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

USPOŘÁDÁNÍ FRAGMENTŮ TEXTU S POMOCÍ JAZYKOVÉHO MODELU

REORDERING TEXT FRAGMENTS USING A LANGUAGE MODEL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. MICHAEL HOLUBEC**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. KAREL BENEŠ**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

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

Zadání diplomové práce

Student:	**Holubec Michael, Bc.**
Program:	Informační technologie a umělá inteligence
Specializace:	Informační systémy a databáze
Název:	**Uspořádání fragmentů textu s pomocí jazykového modelu** **Reordering Text Fragments Using a Language Model**
Kategorie:	Zpracování řeči a přirozeného jazyka

Zadání:

1. Seznamte se se statistickým jazykovým modelováním
2. Seznamte se s problematikou uspořádávání fragmentů textu ve výstupu OCR
3. Navrhněte postup jak pro uspořádávání využít jazykový model
4. Experimentálně ověřte jeho účinnost

Literatura:

- C. Clausner, S. Pletschacher and A. Antonacopoulos, "The Significance of Reading Order in Document Recognition and Its Evaluation," *2013 12th International Conference on Document Analysis and Recognition*, 2013, pp. 688-692.
- Sundermeyer, Martin, Ralf Schlüter and Hermann Ney. "LSTM Neural Networks for Language Modeling." Interspeech 2012.

Při obhajobě semestrální části projektu je požadováno:

- Body 1, 2 a rozpracování bodů 3 a 4.

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

Vedoucí práce:	**Beneš Karel, Ing.**
Vedoucí ústavu:	Černocký Jan, doc. Dr. Ing.
Datum zadání:	1. listopadu 2021
Datum odevzdání:	18. května 2022
Datum schválení:	6. května 2022

Abstrakt

Cílem této práce je sestrojit a experimentálně ověřit účinnost jazykového modelu při identifikaci posloupnosti čtení (Reading Order). K tomuto účelu byl sestrojen jazykový model využívající rekurentní neuronovou síť LSTM. Práce dále navrhuje a implementuje celkem tři metody, jazykovou analýzu, prostorovou analýzu a kombinovanou analýzu, pomocí kterých je posloupnost čtení identifikována. Jazyková a kombinovaná analýza ke své činnosti přímo používají vytvořený jazykový model. Úspěšnost identifikace posloupnosti prostřednictvím všech tří metod byla změřena na třech datasetech obsahující novinové články s různým rozložením. Jazyková analýza dosahuje úspěšnosti 57,6 %, prostorová analýza dosahuje 91,6 %. Nejlepších výsledků dosahuje kombinovaná analýza, která vykazuje úspěšnost 92,9 %. Práce ukazuje, že jazykový model lze pro identifikaci posloupnosti čtení použít, avšak výsledky experimentů naznačují, že je vhodné zpracování odhadu posloupnosti doplnit o další informace, jako je to například v kombinované analýze, která pracuje jak s jazykovým modelem, tak s prostorovými informacemi.

Abstract

The aim of this work is to construct and experimentally verify the effectiveness of the language model in identifying the reading order. For this purpose language model with LSTM architecture was constructed. This work designs and implements three methods which are used to identify reading order. These methods are Language analysis, Spatial analysis and Combined analysis. Language analysis and combined analysis used constructed language model. The success of the language model, and all three methods, was measured on three datasets containing newspaper articles. Language analysis reaches 57,6 % and spatial analysis reaches 91,6 %. Combined analysis achieved the best results 92,9 %. The work shows that the language model can be used to identify reading order but use of additional data (e.g. spatial data) is recommended.

Klíčová slova

Posloupnost čtení, Jazykový model, Jazyková analýza, Prostorová analýza

Keywords

Reading order, Language model, Language analysis, Spatial analysis

Citace

HOLUBEC, Michael. *Uspořádání fragmentů textu s pomocí jazykového modelu*. Brno, 2022. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Karel Beneš

Uspořádání fragmentů textu s pomocí jazykového modelu

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Karla Beneše. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Michael Holubec
17. května 2022

Poděkování

Touto cestou bych rád poděkoval vedoucímu své práce Ing. Karlu Benešovi za připomínky, rady a odborné informace, které mi v průběh zpracování této práce poskytl.

Dále bych chtěl poděkovat své přítelkyni, která mi byla při vypracování práce velkou oporou.

Obsah

1 Úvod **2**

2 Digitalizace a rekonstrukce textového dokumentu **4**
2.1 Optické rozpoznávání znaků 4
2.2 Analýza rozložení 6
2.3 Analýza logické struktury 8
2.4 Posloupnost čtení 9

3 Statistické jazykové modelování **15**
3.1 Modely založené na četnostech 15
3.2 Neurální jazykové modely 16

4 Návrh analýzy posloupnosti čtení **23**
4.1 Návrh systému 24
4.2 Vstupní data 25
4.3 Struktura posloupnosti čtení 26
4.4 Metriky 27
4.5 Analyzátor 27

5 Implementace analýzy posloupnosti čtení **29**
5.1 Použití jazykového modelu pro analýzu posloupnosti čtení 29
5.2 Jazyková analýza 30
5.3 Prostorová analýza 32
5.4 Kombinovaná analýza 36
5.5 Ukázka analýzy posloupnosti čtení 38

6 Experimenty **40**
6.1 Sestrojení jazykového modelu 40
6.2 Experimenty s jazykovým modelem 42
6.3 Popis experimentu identifikace posloupnosti čtení a použitých dat 44
6.4 Identifikace posloupnosti čtení 46
6.5 Zhodnocení 49

7 Závěr **51**

Literatura **52**

Kapitola 1

Úvod

V době moderních informačních technologií lze zaznamenat ústup od používání dokumentů v jejich listinné podobě, které se snažíme nahradit jejich digitální formou. Spousta dokumentů však vznikla v době před nástupem elektronických systémů. Proto jsou dokumenty často digitalizovány. Důvody pro digitalizaci dokumentů jsou různé. Od zachování podoby a obsahu historických pramenů přes obchodní a legislativní důvody až po cílený převod původně tištěných a ručně psaných dokumentů do elektronické podoby. V některých případech může být dostačující pořízení fotky nebo skenu dokumentu a jeho uchování ve formě digitálního obrázku. Jindy je však vhodné získat ze snímku dokumentu informace v něm obsažené, jako je například textový obsah, případně identifikovat vhodné segmenty dokumentu (nadpis, autor, jednotlivé odstavce) pro cílené vyhledávání či strojové zpracování těchto segmentů. Jednou z komplexních úloh digitalizace je kompletní rekonstrukce snímku dokumentu do jeho elektronické podoby se zachováním formátování, struktury, případně i typem písma. Rekonstrukce se skládá z několika dílčích kroků a jedním z těchto kroků je identifikace posloupnosti čtení (Reading Order) jednotlivých elementů stránky, která definuje tok textu dokumentem.

Běžně identifikace posloupnosti používá geometrických metod, pomocí kterých analyzuje rozložení textu v dokumentu a na základě vzdálenosti mezi regiony, sousedností a jiných prostorových údajích definuje mezi těmito regiony posloupnost čtení. Cílem této práce je experimentálně ověřit, zda lze posloupnost čtení identifikovat pomocí moderních jazykových modelů, konkrétně pomocí jazykového modelu postaveného na rekurentní neuronové síti LSTM. Jazykový model, na rozdíl od geometrických metod, nepracuje s prostorovými daty, ale s textovým obsahem regionů. V této práci je jazykový model použit pro odhadování pravděpodobnosti návaznosti regionů.

K identifikaci posloupnosti jsou navrženy a implementovány tři metody, pomocí kterých je posloupnost čtení definována. Těmito metodami jsou jazyková analýza, prostorová analýza a kombinovaná analýza. Jazyková analýza odhaduje posloupnost čtení čistě z textového obsahu prvků dokumentu. Prostorová analýza využívá k definici posloupnosti prostorové informace, jako je umístění prvku na stránce, či vzájemné vztahy prvků. Kombinovaná analýza spojuje obě zmíněné metody tak, aby využila všech jejich výhod.

Při tvorbě jazykového modelu a implementaci jednotlivých metod bylo nutné využít poznatků jak z odvětví zabývajících se digitalizací dokumentů, tak také z oboru statistického jazykového modelování, který se zabývá stavbou jazykových modelů. Tyto informace jsou v práci detailně popsány v kapitolách 2 a 3. Návrh a implementace zmíněných metod jsou podrobně popsány v kapitole 4, respektive 5. Metody pro identifikaci posloupnosti byly experimentálně vyhodnoceny na třech datasetech obsahující novinové články s růz-

ným rozložením elementů. Výsledky jsou uvedeny v kapitole 6 spolu s popisem použitého jazykového modelu.

Kapitola 2

Digitalizace a rekonstrukce textového dokumentu

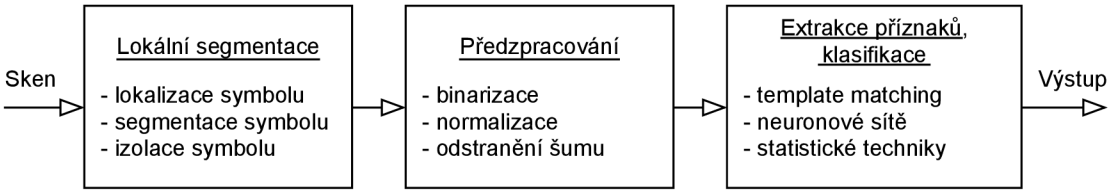
Proces digitalizace sestává z několika částí. Prvním krokem je samotné pořízení snímku dokumentu, jehož kvalita má zásadní vliv na jeho následné zpracování. Textový obsah je z dokumentu extrahován pomocí nástroje poskytující optické rozpoznávání znaků (OCR). Následuje proces analýzy rozložení (Layout Analysis), který se zabývá segmentací stránky a identifikací jednotlivých komponent (slova, věty, odstavce), ovšem bez jejich hlubšího pochopení a rozdělení. Veškeré dosud získané informace jsou předány procesu, který analyzuje logickou strukturu dokumentu (Logical Structure Analysis nebo také Logical Layout Analysis). Analýza a detekce logické struktury se snaží identifikovat logické oblasti (například nadpis, autor článku, tabulky obsahu) a jejich vzájemné vztahy (Reading Order). Proces analýzy logické struktury označujeme také jako Document Understanding, protože v rámci této analýzy se snažíme pochopit význam elementů a využít je při dalším zpracování, například při extrakci informací nebo celkové rekonstrukci dokumentu.

Při digitalizaci je nutné rozlišovat, zda zpracováváme dokumenty tištěné či ručně psané. Především z hlediska rozpoznávání znaků, kdy u tištěných dokumentů jsou zpravidla použity standardizovaná písma, pro která dokážeme vyrobit vhodný klasifikátor, digitalizace ručně psaných dokumentů je však odvislá od autorova rukopisu. Nástroje zpracovávající ručně psané dokumenty musí obsahovat robustní klasifikátor pro rozpoznávání znaků, který správně klasifikuje ručně psané znaky od různých autorů.

Principy a postupy popsané touto prací se pro oba typy dokumentů částečně shodují, avšak tato práce popisuje primárně algoritmy pro zpracování tištěných dokumentů.

2.1 Optické rozpoznávání znaků

Optické rozpoznávání znaků (OCR) je proces převodu obrázku tištěného nebo ručně psaného symbolu do elektronické podoby [42]. První zmínky o optickém rozpoznávání se objevují na počátku 19. století s patentem popisující pomocné zařízení pro nevidomé, v roce 1912 je OCR využito pro převod textu do Morseovy abecedy a přenos skrze telegram. Prvním zařízením považovaným za moderní OCR systém je GISMO vyvinutý v roce 1954 D. Shepardem pracující rychlostí 1 znak za minutu [42]. Dnešní OCR systémy pracují mnohem rychleji, podporují různé jazykové sady a jsou standardní součástí nejen domácích skenerů, ale lze je používat i v mobilních telefonech.



Obrázek 2.1: Rozvržení jednotlivých procesů při převodu snímku dokumentu, respektive obrázků znaků do jejich odpovídající elektronické podoby.

Proces rozpoznávání znaků se skládá z několika dílčích kroků, které jsou znázorněný na obrázku 2.1. Prvním a kritickým krokem je samotné pořízení snímku dokumentu, jehož kvalita má zásadní vliv na následné zpracování. V další fázi probíhá lokalizace, segmentace a izolace jednotlivých symbolů, pro který bude proveden převod do elektronické podoby. Následuje fáze předzpracování, která zpracuje a upravuje obrázky znaků pro jejich lepší rozpoznání klasifikátorem. Předzpracování využívá řadu algoritmů, pomocí kterých je obrázek upraven, od binarizace (převod do černé a bílé barvy pro odstranění pozadí), normalizace (jednotná velikost symbolů, odstranění sklonu písma, identifikace orientace stránky a její správné natočení), až po vyhlazení a odstranění šumu v obrázku [12]. Část z těchto technik je znázorněna na obrázku 2.2.

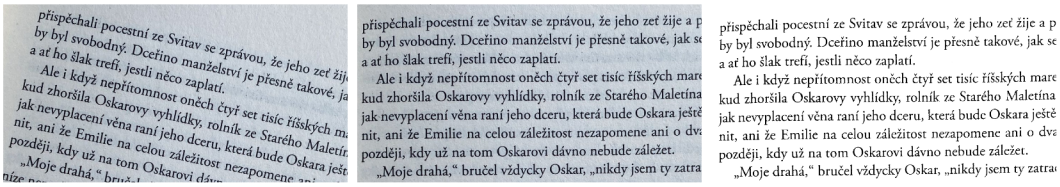
Upravené obrázky symbolů jsou vstupem procesu segmentace, během kterého jsou z obrázku symbolu separovány jednotlivé komponenty, jako například vertikální a horizontální linky, smyčky a podobně. Kvalita segmentace a množství separovaných komponent přímo ovlivňuje míru úspěšnosti rozpoznávání. Protože pro některé klasifikátory nemusí být reprezentace v podobě oddělených linek symbolu dostačují, je výstup segmentace vstupem fáze reprezentace, která identifikuje další charakteristiky symbolu. Může se jednat o statistické nebo topologické charakteristiky. Veškeré vlastnosti symbolu jsou extrahovány a předány klasifikátoru, který na základě těchto vlastností klasifikuje a vrátí třídu symbolu, případně vrátí přímo symbol. Poslední fází je post-processing, který může využít syntaktických a sémantických pravidel cílového jazyka pro rozpoznání případné chyby a zpětné úpravy klasifikátoru [9, 13].

Pro klasifikaci symbolu na základě jeho charakteristik existují různé přístupy. Jedním z nich je rozpoznání symbolu na základě šablony (Template Matching nebo také Matrix matching). Metoda je založena na porovnání vstupu se sadou prototypů z každé třídy, do které může znak patřit. Porovnávají se jednotlivé pixely vstupu s pixely prototypu, pro které je na základě shody definováno skóre. Pokud dojde ke shodě, skóre je navýšeno, a naopak pokud ke shodě nedojde, skóre se sníží. Jako vhodný znak je vybrán ten, jehož skóre podobnosti je ze všech porovnání nejvyšší [35].

Další metody jsou založeny na statistickém přístupu maximalizující pravděpodobnost zařazení vzoru do určité třídy. Používány jsou také techniky využívající různých struktur, které reprezentují jednotlivé znaky ve formě grafů, především v podobě stromů (užívané například u rozpoznávání čínských znaků).

Moderním přístupem pro rozpoznávání znaků na základě extrahovaných příznaků je užití neuronových sítí [9]. Při použití neuronových sítí je možné vynechat některé kroky procesu předzpracování, protože neuronová síť je schopna se vlastnostem dokumentu přizpůsobit, pokud k tomu byla natrénována.

Optické rozpoznávání a celkové zpracování se odlišuje dle typu transkripce, celosvětově se používá 26 různých typů písma (latinka, cyrilice, arabské a hebrejské písmo atd.).



(a) Originál (b) Natočení (c) Binarizace a vyhlazení

Obrázek 2.2: Úkolem předzpracování je upravit snímek dokumentu nebo symbolu do vhodné podoby tak, aby snímek dokázaly zpracovat návazné procesy segmentace, extrakce příznaků a klasifikace symbolů. V průběhu předzpracování probíhá například identifikace orientace dokumentu na snímku a jeho rotace do správné polohy 2.2b. Pomocí binarizace 2.2c je odstraněno pozadí a zvýrazněn samotný obsah dokumentu.

U některých jazyků je rozpoznávání značně náročné. Například systém pro zápis japonštiny Kandži obsahuje přes 3300 různých znaků. Pro takto rozsáhlé znakové sady se zpravidla používají dva klasifikátory. První klasifikátor definuje na základě hrubého odhadu množinu možných kandidátů (například 100) které odpovídají zkoumanému znaku. Druhý klasifikátor využívá takto definovanou množinu a s pomocí komplexních informací provádí konečné rozhodnutí o třídě, do které symbol přísluší [42].

Optické rozpoznávací systémy dělíme dle typu užití. Prvním typem jsou obecné OCR, které pracují s různými druhy dokumentů. Z důvodu své obecnosti nemusí být přesné a jsou zaměřeny především na tištěné dokumenty. Druhým typem jsou doménově specifické OCR, které zpracovávají určitý typ dokumentů, například bankovní šeky, faktury, dopisy (hledání adresáta) či jinak strukturované a případně ručně psané dokumenty [12].

Výstup OCR je dostačující v případě, kdy požadujeme prostý převod znaku do digitální podoby. Pokud požadujeme důkladnější rozpoznání elementů na stránce, například rozpoznání slov, odstavců nebo obrázků, je nutné navázat důkladnější analýzou ve formě analýzy rozložení.

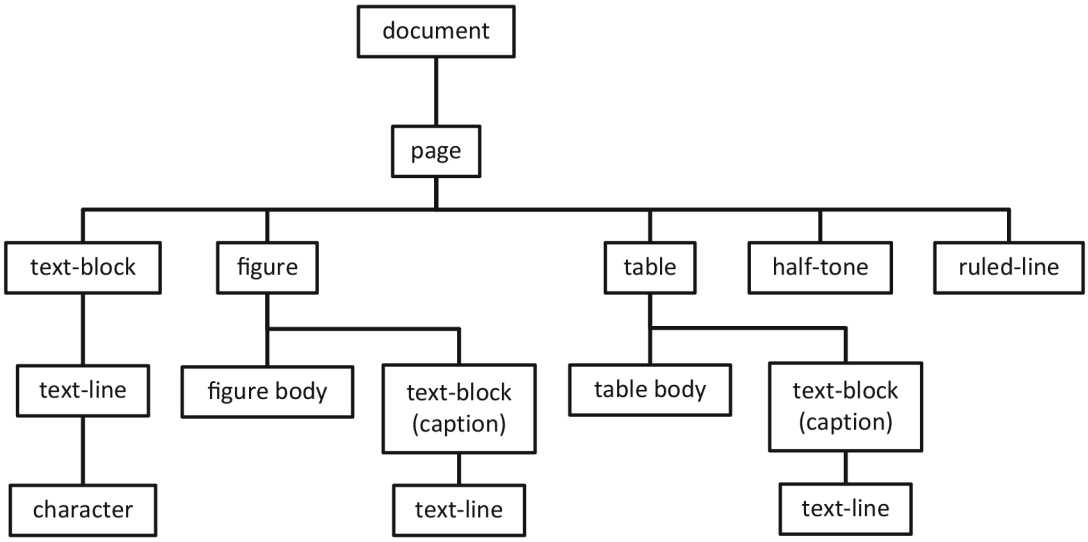
2.2 Analýza rozložení

Cílem analýzy rozložení je identifikovat homogenní komponenty vyskytující se na snímku dokumentu. Homogenní komponentou může být obrázek, graf, odstavec a nebo slovo. Zpravidla mají hierarchickou strukturu, každá komponenta se může skládat z jiných jednodušších komponent. Například odstavec se skládá z vět, věty ze slov a slova z písmen (které byly identifikovány pomocí OCR) [12]. Různé typy komponent a jejich složení jsou zobrazeny na obrázku 2.3. Analýza rozložení se dělí na dvě části: segmentaci stránky a klasifikaci komponent.

2.2.1 Segmentace

Segmentace je proces, který rozděluje stránku na jednotlivé komponenty a identifikuje jejich hierarchickou strukturu. Nezajímá se však o konkretní třídu komponenty. Ta je určena až ve fázi klasifikace (podsekce 2.2.2). Segmentaci dělíme na metody, které jsou založeny na pravidlech a metody využívající strojového učení.

Metody založené na pravidlech používají dvě strategie: Top-down a Bottom-up. Top-down analyzuje a nahlíží na celkovou strukturu dokumentu, kterou se postupnými kroky



Obrázek 2.3: Schéma komponent dokumentu. Převzato z [12].

snaží rozdělit na menší celky (splitting). V kontextu obrázku 2.3 prochází od kořene a jednotlivé komponenty rozděluje na subkomponenty. Bottom-up přistupuje k segmentaci přesně naopak, kdy začíná s triviálními komponentami, například znaky, které dále slučuje (merging) a formuje tak komponenty složitější [12].

Standardním zástupcem strategie Top-down je XY Cut [16]. Jedná se o algoritmus, který postupně dělí komponenty na základě největší mezery mezi nimi, a to jak ve vertikálním směru (Y cut), tak i horizontálním směru (X cut). Postupným dělením vzniká stromová struktura, jejíž listy představují jednotlivé komponenty stránky. Dalšími algoritmy pracující na principu Top-down jsou například algoritmus RLSA (Run-Length Smearing Algorithm) [44] který kóduje vstupní snímek sekvencí binárních hodnot jak ve vertikálním, tak horizontálním směru. Sloučením výsledku z obou směru logickým operátorem AND jsou identifikovány jednotlivé komponenty. Další algoritmy pracují s hranicemi komponent, které jsou definovány pomocí Voroného diagramu [27, 2]. Ty pracují na principu sousednosti, kterou definuje Voroného diagram mezi jednotlivými regiony. Snahou je identifikovat a odstranit ty hrany mezi takovými sousedním regiony, které spolu sémanticky souvisí. Výhodou algoritmů pracující s Voroného diagramy je, že tyto algoritmy jsou invariantní vůči rotaci stránky.

Strategie Bottom-up je založena na spojování komponent, proto jsou jejími zástupci především algoritmy pracující se vzájemnou vzdáleností jednotlivých komponent. Jeden z takových algoritmů pracuje s MST (Minimum Spanning Tree) [25] a s euklidovskou vzdálenosti geometrických středů komponent. Pomocí Kruskalova algoritmu hledá minimální kostru grafu, která spojuje uzly s minimální vzdáleností. Výstupem je stromová struktura, kde jednotlivé podstromy reprezentují komponenty stránky. Ve skutečnosti ne vždy platí, že nejmenší vzdálenost na stránce je mezi komponentami, které spolu sémanticky souvisí. Pro tento problém je optimalizován algoritmus Docstrum [36], který nepracuje pouze s nejmenší vzdáleností, ale definuje shluky komponent pomocí k-nejbližších sousedů (k-nearest neighbors).

Pro segmentaci se dále používají metody strojového učení a neuronových sítí. Autoři práce [43] segmentaci provádí pomocí tří modelů: SVM (Support Vector Machine), GMM

(Gaussian mixture models) a dopřednou neuronovou síť. Z dokumentu extrahují pro každý pixel příznaky v podobě souřadnic, barvy a informace o jeho okolí. Následně je provedena klasifikace všemi třemi modely, které klasifikují do jedné ze čtyř tříd, kterými jsou text, pozadí, okraj stránky a obrázky. Třída je zvolena dle většinového rozhodnutí, případně dle jiné heuristiky, pokud dojde k situaci, že každý klasifikátor predikuje odlišnou třídu.

Další možností je segmentace pomocí moderní konvoluční neuronové sítě, které jsou schopny řešit nejen problém segmentace, ale také klasifikace jednotlivých komponent (nadpis, autor apod.). Vstupem takové neuronové sítě je přímo obrázek dokumentu. Neuronová síť provede extrakci příznaků, na základě kterých se rozhoduje o příslušnosti komponenty do dané třídy. Výstupem takových sítí je často také bounding-box, který definuje hranice dané komponenty [29].

2.2.2 Klasifikace komponent

V průběhu klasifikace je cílem zařadit identifikované komponenty do vhodné třídy. Základní rozdělení definuje textovou a grafickou třídu. Grafická třída se dále dělí na podtřídy zahrnující obrázky nebo geometrické tvary (vertikální nebo horizontální čára). Ke klasifikaci se používají různé klasifikátory, například rozhodovací stromy. Ty o třídě rozhodují na základě různých příznaků, například dle rozměru komponenty, počtu černých pixelů, poměru černobílých přechodů kopírujících pomyslnou vodorovnou linku a podobně. Některé klasifikátory jsou schopny jemnějšího rozlišení komponent a rozšiřují třídy na velký a malý text, matematické zápisy, tabulky, loga a podobně [12].

2.3 Analýza logické struktury

Analýza logické struktury (Logical layout analysis) je proces, který se snaží pochopit sémantiku dat a zpracovat informace vyskytující se na snímku dokumentu. Účel zpracování informací může být různý od pokročilého hledání (například dle autora dokumentu), přes automatickou tvorbu obsahu s odkazy na příslušné obrázky, grafy a tabulky, klasifikaci dokumentů až po jejich celkovou rekonstrukci do digitální podoby. Proces je často nazýván také jako porozumění dokumentu (Document understanding) nebo extrakce informací (Information extraction) [12].

Práce popisuje tři oblasti analýzy logické struktury, kterými jsou kategorizace dokumentů, logická klasifikace komponent a identifikace posloupnosti čtení.

2.3.1 Kategorizace dokumentů

Správná kategorizace dokumentů (Document categorization) je důležitou prerekvizitou pro úspěšnou analýzu struktury a extrakci logických komponent z dokumentu. Například logická klasifikace (sekce 2.3.2) silně závisí na typu dokumentu, u kterého chceme komponenty klasifikovat. Jiné třídy komponent budeme hledat u vědeckých prací, kde nás bude primárně zajímat název, autor, abstrakt, a jiné hledáme u dopisů, kde se můžeme snažit najít odesílatele, respektive příjemce. Také posloupnost čtení je u každého typu dokumentu odlišná. Kategorizace by se měla umět vypořádat se situací, kdy dva dokumenty vypadají rozdílně, ale nesou obdobné nebo stejné informace (například faktury) [12].

Jedním ze způsobů kategorizace dokumentu je klasifikace pomocí struktury XY stromu dokumentu (výstup XY Cut algoritmu, viz sekce 2.2.1) [8]. Výsledná stromová struktura je klasifikována pomocí neuronové sítě, která odhadne příslušnost dokumentu do určité třídy.

2.3.2 Logická klasifikace

Stejně jako se klasifikace komponent v analýze rozložení (sekce 2.2.2) zaobírala zařazením komponenty do určité třídy, i logická klasifikace (Logical labeling) se snaží určit typ komponenty, avšak na pokročilejší úrovni. V rámci logického klasifikace se snažíme zjistit, zda je textová komponenta titulkem, popiskem, zda definuje číslo stránky, abstrakt a podobně. Všechny tyto informace mohou poté sloužit při pokročilém vyhledávání, kdy hledání řetězce omezíme jen na určitý typ komponent [15]. Je jedním z nejběžnějších procesů analýzy logické struktury, protože jeho výstup se používá v dalších procesech, například při určení posloupnosti čtení.

Některé metody využívají specifických pravidel a předpokladů, například že velikost titulku je běžně větší než velikost písma, kterým je psaný obsah, případně je sázený jiným typem písma nebo že číslování se standardně nachází na horním nebo spodním okraji stránky [3].

Některé práce používají ke klasifikaci jazykový model, textový obsah a umístění komponenty v prostoru. Při odhadování třídy pro danou komponentu vyrobí vektor obsahující emebeddingy textové sekvence a souřadnice komponenty. Na základě těchto příznaků přímo odhaduje třídu komponenty [45]. Jiné metody místo jazykového modelu používají dopřednou neuronovou síť, která zpracovává příznaky v podobě souřadnic, morfologických vlastností (tučné písmo, velikost písma a podobně) a sémantických vlastností textu (například zda se jedná o klíčové slovo apod) [40].

Další možností je rozdělení dokumentů na zóny (mřížka $m \times n$), kdy se pro každou zónu, dle jejího obsahu, definuje vektor příznaků [23]. Tyto vektory popisují obsah stránky a jsou vstupem skrytého Markovova modelu, který rozhoduje o třídě dokumentu.

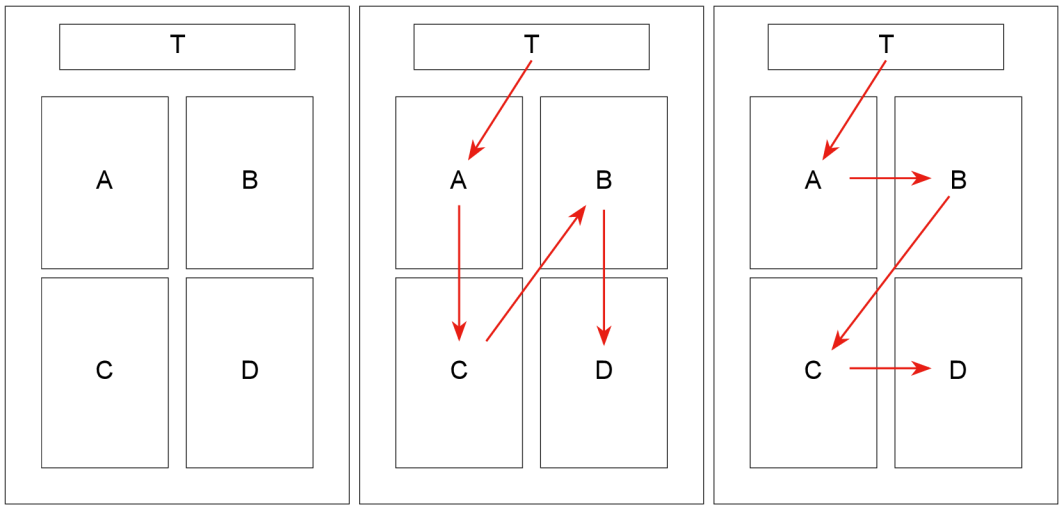
Kategorizace může být založena také čistě na klasifikaci obrázku pomocí konvolučních neuronových sítí [1]. V tomto případě je vstupem černobílý obrázek, ze kterého jsou čtyřmi konvolučními vrstvami extrahovány příznaky. Na konci sítě jsou tři plně propojené vrstvy, z nichž poslední vrstva klasifikuje do deseti tříd (vědecký článek, novinový článek, reklama, formulář a podobně).

2.4 Posloupnost čtení

Posloupnost čtení (Reading order) definuje tok textu v dokumentu. Je jedním z cílů analýzy logické struktury (viz sekce 2.3), v průběhu které se ze snímku dokumentu snažíme získat informace důležité pro další zpracování. Odhad posloupnosti čtení je pro člověka intuitivní záležitostí, která je značnou měrou ovlivněna kulturou. V některých případech však i člověk musí identifikaci věnovat zvýšenou pozornost, protože tok textu nemusí být na první pohled zřejmý. K nalezení posloupnosti často používáme různá vodítka, jako je zarovnání textu, grafické separátory, případně si dopomůžeme jazykovými vlastnostmi textu, či přečtením krátké části a pochopením jejího významu. Tyto aspekty je vhodné zohlednit také při automatickém odvození posloupnosti z obrázku dokumentu.

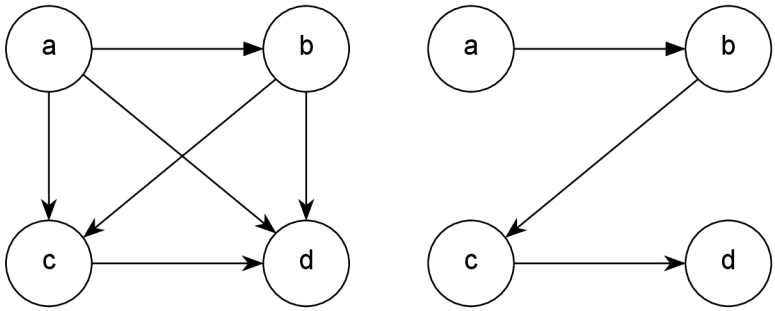
Automatická identifikace je klíčová například při celkové rekonstrukci dokumentu, kdy je zapotřebí zachovat původní tok textu [11]. Zpravidla ji proto odvozujeme pouze pro *textové regiony*, jako jsou odstavce, titulky, případně řádky nebo slova. Zatímco pro jednodušší rozložení (knihy) je automatická identifikace posloupnosti vcelku přímočará, u složitěji strukturovaných dokumentů může být identifikace posloupnosti značně obtížná. V některých případech nelze bez širšího kontextu, například bez jazykové či jiné analýzy, určit, která posloupnost je správná, viz obrázek 2.4. Na stránce může existovat více nezávislých

posloupností. Kupříkladu stránka novin může obsahovat více různých článku, které jsou na sobě vzájemně nezávislé. I když člověk běžně posloupnost čtení jednotlivých článků intuitivně odhadne, nebo ke čtení přistupuje dle zvyklostí, ve skutečnosti lze takové články číst samostatně nehledě na jejich pořadí. Modelovat a identifikovat nezávislé posloupnosti pomocí algoritmů je značně obtížné, proto je běžné, že posloupnost mezi nezávislými články nerozlišují a definují posloupnost i mezi těmito články, například na základě polohy na stránce (článek umístěný výše je v posloupnosti definovaný dříve, než článek umístěný na konci stránky).



(a) Dokument (b) Vertikální posloupnost (c) Horizontální posloupnost

Obrázek 2.4: Znázornění posloupnosti čtení. Posloupnost může být v některých případech nejednoznačná. V případě rozložení stránky 2.4a existují dvě varianty posloupnosti. Zatímco posloupnost 2.4b respektuje směr čtení po sloupcích vertikálně, posloupnost 2.4c udává směr čtení horizontálně.



(a) Complete chain (b) Chain reduction

Obrázek 2.5: Grafické znázornění relace Complete chain a její redukce, jejíž výsledek lze interpretovat jako posloupnost čtení mezi regiony *a*–*d*. Diagram vychází z rozložení uvedené na obrázku 2.4, výsledný Chain reduction respektuje horizontální posloupnost. Pro zjednodušení grafu nebyl uvažován titulek T, který by ve skutečnosti byl zahrnut jak v Complete chain, tak Chain reduction.

Formálně lze posloupnost čtení definovat jako *Complete chain* 2.4.1, respektive jako *Chain reduction* 2.4.2 [30].

Definice 2.4.1 (Complete chain) *Nechť*

- *A je množina textových regionů stránky*
- *D je slabé částečné uspořádání nad A*
- *$B = \{a \in A \mid \exists b \in A : (a, b) \in D \vee (b, a) \in D\}$*

Pokud $D \cup \{(a, a) | a \in B\}$ je úplné uspořádání nad B, pak D je Complete chain nad A.

Complete chain je množina dvojic textových regionů, které v dokumentu mají vztah předchůdce a následníka. Protože definice vychází z uspořádání, lze zachytit některé informace o struktuře dokumentu, například začátek a konec posloupnosti. Pokud budeme uvažovat graf z obrázku 2.5a a dvojice jako hrany grafu, pak pro vrchol, stojící na počátku posloupnosti platí, že neexistuje orientovaná hrana směřující k tomuto vrcholu. Naopak konec posloupnosti charakterizuje vrchol, ze kterého žádná hrana nevychází. Takovými vrcholy jsou na obrázku 2.5a a, respektive d.

Definice 2.4.2 (Chain reduction) *Nechť D je Complete chain nad A. Relace $C = \{(a, b) \in D \mid \neg\exists c \in A : (a, c) \in D \wedge (c, b) \in D\}$ je redukcí D nad A.*

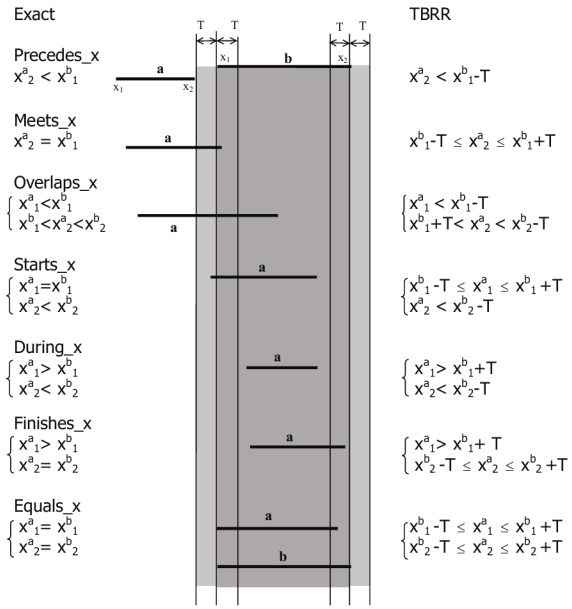
Chain reduction eliminuje takové dvojice z D, které tvoří, vůči jiným, tranzitivní relaci. V kontextu grafu z obrázku 2.5a je vytvořena taková posloupnost vrcholů, kterou můžeme považovat za jednu z možných posloupnosti čtení textových regionů v dokumentu. Uveďme příklad pro rozložení stránky uvedeného na obrázku 2.4c. Příklad pro zjednodušení neuvažuje titulek T.

Nechť $A = \{a, b, c, d\}$. Pokud $D = \{(a, b), (a, c), (a, d), (b, c), (b, d), (c, d)\}$ je *Complete chain* nad A, pak $C = \{(a, b), (b, c), (c, d)\}$ je její redukcí, kterou lze interpretovat jako posloupnost čtení, která je dána pořadím regionů a, b, c, d. Prvky, které tvoří dvojici, jsou sousedními regiony, například (a, b). V takové dvojici je b *bezprostředním následníkem* a. Prvek c je vůči a *prostým následníkem*. Grafické znázornění relace je na obrázku 2.5.

2.4.1 Identifikace posloupnosti

Existují různé přístupy, jak posloupnost čtení identifikovat. Jeden z přístupů je založený na geometrických vztazích s využitím textového obsahu a zpracování pomocí jazykového modelu [3]. Autoři pracují s prostorovými vztahy textových regionů a s jejich sousedností, kterou určují pomocí Voroného diagramu. Definují celkem 13 vztahů, například *precedes* určující, že region A předchází regionu B, *overlaps* který definuje, že region A se s regionem B na dané ose překrývá nebo *equals* určující, že A i B jsou na ose shodně postaveny a podobně. 7 vztahů je standardních, 6 je inverzních, tedy například *i_precedes* je inverzní vztah a říká, že region A následuje až po regionu B. Kompletní výčet vztahů a jejich význam je znázorněn na obrázku 2.6. Pro takto identifikované vztahy identifikují cestu grafem, která představuje posloupnost čtení. Pokud je identifikováno více cest, je s pomocí jazykového modelu, konkrétně s pomocí *part-of-speech-tagging* vypočtena pravděpodobnost jednotlivých posloupností. Pomocí taggeru jsou pro jednotlivá slova identifikovány slovní druhy a další role, které v textu zastupují. Tedy zda se jedná o spojku, sloveso, případně zda se slovo dle interpunkce nachází na konci věty. Následně je pro odhad pravděpodobnosti použit trigramový model. Cesta s nejvyšší pravděpodobností je označena jako konečná posloupnost.

Další metoda naopak užití jazykového modelu odmítá z důvodu ztráty obecnosti (jazykový model je zaměřený na konkrétní jazyk) [14]. Tato metoda navrhuje identifikaci



Obrázek 2.6: Znázornění vztahů možných vztahů regionů A a B a podmínek, které daný vztah musí splňovat. Mezi regiony je v každé ose právě jeden vtah. Výpočet ve sloupci Exact se spoléhá na konkrétní hodnoty souřadnic regionů. Ty mohou být vlivem mnoha faktoru při digitalizaci nepřesné. Proto zavádí prahovou hodnotu T, pomocí které rozšiřuje hranice regionů, viz sloupec TBRR. Převzato z [3].

posloupnosti čtení pomocí jednoduchých pravidel vycházejících z lidského chování. Pro vstupní dokument v prvé řadě identifikuje strukturu rozložení dokumentu, tedy zda se jedná o knihu, noviny a podobně, obdobným způsobem, jako je popsáno v 2.3.1. Z rozložení extrahuje textové regiony a separátory, které vizuálně oddělují bloky textu, například jednotlivé články. Následně definuje a využívá tři základní pravidla, podle kterých sestaví dvojice, které by měly v toku textu následovat po sobě. Prvním pravidlem je kontrola sousednosti textových bloků v horizontálním a vertikálním směru. Druhé pravidlo definuje, že poslední textový blok ve sloupci by měl být následován prvním textovým blokem v sousedním sloupci. Obdobně třetí pravidlo specifikuje, že textový blok umístěný v daném segmentu nejvíce vpravo může být následován textovým blokem nejvíce vlevo na dalším, sousedním řádku. Takto sestavená množina dvojic je analyzována pomocí argumentačního frameworku, který je postavený na nemonotónní logice a konceptu zrušitelných argumentů (protichůdných informací). Protichůdnou informací jsou například dvojice (a, b) a (a, c). První dvojice v kontextu posloupnosti čtení definuje, že b je bezprostředním následníkem a, ale zároveň druhá dvojice říká, že c je bezprostředním následníkem a. Tyto dvojice se argumentační framework snaží eliminovat a tím identifikovat konečnou posloupnost čtení.

Další možností pro definici posloupnosti je využití XY-Cut algoritmu. Jedná se o segmentační algoritmus, pomocí kterého je stránka postupně, dle pomyslného největšího obdélníka, rozdělena horizontálními a vertikálními řezy (více viz sekce 2.2.1). Práce [24] obdobně jako výše uvedené metody definuje pravidla (vzdálenost, sousednost), dle kterých textové regiony zařadí do jednotlivých skupin. Na tyto skupiny je následně aplikován XY-Cut alogoritmus, pomocí kterého je identifikována posloupnost mezi těmito skupinami. Následně je aplikován XY-Cut lokálně v jednotlivých skupinách. Tím je určena posloupnost jednotlivých elementů na lokální úrovní. Sdružením informací z lokální posloupnosti mezi elementy

a globální posloupnosti mezi skupinami je vytvořena konečná posloupnost čtení jednotlivých textových regionů. Obdobným způsobem je algoritmus použit také v práci [31], která navíc upravuje nedostatek XY-Cut. Ten v základní verzi nedokáže provést řez pro elementy tvaru L, čímž může být výsledná posloupnost původní metody zkreslena.

2.4.2 Metriky

Pro měření úspěšnosti identifikované posloupnosti čtení proti ground truth práce pracuje se dvěma metrikami: Recall a Prima. Recall je jednoduchá metrika počítající poměr správně identifikovaných dvojic posloupnosti proti všem dvojicím ground truth. Prima je komplexní metrika, která pracuje s bezprostředními i s prostými následníky a porovnává jejich umístění v posloupnosti.

Recall

Recall je základní metrika, která udává podíl počtu správně identifikovaných dvojic (#Hits) vůči celkovému počtu dvojic v ground truth posloupnosti čtení (#Total). Jednoduchost výpočtu této metriky je také její nevýhodou, protože bere v úvahu pouze bezprostřední následníky a nepočítá s případy správně identifikovaných prostých následníků.

$$\text{Recall} = \frac{\#\text{Hits}}{\#\text{Total}} \tag{2.1}$$

Prima

Prima [11] je komplexní metrika, která kromě bezprostředního a prostého následníka zavádí také pojem *uspořádané* (Ordered) a *neuspořádané* (Unordered) skupiny. Uspořádaná skupina je synonymem pro posloupnost čtení. V takové skupině záleží na pořadí jednotlivých textových regionů. U neuspořádané skupiny naopak na pořadí nezáleží.

Pomocí těchto skupin je možné zachytit, které textové regiony na stránce spolu souvisí a které ne. V případě novinové stránky jsou textové regiony jednoho článku v jedné uspořádané skupině, protože záleží na pořadí čtení těchto regionů. Na pořadí čtení jednotlivých článků na stránce zpravidla nezáleží, tedy alespoň ne přímo, a proto je možné články zařadit do neuspořádané skupiny. Skupiny je do sebe možné různě zanořovat a tím vytvořit stromovou strukturu závislostí.

Metrika nejdříve stanoví pro každý prvek vztah vůči všem ostatním prvkům ve stromové struktuře. Tím vznikne matice vzájemných vztahů mezi jednotlivými prvky. Matice vztahů se sestaví jak pro identifikovanou posloupnost čtení $\mathbf{I}$, tak pro ground truth posloupnost čtení $\mathbf{G}$. Metrika definuje celkem 7 různých vztahů, které ve stromové struktuře identifikuje, viz obrázek 2.7a. Následně jsou porovnány jednotlivé vztahy těchto dvou matic vztahů a uděleny penalizace pomocí chybové matice $\mathbf{M}$, viz obrázek 2.7b. Mějme textové regiony i a j, identifikovaný vztah $x = \mathbf{I}_{i,j}$ a ground truth vztah $y = \mathbf{G}_{i,j}$ těchto regionů. Hodnotu penalizace získáme jako:

$$p = \mathbf{M}_{xy} \tag{2.2}$$

Následně je spočtena celková chyba e, která je sumou hodnot penalizací jednotlivých dvojic k.

$$e = \sum_{k} p_k \tag{2.3}$$

→	Direct successor
←	Direct predecessor
--	Fully unordered relation (e.g. both in same unordered group)
→→	Somewhere before (but unordered group involved)
←←	Somewhere after (but unordered group involved)
-x-	Neither direct nor unordered relation
n.d.	Relation not defined (one or both regions not in reading order tree)

(a) Tabulka vztahů metriky

Identified \ Ground truth	→	←	--	-x-	n.d.	→→	←←
→	0	40	10	20	0	0	10
←	40	0	10	20	0	10	0
--	20	20	0	10	0	10	10
-x-	20	20	10	0	0	10	10
n.d.	20	20	10	0	0	10	10
→→	0	20	5	5	0	0	10
←←	20	0	5	5	0	10	0

(b) Chybová matice metriky

Obrázek 2.7: Tabulka 2.7a představuje vztahy, které jsou v rámci metriky mezi jednotlivými textovými regiony identifikovány. Matice 2.7b definuje velikost chyby identifikovaného vztahu vůči vztahu ground truth. Pokud jsou vztahy stejné, není udělena žádná penalizace. Pokud se vztahy nerovnají, je na základě rozdílnosti vztahů definována velikost penalizace. Převzato, respektive v případě matice inspirováno z [11].

Ta je dále normalizována maximální hodnotou chybové matice p_{max} a počtem textových regionů v ground truth posloupnosti čtení n_{GT} a vyjádřena procentuální hodnotou s vyjadřující celkové skóre.

$$e_{50} = \frac{p_{max} \cdot n_{GT}}{2} \tag{2.4}$$

$$s = \frac{1}{e \cdot \frac{1}{e_{50}} + 1} \tag{2.5}$$

Kapitola 3

Statistické jazykové modelování

Statistické jazykové modelování je obor zabývající tvorbou modelů, které jsou schopné určit pravděpodobnost výskytu sekvence o dané skladbě slov. Pravděpodobnost sekvence slov $w_1 \dots w_i$ je pravděpodobnost společného výskytu těchto slov $P(w_1 \dots w_i)$. Takovou pravděpodobnost je možné rozdělit na jednotlivé složky:

$$P(w_1 \dots w_i) = P(w_1) \cdot P(w_2|w_1) \cdot \dots \cdot P(w_i|w_1 \dots w_{i-1}) \tag{3.1}$$

kde $P(w_1)$ je pravděpodobnost výskytu slova w_1, $P(w_2|w_1)$ je podmíněná pravděpodobnost výskytu slova w_2, a tak dále. Jazykové modely nachází uplatnění v oblastech jako je zpracování řeči, v překladech jazyka nebo v systémech zabývajících se opravou gramatiky a překlepů.

Metrikou pro vyhodnocení jazykového modelu je perplexita. Perplexita se počítá jako geometrický průměr inverzních pravděpodobností jednotlivých slov. Není závislá na úloze, pro jejíž řešení byl jazykový model vytvořený. Proto jí je možné použít pro porovnání jednotlivých modelů. Hodnoty však lze porovnat pouze, pokud byly měřeny na stejné datové sadě. Zpravidla se měří na testovací sadě korpusu. Čím nižší je hodnota perplexity vytvořeného modelu, tím lépe tento model aproximuje chování hypotetického reálného modelu [18].

$$\mathrm{PPL} = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \dots w_{i-1})}} \tag{3.2}$$

3.1 Modely založené na četnostech

Při výpočtu pravděpodobnosti $P(w_i|w_1 \dots w_{i-1})$ získáváme pro velká i méně spolehlivé odhady [33]. Modely založené na četnostech proto takovou pravděpodobnost aproximují pomocí n-gramových modelu, kde n představuje počet slov sekvence se kterou model pracuje. Pro ilustraci je dále v textu uvažován model pracující se třemi slovy, který nazýváme jako *trigram*. Trigram odhaduje pravděpodobnost následujícího slova na základě dvou předchozích slov:

$$P(w_i|w_1 \dots w_{i-1}) \approx P(w_i|w_{i-2}w_{i-1}) \tag{3.3}$$

Pokud model odhaduje pravděpodobnost následujícího slova na základě jednoho předchozího, pracuje takový model s *bigramy*, $P(w_i|w_{i-1})$. S *unigramy* model pracuje pokud odhaduje pravděpodobnost slova nezávisle na předešlých slovech, $P(w_i)$.

N-gramové modely využívají statistiky frekvencí společného výskytu n-gramů v trénovací sadě. Pokud $C(w_{i-2}w_{i-1}w_i)$ reprezentuje četnost sekvence $w_{i-2}w_{i-1}w_i$ a obdobně pokud $C(w_{i-2}w_{i-1})$ reprezentuje četnost sekvence $w_{i-2}w_{i-1}$, pak lze pravděpodobnost aproximovat jako [18]:

$$P(w_i|w_{i-2}w_{i-1}) \approx \frac{C(w_{i-2}w_{i-1}w_i)}{C(w_{i-2}w_{i-1})} \tag{3.4}$$

Problémem modelů založených na četnosti je, že sady, ze kterých jsou statistiky četností počítány, pokrývají konečný prostor. Pokud odhadujeme pravděpodobnost sekvence slov, která se nevyskytuje trénovací sadě modelu, je takové sekvenci přiřazená nulová pravděpodobnost výskytu. Tento problém se prohlubuje tím více, čím bohatší je morfologie jazyka, který modelujeme (slovanské jazyky, arabština) [34]. Proto se u n-gramových modelů používají techniky vyhlazení, *smoothing* [10].

Smoothing jsou techniky, pomocí kterých se snažíme eliminovat nulové pravděpodobnosti výskytu sekvence v trénovací sadě. Základní technikou je Laplaceho smoothing (aditivní smoothing), který k výskytu každé sekvence přičítá jedničku. Pravděpodobnost je pak aproximována jako:

$$P(w_i|w_1 \dots w_{i-1}) \approx \frac{C(w_{i-2}w_{i-1}w_i) + 1}{C(w_{i-2}w_{i-1}) + |V|} \tag{3.5}$$

kde $|V|$ je velikost slovníku. Problémem této techniky je v přiřazení vysoké pravděpodobnosti neznámým n-gramům, jejichž výskyt je ve skutečnosti nepravděpodobný. Proto se užívají další techniky jako například Lambda smoothing (stejný princip, místo jedničky přičítá konstantu značenou jako λ), interpolace (odhad pravděpodobnosti je určen kombinací n-gramových modelů zpracovávající různé délky sekvence, například unigramu a trigramu), caching (vychází z předpokladu, že jednou užité slovo bude v blízké budoucnosti použité znovu), skipping (místo $P(w_i|w_{i-2}w_{i-1})$ počítá pravděpodobnost jako $P(w_i|w_{i-3}w_{i-2})$) a další techniky [18, 33]. Limity n-gramových modelů v podobě omezené historie při odhadu pravděpodobnosti je možné řešit pomocí neuronových jazykových modelů.

3.2 Neurální jazykové modely

Neuronové sítě jsou jednou z technik strojového učení. Jejich pomocí se snažíme modelovat a řešit problémy reálného světa od zpracování zvukových nahrávek (například přepis řeči), přes zpracování obrazu (detekce objektů, identifikace osob v obraze apod.) až po zpracování textu a jazykové modelování.

Základní složkou neuronových sítí je neuron. Každý neuron počítá skalární funkci nad svým multidimenzionálním vstupem:

$$y = g(\mathbf{w}^T\mathbf{x} + b) \tag{3.6}$$

kde $\mathbf{x}$ představuje vstupní vektor a $\mathbf{w}$ je vektor vah neuronu, b je bias a g je aktivační funkce (některá z nelineárních spojitých nebo skokových funkcí, například hyperbolický tangens, sigmoida). Neurony jsou seskupeny do vrstev, které jsou vzájemně propojeny. Protože v rámci jedné vrstvy provádí každý neuron j stejnou operaci $y_j = g(\mathbf{w}_j^T\mathbf{x}_j + b_j)$, můžeme celou vrstvu uvažovat jako vektor těchto neuronů $\mathbf{h}_i$ vrstvy i. Potom lze výpočet jedné vrstvy definovat jako:

$$\mathbf{h}_{i+1} = g(\mathbf{W}\mathbf{h}_i + \mathbf{b}) \tag{3.7}$$

kde $\mathbf{W}$ je matice vah (vektory vah jednotlivých neuronů) vrstvy i a $\mathbf{b}$ je vektor biasů. Aktivační funkce g je aplikována po jednotlivých složkách. Jednotlivé architektury neuronových sítí se ve zpracování a výpočtech zpravidla odlišují.

Váhy a biasy jsou pro správnou aproximaci požadované funkce klíčové. Jejich hodnoty se proto upravují v průběhu trénování neuronové sítě, při kterém jsou síti předávána trénovací data, ze kterých je počítán výstup sítě. Výstup sítě je porovnán s požadovaným výsledkem pomocí objektivní funkce (někdy také loss funkce, chybová funkce), která hodnotí rozdíl mezi výstupem a požadovanou hodnotou (určuje velikost chyby). Příkladem takové objektivní funkce je střední kvadratická chyba:

$$E = \frac{1}{n}\sum_{i=1}^{n}(t_i - o_i)^2 \tag{3.8}$$

kde n je celkový počet prvků trénovacího datasetu (případně dávky), t_i je požadovaná hodnota a o_i je aktuální výstup sítě pro prvek i. Cílem trénování je upravit váhy neuronů tak, aby výstup sítě vykazoval proti očekávaným výsledkům co nejmenší chybu (minimalizace hodnoty objektivní funkce) [17]. Váhy jsou upravovány iterativně pomocí algoritmu Back-Propagation.

Back-propagation je algoritmus, pomocí kterého je vypočtena hodnota gradientu parametrů sítě vůči hodnotě chybové funkce, značeno jako $\nabla E(\mathbf{w})$. Hodnota gradientu vůči parametrům sítě $\nabla E(\mathbf{w})$ udává, jak hodnoty vah $\mathbf{w}$ ovlivňují výstup sítě, respektive jaký mají vliv na celkovou chybu. Algoritmus počítá gradient od výstupu sítě směrem k jejímu vstupu. Při výpočtu gradientů jednotlivých vrstev využívá řetězcového pravidla, při kterém používá dříve spočtené parciální derivace z předchozích vrstev [17].

Po vypočtení gradientu pro všechny parametry jsou tyto parametry, na základě jejich gradientu, upraveny. O to se stará optimalizační algoritmus, například SGD [7]. Ten upravuje parametry odečtením hodnot gradientu s učícím krokem η (learning rate):

$$\mathbf{w} = \mathbf{w} - \eta\nabla E(\mathbf{w}) \tag{3.9}$$

Dalším optimalizačním algoritmem je RMSProp [20]. Ten při úpravě vah využívá nejen aktuální hodnoty gradientu, ale také průměrnou energii gradientů z minulých iterací. Při úpravě parametrů sítě je aktuální gradient touto průměrnou hodnotou normalizován. Algoritmus Adam [26] využívá podobného mechanismu jako RMSProp, ale do výpočtu navíc zavádí setrvačnost, která vyjadřuje průměrnou rychlost změny gradientu.

Trénování probíhá v jednotlivých iteracích, které nazýváme *epochy*. Počet epoch nemusí být přesně stanoven, zpravidla je síť trénována, dokud po dokončení epochy vykazuje zlepšení oproti předchozí. V rámci jedné epochy je vždy zpracován celý trénovací dataset. K výpočtu chyby, gradientů a úpravě vah lze přistoupit dvěma způsoby; pomocí Gradient Descent (GD) nebo pomocí Stochastic/Mini-batch Gradient Descent [41]. Gradient Descent počítá chybu sítě a upravuje váhy až poté, co je sítí zpracován celý trénovací dataset, tedy až na konci epochy. Protože je trénovací dataset zpravidla značně rozsáhlý, je výpočet velmi náročný a pomalý.

Z toho důvodu častěji používáme Stochastic/Mini-batch Gradient Descent, který oproti GD trénovací dataset rozdělí na menší části o určité velikosti, kterým říká *dávka* (mini-batch). Každá dávka obsahuje náhodný vzorek dat z trénovací množiny. Výpočet chyby a úprava vah sítě probíhá po každé zpracované dávce. Stále platí, že během jedné epochy je zpracován celý trénovací dataset, tedy v případě Mini-batch Gradient Descent je zpracováno několik dávek. K výpočtu chyby a úpravě vah dojde v průběhu epochy hned několikrát. To

má za následek určité zkreslení hodnot gradientu, avšak metoda je oproti Gradient Descent efektivnější.

V oblasti statistického jazykového modelování lze použít různé architektury neuronových sítí. Tato práce popisuje dvě z nich; dopředné neuronové sítě a rekurentní neuronové sítě.

3.2.1 Dopředné neuronové sítě

Statistické jazykové modelování pomocí dopředných neuronových sítí (FFNN) je v některých ohledech podobné n-gramovému modelování. Model neuronové sítě pracuje stejně jako n-gramový model s omezenou délkou sekvence na vstupu. Výstup v podobě rozdělení pravděpodobností však není ovlivněn pouze četností výskytu slov v trénovacím datasetu, ale také jazykovými vlastnostmi jako je syntaxe, sémantika a podobnost slov (dvojice slov auto; autobus je si významem více podobná, než dvojice auto; květina) [5]. Dopředná neuronová síť použitá jako jazykový model, je vyobrazena na obrázku 3.1. Lze ji definovat jako funkci f zpracovávající vstupní vektor $\mathbf{x}$ na základě parametrů $\boldsymbol{\theta}$ (parametry, které se učí) [17]:

$$\mathbf{y} = f_{\boldsymbol{\theta}}(\mathbf{x}) \tag{3.10}$$

Vstupem dopředné neuronové sítě je sekvence slov, pro která je počítaná pravděpodobnost. Jednotlivá slova mohou být reprezentována pomocí one-hot kódování, kdy je každé slovo slovníku V reprezentováno vektorem o velikosti $|V|$ obsahující nuly, pouze na místě indexu slova se nachází jednička, případně je dané slovo reprezentováno číslem vyjadřující pozici slova ve slovníku.

Jednotlivá slova se posléze převádí do n-dimenzionálního spojitého prostoru pomocí projekce C. Každému slovu je přiřazen vektor reálných čísel o velikosti n. Pomocí projekce se síť snaží zachytit a naučit podobnost slov. Podobná slova pak mají podobné vektory, respektive se v n-dimenzionálním prostoru nachází blízko sebe. V konečném důsledku se neodhaduje pravděpodobnost P na základě sekvence slov, ale na základě jejich příznaků [5], tedy:

$$P(w_i|w_1 \dots w_{i-1}) = P(w_i|C(w_1) \dots C(w_{i-1})) \tag{3.11}$$

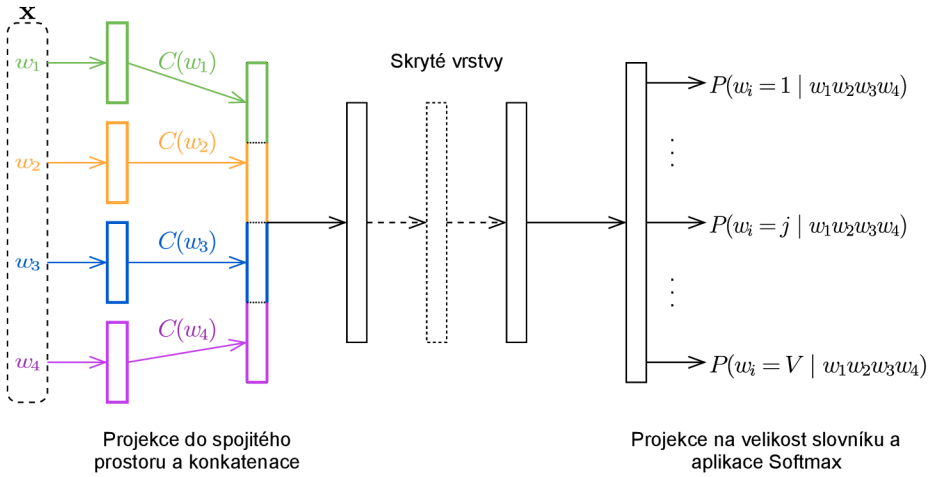
Projekce C je zpravidla jednou z vrstev (standardně vstupní vrstvou) neuronové sítě. Hodnoty vektoru slova se v průběhu trénování sítě upravují tak, aby sémanticky podobná slova spolu v n-rozměrném prostoru sousedila. Pro převod je také možné použít samostatné předtrénované modely jako například Word2Vec [32] nebo GloVe [39].

Ostatní (skryté) vrstvy dopředné neuronové sítě jsou standardně plně propojené lineární vrstvy. Počet skrytých vrstev definuje hloubku sítě, která může mít různý vliv na úspěšnost modelu, stejně jako mají vliv různé velikosti vektoru příznaků [4]. Počet neuronů výstupní vrstvy odpovídá počtu slov slovníku V (po provedení projekce). Aplikováním funkce Softmax získáme na výstupu sítě distribuci pravděpodobnosti přes všechna slova ve slovníku.

$$\text{softmax}\left(\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_V \end{bmatrix}\right) = \frac{1}{\sum_{i=1}^{V} \exp(a_i)} \begin{bmatrix} \exp(a_1) \\ \exp(a_2) \\ \vdots \\ \exp(a_V) \end{bmatrix} \tag{3.12}$$

Slabinou dopředných neuronových sítí je omezení délky sekvence, která je předkládána na vstup modelu. Rozhodnutí o velikosti vstupu musí být navíc definováno již při návrhu

sítě před jejím trénováním, velikost zpracovávaného kontextu je poté pevně daná. Tento problém se snaží řešit rekurentní neuronové sítě, které se díky přenosu hodnot skrytých vrstev dokážou učit vlastnosti i mezi jednotlivými vstupy a tím prodloužit kontext, ze kterého odhadují pravděpodobnost.



Obrázek 3.1: Schéma architektury dopředného neurálního jazykového modelu. Vstupem je sekvence slov, které jsou na vstupu pomocí projekce převedeny na vektory v n-rozměrném prostoru. Tyto vektory jsou zkonkatenovány a výsledná struktura je dále zpracována skrytými vrstvami sítě. Výstupní vrstva provádí projekci na velikost slovníku a aplikuje funkci Softmax. Výstup pak lze interpretovat jako rozdělení pravděpodobnosti přes všechna slova slovníku. Konkrétně se jedná o podmíněnou pravděpodobnost slova w_i, pokud jeho prefixem je sekvence $w_1 w_2 w_3 w_4$.

3.2.2 Rekurentní neuronové sítě

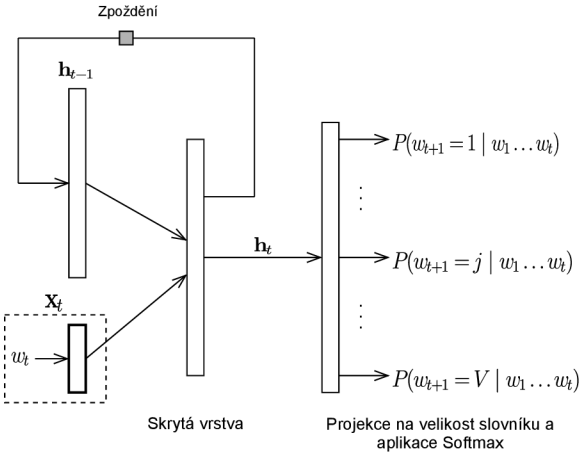
Rekurentní neuronové sítě (RNN) jsou sítě specializující se na zpracování sekvenčních dat o variabilní délce [17]. Klíčovým prvkem jsou rekurentní spoje, pomocí kterých je možné uchovat informace předešlých vstupu ve vnitřním stavu sítě, jenž posléze ovlivňuje výstup [19]. Ve srovnání s dopřednou sítí (rovnice 3.10) je výstup rekurentní sítě $\mathbf{h}_t$ v čase t závislý nejen na aktuálním vstupu $\mathbf{x}_t$, ale také na vstupech předešlých, které jsou zastoupeny ve stavu sítě $\mathbf{h}_{t-1}$ [17]:

$$\mathbf{h}_t = f_{\boldsymbol{\theta}}(\mathbf{h}_{t-1}, \mathbf{x}_t) \tag{3.13}$$

Hodnota $\mathbf{h_t}$ představuje jak výstup, tak také skryté stavy sítě, které je možné použít v čase $t+1$ (rekurence), $\boldsymbol{\theta}$ jsou parametry sítě (váhy, biasy, word-embedding), natrénované a nastavené tak, aby síť aproximovala požadovanou funkci. Vektor $\mathbf{x_t}$ lze v souvislosti jazykového modelování chápat jako vektor příznaků slova w_t, tedy například $\mathbf{x_t} = C(w_t)$, vektor $\mathbf{h_{t-1}}$, respektive $\mathbf{h_t}$ pak představují kontext, na základě kterého odhadujeme pravděpodobnost slova w_{t+1}:

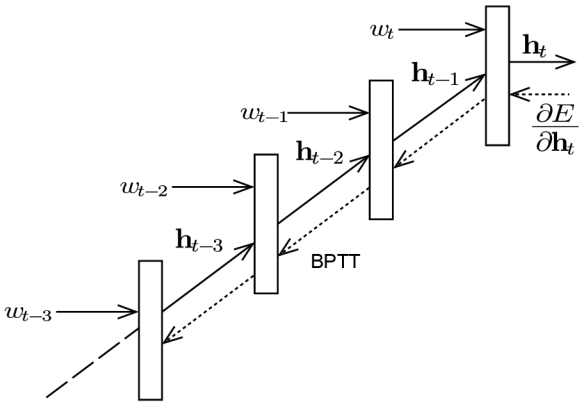
$$P(w_{t+1}|w_1 \dots w_t) = P(w_{t+1}|\mathbf{h}_t) \tag{3.14}$$

Projekcí na $\mathbf{h_t}$ je výstup převeden na velikost slovníku V Aplikováním funkce Softmax, lze na výsledek nahlížet jako na rozdělení pravděpodobností přes všechna slova slovníku V, viz obrázek 3.2.



Obrázek 3.2: Schéma architektury rekurentní neuronové sítě. Vstupem je kromě w_t také vektor $\mathbf{h_{t-1}}$ představující skryté stavy z předešlého kroku. Tento vektor tvoří kontext předešlých vstupů, které ovlivňují aktuální výstup $\mathbf{h_t}$. Projekcí na velikost slovníku a aplikováním funkce Softmax získáme rozdělení pravděpodobnosti.

Váhy sítě jsou upravovány pomocí algoritmu BPTT (Back-Propagation Trough Time, propagace chyby v čase). Ten při výpočtu gradientu a úpravě vah provádí rozvinutí výpočetního grafu v čase (unfolding). To umožňuje upravovat váhy zpětně napříč historií vstupu obdobným způsobem jako v případě algoritmu Back-Propagation [6]. Využití rekurentního spojení a rozvinutí v čase je vyobrazeno na obrázku 3.3.



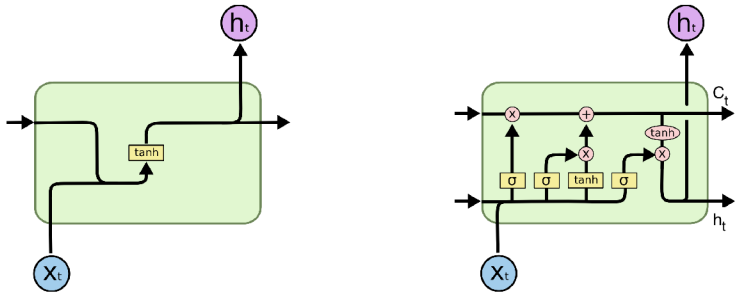
Obrázek 3.3: Rozvinutí výpočetního grafu rekurentní sítě pro výpočet gradientu pomocí BPTT. Šrafované šipky znázorňují směr výpočtu gradientu.

Hlavním problémem při výpočtu gradientu do vzdálené historie, je vyšumění gradientu (gradient vanishing) nebo jeho exploze (gradient explode) [17]. Při vyšumění dochází k tomu, že informace, která je v čase vzdálená oproti aktuálnímu vstupu má mnohem menší vliv na výstup sítě, než informace zachycena poměrně nedávno a obdobně se tento efekt projevuje při zpětné úpravě vah sítě. Při explozi gradientu naopak dochází k exponenciálnímu růstu, což má za následek nestabilitu sítě. Taková síť se pak není schopna učit závislosti v dlouhých sekvencích. Paměť sítě je tak omezena na několik předchozích kroků (krátko-

dobá paměť). Tyto problémy řeší speciální architektury rekurentních sítí, například Long Short-Term Memory, LSTM.

3.2.3 LSTM

LSTM (Long Short-Term Memory) je typ rekurentní sítě, která řeší problém vyšumění a exploze gradientu, který se projevuje u standardních rekurentních sítí [21]. LSTM umožňuje uchovávat a přenášet informace v delším časovém horizontu, a to díky speciálně navržené struktuře výpočetních jednotek. Ty zpracovávají a napříč sítí přenáší dvě hodnoty; stav sítě $\mathbf{h}_t$ (obdobný význam jako u RNN) a stav buňky $\mathbf{C}_t$. Právě stav buňky $\mathbf{C}_t$ je důležitým prvkem LSTM, skrze který se šíří informace na delší vzdálenosti. V průběhu trénování se buňka učí, jak tyto informace regulovat, od drobných úprav až po úplné zapomínání [37]. Struktura buněk RNN i LSTM je srovnána na obrázku 3.4.



(a) RNN (b) LSTM

Obrázek 3.4: Schéma buněk RNN a LSTM. Zatímco RNN 3.4a aplikuje jednu operaci (například hyperbolický tangens), struktura LSTM 3.4b je složitější a aplikuje několik operací. Důležitý je především stav buňky $\mathbf{C}_t$ (horní kanál), skrze který lze přenášet informace v delším časovém horizontu. Převzato z [37].

3.2.4 Tokenizace

Před zpracováním přirozeného jazyka je nutné provést transformaci vstupních dat, v případě této práce textového korpusu, do podoby, které jsou jazykové modely schopné porozumět. Této transformaci říkáme tokenizace. V průběhu tokenizace probíhá zpracování korpusu dat (nejčastěji trénovací množiny) a jeho rozdělení na menší jednotky, které nazýváme tokeny. Tokeny jsou základními prvky, jejichž vzájemné vztahy se jazykové modely učí. V průběhu tokenizace vzniká slovník, který obsahuje jednotlivé, unikátní tokeny. Pozici tokenu ve slovníku pak lze použít jako vstup jazykového modelu. Tokenizaci lze rozdělit na tři typy [38]; tokenizace slov (word tokenization), tokenizace znaků (character tokenization) a tokenizace podslov (subword tokenization). Příklady jednotlivých metod jsou uvedeny v tabulce 3.1.

Tokenizace slov vytváří tokeny dělením korpusu dle bílých znaků, například mezer. Tím dojde k separaci jednotlivých slov korpusu. Velikost vzniklého slovníku roste s velikostí korpusu a především u flektivních[1] jazyků může být velikost enormní. Zároveň je nutné umět oddělit a zpracovat tečky, čárky a jiná interpunkční znaménka, která zpravidla nejdou jednoduše separovat pomocí bílých znaků, protože bývají přímo spojená se slovem.

[1] Jazyky s funkcemi časování a skloňování. Mezi tyto jazyky se řadí čeština nebo ruština.

Tokenizace znaků dělí korpus na jednotlivé symboly. Výhodou takové tokenizace je omezená a velmi malá velikost slovníku v řádu desítek (latinka) až nízkých tisíců (Kandži, systém pro zápis japonštiny) znaků. Díky tomu lze s vytvořeným slovníkem snadno formovat sekvence, které se v trénovací množině nevyskytovaly. Nevýhodou takového rozdělení je, že nepracujeme se slovy, ale se znaky. U rekurentních sítí pak nezachytáváme kontext po sobě jdoucích slov, ale zpracováváme kontext v podobě znaků. To v případě jazykového modelování znamená, že výsledek sítě je ovlivněn několika předcházejícími znaky, namísto několika slovy a tím je tedy zachycen mnohem kratší kontext.

Tokenizace podslov provádí dělení jednotlivých slov na podslova, například dle jejich frekvence výskytu v korpusu. Výhodou tohoto přístupu je možnost definovat požadovanou velikost výstupního slovníku. V průběhu trénování subwordového modelu je slovník redukován tak dlouho, dokud není dosaženo požadované velikosti (například dalším dělením podslov na menší jednotky). Z vytvořených podslov lze jejich opětovným skládáním tvořit i taková slova, která se ve vstupním korpusu nevyskytovala.

Jedním z nástrojů umožňující vytvořit slovník metodou tokenizace podslov je nástroj SentencePiece [28]. Jedná se o jazykově nezávislý tokenizér (zpracuje jak češtinu, tak japonštinu), který je schopný natrénovat subword model ze vstupních dat a se slovníkem o požadované velikosti. Skládá se ze 4 komponent; normalizer, trainer, encoder a decoder. Normalizer převádí a normalizuje sémanticky ekvivalentní Unicode znaky do kanonické formy. Trainer trénuje z normalizovaného korpusu subword model o požadované velikosti. Encoder provádí normalizaci textové sekvence, kterou pomocí vytvořeného subword modelu převádí na tokeny. Tokeny jsou v tomto případě číselné hodnoty reprezentující pozici jednotlivých podslov ve slovníku. Decoder provádí převod seznamu tokenů do textové reprezentace.

SentencePiece byl v rámci této práce použit pro vytvoření vhodného subword modelu na jehož základě byl následně postaven a trénován LSTM jazykový model. Pomocí Traineru byl natrénovat subword model se slovníkem o velikosti 20K podslov. V rámci práce je dále využit Encoder, pomocí kterého jsou převáděny vstupní textové sekvence na odpovídající seznam tokenů pro účely trénování jazykového modelu a vyhodnocení pravděpodobnosti sekvence tímto modelem. Decoder naopak převádí tokeny zpět na textovou reprezentaci. Decoder není v této práci explicitně použit, uplatnění by našel například v generování textu při automatické sumarizaci.

Tabulka 3.1: Příklad tokenizace sekvence 'třista třicet tři' jednotlivými metodami. Slovník obsahuje jednotlivé prvky z trénovací množiny, jejich podoba je odvislá od zvolené metody tokenizace. Kódování znázorňuje převod sekvence na odpovídající vektor. Jednotlivé hodnoty vektoru představují pozici prvku ve slovníku. Symbolem '␣' je znázorněn konec slova. Slovník může ve skutečnosti obsahovat i další symboly, například začátek a konec věty, symbol zastupující neznámý token a podobně. Tyto nejsou v příkladu uvedeny.

	Tokenizace	Slovník	Kódování
Tokenizace slov	třista␣třicet␣tři␣	[třista␣, třicet␣, tři␣]	[1,2,3]
Tokenizace znaků	t_ř_i_s_t_a␣t...	[t,ř,i,s,a␣,c,e,t␣,i␣]	[1,2,3,4,1,5,1,2,3,...]
Tokenizace podslov	tři_sta␣tři_cet␣tři␣	[tři, sta␣, cet␣, tři␣]	[1,2,1,3,4]

Kapitola 4

Návrh analýzy posloupnosti čtení

Cílem této práce je navrhnout a experimentálně ověřit úspěšnost jazykového modelu při identifikaci posloupnosti čtení. V této kapitole je popsaný zvolený postup a na něj navazující návrh systému.

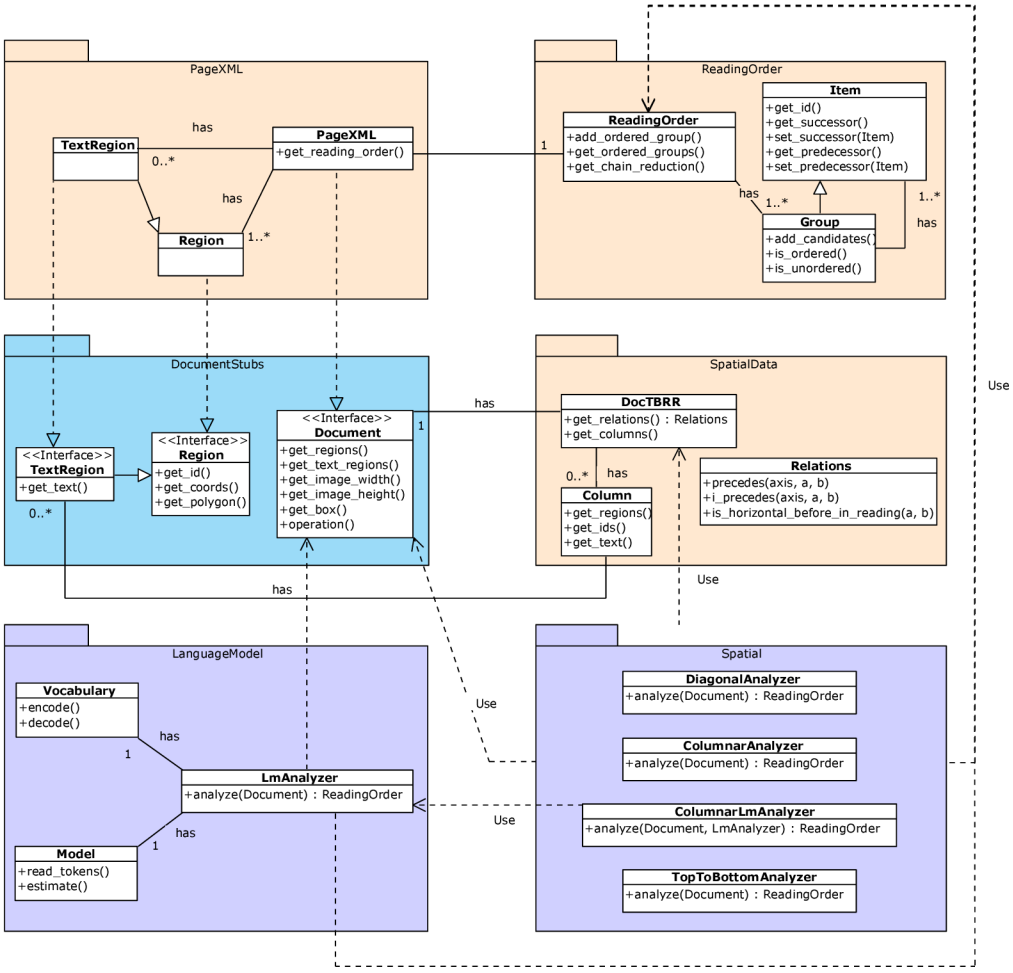
Jazykový model, použitý v této práci, bude využívat architektury LSTM, se slovníkem vytvořený pomocí nástroje SentencePiece. Výstup jazykového modelu bude interpretován jako rozdělení pravděpodobnosti přes všechny tokeny slovníku.

Z hlediska funkcionality budou uvažovány tři přístupy. V první řadě jde o *jazykovou analýzu*, která definuje posloupnosti čtení na základě textového obsahu regionů analyzované stránky. Jazyková analýza provede, pomocí jazykového modelu, odhad pravděpodobnosti návaznosti jednotlivých textových regionů. To provede tak, že jeden z textových regionů použije k inicializaci skrytých stavů jazykového modelu a tyto skryté stavy následně použije pro odhad pravděpodobnosti ostatních textových regionů. Takový postup je možné interpretovat jako výpočet podmíněné pravděpodobnosti, kde textová sekvence regionu použitého k inicializaci skrytých stavů I představuje prefix pro region C, pro který pravděpodobnost odhadujeme, $P(C|I)$. Jazyková analýza provede odhad pravděpodobnosti pro všechny regiony. Ty dva regiony, pro které bude jejich podmíněná pravděpodobnost nejvyšší, budou v posloupnosti tvořit dvojici (více viz sekce 2.4). Jazyková analýza zpracovává dokument tak dlouho, dokud nejsou všechny prvky zařazeny v posloupnosti.

Pro srovnání bude implementována *prostorová analýza*. Ta, na rozdíl od jazykové analýzy, pracuje pouze s prostorovými informacemi regionů vyskytujících se na stránce. Na základě vzájemných vlastností a vztahů v podobě vzdálenosti, sousednosti, přímé viditelnosti a podobně bude mezi textovými regiony definována posloupnost čtení. Identifikace vztahů bude vycházet z článku [3], který je popsaný v sekci 2.4.1.

Využívat výhod, a naopak usilovat o eliminaci případných nevýhod obou metod bude *kombinovaná analýzy*. Jak z názvu vyplývá, metoda kombinuje prvky jazykové a prostorové analýzy. Metoda tak pracuje jak s prostorovými daty, tak textovými informacemi s pomocí jazykového modelu.

Pro podrobnější nástin funkcionality jsem vytvořil objektový návrh, který jsem posléze implementoval v jazyce Python. Implementovány byly struktury a metody zpracovávající vstupní data, metriky pro měření úspěšnosti, a nakonec také jazyková, prostorová a kombinovaná analýza.



Obrázek 4.1: Diagram tříd s rozdělením do jednotlivých balíčků. Modrou barvou je zobrazen balíček obsahující rozhraní pro práci se vstupními daty. Tato rozhraní implementují třídy v balíčku `PageXML` a ke své práci jej využívají jednotlivé analyzátory. Fialovou barvou jsou znázorněny balíčky, které tyto analyzátory obsahují. Balíček `LanguageModel` obsahuje analyzátor pracující na principu jazykové analýzy. Balíček `Spatial` obsahuje analyzátory pracující s prostorovými údaji regionů.

4.1 Návrh systému

Tato sekce popisuje návrh systému z hlediska objektového návrhu. Na obrázku 4.1 je konceptuální diagram tříd, které jsou v programu pro analýzu posloupnosti použity. V balíčku `DocumentStubs` jsou umístěna rozhraní, která definují metody pro zpracování vstupních dat. Tato rozhraní jsou implementovány třídami v balíčku `PageXML`, které mají na starost zpracování konkrétní struktury dat ve formátu xml. Zároveň jsou rozhraní z `DocumentStubs` použita napříč systémem tak, aby implementace analýz byla na konkrétní struktuře dat nezávislá. Zásluhou této abstrakce je možné rozšířit podporu vstupních dat o další formáty. V prostředí jazyka Python budou rozhraní implementovány jako třídy definující metody.

Balíček `ReadingOrder` sdružuje třídy pro zpracování posloupnosti čtení, a to jak identifikované programem, tak definované jako ground truth. Balíček `LanguageModel` obsahuje kromě samotného jazykového modelu a jeho slovníku také jazykový analyzátor. Ten ze

vstupních dat pomocí jazykového modelu identifikuje posloupnost čtení na základě podmíněné pravděpodobnosti dvou sekvencí, *inicializační sekvence* I a *kandidáta* C.

Balíček `Spatial` shromažďuje analyzátory využívající prostorových vlastností dat. K tomuto účelu využívají třídy z balíčku `SpatialData`. Ty obsahují metody pro identifikaci prostorových vztahů mezi regiony, na základě kterých je identifikovaná posloupnost čtení. Chování jednotlivých tříd je rozebráno v samostatných sekcích.

4.2 Vstupní data

Základní strukturou pro zpracování vstupních dat budou v této práci xml soubory, konkrétně PageXML[1]. PageXML je standardizovaný xml formát. Xml soubory, respektující tento formát, obsahují veškeré informace o digitalizovaném dokumentu, včetně OCR výstupu, informace o umístění regionů, typu regionu a dalších anotací. Struktura rozlišuje několik typů regionů jako například textový region, obrázek, separátor a podobně. Umístění v prostoru je pro každý region definováno pomocí souřadnic polygonu, které tvoří hranice daného regionu. Výhodou PageXML je podpora pro uložení struktury posloupnosti čtení. Ta umožňuje zachytit jak uspořádané, tak neuspořádané skupiny (viz sekce 2.4.2, ve které jsou tyto skupiny více popsané). Jeden PageXML soubor obsahuje právě jednu digitalizovanou, anotovanou stránku.

Návrh pro analýzu posloupnosti čtení nabízí objektovou obálku, pomocí které pracuje nad vstupní strukturou. K tomuto účelu definuje několik tříd nabízející metody pro zpracování vstupních dat.

Třída Document

Stěžejní třída definující základní metody pro práci se vstupními daty je třída `Document`. Její struktura je použita především jako rozhraní, těla jednotlivých metod je nutné implementovat třídou, která `Document` dědí. Instance této třídy je vstupem jednotlivých analýz pro identifikaci posloupnosti. Ty jsou díky abstrakci nezávislé na konkrétní vstupní struktuře dat. Program proto bude možné jednoduše rozšířit o podporu různých formátů vstupních dat.

```
1 class Document(object):
2     def get_regions(self) -> {Region}: ...
3     def get_text_regions(self) -> {TextRegion}: ...
4     def get_box(self) -> Polygon: ...
5     ...
```

Třída PageXML

Třída `PageXML` dědí `Document` a implementuje její metody pro zpracování konkrétní struktury dat, v tomto případě xml souboru sledující strukturu PageXML. Kromě toho implementuje metodu `get_reading_order()`, která vrací instanci třídy `ReadingOrder` s anotovanou posloupností čtení. Současně jsou definovány třídy pro zpracování regionů z PageXML struktury, které dědí třídy `Region`, respektive `TextRegion`.

```
1 class PageXML(Document):
2     def get_reading_order(self) -> ReadingOrder: ...
```

[1] https://www.primaresearch.org/schema/PAGE/gts/pagecontent/2019-07-15/

Třída Region

Třída `Region` je základní jednotkou, která pokrývá jednotlivé objekty stránky, ať už se jedná o objekty textové nebo grafické. Obdobně jako třída `Document`, také tato třída je rozhraním, které definuje metody nutné k implementaci. Poskytuje metody určené pro orientaci v prostoru. Obecnou třídu `Region` rozšiřuje třída `TextRegion`, která navíc poskytuje metodu pro načtení textového obsahu regionu. Instance této třídy budou nabývat pouze regiony s textovým obsahem. Posloupnost čtení je v jednotlivých analýzách identifikována právě pro textové regiony.

```
class Region(object):
    def get_id(self) -> str: ...
    def get_coords(self) -> [tuple]: ...
    def get_box(self) -> Polygon: ...
    ...

class TextRegion(Region):
    def get_text(self) -> str: ...
```

4.3 Struktura posloupnosti čtení

Pro zachycení posloupnosti čtení slouží třídy z balíčku `ReadingOrder`. Pomocí těchto tříd bude v programu modelována jak ground truth posloupnost čtení z PageXML souboru, tak také posloupnost identifikovaná analyzátorem.

Třída Item

V kontextu posloupnosti čtení `Item` zastupuje právě jeden textový region. U každé instance této třídy lze definovat bezprostředního následníka, případně předchůdce, stejně tak zařazení do konkrétní skupiny. Zřetězením několika instancí lze vytvořit jednoduchý lineární seznam, který definuje posloupnost regionů.

```
class Item():
    def get_successor(self) -> Optional[Item]: ...
    def get_predecessor(self) -> Optional[Item]: ...
    def get_parent(self) -> Optional[Group]: ...
    ...
```

Třída Group

Třída `Group` sdružuje instance třídy `Item`, případně instance sebe sama. Tím umožňuje do sebe jednotlivé skupiny zanořovat a vytvářet tak stromovou strukturu. Pomocí příznaku je možné definovat, zda se jedná o uspořádanou skupinu, kde záleží na pořadí prvků, nebo jestli se jedná o neuspořádanou skupinu. Pokud je skupina zařazena v jiné, uspořádané skupině, může také ona mít následníka či předchůdce. Proto třída `Group` dědí třídu `Item`, která posloupnost umožňuje definovat.

```
class Group(Item):
    def is_ordered(self) -> bool:
    def is_unordered(self) -> bool:
    def add(self, item: Item):
    def add_candidates(self, source: TextRegion, successor: TextRegion):
    ...
```

Třída ReadingOrder

Třída `ReadingOrder` tvoří kořen struktury. Vlastní právě jednu instanci třídy `Group`, která se může dále větvit. Nabízí podpůrné metody pro zpracování posloupnosti čtení, například metodu `get_chain_reduction()`, která pro každou uspořádanou skupinu vrací množinu dvojic tvořící posloupnost čtení dané skupiny. `ReadingOrder` plně podporuje vyhodnocení jak pomocí metriky Recall, tak i pomocí komplexní metriky Prima.

```
class ReadingOrder(Item):
    def get_all_items(self) -> [Item]:
    def get_by_id(self, id) -> Optional[Item]:
    def get_chain_reduction(self):
    ...
```

4.4 Metriky

V rámci práce budou použity obě metriky představené v sekci 2.4.2. Obě metriky budou implementovány jako samostatné metody. Metrika Recall přijímá parametrem dvě instance třídy `ReadingOrder`, jednu pro identifikovanou posloupnost čtení a druhou pro ground truth.

Metrika Prima obdobně jako Recall příjme parametrem identifikovanou a ground truth posloupnost. Kromě toho příjme parametrem také instanci třídy `Document` a to protože Prima pro výpočet úspěšnosti detekce používá celkový počet textových regionů na stránce (ne všechny textové regiony musí být zařazené v ground truth posloupnosti). Z přijatých posloupností identifikuje matice vztahů. Ty následně porovná a pro každý vztah z chybové matice načte penalizace. Z těchto penalizací vypočte konečnou celková úspěšnost. Kromě číselných hodnot je možné získat textový popis chyb, viz příklad výpisu 4.1.

Výpis 4.1: Příklad textového popisu chyby identifikované metrikou Prima.

```
Ground Truth region r35 - region r36: Relation [-x-] instead of [->]
Ground Truth region r35 - region r38: Relation [<-] instead of [-x-]
Ground Truth region r36 - region r37: Relation [<-] instead of [->]
```

4.5 Analyzátor

Analyzátory jsou třídy, které po předložení instance `Document` provedou analýzu posloupnosti čtení textových regionů vyskytujících se v tomto dokumentu a vrátí instanci třídy `ReadingOrder` obsahující identifikovanou posloupnost čtení. V rámci této práce jsem navrhl a implementoval tři základní analyzátory pro jazykovou, prostorovou a kombinovanou analýzu.

Třída LmAnalyzer

`LmAnalyzer` provádí jazykovou analýzu identifikující posloupnost na základě pravděpodobnosti textové sekvence. Ke své funkci potřebuje a využívá vytvořený jazykový model a jeho slovník vytvořený nástrojem SentencePiece (viz sekce 3.2.4). Pomocí slovníku jsou vstupní textové sekvence převedeny na tokeny. Jazykový model následně odhadne podmíněné pravděpodobnosti současného výskytu sekvencí dokumentu, kdy na základě největší pravděpodobnosti provede spojení odpovídajících textových regionů do výsledné posloupnosti.

```
class LmAnalyzer(object):
    def __init__(self, model: Model, vocab: Vocabulary): ...
    def analyze(self, doc: Document) -> ReadingOrder: ...
```

Třídy DiagonalAnalyzer, ColumnarAnalyzer

Třídy `DiagonalAnalyzer` a `ColumnarAnalyzer` definují posloupnost na základě prostorových vlastností regionů a vzájemných vztahů. Po předložení instance `Document` provedou příslušné akce pro identifikaci posloupnosti a vrátí jej v podobě instance třídy `ReadingOrder`. Konkrétní chování jednotlivých tříd je popsáno dále, viz sekce 5.3.

```
class DiagonalAnalyzer():
    def analyze(self, doc: Document) -> ReadingOrder: ...
```

Třída ColumnarLmAnalyzer

`ColumnarLmAnalyzer` je zástupcem kombinované analýzy, která pracuje jak s prostorovými daty, tak také s textovým obsahem dokumentu. Proto ke své práci potřebuje nejen instanci třídy `Document`, pro který se provádí analýza posloupnosti, ale také instanci `LmAnalyzer`, pomocí kterého provádí v případě nutnosti jazykovou analýzu. Implementace třídy je více popsána v sekci 5.4.

```
class ColumnarLmAnalyzer():
    def analyze(self, doc: StubDocument, lm_analyzer: LmAnalyzer) -> ReadingOrder:
```

Kapitola 5

Implementace analýzy posloupnosti čtení

Kapitola popisuje implementaci jednotlivých tříd a metod analýzy posloupnosti čtení. Program byl implementován v jazyce Python ve verzi 3.9. Výčet použitých knihoven je uveden ve zdrojovém kódu implementace v souboru `requirements.txt`. Implementace vychází z návrhu, který je více popsán v kapitole 4. Kromě tříd a metod, uvedených v této kapitole, byla implementována třída `Plotter`, která má na starosti vizualizaci struktury dokumentu a posloupnosti čtení. Obrázky použité v této kapitole jsou generované instancí této třídy.

Implementace je veřejně dostupná na adrese https://github.com/holubecmichal/reading-order.

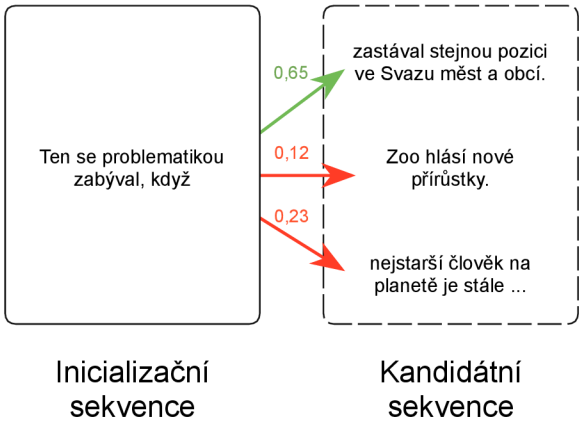
5.1 Použití jazykového modelu pro analýzu posloupnosti čtení

Jazyková analýza posloupnosti čtení je založena čistě na pravděpodobnosti výskytu sekvence o dané skladbě slov. Zavádí dva pojmy; *inicializační sekvence* a *kandidátní sekvence*. Inicializační sekvence je sekvence, pomocí které jsou inicializovány hodnoty skrytých vrstev jazykového modelu. Inicializované hodnoty skrytých vrstev jsou dále použity při odhadu pravděpodobnosti návaznosti kandidátních sekvencí, pro které se snažíme odhadnout, která ze sekvencí skutečně navazuje. Více názvosloví přibližuje obrázek 5.1.

Výpočet pravděpodobnosti inicializační sekvence t a kandidátní sekvence c probíhá ve třech krocích. V prvním kroku je jazykovému modelu předána inicializační sekvence o délce n, pomocí které jsou jak inicializovány hodnoty skrytých vrstev, tak získán výstup podmíněných pravděpodobností výskytu tokenů inicializační sekvence. Výstup má podobu matice $\mathbf{J}$, kde řádky jsou podmíněné pravděpodobnosti následujícího tokenu sekvence t_i na základě prefixu, $P(t_i|t_1 \dots t_{i-1})$. Počet řádků je roven délce inicializační sekvence n. Sloupce matice pak definují pravděpodobnost přes všechny tokeny slovníku a počet sloupců je tedy roven velikosti slovníku. Poslední řádek matice $\mathbf{J}$ představuje podmíněnou pravděpodobnost tokenu c_1, tedy prvního tokenu za inicializační sekvencí.

V druhém kroku je, spolu s inicializovanými hodnotami skrytých vrstev, na vstup modelu předán kandidát o délce m. Výstupem je matice $\mathbf{K}$ s hodnotami podmíněných pravděpodobností tokenů kandidáta c_i na základě prefixu, $P(c_i|t_1 \dots t_n c_1 \dots c_{i-1})$.

Ve třetím kroku je vytvořen vektor $\mathbf{y}$, který obsahuje pouze ty prvky z obou matic, které jsou směrodatné pro výpočet pravděpodobnosti. V tomto případě se jedná o poslední řádek matice $\mathbf{J}$. Dále pak všechny řádky matice $\mathbf{K}$, kromě řádku posledního. Z těchto vybraných



Obrázek 5.1: Příklad inicializační sekvence, pomocí které jsou inicializovány hodnoty skrytých vrstev jazykového modelu a kandidátních sekvencí, pro které odhadujeme pravděpodobnost návaznosti.

řádků jsou dále vybrány prvky z těch sloupců, které nesou pravděpodobnost odpovídajícího tokenu. Vektor $\mathbf{y}$ obsahuje právě m prvků, což je hodnota udávající počet tokenů kandidáta.

$$\mathbf{y} = \begin{bmatrix} P(c_1 \mid t_1 \dots t_n) \\ P(c_2 \mid t_1 \dots t_n c_1) \\ \vdots \\ P(c_m \mid t_1 \dots t_n c_1 \dots c_{m-1}) \end{bmatrix} \tag{5.1}$$

Výsledná pravděpodobnost návaznosti pro daného kandidáta k je spočtena jako geometrický průměr hodnot vektoru $\mathbf{y}$.

$$p_k = \sqrt[m]{\prod_{i=1}^{m} y_i} \tag{5.2}$$

Stejný postup výpočtu návaznosti je zopakován pro všechny zbylé kandidátní sekvence. Region inicializační sekvence tvoří dvojici s tím regionem (kandidátem), jehož hodnota p_k je, v porovnáním s ostatními kandidáty, nejvyšší.

5.2 Jazyková analýza

Jazykovou analýzu provádí třída `LmAnalyzer`, respektive je provedena po zavolání metody `analyze(document)`, která parametrem přijímá dokument určený k analýze. Na začátku jazykové analýzy jsou ze vstupního dokumentu získány všechny textové regiony, které budou následně podléhat analýze. Ty tvoří množinu kandidátů, která je zpracovávána v cyklu, dokud není v množině poslední zbývající prvek:

1. Pro textové regiony v množině kandidátů spočte matici pravděpodobností $\mathbf{M}$.
2. Identifikuje největší hodnotu matice $\max(\mathbf{M}_{i,j})$, která udává nejvyšší pravděpodobnost návaznosti. Index i je poté identifikátorem textového regionu představující inici-

alizační sekvenci a j je identifikátorem kandidáta, pro kterého byla pravděpodobnost návaznosti nejvyšší.

3. Odstraní textový region i a j z množiny kandidátů.
4. Vloží textové regiony i a j do posloupnosti čtení, j je v tomto případě bezprostřední následník textového regionu i.
5. Spojí obsah textových regionů i a j a vloží jej jako nový prvek do množiny kandidátů.
6. Dokud jsou v množině kandidátu alespoň dva prvky, cyklus se opakuje. V opačném případě posloupnost čtení byla plně identifikována a cyklus je ukončen.

Na konci cyklu jsou všechny textové regiony dokumentu zařazeny do posloupnosti čtení, respektive do instance třídy `ReadingOrder`, která je vrácena jako výsledek analýzy. Na obrázku 5.2 je znázorněn jeden krok cyklu, během kterého je vybrána největší hodnota $\max(\mathbf{M}_{i,j})$, odstraněni původní kandidáti i a j a přidán nový prvek vzniklý ze spojení původních kandidátů i a j.

	K1	K2	K3	K4	K5
K1	P11	P12	P13	P14	P15
K2	P21	P22	P23	P24	P25
K3	P31	P32	P33	P34	P35
K4	P41	P42	P43	P44	P55
K5	P51	P52	P53	P54	P55

(a) Matice **M** po dokončení 3. kroku cyklu.

	K1	K3	K5	K24
K1	P11	P13	P15	
K3	P31	P33	P35	
K5	P51	P53	P55	
K24				

(b) Matice **M** po dokončení celého cyklu.

Obrázek 5.2: Znázornění matice pravděpodobností **M**. Řádky znázorňují inicializační sekvence (oranžová barva), ve sloupcích jsou kandidátní sekvence. Červeně podbarvené pole na obrázku 5.2a je nejvyšší hodnota pravděpodobnosti návaznosti. Tato hodnota je v průběhu cyklu zpracovávána. Během zpracování jsou regiony k_2 a k_4 odstraněny z množiny kandidátů a jsou zařazeny do posloupnosti čtení, kde k_4 je bezprostředním následníkem k_2. Je vytvořen nový prvek k_{24}, který se skládá z těchto odstraněných regionů. Hodnoty na diagonále nejsou brány v úvahu, region nemůže být sám sobě bezprostředním následníkem. Červené rámování na obrázku 5.2b znázorňuje nově vytvořený prvek a nutnost výpočtu hodnot tohoto prvku jazykovým modelem v následujícím cyklu.

Proces výpočtu matice **M** je výpočetně náročný, konkrétně je časová složitost rovna $\mathcal{O}(n^2)$, kde n je počet textových regionů dokumentu. Z toho důvodu jsou implementovány různé optimalizace, které ukládají hodnoty skrytých vrstev jednotlivých inicializačních sekvencí. Díky tomu se od druhé iterace cyklu nepočítají znovu všechny hodnoty matice, ale

vypočítá se pouze jeden řádek, odpovídající novému prvku jako inicializační sekvence, a jeden sloupec, ve kterém je nový prvek považován za kandidáta. Výpočet řádku a sloupce je znázorněn na obrázku 5.2b.

5.2.1 Odhad na základě pevného počtu tokenů kandidátní sekvence

Při výpočtu pravděpodobností návaznosti pomocí pevného počtu tokenů nejsou použity všechny podmíněné pravděpodobnosti tokenů kandidáta tak, jak jej definuje vektor **y** v 5.4, ale jen určitý počet. Ten je definován při volání metody `use_hard_limit(tokens: int)`. Pokud je parametrem vyžádáno, aby se pravděpodobnost počítala pro dva tokeny, bude vektor **y** obsahovat pouze dva prvky, ze kterých je posléze vypočtena konečná pravděpodobnost.

$$\mathbf{y} = \begin{bmatrix} P(c_1 \mid t_1 \dots t_n) \\ P(c_2 \mid t_1 \dots t_n c_1) \end{bmatrix} \tag{5.3}$$

5.2.2 Odhad s využitím pravděpodobnosti kandidátní sekvence

Metoda `use_score_hard_limit(tokens: int)` definuje jiný vzorec pro výpočet pravděpodobnosti, než jak jej počítal dříve uvedený vzorec 5.2. Kromě vektoru **y**, který představuje podmíněnou pravděpodobnost kandidátní sekvence na základě inicializační $P(C|I)$, zavádí vektor **z**, který představuje pravděpodobnost samotné kandidátní sekvence $P(C)$. Aby bylo možné spočítat pravděpodobnost sekvence, je nutné inicializovat skryté stavy jazykového modelu vhodným počátečním tokenem c_{init}. Tímto tokenem byl zvolen speciální token `<s>`, který představuje počátek věty.

$$\mathbf{z} = \begin{bmatrix} P(c_1 \mid c_{init}) \\ \vdots \\ P(c_m \mid c_{init} \dots c_{m-1}) \end{bmatrix} \tag{5.4}$$

Celkové skóre se vypočte jako geometrický průměr rozdílu podmíněné pravděpodobnosti y a pravděpodobnosti kandidáta z.

$$p_k = \prod_{i=1}^{m} (\sqrt[m]{y_i} - \sqrt[m]{z_i}) \tag{5.5}$$

Stejně jako v případě předchozí metody, představené v 5.2.1, i tato metoda přijímá parametrem počet tokenů, se kterými bude pracovat.

5.3 Prostorová analýza

Prostorová analýza při identifikaci posloupnosti pracuje s prostorovými údaji jednotlivých regionů. Hranice regionů mohou být k dispozici na výstupu některého z OCR nástrojů, případně je možné hranice dodatečně anotovat manuálně. Podobu regionů, se kterými prostorová analýza pracuje, lze spatřit na obrázku 5.3.

Podstatou prostorové analýzy je identifikace prostorových vztahů mezi regiony. V rámci implementace jsem se pro identifikaci vztahů inspiroval článkem *Document Understanding for a Broad Class of Documents* [3], který je více popsán v sekci 2.4.1. Program pro identifikaci těchto vztahů implementuje třídu `DocTBRR`.

(a) Digitalizovaná stránka

(b) Rozpoznané hranice regionů

Obrázek 5.3: Příklad digitalizované stránky novin a rozpoznání hranic regionů stránky. Všechny tyto regiony jsou v průběhu prostorové analýzy použity pro orientaci a identifikování posloupnosti čtení mezi textovými regiony. Fialová barva označuje textové regiony, světle modrou barvou jsou rámovány obrázky. Růžově jsou zbarveny regiony jejichž smyslem je oddělovat text nebo znázornit hranici stránky, tzv oddělovače.

5.3.1 Třída DocTBBR a identifikace vztahů

Identifikace vztahů mezi regiony je implementována ve třídě `DocTBBR`. Instance je vytvořena pro konkrétní instanci třídy `Document`, ze které čerpá informace o regionech a jejich prostorová data.

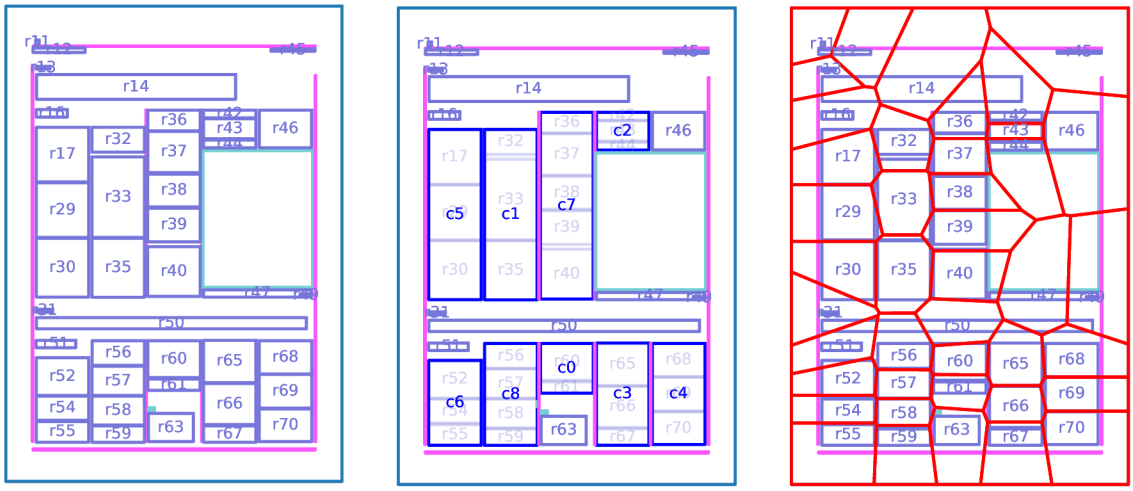
Při vytváření instance `DocTBBR` je spočtena prahová hodnota T. O tuto hodnotu se při identifikaci vztahů rozšiřují hranice regionů. Tím se eliminují nepřesnosti v souřadnicích jednotlivých regionů, které mohly vzniknout při digitalizaci dokumentů například jejich pootočením nebo nepřesností snímače, OCR nebo anotace. Při identifikaci vztahů se tedy program nespoléhá na přesnou hodnotu souřadnic, ale na jejich upravenou podobu (TBRR). Význam hodnoty T je detailněji znázorněn na obrázku 2.6.

Hodnota T je vypočtena z aktuálně analyzovaného dokumentu a neměla by přesahovat polovinu délky nejmenší strany, nejmenšího regionu na stránce. V programu jsou k výpočtu použity pouze textové regiony. Mezi těmito regiony je nalezen region s nejmenší plochou $r_{smallest}$. U tohoto regionu $r_{smallest}$ je následně nalezena jeho nejmenší strana $s_{shortest}$. Z té je posléze vypočtena hodota T.

$$T = \frac{s_{shortest}}{2} \tag{5.6}$$

Hodnotu T používá metoda `get_relations()` identifikující vztahy mezi regiony (viz obrázek 2.6 kde jsou vztahy a princip identifikace znázorněny). Výstupem metody je struktura obsahující dvě matice. Jedna matice nese prostorové vztahy pro osu x a druhá pro osu y. Velikost matice je n^2, kde n je počet regionů na stránce. V této matici lze následně dohledat vztah jakéhokoli regionu vůči jinému. Výsledná struktura je obalena třídou `Relations`, která nabízí metody pro ověření konkrétního vztahu tak. Vytváření matic vztahů je optimalizované a využívá faktu, že pokud region A je vůči B ve vztahu *precedes*, pak B je vůči

A ve vztahu inverzním, tedy *i_precedes*. Při zpracování jednoho regionu jsou identifikovány hned dva vztahy.



(a) Textové regiony (b) Sloupcové regiony (c) Voroného diagram

Obrázek 5.4: Vizualizace nahrazení textových regionů sloupcovými regiony a analýzy sousednosti pomocí voroného diagramu.

Pro zpřesnění identifikace vztahů jsem navrhl a implementoval metodu pro identifikaci sloupců. Sloupce se použijí při identifikaci vztahů namísto regionů, které do těchto sloupců spadají. Je tím jednak redukován počet regionů a počet vztahů, ale také zpřesněna identifikace posloupnosti, protože pro regiony ve sloupcích je již implicitně uvažován směr čtení odshora směrem dolů.

Tuto metodu implementuji ve třídě `DocTBBR`, metodou `get_columns()`. Ta využívá identifikovaných vztahů mezi regiony a snaží se odhadnout, které regiony spolu tvoří sloupcovou strukturu. Vztahy mezi regiony jsou pro účely identifikace sloupců vytvořeny s jinou prahovou hodnotou T_{col}. Při výpočtu této prahové hodnoty je nalezena nejkratší horizontální linie $h_{shortest}$, jejíž délka je následně použita při identifikaci vztahů.

$$T_{col} = h_{shortest} \tag{5.7}$$

Program sloupce identifikuje na základě několika faktorů. Regiony ve sloupci musí být v prvé řadě vertikálně zarovnány, tedy na ose x musí splňovat vztah *equals* a na y ose musí být ve vztahu *precedes*, respektive *i_precedes*.

V druhé řadě musí tyto regiony spolu sousedit, aby nedošlo ke spojení dvou regionů, které spolu nesouvisí. K tomu je implementována analýza sousednosti na základě Voroného diagramu, viz obrázek 5.4c. Analýza sousednosti pomocí Voroného diagramu rozdělí prostor dle množiny bodů A, okolo kterých vytvoří Voroného polygony. Pro každý bod $x \in A$ platí, že každý pomyslný bod uvnitř Voroného polygonu patřící k x, je blíže právě k bodu x, než k jakémukoli jinému bodu z množiny A. V této práci je množinou bodů, pro kterou se vytváří Voroného diagram, množina geometrických středů jednotlivých regionů stránky. Jako sousedi jsou identifikovány ty regiony, jejichž polygony se dotýkají polygonu patřícího k x.

Předposledním kriteriem je, že mezi dvěma sousedními regiony ve sloupci musí existovat přímá viditelnost, tedy nesmí mezi nimi existovat žádný jiný region (obrázek, oddělovač, případně jiný textový region).

Poslední kriterium je faktor vzdálenosti. Vzdálenost mezi takovým dvěma regiony nesmí přesáhnout vzdálenost definovanou vzorcem:

$$d = \text{mean}(distances) + \text{std}(distances) \tag{5.8}$$

kde funkce *std* počítá směrodatnou odchylkou a *mean* počítá průměrnou vzdálenost všech sousedních regionů *distances*, které splnily všechna předchozí kritéria. Rozdíl mezi dokumentem obsahující textové regiony, a dokumentem se sloupcovými regiony je znázorněn na obrázku 5.4.

Třídu `DocTBBR` využívají `DiagonalAnalyzer`, `ColumnarAnalyzer` a `ColumnarLmAnalyzer`.

```
class DocTBRR():
    def get_relations(self) -> Relations:
    def get_columns(self) -> {Column}:
    def is_in_col(self, id) -> bool:
    ...
```

5.3.2 Diagonální analýza

Diagonální analýza využívá identifikovaných vztahů uložených v instanci třídy `Relations`. Ta implementuje dvě, pro analýzu zásadní metody; `is_vertical_before_in_reading(a, b)` a `is_horizontal_before_in_reading(a, b)`. Ty vrací booleovskou hodnotu, která značí, zda je region `a` v posloupnosti čtení před `b` ve vertikálním, respektive horizontálním směru. Těla metod jsou více rozebrány v článku [3], ze kterého jsou tyto metody převzaty.

Analýza porovnává pouze textové regiony. Ta dvojice, která splní obě výše uvedené podmínky, je uložena do seznamu pro pozdější zpracování. Po dokončení cyklu s porovnáním všech regionů je k dispozici struktura `before_in_reading` představující Complete chain, definovaný v 2.4.1.

Výpis 5.1: Kód diagonální analýzy pracující s identifikovanými prostorovými vztahy mezi regiony. Na základě vztahů identifikuje, zda se region `a` nachází v prostoru před regionem `b` jak ve vertikálním směru (osa y), tak ve směru horizontálním (osa x). Pokud ano, je dvojice přidána do struktury představující Complete chain.

```
class DiagonalAnalyzer():
    def analyze(self, doc: StubDocument) -> ReadingOrder:
        tbrr = DocTBRR(doc)
        r = tbrr.get_relations()

        before_in_reading = []

        for a in doc.get_text_regions():
            for b in doc.get_text_regions():
                if a == b:
                    continue

                if r.is_vertical_before_in_reading(a, b)
                and r.is_horizontal_before_in_reading(a, b):
                    before_in_reading.append((a, b))

        chain = list(topological_sort(before_in_reading))
        return chain_to_reading_order(chain, doc)
```

Strukturu lze interpretovat jako acyklický orientovaný graf, kde každá dvojice představuje hranu mezi vrcholy. Abychom získali Chain reduction, definovaný v 2.4.2, je nad grafem

aplikované topologické řazení. Výstupem topologického řazení je uspořádání vrcholů (textových regionů) v podobě lineární cesty grafem, od jeho počátku k jeho konci. Tato cesta zároveň představuje identifikovanou posloupnost čtení. Diagonální analýza pracuje pouze s textovými regiony stránky, viz obrázek 5.4a. Implementace topologického řazení byla převzata z veřejně dostupného zdroje[1].

5.3.3 Sloupcová analýza

Sloupcová analýza pracuje na stejném principu jako analýza diagonální. Hlavním rozdílem je ve využití sloupců a tím redukování počtu elementů, které je nutné porovnávat metodami pro ověření horizontální a vertikální posloupnosti. V konečném důsledku je posloupnost čtení identifikována mezi textovými regiony, které nejsou ve sloupcích a novými sloupcovými regiony. Nahrazení textových regionů sloupci je znázorněné na obrázku 5.4b.

Po získání cesty grafem aplikováním topologického řazení na výslednou strukturu jsou sloupcové regiony nahrazeny původními textovými regiony. U těchto regionů je předpokládaná implicitní posloupnost čtení odshora sloupce směrem dolů.

Výpis 5.2: Kód sloupcové analýzy. Základ metody je podobný diagonální analýze, pouze se neporovnává vztah mezi všemi regiony, ale mezi sloupci a regiony, které se ve sloupcích nenacházejí. V závěru analýzy jsou sloupcové regiony nahrazeny textovými regiony, které v těchto sloupcích nacházejí, viz řádek 15.

```
1  class ColumnarAnalyzer(object):
2      def analyze(self, doc: Document) -> ReadingOrder:
3          tbrr = DocTBRR(doc)
4          r = tbrr.get_relations()
5          before_in_reading = []
6
7          regions = tbrr.get_cols_and_independent_regions()
8          for a in regions:
9              for b in regions:
10                 # kod shodny s~diagonalni analyzou, preskoceno
11
12         chain = []
13         for i in topological_sort(before_in_reading):
14             if tbrr.is_col(i):
15                 chain += tbrr.get_col(i).get_region_ids()
16             else:
17                 chain.append(i)
18
19         return chain_to_reading_order(chain, doc)
```

5.4 Kombinovaná analýza

Představené analýzy vykazují některé nedostatky. Jazyková analýza se o posloupnosti čtení rozhoduje pouze na základě textových dat jednotlivých regionů. Tím může dojít ke spojení dvou regionů, o kterých jazykový model na základě pravděpodobnosti sice usoudil jejich jazykovou návaznost, ale které spolu logický nesouvisí. Například spolu tyto regiony vůbec nesousedí nebo jsou mezi nimi jiné regiony, které měly ve skutečnosti stát v posloupnosti mezi spojenými regiony.

[1] https://pythonwife.com/topological-sort-algorithm-in-python/

Podobný problém existuje také u prostorové analýzy, která naopak posloupnost usuzuje pouze na základě prostorových informací a nehledí na jazykovou návaznost textu. Může tak dojít ke spojení dvou regionů, které spolu sice sousedí a splňují všechny podmínky návaznosti prostorové analýzy, ale skutečný tok textu je jiný.

Kombinovaná analýza se snaží tyto nedostatky eliminovat a využít výhod obou analýz jejich spojením. Nejprve zpracuje vstupní dokument pomocí prostorové sloupcové analýzy. Její výstup následně upravuje jazykový model. Před zpracováním jazykovým modelem se identifikují všechny textové regiony, které jsou na konci sloupce. Sloupce, ve kterých se tyto regiony nachází, představují inicializační sekvence. Pro každý koncový region se dle sestavených pravidel identifikují regiony, které mohou po tomto regionu ve skutečnosti následovat. Pravidla jsou následující:

- Vyber region se kterým je již nyní koncový region spojen (výstup prostorové analýzy).
- Vyber nejbližší region z osy x napravo od koncového regionu. Ověř, zda se nad tímto regionem na ose y nenachází jiný region a pokud ano, přidej tento region k analýze.
- Vyber nejbližší region z osy y, který se nachází pod koncovým regionem a který na ose x splňuje podmínky *starts, i_starts, during, finishes, overlaps, i_overlaps, equals.* Tyto dva regiony se tedy určitým způsobem na ose x překrývají zleva, zprava nebo jsou shodné.
- Pokud je některý z vybraných regionů ve sloupci, použij místo něj první region v tomto sloupci.

Obrázek 5.5: Kombinovaná analýza spojuje funkce prostorové a jazykové analýzy. Nejprve provede prostorovou analýzu, jejíž výstup posléze koriguje jazykovým modelem. Jazykové analýze podléhají regiony na konci sloupce. Na obrázku je takový region orámován zelenou barvou a z pohledu jazykové analýzy sloupec, ve kterém se tento region nachází, představuje inicializační sekvenci. Červeně jsou orámované regiony, pro které jazykový model odhaduje, zda jsou návazné vůči inicializační sekvenci a představují tak kandidátní sekvence. I když je region `r35` zprava nejblíže analyzovanému regionu, není vybrán k analýze, protože se vyskytuje ve sloupci. Místo něj je použit první region tohoto sloupce `r32`.

Pro každý region (sloupec) představující inicializační sekvenci je jazykovým modelem odhadnuta pravděpodobnost návaznosti kandidátních sekvencí. Pokud jazykový model svým rozhodnutím o návaznosti (dle nejvyšší pravděpodobnosti) potvrdil původní rozhodnutí prostorové analýzy, pokračuje se zpracováním dalšího koncového (inicializačního) regionu. Pokud však jazykový model odhadl návaznost jiného regionu, je přistoupeno ke korekci a vhodné úpravě tak, aby posloupnost respektovala toto rozhodnutí.

Při rozhodování jazykového modelu je použita prahová hodnota, pomocí kterého je rozhodnutí jazykového modelu řízeno. Ke korekci posloupnosti dojde pouze v případě, kdy jazykový model odhadne podmíněnou pravděpodobnost dvou sekvencí alespoň 85 % v případě dvou kandidátních sekvencí. Pokud je počet kandidátních sekvencí vyšší jak dvě, pak je práh nastaven na 65 %. Hodnoty byly zvoleny na základě pozorování a experimentování s výslednou implementací jako nejvíce vhodné.

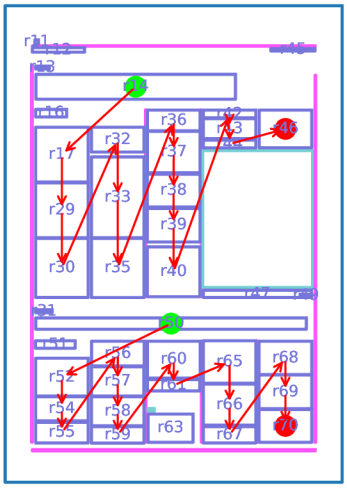
5.5 Ukázka analýzy posloupnosti čtení

V této sekci je pro názornost uveden výstup popsaných analýz na novinové stránce z obrázku 5.2a. Na stránce se vyskytuje celkem 54 regionů, z nichž 42 je textových. Analýza posloupnosti čtení je provedena pouze nad těmito textovými regiony. Počet regionů v ground truth je 30. Rozdíl je způsoben tím, že v ground truth jsou v tomto případě zahrnuty pouze regiony, které mají definovanou posloupnost, a naopak nejsou zahrnuty regiony, které posloupnost definovanou nemají (číslo stránky, datum, název rubriky a podobně). Výstup jednotlivých metod je znázorněn na obrázku 5.6. Pomocí šipek je ilustrována identifikovaná posloupnost čtení. Zelenou značkou je na obrázku vyznačen počátek posloupnosti, červená značka značí konec posloupnosti čtení. Výstup byl vyhodnocen pomocí metrik Prima a Recall, jejichž hodnoty jsou uvedeny v tabulce 5.1.

Z výstupu jazykové analýzy byla vybrána metoda s nejvyšší úspěšností, konkrétně metoda počítající s pravděpodobností kandidáta (skóre), viz 5.6b. Úspěšnost jazykového analýzy při určení posloupnosti je velmi nízká. Protože jazykový model nemá, kromě textového obsahu regionů, žádné jiné informace, vyhodnocuje se pravděpodobnost návaznosti i takových textových regionů, které spolu logicky nijak nesouvisí. Z hlediska vyhodnocení posloupnosti by bylo výhodnější, kdyby byly vyhodnoceny pouze ty regiony, které mohou ve skutečnosti tvořit posloupnost, tedy například sousedící regiony.

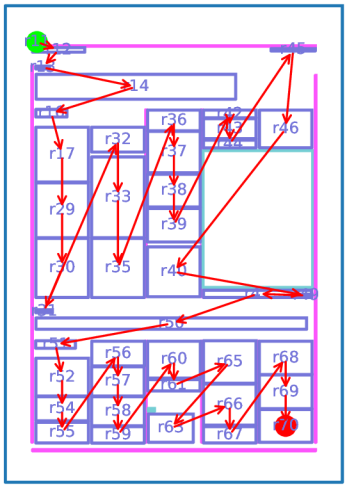
Prostorová analýza 5.6c v podobě diagonální metody dosahuje mnohem lepších výsledků. Přesto stále vykazuje nepřesnosti především v oblastech sloupců, u kterých je posloupnost vcelku zřejmá. V tomto případě například chybí posloupnost mezi regiony `r39` a `r40`, respektive `r61` a `r63`. Takové chyby je schopna identifikovat a opravit sloupcová analýza 5.6d. Ta však v některých případech nedokáže v posloupnosti spojit sloupce samotné, jako například `r35` s `r36`.

Obrázek 5.6e znázorňuje kombinovanou analýzu, která spojuje vlastnosti prostorové a jazykové analýzy. Díky tomu, že má přístup jak k prostorovým, tak textovým datům, se dokáže lépe rozhodnout o posloupnosti. V případě této novinové stránky je úspěšnost kombinované metody nejvyšší.

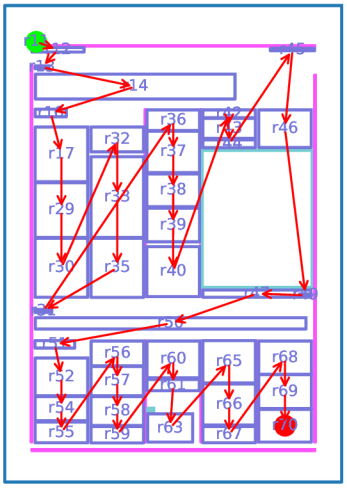


(a) Ground truth

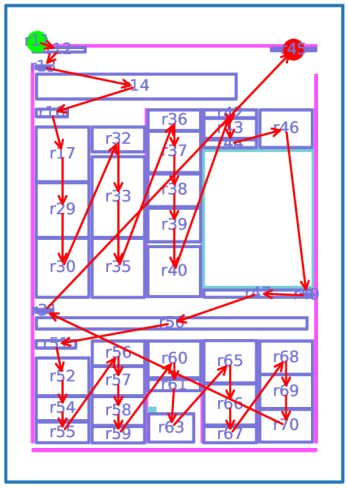
(b) Jazyková analýza, skóre



(c) Prostorová diagonální analýza



(d) Prostorová sloupcová analýza



(e) Kombinovaná analýza

Obrázek 5.6: Grafický výstup metod pro určení posloupnosti čtení. Ground truth 5.6a nemá v anotaci definované neuspořádané prvky, ty však jsou jednotlivými metodami do posloupnosti zahrnuty.

Tabulka 5.1: Vyhodnocení jednotlivých metod z obrázku 5.6

	Prima	Recall
Jazyková analýza, skóre, 5 tokenů	47,73 %	16,67 %
Prostorová diagonální analýza	82,35 %	76,67 %
Prostorová sloupcová analýza	89,36 %	83,33 %
Kombinovaná analýza, skóre, 5 tokenů	93,33 %	90,00 %

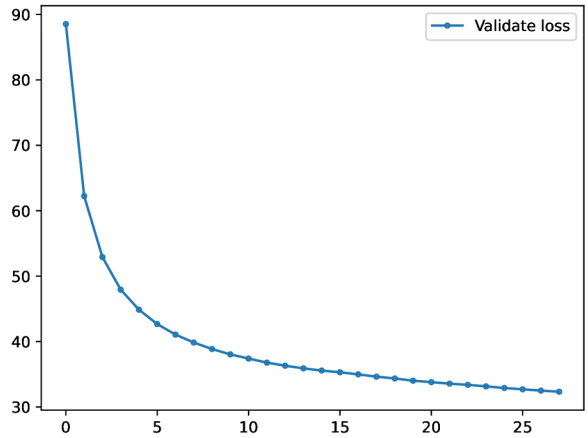
Kapitola 6

Experimenty

Kapitola popisuje charakteristiky použitého jazykového modelu a experimenty, které byly s tímto jazykovým modelem provedeny. Dále kapitola představuje experimenty s jednotlivými metodami identifikace posloupnosti čtení na připravených datasetech.

6.1 Sestrojení jazykového modelu

Pro účely experimentálního ověření schopnosti identifikace posloupnosti čtení jsem vytvořil jazykový model, který jsem natrénoval na výňatku české Wikipedie. Jako implementační prostředí pro sestrojení jazykového modelu jsem vybral jazyk Python s využitím frameworku pro strojové učení PyTorch (kompletní soupis knihoven a jejich verzí je přiložen u zdrojových kódu v souboru `requirements.txt`). Jazykový model je založen na architektuře LSTM. Architektura a trénovací skript byly převzaty z veřejně dostupného PyTorch repozitáře[1]. Zdrojový kód trénovacího skriptu byl upraven pro účely této práce, kromě odstranění nepoužitých segmentů byla hlavní změnou úprava zpracování vstupních dat z tokenizace slov na tokenizaci podslov. Výsledný model pracuje se slovníkem o velikosti 20K tokenů a dosáhl perplexity 31,79. Průběh trénování je zobrazen na grafu 6.1.



Obrázek 6.1: Průběh trénování jazykového modelu. Trénování bylo zastaveno po dokončení 28 epochy s výslednou perplexitou 31,79.

[1] https://github.com/pytorch/examples/tree/main/word_language_model

6.1.1 Trénovací korpus

Jazykový model byl trénován a vyhodnocen na výňatku české Wikipedie poskytnutý vedoucím práce. Korpus byl rozdělen na trénovací, validační a testovací sadu. V tabulce 6.1 jsou uvedeny základní statistiky datových sad. Obsah pokrýval nejrůznější témata od historie přes geografii, popis různých fyzikálních jevů až po biografii umělců.

Původní pokus trénovat model metodou tokenizace slov selhal z důvodu velikosti modelu, respektive z důvodu nedostatečné paměťové kapacity GPU. Proto jsem přikročil k tokenizaci za pomocí podslov.

Slovník byl vytvořen pomocí nástroje SentencePiece, který je popsaný v sekci 3.2.4. Subword model byl natrénován z trénovací sady korpusu. Výsledný subword model je použit jako enkodér, pomocí kterého je textový řetězec převeden na sekvenci tokenů (token = pozice podslova ve slovníku). Statistiky jednotlivých sad po převodu na podslova jsou uvedeny v tabulce 6.1.

Kromě tokenů představující jednotlivá podslova jsou ve slovníku také 3 speciální tokeny; `<unk>`, `<s>` a `</s>`. Token `<unk>` je použit v případě, kdy se textové sekvenci nachází podslovo, které se nevyskytuje ve slovníku. To může být podslovo, které se vůbec neobjevilo ve zdrojové sadě, ze které slovník vznikl, nebo speciální znak typu copyright © a podobně. Tokeny `<s>` a `</s>` pak představují začátek, respektive konec věty.

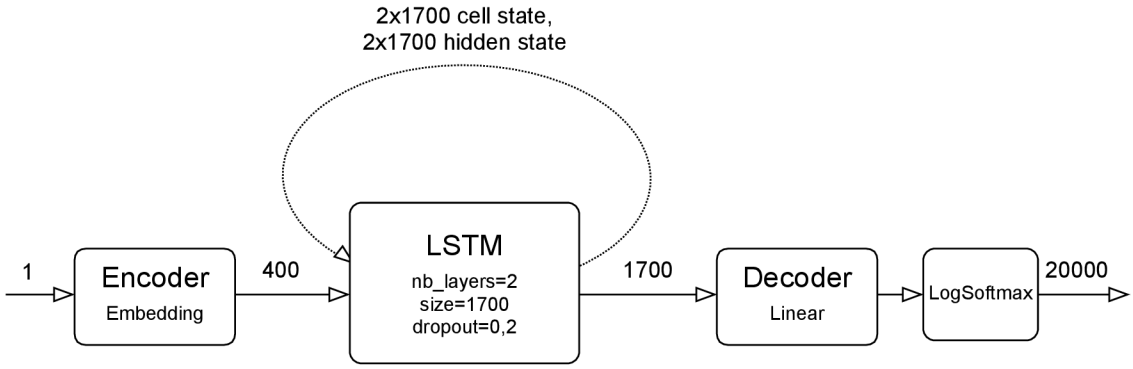
Tabulka 6.1: Statistiky korpusu. Počet slov spočítán pomocí `nltk.word_tokenize()`. Počet tokenů spočítán pomocí vytvořeného subword modelu.

	Velikost [MB]	#slov celkem	#unikátních slov	#tokenů celkem
train.txt	725,1	115 271K	2 356K	332 083K
valid.txt	3,1	489K	82K	706K
test.txt	2,7	424K	73K	617K

6.1.2 Architektura jazykového modelu

Architektura modelu sestává ze tří vrstev; enkodéru, LSTM a dekodéru. Enkodér převádí vstupní token do spojitého prostoru, kde je tokenu přiřazen 400 rozměrný vektor reálných čísel (embedding). Vektor je vstupem LSTM. LSTM sestává ze dvou vrstev (stacked LSTM), každá z vrstev obsahuje 1700 jednotek. Výstupem je několik 1700 rozměrných vektorů. Jeden výstupní vektor LSTM představuje vektor příznaků, který je dále zpracovány dekodérem. Zbývající vektory tvoří dvojici (*cell state*, *hidden state*) a reprezentují hodnoty skrytých stavů rekurentní sítě, které jsou přenášeny na vstup LSTM jako kontext pro zpracování dalších sekvenčních dat. Dekodérem je plně propojená lineární vrstva, která převádí vektor příznaků na vektor o velikosti slovníku $|V|$. Architektura a tok dat jsou znázorněny na obrázku 6.2.

Obecně je vstupem jazykového modelu vektor $\mathbf{x}$ o n prvcích. Jednotlivé prvky vektoru jsou tokeny původní textové sekvence. Výstup enkodéru, respektive vstup LSTM je pak matice o rozměru $n \times 400$. Obdobně výstup LSTM, který je vstupem dekodéru je maticí o rozměru $n \times 1700$. Dvojice hodnot (cell state, hidden state) skrytých vrstev není na n závislá. Pro každou sekvenci existuje právě jedna dvojice, která reprezentuje stav modelu, respektive hodnoty skrytých vrstev, které jsou z této vstupní sekvence inicializovány.



Obrázek 6.2: Schéma architektury jazykového modelu. Vstupní hodnota je převedena na 400 rozměrný vektor, který je vstupem LSTM. Výstupem LSTM je vektor příznaků a hodnoty skrytých stavů. Vektor příznaků je dekodérem převeden na výstupní vektor o velikosti slovníku. Hodnoty skrytých vrstev jsou v případě zpracování dalších sekvenčních dat předány na vstup LSTM. Hodnota x na obrázku představuje právě jeden token.

6.1.3 Trénování

Trénování jazykového modelu probíhalo na GPU a celkem proběhlo 28 epoch. Model byl trénován metodou Mini-batch gradient descent, kdy gradient byl počítán a váhy sítě byly upraveny po každé zpracované dávce dat (minibatch). Každý batch obsahoval 20 sekvencí, každá sekvence pak obsahovala 35 tokenů. Délka sekvence zároveň definovala rozsah výpočtu algoritmu Back-propagation Through Time (BPTT), během kterého se počítal gradient napříč skrytými stavy. Jako objektivní funkce byl použit Negative Log Likelihood, NLLLoss. Tomu byl předán výstup modelu, po použití funkce Softmax, a hodnoty ve formě tokenů, vůči kterým byla následně spočtena chyba modelu. Po vypočtení chyby a gradientu algoritmem BPTT došlo k úpravě vah odečtením hodnoty gradientu s učícím krokem o hodnotě 2 (SGD).

Po dokončení každé epochy proběhlo ověření úspěšnosti modelu na validační sadě. Pokud model vykazoval oproti předešlé epoše na této sadě menší chyby, došlo k uložení modelu. Po dokončení poslední epochy byl model s nejlepšími parametry vyhodnocen na testovací sadě, na které bylo dosaženo perplexity 31,79. Tento model je dále v textu uvažován jako stěžejní, který byl použit při všech publikovaných experimentech.

6.2 Experimenty s jazykovým modelem

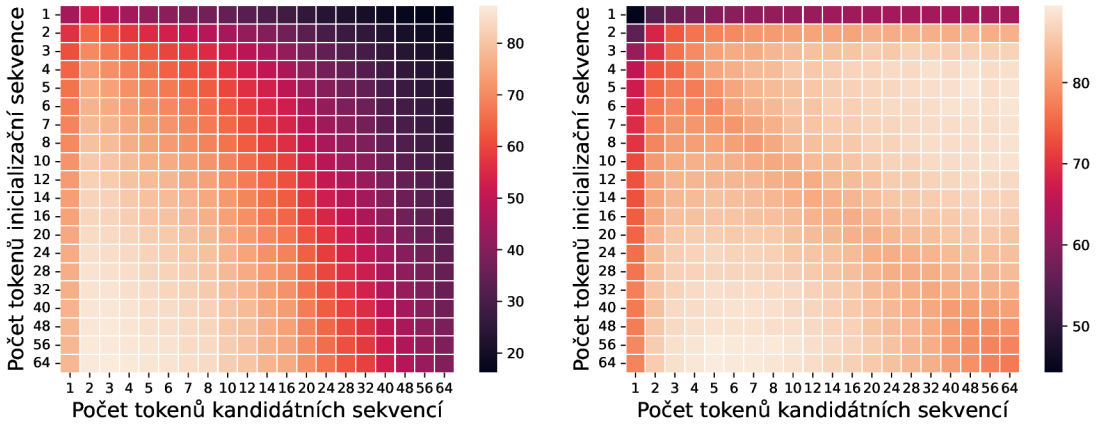
Po sestrojení jazykového modelu jsem provedl sadu experimentů s cílem zjistit chování jazykového modelu. Prvním experimentem je ověření chování modelu základě různě dlouhých sekvencí, dále nazývaný jako experiment délky (popsáno v sekci 5.2.1). Druhý experiment je obdobný, avšak měří úspěšnost modelu s využitím pravděpodobnosti kandidátní sekvence (popsáno v sekci 5.2.2), dále nazývaný jako experiment skóre.

Při experimentu zjišťování vlivu různě dlouhých sekvencí byly jazykovému modelu předkládány sekvence o délce 1–64 tokenů a měřen počet správného výběru kandidáta. Rozsah délek byl použit jak pro měření vlivu délky inicializační sekvence, tak také vlivu délky kan-

didátní sekvence. Pro každou konfiguraci bylo vyhodnoceno 10K vzorků, v každém vzorku bylo vybíráno z 8 kandidátů, z nichž 1 byl správný. Ostatní kandidáti byly vybráni náhodně a proti inicializační sekvenci obsahují náhodný text. Pravděpodobnosti byly počítány dle vzorce 5.2. Měření bylo provedeno na testovací sadě korpusu Wikipedie.

Výsledkem experimentu jsou úspěšnosti jednotlivých konfigurací, které jsou zobrazeny pomocí heatmapy, viz 6.3a. Z experimentů vyplynulo, že model lépe odhaduje, pokud je délka inicializační sekvence co největší. Naopak délka kandidátních sekvencí by měla být, pro úspěšný odhad návaznosti, omezená na pár počátečních tokenů. Toto chování je zapříčiněno architekturou modelu, pro kterou je charakteristické, že s přibývající délkou inicializační sekvence jsou skryté stavy sítě lépe inicializovány a model poté lépe odhaduje. Naopak s přibývající délkou kandidátní sekvence dochází k postupnému vytrácení informace inicializační sekvence, což ovlivňuje konečné rozhodnutí modelu.

Nejmenší naměřená úspěšnost je 16.2 % při konfiguraci (64,1), maximální je 87.3 % při konfiguraci (3, 64). Graf 6.4a představuje průměrnou úspěšnost přes všechny délky inicializační sekvence. V případě použití jazykové analýzy, konkrétně metody s pevným počtem tokenů `use_hard_limit(tokens: int)` (sekce 5.2.1), je tedy vhodné pracovat se dvěma až třemi tokeny.

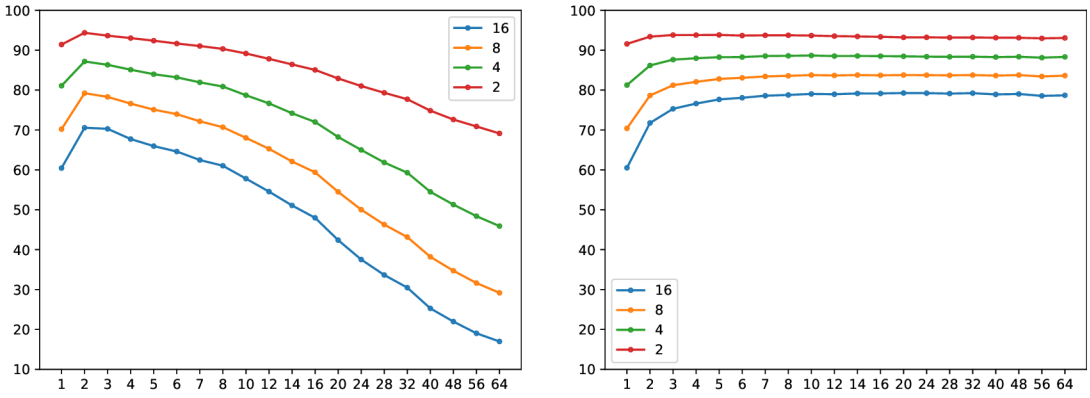


(a) Experiment délky

(b) Experiment skóre

Obrázek 6.3: Výsledky úspěšnosti modelu na základě jednotlivých konfigurací, výběr jednoho kandidáta z 8 možných. Osa x udává počet tokenů kandidátní sekvence, v daném vzorku měli všichni kandidáti stejnou délku. Osa y značí délku inicializační sekvence. Každá dlaždice zahrnuje 10K vzorků a znázorňuje průměrnou hodnotu úspěšnosti v dané konfiguraci. Měřítko je uvedeno v procentech. Oproti 6.3a vykazuje 6.3b větší úspěšnost i v případě delších kandidátních sekvencí.

Druhý experiment, experiment skóre, je v principu podobný prvnímu s tím rozdílem, že pro určení výběru kandidáta je vypočteno skóre s pomocí pravděpodobnosti kandidátní sekvence (popsáno v sekci 5.2.2). I v tomto případě bylo cílem zjistit vliv délky sekvencí. Výpočet pravděpodobnosti kandidáta $P(C)$ je specifický, protože pro odhad pravděpodobnosti s vytvořeným jazykovým modelem je nejdříve nutné vhodně inicializovat skryté stavy modelu a tedy vybrat vhodný token, kterým budou stavy inicializovány. Experiment byl proto proveden pro dvě různé inicializace skrytých stavů, v jednom případě byly použité vynulované skryté stavy, v druhém případě byly skryté stavy inicializovány tokenem `<s>`, který představuje počátek věty. Při použití tokenu `<s>` model dosahoval lepších výsledků a je proto popsán v rámci tohoto experimentu a také použit v implementaci.



(a) Experiment délky

(b) Experiment skóre

Obrázek 6.4: Průměrná úspěšnost pro různý počet kandidátů. Graf 6.4a ukazuje, že při výpočtu podmíněné pravděpodobnosti úspěšnost jazykového modelu, s rostoucí délkou kandidátní sekvence, klesá. Naopak úspěšnost při použití skóre 6.4b je trend zpočátku rostoucí a se zvyšující se délkou kandidátní sekvence se úspěšnost stabilizuje. Osa x není lineární.

Experiment byl proveden s nastavením délek 1–64 tokenů, při 8 kandidátech. Pro každou konfiguraci bylo vyhodnoceno 10K vzorků. Výsledek měření je vyobrazen v grafu 6.3b. Výpočet skóre eliminoval nevýhody jazykového modelu, který s delší kandidátní sekvencí potlačoval informace z inicializační sekvence.

Nejmenší naměřená úspěšnost je 44.2 % při konfiguraci (1,1), maximální je 89.5 % při konfiguraci (5, 64). Graf 6.4b představuje průměrnou úspěšnost přes všechny délky inicializační sekvence. Na rozdíl od předchozího experimentu, viz graf 6.4a, se průměrná úspěšnost s rostoucí délkou sekvence stabilizuje. To je nejspíše zapříčiněno právě použitím pravděpodobnosti kandidáta $P(C)$, protože jeho použitím při výpočtu skóre je potlačen vliv kandidáta, který se projevuje ovlivněním skrytých stavů při výpočtu podmíněné pravděpodobnosti $P(C|I)$ jazykovým modelem, jak ukázal experiment délky.

Ve výsledku experimentu skóre 6.3b se na diagonále objevil zajímavý artefakt, který ukazuje, že pokud jsou délky inicializační sekvence a kandidátní sekvence přibližně stejné, úspěšnost skóre je nižší, než když jsou tyto délky rozdílné. Příčinu tohoto chování se mi nepodařilo odhalit.

Pro experimenty s využitím této metody `use_score_hard_limit(tokens: int)` budou použity délky sekvencí v rozsahu 4–6 tokenů.

6.3 Popis experimentu identifikace posloupnosti čtení a použitých dat

V této sekci jsou popsány experimenty identifikace posloupnosti čtení na připravených datasetech. V experimentech jsou použity všechny představené analýzy pro identifikaci posloupnosti, tedy jazyková, prostorová a kombinovaná analýza. V případě jazykové a kombinované analýzy jsou ověřeny obě metody pro nastavení chování jazykového modelu: metoda pevného počtu tokenů 5.2.1 a metoda pracující s pravděpodobností kandidátní sekvence 5.2.2. V případě metody pevného počtu tokenů jsou pro experimenty použity 2–3 tokeny. U druhé zmíněné metody je v experimentech použit rozsah 4–6 tokenů.

6.3.1 Datasety

Vyhodnocení proběhlo na 3 datasetech. Prvním datasetem je dataset Hospodářských novin, který jsem vytvořil pro účely této práce. Jedná se o vydání ze dne 12. 1. 2022, obsahující celkem 16 stránek, z nichž k analýze bylo vybráno 14 stránek. Zbývající obsahovaly pouze reklamy nebo jeden textový region. Vydání jsem měl k dispozici ve formátu PDF, který obsahoval pouze obrázky jednotlivých stránek (bez možnosti extrakce textu). Stránky jsem vyexportoval do formátu TIFF a zpracoval OCR enginem Tesseract v nástroji Aletheia[2]. Ve stejném nástroji jsem vytvořil ground truth posloupnost čtení, proti kterému je identifikovaná posloupnost porovnána. Prostorová data jednotlivých regionů byla identifikována automaticky, případné nesrovnalosti byly manuálně upraveny. Každá takto upravená stránka byla exportována do samostatného PageXML souboru. Každý soubor obsahuje průměrně 46 regionů, průměrně 36 je textových. Každý textový region obsahuje průměrně 234 znaků. Dataset obsahuje celkem 491 textových regionů, token `<unk>` obsahuje 10 z nich (2 %).

Druhým datasetem je dataset Lidových novin. Tento dataset mi byl poskytnut vedoucím této práce. Obsahuje vzorek různých novinových stránek, které byly zpracovány v rámci projektu PERO[3]. Výchozí dataset sestával z PageXML souborů, které obsahovaly textový obsah a souřadnice definující hranice jednotlivých regionů. Pomocí nástroje Aletheia jsem pro jednotlivé stránky definoval ground truth posloupnosti čtení. Celkem dataset obsahuje 66 souborů. Obsahuje pouze textové regiony, kterých je průměrně 22 na soubor. Průměrně každý textový region obsahuje 1119 znaků. Dataset obsahuje celkem 1442 textových regionů, 96 textových regionů (7 %) obsahuje token `<unk>`.

Třetí dataset byl vytvořen v rámci projektu *Europeana Newspapers Project*. Jedná se o projekt zabývající se shromažďováním, digitalizací a anotací historických novinových článků. K databázi jsem získal přístup na požádání skrze systém Salfordské univerzity[4]. Z databáze bylo staženo celkem 117 dokumentů ve formátu PageXML s ground truth posloupností čtení. Jedná se o novinové stránky s klasickým sloupcovým rozložením. Některé novinové stránky obsahují neuspořádaný obsah, například reklamy. V průměru je v každém souboru 66 regionů, z nichž 53 je textových. Průměrně každý textový region obsahuje 301 znaků. Dataset obsahuje celkem 6181 textových regionů, token `<unk>` obsahuje 4458, to je 72 % z nich. Tedy téměř tři čtvrtiny regionů obsahují alespoň jeden symbol, který nebyl rozpoznán. Nejčastěji jde o chybu v OCR.

Textový obsah dokumentů tohoto datasetu je v německém jazyce. Z toho důvodu byl pro vyhodnocení textového obsahu natrénován samostatný jazykový model pro německy jazyk. Architektura modelu, velikost slovníku a další náležitosti jsou shodné s českým jazykovým modelem, který je představen v kapitole 6.1. Trénován byl na německém korpusu, který mi byl poskytnut vedoucím práce (velikost trénovací sady 576MB). Model na testovací sadě dosahuje perplexity 82,48 po 27 epochách.

6.3.2 Prezentace výsledků

Výsledky jednotlivých experimentů jsou zobrazeny ve formě grafů, které ukazují průměrné výsledky jednotlivých metod přes celý dataset. Pro přehlednost jsou metody v grafu uvedeny pod zkratkami.

- `lm`: jazyková analýza

[2] https://www.primaresearch.org/tools/Aletheia
[3] https://pero-ocr.fit.vutbr.cz/
[4] https://www.primaresearch.org/datasets/ENP

- `diag`: prostorová analýza, diagonální metoda
- `col`: prostorová analýza, sloupcová metoda
- `comb`: kombinovaná analýza

Vyhodnocení dále rozlišuje nastavení jazykového analyzátoru.

- `H`: odhad s pevným počtem tokenů
- `S`: odhad s využitím pravděpodobnosti kandidátní sekvence

Hodnoty ve zkratkách udávají počet tokenů, který byl pro dané nastavení použit. Například zkratka `comb-H-3` říká, že výsledek byl dosažen pomocí kombinované analýzy s pevným počtem tokenů, konkrétně se třemi tokeny.

6.4 Identifikace posloupnosti čtení

6.4.1 Dataset Hospodářských novin

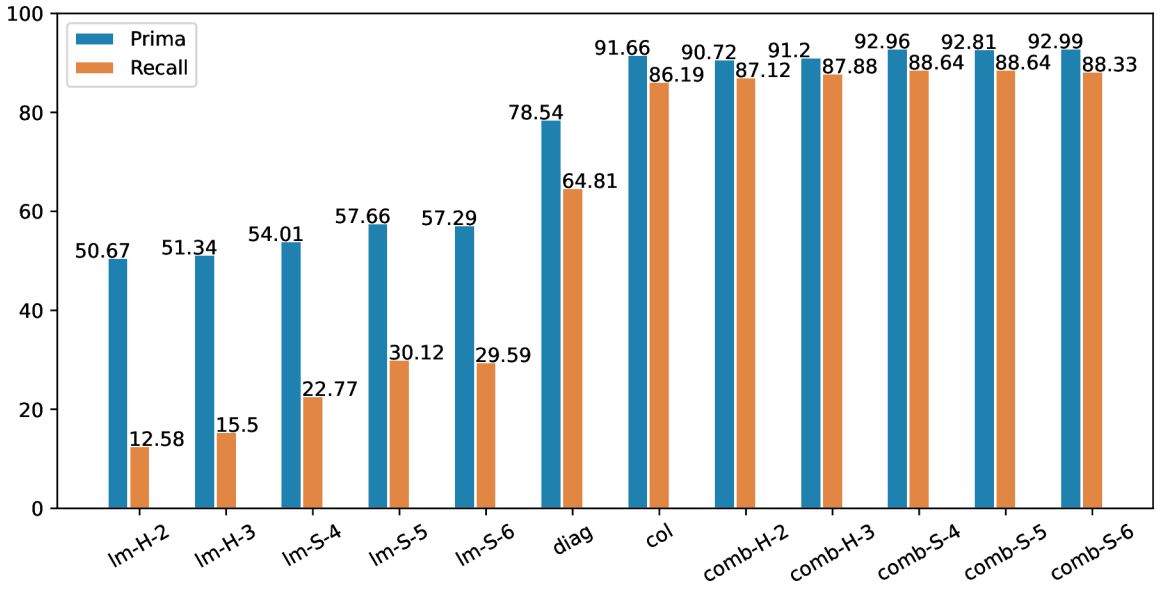
Prvním vyhodnoceným datasetem je dataset Hospodářských novin. Výsledek vyhodnocení je zobrazen na obrázku 6.5. Nejlepších výsledků dosáhla kombinovaná analýza s metodou odhadu pravděpodobností kandidáta, `comb-S-4`, respektive `comb-S-6`. Obě metody dosahují velmi podobných hodnot, hodnota metriky Prima dosahuje téměř 93 %. Recall se pohybuje okolo hodnoty 88,64 %, respektive 88,33 %. Rozdíl je minimální, avšak je pravděpodobně způsoben použitým rozsahem délek, kdy v případě `comb-S-4` jazykový model lépe korigoval výstup prostorové analýzy.

Je však potřeba říct, že takového výsledku je dosaženo především zásluhou prostorové sloupcové analýzy. Ta je součástí kombinované analýzy a její výstup je upraven právě jazykovým modelem. Samostatně dosahuje prostorová sloupcová analýza v Prima metrice hodnoty 91,66 %, a v Recall 86,19 %. Rozdíl mezi kombinovanou a prostorovou sloupcovou analýzou je 1–2 % dle metriky. Tento mírný posun je zásluhou jazykového modelu, který je v rámci kombinované analýzy použit ke korekci výstupu prostorové analýzy.

Samotná jazyková analýza dosahuje nejvyšší úspěšnosti v `lm-S-5`, 57,66 % metriky Prima a 30,12 % metriky Recall. Jazyková analýza obecně dosáhla nejnižších výsledků ze všech testovaných metod. To je způsobeno především proto, že v rámci jazykové analýzy jsou spolu porovnány i takové regiony, které spolu logicky nijak nesouvisí, například region na začátku stránky a region na konci stránky, případně regiony mezi dvěma nezávislými články. Jazykový model nemá k vyhodnocení žádné dodatečné informace a pracuje pouze s textovým obsahem. Oproti tomu kombinovaná analýza, která dosahuje nejlepších výsledků, těží z vlastnosti prostorové sloupcové analýzy, která je její součástí a která samostatně dosahuje obdobně dobrých výsledků. Jazykový model, který výstup koriguje, pak pracuje jen s těmi kandidáty, které jsou poblíž regionu představující inicializační sekvenci a tím je redukován počet možných kandidátů.

6.4.2 Dataset Lidových novin

Druhým vyhodnoceným datasetem je dataset Lidových novin. Výsledek vyhodnocení je vyobrazen na obrázku 6.6. Oproti vyhodnocení nad Hospodářskými novinami je v tomto případě vidět obecně znatelný pokles úspěšnosti. To může být způsobeno několika faktory.



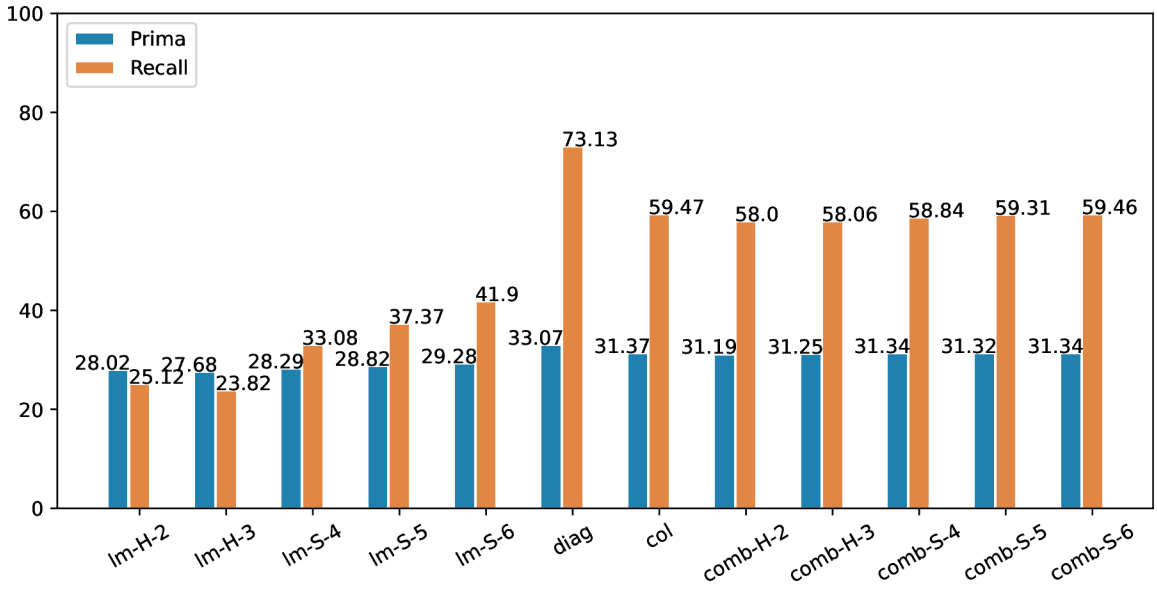
Obrázek 6.5: Vyhodnocení jednotlivých metod pro identifikaci posloupnosti čtení nad datasetem Hospodářských novin. Na ose x jsou uvedeny jednotlivé metody vyhodnocení, na ose y je procentuální hodnota úspěšnosti v dané metrice. Nejnižší úspěšnosti dosahuje jazyková analýza s dvěma tokeny. Nejlepších výsledků dosahuje kombinovaná analýza s využitím pravděpodobnosti kandidáta.

Prvním faktorem je, že jsou stránky tohoto datasetu mnohem rozmanitější. Na jedné stránce lze nalézt jak články, které mají definovanou posloupnost, tak také mnoho jiných elementů, jako například reklamy, které posloupnost definovanou nemají. Program nedokáže identifikovat nezávislé prvky, respektive tyto automaticky zahrne do posloupnosti s definovaným pořadím. To je také důvod razantního poklesu metriky Prima, která kontroluje nejen seřazené posloupnosti, ale také skupiny, které obsahují nezávislé elementy.

Druhým faktorem je užití specifických rozložení, například užití tvaru L pro postavení článku na stránce. V takovém případě je tok textu souvislý a zarovnaný se sloupcem. Ve svém závěru se však tok láme a zanořuje dovnitř stránky, kde je text shora i zespodu obalený nesouvisejícími elementy, například dalšími články. Z toho důvodu selhává dříve úspěšná prostorová sloupcová analýza, která nedokáže tvar L správně podchytit. Příklad článku ve tvaru L je znázorněn na obrázku 6.8.

Nejvyšší úspěšnosti dosahuje prostorová diagonální metoda `diag`. Ta v metrice Recall získala 73,13 %. V metrice Prima dosahuje hodnoty 33,07 %. V rámci této metriky se sice jedná o nejvyšší hodnotu, avšak celkově je tato hodnota velmi nízká. Především z důvodu existence nezávislých prvků na stránce, které však program automaticky zahrne do posloupnosti čtení. Stejný problém nastává u sloupcové metody `col`.

Dále následuje kombinovaná analýza s nejvyšším výsledkem v `comb-S-6`, kde dosahuje 31,34 % Prima a 59,46 % Recall. Oproti předchozímu experimentu se naopak zlepšila jazyková analýza, která v `lm-S-6` dosahuje sice nízké hodnoty Prima 29,28 %, především kvůli zařazení neuspořádaných elementu do posloupnosti, ale počet správně uhodnutých dvojic hodnocený metrikou Recall je 41,9 %. To tedy znamená, že v průměru byla, na tomto datasetu jazykovou analýzou, správně identifikována více jak třetina, respektive téměř polovina dvojic. To může být především z toho důvodu, že počet textových regionů na stránce je v tomto datasetu v průměru nejmenší ve srovnání s ostatními datasety a zároveň textové



Obrázek 6.6: Vyhodnocení jednotlivých metod pro identifikaci posloupnosti čtení nad datasetem Lidových novin. Na ose x jsou uvedeny jednotlivé metody vyhodnocení, na ose y je procentuální hodnota úspěšnosti v dané metrice.

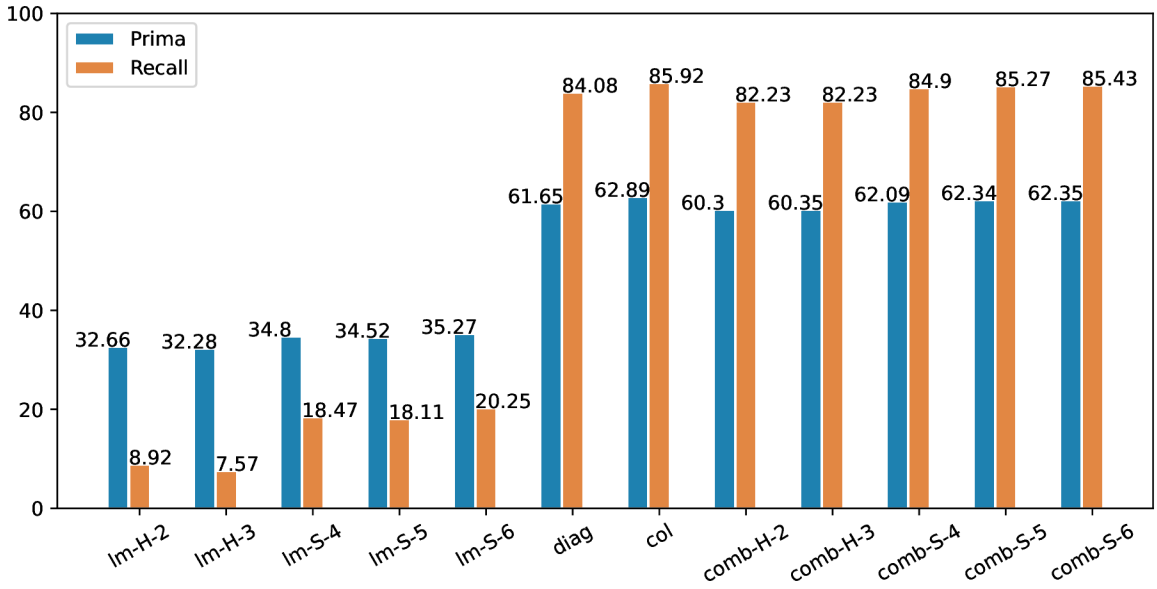
regiony, v porovnání s ostatními datasety, obsahují průměrně nejvíce textu. To má vliv na inicializaci skrytých stavů jazykového modelu, který poté může lépe odhadovat pravděpodobnosti jednotlivých kandidátů.

6.4.3 Europeana Newspapers Project Dataset

Posledním datasetem je dateset novin v německém jazyce, pro který byl natrénován samostatný model. Kromě odlišného jazykového modelu je proces experimentu shodný s předchozími. Výsledky experimentu jsou zobrazeny na obrázku 6.7.

Výsledek je podobný experimentu s Hospodářskými novinami. Úspěšnost jazykové analýzy není velká, nejvíce dosahuje `lm-S-6` 35,27 % v případě metriky Prima a 20,25 % v případě Recall a to při použití 6 tokenů. Avšak použitím prostorové analýzy se úspěšnost zvyšuje. Základní diagonální metoda `diag` dosahuje 61,65 % v metrice Prima a 84,08 % Recall. Důvodem nižší hodnoty metriky Prima je výskyt většího množství regionů, pro které není definovaná posloupnost. Zařazení takových regionů do posloupnosti čtení je ze strany metriky penalizováno. Na výpočet celkové chyby pak mají vliv všechny prvky zařazené v ground truth, tedy jak prvky v seřazené tak v neseřazené posloupnosti. Recall oproti tomu pracuje a vyhodnocuje identifikované prvky pouze proti prvkům v seřazené posloupnosti. To tedy znamená, že počet správně identifikovaných dvojic byl poměrně vysoký (vyšší Recall), ale zároveň bylo do posloupnost zahrnuto větší množství prvků, které jsou ve skutečnosti neuspořádané (nižší Prima).

Mírně vyššího výsledku než diagonální metoda, dosáhla prostorová sloupcová metoda `col`, 62,89 % Prima a 85,92 % Recall. Použitím kombinované analýzy naopak došlo k mírnému zhoršení. To může být způsobeno buď horší kvalitou sestrojeného jazykového modelu anebo kvalitou OCR výstupu dokumentu. Pokud by zhoršení bylo způsobeno kvalitou jazykového modelu, je možné jej dotrénovat na větším korpusu, případně na stávajícím korpusu provést více epoch. Kvalitu OCR výstupu však ovlivnit nelze. Nejlepšího výsledku v rámci



Obrázek 6.7: Vyhodnocení jednotlivých metod pro identifikaci posloupnosti čtení nad datasetem německých novin. Na ose x jsou uvedeny jednotlivé metody vyhodnocení, na ose y je procentuální hodnota úspěšnosti v dané metrice.

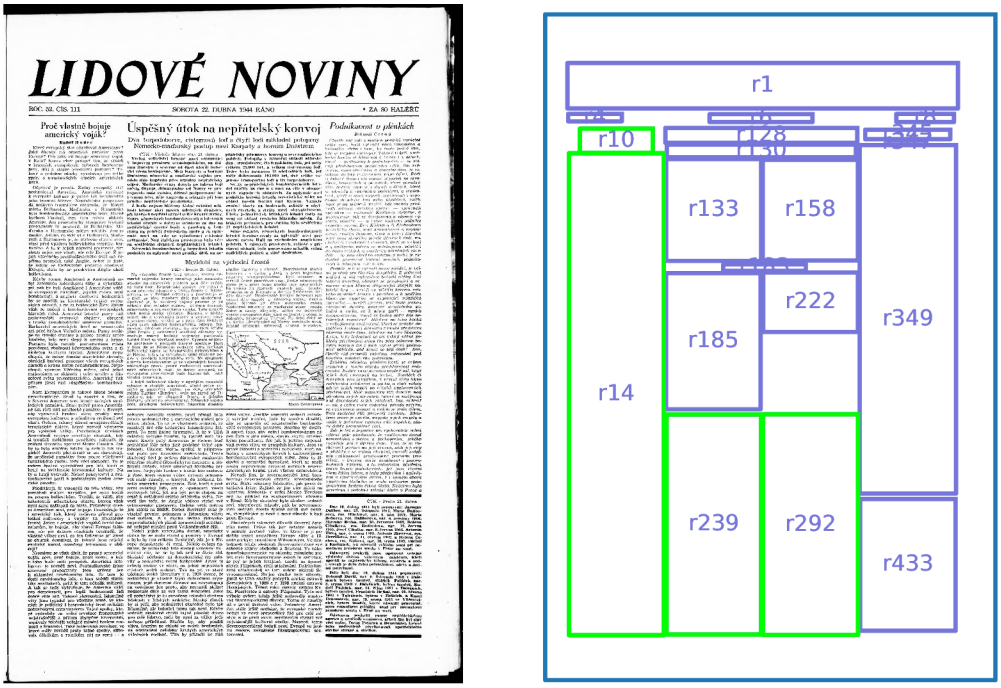
kombinované analýzy dosáhla metoda, která pracuje s pravděpodobností kandidátní sekvence a 6 tokeny, `comb-S-6`. Ta dosáhla hodnoty 62,35 % Prima a 85,43 % Recall, což je zhoršení vůči sloupcové analýze zhruba o půl procenta.

6.5 Zhodnocení

Experimenty ověřily úspěšnost jazykového modelu při identifikaci posloupnosti čtení a úspěšnost identifikace navrženými metodami. Nejnižší úspěšnost vykazuje jazyková analýza (`lm`), která proti ostatním dosahovala velmi nízkých hodnot. Důvod je však zřejmý. Jazykový model se řídí pouze textovým obsahem regionů. To nemusí být vždy dostačující, protože nejen v případě novinových článku se na stránce vyskytují různé elementy, například číslo stránky, jméno autora a podobně, u kterých nelze na základě jazykových vlastností správně odhadnout pořadí v posloupnosti. Protože jazyková analýza nemá k dispozici žádné další informace než textový obsah, porovnává pravděpodobnost posloupnosti i u takových prvků, které spolu logicky nijak nesouvisí. Jak však ukázal experiment s Lidovými novinami (sekce 6.4.2), úspěšnost identifikace posloupnosti čtení pomocí jazykové analýzy se zvyšuje, pokud textové regiony obsahuje větší množství textu. Množství textu však při analýze již vytvořeného dokumentu nelze nijak ovlivnit. Jazyková analýza proto dosahuje, v této oblasti, nejnižší úspěšnosti.

Prostorová analýza vykazuje mnohem lepší výsledky a to proto, že při identifikaci posloupnosti čtení pracuje především se svým okolím. V takovém případě se zvyšuje šance na správný odhad posloupnosti dvou prvku a snižuje počet regionů, mezi kterými je rozhodováno o posloupnosti. Kombinovaná analýza, která spojila výhody prostorové i jazykové analýzy, dosahuje podobných výsledků, jako prostorová analýza. Podstatná část úspěšnosti kombinované analýzy však závisí na výstupu prostorové analýzy, která je posléze korigována jazykovým modelem.

V konečném důsledku lze tedy říci, že použití samostatného jazykového modelu pro určení posloupnosti čtení bez dalších doprovodných informací, na základě kterých by se jazykový model lépe rozhodoval, není vhodné. Alespoň ne v případě analýzy novinových stránek, kterými se tato práce v převážné většině zabývala. Vhodnost jazykové analýzy pro jiné typy dokumentů nelze vyloučit, avšak lze se domnívat, že úspěšnost analýzy by i na jiných typech dokumentů byla nízká, pokud bychom k rozhodování použily pouze textový obsah a nevyužili bychom další užitečná data, jako například prostorové informace.



(a) Digitalizovaná stránka (b) Znázornění článku ve tvaru L

Obrázek 6.8: Příklad novinové stránky obsahující článek, který má tvar písmene L. V některých případech může být článek takového tvaru ohraničen jiným článkem nejenom shora, ale také zespodu.

Kapitola 7

Závěr

Cílem práce bylo seznámit se se statistickým jazykovým modelováním a s problematikou uspořádávání fragmentů textu ve výstupu OCR, navrhnout postup, jak pro uspořádávání využít jazykový model a experimentálně ověřit jeho účinnost.

V rámci práce jsem sestrojil jazykový model a ověřil a popsal jeho chování, především úspěšnost odhadu pravděpodobností na různě dlouhých textových sekvencí. Dále jsem navrhl a implementoval celkem tři metody pro identifikaci posloupnosti čtení. Jsou jimi jazyková analýza, prostorová analýza a kombinovaná analýza.

Jazyková analýza pracuje s jazykovým modelem a s textovým obsahem regionů. Provádí analýzu nad jednotlivými regiony dokumentu a dle nejvyšší pravděpodobnosti postupně tyto regiony spojuje, čímž vytváří posloupnost čtení. Prostorová analýza definuje posloupnost na základě prostorových dat regionů. Vytváří strukturu prostorových vztahů, pomocí kterých pro každé dva prvky identifikuje jejich vzájemný vztah, který je využit při sestavení posloupnosti čtení. Kombinovaná analýza pracuje jak s textovým obsahem, tak s prostorovými informacemi regionů. Nejprve provede prostorovou analýzu, jejíž výstup je následně úpraven pomocí jazykového modelu.

Úspěšnost jsem experimentálně ověřil na třech datasetech s pomocí metrik Prima a Recall. Obsah datasetů sestával primárně z novinových stránek, které jsou kvůli své struktuře a rozložení jedním z nejsložitějších formátů pro identifikaci posloupnosti čtení. Nejnižší úspěšnosti na všech třech datasetech dosáhla jazyková analýza, která získala maximum 57,6 % v metrice Prima. Lepší výsledky vykazovala prostorová analýza, která dosáhla maxima 91,6 %. Obdobných výsledků dosáhla kombinovaná analýza, která získala 92,9 %. Z těchto hodnot lze odvodit, že použití samotného jazykového modelu pro odhad posloupnosti čtení není vhodné, pokud rozhodování modelu není podpořeno dalšími podpůrnými informacemi. Toto se týká především rozhodování posloupnosti nad novinovými stránkami, kterými se tato práce převážně zabývala.

Práci jsem publikoval na studentské konferenci Excel@FIT 2022, kde získala ocenění odborným panelem [22].

Literatura

[1] Afzal, M. Z., Capobianco, S., Malik, M. I., Marinai, S., Breuel, T. M. et al. Deepdocclassifier: Document classification with deep Convolutional Neural Network. In: *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. 2015, s. 1111–1115. DOI: 10.1109/ICDAR.2015.7333933.

[2] Agrawal, M. a Doermann, D. Context-Aware and Content-Based Dynamic Voronoi Page Segmentation. In: *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*. New York, NY, USA: Association for Computing Machinery, 2010, s. 73–80. DAS '10. DOI: 10.1145/1815330.1815340. ISBN 9781605587738. Dostupné z: https://doi.org/10.1145/1815330.1815340.

[3] Aiello, M., Monz, C., Todoran, L. a Worring, M. Document Understanding for a Broad Class of Documents. *International Journal on Document Analysis and Recognition*. Srpen 2002, 5(1). DOI: 10.1007/s10032-002-0080-x.

[4] Arisoy, E., Sainath, T. N., Kingsbury, B. a Ramabhadran, B. Deep Neural Network Language Models. In:. USA: Association for Computational Linguistics, 2012, s. 20–28. WLM '12.

[5] Bengio, Y., Ducharme, R., Vincent, P. a Janvin, C. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.* JMLR.org. mar 2003, sv. 3, s. 1137–1155. ISSN 1532-4435.

[6] Bodén, M. A Guide to Recurrent Neural Networks and Backpropagation. Prosinec 2001.

[7] Bottou, L. et al. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*. Nimes. 1991, sv. 91, č. 8, s. 12.

[8] Cesarini, F., Lastri, M., Marinai, S. a Soda, G. Page classification for meta-data extraction from digital collections. In: Springer. *International Conference on Database and Expert Systems Applications*. 2001, s. 82–91.

[9] Chaudhuri, A., Mandaviya, K., Badelia, P. a Ghosh, S. K. *Optical Character Recognition Systems for Different Languages with Soft Computing*. 1st. Springer Publishing Company, Incorporated, 2016. ISBN 3319502514.

[10] Chen, S. F. a Goodman, J. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*. 1999, sv. 13, č. 4, s. 359–394. DOI: https://doi.org/10.1006/csla.1999.0128. ISSN 0885-2308. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0885230899901286.

[11] Clausner, C., Pletschacher, S. a Antonacopoulos, A. The Significance of Reading Order in Document Recognition and Its Evaluation. In:. Srpen 2013. DOI: 10.1109/ICDAR.2013.141.

[12] Doermann, D. a Tombre, K. *Handbook of Document Image Processing and Recognition*. Springer Publishing Company, Incorporated, 2014. ISBN 0857298585.

[13] Eikvil, L. Optical Character Recognition. In:. 1993. Dostupné z: http://home.nr.no/~eikvil/OCR.pdf.

[14] Ferilli, S., Grieco, D., Redavid, D. a Esposito, F. Abstract Argumentation for Reading Order Detection. In: *Proceedings of the 2014 ACM Symposium on Document Engineering*. New York, NY, USA: Association for Computing Machinery, 2014, s. 45–48. DocEng '14. DOI: 10.1145/2644866.2644883. ISBN 9781450329491. Dostupné z: https://doi.org/10.1145/2644866.2644883.

[15] Gao, L., Tang, Z., Lin, X., Liu, Y., Qiu, R. et al. Structure Extraction from PDF-Based Book Documents. In:. New York, NY, USA: Association for Computing Machinery, 2011, s. 11–20. JCDL '11. DOI: 10.1145/1998076.1998079. ISBN 9781450307444. Dostupné z: https://doi.org/10.1145/1998076.1998079.

[16] George, N. a Sharad, S. Hierarchical representation of optically scanned documents. In:. 1984, s. 347 – 349.

[17] Goodfellow, I., Bengio, Y. a Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[18] Goodman, J. A Bit of Progress in Language Modeling. *CoRR*. 2001, cs.CL/0108005. Dostupné z: https://arxiv.org/abs/cs/0108005.

[19] Graves, A. Supervised Sequence Labelling with Recurrent Neural Networks. In: *Studies in Computational Intelligence*. 2008.

[20] Hinton, G., Srivastava, N. a Swersky, K. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*. 2012, sv. 14, č. 8, s. 2.

[21] Hochreiter, S. a Schmidhuber, J. Long Short-term Memory. *Neural computation*. Prosinec 1997, sv. 9, s. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[22] Holubec, M. Uspořádání zpřeházených řádků s pomocí jazykového modelu. *Excel@FIT*. 2022. Dostupné z: https://excel.fit.vutbr.cz/submissions/2022/023/23.pdf.

[23] Hu, J., Kashi, R. a Wilfong, G. Comparison and Classification of Documents Based on Layout Similarity. *Information Retrieval*. Prosinec 1999, sv. 2. DOI: 10.1023/A:1009910911387.

[24] ISHITANI, Y. Document Transformation System from Papers to XML Data Based on Pivot XML Document Method. In: *Pivot XML Document Method, International conference on document analysis and recognition, ICDAR 2003*. 2003, s. 250–255.

[25] Ittner, D. a Baird, H. Language-free layout analysis. In: *Proceedings of 2nd International Conference on Document Analysis and Recognition (ICDAR '93)*. 1993, s. 336–340. DOI: 10.1109/ICDAR.1993.395720.

[26] Kingma, D. P. a Ba, J. Adam: A method for stochastic optimization. *ArXiv preprint arXiv:1412.6980*. 2014.

[27] Kise, K., Sato, A. a Iwata, M. Segmentation of Page Images Using the Area Voronoi Diagram. *Computer Vision and Image Understanding*. 1998, sv. 70, č. 3, s. 370–382. DOI: https://doi.org/10.1006/cviu.1998.0684. ISSN 1077-3142. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1077314298906841.

[28] Kudo, T. a Richardson, J. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Brussels, Belgium: Association for Computational Linguistics, Listopad 2018, s. 66–71. DOI: 10.18653/v1/D18-2012. Dostupné z: https://aclanthology.org/D18-2012.

[29] Li, X.-H., Yin, F. a Liu, C.-L. Page Object Detection from PDF Document Images by Deep Structured Prediction and Supervised Clustering. In: *2018 24th International Conference on Pattern Recognition (ICPR)*. 2018, s. 3627–3632. DOI: 10.1109/ICPR.2018.8546073.

[30] Malerba, D., Ceci, M. a Berardi, M. Machine Learning for Reading Order Detection in Document Image Understanding. In:. Prosinec 2007, sv. 90, s. 45–69. DOI: 10.1007/978-3-540-76280-5_3. ISBN 978-3-540-76279-9.

[31] Meunier, J.-L. Optimized XY-cut for determining a page reading order. In: *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*. 2005, s. 347–351 Vol. 1. DOI: 10.1109/ICDAR.2005.182.

[32] Mikolov, T., Chen, K., Corrado, G. a Dean, J. Efficient Estimation of Word Representations in Vector Space. *Proceedings of Workshop at ICLR*. Leden 2013, sv. 2013.

[33] Mikolov, T. *Modelování jazyka v rozpoznávání češtiny*. Brno, CZ, 2007. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/3645/.

[34] Mikolov, T. *Statistical language models based on neural networks*. Brno, CZ, 2012. Disertační práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/phd-thesis/283/.

[35] Muda, N., Ismail, N. K. N., Bakar, S. A. A. a Zain, J. M. Optical character recognition by using template matching (alphabet). In: *National Conference on Software Engineering & Computer Systems 2007 (NACES 2007)*. 2007.

[36] O'Gorman, L. The document spectrum for page layout analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1993, sv. 15, č. 11, s. 1162–1173. DOI: 10.1109/34.244677.

[37] Olah, C. *Understanding LSTM Networks* [online]. 2015 [cit. 2021-12-28]. Dostupné z: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[38] Pai, A. *What is Tokenization in NLP? Here's All You Need To Know* [online]. 2020 [cit. 2022-04-11]. Dostupné z: https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/.

[39] Pennington, J., Socher, R. a Manning, C. Glove: Global Vectors for Word Representation. In:. Leden 2014, sv. 14, s. 1532–1543. DOI: 10.3115/v1/D14-1162.

[40] Rangoni, Y., Belaïd, A. a Vajda, S. Labelling logical structures of document images using a dynamic perceptive neural network. *IJDAR*. Březen 2012, sv. 15. DOI: 10.1007/s10032-011-0151-y.

[41] Ruder, S. An overview of gradient descent optimization algorithms. *CoRR*. 2016, abs/1609.04747. Dostupné z: http://arxiv.org/abs/1609.04747.

[42] Srihari, S. N., Shekhawat, A. a Lam, S. W. Optical Character Recognition (OCR). In:. GBR: John Wiley and Sons Ltd., 2003, s. 1326–1333. ISBN 0470864125.

[43] Wei, H., Baechler, M., Slimane, F. a Ingold, R. Evaluation of SVM, MLP and GMM Classifiers for Layout Analysis of Historical Documents. In: *2013 12th International Conference on Document Analysis and Recognition*. 2013, s. 1220–1224. DOI: 10.1109/ICDAR.2013.247.

[44] Wong, K. Y., Casey, R. G. a Wahl, F. M. Document Analysis System. *IBM Journal of Research and Development*. 1982, sv. 26, č. 6, s. 647–656. DOI: 10.1147/rd.266.0647.

[45] Zulfiqar, A., Ul Hasan, A. a Shafait, F. Logical Layout Analysis using Deep Learning. In:. Prosinec 2019, s. 1–5. DOI: 10.1109/DICTA47822.2019.8946046.