VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

VIZUÁLNÍ SEGMENTACE WEBOVÝCH STRÁNEK

VISION-BASED WEB PAGE SEGMENTATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. FRANTIŠEK MAŠTERA**
AUTHOR

VEDOUCÍ PRÁCE **doc. Ing. RADEK BURGET, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání diplomové práce

144821

Ústav:	Ústav informačních systémů (UIFS)
Student:	**Maštera František, Bc.**
Program:	Informační technologie a umělá inteligence
Specializace:	Softwarové inženýrství
Název:	**Vizuální segmentace webových stránek**
Kategorie:	Web
Akademický rok:	2022/23

Zadání:

1. Seznamte se s experimentálním nástrojem FitLayout, jeho aplikačním rozhraním a způsobem reprezentace dokumentů.
2. Prostudujte současné metody segmentace webových stránek se zaměřením na metody analyzující vizuální prezentaci informací na stránce.
3. Po dohodě s vedoucím zvolte konkrétní publikovanou metodu segmentace webových stránek a navrhněte způsob její integrace do nástroje FitLayout.
4. Implementujte navržené rozšíření pomocí vhodných technologií.
5. Proveďte experimentální testování segmentace na vhodné množině webových stránek.
6. Zhodnoťte dosažené výsledky.

Literatura:

- Zeleny, J.; Burget, R.; Zendulka, J.: Box clustering segmentation: A new method for vision-based web page preprocessing. Information Processing & Management. vol. 53, no. 3. 2017: pp. 735 - 750. ISSN 0306-4573.
- Cai, D.; Yu, S.; Wen, J.-R.; et al.: VIPS: a Vision-based Page Segmentation Algorithm. Microsoft Research. 2003.
- Dokumentace projektu FitLayout https://github.com/FitLayout

Při obhajobě semestrální části projektu je požadováno:
Body 1 až 3

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

Vedoucí práce:	**Burget Radek, doc. Ing., Ph.D.**
Vedoucí ústavu:	Kolář Dušan, doc. Dr. Ing.
Datum zadání:	1.11.2022
Termín pro odevzdání:	17.5.2023
Datum schválení:	24.10.2022

Abstrakt

Knihovna FitLayout nabízí sadu implementovaných metod segmentace webových stránek spolu s řadou nástrojů k jejich vyhodnocení a dalšímu vývoji. Cílem této práce je rozšíření této knihovny o další implementaci již existující metody. Pro splnění tohoto cíle byla zvolena a následně integrována metoda Cormier et al. Věrohodnost její implementace oproti její publikaci byla řádně ověřena. Rovněž bylo provedeno její rozsáhlé vyhodnocení s cílem zjištění jejích vlastností a chování za různých okolností, v jehož rámci byly objeveny nastavení metody, které zlepšují kvalitu jejích výstupů na testovaném vzorku dat až o 9,89 %. Výsledkem této práce je rozšíření knihovny FitLayout o novou metodu segmentace webových stránek, která může být zahrnuta v rámci dalšího výzkumu v této oblasti, který může být založen na výsledcích dosažených v této práci.

Abstract

The FitLayout library offers a suite of implemented web page segmentation algorithms along with a number of tools for their evaluation and further development. The goal of this thesis is to extend this suite by another of already existing algorithms. To meet this goal, the Cormier et al. algorithm was chosen and integrated into the FitLayout. The plausibility of its implementation against its publication has been duly verified. Its extensive evaluation was also carried out to determine its properties and behaviour under different circumstances, which revealed algorithm settings that improve the quality of its outputs on the tested data sample by up to 9.89 %. As a result of this thesis, the FitLayout library has been extended with a new web page segmentation algorithm, which can be used in further research in this area that can be supported with the results found in this thesis.

Klíčová slova

segmentace webových stránek, FitLayout, vizuální analýza dokumentů, detekce hran

Keywords

web page segmentation, FitLayout, visual document analysis, edge detection

Citace

MAŠTERA, František. *Vizuální segmentace webových stránek*. Brno, 2023. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. Radek Burget, Ph.D.

Vizuální segmentace webových stránek

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana doc. Ing. Radka Burgeta, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
František Maštera
15. května 2023

Poděkování

Rád bych poděkoval vedoucímu této práce, doc. Ing. Radkovi Burgetovi, Ph.D., za odbornou pomoc a vedení při její tvorbě.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Webové stránky představují rozsáhlý zdroj dat, jejich strojové zpracování je ovšem z mnoha důvodů značně problematické. Problematika segmentace webových stránek přichází s přístupem rozdělení stránek na sémanticky ucelené bloky. Toto rozdělení na jemnější jednotky informace usnadňuje práci s daty obsaženými na webu a jejich extrakci. Nejedná se ovšem o jediné možné využití segmentace – jejích výsledků lze využít také např. při tvorbě nástrojů pro přístupnost webového obsahu zrakově postiženým, zlepšení výsledků vyhledávání, či anotaci obsahu stránek.

Existuje mnoho druhů přístupů, jak stránky segmentovat a neustále jsou publikovány další. Tyto různé přístupy produkují různé výsledky pro různé webové stránky a každý přístup má svoje výhody a nevýhody. Žádná z metod tedy nepředstavuje univerzální řešení, které lze použít pro libovolný účel a je nutné počítat s jejich vlastnostmi, které mohou omezit výběr metody pro různé případy užití.

Nástroj FitLayout[1] pak představuje mimo jiné rámec pro analýzu těchto metod poskytováním podpůrných nástrojů k testování a experimentování jako je vykreslování stránky, vizualizace výsledků segmentace či tvorbě perzistentní datové sady z neustále se měnících stránek.

Tento nástroj také poskytuje několik již implementovaných metod, které lze díky tomu analyzovat přímo v rámci nástroje FitLayout. Cílem této práce je zapojení do vývoje tohoto nástroje a jeho rozšíření o implementaci zvolené metody segmentace a její následné experimentální vyhodnocení.

Účelem následující kapitoly je úvod do oblasti segmentace webových stránek – představení motivace a základního principu zpracování obsahu stránky, následované popisem principu metod segmentace, přehledem jejich různých přístupů a možností využití. V kapitole 3 je představen nástroj FitLayout, jeho architektura a jím nabízené nástroje pro segmentaci. Práce dále pokračuje kapitolou 4 popisující metodu segmentace zvolenou k implementaci. Následuje kapitola 5 obsahující ověření a navržení integrace metody do nástroje FitLayout, na kterou kapitola 6 navazuje konkrétním popisem implementačního řešení integrace. Stěžejní část práce pak představuje kapitola 7, která obsahuje způsob testování implementace, následnou analýzu jejího chování a představuje výsledky experimentů, jejichž cílem bylo zlepšení kvality výstupu metody pomocí úpravy jejích parametrů.

[1] FitLayout – https://github.com/FitLayout

Kapitola 2

Segmentace webových stránek

Kapitola obsahuje teoretický úvod do problematiky zpracování obsahu webových stránek a s ní spojené segmentace webových stránek. V sekci 2.1 je shrnutý potenciál stránek jako zdroje dat a jejich problémy, dále v 2.2 pak úvod do zpracování jejich obsahu. Sekce 2.3 představuje proces segmentace webových stránek spolu s několika příklady. Různé přístupy k procesu jsou pak shrnuty v sekci 2.4 a nakonec je kapitola zakončena možnými aplikacemi segmentace, které jsou uvedeny v sekci 2.5.

2.1 Webové stránky jako zdroj informací

Webové stránky jsou rozsáhlým, distribuovaným a globálním zdrojem zejména textových informací, ať už jde např. o zprávy, reklamu, zákony, elektronické obchody, nebo o příspěvky na sociálních sítích. Právě kvůli množství obsahu jde o cenný zdroj pro získávání znalostí z dat.

Zdaleka ovšem nejde o ideální zdroj dat. Jedním z problémů jakéhokoliv zpracování dat z webových stránek je jejich rozsah. Množství informací ve webových stránkách je příliš velké pro efektivní reprezentaci a ukládání v databázi, je proto potřeba vybrat nějakou vhodnou podmnožinu. [5]

Dalším velkým problémem je struktura informací ve webových stránkách. V podstatě jediná pravidla struktury, podle které se musí autoři webových stránek řídit, jsou dána značkovacím jazykem HTML [24]. Obsah ve webových stránkách je navíc díky neustálému vývoji ve tvorbě webových stránek postupem času, zejména v posledních letech, více sofistikovaný [16].

Způsob prezentování stránek je pak navržen takovým způsobem, aby byly primárně dobře zpracovatelné lidským zrakem, tj. se zaměřením na jejich vizuální aspekt. Cílem je, aby se v nich uživatel dobře orientoval a aby ho zaujaly ty informace na stránce, které autor stránky chce uživateli prezentovat. K tomu jsou využité různé vizuální prvky, které takové informace oddělují, zvýrazňují nebo na ně upozorňují různými dalšími způsoby. Zpracování strojovým způsobem má pak často při tvorbě stránek nižší prioritu, nebo není bráno v potaz vůbec. [9]

Ve webových stránkách se navíc často nenachází pouze informace, které chceme v daném kontextu získat. Může obsahovat také reklamy, nepotřebné informace v zápatí stránky, nebo menu s odkazy na další stránky. Jedna stránka může také obsahovat informace týkající se vícero témat, která mohou být v daném kontextu úlohy různě důležitá. [24]

2.2 Zpracování obsahu webové stránky

Klíčovou úlohou je identifikace obsahu stránky, která vzhledem k problémům zmíněným výše není vůbec jednoduchá. Je k ní potřeba se orientovat v různorodé struktuře webových stránek a v informacích v ní, které mohou být vztaženy k více tématům, případně nás některé z nich často nemusí vůbec zajímat.

K úloze lze přistoupit z pohledu struktury různými způsoby. Prvním je převážné ignorování struktury stránky a zaměření na její obsah, tj. na stránku je pohlíženo jako na textový dokument a jsou na ni aplikovány postupy z oblasti dolování z textu. To ovšem není ideální řešení. Přestože struktura webových stránek není ideální, poskytuje svým způsobem také nějakou informaci o datech obsažených ve stránce. Na samotnou strukturu webové stránky se ovšem také často nelze spolehnout. Typickým přístupem je proto kombinace obou přístupů. [5]

Informace o struktuře webové stránky ale také nemusí být dostačující. Jak již bylo zmíněno, stránky jsou prezentovány především vizuálně. Zpracováním HTML kódu stránky lze zjistit informace jako nadpisy, odstavce, odkazy apod., nikoliv ovšem vizuální atributy těchto prvků a jejich okolí, které mohou být velmi důležité. Například pro odlišení odkazů v menu oproti odkazům výrobku v internetovém obchodě. Obojí je odkaz na nějakou stránku, který je nějakým řetězcem, ale autor stránky bude chtít spíše upoutat pozornost uživatele na výrobky a tím pádem bude odkaz na výrobek větší a výraznější. Naopak odkaz v menu bude součástí bloku na boku stránky, typicky nějakým způsobem ohraničeného – třeba jinou barvou pozadí. [6]

2.3 Segmentace obsahu na sémantické bloky

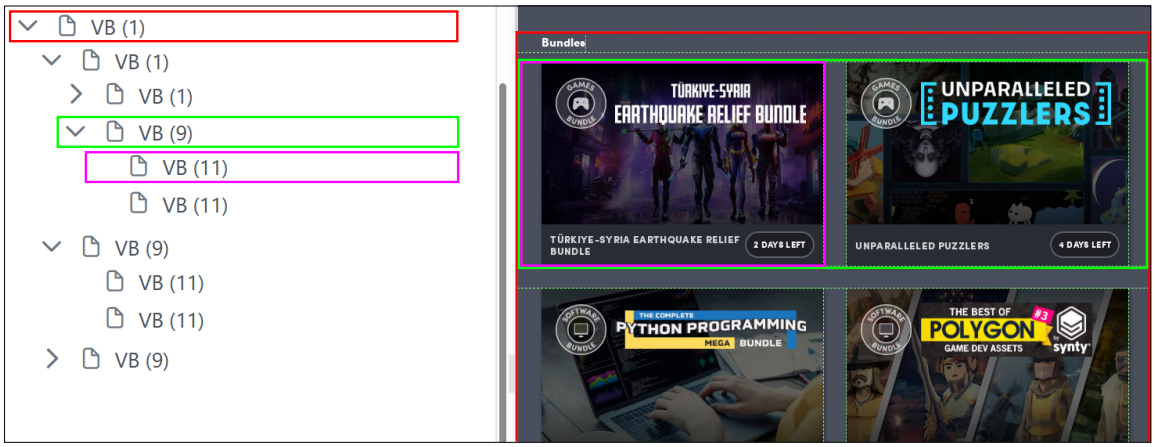
Vizuální prvky lze také využít k rozdělení obsahu webové stránky na bloky oddělené různými vizuálními prvky (jinými odstíny pozadí, oddělovači atd.). Tento proces se nazývá segmentace webové stránky a k nalezení těchto bloků je možné využít i jiné informace o webové stránce, jako její struktura nebo obsah. Typickými bloky může být např. záhlaví, nebo navigační menu. Příklad možného výsledku segmentace lze vidět na obrázku 2.1.

Takové bloky jsou obdobným principem typicky dělitelné na další, menší bloky. Celková struktura sémantických bloků, ze kterých je stránka sestavena, pak tvoří hierarchickou stromovou strukturu, kde každý uzel reprezentuje jeden segment vytvořený segmentační metodou. Kořenový uzel představuje typicky celou stránku a listové uzly pak segmenty nejmenší granularity, které už nejsou danou metodou dělitelné na menší segmenty. Příklad takové struktury lze pozorovat na obrázku 2.1.

Zpracování obsahu takové jednotky je pak jednodušší než zpracování celé webové stránky najednou. Bloky na rozdíl od celé stránky obsahují typicky souvislý obsah s užším zaměřením a jednotnější důležitostí v kontextu dané úlohy. Jednotlivým blokům lze také přidělit různé stupně důležitosti, typicky na základě umístění – například zápatí nebude důležité, zatímco bloky uprostřed stránky budou velmi pravděpodobně obsahovat hlavní informace dané stránky. [6]

Segmentace stránky na sémanticky oddělené bloky je poměrně jednoduchým úkolem pro člověka, kterému je prezentace stránky určena a který dokáže odhalit její logiku. Syntaktická struktura stránky bez jasně daného syntaktického popisu je ovšem výzvou pro algoritmy segmentace, které musí logiku nějakým jasně daným způsobem odhadnout. [16]

[3] https://www.humblebundle.com/
[4] FitLayout – https://github.com/FitLayout



Obrázek 2.1: Příklad segmentace části webové stránky[3] na sémantické bloky tvořící hierarchickou strukturu, vytvořené metodou VIPS [9] a následně vizualizované ve webovém rozhraní nástroje FitLayout[4] s doplněným zvýrazněním vybraných zanořených segmentů.

Pro vytvoření bloků lze samozřejmě použít také strukturu jazyka HTML formující DOM[5] extrakcí strukturálních značek. Jde ovšem o reprezentaci, která není určena uživateli a nemusí vždy odrážet sémantickou strukturu stránky. Může se tak stát, že syntaktické bloky vytvořené tímto způsobem mohou seskupovat elementy, které dávají dohromady smysl ve stromové struktuře, zatímco významově nikoliv. [6]

2.4 Různé přístupy k segmentaci

Je publikováno mnoho metod řešení segmentace webových stránek, které k problematice přistupují různými způsoby. Ty lze dělit na základě různých kritérií, jako např. podle různých pohledů na obsah stránky, nebo dle postupu samotného procesu segmentace.

První rozdělení, které lze aplikovat, je podle toho, kterým způsobem metody používají informace na webové stránce k segmentaci:

- Čistě vizuálního aspektu využívá metoda implementovaná v rámci této práce – [13] (podrobněji popsána později v kapitole 4). Výhody tohoto přístupu byly v této kapitole již popsány.
- Velkou skupinu tvoří metody, které využívají logickou strukturu DOM uzlů stránky. Např. metoda [9] k rekurzivnímu dělení segmentů jak informace o struktuře stránky, tak její vizuální prvky. Průběžně pak jednotlivým segmentům přiděluje hodnotu, která závisí na konzistenci vizuálních prvků uvnitř daného segmentu, na základě které vybírá segmenty do výsledného výstupu.
- Na textový obsah se zaměřuje metoda publikovaná v [19]. Ta postupně kombinuje bloky do větších, kdy jako hlavní rozhodovací parametr pro sjednocení používá srovnání hustoty textu uvnitř sousedních bloků.

Na metody se lze dívat odlišným způsobem i podle postupu formování bloků:

[5]DOM – Document Object Model (objektový model dokumentu)

- *Top-down* postup, kdy je proces segmentace jako postupný rozklad celé stránky na menší a menší bloky, tvořící hierarchickou stromovou strukturu. Příkladem může být metoda implementovaná v rámci této práce [13], která na základě pravděpodobnosti výskytu hranice dělící segmenty postupně půlí existující bloky na menší.

- *Bottom-up* je zcela opačný přístup, jehož principem je postupné formování bloků spojováním menších. Publikovaná metoda [3] má fixně daný výsledný počet segmentů, k němuž se dostane rozprostřením počátečních bodů podle různých strategií, ke kterým postupně přiřazuje hranice podle vzdálenosti, zarovnání nebo vizuální podobnosti na základě CSS[6] atributů.

- Publikovány jsou ovšem i metody, které mají vlastní přístup, který nelze jednoduše zařadit ani do jedné z těchto dvou kategorií. Např. [16] staví na shlukovacím algoritmu K-means a využívá evolučních postupů pro stanovení počtu segmentů a jejich iniciálních pozic.

Některé metody jsou pak použitelné přímo za běhu. Např. [19] nevyžaduje žádné komplexní předzpracování stránky, díky čemuž je metoda velmi rychlá. Jiným metodám může zabrat zpracování a rozdělení obsahu delší dobu a nejsou tak vhodné pro časově kritičtější případy užití. Typicky u složitějších algoritmů [1] nebo při vizuálním zpracování – [13] může díky potřebnému vizuálnímu předzpracování větší stránky na modernějších procesorech zpracovávat až hodinu. [18]

Pro optimální výsledky některých metod je potřebné nastavení parametrů [1], zatímco u jiných nikoliv [11]. Metody mohou používat různé nové přístupy zamýšlené přímo k segmentaci webových stránek [13], nebo mohou problém segmentace namapovat na již osvědčený teoretický základ metod shlukování [1] nebo i grafových algoritmů [11].

K segmentaci lze v neposlední řade také využít přístupu učení s učitelem. Obecným problémem u těchto metod je ovšem jejich potřeba velké datové sady manuálně nasegmentovaných stránek k naučení, s ideálně co největším pokrytím různých přístupů k designu stránek [11] [23]. Výsledkem jsou pak často horší výsledky pro nové stránky. [16]

2.5 Možnosti využití

Výslednou segmentaci webové stránky na sémantické bloky lze využít např. pro vstup klasifikačních algoritmů. Klasifikovat lze jednotlivé bloky pro lepší určení důležitosti. Této informace lze pak využít jako vstup pro klasifikaci celých webových stránek a zlepšit tím její přesnost nastavením váhy trénovacích dat podle důležitosti bloků. [15] dělí klasifikaci do dvou fází. V první klasifikuje bloky získané vizuální segmentací. V té druhé pak klasifikuje celé stránky s využitím informací získaných z první fáze a na základě metod dolování textu v jednotlivých blocích.

Bloky lze ale využít i k dalším účelům. Např. anotace obrázků dosahuje lepších výsledků, pokud je extrahována z textu, který k obrázku skutečně patří. Webové stránky navržené pro velké obrazovky mohou být pomocí segmentace efektivněji prezentovány na zařízení s menšími obrazovkami. Lze také dosáhnout zlepšení přístupnosti stránek odstraněním obsahu, který by mohl odvádět pozornost uživatele od důležitých částí stránky. [16]

Segmentace může také přispět ke zlepšení použitelnosti internetu pro nevidomé uživatele. Ti potřebují stránky interpretovat nějakým jiným způsobem než vizuálním, na který

[6] CSS - Cascading Style Sheets (kaskádové styly)

stránky většinou spoléhají. Typicky k tomuto slouží nástroje pro předčítání obsahu stránek. Hlavním problémem tohoto řešení je ovšem ztráta dimenze při tomto přechodu – vizuální informace má dva rozměry, zatímco zvuková stopa je jediným proudem dat, který navíc nedovoluje jednoduše „nahlédnout“ a pochopit tak strukturu stránky v řádu sekund. Je proto velmi důležité, aby předčítaný obsah odrážel co nejpodstatnější informace z dané stránky. Pro tento účel existují prostředky, jako jsou např. WAI-ARIA štítky poskytující možnost notace elementů stránky a tím doplnění jejich sémantiky pro nástroje k předčítání. Bohužel mnoho stránek takových prostředků nevyužívá, nebo je jejich nasazení neefektivní. Pro extrakci těchto informací tak může sloužit právě segmentace webových stránek, která nalezne bloky stránky s podstatným obsahem, na který se nástroje pro předčítání mohou zaměřit. [13]

Publikace [4] přichází s rámcem pro web crawling[7], který dokáže na základě výsledků segmentace určit významné sémantické bloky a tím usnadnit a zlepšit efektivitu prohledávání webu.

[7] web crawling – Prohledávání webu za účelem vytvoření velké databáze dat

Kapitola 3

FitLayout

Jak je zmíněno v kapitole 2, webové stránky jsou mířeny primárně na uživatele, kteří k nim přistupují skrze webové prohlížeče. Tento způsob je pak často jediný, jak spolehlivě získat úplnou reprezentaci webové stránky tak, jak ji autor zamýšlel. S tím musí počítat také metody analýzy webových stránek, včetně metod jejich segmentace.

Dalším nepříjemným problémem, se kterým se metody analýzy webových stránek potýkají, je tvorba datových sad. Především takových, které zůstanou dlouhodobě použitelné. Vzhledem k proměnlivosti webových stránek je potřeba uložit všechna data potřebná ke správné reprezentaci webové stránky.

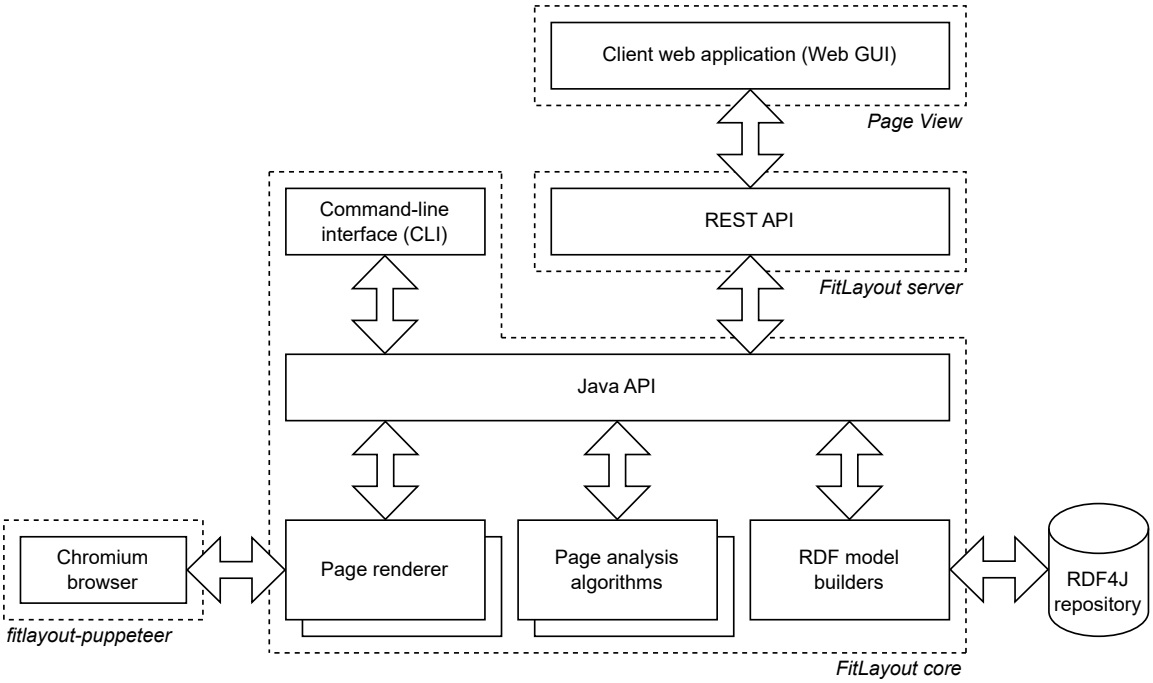
Knihovna FitLayout, implementovaná primárně v programovacím jazyce Java, poskytuje jednotnou sadu nástrojů pro zjednodušení a sjednocení různých přístupů k vývoji metod analýzy webových stránek. Nabízí již implementované a otestované způsoby vykreslování webových stránek pro vstupní data implementovaného algoritmu. Také obsahuje běžné datové struktury pro segmentaci webových stránek, se kterými může algoritmus přímo pracovat a které může využít jako výstup. A v neposlední řadě poskytuje způsob dlouhodobého ukládání webových stránek pro tvorbu datových sad a vizualizaci pro experimentování.

Knihovna je využívána v rámci implementace a cílem práce je zahrnutí implementované metody do sady již připravených algoritmů knihovny FitLayout. Je proto vhodné představení funkcionality knihovny a jejích částí. Kapitola je pak především zaměřena na ty, které implementovaný algoritmus využívá. Literárními prameny s vyčerpávajícím popisem knihovny, ze kterých kapitola čerpá, jsou [8], [20]. Celý projekt v aktuálním stavu lze i s dokumentací najít na portálu GitHub[1].

Kapitola je rozdělena do 3 sekcí – sekce 3.1 popisuje základní strukturu knihovny, navazující sekce 3.2 představuje možnosti vykreslení webových stránek pomocí knihovny a sekce 3.3 zahrnuje základní přehled artefaktů využitých v implementaci metody.

[1] FitLayout – https://github.com/FitLayout

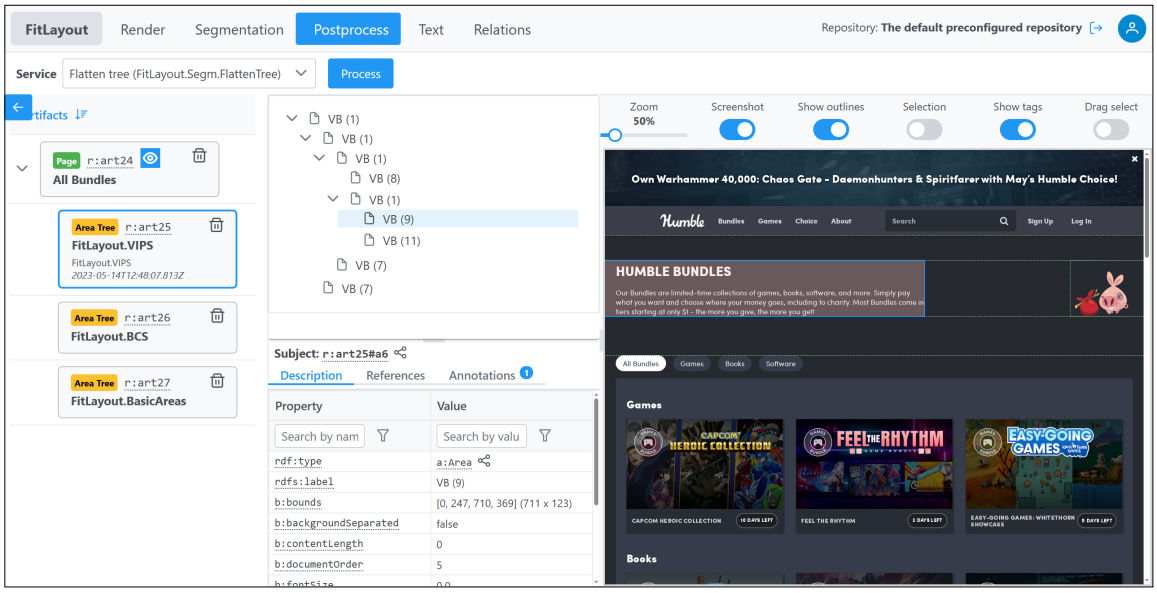
3.1 Architektura



Obrázek 3.1: Architektura a jednotlivé komponenty knihovny FitLayout. Převzato z [8].

Knihovna je složena z několika modulů o různých komponentách, které jsou navzájem propojené v rámci architektury na obrázku 3.1. Každý z modulů poskytuje určité funkcionality [8]:

- `FitLayout core` poskytuje základní sadu struktur a algoritmů pro podporu implementace a vyhodnocování metod analýzy webových stránek. Cílem práce je zahrnutí implementované metody do selekce algoritmů poskytovaných tímto modulem.
- Moduly `Page View` a `FitLayout server` zprostředkovávají uživatelské rozhraní pro potřeby pohodlnějšího experimentování a práce s databází uložených artefaktů. Jeho ukázka je v obrázku 3.2.
- `fitlayout-puppeteer` je pak separátní aplikace v Node.js prostředí zajišťující vykreslovací jádro pro rozhraní `PuppeteerTreeProvider`.



Obrázek 3.2: Ukázka grafického rozhraní nástroje FitLayout. V horní části se nachází nástroje, které lze použít na artefakt vybraný v levé části. Uprostřed je pak přehled informací o vybraném artefaktu (v tomto případě segmentaci stránky[3] segmentační metodou VIPS [9]), který je vizualizován napravo.

3.2 Vykreslování webové stránky

Pro vykreslení a získání kompletní reprezentace webové stránky FitLayout nabízí dvě různá rozhraní: [7]

- `CSSBoxTreeProvider` k vykreslení používá získané zdrojové soubory webové stránky (HTML a CSS). Přestože tento způsob může být rychlejší a bez přidané zátěže webového prohlížeče na pozadí, pro mnohé stránky je již v současnosti nedostačující, protože nezvládá vykreslení prvků zajištěných mimo zmíněné typy souborů – včetně souborů jazyka JavaScript, které dnes vyžaduje mnoho webových stránek.
- Tím druhým je `PuppeteerTreeProvider`, který k vykreslení využívá webový prohlížeč běžící bez uživatelského rozhraní Puppeteer[4], se kterým komunikuje pomocí Node.js API v rámci modulu `fitlayout-puppeteer`[5]. Díky tomu získaná reprezentace stránky obsahuje vše, co by obsahovala po načtení v běžném webovém prohlížeči.

Zaměřením práce je samotná implementovaná metoda, která na vstupu počítá s již kompletní vizuální reprezentací segmentované webové stránky. Pro vstupní data je tedy klíčovější spolehlivost jejich obsahu, která může ovlivnit vyhodnocování – na rozdíl od doby jejich získání, kterou lze při vyhodnocení jasně oddělit. V rámci implementace metody a jejího následného vyhodnocení je tedy použité vykreslování pomocí rozhraní `PuppeteerTreeProvider`.

[3] https://www.humblebundle.com/bundles
[4] Puppeteer – https://github.com/puppeteer/puppeteer
[5] `fitlayout-puppeteer` – https://github.com/FitLayout/fitlayout-puppeteer

3.3 Artefakty

FitLayout pracuje s artefakty – datovými strukturami reprezentujícími webovou stránku na nějaké úrovni abstrakce. Artefakty jsou dále děleny na obsahové elementy a celá struktura společně tvoří RDF[6] graf uložený v centrálním RDF repozitáři. K implementaci jsou relevantní zejména: [8]

- Na vykreslovací úrovni artefakt `Page` představuje webovou stránku jako strom boxů – obdélníkových ploch určených na základě [7] stromu podle CSS Visual Formatting Model[8]. Protože implementovaná metoda je čistě vizuální, stromová struktura pro implementaci není potřeba a stačí snímek obrazovky webové stránky získatelný prostřednictvím artefaktu. Pro implementaci je tedy klíčovou vlastností artefaktu spíše obstarání kompletního načtení webové stránky.
- Výsledkem zpracování stránky metodou segmentace je artefakt `AreaTree`, který je stromem ploch. Ty jsou opět obdélníkového tvaru a představují jednotlivé segmenty webové stránky tak, jak je určila daná segmentační metoda. Stejným způsobem se na webovou stránku dívá i implementovaná segmentační metoda a tak se artefakt nabízí jako vhodný výstupní formát.

Knihovna FitLayout nabízí i další artefakty, které představují další úhly pohledu a dělení obsahu webových stránek. Žádné z nich ale nejsou dále potřeba a tudíž jsou v rámci tohoto úvodu do knihovny vynechány.

Způsob ukládání artefaktů pomocí serializace kompletního RDF popisu umožňuje také tvorbu datových sad takovým způsobem, aby byly dlouhodobě použitelné a nezávislé na původních serverech daných webových stránek – a tudíž neměnné.

Implementace algoritmů v rámci knihovny FitLayout je pomocí rozhraní `ArtifactService`, které definuje vstupní a výstupní artefakt daného algoritmu. Vstupem segmentačních metod je typicky artefakt `Page` a výstupem pak artefakt `AreaTree`.

[6] RDF – Resource Description Framework (obecný rámec pro reprezentaci souvisejících dat na webu)
[7] DOM – Document Object Model (objektový model dokumentu)
[8] CSS 2.2 Specification – http://www.w3.org/TR/2016/WD-CSS22-20160412/

Kapitola 4

Implementovaná metoda

Metoda implementovaná v rámci této práce byla publikována v [14] a později její vylepšená verze v [13]. Obě verze jsou společně obsahem práce [12], která obsahuje širší záběr zpracování webových stránek použitím technologií z oblasti zpracování obrazu. Cílem této kapitoly je představení metody a jejích vlastností, včetně jejího fungování. K tomuto účelu jsou všechny informace a definice v ní obsažené čerpány právě z těchto publikací.

Pokud by měla být metoda zařazena do kategorií prezentovaných v 2.4, pak by se jednalo o metodu, jejíž postup je založen na *top-down* přístupu tvorby hierarchie segmentací, která byla navržena se zaměřením na segmentaci webových stránek, která nabízí několik nastavitelných parametrů, jejíž proces segmentace trvá delší dobu a která využívá výhradně vizuálních prvků stránky.

Poslední zmíněná vlastnost je tou nejzajímavější. Protože je vstupem metody pouze obrázek pořízený z vykreslené stránky, tak metoda výrazně čerpá z oblasti zpracování obrazu. Tento přístup je autory metody obhájen prezentováním několika výhod, jako je kompletní nezávislost na vnitřní implementaci stránky, možnost dělení na segmenty uvnitř objektů, které jsou v HTML chápány jako monolitické (typicky obrázky) a již zmíněné obecnější výhody vizuálního přístupu – algoritmus ke stránce přistupuje stejným způsobem jako člověk, který ji taktéž vnímá primárně vizuálně.

Tato metoda byla vybrána pro integraci do nástroje FitLayout kvůli tomu, že je relativně nová (publikace první verze je z roku 2016 [14], kompletní publikace nejnovější verze je pak z roku 2018 [12]), má oproti jiným metodám dobré výsledky kvality segmentace ([18]) a tím, že pracuje pouze s vizuálními prvky stránky, se liší od ostatních metod segmentace, které jsou aktuálně dostupné v nástroji FitLayout.

Jak je zmíněno výše, od publikace první verze implementované metody prošla metoda několika úpravami, které výrazně ovlivnily fungování všech jejích fází, jejichž popis je obsažen v následujících sekcích. Dle výsledků publikovaných v [13] úpravy celkově zlepšily výsledky metody a publikace srovnávající tuto metodu s jinými (jako např. [16], [18]) používají její upravenou verzi. Tudíž i v rámci této práce je implementována její poslední publikovaná verze.

Implementovaná metoda sestává ze tří klíčových fází:

1. Výpočet pravděpodobnosti lokálně významné hrany pro každý bod vstupního obrázku stránky

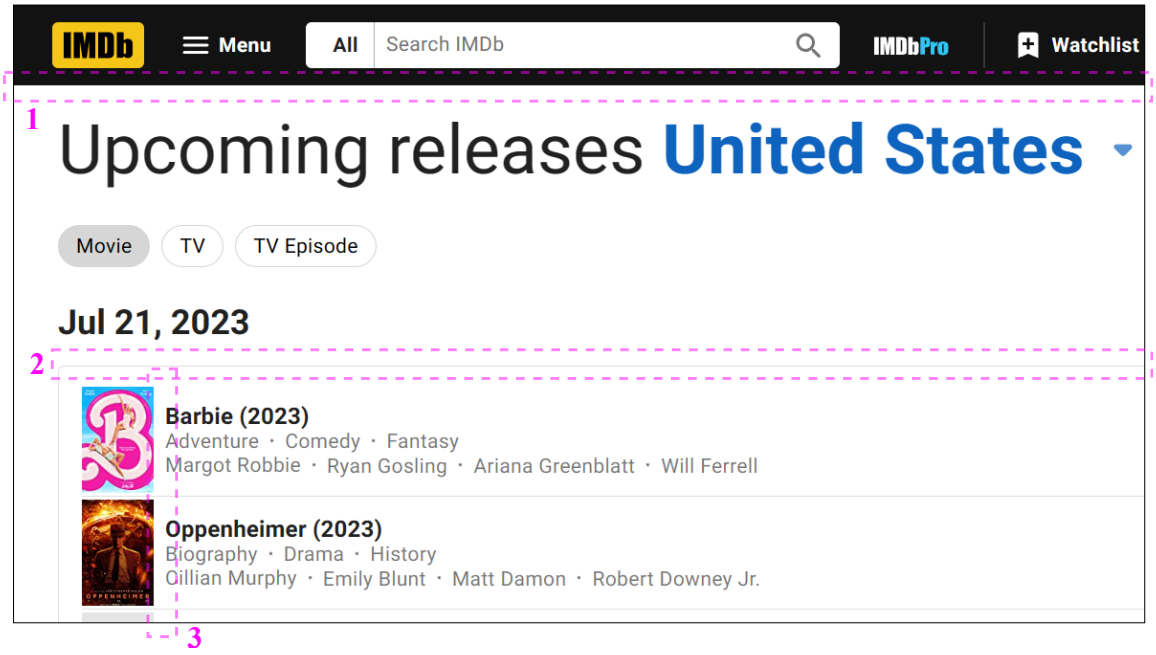
2. Výpočet pravděpodobnosti sémanticky významné hranice segmentace mezi dvěma body

3. Výpočet nejpravděpodobnější segmentace stránky na základě specifických předpokladů její struktury

Před vysvětlením jednotlivých fází sekce 4.1 popisuje formu výstupu a pravidla pro jednotlivé segmenty stránky definované metodou. Následující sekce se pak věnují jednotlivým fázím metody – 4.2 popisuje způsob získání pravděpodobnosti lokálně významných hran využitím jádrového odhadu hustoty pomocí Gaussova jádra, 4.3 obsahuje popis, jak výpočet pravděpodobnosti sémantických hranic využívá výsledků první fáze a nakonec 4.4 vysvětluje využití výsledků předchozích fází k tvorbě finální segmentace vstupní webové stránky.

4.1 Základní prvky

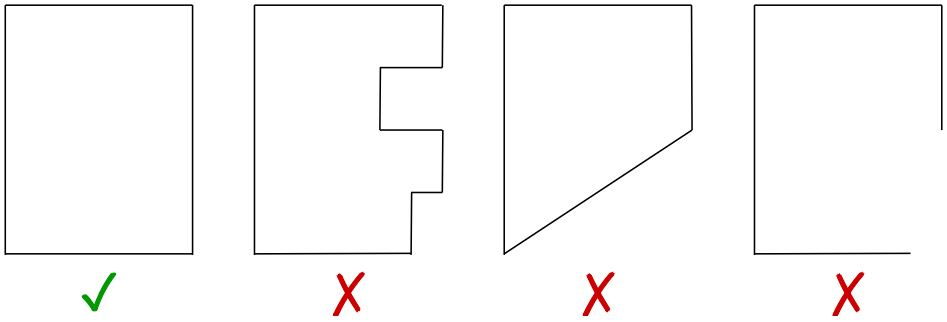
Klíčovým cílem metody je pro tvorbu jednotlivých segmentů detekovat oddělující hrany tvořené buď oddělovači (tedy kontrastní rovnou čárou), přechodem mezi dvěma barvami pozadí, zarovnáním objektů (např. položek elektronického obchodu), nebo hranicí objektu (typicky obrázku). Příklady takových hran lze pozorovat na obrázku 4.1.



Obrázek 4.1: Výřez obrázku stránky[2], na kterém lze pozorovat příklady druhů hran relevantních z pohledu metody, zvýrazněné purpurovým ohraničením. V pořadí dle očíslování: přechod mezi barvami pozadí; oddělovač; zarovnání objektů.

Metoda pak předpokládá několik vlastností těchto hran pro to, aby mohly být použity pro segmentaci. Všechny takové hrany musí být buď horizontální, nebo vertikální. Navíc musí tvořit okolo potenciálního segmentu uzavřenou a obdélníkovou hranici. Příklady validity ohraničení lze vidět na obrázku 4.2.

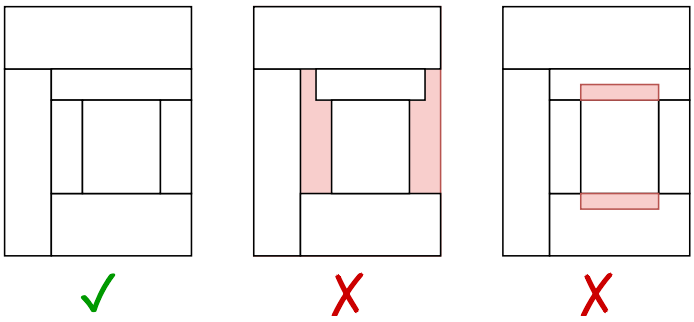
[2] https://www.imdb.com/calendar/



Obrázek 4.2: Hrany tvořící různá ohraničení a jejich validita z pohledu využití k segmentaci metodou. Zleva: validní, splňující všechny podmínky; nevalidní kvůli porušení obdélníkové hranice; nevalidní kvůli hraně, která není horizontálně nebo vertikálně zarovnaná; nevalidní kvůli neuzavřené hranici Obrázek převzat z [14].

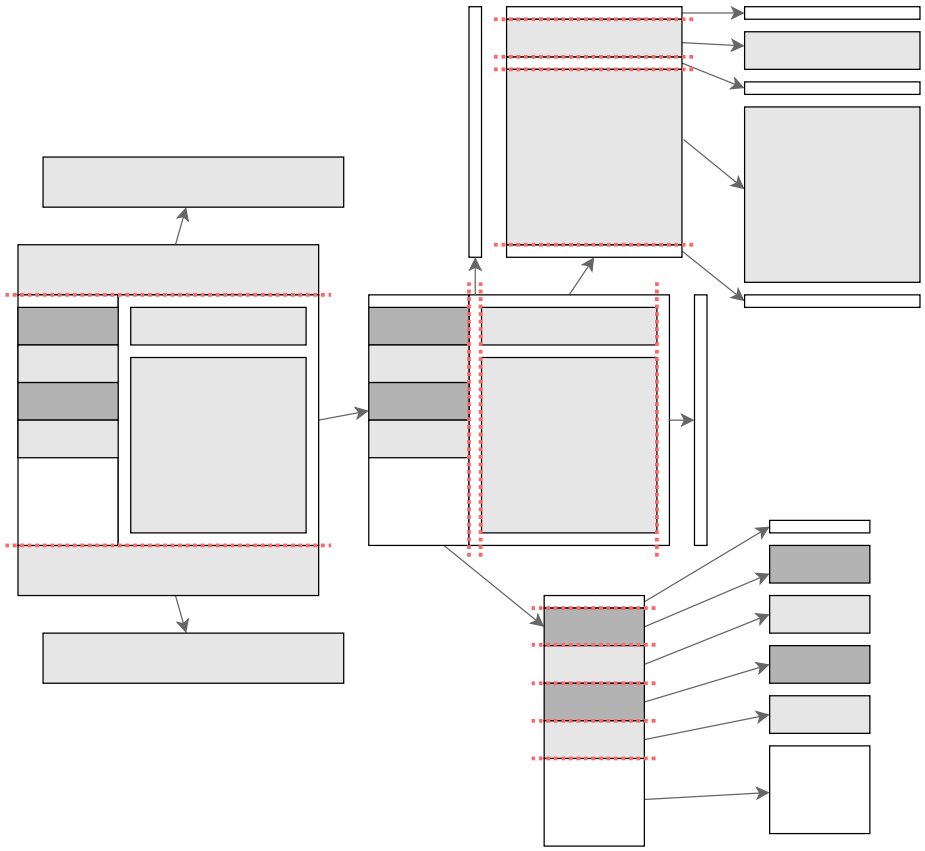
Dále metoda stanovuje pravidla pro vztahy mezi jednotlivými segmenty, které výrazně omezují množinu možných výsledných segmentací a jejichž dopad na validitu rozkladů na segmenty lze pozorovat na obrázku 4.3:

- Každý segment je zcela překrytý svým rodičovským segmentem.
- Každý segment je plně pokrytý svými potomky.
- Segmenty se nepřekrývají, pokud jeden není rodičem druhého.
- Kořenový segment představuje celou stránku.



Obrázek 4.3: Segmenty tvořící různé rozklady plochy a jejich validita z pohledu stanovených pravidel. Zleva: validní, splňující všechny podmínky; nevalidní, protože segmenty nepokrývají celou plochu rodičovského segmentu; nevalidní kvůli překrytí s nerodičovskými regiony. Obrázek převzat z [14].

Výslednou výstupní strukturou algoritmu je pak hierarchická stromová struktura, která je sestavena z jednotlivých segmentů vytvořených metodou, která splňuje vlastnosti typické pro metody segmentace webových stránek popsané v rámci sekce 2.3 a jejíž příklad lze vidět na obrázku 4.4.



Obrázek 4.4: Příklad výsledné stromové struktury tvořené jednotlivými segmenty. Obrázek převzat z [14] a aktualizován pro novější verzi metody, jejíž výstup musí navíc oproti starší verzi dodržovat vlastnosti X-Y stromu.

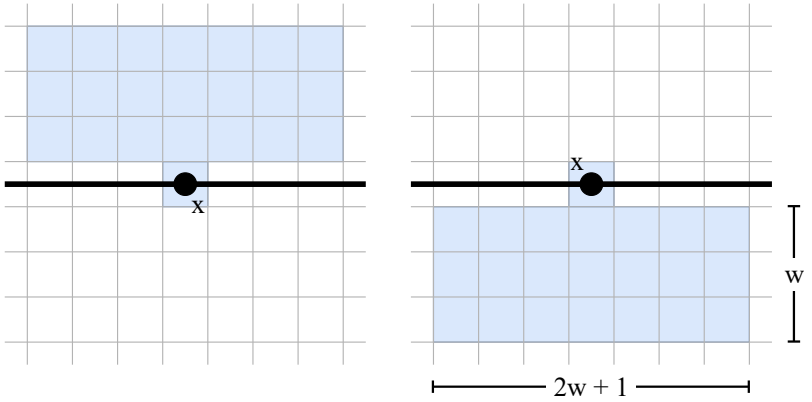
4.2 Lokálně významné hrany

Cílem první fáze algoritmu je detekce hran, na základě jejichž přítomnosti může další fáze určovat pravděpodobnost existence rozdělovací hranice. Aplikace tradiční detekce hran z oblasti zpracování obrazu je ovšem sama o sobě nedostačující. Hlavním důvodem je text, který je obsažen ve většině stránkách a pro dobro čtenáře je stylován takovým způsobem, aby byl kontrastní oproti svému pozadí. To způsobuje, že detekce hran odhaluje na hranicích písmen velmi výrazné hrany. A naopak jemnější přechody mezi barvami pozadí pak detekcí hran neprodukují tak významné hrany, přestože právě takové dělení je pro nalezení segmentace stránky často velmi důležitou indícií.

Algoritmus řeší tento problém srovnáním síly hrany získané tradiční metodou detekce hran v daném bodě se silou hran bodů v jeho okolí, získanou identickým způsobem.

K tomuto řešení definuje metoda pro každý bod čtyři různá okolí, každé na jedné straně potenciální hrany (tj. dvě pro horizontální a dvě pro vertikální). Rozdělení okolí na dvě poloviny je z důvodu lepšího pokrytí případu, kdy jedna polovina má významnější hrany než druhá, tj. že hrana vyniká vůči polovině s méně významnými hranami (např. pokud odděluje plochu s nějakým vzorem od prázdné plochy). Takovou hranu oddělující dvě odlišně významné plochy lze považovat za významnou, proto metoda dává prioritu vyššímu z výsledků z obou polovin okolí. Každé z těchto okolí obsahuje i daný bod, avšak neobsa-

huje body podél potenciální hrany – jejich zahrnutí by způsobilo snížení významnosti bodu v případě, kdy by podél hrany byly body významné, což nedává smysl. Velikost všech okolí je pak určena nastavitelným parametrem. Znázornění definice okolí bodu lze pozorovat na obrázku 4.5. Takto definované okolí bodu (x, y) bude dále odkazováno jako $N(x, y)$.



Obrázek 4.5: Okolí brané v potaz při výpočtu významnosti horizontální hrany v bodu x. Tlustá čára znázorňuje potenciální horizontální hranu. Modrou barvou jsou znázorněny body daného okolí. Parametr w určuje vzdálenost v bodech, do které je okolí bráno v potaz od posuzovaného bodu. Pro výpočet lokální významnosti vertikální hrany je struktura otočená o 90°. Obrázek převzat z [14].

Pro to, aby mohla být srovnána významnost hrany v daném bodě s jejím okolím, musí být pro každý bod tato významnost hrany nejdříve určena. Metoda pro tento účel používá Sobelova operátoru. [22] Výsledná hodnota získaná tímto způsobem bude dále referována jako $S(x, y)$, tedy síla hrany v bodu (x, y).

Vylepšená verze algoritmu navíc navrhuje aplikaci Sobelova operátoru pro různě škálované verze vstupního obrázku získané aplikací Gaussovy pyramidy. Výsledná síla hrany, se kterou metoda dále pracuje, je pak druhou odmocninou sumy jejich druhých mocnin.

K popisu rozložení této hodnoty v okolí $N(x, y)$ je použitý neparametrický odhad rozložení síly hran. Funkce hustoty rozložení je následně definována použitím jádrového odhadu hustoty s Gaussovým jádrem na dané distribuci:

$$P(s) = \frac{1}{|N|} \sum_{(x,y)\in N} \mathrm{Norm}(s; S(x, y), \sigma) \tag{4.1}$$

kde $\mathrm{Norm}(s; S(x, y), \sigma)$ představuje hustotu v bodu s s normálním rozdělením se střední hodnotou $S(x, y)$ a směrodatnou odchylkou σ, která je vstupním parametrem metody.

Jestliže $S_{x,y,s}$ určuje jev výskytu bodu v okolí $N(x, y)$ s lokální významností hrany vyšší nebo stejnou jako v bodu (x, y), pak je pravděpodobnost tohoto jevu definována následovně:

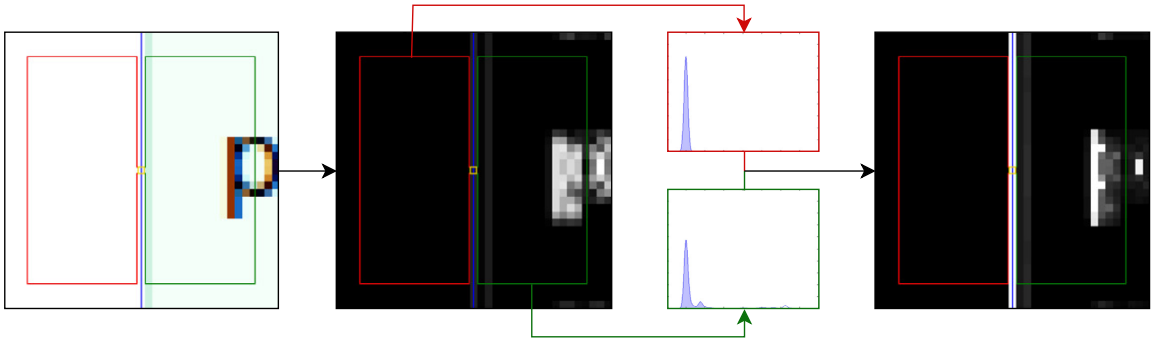
$$\Pr(S_{x,y,s}|\overline{E_{x,y}}, P) = 1 - \int_0^s P(t)\, dt \tag{4.2}$$

Lokální významnost hrany pro bod (x, y) je výsledně definována jako pravděpodobnost, že tímto bodem prochází lokálně významná hrana za předpokladu, že platí již výše definovaný jev $S_{x,y,s}$ v rámci distribuce sil hran v daném okolí $N(x, y)$ P: $Pr(E_{x,y}|S_{x,y,s}, P)$. S danou apriori pravděpodobností výskytu hrany v bodu (x, y) $\Pr(E_{x,y})$ (která je dalším

parametrem metody), pravděpodobností definovanou v 4.2 a pomocí Bayesova teorému ji lze definovat následovně:

$$\Pr(E_{x,y}|S_{x,y,s},P) = \frac{\Pr(E_{x,y})\,(1+\Pr(S_{x,y,s}|\overline{E_{x,y}},P) - \Pr(E_{x,y})\Pr(S_{x,y,s}|\overline{E_{x,y}},P))}{\Pr(E_{x,y}) + \Pr(S_{x,y,s}|\overline{E_{x,y}},P) - \Pr(E_{x,y})\Pr(S_{x,y}|\overline{E_{x,y}},P)} \tag{4.3}$$

Podrobný postup získání vzorce 4.3 lze nalézt v publikaci [12], k základnímu pochopení principu a implementaci této fáze metody ovšem není nutný.



Obrázek 4.6: Ilustrace procesu získání lokální významnosti vertikální hrany. Levé (červeně ohraničené) a pravé (zeleně ohraničené) okolí patří ke žlutě ohraničenému bodu uprostřed. Zleva první obrázek je originální výřez obrázku webové stránky. Na druhém obrázku je síla hran získaná Sobelovým operátorem. Další dva obrázky vyjadřují distribuci síly hran pro jednotlivá okolí. Obrázek vpravo znázorňuje výsledné lokální významnosti, kde tmavší body představují významnost nižší, než ve žlutě ohraničeném bodu. Obrázek převzat z [14].

4.3 Sémanticky významné hranice

K formování segmentací lokálně významné hrany nestačí. Jde v podstatě pouze o distribuci významnosti hran rozprostřené po celé ploše obrázku stránky. Úkolem následující fáze popsané v této sekci je proto využití této informace k určení potenciálních hranic, ze kterých už poslední fáze bude moci sestavovat výsledné segmentace.

Pro formování potenciálních hranic metoda definuje rekurzivní algoritmus 1, jehož hlavním vstupním parametrem je potenciální hranice. Jestliže je hranice dostatečně velká (tj. delší než vstupní parametr t_l), definuje pravděpodobnost sémantické významnosti této hranice jako pravděpodobnost, že obě její poloviny jsou sémanticky významné. V opačném případě (tj. hranice je kratší než vstupní parametr t_l) je definována pravděpodobností jevu, že podél hranice je dostatečná lokální významnost.

Takový jev metoda definuje s využitím Poissonova rozložení, které vyjadřuje výskyty dostatečné (tj. alespoň rovné vstupnímu parametru t_p) lokální významnosti hrany na dané hranici, v rámci intervalu daným velikostí posuzované hranice, jako na sobě nezávislé. Pro

určení výsledné pravděpodobnosti, která představuje výslednou sémantickou významnost hranice, implementovaná metoda navrhuje použití metody Monte Carlo.

Algoritmus 1: LinePr – Výpočet pravděpodobnosti, že daná hranice L je sémanticky významná. Převzato z [12].

Vstup: $L,\ t_l,\ t_p$
Výstup: pravděpodobnost P, že L je sémanticky významná hranice

```
1 l = délka(L)
2 if l > t_l then
      Pravděpodobnost, že obě poloviny jsou sémanticky významné.
3     P = LinePr(L_{1 ... ⌊l/2⌋}, t_l, t_p) × LinePr(L_{⌊l/2⌋+1 ... l}, t_l, t_p)
4 else
5     n = počet bodů v L, které mají lokálně významné hrany
6     P = Pr(⌈n/l⌉ ≥ t_p)
7 end if
8 return P
```

Motivací práce s lokální významností podél hranice jako s rozložením je ve snazším odhalování přerušovaných hran, které jsou typicky formovány zarovnanými elementy na stránce. Tato výhoda je podložena výsledky zkoumání rozdílů sémantické významnosti (získané tímto přístupem) kratších nepřerušovaných hran oproti delším přerušovaným v publikaci [13], pro pochopení principu a implementaci ovšem není potřeba.

4.4 Segmentace

Poslední fáze používá sémanticky významné hranice, respektive jejich pravděpodobnost, k určení výsledných hranic dělící stránku na jednotlivé segmenty. Díky omezení prostoru možných segmentací definováním pravidel pro segmenty v sekci 4.1 je tato fáze výrazně zjednodušena. Prohledávání tohoto prostoru je dále omezeno tím, že rozklady segmentů na menší tvoří tzv. X-Y strom, tj. každé horizontální dělení je následováno vertikálním a naopak každé vertikální je následováno horizontálním.

K tomuto přístupu segmentace metoda nejprve představuje algoritmus 2, který postupně rekurzivním postupem dělí segmenty n (definované čtveřicí hodnot (n_t, n_b, n_l, n_r) ohraničující segment) na menší, počínaje kořenovým segmentem představujícím celou stránku. Princip algoritmu spočívá v nalezení nejpravděpodobnější hranice k rozdělení aktuálního segmentu na základě pravděpodobností získaných pomocí algoritmu 1, kterému předává jako vstupní parametry lokální významnost možných dělících hranic H, resp. V, pro horizontální, resp. vertikální hranice. Zároveň musí oba výsledné podsegmenty splňovat minimální délku v obou dimenzích určenou vstupním parametrem s_{min}. Pokud pravděpodobnost ta-

kové hranice přesahuje 50%, je segment touto hranicí rozdělen na dva menší. Jinak segment dále dělen není a v segmentačním stromu je jako listový uzel.

Algoritmus 2: SplitReg – Algoritmus pro dělení segmentu na dva podsegmenty. Převzato z [12].

Vstup: $n = (n_t, n_b, n_l, n_r),\ H,\ V,\ s_{min}$
Výstup: podsegmenty N; směr rozdělení D

1 **for** $i \in \langle n_t + s_{min},\ n_b - s_{min} \rangle$ **do**
  Určení pravděpodobnosti horizontální hranice na řádku i.
2 $P_{H,i} \leftarrow \text{LinePr}(H(i, n_l \dots n_r), t_p, t_l)$
3 **end for**
4 **for** $i \in \langle n_l + s_{min},\ n_r - s_{min} \rangle$ **do**
  Určení pravděpodobnosti vertikální hranice ve sloupci i.
5 $P_{V,i} \leftarrow \text{LinePr}(V(n_t \dots n_b, i), t_p, t_l)$
6 **end for**
7 **if** $\max(P_H) \geq \max(P_V) \wedge \max(P_H) \geq 0.5$ **then**
  Nejlepší rozdělení je horizontální.
8 $i = \text{argmax}(P_H)$
9 $N = \{(n_t, i, n_l, n_r), (i, n_b, n_l, n_r)\},\ D = \text{horizontální}$
10 **else if** $\max(P_V) \geq 0.5$ **then**
  Nejlepší rozdělení je vertikální.
11 $i = \text{argmax}(P_V)$
12 $N = \{(n_t, n_b, n_l, i), (n_t, n_b, i, n_r)\},\ D = \text{vertikální}$
13 **else**
  Segment nesplňuje podmínky pro rozdělení.
14 $N = \emptyset,\ D = \text{list}$
15 **end if**
16 **return** N, D

Výsledný segmentační algoritmus 3 využívající algoritmu 2 je založený na hladovém dělení (tj. vždy volí nejlepší z právě možných rozdělení) v lokálně optimálních bodech s cílem vytvoření množiny podsegmentů. Podsegmenty v takto vytvořených množinách jsou dále děleny stejným principem až dokud jsou všechny dále nedělitelné. Vlastnosti X-Y stromu jsou zachovány seskupováním segmentů po sobě dělených hranicemi stejného směru.

Výsledkem algoritmu je X-Y strom segmentů (N, E) dané stránky, kde N jsou jeho uzly a E hrany.

Algoritmus 3: Segment – Algoritmus pro rozklad stránky na X-Y strom segmentů. Převzato z [12].

Vstup: $n,\ H,\ V,\ s_{min}$
Výstup: strom segmentů (N, E)

$N = \{n\},\ E = \emptyset,\ N_{child} = \emptyset$

Pokus o prvotní rozdělení segmentu.

$(N_{temp},\ D) = \text{SplitReg}(n,\ H,\ V,\ s_{min})$

for $n_t \in N_{temp}$ **do**

  Pokusy o další rozdělování segmentů na této úrovni stromu.

  $(N',\ D') = \text{SplitReg}(n_t,\ H,\ V,\ s_{min})$

  if $D = D'$ **then**

    Stejný směr rozdělení ⇒ příprava pro další dělení.

    $N_{temp} = (N_{temp} \setminus \{n_t\}) \cup N'$

  else

    Různý směr nebo žádné rozdělení ⇒ přidání k podsegmentům.

    $N_{child} = N_{child} \cup \{n_t\}$

  end if

end for

for $n_c \in N_{child}$ **do**

  Rekurzivní segmentace podsegmentů pro další úroveň stromu.

  $(N'',\ E'') = \text{Segment}(n_c,\ H,\ V,\ s_{min})$

  $N = N \cup N''$

  $E = E \cup E'' \cup \{(n,\ n_c)\}$

end for

return $(N,\ E)$

Kapitola 5

Návrh integrace

Cílem této práce je integrace segmentační metody popsané kapitolou 4 do nástroje FitLayout. Tato kapitola má pak za cíl před samotnou implementací ověřit, jestli je metoda kompatibilní s nástrojem a toto tvrzení následně podložit návrhem její implementace s využitím nástroje FitLayout. Ověření kompatibility je obsaženo v sekci 5.1, samotný návrh je pak detailněji rozebrán sekcí 5.2.

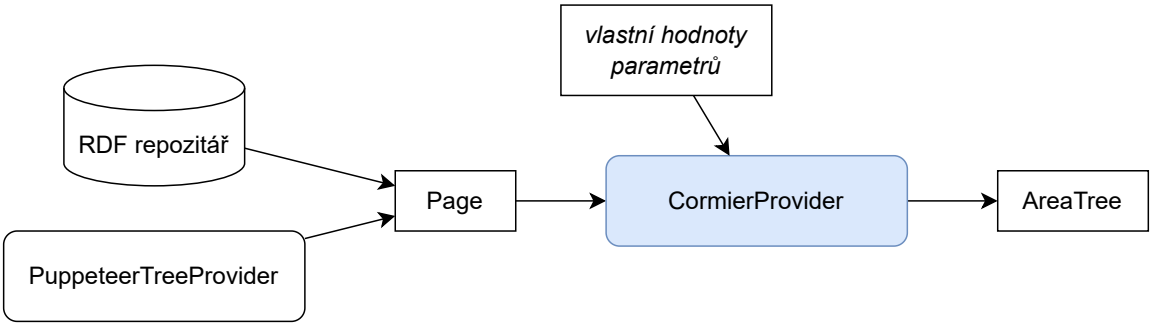
5.1 Kompatibilita s nástrojem FitLayout

Jako vstupní data metoda vyžaduje obrázek kompletní vykreslenou stránku. K získání tohoto obrázku lze využít některé z nabízených rozhraní blíže popsané v sekci 3.2. Protože metoda žádné další vstupní informace o stránce nepotřebuje, žádné další informace o stránce získat není potřeba. Jedinými dalšími volitelnými vstupy mohou být vlastní nastavení vstupních parametrů, které by jinak byly nastaveny na výchozí hodnoty na základě vyhodnocení.

Výstupem metody je X-Y strom výsledných segmentů. K reprezentaci této struktury lze využít artefakty `AreaTree` a `Area`. Žádným způsobem sice nevyžadují specifické vlastnosti X-Y stromu, ale protože je X-Y strom specifickým případem stromové struktury, tak lze tyto artefakty využít pro jeho reprezentaci. Dodržení vlastností X-Y stromu pak musí být zaručeno v rámci implementace metody. Ostatní vlastnosti výstupu implementované metody tyto artefakty respektují.

Protože je nástroj schopný poskytnout metodě potřebná data a protože poskytuje vhodné datové struktury, do kterých lze uložit výstupy této metody, lze říci, že nástroj FitLayout je kompatibilní s metodou a její integrace by měla být možná. Tato kompatibilita je znázorněna schématem 5.1.

Způsob integrace metody do nástroje FitLayout je implementací rozhraní `ArtifactService`, sloužící pro tvorbu služeb tvořících artefakty z jiných – tj. v případě implementované metody tvořící artefakt `AreaTree` z artefaktu `Page`, vykresleného některým z vykreslovacích jader popsaných v sekci 3.2, nebo případně načtené z již dříve získané stránky z RDF repozitáře. Takto implementované rozhraní bude dostupné v rámci vlastního modulu, který bude poskytován nástrojem FitLayout, obdobně jako u jiných segmentačních metod, které nástroj již nabízí.



Obrázek 5.1: Třída `CormierProvider` implementující rozhraní `ArtifactService` požaduje na vstup artefakt `Page` vykreslený některým z vykreslovacích jader – na obrázku např. `PuppeteerTreeProvider` –, nebo případně získaným z RDF repozitáře. Volitelným vstupem jsou vlastní hodnoty nastavitelných parametrů segmentační metody. Pro vykonání segmentace je výstupem artefakt `AreaTree`, reprezentující výsledný X-Y strom segmentů.

5.2 Návrh implementace

Samotná třída implementující rozhraní `ArtifactService` má za úkol zpracovat vstupní parametry pro segmentační metodu, tedy vstupní artefakt stránky `Page` a případné další nastavitelné parametry. Volání s těmito parametry pak deleguje na další třídy se samotnou implementací logiky implementované metody, jak lze vidět v diagramu 5.2. Jde o třídy implementující jednotlivé fáze průběhu metody. Každá z nich obsahuje relevantní parametry k dané fázi, které jsou nastaveny během vytváření instancí těchto tříd. Výsledkem delegovaného volání jsou segmenty v podobě artefaktu `AreaTree`.

Transformaci obrázku na lokálně významné hrany zajišťuje třída `EdgeDetector`. Ta nabízí rozhraní, které na vstupu očekává obrázek (kterým bude snímek obrazovky segmentované stránky), který podle popisu v rámci sekce 4.2 transformuje na 2 matice o stejné velikosti s lokální významností horizontálních a vertikálních hran. Výsledky tohoto procesu lze ovlivnit čtyřmi parametry:

- `halfWindowWidth` – Vzdálenost od posuzovaného bodu k okraji jeho zpracovávaného okolí (tj. polovina šířky zpracovávaného okolí).
- `standardDeviation` – Směrodatná odchylka pro odhad hustoty podle vzorce 4.1.
- `priorEdgeProbability` – Apriori pravděpodobnost výskytu hrany v posuzovaném bodu použitá ve vzorci 4.3
- `pyramidLevels` – Počet úrovní Gaussovy pyramidy pro výpočet síly hran.

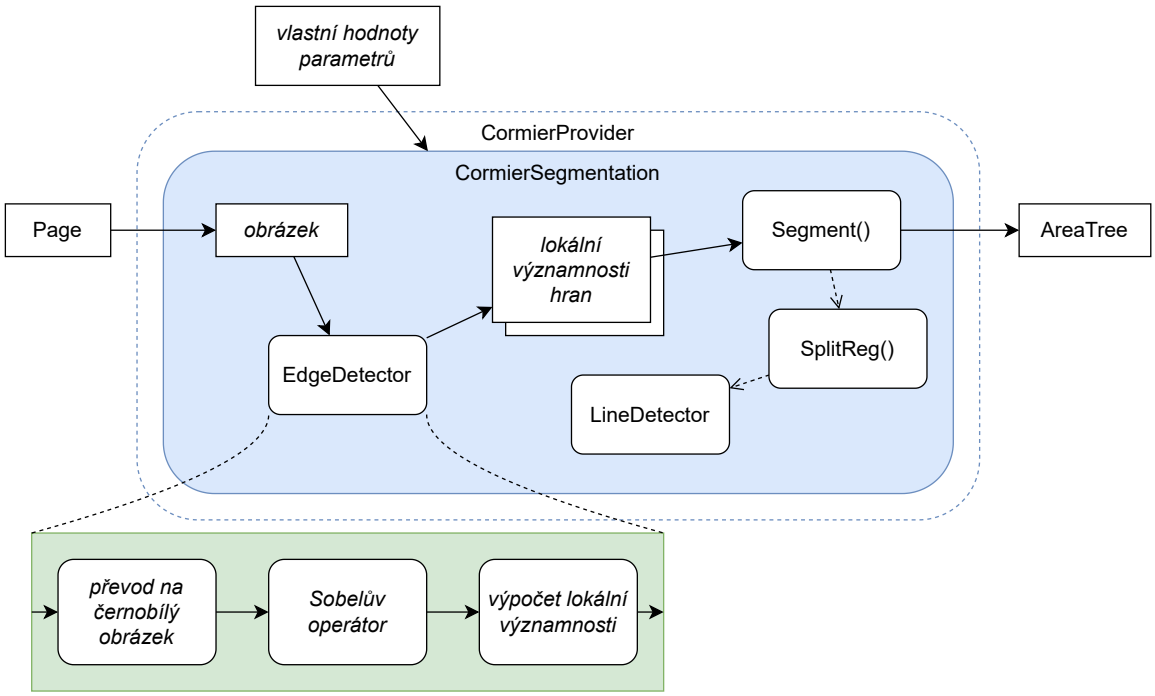
Funkci posuzování hranic zda jsou sémanticky významné poskytuje třída `LineDetector`. Její rozhraní představuje implementaci algoritmu 1. Vstupem je tedy hranice bodů s jejich lokální významností a výstupem pravděpodobnost, že jde o sémanticky významnou hranici. Způsob výpočtu pravděpodobnosti lze ovlivnit následujícími parametry:

- `maxLineLength` – Maximální délka vstupní hranice, než algoritmus hranici rozdělí na poloviny a posoudí pravděpodobnosti každé z nich zvlášť rekurzivním voláním.

- `edgeProbabilityThreshold` – Minimální pravděpodobnost lokální významnosti posuzovaného bodu v hranici k tomu, aby byl považován za významný.
- `monteCarloTrials` – Počet pokusů metody Monte Carlo pro aproximaci výsledné pravděpodobnosti.

Implementaci algoritmů 2 a 3, které vyžadují výstupy předchozích dvou tříd, obsahuje třída `CormierSegmentation`. Tato třída zároveň funguje jako vstupní bod pro spuštění metody – její rozhraní umožňuje delegaci volání z implementace rozhraní `ArtifactService`. A stejně jako implementovaná metoda očekává na vstupu obrázek a jejím výstupem je X-Y strom segmentů. I fungování logiky této třídy lze ovlivnit následujícími parametry:

- `minSegmentLength` – Nejmenší možná délka strany segmentu.
- `signLineProbThreshold` – Nejmenší možná pravděpodobnost, že je hranice sémanticky významná, pro to, aby mohla být použita k tvorbě segmentů.



Obrázek 5.2: Z artefaktu `Page` je získán snímek obrazovky, který je vstupem segmentační metody. Obrázek je předán třídě `EdgeDetector`, kde je převeden na černobílý, a následně je na něj aplikován Sobelův operátor. Ze získaných hran je vypočítána jejich lokální významnost, která je vstupem pro operaci `Segment()`. Ta volá operaci `SplitReg()` takovým způsobem, aby výsledné segmenty dodržovaly strukturu X-Y stromu. Operace `SplitReg()` pak využívá třídu `LineDetector` k určení, jestli lze segmenty dělit a případně ve kterém místě.

Kapitola 6

Implementační řešení

Cílem implementace je dodržení popisu segmentační metody z kapitoly 4 při její integraci s nástrojem FitLayout. Tato kapitola představuje detaily této implementace v sekcích rozdělených podobným způsobem, jako sekce kapitoly s popisem segmentační metody. Sekce 6.1 popisuje, jakým způsobem je implementován výpočet lokální významnosti hran, jehož popis je v sekci 4.2. Navazující sekce 6.2 a 6.3 pak obdobným způsobem popisují implementační řešení fází metody popsaných v rámci sekcí 4.3 a 4.4.

6.1 Lokálně významné hrany

Implementaci první fáze metody lze rozdělit na 2 části – první část předzpracovává obrázek stránky pro druhou část, jejíž výsledkem jsou výsledné pravděpodobnosti lokální významnosti jednotlivých bodů obrázku. Výpočty v první části jsou z větší části implementovány knihovními funkcemi, které nejsou oproti jiným částem metody tak výpočetně náročné. Druhá část obsahuje náročnější výpočty a tudíž jsou na ni v rámci implementace aplikované optimalizace, které se na tento problém zaměřují.

6.1.1 Předzpracování obrázku stránky

Pro předzpracování obrázku k získání významnosti hran před jejím použití pro výpočet lokální významnosti je využita knihovna OpenCV[1]. Konkrétně jde o převod obrázku na černobílý a následnou aplikaci Sobelova operátoru. [22]

Součástí předzpracování je také aplikace Sobelova operátoru v rámci Gaussovy pyramidy, tj. jeho aplikace na různé velikosti (dále „vrstvy“) vstupního obrázku. Počet těchto vrstev je dán hodnotou parametru `pyramidLevels`, kde každá další vrstva je oproti předchozí zmenšena na poloviční rozměry. Výsledkem předzpracování je pak odmocnina ze sumy druhých mocnin hodnot jednotlivých vrstev, tak jak je zmíněno sekcí 4.2. Jestliže je parametr `pyramidLevels` nastaven na hodnotu 1, pak tento krok nemá žádný efekt a je přeskočen.

6.1.2 Výpočet lokální významnosti

Nad každým bodem předzpracovaného obrázku následuje výpočet pravděpodobnosti, že daný bod obsahuje lokálně významnou hranu. Tento výpočet je implementován podle vzorců ze sekce 4.2, včetně nastavitelných parametrů `standardDeviation` pro určení směrodatné

[1] OpenCV – https://opencv.org/

odchylky σ a `priorProbability` pro určení apriori pravděpodobnosti výskytu hrany v bodu (x, y) $\Pr(E_{x,y})$.

Pro výpočet této pravděpodobnosti je potřeba pro každý bod obrázku pracovat s jeho okolím, jehož velikost je určena parametrem `halfWindowWidth` – pravděpodobnost výskytu bodu v okolí s lokálně významnější hranou, potřebná pro výsledný výpočet pravděpodobnosti lokální významnosti, je sumou hodnot distribučních funkcí normálního rozdělení (definovaného vzorcem 4.1) v tomto okolí. To znamená, že pokud označíme šířku a výšku vstupního obrázku w a h, algoritmus pro výpočet pravděpodobnosti lokální významnosti nad předzpracovaným obrázkem má složitost $\mathcal{O}(h \cdot w \cdot \texttt{halfWindowWidth}^2)$. Lze tedy říci, že velikost obrázku a parametr `halfWindowWidth` mají zásadní vliv na výslednou dobu trvání výpočtu.

Tento výpočet je navíc proveden zvlášť pro horizontální a zvlášť pro vertikální hrany. Parametr `halfWindowWidth` také určuje vzdálenost hranice okolí od posuzovaného bodu – reálná velikost okolí je tedy $(2 \cdot \texttt{halfWindowWidth} + 1) \cdot (2 \cdot \texttt{halfWindowWidth} + 1)$. Je ovšem nutno brát v potaz, že okolí krajních bodů obrázku má menší velikost (až 25% v extrémním případě jednoho ze čtyř bodů v rozích obrázku) a body v linii protínající posuzovaný bod (souběžně se směrem posuzované hrany) nejsou součástí výpočtu.

6.1.3 Zrychlení výpočtu

K výpočtu distribuční funkce normálního rozdělení je využité rozhraní knihovny Apache Commons Math™[2] `NormalDistribution`. Skrze toto rozhraní je pro výpočet pravděpodobnosti výskytu bodu s lokálně významnější hranou v okolí daného bodu získána hodnota distribuční funkce normálního rozdělení pro každý bod z jeho okolí. Inicializace třídy `NormalDistribution` je ovšem poměrně nákladná operace (i přes explicitní vynechání inicializace generátoru náhodných čísel, který je součástí nabízeného rozhraní a není pro výpočet potřeba) a je prováděna s již zmíněnou složitostí $\mathcal{O}(h \cdot w \cdot \texttt{halfWindowWidth}^2)$.

Podle vzorce 4.1 jsou ovšem pro inicializaci rozdělení potřeba pouze parametry síly hrany z předzpracovaného obrázku a vstupní parametr `standardDeviation`, který je konstantní. To znamená, že rozdělení není závislé na posuzovaném bodu a tedy třídu `NormalDistribution` lze inicializovat pouze jednou pro každý bod obrázku. Nad takto sdílenými instancemi lze pak pro výpočet zjišťovat hodnoty distribučních funkcí závislých na posuzovaném bodu pomocí metody `cumulativeProbability(double)`.

Počet potřebných inicializací třídy `NormalDistribution` se tak sníží na $\mathcal{O}(h \cdot w)$. To vede k odstranění závislosti na velikosti okolí a tím pádem i ke zrychlení, které nabývá na efektivitě s rostoucí velikostí okolí (a naopak pro extrémní případ $\texttt{halfWindowWidth} = 1$ nepřináší žádné zrychlení, takové nastavení ovšem nedává smysl). Tato optimalizace má ovšem za následek obdobnou paměťovou náročnost $\mathcal{O}(h \cdot w)$, protože je potřeba uchovávat instanci třídy `NormalDistribution` pro každý bod obrázku.

Další využitelnou vlastností procesu výpočtu pravděpodobnosti lokální významnosti hran je nezávislost procesu získání této hodnoty pro každý z bodů obrázku. Tj. pro výpočet této hodnoty pro daný bod není potřeba znát výsledky pro kterýkoliv jiný z bodů. To znamená, že iterace pro každý bod obrázku (tedy iterace smyčky s časovou složitostí $\mathcal{O}(h \cdot w)$) mohou běžet paralelně, kdy každá bude mít časovou složitost $\mathcal{O}(\texttt{halfWindowWidth}^2)$.

[2] Apache Commons Math™– https://commons.apache.org/proper/commons-math/

6.2 Sémanticky významné hranice

Třída `LineDetector` pro posuzování hranic, zda jsou sémanticky významné, implementuje algoritmus 1. U jeho implementace jde především o přístup k výpočtu pravděpodobnosti výskytů bodů s dostatečnou lokální významnosti podél posuzované hranice. Publikace implementované segmentační metody k tomuto účelu navrhuje použití metody Monte Carlo. [12]

Obecně je metoda Monte Carlo skupina algoritmů, jejichž cílem je stochastickým způsobem aproximovat žádoucí výsledek. Tuto aproximaci provádí metoda v předem omezeném počtu pokusů, kde výsledná aproximace následně vzniká zpracováním jejich jednotlivých výsledků. Obecně rovněž platí, že přesnost výsledku roste spolu s počtem těchto pokusů, ovšem doba potřebná pro výpočet roste také. [25]

U implementace metody Monte Carlo k účelu určení sémantické významnosti hranic je v rámci jednoho pokusu nejdříve stanoven počet významných bodů podél dané hranice. Tyto body jsou vybrány z bodů hranice s pravděpodobností jejich lokální významnosti. Výsledkem pokusu je posouzení, zda se na hranici vyskytuje dostatečný poměr těchto bodů, kde tento poměr je definován parametrem `edgeProbabilityThreshold`. Pokud ano, je pokus započítán jako úspěšný. Výsledná pravděpodobnost, že jde o sémanticky významnou hranici, je agregována z výsledků jednotlivých pokusů jako poměr úspěšných pokusů vůči celkovému počtu pokusů, který je definován parametrem `monteCarloTrials`.

Díky vlastnostem metody Monte Carlo je pak tato část segmentační metody (jediným) zdrojem nedeterminističnosti výsledků. Jak již bylo naznačeno výše, tento potenciálně nežádoucí efekt lze minimalizovat zvýšením počtu pokusů, tedy navýšením hodnoty parametru `monteCarloTrials`. Vyšší hodnoty ovšem způsobí zpomalení algoritmu – záleží tedy na prioritě požadavků dané konkrétní aplikací segmentační metody.

6.3 Segmentace

Výsledná fáze, implementovaná v rámci třídy `CormierSegmentation`, využívá tříd popsaných v sekcích 6.1 a 6.2 k implementaci algoritmů pro výslednou tvorbu segmentů.

Algoritmus pro dělení segmentu na podsegmenty je implementován v podstatě přesně podle předpisu 2. Jedinou zvláštností je přidání dalšího parametru `signLineProbThreshold`, skrze který lze nastavit vlastní minimální sémantickou významnost hranice pro to, aby mohla být použita pro rozdělení segmentu (místo pevně dané minimální hodnoty 50% definované v sekci 4.3). Lze předpokládat, že dopad tohoto parametru bude velmi podobný parametru `edgeProbabilityThreshold`, nicméně implementace použitá publikací [18] parametrizuje tuto hodnotu také, je tedy pro kompletnost zahrnut.

Implementace rekurzivního algoritmu pro tvorbu výsledného stromu segmentací už obsahuje několik zajímavostí oproti předepsané verzi. První z nich je čistě logická – originální algoritmus 3 tak, jak je publikován v [12], využívá výsledek rozdělení, který je v odlišném směru než aktuální směr dělení, pouze k určení dalšího kroku. Samotnou dvojici segmentů, která vznikla dělením v tomto směru, pak zahazuje. Přitom v rekurzivně volaném algoritmu je nad tímto segmentem provedeno stejné dělení znovu. Jediným rozdílem je, že protože rekurzivně volaný algoritmus dělí v opačném směru, tak je již výsledek použit celý.

To platí ovšem za předpokladu, že rozdělení dopadne stejným výsledkem. Algoritmus rozdělení segmentu ale využívá algoritmus 1, jehož implementace, popsaná v sekci 6.2 výše, způsobuje nedeterminističnost výsledku. Díky tomu by mohlo rozdělení ovlivnit strukturu X-Y stromu takovým způsobem, že rozdělení segmentu na podsegmenty je „odloženo“ o tolik

úrovní, kolikrát by byl rekurzivně volán segmentační algoritmus, dokud by výsledný směr rozdělení nebyl pro 2 po sobě jdoucí volání stejný. Výsledné rozdělení daného segmentu by ovšem nijak ovlivněno nebylo – pouze by byl vybrán výsledek posledního rozdělení.

Takový jev by se vyskytoval častěji s nižšími hodnotami parametru `monteCarloTrials`. Teoreticky by mohl způsobit i nekonečný běh segmentace – v případě, že by pro daný segment k rozdělení byl vybrán v každém rekurzivním volání segmentačního algoritmu jiný směr, než v předchozím volání. Ovšem i pokud by pro daný segment algoritmus pro jeho rozdělení vracel směr rozdělení náhodně s (pro tento případ) ideální pravděpodobností 50%, pravděpodobnost nekonečného běhu se exponenciálně snižuje s počtem rekurzivních volání.

I bez ohledu na dopad tohoto jevu na výsledek segmentace způsobuje nevyužití výsledku rozdělení segmentu, který je potřebný při dalším rekurzivním volání, jisté zpomalení algoritmu. V rámci implementace je proto jev ošetřen takovým způsobem, že výsledek rozdělení segmentu s různým směrem od aktuálního směru dané úrovně X-Y stromu je uložen a během rekurzivního volání segmentačního algoritmu předán jako parametr. Segmentační algoritmus pak s tímto rozdělením pracuje jako s výsledkem prvotního rozdělení daného segmentu.

Protože k tvorbě segmentačního stromu algoritmus pracuje pouze s kořenovým uzlem dané části stromu (a jednotlivá rekurzivní volání tak mezi sebou nenesou žádnou závislost), je možné je zpracovávat paralelně. K tomuto účelu je rekurzivní algoritmus implementován v rámci podtřídy `SegmentTask`, rozšiřující třídu `RecursiveTask`. Každé rekurzivní volání je tak implementováno jako vytvoření nové instance této třídy, které jsou paralelně spuštěny nezávisle na sobě.

Třída `CormierSegmentation` pak implementuje metodu `run`, jejímž vstupem je obrázek segmentované stránky. Nad obrázkem pomocí třídy `EdgeDetector` získá matice pravděpodobností lokální významnosti a následně začíná s tvorbou segmentačního stromu voláním segmentačního algoritmu s kořenovým uzlem (se segmentem pokrývajícím celou stránku), výsledkem iniciálního volání rozdělení tohoto segmentu (protože algoritmus vyžaduje počáteční rozdělení segmentu jako parametr z důvodů uvedených výše v této sekci) a maticemi pravděpodobností lokální významnosti obrázku. Výsledkem volání je výsledný strom segmentů, který je vrácen jako výsledek segmentační metody.

Kapitola 7

Vyhodnocení implementované metody

Vyhodnocení a detailní analýza implementované metody je stěžejní částí této práce. Přístup k vyhodnocení zahrnující volbu vhodné datové sady a prezentaci metrik použitých napříč touto kapitolou je shrnut v sekci 7.1. Samotné vyhodnocení následně začíná sekcí 7.2, která ověřuje implementaci metody vůči publikovaným materiálům. Do největší hloubky pak sahá sekce 7.3 obsahující důkladnou analýzu chováni metody a následné experimentování nad různými nastaveními jejích parametrů. Závěrečná sekce 7.4 nakonec analyzuje rychlost metody, zejména v kontextu optimalizací aplikovaných během implementace.

7.1 Způsob vyhodnocení

Získání obecných, objektivně měřitelných a srovnatelných výsledků segmentačních metod není jednoduché. Prvním problémem tohoto úkolu je samotná definice segmentů. Které rozdělení stránky je objektivně správně? Dle publikace [17] jsou segmenty nejčastěji definovány jako vizuální bloky se souvislým obsahem, který se liší od zbytku stránky. Zmiňuje ovšem ale také to, že tato definice není ustálená a v různých publikacích se liší. Uvedeným příkladem jsou publikace implementované metody ([14], [13], [12]), které definují segment pomocí hran. Jde také o zaměření metody segmentace a o následné případy využití jejích výsledných segmentací. Zejména pokud jde o ideální granularitu segmentů. [24]

Tento problém způsobuje, že v podstatě není možné vytvořit univerzální jednotnou datovou sadu se segmentacemi webových stránek, které by byly cílem všech metod pro všechny možné využití. Různé publikace tak tvoří vlastní menší datové sady, na kterých demonstrují jimi prezentované metody. Opět mohou být příkladem publikace implementované metody, nebo např. [21]. Publikace [17] se pak snaží přijít s větší, univerzálnější datovou sadou, vytvořenou na základě manuálních segmentací velkým počtem uživatelů s cílem zachytit segmenty takovým způsobem, jakým je vnímají uživatelé webových stránek.

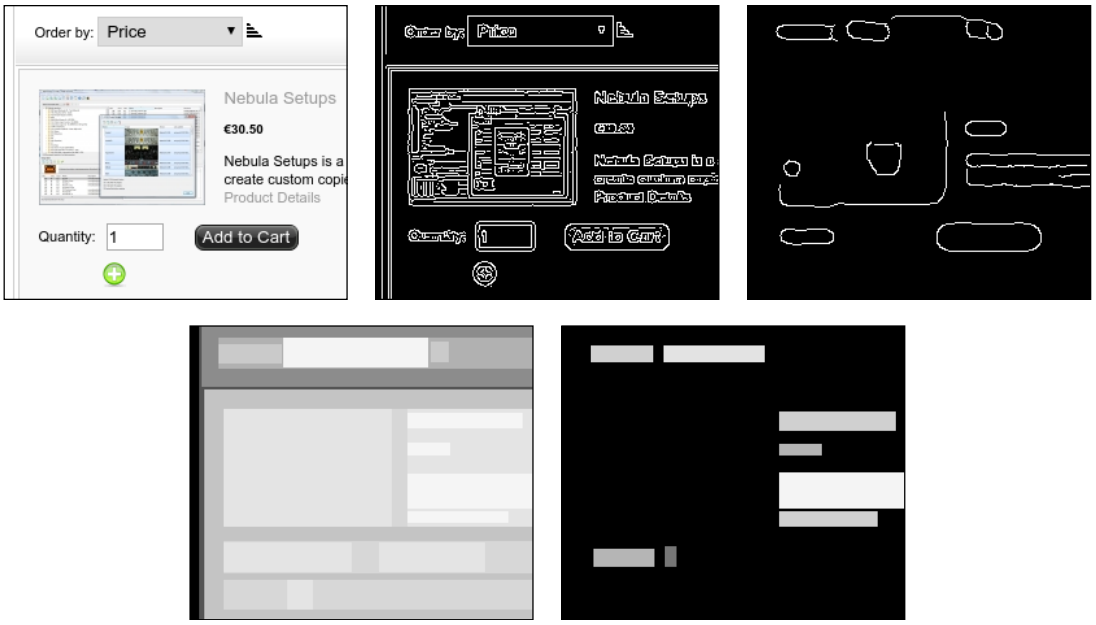
Navazujícím problémem je způsob srovnání výsledků segmentačních metod. Jako výchozí vodítko pro získání měřitelné hodnoty lze použít překryv jednotlivých segmentů. Od tohoto se odráží postup použitý v sekci 7.2, kde pro srovnání výsledků stejných metod lze počítat přímo s překryvy každého segmentu a získat tím poměrně přesný poměr „stejnosti“ těchto segmentací. Srovnání segmentů vytvořených různými způsoby je ovšem obtížnější – např. pokud jedna z metod vyplní segmenty celou stránku, zatímco druhá může části stránek vynechat, nebo pokud první metoda tvoří strom segmentů, zatímco druhá metoda

nevytváří žádnou hierarchickou strukturu. Jak se v takových případech chovat? Nepočítat s prostorem mezi segmenty? Ignorovat segmenty pod určitou hranicí stromové hierarchie? Se svým řešením přichází opět publikace [17], která se snaží upravit metriky používané pro srovnání výsledků shlukování tak, aby byly lépe aplikovatelné na segmenty.

Následující podsekce obsahují popis kroků podniknutých v rámci vyhodnocení implementované metody, které adresují výše zmíněné problémy. Protože ovšem nejde o problémy s univerzálním řešením, představují podsekce také úskalí použitých řešení.

7.1.1 Datová sada

Jako datová sada pro vyhodnocování byla zvolena již výše zmíněná datová sada vytvořená publikací [17]. Jedná se o poměrně novou datovou sadu o 8 490 webových stránkách z různých oblastí, o různých strukturách i v různých jazycích. Pro každou stránku obsahuje datová sada její snímek, který sám o sobě jako vstup pro implementovanou metodu stačí. Rovněž ovšem obsahuje další data o stránkách, jako jsou souřadnice jednotlivých elementů DOM[1] modelu každé z nich, jejich celé zdrojové kódy, nebo pozice hran získaných Cannyho detektorem hran ze snímků stránek. [10] Všechny druhy dat stránek v datové sadě jsou znázorněny obrázkem 7.1. Publikace této datové sady rovněž zveřejňuje nástroje pro práci s těmito vstupními daty, které jsou jedním z důvodů její volby. O jejich využití dále pojednává podsekce 7.1.2.



Obrázek 7.1: Vizualizace dat obsažených v datové sadě pro každou stránku, které lze využít jako atomické elementy pro publikací navržený výpočet ke srovnání segmentací (viz podsekce 7.1.2). Zleva: obrázek stránky (`area`); jemné hrany získané Cannyho detektorem hran (`edges_fine`); hrubé hrany získané stejnou metodou (`edges_coarse`); elementy modelu DOM (`nodes`); textové elementy modelu DOM (`chars`). Označení v závorkách bude dále použito pro identifikaci jednotlivých druhů dat. Převzato z [17].

[1] DOM – Document Object Model (objektový model dokumentu)

Vzorové segmentace stránek v datové sadě byly vytvořeny ručně uživateli webových stránek pomocí *crowdsourcing* kampaně. Každý uživatel dostal pokyny k rozdělení stránky na takové obdélníkové segmenty, které seskupují části stránek patřících k sobě. Konkrétně bylo pro každou stránku takto ručně vytvořeno 5 různých segmentací, které dále prošly kontrolami kvality, byly napasovány na DOM elementy daných stránek a jejich kombinací vznikly výsledné segmentace, označené datovou sadou jako vzorové segmentace. [17]

Tyto segmentace jsou uložené ve vlastním formátu, jehož struktura je znázorněna v úryvku 7.1. Výstupní formát segmentací implementované metody lze konvertovat do této podoby, během které je ztracena pouze explicitní informace o zanoření segmentů. V případě potřeby by navíc bylo možné tuto informaci zrekonstruovat díky vlastnosti segmentace, že hranice segmentů nikdy nepřesahují hranice jejích rodičovských segmentů. Konverze na tento formát bude potřeba pro možnost použití skriptů zveřejněných v rámci publikace [17] (jejichž využití je popsáno později), které s tímto formátem pracují.

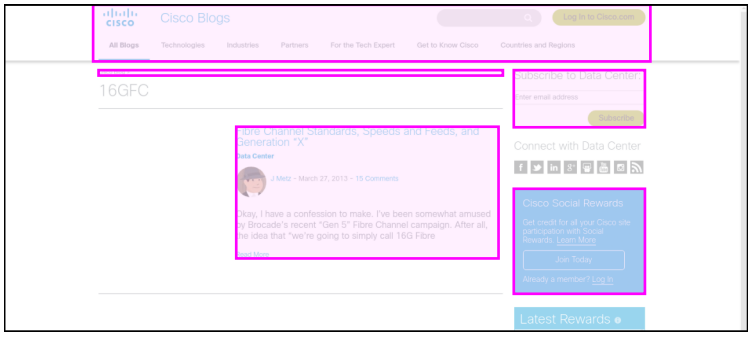
```
{
  "height": 1106,
  "width": 1366,
  "id": "000430", // Identifikátor segmentované stránky.
  "segmentations": {
    "cormier": [ // Seznam segmentů dané metody.
      [[[201, 0], [201, 103], [1176, 103], [1176, 0], [201,  0]]],
      [[[0, 191], [0, 529], [426, 529], [426, 191], [0, 191]]],
      [[[0, 530], [0, 1106], [936, 1106], [936, 530], [0, 530]]],
      [[[937, 0], [937, 337], [1089, 337], [1089, 0], [937, 0]]],

      // ...

      [[[937, 0], [937, 174], [1177, 174], [1177, 0], [937, 0]]],
      [[[937, 53], [937, 450], [1177, 450], [1177, 53], [937, 53]]]
    ]
  }
}
```

Výpis 7.1: Ukázkový soubor se segmentací stránky z datové sady s identifikátorem `000430`, která má výšku 1 106 a šířku 1 366 bodů, serializované do formátu ve značkovacím jazyce JSON, který dodržují vzorové segmentace stránek využité datové sady. Tento formát ukládá segmentace do atributu `segmentations`, kde pod identifikátory jednotlivých segmentačních metod (v ukázce pod atributem `cormier`) ukládá seznam segmentů jako seznamy souřadnic bodů tvořících jejich obvod.

Přístup tvorby vzorových segmentací se ovšem od způsobu segmentace implementovanou metodou značně liší a obvykle jsou jeho výsledkem segmentace, které ani implementovanou metodou vzniknout nemohou. Např. metoda tvoří segmenty takovým způsobem, že postupně dělí segmenty na poloviny – takové omezení vzorové segmentace nemají. To sice nijak nebrání objektivnímu vyhodnocení kvality segmentace, ale je nutno podotknout, že implementovaná metoda není schopná dosáhnout vzorových výsledků jako např. v obrázku 7.2.



Obrázek 7.2: Vzorová segmentace, vizualizovaná pomocí okrajů jednotlivých segmentů purpurovou barvou, kterou implementovaná metoda není schopna vytvořit.

Vzorové segmentace vykazují ovšem další vlastnost, kterou implementovaná metoda nedokáže napodobit a tudíž díky které jsou již výsledky srovnání negativně ovlivněny. Jedná se o prázdný prostor mezi segmenty. Tato vlastnost je pro srovnání se vzorovými segmentacemi ošetřena dvěma různými způsoby. Buď přizpůsobením vzorové segmentace vyplněním prázdného prostoru, nebo naopak přizpůsobením srovnávané segmentace vytvořením prázdného prostoru. Oba způsoby se navzájem vylučují, proto budou v rámci vyhodnocení prezentovány 2 druhy výsledků, každý z nich získaný po ošetření segmentací jedním z těchto způsobů.

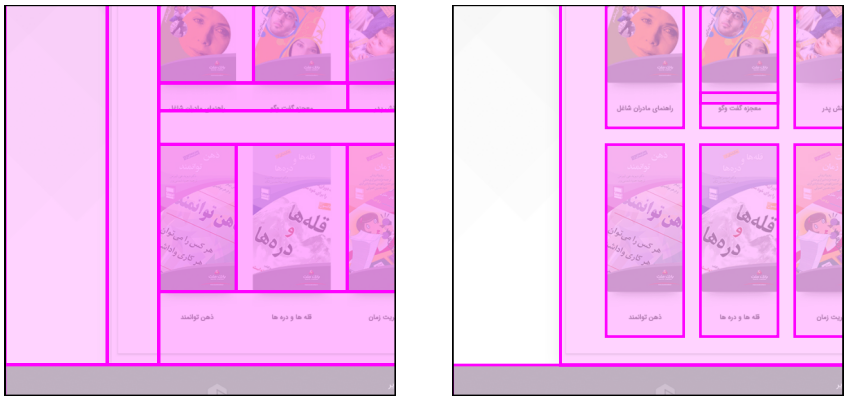
K vyplnění prázdného prostoru vzorových segmentací lze přistoupit tak, že je vytvořen další segment, který svou plochou pokrývá celý prázdný prostor. Tento způsob odpovídá přístupu implementované metody k prázdným částem stránky (tj. také pro ně vytváří segmenty, které jsou oddělené od zbytku stránky s obsahem). Struktura prázdného prostoru může mít ovšem vlastnosti, které tvorbu takového segmentu komplikují a je potřeba je ošetřit.

První takovou vlastností je, že prázdný prostor nemusí tvořit jednu souvislou plochu, ale může být složený z více částí, které na sebe nijak nenavazují. Takové případy jsou vyřešeny jednoduše tak, že je vytvořen nový segment pro každou takovou část.

Druhá, výrazně častější vlastnost, je výskyt „děr“ uvnitř prázdného prostoru. Ty vznikají, pokud jsou segmenty ze všech stran obklopeny prázdným prostorem. Tato vlastnost je problematická, protože formát uložení vzorových segmentací datové sady nepodporuje segmenty, které by v sobě měly díry. Možným ošetřením této vlastnosti je prosté odstranění děr, to ovšem způsobuje překryv prázdného prostoru se segmenty, které díry způsobily, a tím negativní efekt na výsledky srovnání s takto upravenou vzorovou segmentací. Díry jsou tak zachovány, ale zároveň upraveny tak, aby šel prázdný prostor uložit do formátu vzorových segmentací. Toho je docíleno propojením každé z děr s obvodem segmentu prázdného prostoru, ve kterém se nachází, pomocí jeho řezu tak, jak je vidět na obrázku 7.3. Řezy nemají žádnou tloušťku, díky čemuž nijak neovlivňují obsah segmentu prázdného prostoru. Zároveň jsou prováděny takovým způsobem, aby nedělily segmenty prázdných prostorů na menší. Tím pádem neovlivňují výsledky použitých metrik (popsaných později v podsekci 7.1.2)

Obrázek 7.3: Výřez stránky z datové sady s identifikátorem `000430` s vizualizací její vzorové segmentace. Vlevo je segmentace bez jakýchkoliv úprav, vpravo je ta samá segmentace po aplikování vyplnění prázdného prostoru. V pravém výřezu lze pozorovat v segmentu vyplňující prázdný prostor řezy, které zajišťují, že tento segment v sobě nemá žádné díry a přitom zůstává jednotnou plochou.

Druhý způsob řešení prázdného prostoru, vytvoření prázdného prostoru ve srovnávané segmentaci, je odvozen od způsobu tvorby vzorových segmentací. Jak je zmíněno výše, součástí jejich tvorby je napasování segmentů na DOM uzly stránky. Součástí zveřejněných materiálů publikace [17] je také skript pro aplikování stejného procesu napasování na kteroukoliv segmentaci (za podmínky, že je ve formátu vzorových segmentací). Použitím tohoto skriptu na výstupní segmentace implementované metody má za následek úpravu segmentů takovým způsobem, aby pokrývaly DOM uzly, které mají s danými segmenty nejvhodnější překryv. Výsledek lze vidět na demonstraci v obrázku 7.4. Vedlejším efektem je právě vytvoření prázdného prostoru okolo takových segmentů.



Obrázek 7.4: Výřez stránky z datové sady s identifikátorem `000280` s vizualizací její segmentace. Vlevo je segmentace bez jakýchkoliv úprav, vpravo je ta samá segmentace po aplikování napasování na DOM uzly dané stránky.

I přes tyto úpravy je také potřeba zmínit, že přestože je datová sada tvořena s úmyslem srovnání a vyhodnocování různých segmentačních metod, tak (jak již bylo řečeno výše) různé metody mohou tvořit segmenty za různými účely. Hlavní motivací implementované metody bylo zaměření na její aplikaci pro transformaci stránek skrze asistivní technologie pro uživatele s tělesným postižením. [14] Ideální segmentace se může pro tento účel v ur-

čitých situacích lišit od segmentací vytvořených běžnými uživateli, kterými byly vytvořeny segmenty zvolené datové sady.

Zároveň stojí za zmínku, že podle publikace [17] pouhé 3% segmentací obsahují vnořené segmenty. Přitom vzhledem ke stromové hierarchii výstupního formátu segmentů implementovaná metody obsahuje vnořené segmenty v každém z jejích výsledků.

Přestože tyto faktory nezabraňují použití zvolené datové sady pro získání měřitelných údajů, je potřeba na ně myslet během vyhodnocování výsledků srovnání segmentů datové sady s výstupními segmenty implementované metody.

Vzhledem ke kombinaci časové náročnosti implementované metody a časovým omezením k vyhodnocení není využito všech 8 490 stránek, ale její podmnožina. Z podobných důvodů ke zmenšení datové sady k vyhodnocení přistupuje i publikace využívající tuto datovou sadu [18]. Ta ke srovnání používá prvních 1 000 stránek z datové sady. Také ovšem srovnává pouze 4 různé kombinace parametrů implementované metody. K vyhodnocení implementace v rámci této práce tak vybrán vzorek o 100 stránkách – stále dostačující počet pro sledování změn v chování metody s různým nastavením a na různých případech.

K tomu, aby počet stačil, bylo ovšem potřeba omezit podobné stránky a zajistit tak různorodost mezi jednotlivými vzorky. Stránky ze stejných domén mají tendenci se v datové sadě vyskytovat u sebe, je proto vzorek 100 stránek vybrán takovým způsobem, že je vybrána každá desátá stránka datové sady (navíc jsou přeskočeny některé nevhodné stránky).

7.1.2 Interpretace výsledků

K interpretaci a zpracování výsledků je přistupováno několika způsoby. Pro získání měřitelných a srovnatelných hodnot jsou aplikována jak vlastní řešení, tak již prověřená řešení jiných publikací. Důvody za takto získanými hodnotami a případnými zajímavějšími jevy jsou dále rozebírány ruční analýzou, jejíž výsledky jsou taktéž zahrnuty v rámci vyhodnocení, pakliže stojí za zmínku.

Ať jde o kterýkoliv ze způsobů získání takových hodnot, ze srovnávaných segmentací implementované metody jsou nejprve odstraněny kořenové segmenty. Jde vždy o segmenty pokrývající celou plochu segmentované stránky a nepřináší tak žádnou informační hodnotu.

Zmíněným vlastním řešením pro získání měřitelných výsledků je skript srovnávající výsledné segmentace, se zaměřením na překryv jednotlivých segmentů. Vstupem tohoto skriptu jsou tedy 2 soubory s výsledky segmentačních metod. Následně načte jejich jednotlivé segmenty, uloží si jejich plochu a najde všechny překryvy mezi segmenty z těchto dvou souborů (tak, aby byl překryv mezi segmentem z prvního a z druhého souboru). Pro každý takový překryv si pak krom segmentů v překryvu zapamatuje také poměr, který představuje společná plocha obou segmentů vůči součtu jejich ploch – tj. kolik % z jejich celkové plochy překryv zabírá.

Samotný výpočet pak probíhá takovým způsobem, že skript iteruje postupně po těchto překryvech, seřazených sestupně podle poměru překrytí, a postupně je zpracovává. Zpracování každého překryvu znamená započítání jejich hodnot (celkové a sdílené plochy) do výsledné sumy a jejich odstranění ze seznamu nalezených překryvů. Řazení hraje roli z důvodu, že po zpracování každého překryvu je odstraněn ze seznamu nejen daný překryv, ale také všechny překryvy, které obsahují některý ze segmentů zpracovaného překryvu, tj. každý segment je započítán do výsledných hodnot maximálně jednou.

Pro zpracování všech překryvů zbývá zpracovat tzv. nadbytečné segmenty. Jedná se o segmenty, pro které buď nebyl nalezen žádný překryv (a tudíž nikdy nebyly v seznamu

překryvů ke zpracování), nebo pro ně byl nalezen 1 nebo více překryvů, ale byly odstraněny ze seznamu překryvů dříve, než byly zpracovány – tj. měly menší poměr překrytí v každém z jejich překryvů, než jiné překryvy se segmenty, se kterými byly v překryvu. Tyto segmenty jsou jednoduše zpracovány tím způsobem, že je jejich plocha započtena do celkové sumy plochy, bez započtení jakékoliv plochy překryvu.

Výsledné hodnoty skriptu obsahují celkovou plochu všech segmentů, celkovou plochu zpracovaných překryvů a počet všech a všech nadbytečných segmentů obou vstupních segmentací. Stěžejní hodnotou je pak celkový poměr překryvu obou vstupních segmentací, dále označován jako O_{sum}, který je vypočten jako poměr mezi celkovou plochu zpracovaných překryvů a celkovou plochou všech segmentů. Tato hodnota představuje, kolik % plochy bylo nasegmentováno shodně pro oba vstupní výsledky segmentace.

Další zahrnutou hodnotou, dále označovanou jako O_{avg}, je pak další, obdobný poměr, který ovšem nebere v potaz nadbytečné segmenty (představuje tedy poměr překryvu pouze těch segmentů, u kterých byl nějaký překryv zpracován). Druhý z poměrů bude přirozeně vykazovat lepší výsledky a může být užitečný pro případy srovnání segmentací s různě nastavenou granularitou, kdy jeden výsledek segmentace bude mít segmenty dělené do hlubších úrovní segmentačního stromu, než druhý.

Pro krajní případ, kdy obě ze srovnávaných segmentací nemají ani jeden segment, skript vrací hodnotu $O_{sum} = 1$ (segmentace jsou shodné). Hodnotu O_{avg} pak nedefinuje.

Vzhledem k tomu, jakým způsobem tento skript funguje, je jeho hlavní výstup užitečný pouze v případě, že oba srovnávané výstupy segmentačních metod mají podobné vlastnosti. V případě použití zvolené datové sady popsané v sekci 7.1.1 jde zejména o hierarchii segmentů. Segmenty v datové sadě nevykazují téměř žádnou hierarchickou strukturu, zatímco výstupy implementované metody jsou vždy striktně hierarchické. Výsledkem při srovnání segmentací z těchto dvou zdrojů jsou pak horší výsledné hodnoty při hlubších stromech segmentací implementované metody.

Pokud jsou ovšem obě vstupní segmentace tvořeny identickým, nebo alespoň velmi podobným přístupem, pak lze výsledky tohoto skriptu pokládat za poměrně spolehlivou reprezentaci míry, jak moc jsou segmentace skutečně identické. K tomuto účelu je skript využit zejména v rámci sekce 7.2 k ověření výstupů implementované metody.

Mimo datovou sadu publikace [17] také navrhuje způsob měření podobnosti dvou různých segmentací použitím rozšířeného *BCubed* F1-skóre F_{B^3}, které bylo původně navrženo publikací [2] pro vyhodnocení podobnosti výsledků shlukovacích metod.

Obdobným způsobem jako u standardního F1-skóre je její výsledek dosažen harmonickým průměrem rozšířených *BCubed* precision P_{B^3} a recall R_{B^3}. Metrika P_{B^3} nebere v potaz přílišnou segmentaci, tj. případy, kdy posuzovaná segmentace S dělí segmenty vzorové segmentace S^* na menší. Naopak metrika R_{B^3} nebere v potaz nedostatečnou segmentaci, tj. kdy posuzovaná segmentace S spojuje segmenty vzorové segmentace S^* do větších. Výpočet těchto metrik je pak definován následovně: [17]

$$P_{B^3}(S, S^*) = \frac{1}{|E^S|} \sum_{e \in E^S} \left(\frac{1}{|E^S_e|} \sum_{e' \in E^S_e} \left(\frac{\min\left(|S_e \cap S_{e'}|, |S^*_e \cap S^*_{e'}|\right)}{|S_e \cap S_{e'}|} \right) \right) \tag{7.1}$$

$$R_{B^3}(S, S^*) = P_{B^3}(S^*, S) \tag{7.2}$$

$$F_{B^3}(S, S^*) = \frac{2 \cdot P_{B^3}(S, S^*) \cdot R_{B^3}(S, S^*)}{P_{B^3}(S, S^*) + R_{B^3}(S, S^*)} \tag{7.3}$$

kde

- e je element, který je seskupován do segmentů. V případě použití dat ze zvolené datové sady může jít o body obrázku stránky (`area`), hrany (`edges_coarse`, `edges_fine`), či elementy modelu DOM (`nodes`, `chars`) (viz sekce 7.1.1).
  - Pro zjednodušení bude druh elementu dále v textu uváděn následujícím způsobem: $F_{B^3\texttt{elem}}$, kde *elem* je použitý element k výpočtu metriky F_{B^3}.
- $S_e \subseteq S = \{s \mid s \in S \wedge e \in s\}$ je podmnožina segmentů S obsahující element e.
- $E^S \subseteq E = \{e \mid e \in E \wedge S_e \neq \emptyset\}$ je podmnožina elementů E, které jsou součástí alespoň jednoho ze segmentů S.
- $E_e^S \subset E = \{e' \mid e' \in E \wedge S_e \cap S_{e'} \neq \emptyset\}$ je podmnožina elementů E, které sdílí alespoň jeden ze segmentů S s elementem e.

Použití metrik na segmenty místo shluků teoreticky nic nebrání. Jejich definice by měla zvládat ohodnocení částečných či překrývajících se segmentů, zanořených segmentů, ale i triviální segmentace, jako rozdělení každého elementu do vlastního segmentu, či všech elementů do jediného segmentu. Navíc nemusí být použity pouze k ověření kvality segmentace oproti vzorové, ale obecně ke změření podobnosti různých výstupů segmentačních metod. Lze je tedy použít jak pro získání kvality segmentace oproti zvolené datové sadě, tak i pro ověření výstupů implementované metody (viz sekce 7.2). [17]

Definice ovšem předpokládá existenci alespoň jednoho segmentu u obou srovnávaných segmentací. Vzhledem k odstraňování kořenových segmentů z výsledků implementované metody může nastat případ, kdy některé ze srovnávaných segmentací mít segmenty nemusí. Takové případy jsou ošetřeny následovně:

- $F_{B^3} = R_{B^3} = P_{B^3} = 0$ pro případ, kdy žádné segmenty neobsahuje pouze jedna ze srovnávaných segmentací.
- $F_{B^3} = R_{B^3} = P_{B^3} = 1$ pro případ, kdy žádné segmenty neobsahují obě srovnávané segmentace (a jsou tudíž shodné).

Součástí publikace [17] je pak zveřejněna také implementace pro výpočet těchto metrik[2], která zahrnuje i optimalizace pro zlepšení výkonu pro případ většího počtu elementů (jako v případě bodů větších obrázků stránek) a která byla použita pro výpočet během vyhodnocení.

Použití publikovaných nástrojů k vyhodnocení přináší potenciální výhodu srovnatelnosti dosažených výsledků s jinými. Nabízí se třeba srovnání s výsledky publikace [18], srovnávající výsledky několika segmentačních metod. Je třeba ovšem uvažovat některé rozdíly oproti vyhodnocení v rámci této práce, způsobené zaměřením na pouze implementovanou metodu za účelem přesnějšího vyhodnocení:

- Jako vstup jsou použity originální rozměry stránek, namísto jejich oříznutí (nebo naopak vycpání) na výšku 4 096 bodů.
- Z výstupů segmentační metody je vždy odstraněn kořenový segment.

[2] webis-de/cikm20-web-page-segmentation-revisited-evaluation-framework-and-dataset – https://github.com/webis-de/cikm20-web-page-segmentation-revisited-evaluation-framework-and-dataset#algorithm-evaluation

- Prázdný prostor vzorových segmentací je vyplněn (jak je popsáno v podsekci 7.1.1).
  - Napasování srovnávaných segmentací na DOM uzly probíhá ovšem identickým způsobem.

7.2 Srovnání s publikovanou implementací

V rámci této práce je rozšiřován nástroj FitLayout o segmentační metodu popsanou v kapitole 4. Cílem implementace této metody je držet se tohoto popisu tak, aby mohly být její výsledky použité pro další účely. Určení splnění tohoto cíle je ovšem problematické. Jedním z možných způsobů by bylo srovnání výsledků dosažených publikací implementované metody s výsledky implementace v rámci této práce, ty ale nejsou veřejně dostupné.

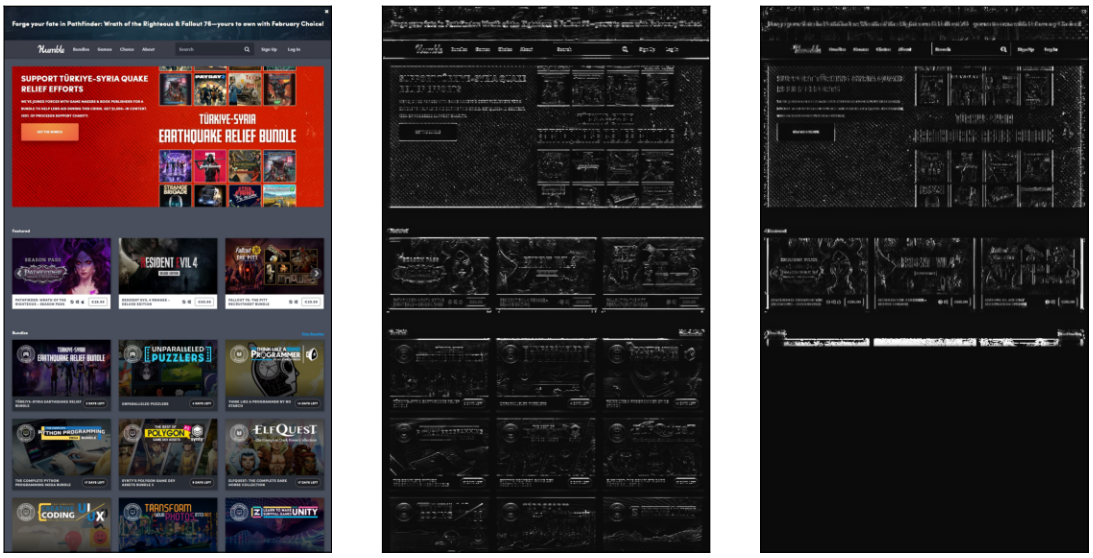
Součástí publikace [18] je ovšem zveřejněná implementace této metody[3], údajně poskytnuta autorem publikací, ze kterých popis segmentační metody v této práci čerpá. Výstup této implementace je ve stejném formátu, jako vzorové segmentace použité datové sady (popsaném v podsekci 7.1.1). Díky tomu lze tyto výsledky k účelu této práce brát jako vzorové a mohou být využity ke srovnání, jak moc implementace odpovídá popisované metodě (v ideálním případě by se měly výsledky obou implementací shodovat) a to pro libovolný vstupní obrázek.

Ani tento způsob ovšem není bezchybný. Během srovnávání výsledků bylo vypozorováno, že zveřejněná implementace má problém se segmentací spodních částí některých obrázků. Po detailnějším průzkumu a testování bylo zjištěno, že se tato anomálie projevuje u „vyšších“ obrázků a je pravděpodobně způsobena chybou ve výpočtu lokální významnosti hran. Ten během zpracovávání obrázku po určitém řádku začíná určovat pravděpodobnost lokální významnosti vertikálních hran všech bodů jako 0%, jak je vidět na obrázku 7.5. Index tohoto řádku během testování zdánlivě odpovídal šířce obrázku. Následné testování obrázků s maximální výškou rovnou vlastní šířce již tuto anomálii nevykazovalo. Proto byly pro účel srovnání implementací vstupní obrázky ořízntuty takovým způsobem, aby splňovaly tuto vlastnost.

Dalším nedostatkem této dostupné verze je absence Gaussovy pyramidy ve výpočtu Sobelova operátoru. Její implementace tak tímto způsobem být ověřena nemůže.

Po aplikování oříznutí byly nad datovou sadou spuštěny nad vzorkem 100 stránek ze zvolené datové sady s výchozími hodnotami parametrů obě implementace. Tyto hodnoty parametrů, vypsané v tabulce 7.1, jsou nastaveny tak, aby odpovídaly výchozímu nastavení publikované implementace.

[3] webis-de/ecir21-an-empirical-comparison-of-web-page-segmentation-algorithms – https://github.com/webis-de/ecir21-an-empirical-comparison-of-web-page-segmentation-algorithms#cormier-et-al

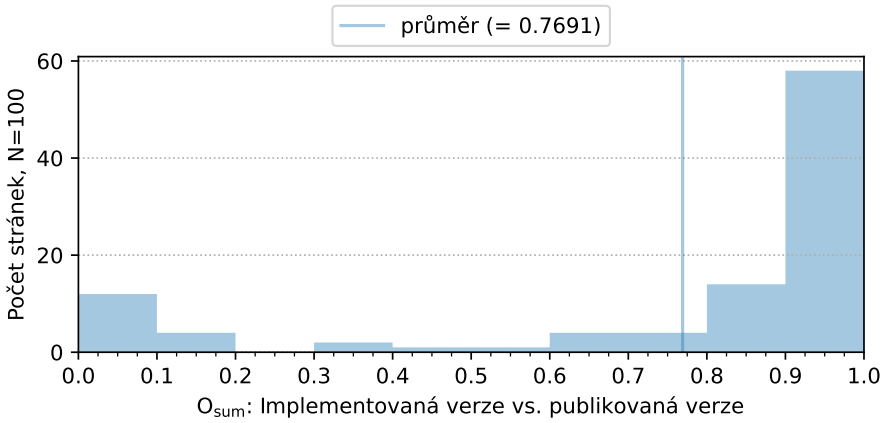


Obrázek 7.5: Znázornění chyby v publikované implementaci na obrázku stránky https://www.humblebundle.com/. Vlevo je vstupní obrázek segmentované stránky. Napravo od něj jsou vizualizace pravděpodobnosti lokální významnosti hran v jednotlivých bodech tohoto obrázku. Prostřední vizualizace zobrazuje tuto hodnotu pro horizontální hrany, napravo pak pro vertikální hrany. Na vizualizaci napravo lze pozorovat, že pod určitou hranicí je pravděpodobnost vždy 0% (černá barva).

Parametr	Hodnota
`halfWindowWidth`	45
`standardDeviation`	0.1
`priorEdgeProbability`	0.01
`pyramidLevels`	1
`maxLineLength`	256
`edgeProbabilityThreshold`	0.3
`monteCarloTrials`	100
`minSegmentLength`	45
`signLineProbThreshold`	0.5

Tabulka 7.1: Použité nastavení parametrů pro srovnání implementací.

Výsledky obou implementací byly srovnány proti sobě (bez žádných dodatečných úprav, které budou aplikovány při srovnávání se vzorovými segmentacemi) pro získání objektivní míry jejich podobnosti. Z výsledků srovnání v grafu 7.6 lze pozorovat, že u většiny stránek jsou segmentace takřka shodné. I přes ošetření výše zmíněného problému lze ovšem pozorovat drobné odlišnosti, u menšího počtu stránek jsou pak výsledky zcela odlišné. Spolu s hodnotami v tabulce 7.2 lze říci, že se shoda s výstupem publikované implementace pohybuje okolo 80 %.



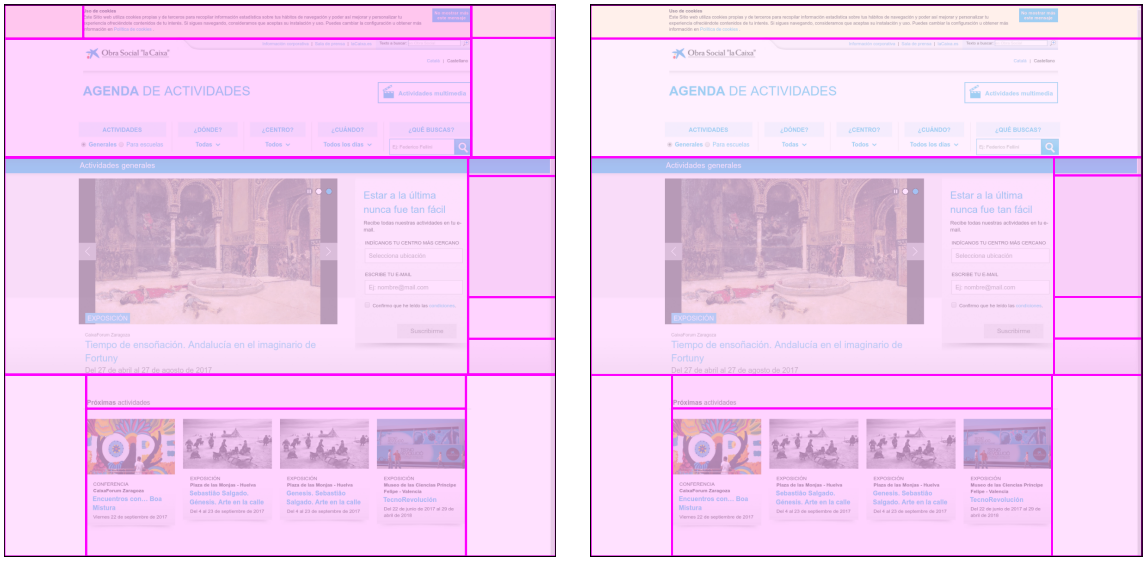
Obrázek 7.6: Histogram znázorňující naměřené hodnoty O_{sum} ze srovnání výsledků obou implementací nad daným vzorek dat. Více než polovina výsledků jsou téměř shodné ($O_{sum} > 90\,\%$), zatímco u 12 výsledků jsou naopak úplně odlišné ($O_{sum} < 10\,\%$).

Drobné odlišnosti mezi výsledky *BCubed* metrik s odlišnými elementy (v tabulce 7.2) vykazují také jisté informace o shodě mezi výsledky segmentací. Pro elementy `area` mají metriky nejlepší hodnoty – shoda segmentace okolního prostoru, kde jiných druhů elementů není tolik, je tím pádem vyšší. Naopak pro elementy `edges_fine` a `edges_coarse` jsou hodnoty horší. To lze vysvětlit tím, že metoda používá hrany jako hlavní vodítko dělení a tudíž se na nich často nachází hranice segmentů. Tyto elementy jsou tedy na místech, kde je nejpravděpodobnější, že se segmentace budou lišit.

O_{sum}	**76,91 %**		
Element	P_{B^3}	R_{B^3}	F_{B^3}
`area`	82,12 %	85,94 %	**83,48 %**
`edges_fine`	81,28 %	83,59 %	**81,60 %**
`edges_coarse`	80,78 %	82,06 %	**80,65 %**
`nodes`	80,50 %	85,55 %	**81,94 %**
`chars`	80,48 %	86,04 %	**82,29 %**

Tabulka 7.2: Přehled průměrných výsledků metrik naměřených ze srovnání výsledků obou implementací nad daným vzorkem dat. Výsledek metriky O_{sum} vychází s hodnotou $76,91\,\%$ nejhůře. *BCubed* metriky vykazují o něco lepší výsledky, kde nejvyšší z nich je hodnota $F_{B^3} = 83,48\,\%$.

Obrázek 7.7 demonstruje tyto drobné odlišnosti na dvojici segmentací stejné stránky oběma verzemi metody. Pro spodní dvě třetiny stránky obě verze vytvořily stejné segmentace. Horní třetinu však začala implementovaná verze segmentovat vertikálně, zatímco publikovaná verze začala horizontální segmentací. Implementovaná verze pak navíc pokračovala vytvořením několika dalších segmentů, zatímco publikovaná verze nikoliv. Mnoho odlišností mezi výsledky je způsobeno podobným chováním, které je důsledkem drobných implementačních odlišností.



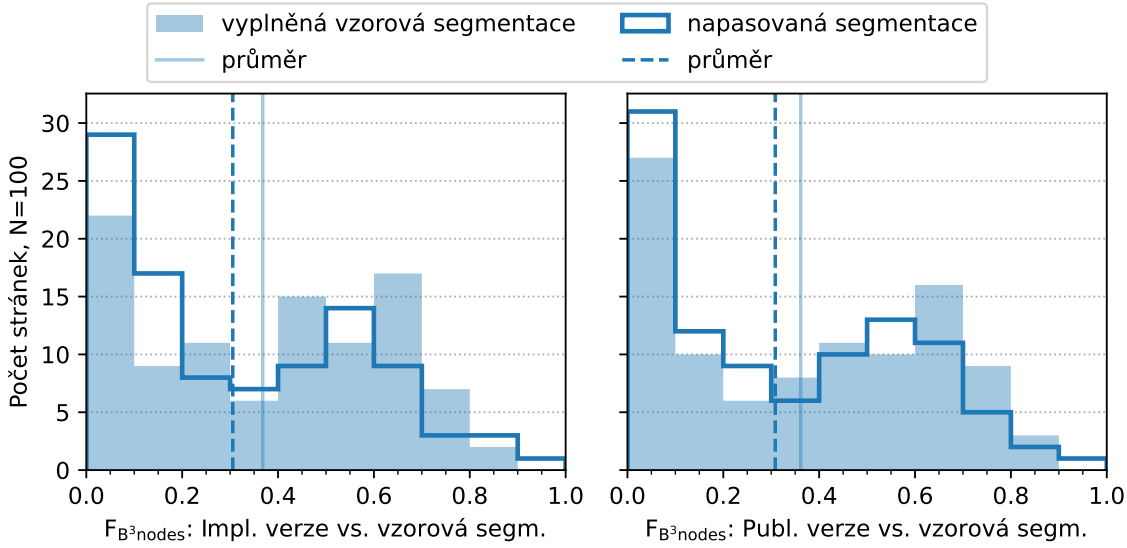
Obrázek 7.7: Ilustrace výsledků segmentace jedné ze stránek (s identifikátorem `000110`) ze vzorku datové sady použitém k vyhodnocení, u kterých byly hodnoty metrik $O_{sum} = 85\,\%$ a F_{B^3} mezi 79,63 % a 89,68 % podle použitého elementu (nejnižší pro `chars`, nejvyšší pro `area`). Purpurové čáry značí dělení stránky na segmenty, sytost jejich výplně pak zanoření v hierarchii segmentačního stromu. Vlevo je výsledek implementované verze, napravo pak výsledek publikované verze.

Při bližší analýze výsledků srovnání lze ovšem pozorovat celkem 12 případů, kdy právě jedna ze srovnávaných segmentací selhala (tj. obsahuje pouze jediný, kořenový segment), zatímco druhá nikoliv (tj. obsahuje více segmentů). Z nich 9 případů je selhání publikované implementace, zatímco u zbylých 3 případů je tomu naopak. To naznačuje, že implementovaná metoda dokázala oproti publikované verzi nasegmentovat více stránek. Tomu odpovídá i průměrný počet segmentů na stránku. Implementovaná metoda vytvořila průměrně 15,9 segmentů pro každou stránku (9,38 po jejich napasování na DOM uzly), zatímco publikovaná implementace 12,43 (7,25 pro jejich napasování na DOM uzly).

Jestliže je aplikováno výše zmíněné oříznutí i na vzorové segmentace, lze je srovnat s výsledky obou implementací pro porovnání jejich kvality. Výsledky této analýzy lze pozorovat v grafech 7.8. Ty potvrzují nižší počet neúspěšných segmentací u implementované verze. Ovšem segmentace stránek, u kterých publikovaná verze selhává, nedosahují příliš dobrých výsledků – jak lze opět vidět v nárůstu nižších hodnot v histogramu 7.8. I přesto ovšem hodnoty v tabulky 7.3 vykazují mírné zlepšení průměrného hodnocení kvality segmentací oproti publikované verzi. Rozdíly jsou ale velmi malé.

Po bližší analýze zveřejněné implementace může být jedním z důvodů drobných odchylek např. odlišná implementace Sobelova operátoru. Zatímco zveřejněná implementace používá pro převod do černobílé podoby jednoduchým součtem hodnot všech barev, implementovaná metoda v rámci této práce používá dostupnou metodu knihovny OpenCV, která používá vlastní vzorec s různými váhami pro každý z barevných kanálů[4]. Rozdíl je demonstrován obrázkem 7.9.

[4]Dokumentace knihovny OpenCV: `Imgproc` color conversions – https://docs.opencv.org/4.6.0/de/d25/imgproc_color_conversions.html#color_convert_rgb_gray



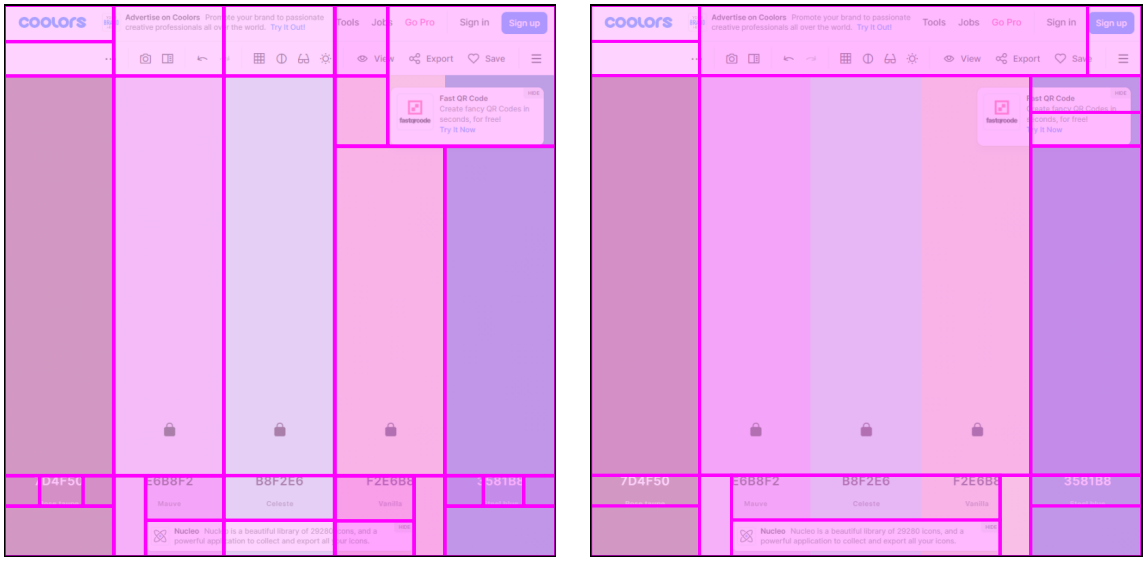
Obrázek 7.8: Histogramy s naměřenými hodnotami F_{B^3} pro element `nodes`, pro obě implementace. Na každém z histogramů jsou znázorněny hodnoty naměřené jak ze srovnání segmentací se vzorovu, jejíž prázdný prostor byl vyplněn, tak ze srovnání segmentací napasovaných na DOM uzly segmentované stránky se vzorovou segmentací bez úprav. Největší rozdíl mezi histogramy lze pozorovat v menším počtu stránek s nízkými hodnotami u implementované verze. Výsledné průměry se liší jen nepatrně, v obou případech o méně než 1 %.

		Implementovaná verze			Publikovaná verze		
Vyplněná vzorová segmentace	O_{sum}	40,22 %			37,72 %		
	Element	P_{B^3}	R_{B^3}	F_{B^3}	P_{B^3}	R_{B^3}	F_{B^3}
	`area`	35,38 %	67,28 %	**43,66 %**	33,88 %	61,32 %	**40,74 %**
	`edges_fine`	30,17 %	71,30 %	**39,00 %**	29,56 %	66,19 %	**37,29 %**
	`edges_coarse`	30,64 %	70,22 %	**39,16 %**	28,98 %	65,79 %	**36,62 %**
	`nodes`	28,48 %	72,60 %	**36,81 %**	28,86 %	66,98 %	**36,14 %**
	`chars`	32,41 %	77,21 %	**41,37 %**	32,03 %	69,79 %	**39,61 %**
Napasovaná segmentace	O_{sum}	30,35 %			28,36 %		
	Element	P_{B^3}	R_{B^3}	F_{B^3}	P_{B^3}	R_{B^3}	F_{B^3}
	`area`	19,90 %	71,67 %	**24,48 %**	19,84 %	65,17 %	**23,70 %**
	`edges_fine`	23,70 %	72,76 %	**29,64 %**	24,18 %	65,98 %	**29,14 %**
	`edges_coarse`	23,43 %	74,66 %	**29,79 %**	23,92 %	67,75 %	**29,46 %**
	`nodes`	22,46 %	77,63 %	**30,56 %**	23,22 %	70,99 %	**30,84 %**
	`chars`	25,99 %	77,75 %	**33,34 %**	25,64 %	70,58 %	**32,00 %**

Tabulka 7.3: Shrnutí průměrných výsledných hodnot metrik ze srovnání výsledků obou implementací nad daným vzorkem dat. Drtivá většina hodnot vychází lépe pro implementovanou verzi, s výjimkou *BCubed* metrik s elementem `nodes` po napasování segmentací na DOM uzly segmentovaných stránek, kde hodnota F_{B^3} je vyšší u publikované verze o 0,28 %. Naopak nejpříznivějšího rozdílu pro implementovanou verzi dosahují hodnoty *BCubed* metrik s elementem `area` po vyplnění vzorové segmentace, kde $\Delta F_{B^3} = 2,92\,\%$.

Obrázek 7.9: Srovnání převodu na černobílý obrázek před aplikací Sobelova operátoru. Vlevo je originální obrázek, uprostřed výsledek pomocí převodu knihovny OpenCV, napravo výsledek zveřejněné implementace. Na výsledku vpravo lze pozorovat menší rozdíl mezi odstíny, které měly v originále odlišné zbarvení.

Segmentace na obrázku 7.10 pak demonstrují, jak tato vlastnost může způsobit rozdíl v segmentaci stránky. Barvy prostředních tří pruhů, které jsou publikovanou verzí zahrnuty do společného segmentu, mají po převodu do černobílé verze publikovanou verzí identickou hodnotu. Takto uměle vytvořený příklad lze samozřejmě aplikovat i obráceně, kdy by byly barvy nastaveny takovým způsobem, aby jejich hodnota byla stejná v případě převodu do černobílé varianty implementovanou verzí metody. Jde ovšem o demonstraci, že tato odlišnost může způsobovat odlišnosti v segmentacích mezi těmito dvěma verzemi.



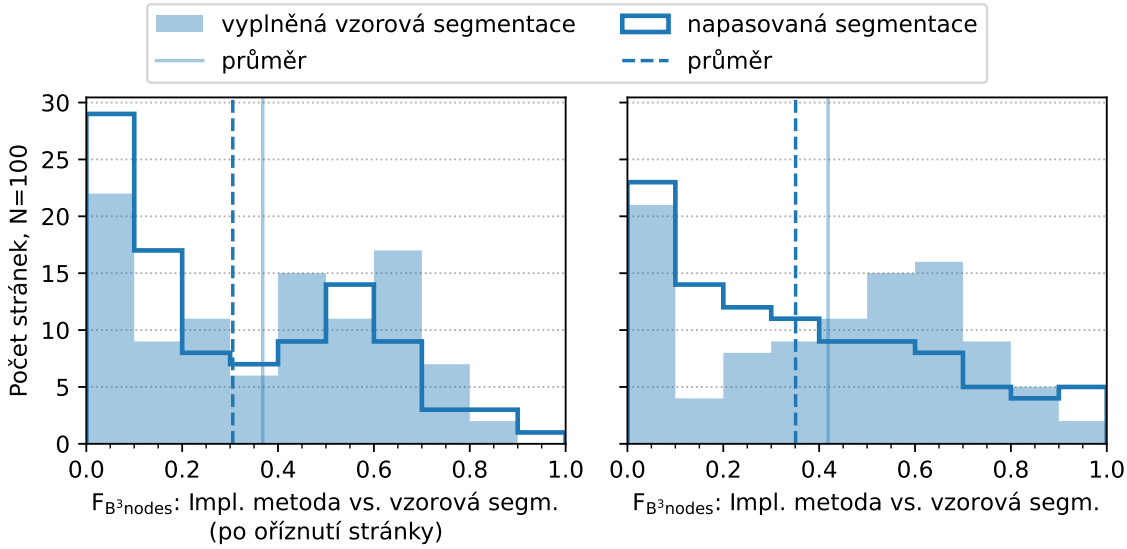
Obrázek 7.10: Vizualizace výsledků segmentace stránky `https://coolors.co/`. Vlevo je výsledek implementované verze, napravo pak výsledek publikované verze. Klíčovým rozdílem je rozdělení všech barevných pruhů na stránce do vlastních segmentů implementovanou verzí, zatímco publikovaná verze sloučila prostřední 3 pruhy do jednoho segmentu.

7.3 Kvalita segmentace

Po ověření funkčnosti implementované metody přichází tato sekce s analýzou kvality jejího procesu segmentace. Cílem této analýzy je zjistit, jak fungují vlastnosti metody na reálných stránkách, případy kdy funguje lépe či hůře a celkové objektivní hodnocení její kvality. Zároveň budou představeny provedené pokusy o její zlepšení experimentováním s různými kombinacemi hodnot nastavitelných parametrů, které metoda nabízí.

7.3.1 Úvodní měření kvality segmentace

Jako úvodní experiment byla implementovaná metoda spuštěna nad vzorkem datové sady ve výchozím nastavení parametrů (použitém v rámci srovnání v sekci 7.2, viz tabulka 7.1). Tímto způsobem získané výsledky se od výsledků ze sekce 7.2 liší tím, že vstupní obrázky stránek k segmentaci již nejsou oříznuty. Na ilustraci rozložení kvality segmentace 7.11 lze vidět, že oříznutí stránek metodě příliš nepomohlo a na jejích kompletních formách jsou výsledky obecně lepší. Značný je pokles stránek, u kterých segmentace selhala – na větší ploše stránky je větší šance, že metoda najde nějaké vodítko pro vhodnou segmentaci.



Obrázek 7.11: Histogramy s naměřenými hodnotami F_{B^3} pro element `nodes` pro neupravené a pro oříznuté vstupní stránky. Na každém z histogramů jsou znázorněny hodnoty naměřené jak ze srovnání segmentací se vzorovou, jejíž prázdný prostor byl vyplněn, tak ze srovnání segmentací napasovaných na DOM uzly segmentované stránky se vzorovou segmentací bez úprav. Po odstranění oříznutí ze stránek lze pozorovat pokles počtu stránek s velmi špatnou segmentací ($F_{B^3\texttt{nodes}} < 10\,\%$) a celkově předvídatelnější výsledky tvořící normální rozložení se středem v $F_{B^3\texttt{nodes}} \sim 60\,\%$. Průměr se také zlepšil, o $F_{B^3\texttt{nodes}} \sim 5\,\%$ – díky neúspěšným segmentacím pořád ovšem znatelně níž než střed rozložení.

Shrnutí výsledků v tabulce 7.4 pak dále vykazuje zlepšení kvality segmentace odstraněním oříznutí. Jedinou metrikou, která zlepšení nevykazuje, je R_{B^3}. To indikuje, že po odstranění oříznutí je vyšší výskyt segmentů, které jsou menší než vzorové segmenty. Důvodem může být to, že okolní prostor je členěn metodou do menších segmentů (protože nedokáže vytvořit jednotný segment pro celý prázdný prostor). Tento efekt je pak zna-

telnější u kompletních stránek, kde je více prázdného prostoru, který tak metodou může být dále rozdělen, zatímco ve vzorové segmentaci zůstává stále v jediném segmentu. To by vysvětlovalo také důvod, proč u segmentace napasované na DOM uzly stránky není rozdíl tak výrazný.

		Po oříznutí stránky			Bez úpravy stránky		
Vyplněná vzorová segmentace	O_{sum}	40,22 %			42,57 %		
	Element	P_{B^3}	R_{B^3}	F_{B^3}	P_{B^3}	R_{B^3}	F_{B^3}
	`area`	35,38 %	67,28 %	**43,66 %**	39,47 %	66,62 %	**46,54 %**
	`edges_fine`	30,17 %	71,30 %	**39,00 %**	35,21 %	68,68 %	**42,14 %**
	`edges_coarse`	30,64 %	70,22 %	**39,16 %**	36,20 %	67,10 %	**42,30 %**
	`nodes`	28,48 %	72,60 %	**36,81 %**	34,90 %	70,75 %	**41,87 %**
	`chars`	32,41 %	77,21 %	**41,37 %**	39,77 %	72,32 %	**46,25 %**
Napasovaná segmentace	O_{sum}	30,35 %			30,85 %		
	Element	P_{B^3}	R_{B^3}	F_{B^3}	P_{B^3}	R_{B^3}	F_{B^3}
	`area`	19,90 %	71,67 %	**24,48 %**	20,83 %	76,23 %	**26,46 %**
	`edges_fine`	23,70 %	72,76 %	**29,64 %**	28,96 %	76,87 %	**36,62 %**
	`edges_coarse`	23,43 %	74,66 %	**29,79 %**	28,97 %	76,76 %	**36,58 %**
	`nodes`	22,46 %	77,63 %	**30,56 %**	26,81 %	77,62 %	**35,09 %**
	`chars`	25,99 %	77,75 %	**33,34 %**	34,05 %	76,77 %	**41,91 %**

Tabulka 7.4: Shrnutí průměrných výsledných hodnot metrik ze srovnání výsledků pro oříznuté a pro neupravené vstupní stránky. Odstranění oříznutí ze segmentované stránky zlepšilo všechny výsledky s výjimkou metriky R_{B^3}. Průměrný nárůst hodnot F_{B^3} je 4,8 %, největší nárůst, 8,57 %, pak zaznamenala hodnota metriky $F_{B^3\texttt{chars}}$ pro segmentace napasované na DOM uzly stránky.

Izolovaně od oříznutých stránek lze pozorovat rozdíly mezi srovnáním po vyplnění vzorové segmentace oproti srovnání segmentace po jejím napasování na DOM uzly stránky, kde výsledky po vyplnění vzorové segmentace vypadají obecně lépe. Největší rozdíl je u hodnot metriky $F_{B^3\texttt{area}}$, kde je rozdíl 20,08 %. S elementy, které pak obvykle nezahrnují okolní prostor, jako `edges_fine` či `nodes` pak rozdíl klesá, a to až na minimum 4,34 % pro element `chars`. Z tohoto trendu, který bude pozorovatelný napříč výsledky i v následujících sekcích, lze odvodit, že metoda zvládá lépe segmentaci okolního prostoru, který je při vyplnění vzorové segmentace zahrnutý ve srovnávání segmentů a zlepšuje tak celkové výsledky.

Obecně lze ale říci, že výsledky nevypadají příliš pozitivně. Průměrná hodnota metriky F_{B^3} je 43,82 % po vyplnění vzorové segmentace, či dokonce 35,33 % po napasování segmentace na DOM uzly stránky. Tyto výsledky neodpovídají prezentovaným hodnotám naměřeným v rámci publikace [18], kde pro obdobné nastavení parametrů bylo dosaženo hodnot stejné metriky v průměru okolo 57 % pro segmentace napasované na DOM uzly stránky, což je rozdíl téměř 22 %.

Jak bylo zmíněno v sekci 7.1.2, data z publikace [18] ovšem nejsou tak jednoduše srovnatelná. Například pro stránku z datové sady s identifikátorem `000690` metoda selhala jak v případě implementované verze, tak v případě té publikované. Ale zatímco v rámci tohoto vyhodnocení byla takovému případu přidělena hodnota 0 % kvůli absenci segmentů (jelikož kořenový segment postrádající jakoukoliv informaci je odstraněn), v publikaci [18] dosáhla tato segmentace průměrné hodnoty F_{B^3} 68,5 %. A to díky tomu, že vzorová segmentace pro tuto stránku obsahuje velký segment, tudíž P_{B^3} zůstává na poměrně vysokých hodnotách, zatímco metrika R_{B^3} dosahuje očekávaných 100 %.

Důvodem rozdílů je tedy zejména nedokonalost použitých metrik, které v určitých případech (velkých segmentech ve vzorové segmentaci) nedokážou dostatečně penalizovat jediný segment přes celou plochu stránky. Vzhledem k ošetření v rámci této publikace jsou pak naměřené výsledky výrazně horší, než v případě, kdy ošetření aplikováno není. Důsledkem jsou ovšem výsledky, které lépe odráží kvalitu segmentací s ohledem na neúspěšné segmentace.

7.3.2 Analýza chování metody

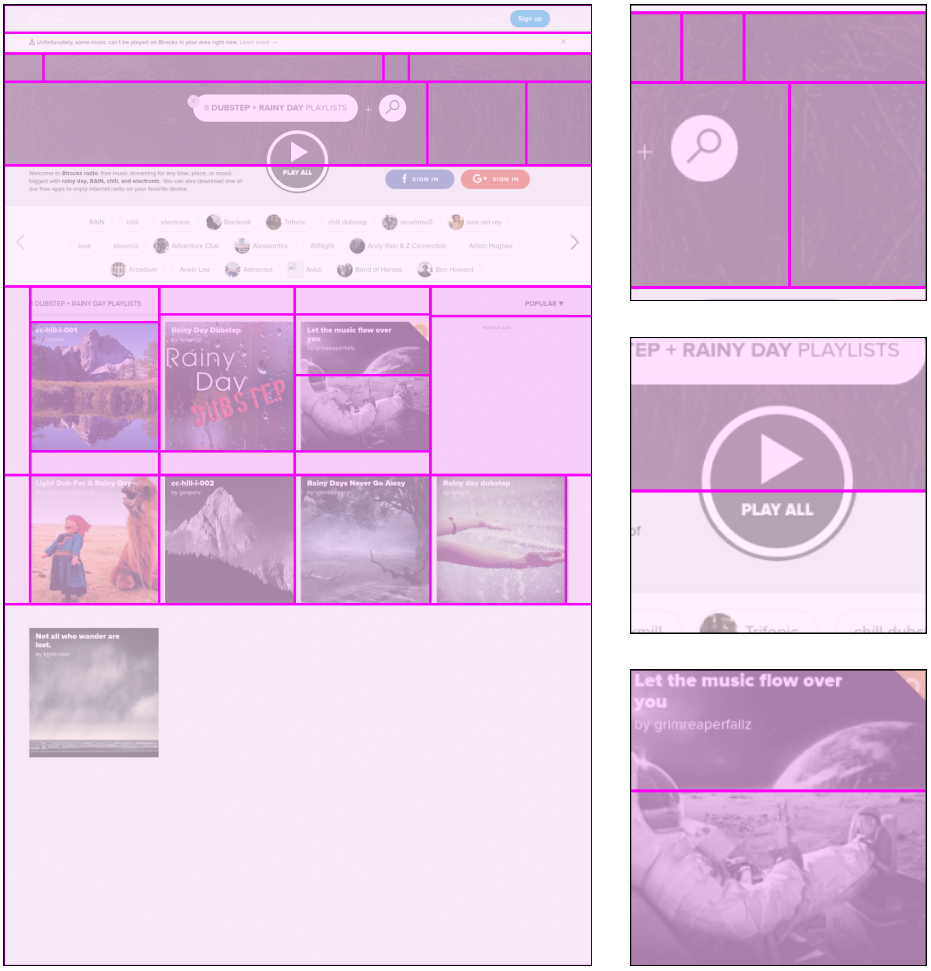
Před experimentováním nad různými parametry rozebírá tato podsekce výsledky segmentace získané úvodním experimentem. Cílem je zjistit její vlastnosti a nedostatky, které se mohou následující experimenty pomocí úpravou parametrů pokusit zmírnit či úplně odstranit.

První, na první pohled zřetelnou nedokonalostí metody, je její tendence vytvářet segmentace na místech, kde nejsou žádoucí. Několik takových případů lze pozorovat na obrázku 7.12. Může jít o nepatrné textury v obrázku na pozadí, které mohou díky použití relativní síly hran oproti okolí stačit pro formování hranic segmentací, přestože jde o souvislou část stránky. Nebo mohou být důvodem objekty v obrázcích, které tvoří výraznou linku. Je zřejmé, že v těchto případech metodě chybí informace o struktuře stránky a schopnost segmentace obrázků nemusí být vždy žádoucí.

Co z vlastností metody funguje na příkladu v obrázku 7.12 dobře, je rozdělení okolí pro výpočet lokální významnosti na poloviny za účelem přidělení priority při oddělení dvou ploch s výrazným rozdílem síly hran. Metoda díky tomu v tomto případě dokázala velmi dobře oddělit obrázky jak jednotlivých hudebních položek, tak i hlavičky od světlého okolí.

Obrázek 7.12 ovšem ukazuje i další znak stránek, který metoda žádným způsobem neošetřuje a který výrazně zhoršuje schopnost metody vytvářet segmenty. Jde o kulaté tvary, jejichž okraje mají kvůli vlastnostem použitého přístupu pro získání síly hran nižší lokální významnost. Ovšem i kdyby byl tento problém ošetřen, tak metoda již z principu, kterým tvoří segmentace (hledáním horizontálních či vertikálních dělících hranic), v podstatě nikdy kulaté tvary k segmentaci nevyužije. Ještě více metodou ignorované rysy jsou pak rovné linky, které jsou vychýlené z horizontální či vertikální osy. Oproti kulatým tvarům, kde metoda zachytí alespoň malou část na okrajích v extrémech každé souřadnice, zde nevyužije nic. Ovšem také oproti kulatým tvarům, výskyt takových linií není ve webových stránkách úplně běžný.

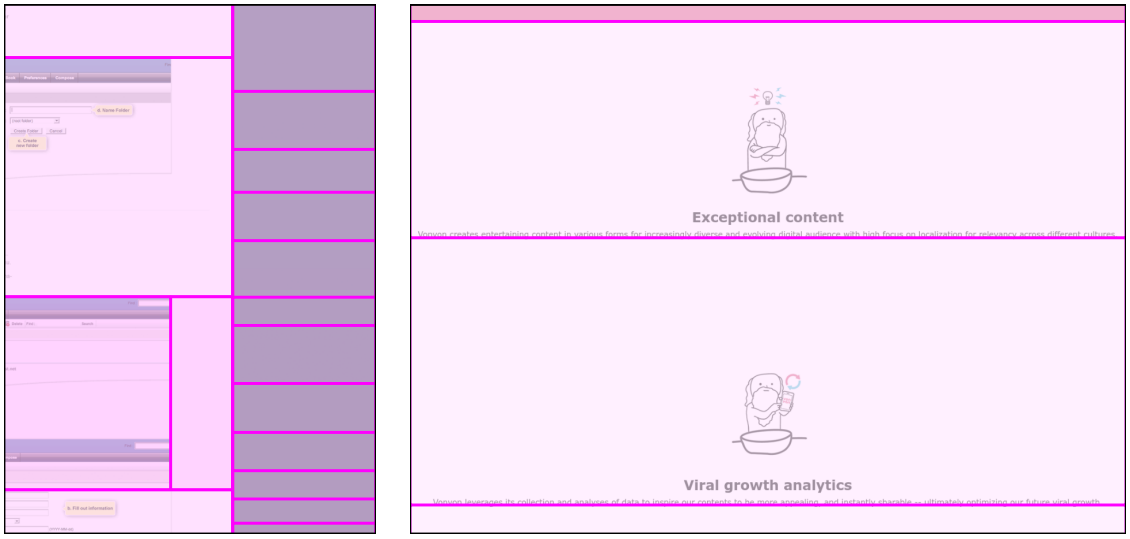
Poslední zajímavá vlastnost pozorovatelná z obrázku 7.12 je také jedním z nejčastějších důvodů neúspěšných segmentací. Jak lze na obrázku vidět, všechny hudební položky jsou segmentovanou metodou roztříděny do vlastních segmentů – až na poslední, pro kterou je přidělena celá spodní část stránky. Důvodem je způsob, kterým metoda vytváří dělící linie jednotlivých segmentů. Ty jsou vždy tvořeny tak, že dělí daný segment po celé jeho délce. To znamená, že pokud po celé délce není dostatečná sémantická významnost, tak metoda takovou linii pro rozdělení segmentu nepoužije. Takové případy pak nastávají poměrně často v situacích, jako je právě na obrázku – objekt tvořící jasnou dělící linii, která je vzhledem k velikosti segmentu, ve kterém se nachází, příliš krátká a není tak použita pro jeho rozdělení.



Obrázek 7.12: Ukázka segmentace z úvodního experimentu na stránce z datové sady s identifikátorem `000070`. Výřezy obrázku vpravo se zaměřují na části, na kterých lze pozorovat dané vlastnosti metody – shora dolů: nechtěné segmentace nevýrazných hran; neschopnost segmentovat komplexní tvary; nechtěné segmentace způsobené rovnostmi v obrázcích.

Poměrně zajímavě reaguje metoda na postupné horizontální či vertikální přechody mezi barvami. Díky způsobu výpočtu lokální významnosti jsou jemným přechodům v barvách přiděleny vysoké hodnoty, které pak způsobují tvorbu segmentů napříč takovými přechody. Tuto vlastnost lze dobře vidět na okraji levé vizualizace segmentace v obrázku 7.13.

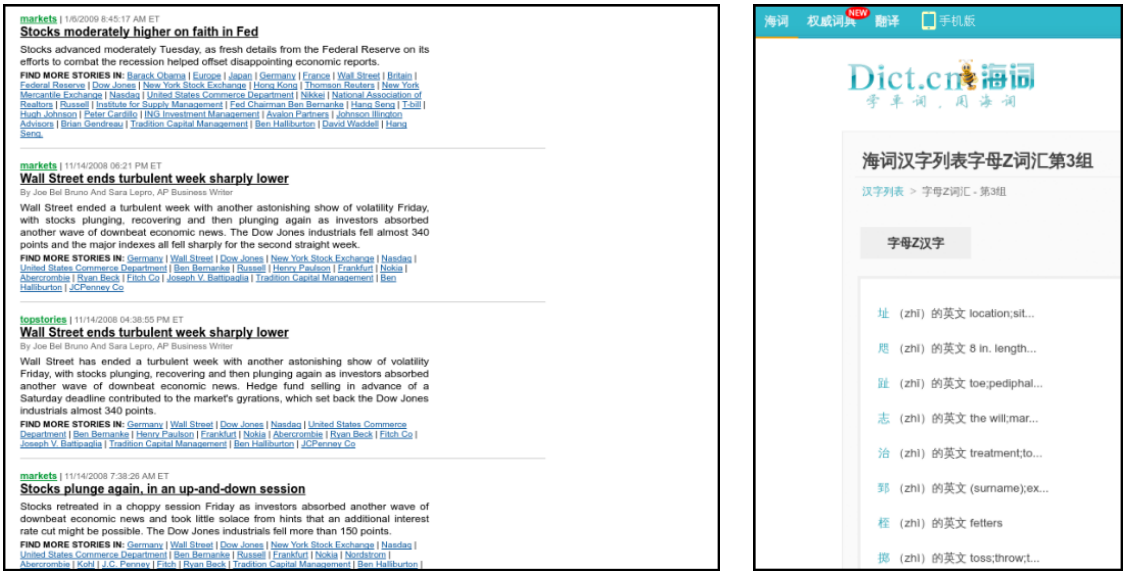
Metoda má také ke tvorbě segmentů tendenci využívat delších textových řetězců, které tvoří souvislou linii napříč segmentem – zejména pokud v okolí takového řetězce není příliš mnoho dalších objektů, které by lokální významnost textu snížily (jako např. v pravém obrázku 7.13, kde je okolí textu prázdné). Vytvořené segmenty tímto způsobem mohou dopadnout poměrně úspěšně, přestože např. běžný uživatel by segment vytvořil spíše takovým způsobem, aby byl textový obsah více uprostřed daného segmentu. Proti vytvoření segmentace tímto způsobem uprostřed textu přes více řádků pak brání lokální významnost, alespoň dokud prostor mezi řádky takového textu není větší než definované okolí.



Obrázek 7.13: Výřezy vizualizací segmentací stránek z úvodního experimentu z datové sady s identifikátory `000860` (vlevo) a `000980` (vpravo). Nalevo lze pozorovat segmentaci přechodů mezi barvami na okraji stránky, napravo pak využití textu k vytvoření dělící linie.

Podstatná je rovněž analýza stránek, u kterých byla metoda neúspěšná. V levém obrázku 7.14 je výřez jedné z nich. V tomto případě jde zejména o malý výskyt vizuálních prvků a ty, které se na ní vyskytují, metoda nezvládá využít. Příkladem je dělící linie mezi jednotlivými články, její lokální významnost je ovšem snížená okolním textem a její délka z obou stran obklopena velkým prázdným prostorem. Tím druhým zmíněným problémem trpí také linie tvořené zarovnáním textu článků, díky zápatí stránky, které má rovněž bílé pozadí a poměrně velkou výšku.

V pravém obrázku 7.14 je pak výřez další stránky se stejně neúspěšným výsledkem segmentace. Její obsah je pro metodu poměrně příznivě vizuálně označen. Problémem je ovšem opět izolace tohoto obsahu prázdným okolím. Navíc stránka odhaluje další, zcela odlišný problém nastavení parametrů – přestože by měla být metoda schopná vytvořit segment pro záhlaví stránky, které je jasně vizuálně oddělené po celou šířku stránky, nedokáže tak učinit, protože má záhlaví menší výšku, než je parametrem nastavená minimální šířka segmentů.



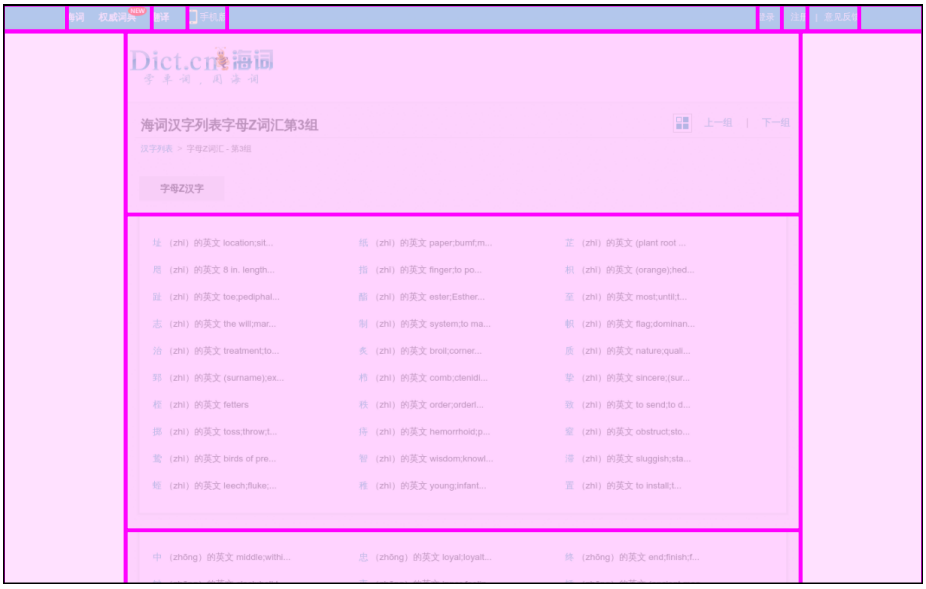
Obrázek 7.14: Výřezy obrázků stránek z datové sady, které se v rámci úvodního experimentu metodě nepodařilo nasegmentovat. Obrázek vlevo obsahuje stránku s identifikátorem `000690`, vpravo `000800`.

7.3.3 Experimentování s parametry metody

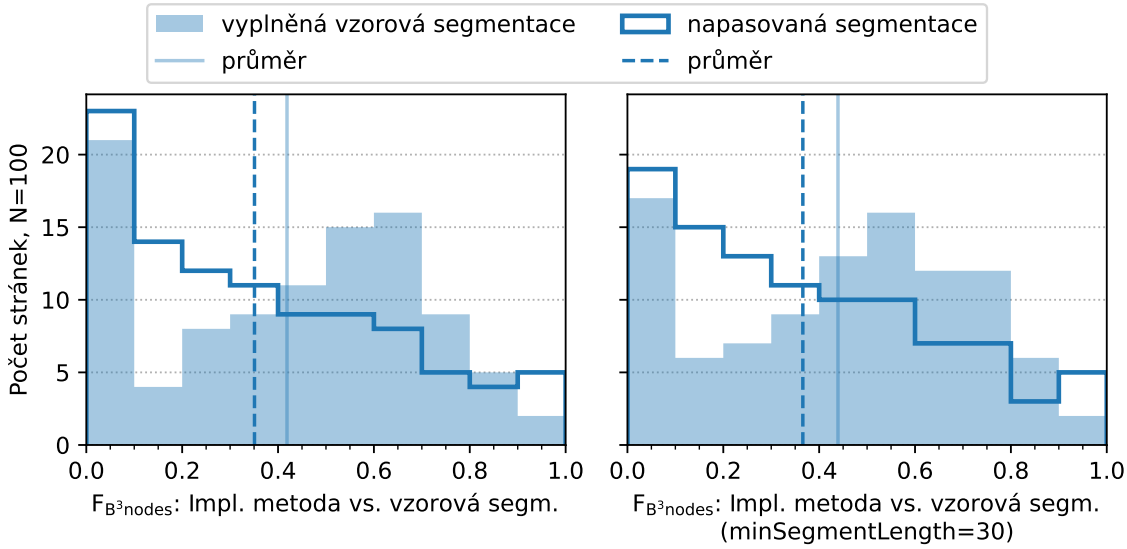
První experiment je zaměřen na na poslední zmíněný případ v předchozí sekci 7.3.2. Protože má hlavička stránky výšku 40 bodů a hodnota parametru `minSegmentLength` byla v úvodním experimentu nastavena na 45, metoda pro hlavičku nedokázala vytvořit vlastní segment. Pro ošetření podobných problémů a je tedy v rámci tohoto experimentu hodnota parametru `minSegmentLength` snížena na 30 bodů.

Jak lze vidět na vizualizaci segmentace 7.15, pro hlavičku stránky byl úspěšně vytvořen vlastní segment. Metoda navíc díky zmenšení vzdálenosti hlavního obsahu od horní hranice segmentu dokázala použít vertikální hranice tohoto obsahu k oddělení prázdného prostoru po stranách, což následně vedlo k úspěšné segmentaci zbytku obsahu uprostřed stránky.

Z výsledků experimentu nad celým vzorkem dat pak lze pozorovat rovněž příznivé výsledky. Ve srovnání s úvodním experimentem lze v histogramech 7.16 pozorovat úbytek stránek, pro které se metodě nepodařilo vytvořit segmentaci. Konkrétně počet takových stránek klesl z 20 na 16. Díky tomu a celkově díky volnějším podmínkám ke tvorbě segmentů také stoupl průměrný počet segmentů na stránku z 20,9 (13,1 po napasování) na 30 (16,8 po napasování). Zároveň lze na rozložení vidět, že i mimo nápravy neúspěšných segmentací změna parametru nijak výrazně neublížila. Celkové výsledky, které jsou vidět v tabulce 7.5 jsou pak celkově pozitivní.



Obrázek 7.15: Výřez vizualizace segmentace stránky z datové sady s identifikátorem 000800 po změně parametru `minSegmentLength` na hodnotu 30, díky které nyní metoda pro tuto stránku dokázala vytvořit segmentaci.



Obrázek 7.16: Histogramy s výslednými hodnotami metriky F_{B^3} pro element `nodes` pro experiment s nastavením parametru `minSegmentLength` na hodnotu 30 (vpravo) a pro úvodní experiment (vlevo). Z rozložení výsledků v histogramech lze pozorovat příznivý dopad úpravy hodnoty metriky. Nejvýraznější rozdíl představuje pokles počtu stránek s velmi špatnou segmentací ($F_{B^3\texttt{nodes}} < 10\,\%$).

	Vyplněná vzorová segm.			Napasovaná segm.		
O_{sum}	44,24 %			31,33 %		
Element	P_{B^3}	R_{B^3}	F_{B^3}	P_{B^3}	R_{B^3}	F_{B^3}
`area`	40,84 %	70,54 %	**48,50 %**	21,17 %	80,21 %	**27,05 %**
`edges_fine`	36,68 %	72,17 %	**44,11 %**	30,05 %	80,99 %	**38,14 %**
`edges_coarse`	37,74 %	70,44 %	**44,22 %**	29,99 %	80,86 %	**38,07 %**
`nodes`	36,53 %	74,46 %	**43,93 %**	27,85 %	81,63 %	**36,59 %**
`chars`	41,17 %	76,28 %	**48,23 %**	34,96 %	80,97 %	**43,25 %**

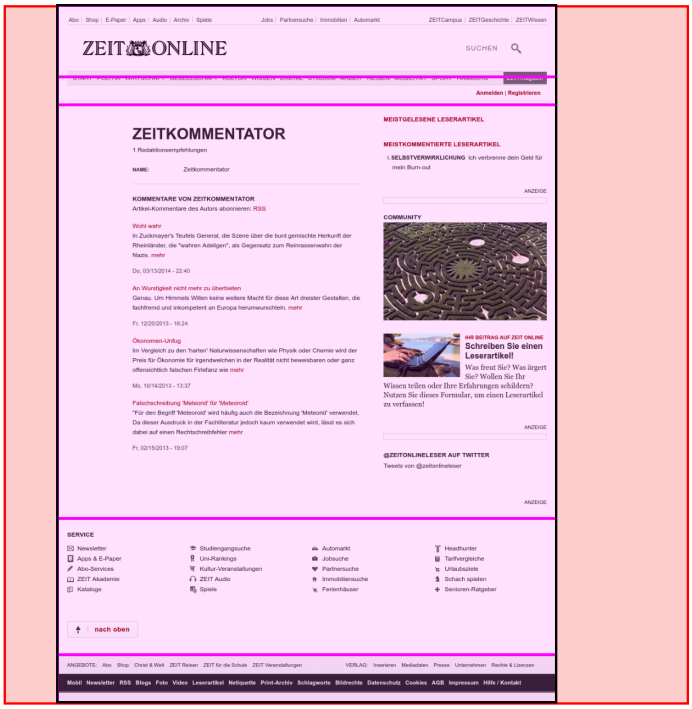
Tabulka 7.5: Přehled průměrných výsledných hodnot metrik po srovnání výsledků segmentace s hodnotou parametru `minSegmentLength` = 30 oproti vzorku datové sady. Oproti obdobným výsledkům úvodního experimentu lze pozorovat konzistentní mírné zlepšení napříč všemi metrikami. Průměrné zlepšení metrik F_{B^3} činí 1,63 %.

Další experiment se bude zaměřovat na častější problém metody, rovněž zmíněný v předchozí sekci – nevyužití vizuálních linií k segmentaci, pokud nepokrývají dostatečnou část segmentu. Jeden z mnoha příkladu selhání segmentace z tohoto důvodu je vizualizován na obrázku 7.17. Stránka na obrázku obsahuje vizuální prvky, které může metoda využít pro její horizontální rozdělení – ty jsou ale obklopeny prázdným prostorem po bocích stránky. Původní obrázek stránky pak v rámci úvodního experimentu metoda nedokázala nasegmentovat.

Obrázek 7.17 demonstruje možnost řešení tohoto problému oříznutím prázdných okrajů, díky kterému metoda již zvládne horizontální linie k segmentaci využít. Toto řešení ovšem nelze použít vždy. Další možností, jak k řešení přistoupit, je modifikací hodnoty parametru `maxLineLength`.

Navýšením hodnoty parametru `maxLineLength` výrazně stoupá sémantická významnost delších dělících linií. Pro následující experiment byla jeho hodnota navýšena na čtyřnásobek oproti úvodnímu experimentu, tedy na hodnotu 1 024, při které je již vytvořena segmentace i pro originální stránku z obrázku 7.17. Tato změna téměř eliminovala případy, kdy metoda nedokázala vytvořit segmentaci – z 20 takových případů zůstal pouze jediný. Tento efekt lze pozorovat i v obrázku 7.18, kde došlo k výraznému poklesu segmentací s nejnižším ohodnocením.

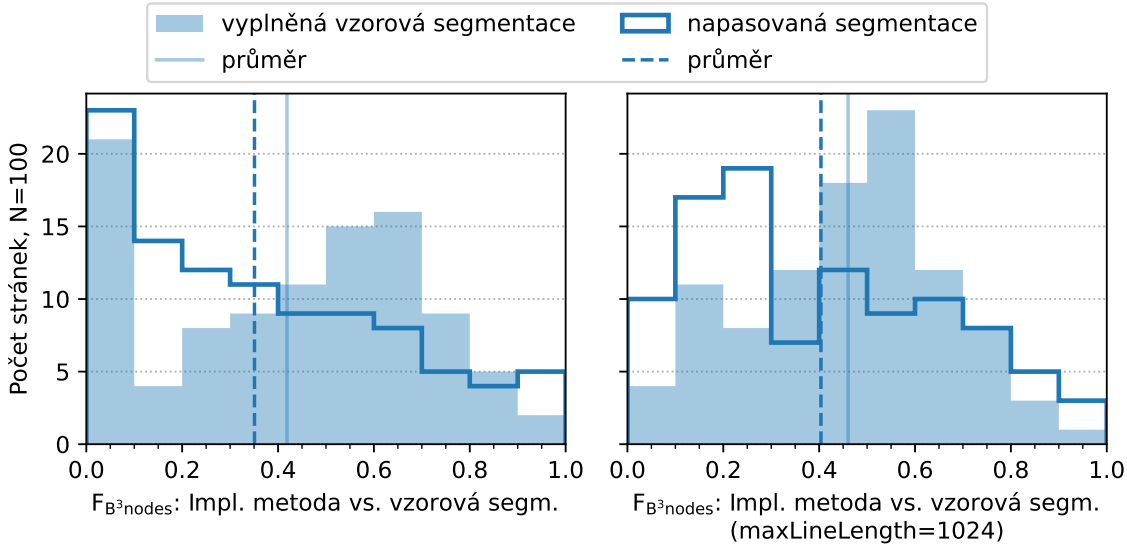
Taková radikální změna hodnoty parametru se ovšem neobchází bez vedlejšího efektu, který lze vyvodit již z průměrného počtu segmentů – ten stoupl ze 20,9 (13,1 po napasování) na 91,7 (52,6 po napasování), tj. více než čtyřnásobek. Důsledek lze pozorovat ve shrnutí výsledků 7.6, které pro lepší ilustraci tentokrát nezahrnuje neúspěšné segmentace. Je z něj tak patrné, že sice nyní metoda zvládne vytvořit segmentace pro většinu stránek, ale za cenu jejich průměrné kvality.



Obrázek 7.17: Vizualizace segmentace stránky z datové sady s identifikátorem `000670` po odstranění prázdných okrajů stránky (označených červenou barvou), díky čemuž metoda dokázala použít horizontální linie pro vytvoření segmentace.

		Výchozí parametry			`maxLineLength` = 1024		
Vyplněná vzorová segmentace	O_{sum}	53,22 %			38,89 %		
	Element	P_{B^3}	R_{B^3}	F_{B^3}	P_{B^3}	R_{B^3}	F_{B^3}
	`area`	49,34 %	83,28 %	**58,18 %**	46,16 %	68,55 %	**49,97 %**
	`edges_fine`	44,01 %	85,86 %	**52,67 %**	42,46 %	76,69 %	**48,23 %**
	`edges_coarse`	45,26 %	83,87 %	**52,88 %**	42,13 %	77,48 %	**48,24 %**
	`nodes`	43,62 %	88,43 %	**52,34 %**	42,22 %	74,79 %	**46,50 %**
	`chars`	49,97 %	90,86 %	**58,11 %**	46,57 %	82,40 %	**52,19 %**
Napasovaná segmentace	O_{sum}	39,55 %			33,81 %		
	Element	P_{B^3}	R_{B^3}	F_{B^3}	P_{B^3}	R_{B^3}	F_{B^3}
	`area`	26,04 %	95,29 %	**33,07 %**	30,30 %	87,78 %	**35,06 %**
	`edges_fine`	36,20 %	96,09 %	**45,78 %**	37,47 %	90,05 %	**44,67 %**
	`edges_coarse`	36,21 %	95,95 %	**45,72 %**	37,51 %	89,93 %	**45,14 %**
	`nodes`	33,51 %	97,03 %	**43,86 %**	32,44 %	91,59 %	**40,78 %**
	`chars`	42,79 %	96,46 %	**52,65 %**	38,16 %	91,62 %	**46,53 %**

Tabulka 7.6: Souhrn průměrných výsledků segmentace z úvodního experimentu se segmentací s parametrem `maxLineLength=1024` oproti vzorku datové sady, po odebrání výsledků neúspěšných segmentací. Mezi průměrnými hodnotami metriky F_{B^3} lze pozorovat po nastavení parametru `maxLineLength` průměrný pokles o 3,8 %. Nejvýraznější zhoršení zaznamenala metrika $F_{B^3\texttt{area}}$ po vyplnění vzorové segmentace s rozdílem 8,21 %. Výjimku pak tvoří stejná metrika po napasování segmentace na DOM uzly stránky, v takovém případě je její výsledek o 1,98 % lepší.



Obrázek 7.18: Vizualizace rozložení výsledných hodnot metriky F_{B^3} pro element `nodes` pro experiment s hodnotou parametru `maxLineLength` = 1 024 (vpravo) a pro úvodní experiment (vlevo) pomocí histogramů. Na první pohled znatelnou změnou je zlepšení segmentací pro stránky, které v úvodním experimentu měly velmi špatnou segmentaci ($F_{B^3\texttt{nodes}} < 10\,\%$). Zároveň ale stojí za zmínku úbytek kvalitních segmentací ($F_{B^3\texttt{nodes}} > 70\,\%$) oproti vyplněné vzorové segmentaci.

7.3.4 Hledání optimálních hodnot parametrů

Obdobným způsobem, kterým byly provedeny experimenty v předcházející sekci, byly provedeny další, nad dalšími možnými hodnotami a kombinacemi parametrů. Spolu s úvodním experimentem tak proběhlo vyhodnocení nad 40 různými kombinacemi hodnot parametrů. Shrnutí výsledků všech těchto experimentů je prezentováno tabulkou 7.7.

Úvodní experiment skončil v tabulce na 35. pozici. Jeho výsledky nejsou příliš příznivé, ovšem oproti experimentům s lepšími výsledky netrpí přílišným množstvím segmentů. Blíže rozebrané experimenty ze sekce 7.3.3 se pak v tabulce nacházejí na 29. a 19. pozici.

Spolu s úvodním experimentem mají experimenty na 9., 24. a 40. pozici stejné nastavení parametrů, které vyhodnocovala publikace [18]. Konkrétní výsledky se sice liší (z důvodů zmíněných v sekci 7.3.1), relativní pořadí mezi jednotlivými nastaveními parametrů ovšem nikoliv.

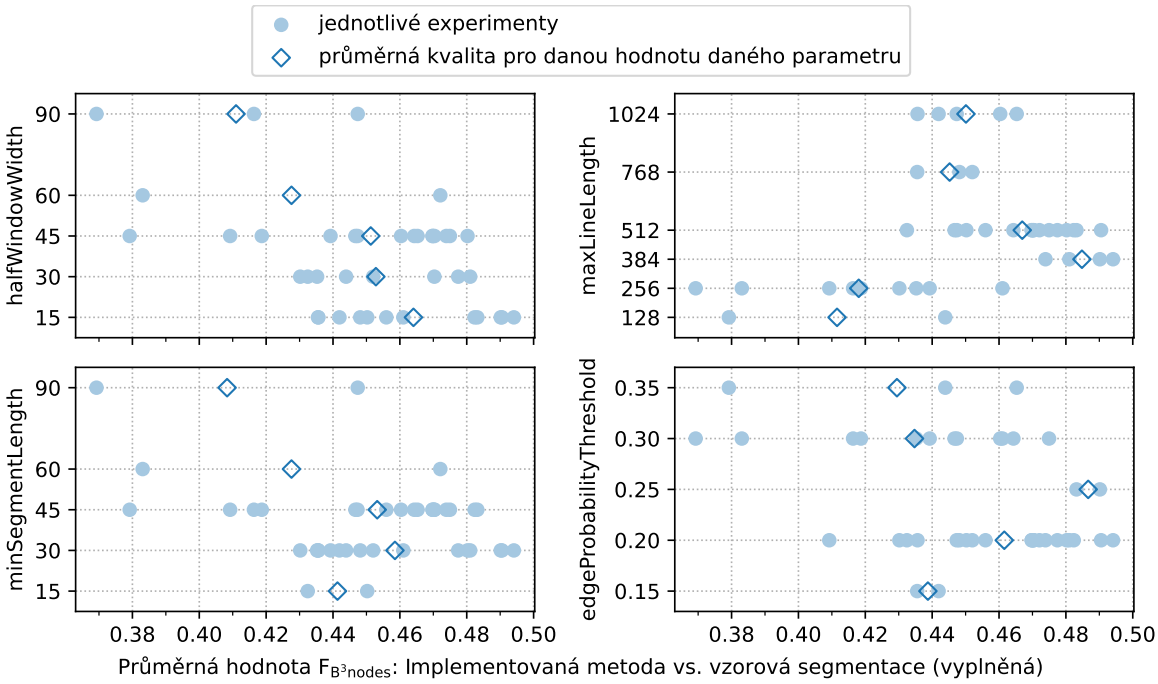
Na první pozici je pak nastavení parametrů, jehož výsledná hodnota metriky $F_{B^3\texttt{nodes}}$ pro vyplněnou vzorovou segmentaci je oproti úvodnímu experimentu vyšší o 7,54 %. Zároveň je ovšem její průměrný počet segmentů více než 15x vyšší. Na 3. pozici jsou výsledky experimentu, který se snaží tento nedostatek experimentu na 1. pozici zmírnit navýšením parametru `EdgeProbabilityThreshold` a za cenu mírného zhoršení výsledků snižuje poměr průměrného počtu segmentů oproti úvodnímu experimentu na 8,8 : 1. První z experimentů s lepšími výsledky, který tento jev redukuje na rozumnou hodnotu (pouhý trojnásobek průměrného počtu segmentů), je pak na 9. pozici.

	halfWindowWidth	standard-Deviation	priorEdge-Probability	pyramidLevels	maxLineLength	edgeProbability-Threshold	monteCarloTrials	minSegmentLength	signLineProb-Threshold	Vyplněná vzorová segmentace		Napasovaná segmentace		Neúspěšných
										Počet segm.	$F_{B^3{}_{nodes}}$	Počet segm.	$F_{B^3{}_{nodes}}$	
01	15	0,10	0,010	1	384	0,20	100	30	0,5	317,1	49,41 %	107,1	44,98 %	2
02	15	0,10	0,010	1	512	0,20	100	30	0,5	367,8	49,05 %	117,5	43,94 %	2
03	15	0,10	0,010	1	384	0,25	100	30	0,5	184,0	49,02 %	83,3	43,49 %	4
04	15	0,10	0,010	1	512	0,25	100	45	0,5	137,4	48,32 %	73,3	40,89 %	6
05	15	0,10	0,010	1	512	0,20	100	45	0,5	212,8	48,24 %	91,8	42,69 %	4
06	30	0,10	0,010	1	384	0,20	100	30	0,5	281,0	48,10 %	100,0	41,53 %	2
07	45	0,10	0,010	1	512	0,20	100	30	0,5	281,2	48,02 %	102,1	41,58 %	2
08	30	0,10	0,010	1	512	0,20	100	30	0,5	324,1	47,75 %	110,0	41,10 %	2
09	45	0,10	0,010	1	512	0,30	100	45	0,5	60,2	47,50 %	36,0	38,57 %	6
10	45	0,10	0,010	1	384	0,20	100	45	0,5	148,8	47,39 %	71,9	40,28 %	4
11	60	0,10	0,010	1	512	0,20	100	60	0,5	111,6	47,21 %	61,2	39,28 %	4
12	30	0,10	0,010	1	512	0,20	100	45	0,5	185,9	47,03 %	86,3	40,18 %	4
13	45	0,20	0,010	1	512	0,20	100	45	0,5	173,3	47,03 %	81,5	40,36 %	4
14	45	0,10	0,010	1	512	0,20	100	45	0,5	174,8	46,98 %	81,3	40,15 %	4
15	45	0,30	0,010	1	512	0,20	100	45	0,5	174,9	46,98 %	81,5	39,84 %	4
16	45	0,10	0,010	1	1024	0,35	100	45	0,5	53,6	46,53 %	33,0	41,11 %	1
17	45	0,05	0,010	1	512	0,30	100	45	0,5	59,9	46,43 %	35,9	38,15 %	6
18	15	0,10	0,010	1	256	0,30	100	30	0,5	74,2	46,10 %	37,4	37,00 %	9
19	45	0,10	0,010	1	1024	0,30	100	45	0,5	91,7	46,04 %	52,6	40,37 %	1
20	15	0,10	0,010	2	512	0,20	100	45	0,5	305,2	45,59 %	120,4	39,34 %	2
21	30	0,10	0,010	1	768	0,20	100	30	0,5	406,2	45,20 %	121,6	42,40 %	0
22	15	0,10	0,010	1	512	0,20	100	15	0,5	912,5	45,02 %	140,7	40,00 %	2
23	15	0,10	0,010	1	768	0,20	100	30	0,5	467,0	44,81 %	128,5	42,89 %	1
24	90	0,10	0,010	1	512	0,30	100	90	0,5	22,8	44,74 %	16,6	37,18 %	11
25	45	0,10	0,010	1	1024	0,20	100	45	0,5	227,7	44,73 %	94,2	39,49 %	0
26	45	0,10	0,005	1	512	0,30	100	45	0,5	33,6	44,67 %	21,3	38,11 %	10
27	30	0,10	0,010	1	128	0,35	100	30	0,5	11,6	44,39 %	6,1	37,65 %	24
28	15	0,10	0,010	1	1024	0,15	100	30	0,5	889,2	44,19 %	152,7	41,46 %	0
29	45	0,10	0,010	1	256	0,30	100	30	0,5	30,0	43,93 %	16,8	36,59 %	16
30	15	0,10	0,010	1	1024	0,20	100	30	0,5	486,1	43,56 %	130,7	40,03 %	1
31	15	0,10	0,010	1	768	0,15	100	30	0,5	869,0	43,55 %	151,5	41,44 %	0
32	30	0,10	0,010	1	256	0,30	100	30	0,5	45,9	43,52 %	19,5	34,85 %	13
33	30	0,10	0,010	1	512	0,20	100	15	0,5	718,3	43,24 %	126,3	38,06 %	2
34	30	0,10	0,010	1	256	0,20	100	30	0,5	167,3	43,02 %	64,1	34,82 %	13
35	45	0,10	0,010	1	256	0,30	100	45	0,5	20,9	41,87 %	13,1	35,09 %	20
36	90	0,10	0,010	1	256	0,30	100	45	0,5	13,7	41,63 %	7,8	34,23 %	21
37	45	0,10	0,010	1	256	0,20	100	45	0,5	85,1	40,92 %	44,1	32,60 %	17
38	60	0,10	0,010	1	256	0,30	100	60	0,5	15,1	38,31 %	9,6	32,12 %	25
39	45	0,10	0,010	1	128	0,35	100	45	0,5	6,1	37,92 %	3,9	31,74 %	33
40	90	0,10	0,010	1	256	0,30	100	90	0,5	7,1	36,93 %	4,5	30,49 %	31

Tabulka 7.7: Přehled výsledků všech provedených experimentů s různými kombinacemi parametrů implementované metody nad identickým vzorkem dat o 100 stránkách. Jednotlivé výsledky jsou seřazeny (a očíslovány) podle kvality na základě hodnoty metriky $F_{B^3\texttt{nodes}}$ pro vyplněnou vzorovou segmentaci. Sytost barevného zbarvení hodnot parametrů je závislá na odchýlení od výchozí hodnoty parametru (žlutě či modře v případě kladného či záporného odchýlení), nebo jednoduše na velikosti hodnot v případě průměrného počtu segmentů a počtu neúspěšných segmentací.

Na shrnutí experimentů lze rovněž pozorovat, že nejvíce byly otestovány různé kombinace hodnot 4 parametrů `halfWindowWidth`, `maxLineLength`, `edgeProbabilityThreshold` a `minSegmentLength`. Jejich dopad na výsledky jednotlivých experimentů je vizualizován pomocí grafů v obrázku 7.19. Důvodem volby právě těchto parametrů je, že právě tyto parametry byly pro modifikaci chování metody nejvhodnější a zároveň je úprava právě těchto parametrů doporučena autorem metody [12].

Nastavení parametru `standardDeviation` mělo minimální, často negativní a nepředvídatelný efekt. Parametry `priorEdgeProbability` a `signLineProbThreshold` mají pak velmi podobný dopad na výslednou segmentaci jako parametr `edgeProbabilityThreshold` (tj. všechny mají obdobný vliv na potřebnou lokální významnost k výběru sémanticky významné hranice). U parametru `pyramidLevels` byl pozorován negativní efekt na výslednou segmentaci. Nastavování různých hodnot posledního z méně testovaných parametrů `monteCarloTrials` pak nemá pro měření kvality význam.

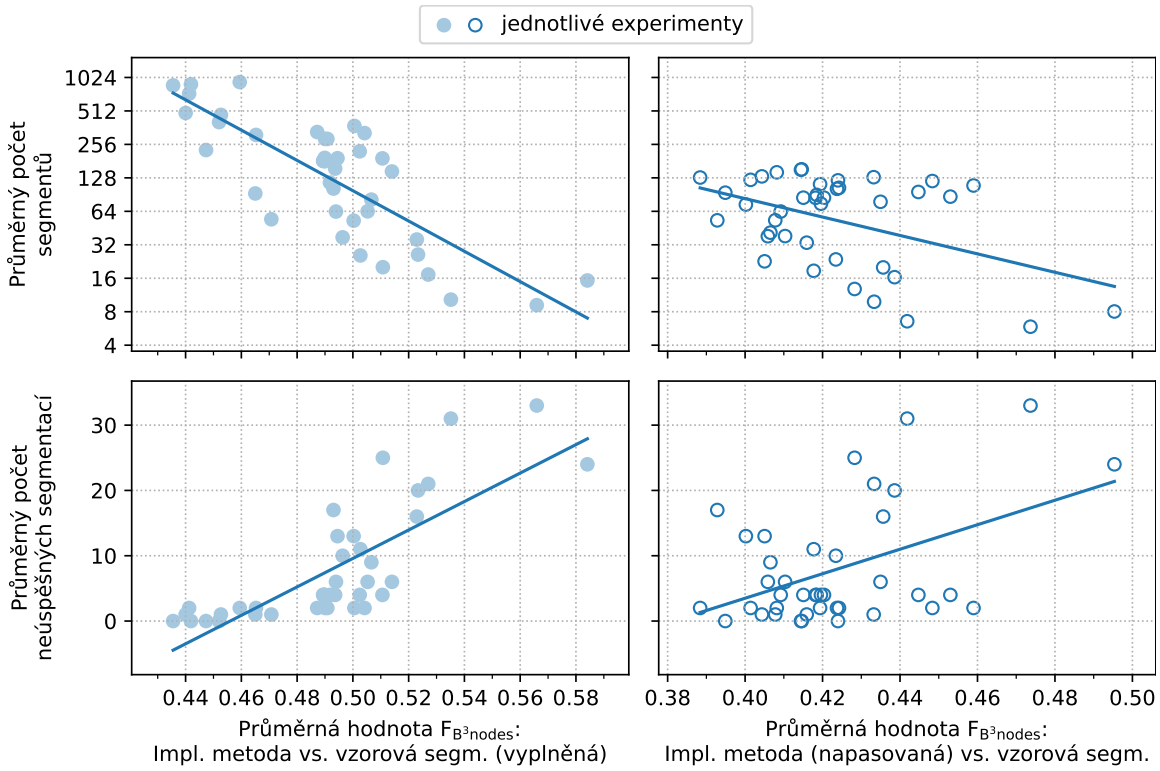


Obrázek 7.19: Vizualizace kvality segmentací jednotlivých experimentů v závislosti na zvolených parametrech. Průměrně nejlepších výsledků nezávisle na sobě dosahují nastavení parametrů `halfWindowWidth` = 15, `maxLineLength` = 384, `minSegmentLength` = 30 a `edgeProbabilityThreshold` = 0,25. Kombinace tohoto nastavení parametrů dosahuje podle tabulky 7.7 nejlepších výsledků.

Ze shrnutí výsledků lze dále např. pozorovat konzistentně horší výsledky v případě napasování segmentací na DOM uzly stránky, průměrně o 6,22 % oproti výsledkům s vyplněnou vzorovou segmentací. Zajímavé je, že tento rozdíl má tendenci se snižovat v případech, kdy výsledky trpí tak výrazným přesegmentováním, že napasováním je většina z nich odstraněna – díky čemuž je problém po napasování zmírněn. Tento jev se vyskytuje např. u výsledků na 21., 28. nebo 31. pozici tabulky 7.7.

Výraznějším znakem výsledků je ovšem již zmíněný trend vyššího počtu segmentů u experimentů s lepšími výsledky. Ten je způsoben volnějším nastavením parametrů, které dovolují vytvářet segmenty, kde jiné nastavení parametrů nikoliv. To sice způsobuje snížení

počtu neúspěšných segmentací a zlepšení celkové kvality segmentace, zároveň je ale snížena kvalita těch segmentací, které jsou úspěšné i pro přísnější nastavení. Tento efekt je vizualizován v grafech na obrázku 7.20.



Obrázek 7.20: Vizualizace souvislosti průměrného počtu segmentů na stránku a průměrného počtu neúspěšných segmentací s kvalitou segmentací kvantifikovanou metrikou $F_{B^3\text{nodes}}$ po odstranění neúspěšných segmentací. Jasný trend lze pozorovat zejména po vyplnění vzorové segmentace, ze kterého lze pozorovat, že kvalita úspěšných segmentací roste spolu s klesajícím počtem segmentů a s rostoucím počtem neúspěšných segmentací.

Jako vhodný kompromis se jeví nastavení parametrů na 4. pozici v tabulce 7.7. To má oproti jiným experimentům ve vyšších pozicích v tabulce poměrně rozumný průměrný počet segmentů 137,4 (73,3 po napasování) a přitom pouze 6 neúspěšných segmentací z celkového vzorku dat o 100 stránkách. Kvalita pouze úspěšných segmentací tohoto nastavení je zároveň s hodnotou metriky $F_{B^3\text{nodes}}$ po vyplnění vzorové segmentace 43,5 % stále 7. nejlepším výsledkem ze všech experimentů (a zároveň lepší, než kterékoliv nastavení parametrů z prvních 26 pozic tabulky 7.7).

Pokud by v dané situaci nebyl problém s vícero běhy metody, pak by mohlo být řešením spuštění s parametry v experimentu na 27. pozici v tabulce 7.7, jehož úspěšné segmentace na základě hodnoty metriky $F_{B^3\text{nodes}}$ pro vyplněné vzorové segmentace rovné 58,41 % dosahují nejlepší kvality ze všech experimentů. Toto nastavení má ovšem poměrně vysoký podíl neúspěšných segmentací, s celkem 24 ze vzorku o 100 stránkách. Pro takové případy (které lze snadno detekovat tím, že obsahují pouze jediný segment přes celou stránku) by pak mohlo být použito některé z nastavení z vyšších pozic tabulky 7.7, jako např. výše zmíněné ze 4. pozice.

Jako další možnou alternativou se nabízí rovnou použití některých z nastavení parametrů z vyšších pozic v tabulce 7.7 a následné oříznutí stromu segmentací po určitou hloubku za účelem zmírnění problému nadbytečné segmentace. Výsledné segmentací stromy jsou ovšem rozsáhlé zejména do šířky, což značně omezuje efektivitu takového přístupu.

7.4 Rychlost segmentace

Vzhledem ke značné časové náročnosti metody a aplikovaných optimalizací v průběhu její implementace je vhodné se zaměřit také na její rychlost. Pro tento účel byla naměřena doba trvání segmentace implementovanou metodou pro různé případy. Měření byly provedeny na systému s procesorem AMD Ryzen 7 3700X o 16 vláknech (ve výchozím nastavení) a 32 GB DDD4 paměti s frekvencí 3 600 MHz a CL[5] 16 (v XMP[6] nastavení). Jako vstup pro měření nebyl použit doposud využívaný vzorek z datové sady, ale uměle vytvořený vstupní obrázek stránky s různými, pevně danými velikostmi pro přesnější analýzu závislosti na velikosti vstupního obrázku. Výsledky všech měření budou rozebrány touto sekcí.

V rámci úvodní analýzy byla změřena celková doba běhu implementované metody. Graf 7.21 vizualizuje její závislost na velikosti vstupního obrázku stránky a parametru `halfWindowWidth`, který má velmi výrazný vliv na časovou náročnost výpočtu lokální významnosti hran (viz sekce 6.1.3). V grafu lze pozorovat nárůst doby výpočtu metody po lineárním navýšení tohoto parametru, který je velmi podobný tomu po exponenciálním navýšení velikosti vstupního obrázku stránky – což významný dopad parametru `halfWindowWidth` potvrzuje.

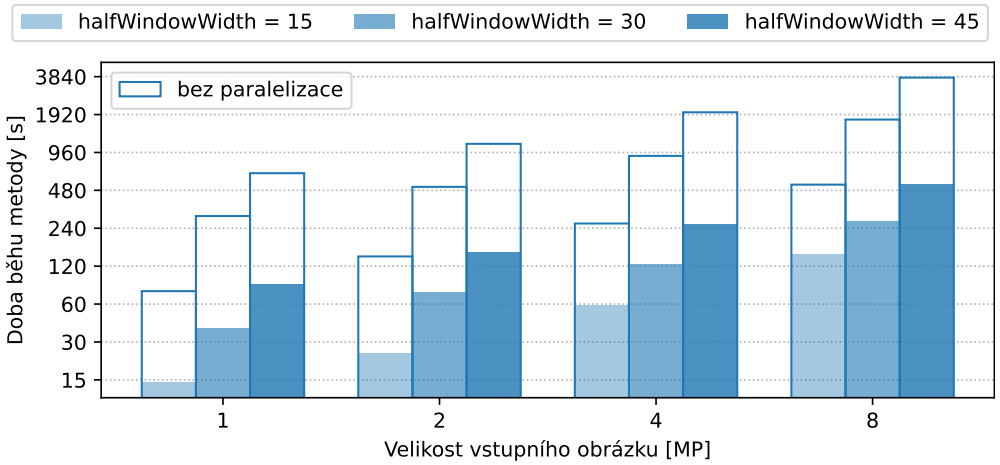
Dalším výrazným rysem grafu 7.21 je efektivita zrychlení výpočtu paralelizací. Spuštění metody na 16 jádrech zajistilo více než 7× kratší celkovou dobu výpočtu. To ovšem platí pouze pro hodnoty parametru `halfWindowWidth` ≥ 30, pro `halfWindowWidth` $= 15$ zrychlení klesá na 5násobek.

Grafy v obrázku 7.22 se následně soustředí pouze na nejnáročnější fázi výpočtu lokálně významných hran. V rámci implementace byly na tuto fázi aplikovány 2 hlavní optimalizace, jejichž efektivitu každý z grafů vizualizuje. Paralelizace zajišťuje v případě výpočtu na 16 vláknech až 8násobné zrychlení. Ukládání instancí `NormalDistribution` pak sice představuje pouze 4násobné, ovšem nezávisle na počtu dostupných vláken pro výpočet.

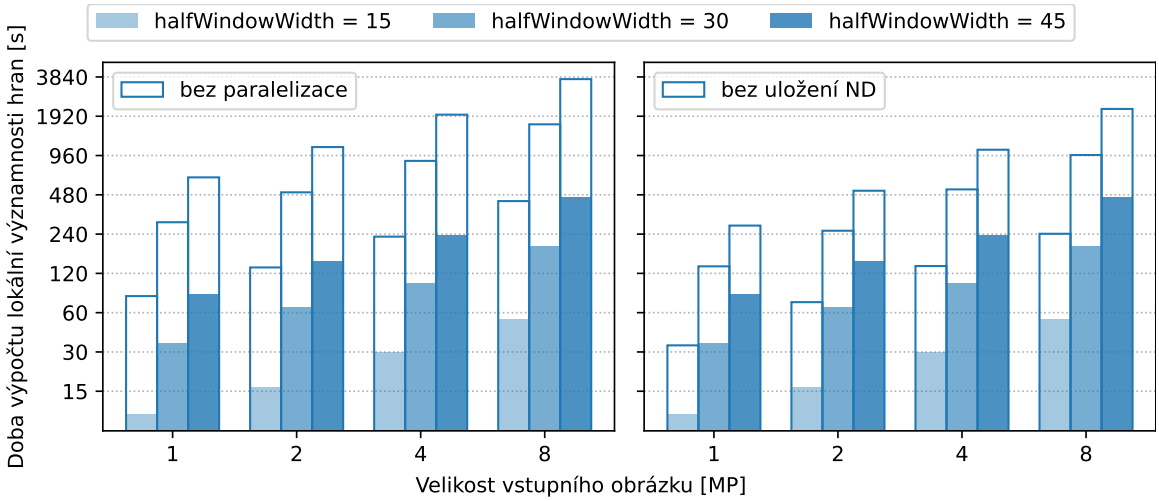
Ukládání instancí `NormalDistribution` má ale výrazný dopad na paměťové nároky metody. Ty jsou ilustrovány jednoduchým grafem 7.23. Pro velikost vstupního obrázku 8 MP používá metoda až 5 GB paměti, tedy 5násobek potřebné paměti bez této optimalizace. Zajímavé je, že pro menší obrázky (velikost ≤ 2 MP) je dopad ukládání instancí `NormalDistribution` na paměť minimální.

[5] CL - *CAS Latency* je jeden z klíčových parametrů paměťových modulů, díky kterému lze, spolu s jeho frekvencí, poměrně přesně určit rychlost daného modulu.

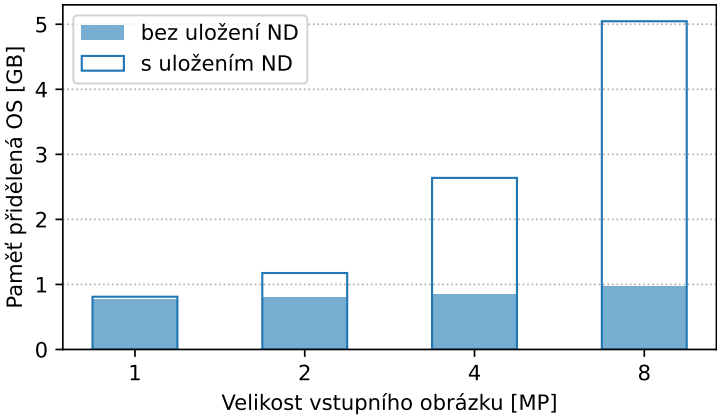
[6] XMP – *Extreme Memory Profile* je výrobcem předpřipravené nastavení paměti, jako např. její frekvence.



Obrázek 7.21: Doba běhu metody (v logaritmickém měřítku) v závislosti na velikosti vstupního obrázku (kde jednotka MP jsou *megapixely*, tj. 1 MP = 1 000 000 bodů obrázku) a na hodnotě parametru `halfWindowWidth`. Pro každý případ je navíc zobrazena doba trvání metody po vypnutí paralelizace výpočtu lokální významnosti hran a tvorby stromu segmentací (blíže popsané v rámci kapitoly 6). Pro hodnotu parametru `halfWindowWidth` = 15 lze pozorovat po aplikování paralelizace průměrné zrychlení na 482 %. Pro hodnoty tohoto parametru 30 a 45 je pak zrychlení až na 709 % a 744 %.



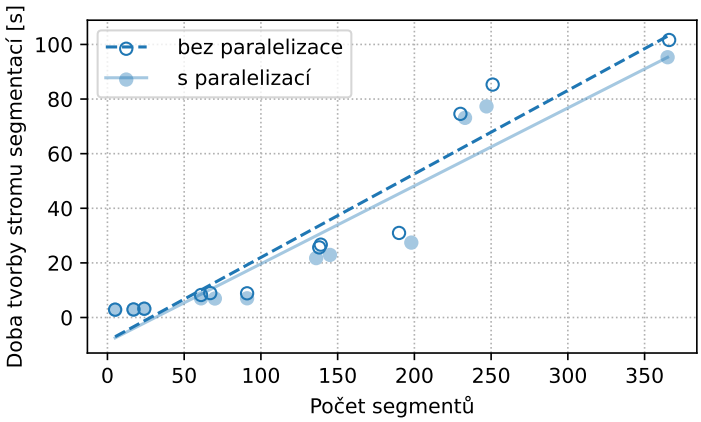
Obrázek 7.22: Doba výpočtu lokální významnosti hran (v logaritmickém měřítku) v závislosti na velikosti vstupního obrázku. V levém grafu je navíc doba výpočtu po vypnutí paralelizace, v pravém je pak navíc doba výpočtu po vypnutí ukládání instancí `NormalDistribution` (blíže popsané v sekci 6.1.3). Průměrné zrychlení dosaženého paralelizací je 814 %. Oproti tomu uložením instancí `NormalDistribution` je rychlost výpočtu průměrně zvýšena na 427 %.



Obrázek 7.23: Paměť přidělená operačním systémem během výpočtu implementované metody v závislosti na velikosti vstupního obrázku. Pro každou velikost vstupního obrázku jsou zobrazeny hodnoty bez ukládání instancí `NormalDistribution` a s ukládáním těchto instancí.

Cílem posledního pozorování je proces tvorby stromu segmentací. V grafu 7.24 lze pozorovat efekt paralelizace tohoto procesu, který není příliš významný – zejména pro nižší počty segmentů. To lze vysvětlit principem tvorby jednotlivých segmentů, jehož hlavní částí tvoří výpočet sémantické významnosti potenciálních dělících linií. Tento proces trvá zpravidla nejdéle pro kořenový segment, který je největší a doba hledání hranic pro jeho rozdělení je tak nejdelší. Zároveň je díky způsobu paralelizace, kdy každé vlákno zpracovává jeden segment, kořenový segment zpracováván jediným vláknem.

Horší efektivita této optimalizace rovněž vysvětluje nižší efektivitu paralelizace pro menší hodnoty parametru `halfWindowWidth` v grafu 7.21: V takových případech má výsledný strom segmentací pro testovaný obrázek stránky větší počet segmentů a tudíž tvorba segmentačního stromu představuje větší část celkového výpočtu.



Obrázek 7.24: Doba výpočtu potřebná pro tvorbu výsledného stromu segmentací v závislosti na počtu vytvořených segmentů. Graf vizualizuje hodnoty před a po zapnutí paralelizace tohoto výpočtu. Úsečky pak vyjadřují trend pro každý z těchto případů aproximovaný pomocí lineární regrese.

Kapitola 8

Závěr

Cílem této práce bylo rozšíření nástroje FitLayout[1] o další publikovanou metodu segmentace webových stránek. V rámci práce byla taková metoda vybrána a úspěšně integrována do tohoto nástroje, čímž byl tento cíl splněn.

Práce obsahuje seznámení s nástrojem FitLayout a uvedení jeho součástí potřebných pro integraci nové metody. Zároveň je jejím obsahem úvod do problematiky segmentace webových stránek, na kterou navazuje popis vybrané metody, včetně odůvodnění výběru právě této metody. Zvolená metoda byla dále implementována a úspěšně integrována do nástroje FitLayout. Následovalo ověření věrohodnosti této implementace vůči publikacím metody a nakonec se práce zaměřila na její podrobnou analýzu, jejíž výsledky jsou shrnuty ve výsledném vyhodnocení.

Při ověřování věrohodnosti implementace byla naměřena až 83,91 % shoda s veřejně dostupnými výsledky metody. Součástí analýzy pak bylo hledání optimálních parametrů pro nejlepší možnou kvalitu výstupu implementované metody, v jehož rámci bylo otestováno nastavení parametrů, které dosahovalo zlepšení kvality segmentace až o 9,89 % oproti výchozímu nastavení. Hlavním přínosem analýzy kvality bylo ovšem doporučení použití různých nastavení parametrů v různých situacích. Mimo kvalitu segmentace byla pak metoda implementována tak, aby její rychlost škálovala s počtem vláken procesoru.

Díky této práci jsem získal znalosti v oblasti segmentace webových stránek. Seznámil jsem se s aktuálními problémy v této oblasti a s dostupnými řešeními, které jsou na tyto problémy v současné době aplikovány. Rovněž si odnáším zkušenost s rozšiřováním rozsáhlého vědeckého projektu, jako je nástroj FitLayout.

Na výsledky této práce lze navázat několika způsoby. Jedním z nich je využití integrace nové metody v nástroji FitLayout k vyhodnocování a dalšímu experimentování s dalšími metodami segmentace, o které bude nástroj v budoucnu dále rozšiřován. Dalším způsobem jsou možné modifikace samotné logiky metody v návaznosti na rozbor jejího chování v rámci vyhodnocení s cílem zmírnění jejích specifických problémů. Nakonec lze také navázat na optimalizaci implementace experimentováním s GPU akcelerací, která by mohla potenciálně dále zrychlit výpočet lokální významnosti.

[1] FitLayout – https://github.com/FitLayout

Literatura

[1] Alcic, S. a Conrad, S. Page Segmentation by Web Content Clustering. In: *Proceedings of the International Conference on Web Intelligence, Mining and Semantics*. New York, NY, USA: Association for Computing Machinery, Květen 2011, s. 1–9. WIMS '11. DOI: 10.1145/1988688.1988717. ISBN 9781450301480.

[2] Amigó, E., Gonzalo, J., Artiles, J. a Verdejo, F. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*. Srpen 2009, sv. 12, s. 461–486. DOI: 10.1007/s10791-008-9066-8. ISSN 1573-7659.

[3] Andrew, J. J., Ferrari, S., Maurel, F., Dias, G. a Giguet, E. Model-Driven Web Page Segmentation for Non Visual Access. In: Nguyen, L.-M., Phan, X.-H., Hasida, K. a Tojo, S., ed. *Computational Linguistics*. Singapore: Springer Singapore, 2020, s. 191–205. ISBN 978-981-15-6168-9.

[4] Bartík, V. a Burget, R. Two-Phase Categorization of Web Documents. In: *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval*. Valencia, ES: Institute for Systems and Technologies of Information, Control and Communication, 2010, s. 458–462. ISBN 978-989-8425-28-7. Dostupné z: https://www.fit.vut.cz/research/publication/9381.

[5] Bartík, V. *Získávání znalostí z databází: Dolování na webu*. Brno: Fakulta informačních technologií VUT, 2022.

[6] Burget, R. Automatic Document Structure Detection for Data Integration. In: Abramowicz, W., ed. *Business Information Systems*. Berlin, Heidelberg: Springer, Duben 2007, s. 391–397. Lecture Notes in Computer Science. ISBN 978-3-540-72035-5.

[7] Burget, R. FitLayout/2 - Web Page Analysis Framework. *FitLayout README* [online]. Březen 2020. Revidováno 16. 11. 2022 [cit. 2023-01-08]. Dostupné z: https://github.com/FitLayout/FitLayout/blob/main/README.md.

[8] Burget, R. a Salem, H. FitLayout: An RDF-Based Framework and Toolkit for Web Page Content Analysis. *Semantic Web Journal*. IOS Press. 2022. ISSN 1570-0844. V posuzování.

[9] Cai, D., Yu, S., Wen, J.-R. a Ma, W.-Y. *VIPS: a Vision-based Page Segmentation Algorithm*. MSR-TR-2003-79. Redmond, WA 98052: Microsoft Research, listopad 2003.

[10] Canny, J. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. IEEE. Listopad 1986, PAMI-8, s. 679–698. DOI: 10.1109/TPAMI.1986.4767851. ISSN 0162-8828.

[11] Chakrabarti, D., Kumar, R. a Punera, K. A Graph-Theoretic Approach to Webpage Segmentation. In: *Proceedings of the 17th International Conference on World Wide Web*. New York, NY, USA: Association for Computing Machinery, Duben 2008, s. 377–386. WWW '08. DOI: 10.1145/1367497.1367549. ISBN 9781605580852.

[12] Cormier, M. *Computer Vision on Web Pages: A Study of Man-Made Images* [online]. Waterloo, Ontario, Canada, 2018. [cit. 2023-01-13]. Disertační práce. University of Waterloo. Dostupné z: http://hdl.handle.net/10012/13523.

[13] Cormier, M., Mann, R., Moffatt, K. a Cohen, R. Towards an Improved Vision-Based Web Page Segmentation Algorithm. In: *2017 14th Conference on Computer and Robot Vision (CRV)*. IEEE, Květen 2017, s. 345–352. DOI: 10.1109/CRV.2017.38. ISBN 978-1-5386-2818-8.

[14] Cormier, M., Moffatt, K., Cohen, R. a Mann, R. Purely vision-based segmentation of web pages for assistive technology. *Computer Vision and Image Understanding* [online]. Elsevier. Únor 2016, sv. 148, s. 46–66, Revidováno 28. 10. 2015, [cit. 2023-01-09]. DOI: 10.1016/j.cviu.2016.02.007. ISSN 1077-3142. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1077314216000527.

[15] Feng, H., Zhang, W., Wu, H. a Wang, C.-J. Web Page Segmentation and Its Application for Web Information Crawling. In: *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, Listopad 2016, s. 598–605. DOI: 10.1109/ICTAI.2016.0097. ISBN 978-1-5090-4459-7.

[16] Jayashree, S. R., Dias, G., Andrew, J. J., Saha, S., Maurel, F. et al. Multimodal Web Page Segmentation Using Self-Organized Multi-Objective Clustering. *ACM Trans. Inf. Syst.* [online]. New York, NY, USA: Association for Computing Machinery. Březen 2022, sv. 40, č. 3, s. 1–49, [cit. 2023-01-08]. DOI: 10.1145/3480966. ISSN 1046-8188. Dostupné z: https://dl.acm.org/doi/10.1145/3480966.

[17] Kiesel, J., Kneist, F., Meyer, L., Komlossy, K., Stein, B. et al. Web Page Segmentation Revisited: Evaluation Framework and Dataset. In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. New York, NY, USA: Association for Computing Machinery, říjen 2020, s. 3047–3054. CIKM '20. DOI: 10.1145/3340531.3412782. ISBN 9781450368599.

[18] Kiesel, J., Meyer, L., Kneist, F., Stein, B. a Potthast, M. An Empirical Comparison of Web Page Segmentation Algorithms. In: Hiemstra, D., Moens, M.-F., Mothe, J., Perego, R., Potthast, M. et al., ed. *Advances in Information Retrieval*. Cham: Springer International Publishing, Březen 2021, s. 62–74. DOI: 10.1007/978-3-030-72240-1_5. ISBN 978-3-030-72240-1.

[19] Kohlschütter, C. a Nejdl, W. A Densitometric Approach to Web Page Segmentation. In: *Proceedings of the 17th ACM Conference on Information and*

Knowledge Management. New York, NY, USA: Association for Computing Machinery, říjen 2008, s. 1173–1182. CIKM '08. DOI: 10.1145/1458082.1458237. ISBN 9781595939913.

[20] Milička, M. a Burget, R. Information Extraction from Web Sources Based on Multi-aspect Content Analysis. In: Gandon, F., Cabrio, E., Stankovic, M. a Zimmerman, A., ed. *Semantic Web Evaluation Challenges*. Cham: Springer International Publishing, Květen 2015, s. 81–92. Communications in Computer and Information Science. ISBN 978-3-319-25518-7.

[21] Sanoja, A. a Gançarski, S. Block-o-Matic: A web page segmentation framework. In: *2014 International Conference on Multimedia Computing and Systems (ICMCS)*. IEEE, Duben 2014, s. 595–600. DOI: 10.1109/ICMCS.2014.6911249. ISBN 978-1-4799-3824-7.

[22] Sobel, I. An Isotropic 3x3 Image Gradient Operator. *Stanford Artificial Intelligence Project (SAIL)*. Únor 2014. Dostupné z: https://www.researchgate.net/publication/239398674_An_Isotropic_3x3_Image_Gradient_Operator.

[23] Weng, D., Hong, J. a Bell, D. A. Automatically Annotating Structured Web Data Using a SVM-Based Multiclass Classifier. In: Benatallah, B., Bestavros, A., Manolopoulos, Y., Vakali, A. a Zhang, Y., ed. *Web Information Systems Engineering – WISE 2014*. Cham: Springer International Publishing, 2014, s. 115–124. Lecture Notes in Computer Science. DOI: 10.1007/978-3-319-11749-2_9. ISBN 978-3-319-11749-2.

[24] Zeleny, J., Burget, R. a Zendulka, J. Box clustering segmentation: A new method for vision-based web page preprocessing. *Information Processing & Management* [online]. Elsevier. Květen 2017, sv. 53, č. 3, s. 735–750, Revidováno 1. 12. 2016, [cit. 2023-01-08]. DOI: 10.1016/j.ipm.2017.02.002. ISSN 0306-4573. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0306457316301169.

[25] Češka, M. *Statistics and Probability: Probabilistic Reasoning in Computer Science*. Brno: Fakulta informačních technologií VUT, 2021.

Příloha A

Obsah přiloženého média

Základní přehled obsahu přiloženého média:

- `data/` – Adresář obsahující všechna data použitá v rámci vyhodnocení.
- `dependencies/` – Adresář pro potřebné závislosti spustitelných programů.
- `doc/` – Adresář se zdrojovými soubory této práce.
- `src/` – Adresář se zdrojovými soubory spustitelných programů.
- `CormierDemo.jar` – Spustitelný program pro možnost použití implementované metody na již existující obrázky.
- `FitLayout.jar` – Spustitelný nástroj FitLayout v podobě pro příkazový řádek s integrovanou metodou Cormier et al.
- `README.md` a `README.pdf` – Textový soubor (a odpovídající soubor ve formátu PDF) obsahující podrobné informace o obsahu přiloženého média a podrobný návod k instalaci či spuštění přiložených programů, včetně požadovaných nástrojů.
- `thesis.pdf` – Elektronická verze této práce.