se jedná o jméno. Také obsahuje nakladatele v horní části písmem, které je hůře čitelné, hlavně pro digitalní zpracovnání. První ukázka 2.1a obsahuje již jen podtitulek hned pod názvem a ostatní text není součástí bibliografických údajů.

Z hlediska strojového zpracování je třeba tyto informace správně rozpoznat nejen na základě textu, ale i jejich pozice na stránce a vizuální prezentace. Například autor nemusí být vždy výslovně označen jako „autor“, a jeho jméno se může nacházet na různých místech v závislosti na stylistice vydavatelství.

2.2 Přístupy k automatické extrakci bibliografických údajů z dokumentů

Automatická extrakce bibliografických údajů z dokumentů představuje důležitý krok při digitalizaci knih a vědeckých textů. Titulní strany knih se často výrazně liší svým rozvržením a formátem, proto je nezbytné zvolit vhodnou metodu, která dokáže tyto rozdíly efektivně zpracovat. Existují různé přístupy, které lze obecně rozdělit do dvou kategorií: pravidlové metody a metody strojového učení.

Pravidlové metody. Pravidlové metody, označované také jako *rule-based přístupy*, jsou založeny na pevně definovaných lingvistických nebo strukturálních pravidlech, jako jsou například regulární výrazy. Tato pravidla slouží k identifikaci a extrakci klíčových údajů – typicky autora, názvu, vydavatele nebo roku vydání. Výhodou tohoto přístupu je jeho deterministické chování: při stejném vstupu generuje vždy stejný výstup, což je výhodné zejména v aplikacích, kde je vyžadována konzistence a opakovatelnost [5].

Jak uvádí Jurafsky a Martin, „*pravidlové systémy poskytují transparentní a interpretovatelný rámec, což je jejich hlavní výhoda ve srovnání s některými černými skříňkami moderních modelů*“ [12].

Typický postup pravidlového přístupu v oblasti zpracování přirozeného jazyka (*Natural Language Processing, NLP*) se skládá ze čtyř hlavních kroků. Nejprve dochází k vytvoření pravidel, kdy se definují gramatické, syntaktické či sémantické vzory, případně regulární výrazy, které popisují hledané struktury v textu. Následně se tato pravidla aplikují na vstupní text s cílem rozpoznat specifické prvky, jako jsou například jméno autora nebo rok vydání. V dalším kroku probíhá zpracování výstupu, tedy extrakce požadovaných informací nebo jejich další využití na základě identifikovaných struktur. Nakonec následuje zdokonalování pravidel, kdy jsou pravidla průběžně aktualizována na základě zpětné vazby a odhalených chyb během použití [5].

Pravidlové přístupy se často uplatňují v případech, kde je datová struktura dobře definovaná a málo proměnlivá [21]. Přesto zůstává nevýhodou jejich nízká flexibilita – každá odchylka od očekávaného formátu dokumentu často vyžaduje ruční úpravu nebo doplnění pravidel. Vývoj takového systému je časově náročný, jelikož vyžaduje detailní znalost struktury dokumentů i jazykových konvencí. Navíc pravidla vytvořená pro jeden typ dokumentů (např. knihy) obvykle nelze bez úprav použít na jiný typ (např. vědecké články nebo časopisy), což výrazně omezuje schopnost modelu generalizace, přesto se špatně přenášejí na jiný typ dokumentu. Nicméně zůstávají pravidlové přístupy vhodným řešením pro úlohy s velmi konzistentní strukturou dat, případně jako doplněk modernějších metod, kde mohou sloužit např. pro předzpracování textu nebo validaci výstupů modelů strojového učení [5].

Příkladem pravidlových metod pro extrakci bibliografických údajů je práce Ojokoh [23], která představuje systém založený na regulárních výrazech pro identifikaci a extrakci metadat z heterogenních referencí v různých citačních stylech. Systém byl navržen tak, aby dokázal zpracovat knihy, články i konferenční příspěvky, a dosáhl vysoké přesnosti i při práci s různorodými formáty. Podobně Makhija a Ahuja [20] navrhli pravidlový extraktor textových údajů z bibliografických databází, který využívá podmíněné výrazy a syntaktické vzory ke spolehlivé extrakci údajů jako autor, název či vydavatel. Oba přístupy ukazují, že při vhodném návrhu mohou pravidlové systémy dosahovat konkurenceschopných výsledků, zejména v prostředích s dobře definovanými strukturami vstupních dat.

Modely strojového a hlubokého učení Strojové učení je oblast umělé inteligence zabývající se návrhem algoritmů, které se dokáží automaticky zlepšovat na základě dat bez nutnosti explicitního programování všech pravidel. V oblasti automatické extrakce bibliografických údajů se široce uplatňují metody strojového a hlubokého učení. Tyto přístupy se vyznačují schopností přizpůsobit se různorodým strukturám dokumentů a efektivně pracovat s nestrukturovanými daty, jako jsou texty nebo skeny titulních stran.

Neuronové sítě představují speciální třídu modelů inspirovanou strukturou biologického mozku; skládají se z vrstev umělých neuronů propojených váhami [6]. Proces učení neuronových sítí je založen na metodě zpětného šíření chyby (*backpropagation*), která slouží k efektivní aktualizaci vah v síti podle velikosti chyby mezi skutečným a očekávaným výstupem. Tato chyba se šíří zpět od výstupní vrstvy až k vrstvě vstupní, přičemž se na základě řetězového pravidla počítají parciální derivace (gradienty) ztrátové funkce vzhledem ke každému parametru sítě. Optimalizace těchto vah pak probíhá pomocí gradient descent metod, z nichž nejčastěji používané jsou stochastic gradient descent (SGD) a jeho pokročilejší varianty jako Adam (Adaptive Moment Estimation). Tyto algoritmy upravují váhy tak, aby minimalizovaly ztrátovou funkci, a tím síť postupně zlepšovala své výstupy na základě trénovacích dat. Výběr optimalizační metody, počáteční nastavení vah, velikost batchů i rychlost učení (*learning rate*) hrají klíčovou roli v efektivitě a stabilitě trénovacího procesu neuronových modelů [15].

Stochastic Gradient Descent (SGD) je základní a široce používaná metoda optimalizace, která aktualizuje váhy sítě na základě jednotlivých nebo malých skupin vzorků (mini-batchů). Její hlavní výhodou je jednoduchost a nízká paměťová náročnost, díky čemuž je vhodná pro velmi rozsáhlé modely. Nevýhodou je ale pomalá konvergence a náchylnost k uvíznutí v lokálních minimech nebo oscilaci v údolích ztrátové funkce. Naproti tomu Adam kombinuje výhody RMSProp a momentum metody tím, že adaptivně upravuje velikost kroků u jednotlivých parametrů na základě prvního a druhého momentu gradientů. Díky tomu dosahuje rychlejší a stabilnější konvergence zejména u složitějších modelů a dat. Nevýhodou Adamu může být větší paměťová náročnost a v některých případech horší generalizace výsledků ve srovnání s SGD [15, 34].

Z klasických metod strojového učení se využívají například rozhodovací stromy, náhodné lesy nebo SVM (Support Vector Machines). Pro úlohy označování posloupnosti (sequence labeling), kdy je třeba přiřadit každému prvku textu specifickou třídu (např. autor, název, rok vydání), se často uplatňují modely jako CRF (Conditional Random Fields) [18]. Vhodným doplňkem jsou i modely LSTM (Long Short-Term Memory) [8], které dokáží uchovávat kontext v delších textových sekvencích a zohlednit pořadí slov.

V praxi lze tyto modely trénovat na reálných datech – například na datasetu skenovaných titulních stran – kde se učí rozpoznávat a lokalizovat názvy knih, jména autorů či vydavatelské údaje, a to i v případě netradičního rozvržení nebo vícejazyčných dokumentů.

Metody hlubokého učení (deep learning) rozšiřují možnosti strojového učení o automatické učení reprezentací a lepší škálovatelnost při zpracování komplexních vstupů. Zvláště efektivní jsou při kombinaci optického rozpoznávání znaků (OCR) a technik zpracování přirozeného jazyka (NLP), které umožňují detailní analýzu textu i jeho vizuálního uspořádání [36].

Pro analýzu obrazových aspektů dokumentů, jako je detekce textových bloků či struktur, se využívají konvoluční neuronové sítě (CNN). Mezi osvědčené architektury patří např. YOLO (You Only Look Once) [25], vhodná pro rychlou a přesnou detekci objektů. Textová složka dokumentu bývá zpracována pomocí rekurentních sítí (např. LSTM) nebo moder-

ních transformerových modelů [32], které využívají self-attention mechanismus a umožňují efektivní zohlednění vztahů mezi jednotlivými částmi textu.

Zvláštní pozornost si zaslouží modely jako LayoutLM [36], které rozšiřují architekturu BERT [4] o informace o prostorovém rozložení textu na stránce. LayoutLM umožňuje přesnou extrakci údajů i z dokumentů s nepravidelným formátováním, což je klíčové při práci s titulními stranami knih a článků. Dále lze uvést model Donut [14], jenž umožňuje přímé zpracování dokumentových obrázků bez použití OCR, nebo DocFormer [22], který integruje textové, vizuální i prostorové informace v rámci jediné architektury. Tyto modely vykazují vysokou přesnost zejména při zpracování nestrukturovaných a vizuálně složitých dokumentů. Mezi hlavní výhody těchto metod patří vyšší přesnost, adaptabilita na reálná data a možnost zpracovávat i vizuálně členité dokumenty. Nevýhodou zůstává vyšší výpočetní náročnost, potřeba kvalitně anotovaných trénovacích dat a nižší míra interpretovatelnosti výsledků [27].

Pro ilustraci úspěšnosti těchto přístupů lze uvést dva příklady z nedávné literatury. Studie Bhardwaj [1] představuje přístup k extrakci bibliografických údajů z obrazových dokumentů pomocí hlubokého učení. Autoři navrhli systém DeepBIBX, který využívá plně konvoluční neuronové sítě (Fully Convolutional Networks, FCN) a transfer learning k semantické segmentaci jednotlivých citačních řetězců přímo z naskenovaných dokumentů. Na rozdíl od tradičních metod, které se spoléhají na převod obrazu na text a následné zpracování, DeepBIBX pracuje přímo s vizuálními informacemi, což umožňuje nezávislost na jazyce dokumentu. Systém byl testován na datasetu obsahujícím 286 dokumentů s celkem 5090 bibliografickými referencemi a dosáhl přesnosti 84,9 %, čímž překonal dřívější metody jako ParsCit [3], který dosáhl přesnosti 71,7 %. V úlohách klasifikace pixelů dosáhl DeepBIBX precision 96,2 % a recall 94,4 %, což potvrzuje jeho efektivitu při zpracování různorodých dokumentů [1].

Druhým příkladem je GROBID (GeneRation Of BIbliographic Data), systém pro extrakci strukturovaných bibliografických údajů z vědeckých dokumentů ve formátu PDF. Využívá modely strojového učení, konkrétně Conditional Random Fields (CRF), k identifikaci a segmentaci klíčových oblastí textu, jako jsou názvy článků, jména autorů, abstrakty nebo bibliografické odkazy. GROBID nabízí několik modulů, včetně extrakce hlavičky článku, zpracování citací a rozpoznávání kontextu, ve kterém se citace vyskytují. Výsledky jsou generovány ve formátu TEI XML, což umožňuje snadnou integraci do dalších nástrojů pro zpracování vědeckých textů. Systém je široce využíván v akademických digitálních knihovnách i ve výzkumných institucích díky své přesnosti [19].

Významnou srovnávací studii provedli Tkaczyk et al. [29], kteří analyzovali výkonnost deseti nástrojů pro extrakci bibliografických údajů z referenčních řetězců. Ve své práci porovnali metody založené na pravidlech a na strojovém učení jak v jejich výchozím nastavení (out-of-the-box), tak po přizpůsobení konkrétním datům (retrained). Výsledky ukázaly, že nástroje využívající strojové učení, jako je například GROBID, nástroj založený na strojovém učení a CRF, dosahují vyššího recallu (0,66 oproti 0,22) než pravidlové systémy, a tím lépe zachycují potřebné informace. GROBID ve výchozím režimu dosáhl nejvyššího F1 skóre 0,89, zatímco po přeučení na specifických datech se jeho výkon zvýšil až na 0,92. Podobný efekt byl pozorován i u dalších nástrojů jako CERMINE [30] a ParsCit. Tato práce tak potvrzuje, že přizpůsobení modelů konkrétním datovým sadám výrazně zvyšuje kvalitu extrakce a zároveň dokládá efektivitu metod strojového učení oproti tradičním přístupům.

Z výše uvedeného vyplývá, že modely hlubokého učení, zejména ty využívající prostorové informace, výrazně posouvají možnosti extrakce bibliografických údajů.

2.3 Optické rozpoznávání znaků (OCR)

Technologie optického rozpoznávání znaků, známá pod zkratkou OCR (Optical Character Recognition), slouží k digitalizaci tištěného textu. Pomocí skeneru či jiného zobrazovacího zařízení je vytvořen obraz dokumentu, který je následně zpracován softwarem rozpoznávajícím jednotlivé znaky a slova. Výstupem je upravitelný digitální text, který lze dále editovat, vyhledávat či archivovat [33].

OCR systémy mohou pracovat buď na základě předem naučených vzorů znaků, nebo využívají metody strojového učení, díky nimž se dokáží přizpůsobit specifickému písmu nebo jazyku. Přesnost rozpoznání závisí na kvalitě předlohy – v případě poškozených, vybledlých nebo nekvalitně naskenovaných dokumentů je často nutná ruční korektura výsledného textu. OCR technologie je využitelná pro většinu standardně tištěných výstupů, jako jsou dokumenty z laserových, či inkoustových tiskáren a klasických tiskových strojů. Naopak u výrazně znehodnocených výtisků, jako jsou výstupy ze starších jehličkových tiskáren nebo dokumenty se slitými znaky, může být efektivnější ruční přepis [33].

Moderní OCR systémy, jako je například Tesseract [7], využívají hluboké neuronové sítě k rozpoznávání textu v různých jazycích a stylech písma, a jsou schopny zvládat i složité případy, jako je zakřivený text nebo text na šumu či složitém pozadí. OCR se uplatňuje v celé řadě oblastí, od archivnictví přes automatizaci administrativy až po rozšířenou realitu.

Optické rozpoznávání znaků (OCR) nachází uplatnění například při digitalizaci knih a historických dokumentů, dále při extrakci informací z různých tištěných formulářů, faktur či vizitek a rovněž v automatizované analýze textu v obrazech, například při snímání a rozpoznávání dopravních značek či nápisů na tabulích.

Tesseract OCR. Tesseract OCR je jeden z nejpoužívanějších volně dostupných systémů pro optické rozpoznávání znaků (OCR). Původně byl vyvinut společností Hewlett-Packard v 80. letech 20. století, ale od roku 2006 je spravován Googlem. Od verze 4.0 Tesseract využívá neuronové sítě (LSTM – Long Short-Term Memory), což výrazně zlepšilo jeho přesnost, zejména při rozpoznávání textu v komplikovaných podmínkách (např. skloněné texty, různé písmo nebo špatná kvalita skenů) [7].

Tesseract OCR pracuje v několika na sebe navazujících krocích. Nejdříve se obraz předzpracuje (konverze na škálu šedi, odstranění šumu), pak se provede binarizace (thresholding) pro oddělení znaků od pozadí. Následuje analýza rozvržení stránky, kdy se detekují bloky textu, řádky a jednotlivé komponenty, které se pak spojí do tvarů odpovídajících znakům. Ve verzi 4 a novější se pro rozpoznávání používá LSTM síť, pro vyšší přesnost se pak aplikuje jazyková korektura pomocí slovníků a nakonec se výsledek exportuje do požadovaného formátu [7].

Tesseract podporuje UTF-8 kódování a více než 100 jazyků. Na vstupu podporuje širokou škálu obrázkových formátů, jako například PNG, JPEG, TIFF a další. Výstupy mohou být ve formátu prostého textu, hOCR (HTML formát), PDF a dalších formátech. Tesseract je často využíván pro digitalizaci knih, faktur, historických dokumentů a pro automatické zpracování velkých objemů textových dat.

PERO-OCR. PERO-OCR je specializovaný nástroj vyvinutý v rámci projektu PERO na Vysokém učení technickém v Brně [16]. Tento systém umožňuje automatický přepis několika typů tištěných i ručně psaných dokumentů, což je zvláště užitečné pro digitalizaci historických nebo špatně čitelných materiálů.

Nástroj je schopen rozpoznávat texty ve většině evropských jazyků, včetně latinky, a to i ve specifických historických písmenech, jako je fraktura (typické pro německé a české texty ve středověku). Kromě tištěného textu dokáže PERO přepisovat i ručně psané dokumenty, přičemž nejvyšší přesnost vykazuje při rozpoznávání češtiny.

Uživatelé si mohou zvolit různé formáty pro export výsledků přepisu, jako jsou ALTO, PAGE XML nebo prostý text, což usnadňuje další zpracování nebo analýzu textů.

2.4 Detekce bibliografických údajů pomocí modelu YOLO

YOLO (You Only Look Once) je moderní metoda pro detekci objektů v obrazech a videích založená na hlubokém učení. Její hlavní výhodou je rychlost a přesnost, díky čemuž je vhodná pro aplikace v reálném čase, jako jsou autonomní vozidla, bezpečnostní systémy nebo analýza videa. Na rozdíl od tradičních přístupů, které analyzují obraz po částech, YOLO dokáže detekovat objekty v jednom průchodu neuronovou sítí, což výrazně zrychluje zpracování.

YOLO pro detekci pozice bibliografických údajů. Model YOLO patří mezi jednofázové detektory objektů, které se vyznačují vysokou rychlostí a přesností detekce. Základní princip spočívá v rozdělení vstupního obrazu na mřížku $S \times S$, kde každá buňka predikuje B ohraničujících obdélníků (bounding boxes), konfidenční skóre a pravděpodobnosti příslušnosti k jednotlivým třídám [25]. Výstup je organizován jako tenzor rozměru $S \times S \times (B \cdot 5 + C)$, kde pět čísel $(x, y, w, h, \textit{confidence})$ popisuje každý box: střed (x, y) relativně k buňce, šířku w, výšku h relativně k celému obrazu a *confidence* jako míru spolehlivosti.

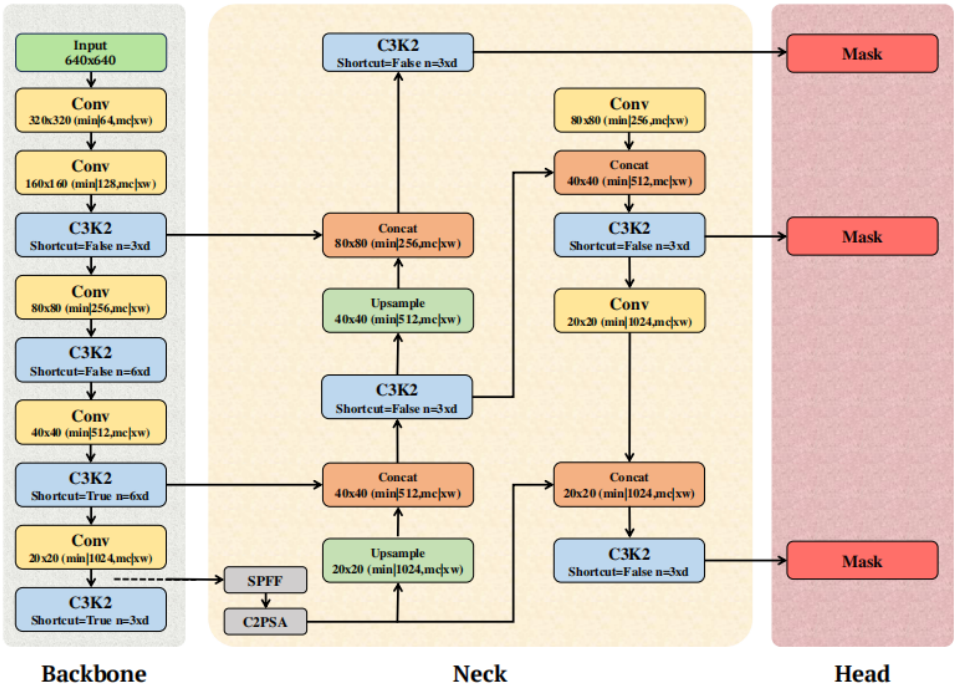
Tento výstup slouží k lokalizaci oblastí zájmu v obraze – v tomto případě jde o oblasti s bibliografickými údaji (např. název publikace, autor, rok vydání) na skenech titulních stran. Model samotný text nerozpoznává, pouze označuje pozice relevantních prvků. Tyto oblasti jsou následně předány do modulu OCR, čímž se výrazně snižuje výpočetní zátěž a zvyšuje přesnost rozpoznání, protože OCR zpracovává jen cílené části dokumentu.

Architektura YOLOv11. Nejnovější verze modelu, *YOLOv11*, přináší řadu architektonických vylepšení zaměřených na zvýšení efektivity a přesnosti detekce. Klíčovými komponentami této verze jsou blok `C3k2`, který využívá dvě menší konvoluce namísto jedné větší – čímž snižuje výpočetní náročnost – a modul `C2PSA` (Cross-Stage Parallel Spatial Attention), který síti umožňuje lépe se soustředit na klíčové oblasti v obraze, např. drobné nebo částečně zakryté objekty. Dále se využívá blok `CBS` (Convolution–BatchNorm–SiLU), který stabilizuje učení díky vhodně zvolené aktivační funkci [13].

Díky těmto komponentám dosahuje YOLOv11 vyšší průměrné přesnosti (mAP) než předchozí verze, a zároveň redukuje počet parametrů přibližně o 22 % oproti YOLOv8 [11]. Výsledná architektura tak zůstává efektivní i při nasazení na výpočetně omezeném hardwaru.

Architektura YOLOv11 je – stejně jako u předchozích generací – rozdělena do tří hlavních částí: backbone, neck a head. **Backbone** slouží k extrakci nízko až středně-úrovňových vizuálních rysů z obrazu. Využívá kombinaci bloků `CBS` a `C3k2`, které zajišťují efektivní výpočty a kvalitní reprezentaci vstupních dat. **Neck** propojuje rysy z různých hloubek sítě a využívá architektury jako `FPN` (Feature Pyramid Network) a `PAN` (Path Aggregation Network) pro víceúrovňové zpracování. Součástí necku je i `C2PSA`, který selektivně zdůrazňuje

důležité oblasti obrazu na základě prostorové pozornosti. **Head** provádí finální detekci – predikuje souřadnice bounding boxů, jejich třídy a konfidenční skóre. Využívá přístup bez anchor boxů (anchor-free), kde se výstupy generují pro více měřítek výstupní mřížky (např. 80×80, 40×40, 20×20), což zajišťuje detekci objektů různých velikostí [13]. Architektura je znázorněna na obrázku 2.2.



Obrázek 2.2: Architektura modelu YOLOv11 [13].

2.5 LayoutLM a jeho použití při extrakci údajů z dokumentů

Model **LayoutLM** je architektura založená na transformeru (konkrétně na modelu BERT) rozšířená o zpracování dvourozměrných informací z dokumentu. Kromě standardních textových tokenů a odpovídajících 1D pozic přidává také 2D poziční embedovací vrstvu, která kóduje souřadnice okrajů ohraničujících rámečků slov na stránce. Každé slovo je tedy reprezentováno nejen textovým embeddingem, ale i embeddingem jeho čtyř souřadnic (x0, y0, x1, y1) ve standardizovaném souřadnicovém systému dokumentu. Dále model obsahuje obrazové embedování – pro každý token se pomocí předtrénovaného detektoru objektů (Faster R-CNN) vygeneruje vektor obrazových rysů z oblasti dokumentu odpovídající danému slovu. Tak lze zachytit vizuální kontext (například font, barvu či pozadí) každého slova. Tři modality (text, poloha, obraz) jsou následně společně zpracovány vrstvami transformeru, který modeluje jejich vzájemné interakce na úrovni dokumentu [36].

Po předzpracování obrazu dokumentu se LayoutLM jemně doladí na úlohy informační extrakce. Vstupem mu je posloupnost slovních tokenů získaných z OCR spolu s jejich normalizovanými souřadnicemi a případně s původním obrazem dokumentu (nebo rysy z něj).

Výstupem modelu bývají anotace třídy přiřazené jednotlivým tokenům. Typickým schématem pro úlohy formuláře či účtenky je označování entit posloupnosti podle BIO/BIESO konvencí (např. „B-titulek, I-titulek, E-titulek, S-datum, O-ostatní“), čímž model detekuje klíčové položky (pole) dokumentu [36]. Při doladění se používaly standardní datasety, například FUNSD pro porozumění formulářům (199 ručně označených skenovaných formulářů) [10] nebo CORD pro zpracování pokladních účtenek (tisíce indonéských účtenek s anotacemi textu a vícero úrovněmi textových tagů) [24]. Na těchto datech se trénuje model tak, že se učí správně přiřazovat každému slovu jeho roli (např. název firmy, suma, datum apod.), což vede k uspořádanému výstupu vhodnému pro výstavbu datových struktur či klíč-hodnota párů.

LayoutLMv2 dále rozšiřuje původní architekturu o plně dvouproudý multimodální transformer a nové úlohy předtrénování. Stejně jako LayoutLM pracuje s textem, rozložením a obrazem dokumentu, ale vrástností propojuje tyto modality hlouběji. K tomu využívá především dva nové úkoly orientované na *cross-modal alignment*: Text-Image alignment a Text-Image matching. V úloze Text-Image Alignment jsou náhodně vybrané řádky textu v dokumentu zakryty na obrazové úrovni a model je trénován tak, aby odhadl zakrytý obsah na základě ostatního textu a layoutu. V úloze Text-Image Matching se model učí rozpoznat, zda daný obraz dokumentu a textový výstup OCR pochází ze stejného dokumentu. V modelu LayoutLMv2 je tato úloha formulována jako binární klasifikační problém. Model se učí předpovídat, zda daný text a obraz patří k sobě (pozitivní pár) nebo ne (negativní pár). Tato úloha využívá klasifikační ztrátovou funkci (např. *binary cross-entropy*). Díky těmto vylepšením LayoutLMv2 výrazně překonává původní model v úlohách form understanding a receipt understanding a dosahuje lepších výsledků na datech FUNSD i CORD [9].

LayoutLMv3 prezentuje další krok, který zjednodušuje a sjednocuje tréninkové úlohy pro obě modality. Architektura zůstává multimodálním transformerem podobně jako u v2, avšak předtrénování nyní probíhá pomocí unifikovaného maskování textu i obrazu. Konkrétně se náhodně maskují nejen textové tokeny, ale i části obrazového vstupu (obrazové „tesserxy“) současně. Úroveň multimodality se dále podporuje novou úlohou *word-patch alignment*: pro každý textový token model předpovídá, zda jeho odpovídající obrazový patch je zamaskovaný či nikoliv. Tím se učení modelu zaměřuje na koordinaci textových a obrazových reprezentací. Jednoduchý sjednocený přístup umožňuje LayoutLMv3 být univerzálním modelem dokumentové AI pro jak textově orientované úlohy (form understanding, zpracování faktur, otázky v dokumentech atd.), tak pro úlohy čistě obrazové (klasifikace typu dokumentu, analýza rozvržení) [9]. Architektura nejnovějšího modelu LayoutLMv3 je vyobrazena na obrázku 2.3.

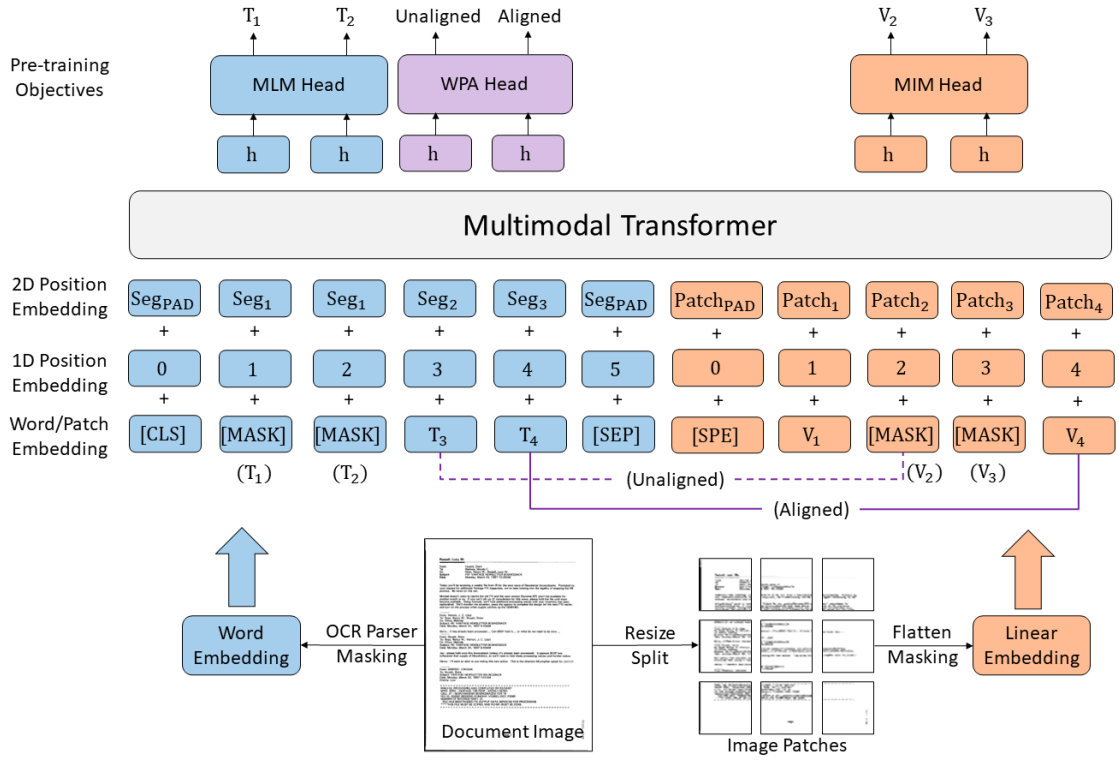
Všechny tyto verze modelu tedy pracují s textem, jeho pozicí a obrazem dokumentu jako společným vstupem a na výstup dávají strukturované informace (např. anotace třídy u slov).

Implementace modelů LayoutLM se nejčastěji realizuje pomocí frameworku *Transformers* od společnosti Hugging Face [35], který poskytuje rozhraní pro práci s předtrénovanými variantami modelu `LayoutLM`, `LayoutLMv2` a `LayoutLMv3` [35]. Pro správné fungování modelu je nutné připravit tři druhy vstupních dat: samotné textové tokeny (dále zpracované pomocí BERT tokenizéru), souřadnice ohraničujících boxů (bounding boxy), které popisují pozici textu na stránce a jsou získávány pomocí OCR nástroje, a také obrazový vstup ve formě rastrového obrázku celé stránky (např. ve formátu PNG nebo JPEG).

Trénování vlastního modelu nebo jeho doladění (*fine-tuning*) probíhá obdobně jako u jiných transformerových architektur. Model lze učit na vlastních anotovaných datech, která jsou typicky ve formátu JSON nebo XML a obsahují nejen text, ale i informace o pozi-

cích a kategoriích jednotlivých entit. Tento proces umožňuje přizpůsobit model specifickým požadavkům dané domény (např. faktury, formuláře, smlouvy apod.).

Model LayoutLM představuje dobrý nástroj pro porozumění strukturovaným dokumentům díky schopnosti integrovat textová, vizuální a prostorová data. Nabízí vysokou flexibilitu a přesnost v různorodých úlohách, zejména tam, kde klasické jazykové modely selhávají kvůli absenci prostorového kontextu. Na druhé straně je třeba počítat s vyšší výpočetní náročností, závislostí na kvalitním OCR a náročnější integrací do reálných systémů. V dobře připraveném prostředí však poskytuje výrazně lepší výsledky než čistě textové nebo vizuální přístupy.



Obrázek 2.3: Architektura modelu LayoutLMv3 [9].

Kapitola 3

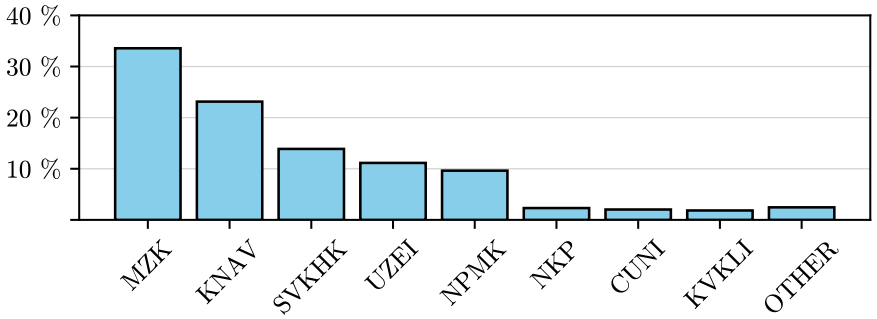
Použité technologie a metody

V této kapitole jsou popsány technologie a postupy použité při tvorbě této bakalářské práce a řešení úlohy detekce a extrakce bibliografických údajů z dokumentů. Nejprve je představen použitý dataset a způsob jeho anotace pomocí nástroje Label Studio. Následně jsou detailně rozebrány dva klíčové modely: YOLO pipeline pro vizuální detekci bibliografických údajů na stránce a LayoutLM pipeline pro klasifikaci textových informací s využitím jazykových, prostorových a vizuálních vstupů. Při formulaci technických částí textu, jazykových úpravách a překladech byl využit nástroj ChatGPT od společnosti OpenAI. Obsah práce však zůstává výsledkem mé samostatné odborné činnosti.

3.1 Dataset titulních stran

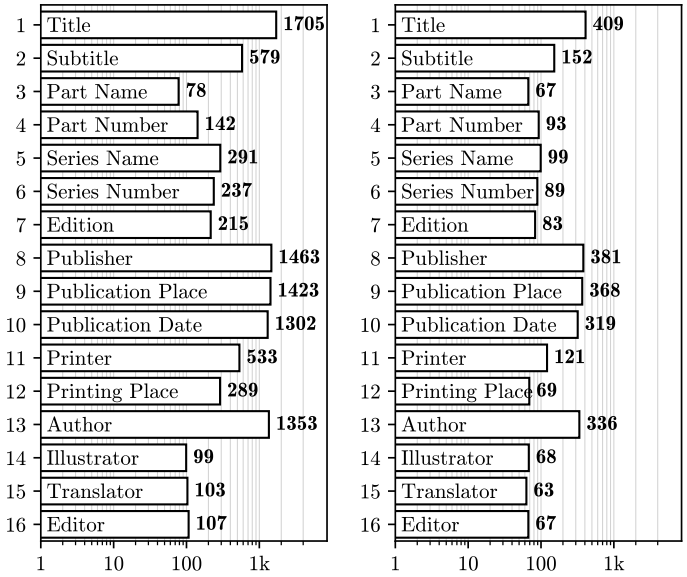
Dataset **BiblioPage** obsahuje celkem 2 118 skenů anotovaných titulních stran, pocházejících z fondů 14 českých knihoven (viz obrázek 3.1) a pokrývajících vydavatelský materiál od roku 1485 až po 21. století. Celkem bylo v datasetu anotováných 12 703 atributů, které spadají do 16 kategorií pokrývajících různé typy bibliografických informací. Mezi tyto atributy patří: název díla, informace o vydání a publikaci, údaje o vydavateli a tisku, informace o přispěvatelích. Kromě samotných hodnot atributů dataset obsahuje i geometrickou polohu těchto prvků na stránkách [17].

Každá anotovaná stránka obsahuje výskyty bibliografických prvků ze 16 kategorií, včetně titulku, údajů o vydání, přispěvatelích a místech či entitách spojených s výrobou a distri-



Obrázek 3.1: Zastoupení knihoven [17]

.



(a) Trénovací část datasetu (b) Validační část datasetu

Obrázek 3.2: Zastoupení jednotlivých entit v datasetu [17]

.

bucí publikace. Celkově bylo zaznamenáno 12 703 anotací, každá doplněná o přesnou polohu v obraze (bounding box).

Stránky lze podle vizuální a typografické struktury rozdělit do tří hlavních typů: jednoduché titulní strany (cca 70 % datasetu) s přehlednou strukturou a základními údaji, složitější víceřádkové nebo typograficky zvýrazněné stránky (25 %) s např. dílčími názvy nebo různě uvedenými přispěvatelskými rolemi, a vizuálně netypické případy (5 %) zahrnující historické nebo kaligrafické tisky, obtížně zpracovatelné běžnými OCR nástroji [17].

Vzhledem k nerovnoměrnému výskytu některých anotací nebylo možné použít čistě náhodný výběr. Byl proto aplikován poloautomatický postup výběru, který kombinoval počíteční náhodný výběr 200 stran a následné doplňování do validační části datasetu, dokud každá z 16 kategorií nebyla zastoupena alespoň 50krát. Tento přístup vedl k rozdělení 1 708 stránek do trénovací sady (9 919 anotací) a 410 stránek do testovací sady (2 784 anotací). Rozložení jednotlivých entit v obou sadách ukazuje obrázek 3.2, osa x je zobrazena v logaritmickém měřítku [17].

Vstupní data tedy tvoří titulní strany z digitálních knihovních sbírek zveřejněných v rámci projektu BiblioPage [17], přičemž součástí této práce bylo provedení poloautomatického zarovnání OCR dat na obrazové vstupy. Anotace byly následně validovány a doplněny ručně tam, kde OCR nebo strukturované metadata selhávala.

Anotace v Label Studiu. Anotace titulních stran probíhala pomocí nástroje Label Studio, který umožňuje vizuální označování entit přímo v obraze. Pro efektivní přípravu anotací byl v rámci této práce vyvinut **poloautomatický skript v jazyce Python**, který kombinoval výstupy optického rozpoznávání znaků (OCR) z nástroje PERO s bibliografickými metadaty ve formátu MODS.

Každá stránka byla analyzována skriptem, který pro každý údaj z MODS záznamu (např. název díla, autor, místo vydání) hledal odpovídající text v OCR výstupech. Nejprve

(a) Automatické bounding boxy (b) Výsledný stav po kontrole

Obrázek 3.3: Ukázka poloautomatické anotace v Label Studiu.

se provedlo normalizované porovnání každého páru textů pomocí knihovny `fuzzywuzzy` [2], konkrétně metriky `partial_ratio` a `token_set_ratio`. Funkce `partial_ratio` určuje míru podobnosti mezi dvěma řetězci tak, že hledá nejlepší možnou shodu podřetězce kratšího řetězce v řetězci delším (vhodné pro případy, kdy hledaný výraz tvoří jen část OCR výstupu, např. „napsal Josef Čapek“). Funkce `token_set_ratio` naopak provádí srovnání na úrovni množin slov – ignoruje pořadí slov a je robustní vůči jejich přeházení nebo duplicitám, čímž umožňuje identifikovat shodu i při změně struktury textu. Tato funkce byla využita zejména v hledání jmen autorů, která se většinou v MODS datech vyskytovala v opačném pořadí. Pokud skóre přesáhlo předem definovaný práh (90 %), považovala se shoda za úspěšnou.

V případě, že bylo nalezeno dostatečně přesné shodné OCR pole, byly souřadnice bounding boxu vypočteny buď z polygonu OCR řádku, nebo v případě podřetězců byl využit výstup OCR enginu PERO – tzv. logity, tedy pravděpodobnostní mapa výskytu znaků v obraze. Tyto logity zachycují, na kterých pozicích v obrázku model detekuje jednotlivé znaky, a umožňují tak odhadnout jejich konkrétní umístění nezávisle na polygonu celého řádku. Pomocí funkce `align_text_to_image` došlo k mapování jednotlivých znaků na obrazovou osu X. Při nalezení hledaného textu v OCR řádku (např. v rámci fuzzy shody) skript spočítal začátek a konec výskytu na základě indexu znaků v řetězci a jejich odpovídající pozice v logitech. Svislé souřadnice byly dopočítány z baseline řádku a jeho typografických výšek.

Po vytvoření návrhů bounding boxů následoval postprocessing, který sloužil ke zpřesnění anotací. V první fázi byly jednotlivé boxy slučovány na základě překryvu a blízkosti – například u titulku či podtitulku byly spojeny vícenásobné boxy, pokud se nacházely těsně pod sebou a měly odpovídající kategorii. Druhý krok postprocessingu se zaměřoval na filtro-

vání nadbytečných anotací – pro atributy, u nichž se očekával pouze jeden výskyt (např. číslo série nebo místo vydání), byl ponechán pouze jeden box, obvykle ten nejvýše (či nejníže) umístěný na stránce dle významu atributu. U vícenásobně se vyskytujících kategorií (např. autor nebo ilustrátor) byly ponechány všechny relevantní boxy. Tato kombinace slučování a výběru významných anotací přispěla ke zvýšení kvality výsledného poloautomatického zarovnání.

V případech, kdy se OCR a MODS záznamy neshodovaly dostatečně přesně, nebo když OCR text chyběl (např. kvůli špatné kvalitě nebo dekorativní typografii), byl atribut vizuálně zvýrazněn pomocí speciálního pomocného boxu v levém horním rohu stránky. Ten zobrazoval text atributu převzatý z MODS jako nápovědu pro anotátora, který mohl údaj případně označit ručně. Text atriburu se zobrazoval i u zarovnaných údajů pro kontrolu správnosti zarovnání.

Jak toto poloautomatické zarovnání vypadá na konkrétní straně, je znázorněno na obrázku 3.3a. Díky tomuto přístupu, kombinujícímu fuzzy matching, zarovnání znaků a vizuální nápovědu, bylo možné automaticky předpřipravit většinu anotací a výrazně tak zkrátit čas potřebný pro ruční práci. Finální bounding boxy byly vždy ručně zkontrolovány anotátorem a výsledná anotace po kontole je k vidění na obrázku 3.3b.

3.2 Model YOLOv11 a celá pipeline

Cílem je ze skenovaných obrázků titulních stran knih získat strukturovaný JSON obsahující klíčové bibliografické prvky (titulek, podtitulek, autor, vydavatel, rok vydání, místo vydání,...) spolu s jejich rozpoznaným textem a odhadem spolehlivosti. Vstupem pro tuto pipeline je vždy dvojice souborů: rastrový snímek titulní strany a k němu odpovídající textový soubor s anotacemi ve formátu YOLO (každý řádek obsahuje `class_id x_center y_center width height` s relativními souřadnicemi vůči rozměrům obrázku).

Nejprve je obrázek zpracován detekčním modelem YOLOv11, který na základě předchozího tréninku na datasetu BiblioPage vrátí seznam bounding boxů s přiřazenými třídami a jejich pravděpodobnostmi. Pro každý nalezený box model poskytne třídu v rozsahu 0–15 a konfidenční skóre, přičemž jednotlivé číselné třídy se mapují na čitelná jména (např. 0 → Title, 1 → SubTitle apod.) podle souboru `data.yaml`.

Následuje příprava XML souboru ve formátu kompatibilním s OCR enginem PERO. Každá detekovaná instance je v tomto XML reprezentována elementem `<TextRegion>`, jehož atribut `type` odpovídá názvu detekované třídy a atribut `coords` obsahuje souřadnice čtyř rohů ohraničujícího polygonu (přepočtené z relativních hodnot YOLO na absolutní pixely). Díky tomu PERO přesně ví, ve kterých oblastech stránky má segmentovat a rozpoznávat text.

OCR krok probíhá tak, že engine PERO načte obrázek i připravené XML a pro každý `TextRegion` provede segmentaci textu, dekódování znaků a přiřazení konfidenčního skóre ke každému rozpoznanému textovému bloku. Výsledkem je XML soubor obsahující pro každou oblast rozeznaný text i jeho skóre spolehlivosti.

V závěrečné fázi je XML soubor převeden do cílové JSON struktury 3.1 pomocí skriptu. Klíčová pole JSON (např. `title`, `subTitle`, `author`, `dateIssued`, `publisher`, `placeTerm`) odpovídají detekovaným třídám a uchovávají pole dvojic [text, confidence]. Nakonec je generovaný JSON validován vůči referenčním datům pomocí evaluačního skriptu [17], který provádí normalizaci diakritiky a interpunkce, případné pre- a post-processingové filtry a vyhodnocení přesnosti na úrovni detekovaných regionů i rozeznaného textu.

```
{
    "task_id": "1",
    "library_id": "0001.jpg",
    "title": [
        ["Model Selection and Simplification Using Lattices", 1.0 ]
    ],
    "seriesName": [
        ["Working Paper Series", 1.0 ]
    ],
    "seriesNumber": [
        ["164", 1.0 ]
    ],
    "author": [
        ["Jaromir Antoch", 1.0 ],
        ["Jan Hanousek", 1.0 ]
    ],
    "dateIssued": [
        ["December 2000", 1.0 ]
    ],
    "publisher": [
        ["CERGE-EI", 1.0 ]
    ],
    "placeTerm": [
        ["Prague", 1.0 ]
    ]
}
```


Výpis 3.1: Příklad výstupního JSON souboru.

3.3 Model LayoutLMv3 a jeho pipeline

Příprava správného formátu datasetu byla časově náročnější než u předchozího YOLOv11. Počáteční přehled o přípravě dat a zpřístupnění *fine-tuningu* byl získán z návodu[1].

Příprava vstupního datasetu. Jako vstupní data pro model LayoutLMv3 byly použity výstupní soubory ALTO XML generované OCR enginem PERO, neboť na daném datasetu byly výsledky segmentace slov přesnější než při použití alternativních řešení (Tesseract). Ze souborů ALTO XML byl pomocí vlastního skriptu extrahován každý rozpoznaný token spolu s jeho absolutním ohraničujícím boxem. Následně bylo každému tokenu přiřazeno odpovídající anotované oblasti (z textových .txt souborů ve formátu YOLO), přičemž kromě původního čísla kategorie (`class_id`) byl uložen také interní index oblasti (unikátní identifikátor anotovaného boxu). Tento přístup umožnil rozlišit situace, ve kterých se ve stejném obrázku vyskytuje více instancí jedné třídy (např. dva autoři), a zároveň zachovat přesnou vazbu mezi každým tokenem a konkrétním anotovaným regionem.

Po přiřazení tokenů k anotovaným regionům byly původní číselné kategorie převedeny na sekvenční BIO (Begin–Inside–Outside) značky pro účely token-level klasifikace v rámci LayoutLMv3[2]. Nejprve byla všechna slova seřazena podle jejich původního `class_id` vze-

[1] https://medium.com/@shivarama/layoutlmv3-from-zero-to-hero-part-2-d2659eaa7dee

[2] LayoutLMv3 používá NER – Named Entity Recognition, neboli názvy štítků ve formátu BIO nebo BIESO.

stupně. Kategorie 0 (Outside) byla označena jako `O`, zatímco pro ostatní třídy byl na základě pořadí tokenu v rámci daného regionu přiřazen prvnímu tokenu prefix `B-<Label>` a všem následujícím prefix `I-<Label>` (např. `id=1` → `B-Title`, `id=2` → `I-Title`; `id=3` → `B-SubTitle`, `id=4` → `I-SubTitle` apod.). V případech, kde jeden token spadal do více regionů (např. překrývající se boxy dvou autorů), byl token označen seznamem všech příslušných kategorií [`id1`, `id5`], čímž byla zachována informace o víceznačné příslušnosti. Pro konzistenci jsou u všech tokenů kategorie reprezentovány jako seznam seznamů BIO značek, což umožňuje unifikované zpracování při tvorbě vstupních vzorků pro model.

Vstupní formát pro jednu titulní stránku vypadal přibližně takto (zkráceno pro příklad):

```
{
    "id": 2,
    "file_name": "002.jpg",
    "tokens": ["Nakladem", "KAREL", "ADAMEK."],
    "bboxes": [[548,1440,558,1475],[497,1529,524,1556],
               [588,1641,670,1672]],
    "ner_tags": [[0],[7,11],[8,12]]
}
```

kde `ner-tag` 0 značí token bez entity (*Nákladem*). Dalším dvěma tokenům obsahujícím jméno byly přiděleny `ner-tag` hodnoty 7, 11 pro počáteční entity `B-author, B-publisher` a 8, 12 pro pokračování těchto entit (`I-author, I-publisher`).

Skript pro přípravu datasetu. Pro načtení a přípravu trénovacího a testovacího datasetu pro LayoutLMv3 byla využita třída `GeneratorBasedBuilder`, která byla následně odvozená a jejíž implementace byla převzata z repozitáře LayoutLMv3[3]. V metodě `_info` bylo definováno schéma příkladů obsahující pole `id`, `tokens`, `bboxes`, `ner_tags`, `image_path` a `image` (formát odpovídá očekávanému vstupu modelu).

Metoda `_split_generators` načítala seznam názvů souborů pro trénink a validaci ze souborů `trainM.txt` a `testM.txt` umístěných ve složce `layoutlmv3`. Každý řádek těchto txt souborů obsahoval slovník (ve formátu Python literal) s klíči `file_name`, `tokens`, `bboxes` a `ner_tags`.

Ve `_generate_examples` bylo nejprve načteno mapování číselných kategorií na textové anotace třídy z `class_list`. Pro každý řetězec v seznamu byl pomocí `ast.literal_eval` proveden převod na Python dict, ze kterého byly extrahovány cesty k obrázkům, seznam tokenů, normalizované bounding boxy a odpovídající BIO tagy. Obrázky byly načteny pomocí PIL, jejich rozměry byly získány funkcí `load_image` a každá čtyřčlenná souřadnice bounding boxu byla převedena na škálu 0–1000 pomocí `normalize_bbox`, aby odpovídala vstupním požadavkům LayoutLMv3.

Výsledné příklady (id, tokeny, bounding boxy, ner tagy, cesta k obrázku a samotné objekty PIL) byly předány do frameworku `datasets`, kde byly automaticky připraveny pro trénink a vyhodnocení modelu.

Načtení datasetu. Před samotným tréninkem byla volána funkce `prepare_dataset`, která načetla dataset přes `load_dataset("./layoutlmv3.py")`. Pro transformaci příkladů do formátu očekávaného modelem byla definována vlastní `Features` obsahující pole `bbox` (normalizované bounding boxy), `input_ids`, `pixel_values` (obrázky přeškálované na rozměry $(3, 224, 224)$), `attention_mask` a `labels` jako multihot vektory $(512, \text{num\_labels})$.

[3] https://github.com/shivarama23/LayoutLMV3

Uvnitř lokální funkce `prepare(ex)` byl aplikován `processor` (kombinující vizuální a textovou část LayoutLMv3) na vstupní obrázek, seznam tokenů a odpovídající bounding boxy, přičemž bylo povoleno oříznutí a doplnění na maximální délku. Z původních BIO labelů, které `processor` vracel v poli `labels`, byla pomocí `convert_to_multihot` vytvořena binární multihot reprezentace a následně přiřazena zpět do `enc['labels']`. Nakonec byly odstraněny nepotřebné sloupce a připravený `Dataset` spolu s počtem tříd vrácen.

Tokenizace a processor. Pro přípravu vstupů pro LayoutLMv3 byla využita komponenta `LayoutLMv3Processor`, která integruje textový `LayoutLMv3Tokenizer` a vizuální `LayoutLMv3ImageProcessor`. LayoutLMv3 používá Byte-Pair Encoding (BPE) tokenizér odvozený z RoBERTa [35]. Tokenizér přijímá seznam slov, word-level bounding boxy a volitelné word-level labely, rozdělí text na subwords, přidá speciální tokeny (`<s>`, `</s>`), aplikuje padding a truncation na maximální délku (512) a současně přiřadí každému subwordu původní normalizovaný box (škála 0–1000) a odpovídající label či multihot vektor. Výstupem processoru jsou pole `input_ids`, `attention_mask`, `token_type_ids`, `bbox` a volitelně `labels`, připravená v jednotném formátu pro vstup do modelu, čímž je zajištěna přesná synchronizace textové, prostorové a anotacemi doplněná informace.

Komponenta `LayoutLMv3ImageProcessor` slouží k převodu vstupního rastrového obrázku na číselnou reprezentaci vhodnou pro model. Načtený obraz je nejprve přeškálován a oříznut tak, aby měl pevné rozměry (224×224 px), poté je převeden z formátu PIL do tenzorů. Výstupem je pole `pixel_values` typu `float32` s tvarem (3, 224, 224), obsahující naškálované a normalizované hodnoty jednotlivých barevných kanálů připravené pro vstup do konvoluční části modelu LayoutLMv3 [35].

Úprava trénovacího procesu pro více-třídní (multilabel) token-level klasifikaci. Protože původní implementace LayoutLMv3 podporuje víceznačnou klasifikaci jen na úrovni celé sekvence (sequence-level), nebylo možné přímo využít standardní výstupy modelu pro potřeby, kde jeden token může patřit do více kategorií současně. Pro tento účel byly BIO tagy z `ner_tags` převedeny na multihot vektory délky odpovídající celkovému počtu tříd. Každý token byl reprezentován binárním vektorem, kde hodnota 1 na příslušné pozici značí, že token náleží do příslušné třídy (např. pokud má token přiřazeny třídy 1 a 5, budou na těchto pozicích hodnoty 1 a zbylé nepřiřazené třídy budou mít hodnotu 0).

Následně byla vytvořena podtřída `MultiLabelNERTrainer` odvozená od základního `Trainer` z Hugging Face Transformers. V konstruktoru bylo umožněno předání váhových koeficientů pro jednotlivé třídy a inicializována funkce ztráty jako `BCEWithLogitsLoss`, vhodná pro binární klasifikaci každého prvku multihot vektoru. Během výpočtu ztráty byly maskovány nevalidní položky (např. pad tokeny) a *binary cross-entropy* byla počítána pouze na platných pozicích.

Pro vyhodnocení modelu byla definována funkce `compute_metrics`, která nejprve aplikuje `sigmoid` na výstupní logity. Výsledné hodnoty označuje 1/0 pomocí prahu 0,5 – hodnoty větší nebo rovné tomuto prahu jsou klasifikovány jako pozitivní (1), nižší jako negativní (0). Následně maskuje nevalidní tokeny a počítá klasické makro metriky (precision, recall, F1) nad true/false hodnotami. Díky tomuto nastavení je získán detailní pohled na schopnost modelu detekovat a správně přiřadit všechny překrývající se entity v rámci jedné textové sekvence.

Konfigurace a spuštění tréninku. V hlavní funkci `train(args)` byly nejprve načteny předzpracované trénovací a evaluační datasety. Poté byl vytvořen konfigurační objekt

`AutoConfig` z předtrénovaného checkpointu `microsoft/layoutlmv3-base`, ve kterém byly nastaveny počet tříd `num_labels` a slovníky `id2label` a `label2id`. Model pro tokenovou klasifikaci byl načten pomocí `AutoModelForTokenClassification`.

Pro vyvážení nerovnoměrného rozložení entit byl spočítán váhový vektor `pos_weight` z četností každé třídy (uložených v poli `counts`). Váhový vektor byl definován logaritmickou funkcí

$$\log\left(1 + \frac{N - n_i}{n_i}\right),$$

kde N je celkový počet tokenů a n_i počet tokenů třídy i. Síla byla maximalizována na 5.0 a výsledek byl uložen na disk. Následně byly definovány tréninkové argumenty pomocí `TrainingArguments`, kde byl specifikován výstupní adresář, velikost batch, počet epoch, strategie evaluace a ukládání modelu (po každé epoše) a sledovaná metrika `F1` pro automatické načítání nejlepšího modelu. Instance vlastního trenéra `MultiLabelNERTrainer` byla vytvořena tak, že mu byl předán model, tréninkové argumenty, datasety, tokenizer (processor), výchozí data collator, funkce `compute_metrics` a váhové koeficienty `class_weights`. Voláním `trainer.train()` bylo spuštěno trénování.

Příprava predikcí pro evaluaci. Po natrénování modelu byly provedeny predikce na testovací sadě, které byly následně převedeny do formátu kompatibilního s evaluačním skriptem. Nejprve byl načten testovací dataset a transformován pomocí `processor` (stejně jako při tréninku) v lokální funkci `prepare_examples`, kde byla opět provedena tokenizace, padding/truncation, normalizace bounding boxů a převod BIO tagů na multihot vektory. Následně bylo zavoláno `trainer.predict(eval_ds)`, čímž byly získány surové logity ve tvaru $(N, 512, C)$ a odpovídající ground-truth multihot labely.

Na každý token byl aplikován `sigmoid` a byly maskovány paddingové pozice (kde součet ground-truth labelů je nulový), aby bylo možné zaměřit se pouze na reálné tokeny. Optimalizační prahová hodnota τ pro převod pravděpodobností na binární predikce byla určena funkcí `find_best_threshold` na základě F1 skóre na validační sadě. Při inferenci pak byly pro každý token v řetězci vytvořeny tři seznamy: skutečných tříd (`true_labels`), predikovaných tříd s jejich skóre (`predicted_labels`) a *top-3* predikcí seřazených podle pravděpodobnosti (`top_predictions`). Nejlepší tři predikce jsou ukládány kvůli pozdějšímu vyhodnocování s nižší prahovou hodnotou. Nakonec byl pro každý vstupní obrázek generován JSON soubor se strukturou k nahlédnutí zde 3.2.

Post-processing predikcí do finálního JSON formátu. Pro převod token-level predikcí do cílové struktury JSON (stejná jako u YOLO 3.1) byl použit skript `process_folder`, který načítá každý z vygenerovaných souborů `.json` obsahujících predikce a seskupuje je do jednotlivých záznamů podle názvu (`library_id`). Nejprve bylo definováno mapování původních labelů na cílové klíče (`label_map`) a množina polí, která může být ukládána jako seznamy (`list_fields`). Skript prochází tokeny v pořadí jejich indexů, rozhoduje o token-level predikovaných labelech podle prahové hodnoty konfidencí a spojuje je do spanů podle BIO schématu (počáteční `B-` a pokračovací `I-` značky).

Každý span je převeden na textovou hodnotu (sloučením tokenů) a ohodnocen nejvyšším skóre z jeho tokenů. Poté je dle `label_map` přidán do výstupního záznamu buď jako jediná položka (pro jedinečné elementy), nebo jako seznam (pro opakující se pole jako `author` či `editor`). Výsledný slovník vždy obsahuje klíče `task_id`, `library_id` a pro každý bibliografický prvek pole dvojic [text, score]. Každý finální JSON je poté uložen do cílového adresáře. Tyto výstupy byly připraveny pro porovnání s ground-truth pomocí evaluačního

skriptu, který provádí normalizaci textu a výpočet metrik (precision, recall, F1) pro každou třídu.


```
{
    "file_name": "003.jpg",
    "tokens": [{
        "token_idx": "Shakespeare",
        "bbox": [508,901,636,917],
        "true_labels": ["I-AUTOR"],
        "predicted_labels": ["I-AUTOR", 0.9978355765342712],
        "top_predictions": [
            {
            "label": "I-AUTOR",
            "score": 0.9978355765342712
            },{
            "label": "I-TITULEK",
            "score": 0.001272094319574535
            },{
            "label": "I-NAKLADATEL",
            "score": 0.0006950270617380738
            }
        ]
    }]
}
```


Výpis 3.2: Ukázka JSON s predikcemi.

Kapitola 4

Experimenty a výsledky

V této kapitole jsou popsány metriky pro hodnocení přesnosti a představeny výsledky experimentů, které byly provedeny za účelem hodnocení v úloze detekce a extrakce bibliografických údajů z dokumentů výkonu modelů YOLOv11 a LayoutLMv3. Nejprve jsou popsány experimenty, kde jsou uvedeny klíčové metriky jako přesnost, úplnost, F1-skóre a specifické chování modelů při detekci a extrakci dat. Dále bude provedeno porovnání těchto dvou přístupů a identifikace jejich silných a slabých stránek. Nakonec budou rozebrány chyby a omezení spojené s těmito modely, které mohou ovlivnit výsledky při použití v reálných podmínkách.

4.1 Hodnocení přesnosti

Pro objektivní vyhodnocení kvality modelů byly využity standardní metriky běžně používané v úlohách detekce objektů. Hlavními ukazateli jsou přesnost (*precision*), úplnost (*recall*), průměrná přesnost (*mAP*) a hodnota *F1-score*. Tyto metriky byly sledovány jak v rámci trénovací, tak validační sady a umožňují porovnání jednotlivých přístupů i jejich vývoj v průběhu trénování.

Precision (přesnost) udává podíl správně predikovaných entit ze všech detekovaných; vysoká precision znamená nízký počet falešně pozitivních detekcí.

$$\text{Precision (P)} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall (úplnost) vyjadřuje podíl správně detekovaných entit ze všech skutečných entit; nízký recall naznačuje, že model některé relevantní objekty přehlíží.

$$\text{Recall (R)} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

mAP@50 (mean Average Precision při IoU ≥ 0.5) shrnuje výkon modelu napříč všemi třídami a kombinuje aspekty precision a recall pro prahovou hodnotu Intersection over Union 0,5.

$$\text{mAP@50} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \Big(\frac{1}{|\mathcal{I}_c|} \sum_{i \in \mathcal{I}_c} \text{AP}_{c,i}(\text{IoU} \geq 0.5) \Big)$$

F1-score je harmonický průměr precision a recall a poskytuje vyvážené hodnocení mezi těmito metrikami.

$$\text{F1-score} = 2 \cdot \frac{\text{P} \cdot \text{R}}{\text{P} + \text{R}} = \frac{2\,\text{TP}}{2\,\text{TP} + \text{FP} + \text{FN}}$$

Loss (ztráta) představuje optimalizační metrikou během tréninku, numericky vyjadřuje rozdíl mezi predikcemi modelu a skutečnými hodnotami; nižší loss značí lepší shodu s realitou.

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)\right]$$

Definice metrik vycházejí z běžně používaných postupů při trénování a vyhodnocování detekčních modelů, zejména v rámci datasetů jako je `COCO` [**?**] a běžně používaných knihoven jako `scikit-learn` [26].

Pro vyhodnocení výsledků byl použit dodaný **evaluační skript**, který automaticky porovnává JSON soubory s predikcemi a referenčními hodnotami. Skript zajišťuje normalizaci textů, párování hodnot podle míry podobnosti (s využitím prahu 10 % CER) a výpočet standardních metrik, včetně průměrné přesnosti (mAP) při různých hodnotách confidence skóre [17].

4.2 Experimenty s modelem YOLOv11

Tato část je zaměřená na analýzu výkonu modelů YOLO (You Only Look Once) různých velikostí, konkrétně porovnání výsledků mezi těmito modely na stejné datové sadě. Poté následuje podrobnější zhodnocení nejlepšího modelu.

Pro účely této analýzy byly porovnány tři různé velikosti modelu YOLOv11 – **YOLOv11S**, **YOLOv11M** a **YOLOv11L** – trénované a testované na totožné datové sadě. Výsledky zde již odpovídají vyhodnocení evaluačním skriptem s GT údaji. Cílem bylo zjistit, jak velikost modelu ovlivňuje detekční výkon. Tabulka níže shrnuje průměrné hodnoty metrik pro všechny třídy a celkovou přesnost modelů.

Model	Recall	Precision	F1	AP
YOLOv11S	0.53	0.63	0.57	0.37
YOLOv11M	0.52	**0.69**	**0.58**	**0.39**
YOLOv11L	**0.54**	0.62	0.57	0.38

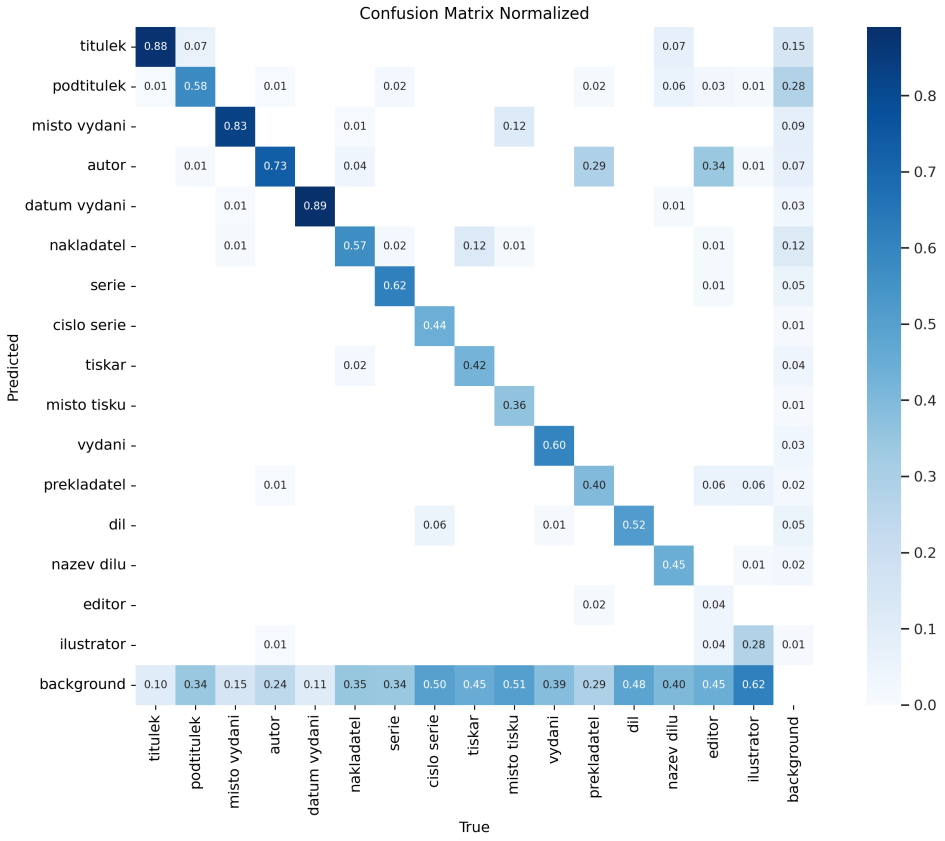
Tabulka 4.1: Srovnání průměrného výkonu modelů YOLOv11 různých velikostí.

Přestože model **YOLOv11L** dosahuje nejvyšší hodnoty *Recall*, model **YOLOv11M** překonává ostatní varianty ve většině klíčových metrik – dosahuje nejvyšší *Precision*, *F1-score* i *Average Precision (AP)*. To naznačuje, že poskytuje nejvyváženější výkon s nižším počtem falešně pozitivních detekcí a zároveň si zachovává solidní celkovou přesnost.

Model **YOLOv11S** mírně zaostává ve všech metrikách a jeho výkon je konzistentně nižší než u větších variant. Model **YOLOv11L** vykazuje nejvyšší *Recall*, což může naznačovat schopnost detekovat více objektů, ale za cenu většího množství falešných detekcí.

Na základě těchto výsledků byl jako nejvýkonnější vyhodnocen model **YOLOv11M**, který byl dále podroben detailnímu vyhodnocení.

Podrobnější vyhodnocení nejlepšího modelu Pro vyhodnocení výkonnosti detekčního modelu YOLOv11M byla použita i normalizovaná matice 4.1 generovaná přímo knihovnou *Ultralytics*, která ukazuje míru správné a chybné klasifikace jednotlivých tříd. Tyto



Obrázek 4.1: Normalizovaná matice.

hodnoty ukazují, jak si vede model pouze při detekování objektů, není zde zahrnuto vyhodnocení pomocí OCR a porovnání s výslednými textovými údaji. Z matice vyplývá, že model si nejlépe vede při detekci kategorií jako `dateIssued` (0.89), `title` (0.88), `author` (0.73) a `placeTerm` (0.83), což značí vysokou přesnost v identifikaci těchto strukturálních prvků dokumentu.

Naopak horších výsledků bylo dosaženo u tříd jako `seriesNumber` (0.44), `subTitle` (0.58), `publisher` (0.57), `manufacturePublisher` (0.42) nebo `subTitle` (0.58), kde je patrná vyšší míra záměny s ostatními třídami. Poměrně častá je i záměna těchto tříd za `background`, což může souviset s vizuální podobností nebo menší četností těchto entit v trénovacím souboru. Významná část chyb se týká právě záměny za třídu `background`, např. u tříd jako `editor`, `illustrator` nebo `partName`, kde podíl nesprávné klasifikace na `background` překračuje 40 %. Tento jev poukazuje na možnou potřebu dalšího ladění modelu, případně na potřebu vyváženějšího datasetu.

Nejlepších výsledků dosáhl model **YOLOv11M**, který byl následně podrobněji analyzován na základě jednotlivých tříd. Následující tabulka shrnuje metriky pro každou kategorii – výpočet přesnosti (Precision), úplnosti (Recall), F1 skóre a průměrné přesnosti (AP). Výsledky zde již odpovídají vyhodnocení evaluačním skriptem s GT údaji.

Výsledky ukazují, že model YOLOv11M dosahuje nejvyšší přesnosti u kategorií s vyšší četností, jako jsou `dateIssued`, `placeTerm` a `author`, kde F1 skóre přesahuje hodnotu 0,70. Naopak výrazně slabší výkonnost byla zaznamenána u méně zastoupených kategorií, jako `editor` nebo `illustrator`, kde jsou hodnoty Recall i Precision nízké. Celkově model

Tabulka 4.2: Výsledky detekce jednotlivých tříd modelem YOLOv11M při confidence 0,25.

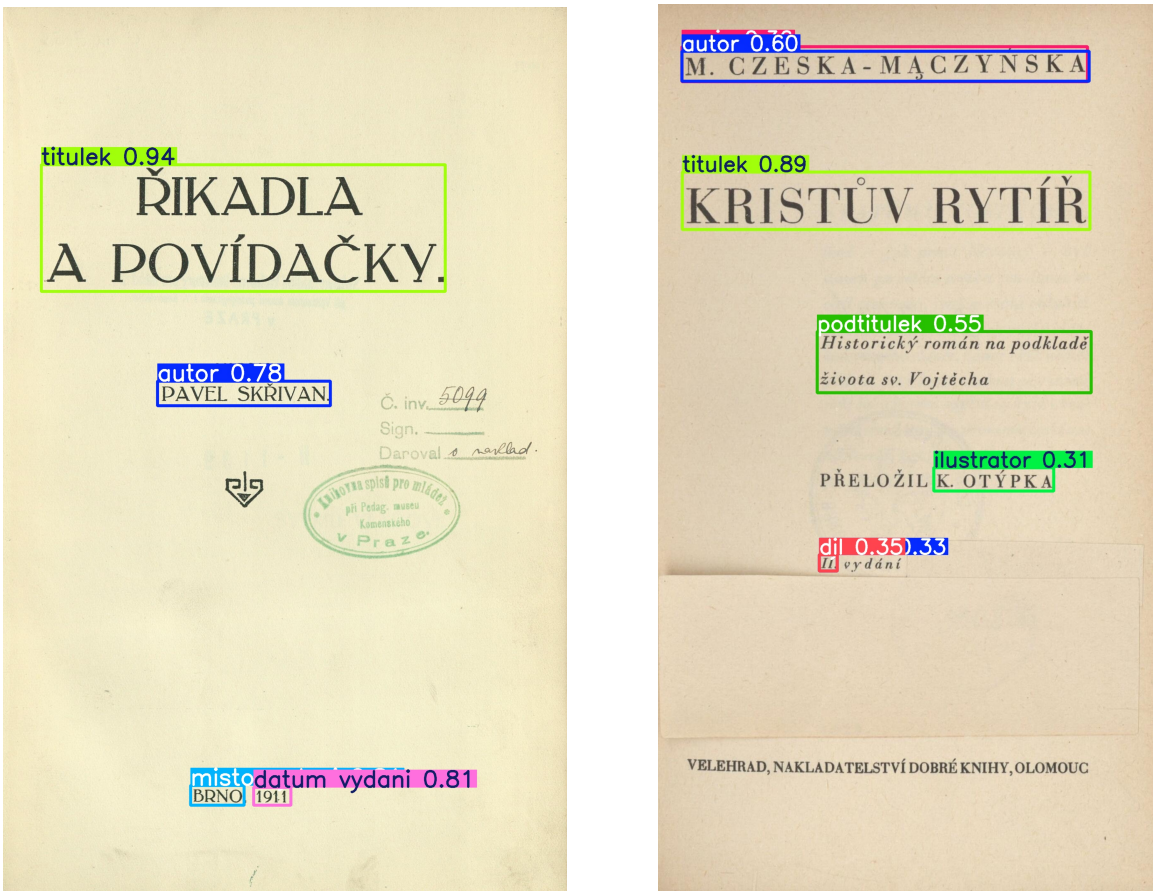
Třída	**Recall**	**Prec.**	**F1**	**AP**
title	0.61	0.62	0.62	0.41
subTitle	0.47	0.55	0.51	0.28
partName	0.37	0.81	0.51	0.29
partNumber	0.52	0.81	0.63	0.42
seriesName	0.57	0.72	0.63	0.42
seriesNumber	0.56	0.82	0.66	0.49
edition	0.57	0.76	0.65	0.48
publisher	0.55	0.63	0.59	0.37
placeTerm	0.78	0.86	0.82	0.68
dateIssued	0.80	0.86	0.83	0.71
manufacturePublisher	0.43	0.57	0.49	0.24
manufacturePlaceTerm	0.33	0.59	0.43	0.20
author	0.73	0.67	0.70	0.52
illustrator	0.35	0.69	0.47	0.26
translator	0.57	0.78	0.66	0.48
editor	0.05	0.25	0.08	0.02
Průměr	**0.52**	**0.69**	**0.58**	**0.39**

vykazuje průměrnou přesnost 0,69 a F1 skóre 0,58, což z něj činí nejvýkonnější variantu mezi porovnávanými modely.

Na obrázku 4.2 jsou znázorněny příklady výstupů modelu YOLOv11M. První část zobrazuje úspěšné detekce, kde jsou objekty správně rozpoznány a lokalizovány s odpovídajícími anotacemi třídy. Tyto případy ilustrují silné stránky modelu při práci s dobře strukturovanými a čitelnými vstupy. Druhá část naopak ukazuje příklady selhání, které zahrnují špatné označení (FP), kdy model identifikoval objekt, ale přiřadil chybný štítek, a rovněž neprovedené detekce (FN), kdy objekt na obrázku nebyl rozpoznán. Nejčastější příčinou těchto chyb bývá nízký kontrast, překrytí textu nebo variabilita ve formátu záznamu. Tyto příklady pomáhají lépe porozumět chování modelu v reálných podmínkách a ukazují možné oblasti pro další zlepšení.

Obrázek 4.2a ukazuje příklad správné a kompletní detekce. Jedná se o stránku prvního typu, kterého je většina titulních stran v datasetu. Obrázek obsahuje výrazně vyznačený titulek v horní části, menším písmem autora pod titulkem a vydavatelské informace ve spodní části, a to pouze místo vydání a rok.

Na obrázku 4.2b je znázorněna ukázka chybné detekce, kde došlo k záměně třídy `illustrator` za `translator`. Tento typ chyby není ojedinělý – model často zaměňuje třídy `illustrator`, `translator` a `editor`. Tyto entity se na stránkách publikací obvykle vyskytují na podobných pozicích a jejich identifikaci napomáhá především okolní text (např. „ilustroval“, „přeložil“, „upravil“). To stejné platí o třídě `part` a `edition`. Jelikož model YOLO pracuje výhradně s vizuálními informacemi a nevyužívá OCR textovou vrstvu, není schopen tyto klíčové jazykové rozdíly zohlednit. To následně vede k vyšší míře záměn právě mezi těmito třídami. Tento problém naznačuje, že pro přesnější klasifikaci by bylo vhodné zvážit kombinaci vizuální detekce s textovou analýzou (např. pomocí OCR).



(a) Ukázka správné detekce.

(b) Ukázka některých chybných detekcí.

Obrázek 4.2: Porovnání správných a některých chybných detekcí modelu YOLOv11M.

4.3 Experimenty s modelem LayoutLMv3

V této části je prezentována sada experimentů zaměřených na hodnocení výkonnosti multimodálního předtrénovaného modelu LayoutLMv3 při zpracování dokumentových obrazů. Cílem je nejprve posoudit, jaký přínos má kombinace textových a vizuálních reprezentací pro úlohu extrakce bibliografických údajů a dále provést detailní analýzu nejlepšího modelu po jeho doladění na datasetu. Cílem této struktury je objasnění, jak multimodální architektura modelu ovlivňuje konečný výkon v praktických scénářích zpracování dokumentů.

Porovnání vlivu modalit na výkon modelu. Tabulka 4.3 ukazuje F1 skóre klasifikace jednotlivých entit při použití tří různých vstupních modalit: čistě poziční informace (`Pozice`), kombinace pozice a textového obsahu (`Pozice + Text`) a dále rozšíření o vizuální informace z obrázku stránky (`Pozice + Text + Obrázek`). Výsledky zde již odpovídají vyhodnocení evaluačním skriptem s GT údaji.

Při použití pouze pozičních informací se ukazuje, že některé entity, zejména ty, které mají jasně definovanou pozici na stránce, jsou rozpoznávány poměrně dobře. Například entity jako `placeTerm` a `dateIssued` mají vysoké F1 skóre při použití pouze pozičních informací (0.76 a 0.74), což naznačuje, že jejich pozice na stránce je dostatečně jednoznačná pro klasifikaci. Naopak u některých entit, jako je `illustrator` (F1 0.28) nebo `editor` (F1

0.23), je pozice sama o sobě nedostatečná a model potřebuje k rozpoznání textový kontext, aby mohl správně klasifikovat.

Přidání textových informací (kombinace `Pozice + Text`) výrazně zvyšuje přesnost napříč všemi entitami. Například F1 skóre u `translator` vzrostlo z 0.25 na 0.41 a u `edition` z 0.68 na 0.78. Tento nárůst ukazuje, že textový kontext poskytuje dodatečné signály, které jsou obzvlášť cenné pro rozpoznání entit s méně předvídatelnou nebo rozmanitou pozicí.

Další rozšíření o vizuální modalitu (kombinace `Pozice + Text + Obrázek`) přináší selektivní zlepšení především u méně frekventovaných nebo graficky zdůrazněných entit. Největší nárůst je pozorován u `partNumber` (0.36 → 0.44), `translator` (0.41 → 0.51) a `edition` (0.78 → 0.84), kde vizuální prvky pomáhají modelu odlišit jemné typografické nebo umístěním dané textové bloky. U těchto entit může vizuální kontext hrát klíčovou roli, protože grafické zdůraznění (například velikost písma, barvy nebo specifická umístění) může pomoci modelu správně určit, že jde o určitou entitu, kterou by text nebo pozice nezachytily.

Naopak u běžně rozpoznávaných entit, jako jsou `placeTerm` či `dateIssued`, vizuální modalita již nepřidává další užitečné signály, a F1 skóre zůstává na úrovni nebo mírně klesá (např. `partName` z 0.55 na 0.52, `title` z 0.57 na 0.55). Tento jev ukazuje, že u některých entit pozice a textová informace poskytují dostatečné signály pro správnou klasifikaci a vizuální kontext není potřeba, nebo dokonce může zavádět.

Celkově tedy platí, že nejpodstatnější zlepšení přináší jazyková modalita (text), zatímco vizuální modalita (obrázek) je nejefektivnější pro entity, u nichž samotná pozice a text nestačí k jednoznačné klasifikaci. Výsledky potvrzují, že multimodální integrace je výhodná především tam, kde je vizuální kontext nezastupitelný. Takové přístupy jsou tedy nejvíce užitečné u entit s nejednoznačnými nebo graficky zdůrazněnými rysy.

Tabulka 4.3: F1 skóre klasifikace entit podle vstupních modalit při confidence 0,25.

Třída	**Pozice**	**Pozice + Text**	**Pozice + Text + Obrázek**
title	0.53	**0.57**	0.55
subTitle	0.48	**0.59**	0.55
partName	0.47	**0.55**	0.52
partNumber	0.31	0.36	**0.44**
seriesName	0.66	**0.72**	0.71
seriesNumber	0.59	0.62	**0.64**
edition	0.68	0.78	**0.84**
publisher	0.61	0.62	**0.63**
placeTerm	0.76	**0.80**	**0.80**
dateIssued	0.74	0.76	**0.77**
manufacturePublisher	0.38	**0.63**	0.57
manufacturePlaceTerm	0.45	**0.69**	0.62
author	0.68	**0.71**	0.70
illustrator	0.19	0.34	**0.36**
translator	0.25	0.41	**0.51**
editor	0.23	**0.45**	0.40
Průměr	**0.50**	**0.60**	**0.59**

Podrobnější vyhodnocení multimodálního modelu Pro podrobnější vyhodnocení byl vybrán model LayoutLMv3, který kombinuje všechny tři modality a zlepšuje rozpoznání u složitějších tříd. Výsledky zde již odpovídají vyhodnocení evaluačním skriptem s GT údaji.

Na základě metrik uvedených v tabulce 4.4 a celkové přesnosti 0.59 lze výkon modelu popsat následujícím způsobem. Nejvyšších hodnot dosáhla třída `edition` (F1 0.84, AP 0.75), což ukazuje na velmi spolehlivou detekci edičních údajů. Rovněž třída `placeTerm` vykázala vynikající výsledky (F1 0.80, AP 0.70) a třída `dateIssued` si udržela vysokou úroveň výkonu (F1 0.76, AP 0.63), což je zásadní pro správné určení data vydání.

Ve střední kategorii výkonu se nacházejí třídy `author`, `seriesName` a `seriesNumber`, které dosahují F1 kolem 0.70 (AP mezi 0.48 a 0.58). To je dostatečné pro většinu bibliografických prvků, i když se objevují občasné chyby v rozlišení více autorů či čísel série. Třídy `publisher` a `manufacturePublisher` vykazují mírně nižší F1 hodnoty 0.63 resp. 0.58, což naznačuje určité potíže s odlišením různých typů vydavatelských entit.

Naopak nejnižší spolehlivost byla zaznamenána u třídy `illustrator` (F1 0.33, AP 0.21), což znamená, že ilustrátoři jsou pro model nejtěžší na detekci, pravděpodobně kvůli omezenému počtu vzorů a větší variabilitě zápisu. Podobné potíže se vyskytly u tříd `partNumber` (F1 0.44, AP 0.24) a `editor` (F1 0.36, AP 0.22), kde model často mylně nerozpozná nebo přidá nesprávné anotace třídy.

Celkově model dosažené průměrné metriky Recall 0.56, Precision 0.63, F1 0.59 a AP 0.44, což je docela dobrý výsledek, ale prostor pro zlepšení stále existuje, zejména u méně zastoupených a variabilních entit.

Tabulka 4.4: Metriky nejlepšího modelu LayoutLMv3 pro každou třídu při confidence 0,25.

Třída	**Recall**	**Prec.**	**F1**	**AP**
title	0.52	0.55	0.53	0.34
subTitle	0.50	0.56	0.53	0.37
partName	0.48	0.52	0.50	0.36
partNumber	0.35	0.57	0.44	0.24
seriesName	0.68	0.72	0.70	0.58
seriesNumber	0.54	0.78	0.64	0.48
edition	0.78	0.90	0.84	0.75
publisher	0.65	0.61	0.63	0.48
placeTerm	0.77	0.83	0.80	0.70
dateIssued	0.73	0.80	0.76	0.63
manufacturePublisher	0.57	0.60	0.58	0.41
manufacturePlaceTerm	0.51	0.71	0.59	0.43
author	0.68	0.72	0.70	0.54
illustrator	0.28	0.40	0.33	0.21
translator	0.52	0.41	0.46	0.31
editor	0.32	0.42	0.36	0.22
Průměr	**0.56**	**0.63**	**0.59**	**0.44**

Z tabulky 4.5 je patrné, že samotná schopnost modelu správně rozpoznat a klasifikovat jednotlivé textové entity (např. název knihy, autora nebo rok vydání) funguje velmi dobře. Při hodnocení na úrovni přesných pozic tokenů (tedy slov nebo jejich částí) model dosahuje vysokých metrik: přesnosti (Precision) 0,87, úplnosti (Recall) 0,86 a F1 skóre 0,86. To

znamená, že pokud má model k dispozici správně rozpoznaný text a dobře vymezené tokeny, dokáže většinu entit přiřadit správně.

Oproti tomu při vyhodnocení výstupu celého systému včetně vstupního OCR (rozpoznávání textu z obrázku) metriky výrazně klesají (Precision 0,68; Recall 0,55; F1 0,60). Tento pokles ukazuje, že největší slabinou není samotný klasifikační model (např. LayoutLMv3), ale kvalita vstupního textu, který model zpracovává. OCR často produkuje chybně rozdělené, překrývající se nebo špatně rozpoznané tokeny, a tím znemožňuje správné přiřazení metadat i přesto, že by model danou entitu za ideálních podmínek správně identifikoval.

Z toho vyplývá, že pro dosažení lepších výsledků je klíčové buď vylepšit kvalitu OCR, nebo provést ruční kontrolu a korekci tokenů v datech, zejména v místech, kde často dochází k chybám (např. sloučená slova, špatné rozdělení řádků, chybně rozpoznané znaky apod.). Tato manuální kontrola či korekce u automatizace není ideální.

Tabulka 4.5: Porovnání celkových metrik mezi klasifikací geometrie a následnou klasifikací i s OCR textem.

Metrika	**Klasifikace geometrie**	**Klasifikace geometrie + OCR**
Precision	0,87	0,63
Recall	0,86	0,56
F1 Score	0,86	0,59

Porovnání různých velikostí modelu Pro ověření, zda by větší model mohl zlepšit výsledky, byl otestován také model `microsoft/layoutlmv3-large` namísto základní verze (base). Cílem bylo zjistit, zda větší kapacita modelu povede k lepšímu zachycení strukturálních i vizuálních vztahů v dokumentech. Výsledky však ukázaly, že tento přístup nebyl úspěšný — metriky zůstaly velmi nízké, s hodnotami F1 skóre nepřesahujícími 0,02. Pravděpodobnou příčinou je špatná adaptace velkého modelu na specifický formát trénovacích dat nebo problémy s příliš malým množstvím dat pro efektivní *fine-tuning* rozsáhlejší architektury. Tento experiment tedy ukázal, že zvětšení modelu samo o sobě nevede ke zlepšení a může naopak vést k výraznému propadu výkonu.

4.4 Srovnání výsledků modelů YOLOv11 a LayoutLMv3

V tabulce 4.4 jsou shrnuty metriky nejlepšího modelu LayoutLMv3 pro jednotlivé třídy, zatímco tabulka 4.2 uvádí výsledky detekce pomocí modelu YOLOv11M. Porovnání průměrných metrik ukazuje, že LayoutLMv3 dosahuje u uvedených metrik lepších výsledků, i když se jedná pouze o setiny hodnot. To naznačuje, že kombinované textově-prostorové zpracování LayoutLMv3 poskytuje mírně lepší schopnost zachytit víceznačné textové entity.

YOLOv11M naopak vyniká u výrazně vizuálně definovaných prvků. Zejména detekce pole `placeTerm` dosahuje F1 0,82 a pole `dateIssued` F1 0,83. Tyto hodnoty jsou u YOLOv11M o něco vyšší než u LayoutLMv3, což lze přičíst přímému využití vizuálních rysů stránky bez potřeby OCR. Hlavní výhodou u YOLOv11 je detekce oblastí, které jsou následně přímo zpracovávané pomocí PERO, které umožňuje detekovat text v celé označené oblasti, ať už jde o víceřádkový region nebo o jedno samostatné slovo/číslo.

Naopak LayoutLMv3 předčí YOLOv11M u tříd, kde je klíčové rozlišení jemných textových rozdílů a vícenásobných entit v rámci jednoho tokenu. U třídy `edition` dosahuje LayoutLMv3 F1 0,84 (oproti 0,65 u YOLOv11M) a u třídy `seriesName` F1 0,70 (oproti

0,63). Tyto výsledky potvrzují výhodu modelu založeného na propojení textových vstupů, jejich prostorového uspořádání a jemných kontextových vazeb.

Oba přístupy však mají své slabiny. YOLOv11M vykazuje nejnižší F1 u třídy `editor` (0,08) a relativně slabý výkon u `manufacturePlaceTerm` (0,43), zatímco LayoutLMv3 nejhůře rozpoznává třídu `illustrator` (F1 0,33) a `partNumber` (F1 0,44), což je způsobeno omezeným počtem trénovacích příkladů a vysokou variabilitou zápisu.

Pro další zlepšení by se nejspíš vyplatilo kombinovat obě metody – využít YOLOv11M pro robustní vizuální detekci velkých a graficky zřetelných polí a LayoutLMv3 pro detailní token-level klasifikaci toho, co z OCR výstupu poskytne kvalitní textovou reprezentaci.

Jak je patrné z detailního porovnání v tabulce 4.6, model LayoutLMv3 dosahuje lepších výsledků v kategoriích s jemnými textovými rozdíly, zatímco YOLOv11M exceluje u výrazných vizuálních polí.

Tabulka 4.6: Porovnání metrik modelů LayoutLMv3 a YOLOv11M pro jednotlivé třídy při confidence 0,25.

Třída	**LayoutLMv3**				**YOLOv11M**			
	Recall	Prec.	F1	AP	Recall	Prec.	F1	AP
title	0.52	0.55	0.53	0.34	**0.61**	**0.62**	**0.62**	**0.41**
subTitle	**0.50**	**0.56**	**0.53**	**0.37**	0.47	0.55	0.51	0.28
partName	**0.48**	0.52	0.50	**0.36**	0.37	**0.81**	**0.51**	0.29
partNumber	0.35	0.57	0.44	0.24	**0.52**	**0.81**	**0.63**	**0.42**
seriesName	**0.68**	**0.72**	**0.70**	**0.58**	0.57	**0.72**	0.63	0.42
seriesNumber	0.54	0.78	0.64	0.48	**0.56**	**0.82**	**0.66**	**0.49**
edition	**0.78**	**0.90**	**0.84**	**0.75**	0.57	0.76	0.65	0.48
publisher	**0.65**	0.61	**0.63**	**0.48**	0.55	**0.63**	0.59	0.37
placeTerm	0.77	0.83	0.80	**0.70**	**0.78**	**0.86**	**0.82**	0.68
dateIssued	0.73	0.80	0.76	0.63	**0.80**	**0.86**	**0.83**	**0.71**
manufacturePublisher	**0.57**	**0.60**	**0.58**	**0.41**	0.43	0.57	0.49	0.24
manufacturePlaceTerm	**0.51**	**0.71**	**0.59**	**0.43**	0.33	0.59	0.43	0.20
author	0.68	**0.72**	**0.70**	**0.54**	**0.73**	0.67	**0.70**	0.52
illustrator	0.28	0.40	0.33	0.21	**0.35**	**0.69**	**0.47**	**0.26**
translator	0.52	0.41	0.46	0.31	**0.57**	**0.78**	**0.66**	**0.48**
editor	**0.32**	**0.42**	**0.36**	**0.22**	0.05	0.25	0.08	0.02
Průměr	**0.56**	0.63	**0.59**	**0.44**	0.52	**0.69**	0.58	0.39

YOLO má výhodu při detekci delších údajů, protože identifikuje celé textové regiony – například víceřádkové názvy článků – což následně usnadňuje jejich zpracování pomocí OCR. Naproti tomu LayoutLM analyzuje dokument na úrovni jednotlivých slov a jejich pozic, což může vést k chybám při spojování více slov do jednoho celku, zejména pokud jsou rozmístěna nepravidelně nebo přes více řádků.

V kontextu extrakce informací z bibliografických dokumentů je YOLO oproti LayoutLM výrazně jednodušší na přípravu i trénování. YOLO vyžaduje pouze vizuální anotace – označení pomocí ohraničujících rámečků – což je rychlé a přímočaré. Naproti tomu LayoutLM pracuje nejen s vizuálními, ale i textovými a strukturálními informacemi (OCR rozpoznání textu na úrovni slov, pozice v dokumentu), a proto vyžaduje složitější předzpracování dat. Výsledkem je vyšší náročnost na tréninkový proces, i když větší přesnost u složitějších bibliografických údajů.

Kapitola 5

Závěr

Podstatnou součástí této bakalářské práce byla spolupráce na tvorbě anotovaného datasetu, nebo-li vývoj nástroje pro automatické zarovnání knihovních metadat s OCR detekcemi, což umožnilo efektivní propojení strukturovaných záznamů s vizuálními daty. Na tomto základu byly navrženy a implementovány dvě metody pro automatickou extrakci bibliografických metadat z titulních stran historických dokumentů – robustní vizuální detektor YOLOv11L a model LayoutLMv3, který kombinuje prostorové informace s vizuálním a textovým kontextem.

YOLOv11 prokázal silné výsledky při detekci výrazných grafických polí, jako jsou místo vydání (`placeTerm`) a datum vydání (`dateIssued`), kde dosáhl F1 skóre 0,82, resp. 0,83. Naopak LayoutLMv3 exceloval při rozpoznávání jemně strukturovaných textových entit, jako jsou ediční údaje (`edition`, F1 0,84) nebo názvy sérií (`seriesName`, F1 0,70), což bylo umožněno díky jeho schopnosti zpracovávat textové informace v kombinaci s jejich přesnou prostorovou lokalizací.

Celkově model YOLOv11 dosáhl průměrných metrik Recall 0,52, Precision 0,69, F1 0,58 a AP 0,39, zatímco LayoutLMv3 vykázal Recall 0,56, Precision 0,63, F1 0,59 a AP 0,44. Hlavním omezením obou přístupů se ukázala chybovost OCR předzpracování, která snižuje kvalitu vstupních tokenů pro LayoutLMv3 a tím i konečnou přesnost systému. Kritické jsou zejména třídy s nízkým zastoupením a vysokou variabilitou zápisu (např. `illustrator`, `editor`, `partNumber`), kde se objevují největší odchylky od požadovaných metrik.

Pro další zlepšení výsledků by stálo za zvážení zkombinovat oba přístupy do jedné pipeline, v níž by YOLOv11 poskytoval přesné ohraničovací boxy pro velké a graficky zřetelné oblasti a LayoutLMv3 by na těchto předzpracovaných segmentech prováděl detailní token-level klasifikaci. Dále je nutné zlepšit detekci geometrie nebo oblasti v rámci OCR. Rozšíření trénovací sady o příklady méně zastoupených tříd by mohly výrazně posílit schopnost modelů generalizovat i na složitější varianty historických titulních stran.

Navržené metody představují pevný základ pro automatizované zpracování velkých digitálních sbírek a otevírají prostor pro další výzkum v oblasti multimodálního zpracování dokumentů, zejména pak pro integraci OCR korekce s vizuálně-textovými modely.

Literatura

[1] Bhardwaj, A.; Mercier, D.; Dengel, A. a Ahmed, S. DeepBIBX: Deep Learning for Image Based Bibliographic Data Extraction. In:. 2017. ISBN 978-3-319-70095-3.

[2] Cohen, S. *Fuzzywuzzy* online. 2011. Dostupné z: https://github.com/seatgeek/fuzzywuzzy.

[3] Councill, I. G.; Giles, C. L. a Kan, M.-Y. *ParsCit: An open-source CRF reference string parsing package* online. 2008. Dostupné z: https://clgiles.ist.psu.edu/pubs/LREC2008-ParsCit.pdf.

[4] Devlin, J.; Chang, M.-W.; Lee, K. a Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv preprint arXiv:1810.04805* online, 2019. Dostupné z: https://arxiv.org/abs/1810.04805.

[5] GeeksforGeeks. *Rule Based Approach in NLP* online. 2023. Dostupné z: https://www.geeksforgeeks.org/rule-based-approach-in-nlp/.

[6] Goodfellow, I.; Bengio, Y. a Courville, A. *Deep Learning*. Cambridge, MA: MIT Press, 2016. ISBN 978-0262035613.

[7] Google. *Tesseract OCR* online. 2023. Dostupné z: https://github.com/tesseract-ocr/tesseract.

[8] Hochreiter, S. a Schmidhuber, J. Long Short-Term Memory. *Neural Computation*. MIT Press, 1997, sv. 9, č. 8.

[9] Huang, W.; Xu, Y.; Lv, T.; Cui, L. a Wei, F. *LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking* online. 2022. Dostupné z: https://arxiv.org/abs/2204.08387.

[10] Jaume, G.; Ekenel, H. K. a Thiran, J.-P. *FUNSD: A Dataset for Form Understanding in Noisy Scanned Documents* online. 2019. Dostupné z: https://arxiv.org/abs/1905.13538.

[11] Jocher, G. a Qiu, J. *Ultralytics YOLOv11* online. 2024. Dostupné z: https://github.com/ultralytics/ultralytics.

[12] Jurafsky, D. a Martin, J. H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models* online. 2025. Dostupné z: https://web.stanford.edu/~jurafsky/slp3.

[13] Khanam, R. a Hussain, M. *YOLOv11: An Overview of the Key Architectural Enhancements* online. 2024. Dostupné z: https://arxiv.org/abs/2410.17725.

[14] Kim, G.; Hong, T.; Yim, M.; Park, S.; Han, D. et al. *OCR-free Document Understanding Transformer* online. 2022. Dostupné z: https://arxiv.org/abs/2111.15664.

[15] Kingma, D. P. a Ba, J. *Adam: A Method for Stochastic Optimization* online. 2014. Dostupné z: https://arxiv.org/abs/1412.6980.

[16] Kodym, O. a Hradiš, M. Page Layout Analysis System for Unconstrained Historic Documents. In: Springer. *ICDAR 2021*. 2021.

[17] Kohút, J.; Dočekal, M.; Hradiš, M. a Vaško, M. *BiblioPage: A Dataset of Scanned Title Pages for Bibliographic Metadata Extraction*. 2025. Dostupné z: https://arxiv.org/abs/2503.19658.

[18] Lafferty, J.; McCallum, A. a Pereira, F. C. N. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: *Proceedings of the 18th International Conference on Machine Learning (ICML)*. San Francisco, CA: Morgan Kaufmann, 2001.

[19] Lopez, P. *GROBID Documentation* online. 2023. Dostupné z: https://grobid.readthedocs.io/en/latest/.

[20] Makhija, V. a Ahuja, S. Rule based text extraction from a bibliographic database. *DESIDOC Journal of Library and Information Technology*, Leden 2018, sv. 38, s. 5–10.

[21] Moens, M.-F. *Information Extraction: Algorithms and Prospects in a Retrieval Context*. Springer, 2006. Dostupné z: https://link.springer.com/book/10.1007/978-1-4020-4993-4.

[22] Mukkavilli Appalaraju, S.; Jasani, B. a Sundaresan, N. *DocFormer: End-to-End Transformer for Document Understanding* online. 2021. Dostupné z: https://arxiv.org/abs/2106.11539.

[23] Ojokoh, B. *Rule-Based Metadata Extraction for Heterogeneous References* online. 2014. Dostupné z: https://www.computerscijournal.org/vol2no2/rule-based-metadata-extraction-for-heterogeneous-references/.

[24] Park, S.; Shin, S.; Lee, B.; Lee, J.; Surh, J. et al. *CORD: A Consolidated Receipt Dataset for Post-OCR Parsing* online. 2019. Dostupné z: https://github.com/clovaai/cord.

[25] Redmon, J.; Divvala, S.; Girshick, R. a Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*. IEEE, 2016.

[26] scikit-learn developers. *Classification Metrics — scikit-learn documentation* online. 2025. Dostupné z: https://scikit-learn.org/stable/modules/model_evaluation.html.

[27] Shyam, R. a Singh, R. A Taxonomy of Machine Learning Techniques, Prosinec 2021, sv. 8, s. 18–25.

[28] Svenonius, E. *The Intellectual Foundation of Information Organization* print. MIT Press, 2009. ISBN 9780262512619.

[29] Tkaczyk, D.; Collins, A.; Sheridan, P. a Beel, J. *Machine Learning vs. Rules and Out-of-the-Box vs. Retrained: An Evaluation of Open-Source Bibliographic Reference and Citation Parsers* online. 2018. Dostupné z: https://arxiv.org/abs/1802.01168.

[30] Tkaczyk, D.; Szostek, P.; Fedoryszak, M.; Dendek, P. J. a Bolikowski, L. CERMINE: automatic extraction of structured metadata from scientific literature. *International Journal on Document Analysis and Recognition (IJDAR)*. Springer, 2015, sv. 18.

[31] Trochtová, E.; Gistingrová, N. N. a Máčková, Z. *KURZ INFORMAČNÍHO VZDĚLÁVÁNÍ: Bibliografické citace I – tištěné dokumenty* online. Brno: Ústřední knihovna FF MU, 2009. Dostupné z: https://is.muni.cz/do/rect/el/estud/ff/ps07/vecera/index.html.

[32] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L. et al. Attention Is All You Need. *CoRR*, 2017, abs/1706.03762. Dostupné z: http://arxiv.org/abs/1706.03762.

[33] Wikipedia. *Optical character recognition* online. 2025. Dostupné z: https://en.wikipedia.org/wiki/Optical_character_recognition.

[34] Wilson, A. C.; Roelofs, R.; Stern, M.; Srebro, N. a Recht, B. *The Marginal Value of Adaptive Gradient Methods in Machine Learning*. 2018. Dostupné z: https://arxiv.org/abs/1705.08292.

[35] Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C. et al. HuggingFace's Transformers: State-of-the-art Natural Language Processing. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, 2020. Dostupné z: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[36] Xu, Y.; Li, M.; Cui, L.; Huang, S.; Wei, F. et al. *LayoutLM: Pre-training of Text and Layout for Document Image Understanding* online. 2019. Dostupné z: https://arxiv.org/abs/1912.13318.

Příloha A

Obsah odevzdaný prostřednictvím virtuálního úložiště NextCloud

```
xparil05_BP ..... Hlavní složka bakalářské práce
├── README.md ..... Popis struktury a použití
├── biblio ..... Skript pro poloautomatické zarovnání dat
├── LayoutLMv3 ..... Složka pro LayoutLMv3
│   ├── READMEllmv3.md ..... Popis použití LayoutLMv3
│   ├── dataset_prepare ..... Příprava dat pro LayoutLMv3
│   └── train ..... Trénování modelu LayoutLMv3 a generování výstupu
├── YOLOv11 ..... Složka pro YOLOv11
│   ├── READMEyolo.md ..... Popis použití YOLOv11
│   ├── train ..... Trénovací skript a konfigurace pro YOLOv11
│   └── predict ..... Predikční skript a generování JSON výstupů pro YOLOv11
├── video.mp4 ..... Video prezentující výsledky práce
└── text ..... Zdrojový kód textu bakalářské práce
```