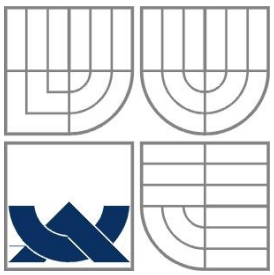
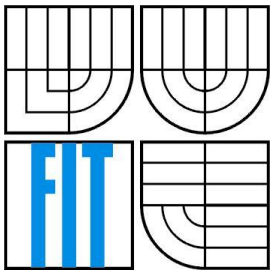


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

NOVÉ METODY SEGMENTACE WEBOVÝCH STRÁNEK

NEW METHODS OF WEBPAGE SEGMENTATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

BC. MICHAL MALANÍK

VEDOUCÍ PRÁCE
SUPERVISOR

ING. RADEK BURGET, PH.D.

BRNO 2016

Abstrakt

Cílem této práce je představit novou metodu segmentace webových stránek založenou na analýze vizuálních vlastností webových dokumentů. Metoda vychází z velmi populárního segmentačního algoritmu VIPS (Vision Based Page Segmentation Algorithm), který se snaží na segmentovaný dokument nahlížet stejně, jako ho ve výsledku vidí jeho uživatel, tedy prostřednictvím vizuální reprezentace v internetovém prohlížeči. Oproti algoritmu VIPS jsou však u metody představené v této práci brány v úvahu optimalizace pro moderní webové stránky, především poté pro dokumenty vytvořené v jazyce HTML verze 5. Práce se rovněž zabývá implementací navržené metody pomocí rámce FITLayout.

Abstract

The aim of this work is to introduce a new vision based web page segmentation method. This method is based on very popular VIPS segmentation algorithm, which is trying to represent the segmented web document in the same way as it is perceived by a user using a web browser. Compared to the VIPS algorithm, there are some optimizations for modern websites in our method, especially for documents created in the HTML 5 language. We also deal with the implementaion of the proposed method using the FITLayout framework.

Klíčová slova

Segmentace webových stránek, analýza webových stránek, algoritmus VIPS, FITLayout rámec

Keywords

Web page segmentation, web page analysis, VIPS algorithm, FITLayout framework

Citace

MALANÍK, Michal. *Nové metody segmentace webových stránek*. Brno, 2016. 52 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Burget Radek.

Nové metody segmentace webových stránek

Prohlášení

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

……
Michal Malaník
24. 5. 2016

Poděkování

Rád bych tímto poděkoval vedoucímu mé diplomové práce Ing. Radkovi Burgetovi, Ph.D. za poskytnuté vedení a odborné rady při tvorbě této práce.

Obsah

1 Úvod

V dnešní době se internet stal nejobsáhlejším zdrojem informací v historii lidstva a taktéž součástí našich každodenních životů. Vzhledem k tomu, že dochází k neustálému zvyšování počtu nově vzniklých webových stránek, jakožto základních nositelů informací na internetu, vzniká stále větší potřeba nástrojů pro automatické zpracování obsahu webových stránek. Stále častěji se začínají dostávat do popředí vědní disciplíny jako získávání znalostí z dokumentů nebo optimalizace internetového vyhledávání.

Většina webových stránek je logicky členěna na menší části, které kromě vlastního užitečného obsahu mohou reprezentovat prvky pro navigaci, design, či kontaktní informace. Proces oddělení těchto významově odlišných prvků a jejich následné klasifikace na relevantní a nerelevantní informace se nazývá segmentace webových stránek.

Velké množství segmentačních metod se snaží o analýzu webových stránek pouze na základě vnitřní hierarchické struktury značek jazyka HTML[1]. Především se soustředí na informace jako název jednotlivých HTML značek a jejich pozice vůči ostatním. Vezmeme-li však v úvahu rozmanitost jazyka HTML v oblasti tvorby webových dokumentů, navíc ještě v kombinaci s technologií kaskádových stylů (CSS[2]), nelze u moderních webových stránek počítat s příliš uspokojivými výsledky. [8]

V rámci této práce se budeme zabývat vytvořením nové segmentační metody založené více na analýze vizuálních informací poskytovaných uživateli webové stránky, než na analýze samotného zdrojového kódu dokumentu. Jako základ nové segmentační metody byl použit velice oblíbený algoritmus VIPS (Vision Based Page Segmentation Algorithm), který i přes svou delší existenci je stále schopen poskytovat poměrně uspokojivé výsledky segmentačního procesu. Protože se však opravdu jedná o jeden z nejstarších algoritmů pro vizuální segmentaci webových stránek a internetové technologie od jeho vzniku pokročili o obrovský krok kupředu, budeme se dále zabývat možnostmi jeho vylepšení a optimalizace pro moderní webové stránky, především poté pro dokumenty vytvořené v jazyce HTML verze 5[3]. Cílem této práce je poté navrženou segmentační metodu implementovat do již existujícího nástroje pro segmentaci webových stránek s názvem FITLayout, vzniklého na Fakultě informačních technologií VUT v Brně.

V kapitole 2 se nachází úvod do segmentace webových stránek spolu se základním popisem některých algoritmů pro vizuální segmentaci.

Kapitola 3 se poté věnuje základnímu popisu nástroje FITLayout, v rámci kterého je implementována vytvořená segmentační metoda.

Podrobným popisem jednotlivých fází algoritmu VIPS, včetně formální definice struktury sloužící jako výsledek segmentačního procesu, se zabývá kapitola 4.

Následující kapitola s číslem 5 se zaměřuje na popis známých omezení algoritmu VIPS spolu s možnostmi jejich optimalizace.

Kapitola 6 se poté zabývá praktickým popisem implementované segmentační metody. Především se zaměřuje na konkrétní problémy, které bylo třeba v rámci implementace vyřešit a také popisuje použité způsoby optimalizace některých omezení popsaných v kapitole 5.

Poslední kapitola s pořadovým číslem 7 poté ukazuje některé výsledky testování vytvořené segmentační metody na reálných webových dokumentech.

[1] HyperText Markup Language - http://www.w3.org/html/
[2] Cascading Style Sheets - http://www.w3.org/Style/CSS/
[3] HyperText Markup Language version 5 - http://www.w3.org/TR/html5/

2 Segmentace webových stránek

Segmentací webových stránek rozumíme činnost, při které se snažíme identifikovat nebo odlišit prvky webové stránky, které obsahující relevantní informace. Přesto, že se v této práci zabýváme především segmentací webových stránek, je důležité zmínit, že pojem segmentace jako takový souvisí s odvětvím informačních technologií zvaným získávání znalostí z dokumentů, kde těmito dokumenty můžeme rozumět i jiné formáty textových dokumentů, než jsou webové stránky. Jako příklad můžeme uvést dokumenty ve formátu PDF[4], které mohou být také velice často cílem segmentačního procesu, a to především z důvodu svého širokého rozšíření mezi uživateli.

Z hlediska segmentace webových stránek však narážíme na zásadní skutečnost, a sice že se způsob tvorby webových dokumentů neustále vyvíjí, objevují se nové trendy a techniky, a tudíž se identifikace významově odlišných prvků nebo částí webové stránky stává v průběhu let čím dál obtížnější. Mimo hlavního obsahu se na moderních webových stránkách vyskytuje také velké množství jiných textových prvků, jako jsou například navigační menu, texty reklam, komentáře uživatelů, nebo části přiložených textových dokumentů. Oddělení, tedy segmentování těchto významově odlišných prvků webové stránky a jejich následné klasifikování na relevantní a nerelevantní části je zásadní myšlenkou segmentace moderních webových stránek. [6]

Některé přístupy se snaží o segmentování obsahu webových dokumentů na čistě textové úrovni, ale výsledky těchto metod nejsou příliš uspokojivé. Své uplatnění nalezly především v počátcích internetu a v dnešní době již nejsou samostatně téměř vůbec využívány.

Daleko častější jsou přístupy, které se snaží vyvodit významově rozdílné části webových stránek na základě DOM[5], neboť DOM poskytuje hierarchickou strukturu značek jazyka HTML, tvořících webové dokumenty. Mnoho výzkumů bylo založeno na použití informací o jednotlivých značkách a snaze dělit webovou stránku na základě některých důležitých typů HTML značek. Jedná se především o značky reprezentující odstavce `<P>`, tabulky `<TABLE>`, seznamy `<UL>` a jednotlivé úrovně nadpisů `<H1>` až `<H6>`. I u tohoto přístupu se však setkáváme s řadou negativních aspektů. Vzhledem k značné volnosti a flexibilitě jazyka HTML mnoho webových stránek nesplňuje standard W3C[6] pro tvorbu webových dokumentů, což může způsobovat chyby ve struktuře DOM stromu a negativně tak ovlivňovat výsledky segmentačního procesu. Jedná se například o situace, kdy tabulka není použita jen k organizaci obsahu, ale také k formátování rozložení jednotlivých prvků na webové stránce. Navíc DOM strom byl spíše vyvinut za účelem jeho interpretace a následné prezentace v internetovém prohlížeči, než za účelem poskytování informací o významu jednotlivých prvků na webové stránce. Když dva uzly DOM stromu mají stejný rodičovský uzel, nemusí to ještě znamenat, že z hlediska jejich významu jsou spolu provázány pevněji, než s ostatními uzly DOM stromu. [3][4]

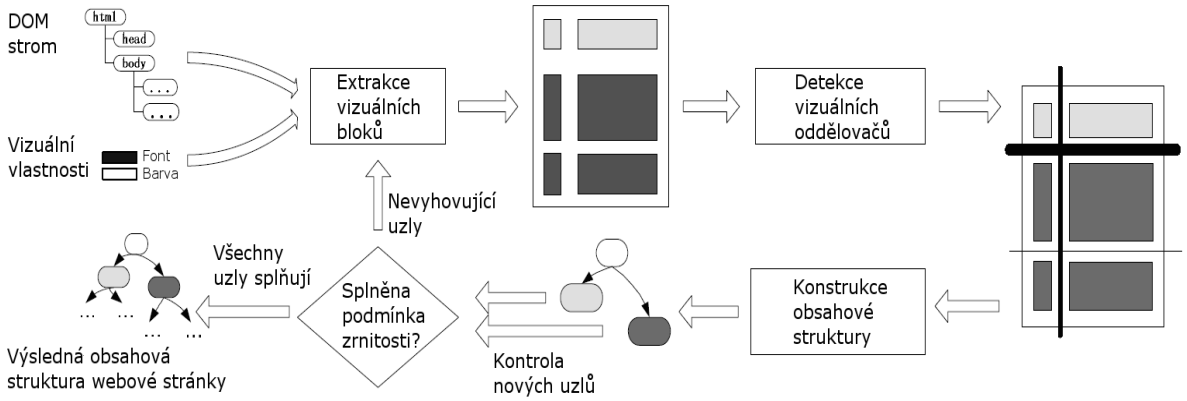
Velmi výhodné je při dělení webové stránky na významově odlišné části využít vizuální reprezentace stránky v internetovém prohlížeči. Tento přístup má hned několik podstatných odůvodnění. Především je webová stránka zpracovávána ve stejné formě, ve které ji vidí její uživatel. Také se dá předpokládat, že tvůrce webové stránky provede návrh tak, aby důležitý obsah byl dobře čitelný, na první pohled viditelný a tedy patřičně oddělený a vizuálně uzpůsobený. Tyto hlediska jsou z pohledu kvalitní segmentace webových stránek nesmírně důležité. V následujícím textu si podrobněji rozebereme některé existující segmentační algoritmy založené na vizuální reprezentaci webových stránek a zaměříme se na výhody, které mohou být z hlediska této práce důležité.

[4] Portable Document Format
[5] Document Object Model - http://www.w3.org/DOM
[6] World Wide Web Consortium - http://www.w3.org/

2.1 Agoritmus Vision-based Page Segmentation (VIPS)

Zásadní myšlenkou algoritmu VIPS je snaha na segmentovaný dokument nahlížet stejně, jako na něj nahlíží jeho uživatel. Běžný uživatel webové stránky k jejímu zobrazení používá internetový prohlížeč a získává tak dvourozměrnou vizuální reprezentaci zobrazovaných prvků. V této reprezentaci jsou snadno rozeznatelné rozdílné části webové stránky, na první pohled jsou z hlediska uživatele viditelné jednotlivé nadpisy, odstavce, obrázky, nebo rozdílné velikosti a barvy písma.

Na základě této myšlenky je výsledná struktura obsahu dokumentu v algoritmu VIPS založená na vzhledu segmentovaného dokumentu. Z pohledu webové stránky to znamená, že výsledná struktura obsahu je odvozená z kombinace DOM stromu a vizuálních vlastností jednotlivých prvků. Po získání těchto informací z internetového prohlížeče započne samotný proces segmentace, který se skládá ze tří hlavních fází. Jedná se o fáze extrakce vizuálních bloků, detekce vizuálních oddělovačů a konstrukce obsahové sktruktury. Celý proces segmentace algoritmu VIPS je poté znázorněn na obr. 1. Ve fázi extrakce vizuálních bloků je procházen DOM strom webové stránky od kořenového uzlu směrem dolů a u každého uzlu je na základě speciálních heuristických pravidel algoritmu posouzeno, zdali se jedná o vizuální blok. Ve druhé fázi detekce vizuálních oddělovačů jsou detekovány linie webové stránky, které se nepřekrývají ani nekříží s žádným vizuálním blokem. Každému detekovanému oddělovači je poté přiřazena jeho váha dle speciálních pravidel. Nakonec je ve fázi konstrukce obsahové struktury sestavována výstupní struktura segmentačního procesu na základě slučování vizuálních bloků obklopujících oddělovače s nejmenší udělenou vahou. [2]

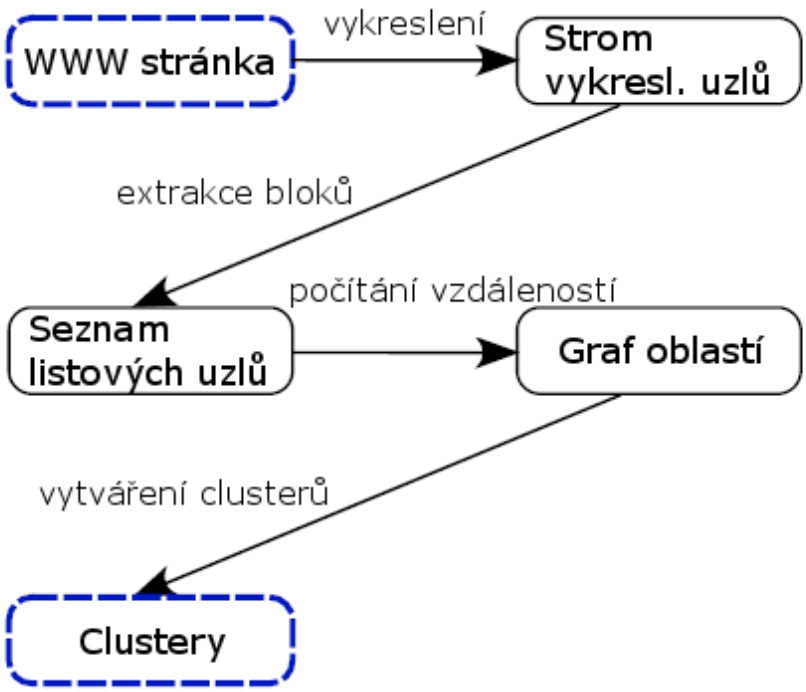


obr. 1: Schématické znázornění algoritmu VIPS [2]

2.2 Algoritmus Box Clustering Segmentation (BCS)

Oproti ostatním segmentačním metodám je algoritmus BCS navržen tak, aby jako výsledek segmentace poskytoval dlaždicové uspořádání jednotlivých segmentů webové stránky namísto hierarchické obsahové struktury, která bývá obvyklým výstupem segmentačních metod. Celý proces

algoritmu BCS je znázorněn na obr. 2, kde každý box reprezentuje stav, ve kterém se nachází zpracovávaná data a každý přechod mezi těmito stavy reprezentuje akci, která je vykonávána. [7]



obr. 2: Grafické znázornění algoritmu BCS [7]

První box znázorňuje vstup algoritmu, tedy webovou stránku ve formě HTML kódu, nebo ve formě odpovídajícího DOM stromu. Akce vykreslení je vykonána mimo samotný algoritmus a k jejímu provedení může být využito libovolného vykreslovacího stroje. Jako příklad můžeme uvést vykreslovací stroj CSSBox, o němž si více povíme v následujících kapitolách. Po provedení těchto počátečních činností přichází na řadu samotný algoritmus BCS, který se skládá ze tří hlavních fází. První z nich je označená jako extrakce bloků. V této fázi se z dat poskytnutých vykreslovacím strojem odfiltrují ty části stromové struktury, které nejsou z pohledu další segmentace užitečné. Ve druhé fázi jsou spočítány vzdálenosti mezi zbývajícími bloky a na jejich základě je sestaven graf oblastí. V poslední fázi vyváření clusterů je zpracován graf oblastí a jsou identifikovány segmenty webové stránky za pomoci shlukování boxů patřících do stejného segmentu. Pro více informací o jednotlivých fázích algoritmu lze nahlédnout do [7].

Pokud algoritmus porovnáme s oblíbeným algoritmem VIPS, který je podrobně rozebrán v kapitole 3, nacházíme nespornou výhodu algoritmu BCS. Ta spočívá v tom, že princip shlukování boxů je výrazně rychlejší než princip algoritmu VIPS, přičemž je dosáhnuto srovnatelné přesnosti segmentačního procesu.

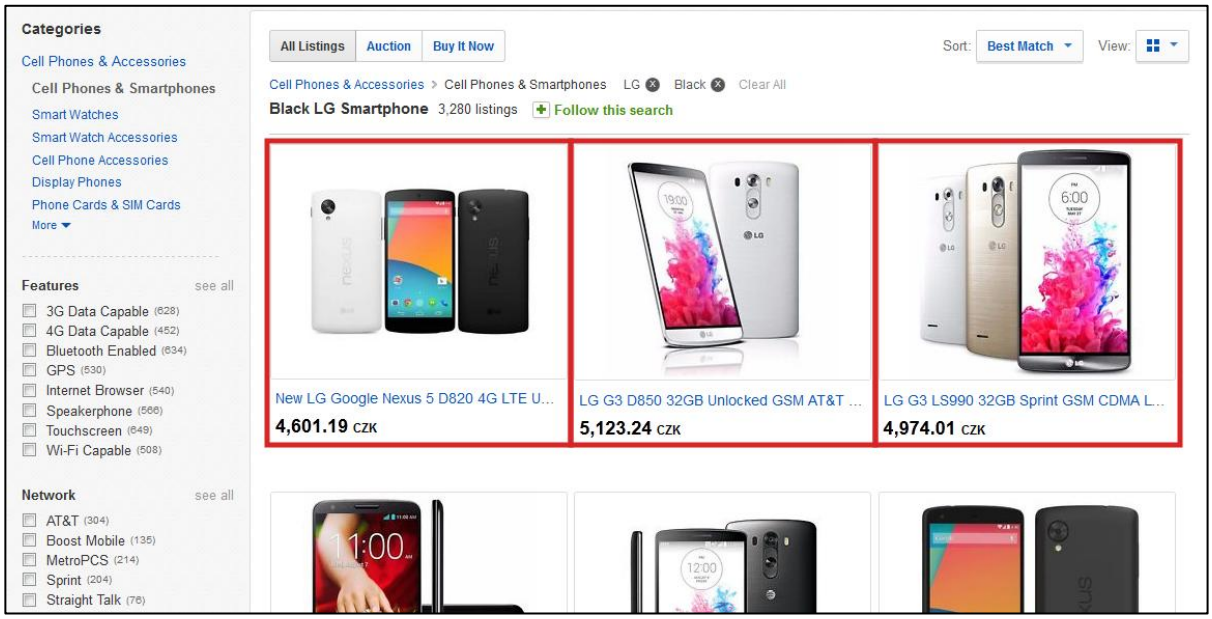
2.3 Algoritmus využívající vizuální sémantiky

Na rozdíl od jiných algoritmů založených na zpracování vizuální reprezentace webových stránek tento algoritmus nepracuje na základě analyzování vizuálních vlastností jednotlivých prvků webové stránky, ale namísto toho využívá tří základních měřítek k formulování vizuálních sémantik. Jedná se o následující měřítka:

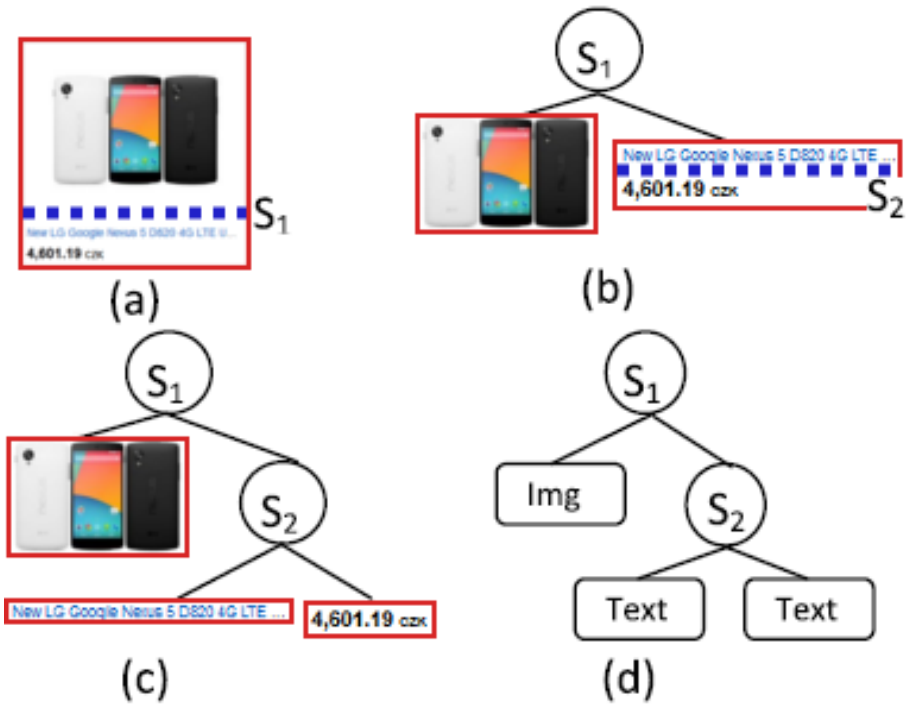
1) Strom rozložení, který je využíván k rozpoznání vizuálně podobných bloků.

2) Stupeň švu, který popisuje, jak úhledně jsou jednotlivé bloky rozmístěny.
3) Obsahová podobnost, která popisuje vzájemnou obsahovou konzistenci mezi jednotlivými bloky.

Některé webové stránky obsahují větší množství velice podobných úseků. Například na obr. 3 můžeme vidět webovou stránku internetového obchodu, kde jednotlivé produkty, ohraničené červeným rámečkem, tvoří velice podobné vizuální bloky. Z pohledu této webové stránky se jedná o bloky, jejichž význam je důležitý a nebylo by vhodné je dále dělit na menší úseky. Konstrukce stromu rozložení je znázorněna na obr. 4, kde jsou také symbolicky znázorněny dva oddělovače S1 a S2 modrou přerušovanou čarou. Každý z těchto oddělovačů může být použit k dělení bloku na menší části. Místo toho však zkonstruujeme strom rozložení, kde bude oddělovač S1 sloužit jako kořenový uzel, neboť dělí blok na dvě hlavní části. Zbytek stromu je zkonstruován obdobně. V okamžiku, kdy máme zkonstruovány stromy rozložení všech bloků na webové stránce, můžeme za pomoci vhodného algoritmu provést porovnání těchto stromů a určení, zdali si bloky jsou, nebo nejsou vizuálně podobné. [9]



obr. 3: Příklad vizuálně podobných bloků webové stránky [9]

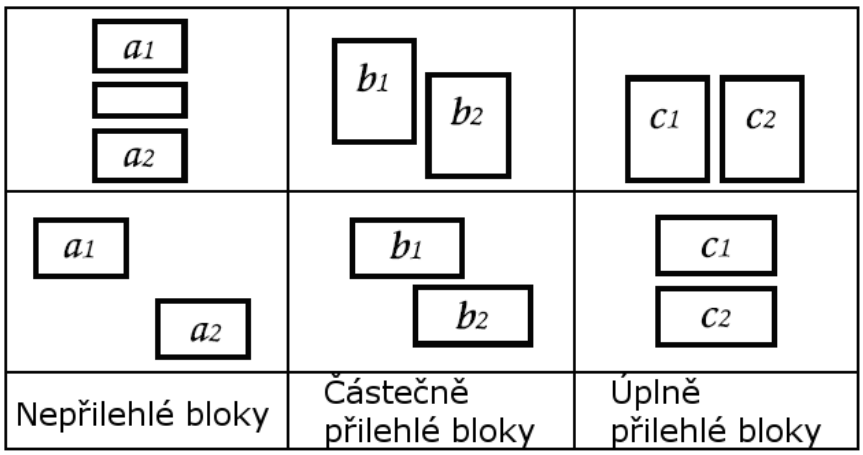


obr. 4: Příklad konstrukce stromu rozložení [9]

Stupeň švu poté vyjadřuje, jak úhledně jsou k sobě dva bloky webové stránky umístěny. Dá se totiž předpokládat, že významné bloky webové stránky k sobě budou vždy blízce a úhledně umístěny, a tudíž spolu mohou tvořit jeden segment webové stránky. Pro jakékoliv dva bloky webové stránky tedy stanovíme klasifikaci na tyto tři typy:

1) Nepřilehlé bloky
2) Částečně přilehlé bloky
3) Úplně přilehlé bloky

Jednotlivé typy rozložení bloků jsou znázorněny na obr. 5. Na základě vzájemného umístění bloků je poté spočítána hodnota stupně švu dle specifických rovnic.



obr. 5: Znázornění jednotlivých typů rozložení bloků [9]

Dá se říci, že bloky webové stránky, které mají rozdílný význam, mají také rozdílný typ obsahu. Například navigační menu obsahuje seznam krátkých textových odkazů, zatímco reklama může obsahovat jen jeden velký obrázek. Evidentně tyto různé typy obsahu budou mít také různé vizuální vlastnosti. Pokud jsou dva bloky webové stránky velmi podobné, mají také vysoký stupeň vzájemné obsahové konzistence. Za tímto účelem definujeme poslední měřítko s názvem obsahová podobnost, které popisuje právě stupeň vzájemné obsahové konzistence. [9]

Algoritmus využívající vizuálních sémantik tedy získá DOM strom dané webové stránky a dále rozpozná podobné bloky, spočítá hodnotu stupně švu a hodnotu obsahové podobnosti pro každý blok webové stránky. Následně je DOM strom procházen od kořenového uzlu směrem dolů a u všech bloků odpovídajících jednomu uzlu DOM stromu je na základě předem vypočítaných hodnot a speciálních pravidel posouzeno, zdali se jedná, nebo nejedná o segment. Výstupem segmentačního procesu je poté seznam jednotlivých segmentů.

2.4 Zhodnocení segmentačních algoritmů

Na Fakultě informačních technologií VUT v Brně byl již v minulosti vyvinut nástroj pro segmentaci webových stránek s názvem FITLayout. Cílem této práce je přidání nového algoritmu pro vizuální segmentaci webových stránek do tohoto existujícího nástroje. Proto bylo třeba zvolit vhodný segmentační algoritmus za účelem jeho implementace do nástroje FITLayout.

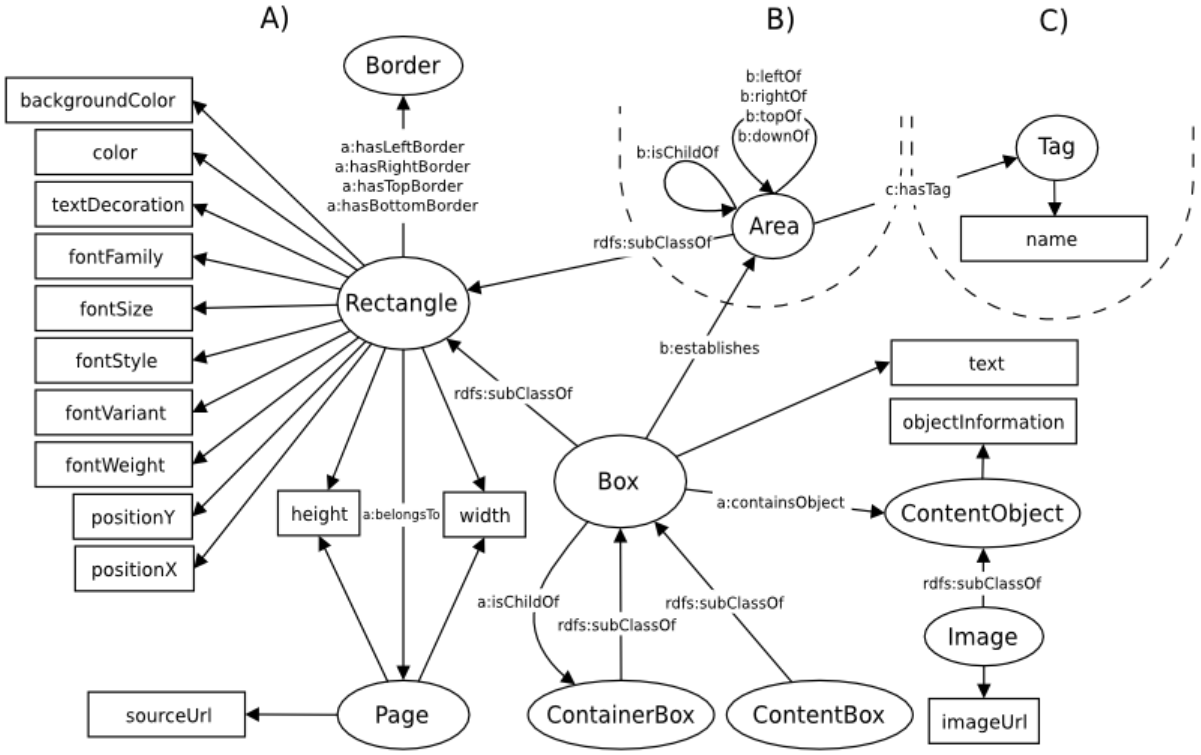
Zvažovány byly výše zmíněné algoritmy pro vizuální segmentaci webových stránek a po konečných úvahách byl zvolen algoritmus VIPS. Hlavním důvodem je skutečnost, že ačkoliv je VIPS z uvedených algoritmů nejstarší, stále je velice s oblibou používán především jako referenční algoritmus pro srovnání s novějšími nebo zcela nově vzniklými algoritmy. Dalším důvodem je fakt, že celá řada novějších algoritmů pro vizuální segmentaci webových stránek z algoritmu VIPS více či méně vychází. Z tohoto důvodu by bylo velice výhodné mít algoritmus VIPS implementován v nástroji FITLayout.

Protože však algoritmus VIPS patří mezi starší segmentační algoritmy a od jeho vzniku se webové technologie posunuly o velký kus kupředu, je třeba implementovaný algoritmus více optimalizovat pro moderní webové stránky, především poté pro stránky vytvořené v jazyce HTML5.

Výsledkem této práce je tedy nová metoda pro segmentaci webových stránek, která vychází ze základního algoritmu VIPS, avšak modifikována tak, aby dokázala segmentovat webové stránky splňující nejnovější trendy v oblasti webových technologií.

3 Nástroj FITLayout

Segmentační metoda vytvořená v rámci této práce je implementována do existujícího nástroje FITLayout vytvořeného na Fakultě informačních technologií VUT v Brně. FITLayout je rozšiřitelný rámec pro segmentaci webových stránek vytvořený v jazyce Java. Jsou zde definována obecná aplikační rozhraní pro reprezentaci renderované webové stránky a jejího členění na vizuální oblasti. Dále je v nástroji FITLayout zajištěn základ pro implementaci segmentačních algoritmů se společným aplikačním rozhraním a nalezneme zde také nástroje pro další zpracování výsledků segmentačního procesu pomocí klasifikace vizuálních nebo textových rysů jednotlivých oblastí. V neposlední řadě nástroj FITLayout poskytuje grafické uživatelské rozhraní pro pohodlné zkoumání výsledků segmentace. Základní popis aplikačního rozraní nástroje FITLayout lze vidět na obrázku obr. 6. [10]

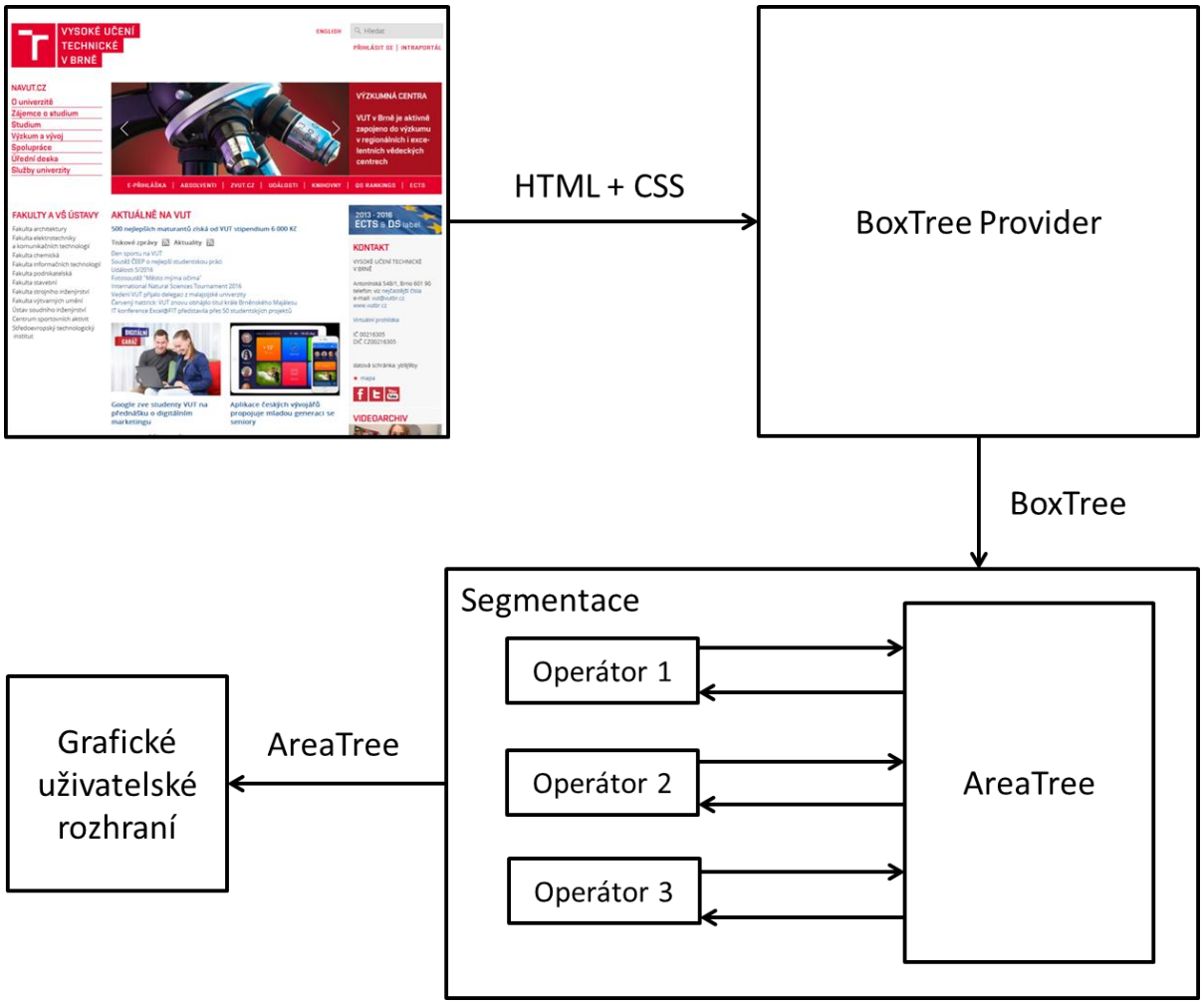


obr. 6: Základní schéma aplikačního rozhraní nástroje FITLayout [10]

Nástroj FITLayout pracuje s vyrenderovanou webovou stránkou, která je reprezentována stromem boxů („BoxTree“). Ten je získán během procesu renderování stránky určením pozice, barvy, fontů a dalších vizuálních vlastností jednotlivých úseků (boxů) obsahu webové stránky. Strom boxů poté slouží jako vstup segmentačních algoritmů.

V rámci samotného segmentačního procesu je poté strom boxů analyzován, na základě čehož je vytvořen strom vizuálních oblastí („AreaTree“), které konkrétně odpovídají detekovaným vizuálním blokům webové stránky. Na vytvořený strom oblastí poté mohou dále být aplikovány operátory reprezentující nezávislé operace provedené po původním segmentačním procesu. Tyto operace mohou změnit výslednou organizaci stromu vizuálních oblastí, například některé uzly mohou být spojeny do nově vzniklých oblastí. [10]

Segmentační metoda vytvářená v rámci této práce je implementována právě jako jeden oddělený segmentační operátor. Od původního segmentačního algoritmu je navrácen pouze strom oblastí reprezentující DOM strom vstupní webové stránky a ten je následně v rámci zmíněného operátoru zpracován a modifikován na výslednou obsahovou strukturu, která je výstupem vytvářené segmentační metody. Jednoduché schéma transformace vstupní webové stránky na výsledný strom oblastí se nachází na obr. 7.



obr. 7: Schéma transformace vstupní webové stránky na výsledný strom oblastí v nástroji FITLayout

3.1 Strom boxů z pohledu implementace

Vykreslená webová stránka je reprezentována jako instance třídy `Page`. Její metoda `getRoot` vrací kořenový uzel stromu boxů, který reprezentuje obsah webové stránky. Jednotlivé uzly stromu boxů jsou poté tvořeny instancemi třídy `Box`, která představuje jednotlivé vykreslené boxy. Každý z těchto boxů má na stránce pevnou pozici, která může být získána pomocí volání metody `getBounds`. Dále lze o boxech získat další informace, jako jsou například velikost fontu, barva atp. Potřebné metody jsou definovány v rámci rozhraní `ContentRect`.

Metoda getType vrací typ jednoho konkrétního boxu. Typ může nabývat těchto hodnot:

- ELEMENT: box vytvořený z DOM elementu.
- TEXT_CONTENT: box reprezentující zobrazený text.
- REPLACED_CONTENT: box reprezentující obrázek, nebo jiný objekt.

Boxy jsou poté organizovány do hierarchické struktury, přičemž boxy typu TEXT_CONTENT a REPLACED_CONTENT jsou vždy listovými uzly a boxy typu ELEMENT se mohou vyskytovat kdekoliv ve stromové struktuře. K navigaci v rámci této struktury slouží metody getParentBox, getChildBox a getChildCount. [10]

3.2 Strom oblastí z pohledu implementace

Jak již bylo řečeno, strom boxů je použit jako vstup segmentačního algoritmu. Výstupem je poté strom vizuálních oblastí, který je reprezentován instancí třídy AreaTree. Její metoda getRoot vrátí kořenový uzel stromu oblastí, reprezentujícího výstup segmentačního procesu. Každý uzel stromu oblastí je poté reprezentován instancí třídy Area, která přímo odpovídá jedné vizuální oblasti detekované na webové stránce.

Metody pro manipulaci a navigaci v rámci stromu oblastí jsou obdobné jako u stromu boxů. Všechny tyto metody jsou poté specifikovány v rámci rozhraní AreaTreeNode. Pozice a další vizuální vlastnosti jako fonty a barvy lze také získat stejným způsobem jako u stromu oblastí, tedy prostřednictvím implementované třídy ContentRect. Protože však jednotlivé oblasti mohou obsahovat více boxů s odlišnými vlastnostmi, vracejí metody odpovídající stromu oblastí průměrné hodnoty v rámci jedné oblasti. [10]

3.3 Moduly

Proces segmentace webové stránky může být ovládán za pomoci skupiny poskytnutých nástrojů. Mezi tyto nástroje se řadí například grafický prohlížeč s grafickým uživatelským rozhraním nebo programovatelný procesor, poskytující možnost použití JavaScriptu ke spouštění úkolů v dávkovém režimu.

FITLayout se skládá z následujících základních modulů:

- API: standardní Java rozhraní a jejich obecná implementace, která definuje obecné aplikační rozhraní pro segmentační metody.
- CSSBox bindings: standardní implementace zdroje renderované webové stránky založená na renderovacím stroji CSSBox [11].
- Segmentation: implementace základních segmentačních metod, která může být do budoucna dále rozšířena přidáním nových operátorů nad stromem oblastí.
- Tools: nástroje sloužící k ovládání segmentačního procesu zahrnující také grafický prohlížeč výsledků segmentačního procesu. [10]

3.4 Služby

Architektura nástroje FITLayout je poměrně jednoduše rozšiřitelná pomocí vytvoření přídavných modulů poskytujících novou funkcionalitu. Může se jednat například o nový segmentační algoritmus, operátor, nebo rozšíření grafického uživatelského rozhraní.

Rozeznáváme zde následující typy služeb:

- BoxTreeProvider: zdroj stromu boxů. Příkladem může být nový vykreslovač (renderer) webových stránek. Na základě vstupních parametrů (například URL) vykreslí webový dokument a poskytne strom boxů.
- AreaTreeProvider: zdroj stromu oblastí. Příkladem může být nový segmentační algoritmus. Na vstup je předán strom boxů a na výstupu je produkován strom vizuálních oblastí, který reprezentuje segmentovanou webovou stránku.
- AreaTreeOperator: operace aplikované na strom vizuálních oblastí po provedení původního segmentačního procesu.
- LogicalTreeProvider: analyzátor, který obdrží na vstupu finální strom oblastí a přiřadí jednotlivým oblastem určitý význam.

Každá služba je poté identifikována svým unikátním identifikátorem, který lze získat pomocí volání metody getId. Dále lze každé implementované službě předávat vstupní parametry na základě implementovaného rozhraní ParametrizedOperation. Pro přístup k jednotlivým službám poté nástroj FITLayout poskytuje třídu ServiceManager, která obsahuje statické metody pro lokalizování služeb zadaného typu. [10]

3.5 CSSBox

CSSBox je (X)HTML/CSS vykreslovací (renderovací) stroj, vytvořený v jazyce Java, který je v současné době používaný v nástroji FITLayout pro vykreslování segmentované webové stránky. Jeho hlavním účelem je tedy poskytovat kompletní informaci o obsahu a rozložení zobrazované webové stránky.

Vstupem renderovacího stroje je DOM strom konkrétního webového dokumentu. CSSBox je poté schopen automaticky načíst kaskádové styly odkazované v dokumentu a určit konkrétní styl každého elementu. Následně je sestaveno výsledné rozložení vykreslované stránky, které je reprezentováno pomocí stromu boxů. Každý box tvoří obdélníkovou oblast dané stránky odpovídající konkrétnímu HTML elementu. Jeden element může být v některých situacích také tvořen více boxy, například víceřádkový odstavec může být rozdělen na jednotlivé řádky. [11]

4 Algoritmus VIPS podrobně

V této kapitole si podrobně rozebereme princip segmentačního algoritmu VIPS, jehož základní vlastnosti byly v opravdu hrubých rysech nastíněny již v kapitole 2.1. Zaměříme se na detailní rozbor jednotlivých fází algoritmu a také na formální popis výsledné obsahové struktury webové stránky. Dále si popíšeme mechanizmy, pomocí kterých je v algoritmu VIPS dosahováno uspokojivých výsledků segmentačního procesu. Především si popíšeme, jakým způsobem jsou určená heuristická pravidla pro detekci výsledných vizuálních segmentů webové stránky. Dále si poté upřesníme mechanizmy, kterými můžeme ovlivňovat jemnost segmentačního procesu nebo postupy pro určení váhy jednotlivých vizuálních oddělovačů. Všechny výše zmíněné mechanizmy výrazným způsobem ovlivňují výsledek segmentačního procesu algoritmu VIPS, a proto je zapotřebí jejich podrobná definice.

Výsledkem segmentačního procesu algoritmu VIPS je hierarchická obsahová struktura, ve které každý uzel odpovídá jednomu vizuálnímu bloku webové stránky. Jak již bylo řečeno, hlavní myšlenkou algoritmu VIPS je snaha na segmentovaný dokument nahlížet stejně, jako na něj nahlíží jeho uživatel, tedy pomocí internetového prohlížeče. Z tohoto důvodu je výsledná obsahová struktura odvozená z kombinace DOM stromu a vizuálních vlastností jednotlivých prvků. Na základě těchto informací, získaných z internetového prohlížeče, algoritmus VIPS analyzuje webovou stránku a provádí extrakci jednotlivých vizuálních segmentů a následné sestavení výsledné obsahové struktury, jejíž formální definice se nachází v kapitole 4.1.

4.1 Výsledná obsahová struktura webové stránky

Pro definování obsahové struktury webové stránky, jež se snažíme v rámci vizuální segmentace docílit, je třeba nejprve definovat následující základní pojmy [1]:

Definice 1:

Základní objekt je listový uzel DOM stromu, který již nemůže být dále rozdělen na menší části.

Definice 2:

Blok je množina obsahující jeden nebo více základních objektů.

Struktura DOM stromu sice poskytuje hierarchii pro základní objekty na webové stránce, avšak tato hierarchie slouží spíše pro reprezentaci obsahu, než pro jeho organizaci. Z tohoto důvodu je třeba definovat obsahovou strukturu založenou na vzhledu webové stránky. Platí, že každý uzel této struktury je blokem. Také je velmi důležité poznamenat, že uzly obsahové struktury nemusí nutně odpovídat uzlům v DOM stromu. [1]

Definice 3:

Webová stránka Ω je trojice:
$\Omega = (O, \Phi, \delta)$, kde

- $O = \{\Omega^1, \Omega^2, \dots, \Omega^N\}$ je konečná množina všech bloků, které se vzájemně nepřekrývají. Na každý blok lze poté rekurzivně nahlížet, jako na podstránku webové stránky Ω, která má svou vlastní obsahovou strukturu.
- $\Phi = \{\varphi^1, \varphi^2, \dots, \varphi^T\}$ je konečná množina vizuálních oddělovačů, obsahující vertikální i horizontální oddělovače. Dále platí, že každý oddělovač má svou vlastní váhu indikující jeho viditelnost na webové stránce a všechny oddělovače ve stejné množině Φ mají totožnou váhu.
- δ vyjadřuje vztah každých dvou bloků z O, jedná se tedy o relaci: $\delta = O \times O \rightarrow (\Phi \cup \emptyset)$.

Nechť Ω^i a Ω^j jsou dva bloky z O, poté $\delta(\Omega^i, \Omega^j) \neq \emptyset$ značí, že Ω^i a Ω^j jsou odděleny oddělovačem $\delta(\Omega^i, \Omega^j)$. [2]

Vzhledem k tomu, že každý blok $\Omega^i, i \leq N$ je zároveň podstránkou webové stránky Ω, rekurzivně můžeme definovat:

Definice 4:

Podstránka webové stránky Ω je trojice:
$\Omega_s^n = (O_s^n, \Phi_s^n, \delta_s^n)$, kde

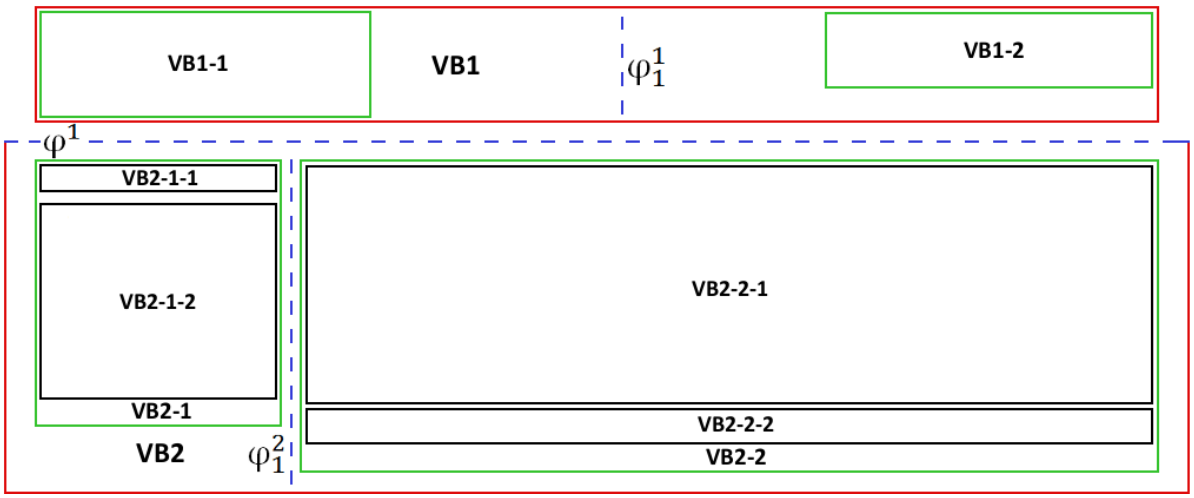
- $O_s^n = \{\Omega_{sn}^1, \Omega_{sn}^2, \dots, \Omega_{sn}^{N_{sn}}\}$
- $\Phi_s^n = \{\varphi_{sn}^1, \varphi_{sn}^2, \dots, \varphi_{sn}^{T_{sn}}\}$
- $\delta_s^n = O_s^n \times O_s^n \rightarrow (\Phi_s^n \cup \emptyset)$

a zároveň platí, že:

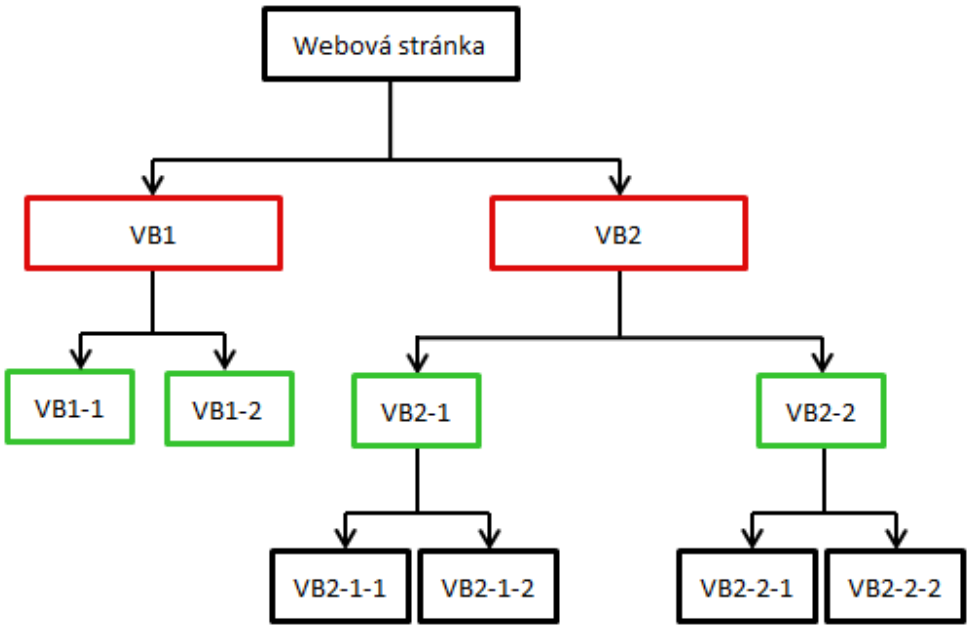
- Ω_s^n je n-tý blok podstránky úrovně s
- N_{sn} je počet bloků v O_s^n
- T_{sn} je počet oddělovačů v Φ_s^n [2]

Na obr. 8 je zobrazena vizuální reprezentace ukázkové webové stránky, jedná se o část webové stránky Vysokého učení technického v Brně. Vizuální znázornění a stromová reprezentace výsledné obsahové struktury této webové stránky se poté nachází na obr. 9, respektive obr. 10. Pro zjednodušení se jedná pouze o ukázkovou obsahovou strukturu, která znázorňuje členění webové stránky na vizuální bloky a jejich vzájemné provázání pomocí vizuálních oddělovačů. Při skutečné segmentaci by listové uzly této obsahové struktury mohly být ještě dále rozděleny na menší části v závislosti na zvolené jemnosti segmentačního procesu. Zobrazená obsahová struktura je tedy na první úrovni rozdělena na dva vizuální bloky VB1 a VB2, které jsou odděleny jedním vizuálním oddělovačem φ^1. Dále lze z obrázků pozorovat, že každý vizuální blok první úrovně je v tomto případě zároveň i podstránkou původní webové stránky, neboť v sobě má obsaženy další vizuální bloky.

obr. 8: Vizuální reprezentace webové stránky VUT v Brně



obr. 9: Vizuální znázornění obsahové struktury webové stránky VUT v Brně



obr. 10: Stromová reprezentace obsahové struktury webové stránky VUT v Brně

Formální vyjádření obsahové struktury ukázkové webové stránky Ω a jejích dvou podstránek VB1 a VB2 poté vypadá následovně:

$\Omega = (O, \Phi, \delta)$, kde	VB1 = $\Omega_1^1 = (O_1^1, \Phi_1^1, \delta_1^1)$, kde	VB2 = $\Omega_1^2 = (O_1^2, \Phi_1^2, \delta_1^2)$, kde
$O = \{VB1, VB2\}$	$O_1^1 = \{VB1\text{-}1, VB1\text{-}2\}$	$O_1^2 = \{VB2\text{-}1, VB2\text{-}2\}$
$\Phi = \{\varphi^1\}$	$\Phi_1^1 = \{\varphi_1^1\}$	$\Phi_1^2 = \{\varphi_1^2\}$
$\delta = \binom{(VB1,VB2)}{jinak} = \binom{\varphi^1}{\emptyset}$	$\delta_1^1 = \binom{(VB1-1,VB1-2)}{jinak} = \binom{\varphi_1^1}{\emptyset}$	$\delta_1^1 = \binom{(VB2-1,VB2-2)}{jinak} = \binom{\varphi_1^2}{\emptyset}$

[2]

4.2 Extrakce vizuálních bloků

V této fázi algoritmu VIPS je DOM strom procházen od kořenového uzlu za účelem získání všech vizuálních bloků. Zdali blok je, nebo není vizuální, je posouzeno na základě informací poskytnutých internetovým prohlížečem. Pokud uzel není označen jako vizuální, je dále rozdělen a nahrazen svými synovskými uzly, které jsou dále zpracovávány stejným způsobem. V podstatě bychom mohli každý uzel DOM stromu označit jako vizuální blok, neboť se dá předpokládat, že bude na webové stránce viditelný. Avšak některé uzly DOM stromu slouží pouze k organizačním účelům a z pohledu segmentace je tedy není vhodné označovat jako samostatný vizuální blok. Příkladem takového uzlu může být tabulka, která sice na výsledné stránce je zobrazená, ale z hlediska segmentace jsou hledané vizuální bloky umístěny v jednotlivých buňkách tabulky a ne v tabulce samotné. Podíváme-li se však na tento problém z druhé strany, není příliš vhodné označit jako vizuální bloky všechny listové uzly DOM stromu z důvodu jejich velkého množství. Je třeba najít optimální kompromis mezi těmito dvěma přístupy. [5]

Jedním z mechanizmů, jak zvolit správnou úroveň segmentace, je stanovení hodnoty stupně koherence (*Degree of Coherence - DoC*). Tato hodnota určuje, jak je daný vizuální blok koherentní, neboli jak moc je jeho vnitřní struktura rozdílná. Dále má hodnota DoC definovány následující vlastnosti:

- Rozsah hodnoty DoC je v intervalu od 0 do 1.
- Hodnota DoC je tím vyšší, čím konzistentnější je obsah uvnitř vizuálního bloku.
- Ve výsledné hierarchické struktuře musí platit, že DoC rodičovského bloku je nižší, než DoC jeho potomků. [1]

Zároveň potřebujeme definovat hodnotu podmíněného stupně koherence (*Permitted Degree of Coherence - PDoC*). S využitím této hodnoty má uživatel algoritmu možnost ovlivnit jemnost segmentačního procesu. Hodnota PDoC má dále definovány následující vlastnosti:

- Rozsah hodnoty PDoC je v intervalu od 0 do 1.
- Podmínka zrnitosti: Ve výsledné hierarchické struktuře musí platit, že DoC všech listových bloků je menší, než hodnota PDoC.

Z výše uvedeného tedy vyplývá, že čím vyšší hodnota PDoC je, tím bude výsledek segmentačního procesu jemnější, neboť listové bloky, které v první iteraci algoritmu nebudou splňovat podmínku zrnitosti, budou znovu předloženy algoritmu a rozděleny na menší bloky.

Pokud je nalezen vizuální blok, je přidán do seznamu všech vizuálních bloků a je nastavena odpovídající hodnota DoC na základě jeho vnitřních rozdílů. O tom, zdali blok je označen jako vizuální, nebo je třeba ho dále rozdělit a zpracovávat jeho synovské bloky, je rozhodnuto na základě heuristických pravidel popsaných níže v podkapitole 4.2.2. [1]

4.2.1 Algoritmus extrakce vizuálních bloků

Pseudokód extrakce vizuálních bloků zajišťuje funkce DivideDomTree, spolu s ostatními funkcemi popsanými v Algoritmu 1.

Algoritmus 1: Extrakce vizuálních bloků

```
1:  DivideDomTree(root) {
2:      if (Dividable(root) == true) {
3:          for each child of root {
4:              DivideDomTree(child);
5:          }
6:      }
7:      else {
8:          put root to listOfVisualBlocks;
9:          set DoC for root;
10:     }
11: }
12:
13: Dividable(root) {
14:     if (root is the top block) {
15:         return true;
16:     }
17:     else {
18:         return not IsVisualBlock(root);
19:     }
20: }
21:
22: IsVisualBlock(root) {
23:     apply heuristic rules to root;
24: }
```

4.2.2 Heuristická pravidla

Abychom mohli uvést heuristická pravidla algoritmu VIPS, je zapotřebí nejprve nadefinovat některé pojmy [2]:

Definice 5:

Validní uzel je uzel DOM stromu, který je v internetovém prohlížeči viditelný, tedy jeho šířka i výška jsou rozdílné od nuly.

Definice 6:

Blokový uzel je uzel DOM stromu s HTML značkou rozdílnou od <A>, <B>, <FONT>, <HR>, <I>, <P>, <STRONG>, <TEXT>.

Definice 7:

Textový uzel je uzel DOM stromu, který obsahuje pouze osamocený text.

Definice 8:

Virtuální textový uzel je takový uzel DOM stromu, který není blokovým uzlem a zároveň jsou jeho potomky pouze textové uzly.

Na základě výše uvedených definic můžeme stanovit následující heuristická pravidla pro uzly DOM stromu:

- Pokud uzel neobsahuje žádné validní uzly jako své potomky, nebude již dále zpracováván a bude zahozen.
- Pokud uzel obsahuje pouze jeden validní uzel mezi svými potomky a tento potomek není textovým uzlem, je tento potomek dále zpracováván místo původního uzlu.
- Pokud jsou všichni potomci uzlu buďto textové uzly, nebo virtuální textové uzly, poté nastav hodnotu DoC tohoto uzlu na 1.
- Pokud je velikost uzlu alespoň třikrát větší než celková velikost všech jeho potomků, uzel bude rozdělen.
- Pokud uzel obsahuje jako potomka textový uzel nebo virtuální textový uzel a zároveň je výška nebo šířka uzlu menší než nadefinovaná prahová hodnota, nastav hodnotu DoC uzlu na 0,8.
- Uzel bude rozdělen, pokud obsahuje jako své potomky dva po sobě následující uzly s HTML značkou
.

Dále je třeba stanovit speciální obslužné rutiny pro uzly DOM stromu s HTML značkami <TABLE>, <TBODY>, <TR>, <TD> a <P>. Jedná se o značky, které jsou na webových stránkách velice hojně používány, a proto u nich lze s vysokou pravděpodobností předpokládat, že budou tvořit obsahově spojitou strukturu. Z tohoto důvodu je třeba uzly DOM stromu s těmito značkami zpracovávat specifickým způsobem a stanovit pro ně vyšší prahové hodnoty ve výše stanovených heuristických pravidlech. [2]

4.3 Detekce vizuálních oddělovačů

Po dokončení fáze extrakce vizuálních bloků jsou všechny nalezené vizuální bloky umístěny v jedné společné datové struktuře typu jednosměrně vázaný lineární seznam. Následně je na základě těchto nalezených bloků zahájena fáze detekce vizuálních oddělovačů. Vizuální oddělovače jsou horizontální nebo vertikální viditelné linie webové stránky, které se nepřekrývají ani nekříží s žádným vizuálním blokem. Jedná se tedy o velice vhodné indikátory pro odlišení významově odlišných částí webové stránky. Nyní si vizuální oddělovač definujeme formálně. [1]

Definice 9:

Vizuální oddělovač je dvojice:
$S = (Ps, Pe)$, kde

- Ps je startovní pixel oddělovače.
- Pe je koncový pixel oddělovače.
- Šířka oddělovače je spočítána jako rozdíl hodnot Ps a Pe.

4.3.1 Algoritmus detekce vizuálních oddělovačů

Samotný algoritmus detekce vizuálních oddělovačů je strukturován následovně:

1) V prvním části algoritmu dojde k inicializaci seznamu všech oddělovačů. Do seznamu je umístěn pouze jeden oddělovač, jehož startovní a koncový pixel odpovídají okrajům webové stránky.
2) V druhé fázi algoritmu je pro každý vizuální blok vyhodnocen jeho vztah se všemi oddělovači a dále je provedena akce dle následujících pravidel:
   - Pokud je vizuální blok celý obsažen uvnitř oddělovače, dojde k rozdělení oddělovače.
   - Pokud se počáteční nebo koncová část vizuálního bloku nachází uvnitř oddělovače, dojde k úpravě šířka oddělovače.
   - Pokud se vizuální blok s oddělovačem navzájem překrývají jinou než počáteční nebo koncovou částí, dojde k odstranění oddělovače.
3) V poslední fázi algoritmu jsou odstraněny oddělovače, které jsou umístěny na okraji webové stránky, a tudíž neoddělují žádné významově odlišné části. [5]

Pro zjednodušení zde uvažujeme detekci vizuálních oddělovačů jen v jednom směru, ve skutečnosti ale musíme tento algoritmus vykonat dvakrát, poprvé pro horizontální oddělovače a podruhé pro vertikální oddělovače.

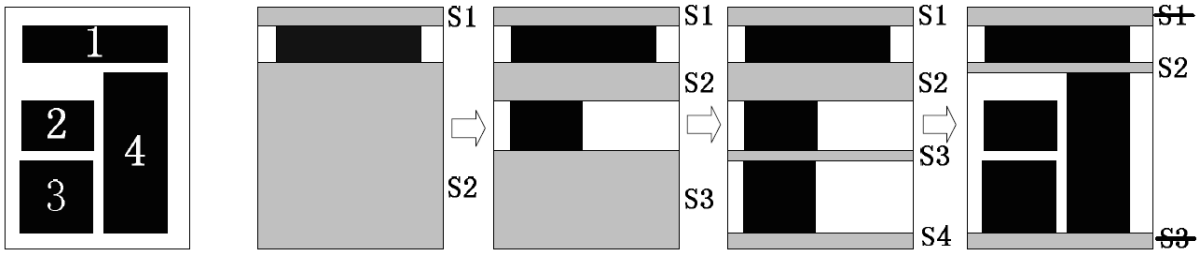
Struktura algoritmu pro detekci vizuálních separátorů v jednom směru se nachází ve funkci separatorsDetection(), které je popsána v Algoritmu 2.

Algoritmus 2: Detekce vizuálních oddělovačů

```
1:  separatorsDetection() {
2:      inicialize visualSeparatorsList;
3:      for-each visualBlock from visualBlocksList {
4:          for-each separator from visualSeparatorsList {
5:              apply relation-rules to separator;
6:          }
7:      }
8:      remove bounding-separators from visualSeparatosList;
9:
10:     for-each separator from visualSeparatorsList {
11:         apply weight-settings-rules to separator;
12:     }
13: }
```

Nyní si předvedeme, jak algoritmus funguje na praktické ukázce. Vezměme obr. 11 jako příklad, ve kterém černé oblasti reprezentují vizuální bloky na webové stránce. Na počátku máme pouze jeden oddělovač, jehož startovní a koncový pixel korespondují s okraji webové stránky. Při zpracování prvního vizuálního bloku je původní oddělovač rozdělen na dva nové oddělovače S1 a S2. Identicky postupujeme v případě druhého i třetího bloku. Při zpracování čtvrtého bloku dojde k situaci, kdy se počátek bloku nachází uvnitř oddělovače S2 a zároveň se blok překrývá s oddělovačem S3. Z tohoto důvodu je upravena šířka oddělovače S2 a je odstraněn oddělovač S3.

Na konci tohoto procesu jsou ještě odstraněny oddělovače S1 a S3, které jsou umístěny na okraji webové stránky a tedy nejsou z pohledu segmentace nikterak významné. [2]



obr. 11: Ukázka algoritmu detekce vizuálních oddělovačů [2]

4.3.2 Nastavení váhy oddělovačů

Po nalezení všech vizuálních oddělovačů je třeba nastavit jejich váhu. Jedná se o hodnotu, která je určena s ohledem na významové rozdíly mezi vizuálními bloky, kterými je daný oddělovač obklopen. Za tímto účelem jsou použity následující pravidla:

- Čím větší je vzdálenost mezi vizuálními bloky na opačných stranách vizuálního oddělovače, tím vyšší bude váha tohoto oddělovače.
- Pokud se vizuální oddělovač nachází na stejné pozici jako nějaký uzel DOM stromu představující viditelné vizuální rozdělení stránky, je váha tohoto oddělovače zvýšena. Příkladem může být uzel s HTML značkou `<HR>` představující vodorovné rozdělení prostřednictvím linky.
- Čím výraznější jsou rozdíly ve vlastnostech fontu písma ve vizuálních blocích obklopujících daný oddělovač, tím více se zvýší váha oddělovače.
- Je-li velikost fontu písma ve vizuálním bloku před oddělovačem menší než velikost fontu písma ve vizuálním bloku za oddělovačem, je zvýšena váha daného oddělovače.
- Pokud vizuální bloky obklopující oddělovače mají rozlišnou barvu pozadí, je váha daného oddělovače zvýšena.
- Je-li struktura vizuálních bloků obklopujících oddělovač velmi podobná, je váha daného oddělovače snížena. Jedná se například o situaci, kdy oba zkoumané vizuální bloky představují prostý text. [2]

4.4 Konstrukce obsahové struktury

Po dokončení fáze detekce vizuálních oddělovačů přechází algoritmus VIPS do poslední fáze s názvem konstrukce obsahové struktury. V této fázi je sestavena výsledná obsahová struktura segmentované webové stránky odpovídající definici z kapitoly 4.1. Konstrukce výsledné obsahové struktury započne od vizuálního oddělovače s nejnižší vahou. Vizuální bloky obklopující tento oddělovač jsou spojeny v jeden nový blok a konstrukce obsahové struktury pokračuje dalším oddělovačem v pořadí. Tímto způsobem je od listových uzlů směrem nahoru budována nová stromová struktura. Celý proces probíhá až do okamžiku, kdy je zpracován i oddělovač s nejvyšší vahou a konstrukce stromové struktury je tak dokončena. Pro každý nově vzniklý blok je třeba nastavit odpovídající hodnotu DoC na základě rozdílů v jeho vnitřní struktuře stejným způsobem, jako tomu bylo u nově nalezených vizuálních bloků v kapitole 4.2.

Po dokončení procesu konstrukce je každý listový uzel nově vzniklé obsahové struktury zkontrolován, zdali splňuje podmínku zrnitosti, tedy jestli platí, že hodnota stupně koherence daného listového uzlu je větší než hodnota podmíněného stupně koherence segmentačního procesu.

V případě, že některý z listových uzlů podmínku zrnitosti nesplní, je znovu předán na vstup algoritmu VIPS, tedy do fáze extrakce vizuálních bloků, avšak tentokrát jako kořenový uzel. Vzhledem k tomu, že dle algoritmu 1 je každý uzel nejvyšší úrovně považován za nevizuální, a tudíž je rovnou rozdělen a nahrazen svými synovskými uzly, je zřejmé, že listový uzel obsahové struktury, který v předchozí iteraci algoritmu nesplnil podmínku zrnitosti, bude nyní rozčleněn na menší části. Tento proces iterativně probíhá až do okamžiku, kdy podmínku zrnitosti splňují všechny listové uzly a je tak získána výsledná obsahová struktura segmentované webové stránky. [2][5]

Pseudokód funkce ContentStructureConstruction zajišťující konstrukci výsledné obsahové struktury je popsán v rámci algoritmu 3.

Algoritmus 3: Konstrukce obsahové struktury

```
1:  ContentStructureConstruction(detectedSeparators) {
2:      sort detectedSeparators ascending by weight;
3:      for-each separator from detectedSeparators {
4:          newBlock := merge separator's surrounding visual blocks;
5:          add newBlock to contentStructure;
6:      }
7:      for-each leafNode of contentStructure {
8:          if( DoC value of leafNode < PDoC value ) {
9:              divideDomTree(leafNode);
10:         }
11:     }
12: }
```

5 Omezení algoritmu VIPS

Jak již bylo řečeno, algoritmus VIPS kombinuje strukturu DOM stromu s vizuálními vlastnostmi segmentované webové stránky k získání optimálního výsledku segmentačního procesu. Vzhledem k této skutečnosti lze algoritmus VIPS označit za velmi efektivní metodu segmentace webových stránek, neboť získaný výsledek je velice blízký lidskému vnímání rozdělení webové stránky na významově odlišné oblasti. Přesto se však u této metody setkáváme s několika výraznými nedostatky, kterým se budeme věnovat v následujícím textu.

5.1 Známé problémy

Jednou z hlavních nevýhod algoritmu VIPS je skutečnost, že množina heuristických pravidel použitých ve fázi extrakce vizuálních bloků nemá striktně definované pořadí, ve kterém je třeba jednotlivá heuristická pravidla aplikovat. Dále některá heuristická pravidla používají prahovou hodnotu k rozhodování, zdali je třeba zpracovávaný uzel DOM stromu označit jako vizuální, nebo jej dále rozdělit. Vlastnosti prahové hodnoty také nejsou pevně definované, tudíž zvolení vhodné hodnoty hodně závisí na rozhodnutí tvůrce algoritmu, na vlastnostech konkrétní webové stránky a na poznatcích získaných testováním na reálných webových dokumentech.

Další výraznou slabinou přístupu k segmentaci webových stránek založeném na algoritmu VIPS je množina HTML značek, využívaná při definování vlastností některých heuristických pravidel. Tato množina byla stanovena v době vzniku algoritmu VIPS a z pohledu dnešních možností jazyka HTML je velice úzká. Především zde nejsou zahrnuty žádné značky z HTML5, což může výrazným způsobem ovlivnit výsledek segmentačního procesu.

Dále se můžeme setkat s nekorektním použitím některých elementů jazyka HTML při tvorbě webového dokumentu. Kupříkladu element <TABLE> se při správném použití skládá z elementů <TR>, které se skládají z elementů <TD>. Pokud by však někdo zanořil element <TD> přímo do elementu <TABLE>, výsledkem by byla reprezentace, která by neobsahovala zalomení řádku, a tudíž by z hlediska algoritmu VIPS nebyla korektně zpracována rutinou pro segmentaci tabulky.

Také je třeba rozlišovat elementy, které jsou označeny jako elementy produkující zalomení řádku (dále jen line-break elementy) a které mají zároveň vnější ohraničení (margin) větší než ostatní line-break elementy. V původní definici algoritmu VIPS takové elementy nejsou nijak rozlišeny, neboť jako line-break elementy jsou označeny všechny elementy, které nejsou řádkové (inline).

Moderní webové stránky také velice často obsahují dynamické prvky. Takové prvky často svůj obsah mění za běhu, nebo až při interakci s uživatelem, avšak pokaždé bez nutnosti přenačtení celé webové stránky. Problémem je, že dynamické prvky jsou většinou neviditelné při prvotním načtení webového dokumentu a viditelnými se stanou až s příchodem nějaké události. Také existuje možnost, že nový obsah bude vložen do existujícího elementu za pomoci volání AJAX[7]. Vzhledem k tomu, že jsou takové elementy při prvotním načtení webové stránky neviditelné, jsou brány jako nevalidní bloky a je velice obtížné je analyzovat na základě DOM stromu a vizuálních vlastností získaných z webového prohlížeče v okamžiku načtení dokumentu. Jak vyplývá z předchozího textu, algoritmus VIPS není příliš ideální pro segmentaci a analýzu webových stránek obsahujících větší množství dynamického obsahu, avšak je možné alespoň učinit jistá přizpůsobení zmírňující negativní dopad dynamických elementů na výsledek segmentačního procesu. [12][13]

[7] AJAX - http://www.w3schools.com/ajax/

5.2 Možnosti optimalizace

Základní možností optimalizace algoritmu VIPS pro zpracování moderních webových stránek je rozšíření množiny HTML značek použité při definici heuristických pravidel o značky, které se objevily s příchodem HTML5.

Dále je třeba odlišit jednotlivé line-break značky podle velikosti jejich vnějšího okraje. Například element s HTML značkou <P> kolem sebe ve výchozím nastavení produkuje větší prostor v rámci svého vnějšího okraje než element se značkou <DIV> . V rámci implementace této práce je tak učiněno ve fázi detekce vizuálních oddělovačů, jelikož oddělovačům, ke kterým přiléhají line-break elementy s větším vnějším okrajem, bude nastavena vyšší váha. Vzhledem k tomu bude jejich zakomponování do výsledné obsahové struktury odlišeno od zakomponování line-break elementů s menší velikostí vnějšího okraje.

Ve fázi extrakce vizuálních bloků jsou v originálním znění algoritmu VIPS použity vizuální vlastnosti velikost a váha fontu a barva pozadí elementu pro definování některých heuristických pravidel. Pokud bychom chtěli více dbát na detekování samostatných vizuálních bloků, které jsou od ostatních bloků odděleny větším prázdným prostorem, nabízí se možnost zahrnout mezi použité vizuální vlastnosti kromě výše zmíněného „margin“ také CSS vlastnosti „padding“ (vnitřní ohraničení elementu) a „float“ (obtékání prvku). Dohromady jsou tyto CSS vlastnosti často používány k oddělení významově odlišných typů obsahu webové stránky pomocí nadpisů, odstavců, atd.

V rámci Definice 5 byl představen pojem validní uzel. Pro rekapitulaci se jedná o takový uzel, který je na webové stránce viditelný, neboť je jeho šířka i výška rozdílná od nuly. Otázka viditelnosti nějakého prvku webové stránky je však ve skutečnosti o něco složitější. Korektním způsobem pro zneviditelnění prvku webové stránky je například nastavení CSS vlastnosti „display“ na hodnotu „none“, nebo vlastnosti „visibility“ na hodnotu „hidden“. Na neštěstí je neviditelnosti prvků na některých webových stránkách dosahováno například pomocí nastavení absolutní pozice elementu mimo okraje stránky. Dalším nekorektním případem může být nastavení CSS vlastnosti „text-indent“ (odsazení prvního řádku odstavce) na zápornou hodnotu. V takovém případě může být výsledek stejný jako u předchozí situace, tedy prvek se ocitne mimo viditelnou část webové stránky. Z praktického hlediska se jedná o neviditelný prvek i v případě, kdy je barva prvku stejná jako barva jeho pozadí. Z výše uvedeného je zřejmé, že oproti původní definici nevalidního uzlu může řešení zahrnující ošetření popsaných situací také vést k jisté optimalizaci výsledku segmentačního procesu.

Z hlediska optimalizace segmentace webových stránek obsahujících dynamické elementy můžeme pozměnit první heuristické pravidlo, používané ve fázi extrakce vizuálních bloků. V původním znění toto pravidlo říká, že veškeré nevalidní bloky mají být odstraněny, a tudíž se neobjeví ve výsledné obsahové struktuře. Je však třeba poznamenat, že nevalidní bloky, které obsahují nějaké validní potomky, by měly být zachovány, neboť se mohou objevit ve výsledném rozložení stránky po příchodu patřičné události. [12]

5.3 Zhodnocení nedostatků

V této kapitole jsme si obecně představili základní nedostatky algoritmu VIPS. Jak vidno, některé problémy vznikly v průběhu času vlivem neustálé modernizace webových technologií, jiné jsou závislé na možnostech konkrétní implementace a ostatní lze zkrátka označit za slabé články VIPS algoritmu. Také jsme si nastínili možná řešení některých těchto zásadních nedostatků. Je zřejmé, že jeden samotný segmentační algoritmus nemůže mít vynikající výsledky na všech typech webových stránek. Ze samotného principu algoritmu VIPS lze vypozorovat, že sám o sobě pravděpodobně nikdy

nebude dosahovat zcela uspokojivých výsledků při segmentaci webových dokumentů obsahujících velké množství prvků s dynamickým obsahem. K tomuto účelu by pravděpodobně bylo velice vhodné segmentaci algoritmem VIPS kombinovat s jiným algoritmem, speciálně zaměřeným na segmentaci dynamického obsahu. V následujících kapitolách se podrobněji zaměříme na implementaci segmentační metody založené na algoritmu VIPS a na problémy, které bylo třeba vyřešit v rámci přizpůsobení této metody tak, aby ji bylo možné provozovat jako oddělený modul nástroje FITLayout. Dále si u výše zmíněných nedostatků popíšeme konkrétní způsoby jejich řešení, pokud tedy takové řešení je v podmínkách určených nástrojem FITLayout možné. V opačném případě nastíníme, jakým způsobem by se daná optimalizace dala provést například s využitím jiného vyreslovacího stroje.

6 Implementace segmentační metody

V této kapitole se budeme zabývat praktickým popisem implementovaného algoritmu, jehož teoretickou strukturu jsme si podrobně představili v kapitole 3. Především se pozastavíme u konkrétních problémů, které bylo třeba v rámci implementace vyřešit. Velká část těchto potíží byla způsobena z důvodu integrace segmentační metody do nástroje FITLayout. Jednotlivé problémy si tedy podrobně rozebereme a názorně si předvedeme způsoby jejich řešení a přizpůsobení podmínkám nástroje FITLayout. Dalším důležitým milníkem této kapitoly bude popis heuristických pravidel použitých ve fázi extrakce vizuálních bloků. V rámci tohoto popisu si také představíme pravidla, která byla modifikována za účelem optimalizace některých nedostatků algoritmu VIPS popsaných v kapitole 5.

6.1 Struktura modulu

Segmentační metoda tvořená v rámci této práce je implementována v programovacím jazyce Java jako oddělený modul využívající existující třídy nástroje FITLayout. Celý Java projekt byl pojmenován „vips-based-algorithm“ a jeho zdrojovou část umístěnou v balíku s názvem „org.fit.vips“ tvoří následující třídy:

- VipsBasedOperator.java
- VipsBasedSeparator.java
- VipsBasedSeparatorSet.java
- VipsBasedVisualBlock.java
- BlockBrowserVips.java

VipsBasedOperator je hlavní třídou algoritmu, která řídí celý proces segmentace a využívá pro svou činnost ostatní zmíněné třídy.

Pomocí třídy VipsBasedSeparator lze vytvářet instance vizuálních oddělovačů používaných v průběhu segmentačního procesu. Tato třída dědí vlastnosti třídy Separator z balíku org.fit.segm.grouping.op, avšak reimplementuje metodu getWeight, sloužící pro stanovení váhy jednotlivých oddělovačů. Reimplementace této metody je zapotřebí z důvodu splnění definice z kapitoly 4.3.2 o stanovení váhy vizuálních oddělovačů v algoritmu VIPS.

Třída VipsBasedSeparatorSet představuje algoritmus pro detekci vizuálních oddělovačů. Dědí své vlastnosti od třídy SeparatorSetHVS z balíku org.fit.segm.grouping.op, a tudíž pro svou činnost využívá algoritmu pro detekci vizuálních oddělovačů obsaženého v nástroji FITLayout [16]. O důvodech tohoto využití se dozvíme více v následujícím textu. Třída VipsBasedSeparator dále reimplementuje metodu filterSeparators. Tato metoda určuje, které z detekovaných vizuálních oddělovačů mají být odfiltrovány, například z důvodu jejich malé šířky. Pro činnost zde představené segmentační metody je nezbytné zachovat všechny detekované oddělovače.

Pomocí třídy VipsBasedVisualBlock lze vytvářet instance extrahovaných vizuálních bloků. Každá nová instance poté obsahuje konkrétní podstrom webové stránky, který byl označen jako vizuální blok spolu s dalšími údaji, jako jsou vizuální vlastnosti nebo stupeň koherence daného bloku.

Třída BlockBrowserVips slouží pouze pro spuštění nástroje FITLayout s integrovaným operátorem implementujícím zde popisovanou segmentační metodu.

6.2 Proces segmentace

Nyní si popíšeme, jak z implementačního hlediska probíhá samotný proces segmentace, budeme se tedy zabývat převážně třídou VipsBasedOperator a pokud nebude uvedeno jinak, všechny zmíněné metody budou pocházet právě z této třídy.

Segmentační proces se spouští pomocí metody apply, která na vstupu očekává datovou strukturu AreaTree reprezentující DOM strom vstupní webové stránky. V rámci metody apply je dále volána metoda performVipsAlgorithm spouštějící jednotlivé fáze segmentačního procesu založeného na algoritmu VIPS.

6.2.1 Detekce vizuálních oddělovačů

V původní definici algoritmu VIPS probíhá fáze detekce vizuálních oddělovačů až po fázi extrakce vizuálních bloků. V implementaci této segmentační metody však byl z důvodu optimalizace použit algoritmus detekce vizuálních oddělovačů obsažený v nástroji FITLayout, neboť funguje na obdobném principu, jako algoritmus definovaný v rámci kapitoly 4.3.1, avšak již řeší řadu problémů, které v původní definici nebyly vůbec zmíněny. Navíc je tento algoritmus daleko výhodnější použít z hlediska lepší adaptace vytvářené segmentační metody na renderovací jádro CSSBox, používané v rámci FITLayoutu. Více informací o použitém algoritmu lze nalézt v [16].

Z důvodu použití jiného algoritmu je třeba fázi detekce vizuálních oddělovačů spustit jako první, neboť tímto algoritmem nelze oddělovače detekovat mezi extrahovanými vizuálními bloky a je třeba detekci provést nad všemi prvky vstupní webové stránky.

Fáze detekce vizuálních oddělovačů se tedy spustí voláním metody collectSeparators, která rekurzivně projde celý vstupní strom oblastí a pro každý element detekuje oddělovače na aktuální úrovni zanoření.

6.2.2 Extrakce vizuálních bloků

Fáze extrakce vizuálních bloků započne voláním metody divideDomTree, která začne opět rekurzivně procházet vstupní strom oblastí, přičemž u každého uzlu dojde k posouzení, zdali se jedná o vizuální blok na základě heuristických pravidel.

V případě pozitivního nálezu vizuálního bloku je dotyčný uzel stromu oblastí přidán do seznamu všech vizuálních bloků pomocí volání metody createNewVisualBlock a jeho potomci již nejsou dále zpracováváni. Novému vizuálnímu bloku je také v okamžiku jeho identifikace spočítán a přiřazen stupeň koherence pomocí metody docEvaluation. Jak již bylo dříve řečeno, stupeň koherence je určen na základě vnitřních vizuálních rozdílů daného vizuálního bloku. Metoda docEvaluation tedy určí hodnotu stupně koherence pomocí vizuálních vlastností daného bloku, jako jsou barva pozadí, velikost fontu, váha fontu, styl a další vlastnosti fontu (obyčejné písmo, kurzíva, podtržení, přeškrtnutí).

V případě, kdy uzel není shledán jako vizuální, dojde naopak k jeho rozdělení a nahrazení jeho potomky. Potomci jsou poté zpracováváni stejným způsobem. S případem rozdělení aktuálně zpracovávaného uzlu také souvisí problém rekonfigurace oddělovačů.

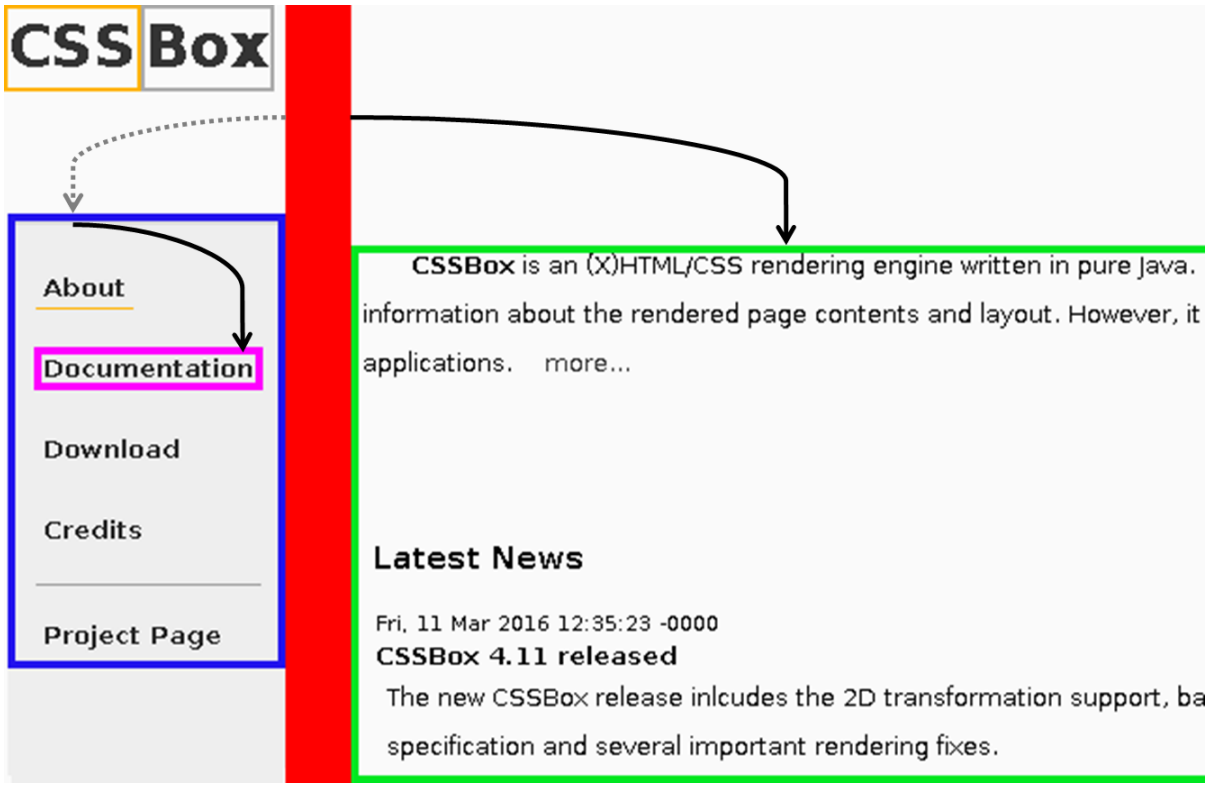
6.2.3 Rekonfigurace vizuálních oddělovačů

V průběhu implementace bylo zjištěno, že v situaci, kdy je aktuálně zpracovávaný uzel vstupního stromu oblastí rozdělen a je tak nahrazen svými potomky, dojde zároveň ke ztrátě detekovaných oddělovačů, které oddělovaly daný uzel od jiného prvku segmentované webové stránky.

Nyní si podrobněji vysvětlíme, z jakého důvodu se tak stane. Dříve jsme řekli, že z důvodu použití jiného algoritmu pro detekci vizuálních oddělovačů jsme nuceni detekovat veškeré oddělovače na webové stránce. Ve výsledku jsou však před započetím fáze konstrukce obsahové struktury zapotřebí pouze oddělovače separující jednotlivé detekované vizuální bloky. Z toho důvodu jsou po ukončení fáze extrakce vizuálních bloků smazány všechny oddělovače, jejichž obě přilehlé oblasti nejsou zároveň vizuálními bloky.

V případě, že však aktuálně zpracovávaný uzel rozdělíme, je třeba zároveň najít všechny oddělovače, ke kterým je tento uzel vázaný a změnit u těchto oddělovačů referenci přilehlé oblasti na jednoho z potomků rozdělovaného uzlu. Vzhledem k tomu, že na webové stránce se potomci nachází uvnitř dotyčného elementu, dojde k přenastavení přilehlé oblasti na oblast nacházející se uvnitř rozdělovaného uzlu a tedy na oblast, která se ještě stále může stát vizuálním blokem.

Tento proces byl nazván rekonfigurace vizuálních oddělovačů a spolehlivě zajišťuje, že žádný z vizuálních oddělovačů nezmizí z důvodu rozdělení některého prvku stránky na jeho potomky. Na obr. 12 je červenou barvou vyobrazen ukázkový vizuální oddělovač. Modrou a zelenou barvou jsou poté znázorněny jeho dvě přilehlé oblasti. Ve chvíli, kdy je rozhodnuto o rozdělení modré oblasti na její potomky, je zapotřebí rekonfigurovat červený oddělovač, neboť modrá oblast zanikne. Reference na přilehlou oblast vizuálního oddělovače je tedy přenastavena na potomka modré oblasti, který je danému oddělovači nejblíže z hlediska absolutní vzdálenosti na webové stránce. V příkladu na obr. 12 se tedy potomek ohraničený růžovou barvou stane novou přilehlou oblastí vizuálního oddělovače.



obr. 12: Ukázka rekonfigurace vizuálního oddělovače

Proces rekonfigurace oddělovačů zajišťuje metoda reconfigureSeparators, která je v průběhu fáze extrakce vizuálních bloků volána pokaždé, když dochází k dělení aktuálně zpracovávaného uzlu vstupního stromu oblastí. Tato metoda nejprve získá seznam všech oddělovačů, ke kterým je

rozdělovaná oblast přilehlá, pomocí volání metody getAssociatedSeparators. Následně je pro každý asociovaný oddělovač nalezen potomek aktuálně rozdělované oblasti, který je od daného oddělovače nejméně vzdálený z hlediska absolutní vzdálenosti na webové stránce. Tento potomek poté nahradí svého rodiče a stane se novou přilehlou oblastí daného oddělovače.

6.2.4 Heuristická pravidla

Za pomoci heuristických pravidel dochází ve fázi extrakce vizuálních bloků k rozpoznání takových prvků webové stránky, které lze označit za vizuální, tedy prvků, které se objeví na výstupu celého segmentačního procesu. Jedná se tedy o jednu z nejdůležitějších částí celého algoritmu, která je implementována v rámci metody isVisualBlock.

Každý uzel rekurzivně procházeného DOM stromu je postupně kontrolován sadou heuristických pravidel ve stejném pořadí, v jakém za sebou pravidla následují v této kapitole. V okamžiku, kdy je některé z heuristických pravidel splněno, je na základě jeho definice rozhodnuto, zdali je aktuálně kontrolovaný uzel vizuálním blokem, či nikoliv.

V následujícím textu si představíme konkrétní znění jednotlivých heuristických pravidel. Některá pravidla byla použita v původním znění algoritmu VIPS (kapitola 4.2.2) [2], jiná byla mírně upravena [1][5][12] a navíc byla přidána 3 nová vylepšující heuristická pravidla [12].

Heuristické pravidlo 1:

Pokud uzel DOM stromu není validním uzlem a zároveň neobsahuje žádné validní uzly jako své potomky, poté tento uzel nemůže být dále rozdělován a bude odstraněn.

Heuristické pravidlo slouží k odstranění takových částí webové stránky, které nejsou z pohledu uživatele viditelné. Viditelnost prvku na webové stránce však není testována pouze na základě podmínky nenulové šířky a výšky. Nejprve je zkontrolováno, zdali se prvek nachází uvnitř viditelné oblasti. Dále, pokud se jedná o textový prvek, je zkontrolováno, jestli je viditelný samotný text a také, jestli barva textu je rozdílná od barvy pozadí textového prvku.

Pokud by nám to použité renderovací jádro umožnilo, mohli bychom také kontrolovat CSS vlastnosti „visibility“ a „display“. Tuto možnost však bohužel musíme prozatím zařadit do možných budoucích rozšíření.

Heuristické pravidlo 2:

Pokud aktuální uzel DOM stromu obsahuje pouze jednoho validního potomka a tento potomek není textovým uzlem, aktuální uzel bude rozdělen.

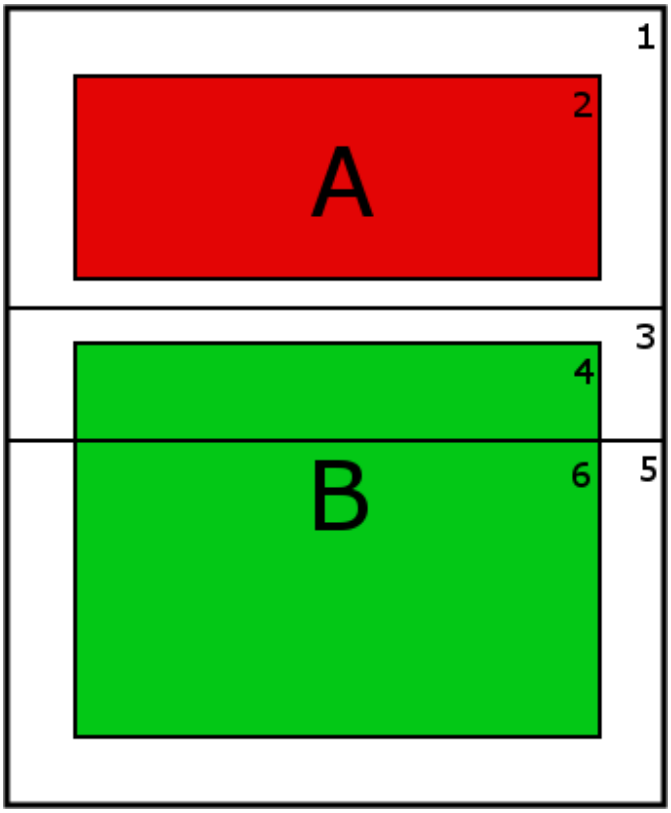
Také by se dalo říci, že v definovaném případě bude aktuální uzel nahrazen svým potomkem, který bude dále zpracováván stejným způsobem. Toto heuristické pravidlo slouží k odstranění nadbytečných HTML značek obalujících jeden textový element a umožňuje vysegmentování samotného textového obsahu webové stránky.

Odstraněné heuristické pravidlo 3:

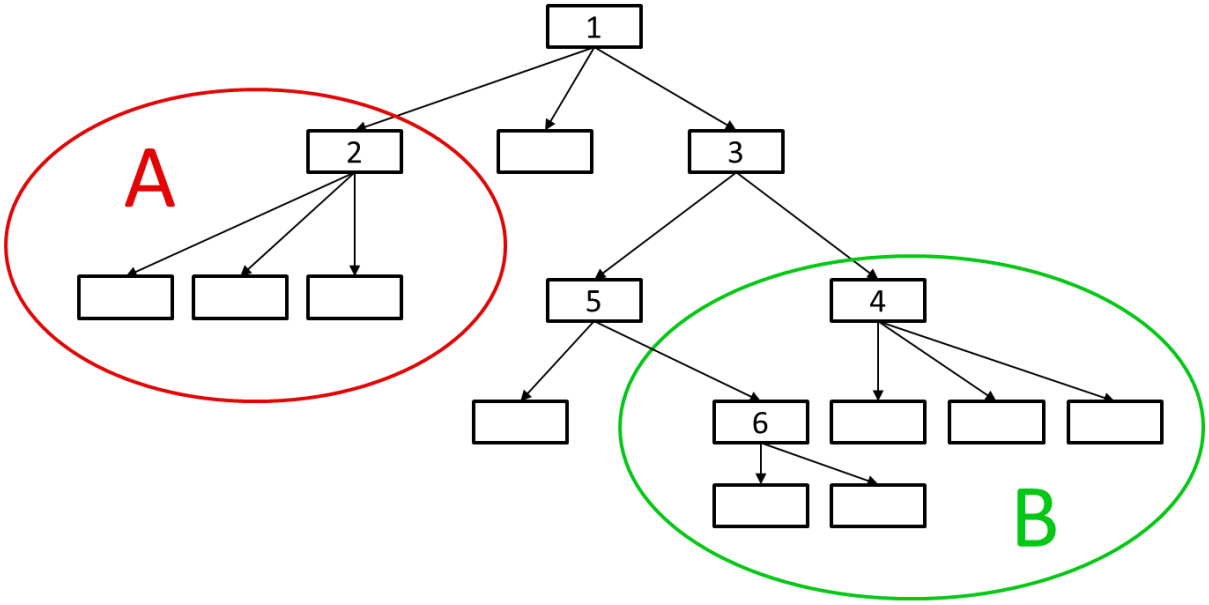
Pokud je uzel DOM stromu kořenovým uzlem DOM podstromu, který odpovídá konkrétnímu bloku webové stránky a zároveň existuje pouze jeden DOM podstrom odpovídající tomuto konkrétnímu bloku, poté bude aktuální uzel rozdělen.

Toto heuristické pravidlo bylo v rámci implementace této práce odstraněno navzdory původní definici algoritmu VIPS. Důvodem je skutečnost, že bez aplikování heuristických pravidel následujících po pravidlu číslo 3 není možné identifikovat konkrétní vizuální blok obsažený na webové stránce. Tudíž nelze s jistotou určit, zdali je aktuálně zpracovávaný uzel DOM stromu kořenovým uzlem nějakého DOM podstromu, odpovídajícího jednomu vizuálnímu bloku.

Další důvod pro odstranění třetího heuristického pravidla si názorně představíme na obr. 13, na kterém je vyobrazeno rozložení ukázkové webové stránky. Základní strukturu DOM stromu této stránky poté můžeme vidět na obr. 14. Předpokládejme, že bychom byli schopni identifikovat vizuální bloky webové stránky bez aplikace ostatních heuristických pravidel. Pokud bychom se v rámci příkladu zobrazeném na obr. 13 rozhodli respektovat znění heuristického pravidla číslo 3, znamenalo by to, že blok „A“, který je tvořen pouze jedním DOM podstromem (číslo 2) by byl rozdělen na menší části, zatímco blok „B“, který je tvořen pomocí kombinace podstromů 4 a 6 by zůstal nerozdělen. Z praktického hlediska takové chování není velice přínosné, alespoň tedy vzhledem k přístupu, kterého se snažíme dosáhnout v rámci této segmentační metody. Korektní chování by mělo vypadat tak, že prvky, které se skládají z více dohromady nekonzistentních částí, by měly být rozděleny na jednotlivé konzistentnější bloky a prvky, které již jsou konzistentní, by měly vytvořit konkrétní vizuální bloky. Konzistentní bloky by poté měly být případně dále rozděleny pouze na základě nastavené jemnosti segmentačního procesu pomocí hodnoty stupně koherence. Z výše uvedeného vyplývá, že pro korektní použití třetího heuristického pravidla by bylo zapotřebí obsáhlejšího popisu účelu tohoto pravidla ze strany původních autorů algoritmu VIPS, neboť v současném stavu by pravidlo přineslo jen znehodnocení výsledků segmentačního procesu.



obr. 13: Rozložení vzorové webové stránky



obr. 14: základní struktura DOM stromu webové stránky zobrazené na obr. 13

Heuristické pravidlo 4:

Pokud jsou všichni potomci aktuálního uzlu DOM stromu textovými uzly, nebo virtuálními textovými uzly, aktuální uzel nebude dále rozdělen (vytvoří vizuální blok).
Dále pokud je velikost a váha fontů všech synovských uzlů stejná, hodnota stupně koherence aktuálního uzlu bude nastavena na 1.
Pokud není stejná pouze velikost fontů všech synovských uzlů, je aktuální uzel dále zpracován vylepšujícími heuristickými pravidly 1 a 2.
Pokud není stejná pouze váha fontů všech synovských uzlů, hodnota stupně koherence aktuálního uzlu bude nastavena na 0,9.

Jedná se o jedno z nejdůležitějších heuristických pravidel, které má za následek shlukování konzistentních oblastí do jednotlivých vizuálních bloků. Zároveň se zde poprvé objevuje pojem vylepšujících heuristických pravidel. Tato pravidla byla implementována za účelem zlepšení uspořádání výstupu segmentačního procesu. Konkrétní chování vylepšujících heuristických pravidel 1 a 2 se nachází v sekci 6.2.5.

Heuristické pravidlo 5:

Pokud je některý ze synovských uzlů aktuálního uzlu DOM stromu prvkem způsobující zalomení řádku (line-break element), poté bude aktuální uzel rozdělen.

Pravidlo má za následek rozdělení oblasti obsahující například prvek s HTML značkou <P> (odstavec) na jednotlivé oddělené podoblasti.

Heuristické pravidlo 6:

Původní znění:
Pokud je některý z potomků aktuálního uzlu DOM stromu prvkem s HTML značkou <HR>, poté bude aktuální uzel rozdělen.
Upravené znění:
Pokud aktuální uzel DOM stromu obsahuje potomka, jehož HTML značka je <HR>,
, nebo se jedná o jakýkoliv prvek způsobující zalomení řádku, který zároveň neobsahuje žádné validní potomky, poté bude aktuální uzel rozdělen.

Důvod úpravy tohoto pravidla je poměrně zřejmý, neboť prvek s HTML značkou
 (řádkový zlom) plní na webové stránce totožnou funkci jako prvek se značkou <HR> (vodorovné rozdělení). Stejně tak u prvku produkujícího zalomení řádku, který neobsahuje žádné validní potomky, se dá očekávat, že slouží pouze jako vizuální oddělení určitých oblastí.

Heuristické pravidlo 7:

Pokud je barva pozadí aktuálního uzlu DOM stromu rozdílná od barvy pozadí některého z jeho synovských uzlů, bude aktuální uzel rozdělen. Zároveň synovské uzly s rozdílnou barvou pozadí nebudou v této iteraci algoritmu dále děleny a jejich hodnota stupně koherence bude nastavena v rozsahu 0,6 až 0,8 na základě konzistence jejich vnitřní struktury.

Pravidlo slouží k rozpoznání oblastí webové stránky s rozdílnou barvou pozadí. S velkou pravděpodobností se dá předpokládat, že oblasti s rozdílnou barvou pozadí slouží k odlišení významově rozdílného obsahu.

Heuristické pravidlo 8:

Pokud aktuální uzel DOM stromu obsahuje alespoň jednoho potomka, který je textovým uzlem, nebo virtuálním textovým uzlem a relativní velikost aktuálního uzlu na webové stránce je menší než prahová hodnota, pak již tento uzel nebude dále rozdělen. Zároveň bude nastavena hodnota stupně koherence v rozsahu 0,5 až 0,8 na základě konzistence vnitřní struktury aktuálního uzlu.

Za pomoci uvedeného heuristického pravidla lze ovlivnit minimální relativní velikost prvků, které mohou být v průběhu segmentačního procesu rozděleny na menší části. Prvky s relativní velikostí menší než zvolená prahová hodnota budou označeny jako vizuální bloky. Konkrétní vlastnosti prahové hodnoty nejsou v původní definici algoritmu VIPS uvedeny, proto byla tato hodnota na základě testování nastavena na jedno procento relativní velikosti celé webové stránky. V případě potřeby by mohla být tato hodnota jednoduše přenastavena pomocí vlastnosti pageThreshold třídy VipsBasedOperator.

Heuristické pravidlo 9:

Pokud je relativní velikost největšího potomka aktuálního uzlu DOM stromu menší než prahová hodnota, nebude aktuální uzel dále rozdělen.

Pravidlo má podobný účel jako předchozí definované pravidlo číslo 8. Pokud aktuální uzel obsahuje pouze potomky s relativní velikostí menší než je prahová hodnota, nechceme tento uzel již

dále rozdělovat. Jedná se o stejnou prahovou hodnotu, která byla použita v heuristickém pravidle číslo 8.

Heuristické pravidlo 10:

Pokud předchozí sourozenec aktuálního uzlu DOM stromu nebyl rozdělen, nebude rozdělen ani aktuální uzel.

V případě, kdy nebylo u aktuálně zpracovávaného uzlu DOM stromu splněno žádné z předcházejících heuristických pravidel, je projevena snaha o zachování konzistentní struktury obsahu webové stránky. Tudíž nebudeme dělit uzel, jehož předchozí sourozenec taktéž nebyl rozdělen. Pravidlo se týká zejména buněk tabulky.

Heuristické pravidlo 11:

Rozděl aktuální uzel DOM stromu.

V případě, že ani doposud nebylo u aktuálně zpracovávaného uzlu DOM stromu splněno žádné z předcházejících heuristických pravidel, je zavoláno pravidlo, které je splněno vždy. Toto pravidlo má pokaždé za následek rozdělení aktuálně posuzovaného uzlu DOM stromu na menší části.

Heuristické pravidlo 12:

Aktuální uzel DOM stromu nebude rozdělen.

Jedná se o pravidlo, které je opakem heuristického pravidla číslo 11. Je voláno ve stejném případě jako předchozí heuristické pravidlo, avšak nikdy nejsou pro jeden konkrétní uzel zavolány obě tato pravidla současně. Rozhodnutí, které z těchto pravidel bude pro konkrétní uzel zavoláno, závisí na HTML značce kontrolovaného uzlu. Více o tomto postupu se dozvíme v sekci 6.2.6.

6.2.5 Vylepšující heuristická pravidla

Do vytvářené segmentační metody byla dále zakomponována nová heuristická pravidla, jejichž účelem je mírné zlepšení uspořádání výsledné obsahové struktury [12]. Tato pravidla byla vhodným způsobem zakomponována tak, aby mohla plnit svou funkci a zároveň nenarušovala běžnou činnost segmentační metody založené na algoritmu VIPS.

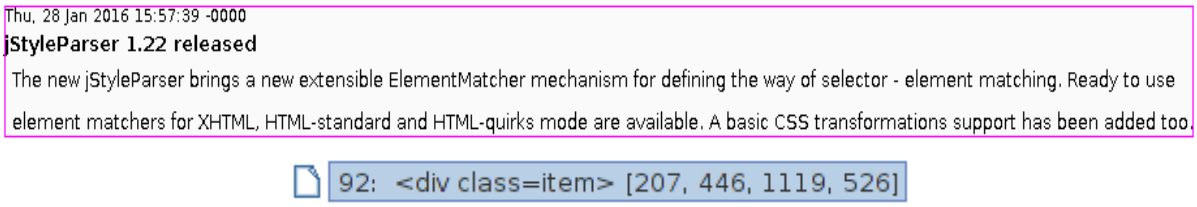
Jedná se o poměrně pokročilá heuristická pravidla, jejichž implementace a zakomponování do dříve definované segmentační metody nebylo zcela triviální činností. U každého vylepšujícího heuristického pravidla si kromě samotné definice předvedeme také praktický dopad jeho aplikace na konkrétní elementy webové stránky.

Vylepšující heuristické pravidlo 1:

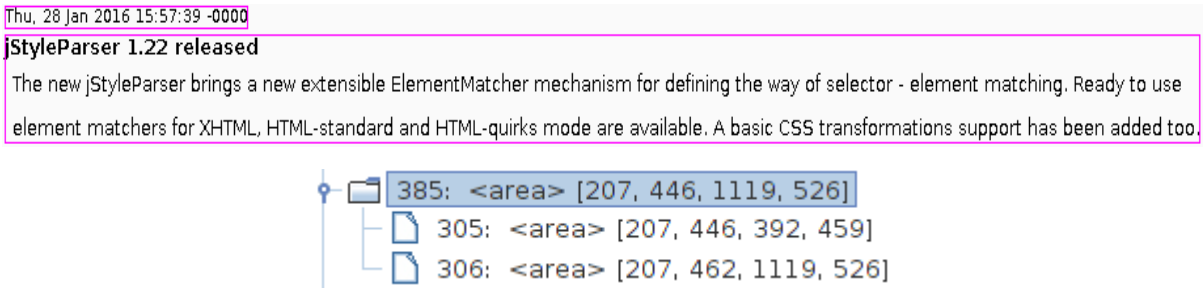
Pokud má některý ze synovských uzlů aktuálně zpracovávaného uzlu DOM stromu větší velikost fontu než jeho předcházející sourozenec, bude aktuální uzel rozdělen na dva nové vizuální bloky. Synovské uzly, které se nachází před uzlem s větší velikostí fontu, se stanou potomky prvního vytvořeného bloku, ostatní synovské se stanou potomky druhého vytvořeného bloku.

Lze předpokládat, že pokud se někde mezi prvním a posledním potomkem jedné oblasti webové stránky nachází element s větší velikostí fontu, jedná se s velkou pravděpodobností o nadpis uvádějící význam textu, který pod ním následuje. Z toho důvodu jsou z dané oblasti vytvořeny dva oddělené bloky, přičemž potomkem druhého bloku se stane zmíněný uzel s větší velikostí fontu a všechny uzly, které po něm následují.

Názorně lze aplikaci zde popisovaného heuristického pravidla vidět na obr. 15 a obr. 16. Růžovou barvou jsou ohraničeny listové uzly, jež se nachází na výstupu segmentačního procesu. Na obrázcích lze dále vidět i výslednou obsahovou strukturu jednotlivých segmentovaných odstavců. Na obr. 15 konkrétně se poté nachází vysegmentovaný odstavec bez aplikace prvního vylepšujícího heuristického pravidla, jak vidno celý odstavec tvoří jeden vizuální blok. Oproti tomu příklad vysegmentovaného odstavce s použitím zde zmíněného heuristického pravidla lze vidět na obr. 16.



obr. 15: Odstavec webové stránky bez aplikace vylepšujícího heuristického pravidla č. 1 včetně obsahové struktury



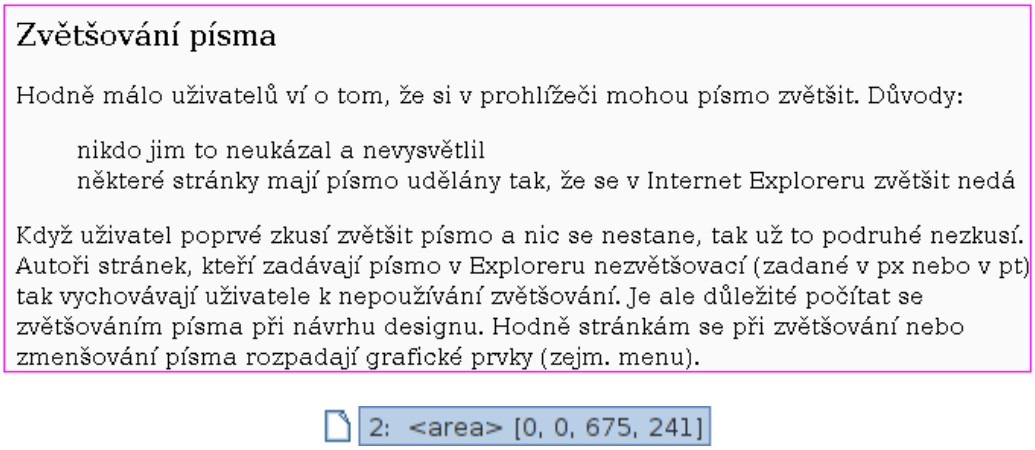
obr. 16: Odstavec webové stránky po aplikaci vylepšujícího heuristického pravidla č. 1 včetně obsahové struktury

Vylepšující heuristické pravidlo 2:

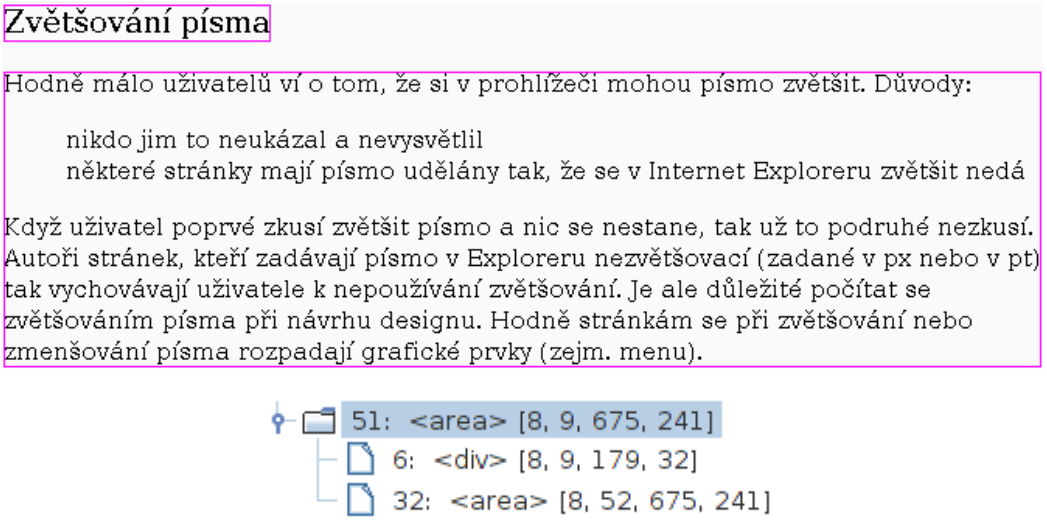
Pokud má první potomek aktuálně zpracovávaného uzlu DOM stromu větší velikost fontu než všichni zbývající potomci, bude aktuální uzel rozdělen na dva nové vizuální bloky. První vytvořený blok se stane rodičovským uzlem potomka s větší velikostí fontu, druhý vytvořený blok poté obsáhne všechny zbývající potomky.

Pravidlo má velice podobný význam jako vylepšující heuristické pravidlo číslo 1. Jediným rozdílem je umístění synovského uzlu s větší velikostí fontu. Výsledkem je tedy opět změna uspořádání obsahové struktury segmentované textové oblasti.

Na obr. 17. a obr. 18 můžeme vidět příklad konkrétního textového segmentu a jeho reprezentaci bez aplikace a po aplikaci druhého vylepšujícího heuristického pravidla. Výsledkem je tedy opět rozdělení původního segmentu na dva oddělené vizuální bloky.



obr. 17: Textový blok webové stránky segmentovaný bez použití druhého vylepšujícího heuristického pravidla



obr. 18: Textový blok webové stránky segmentovaný po použití druhého vylepšujícího heuristického pravidla

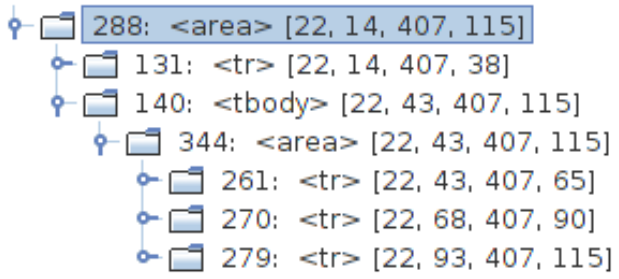
Vylepšující heuristické pravidlo 3:

Pokud je aktuálně zpracovávaný uzel DOM stromu zobrazený na webové stránce jako tabulka a některé jeho sloupce mají odlišnou barvu pozadí než ostatní, je pro každý sloupec tabulky vytvořen jeden samostatný vizuální blok.

Je velice pravděpodobné, že pokud mají některé sloupce tabulky odlišnou barvu pozadí, jsou data v jednotlivých sloupcích tabulky významově sdružené. Tabulka je ve výchozím stavu v jazyce HTML organizována po řádcích. Pokud je tedy podmínka vylepšujícího heuristického pravidla 3 splněna, dojde k restrukturalizaci tabulky tak, že jejími bezprostředními potomky se stanou oblasti zahrnující jednotlivé sloupce.

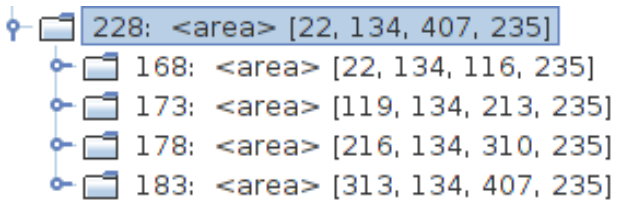
Rozdíl v segmentaci tabulky se stejnou a s rozdílnou barvou pozadí v jednotlivých sloupcích s použitím třetího vylepšujícího heuristického pravidla lze vidět na obr. 19, respektive obr. 20.

Sloupec 1	Sloupec 2	Sloupec 3	Sloupec 4
11	12	13	14
21	22	23	24
31	32	33	34



obr. 19: Segmentace tabulky se shodnou barvou pozadí ve všech sloupcích

Sloupec 1	Sloupec 2	Sloupec 3	Sloupec 4
11	12	13	14
21	22	23	24
31	32	33	34



obr. 20: Segmentace tabulky s rozdílnou barvou pozadí v jednotlivých sloupcích

Dále jsou popsána některá heuristická pravidla, která nejsou do segmentační metody zakomponována z důvodu nekompatibility s vykreslovacím strojem CSSBox použitým v nástroji FITLayout. V aktuální verzi neexistuje spolehlivá možnost, jak získat CSS vlastnosti požadované v konkrétních heuristických pravidlech. Pravidla jsou však uvedena z důvodu možnosti budoucího rozšíření segmentační metody v případě přechodu na jiné vykreslovací jádro. Jedná se tedy o další pravidla z množiny vylepšujících heuristických pravidel, která jsou z důvodu aktuálního nezakomponování do segmentační metody nazvána doplňující heuristická pravidla.

Doplňující heuristické pravidlo 1:

Pokud obsahuje aktuálně zpracovávaný uzel DOM stromu nějaké potomky, jejichž mezera způsobená zalomením řádku je větší než u ostatních potomků, poté bude aktuálně zpracovávaný uzel DOM stromu rozdělen. Každý potomek, jehož mezera způsobená zalomením řádku je větší, bude tvořit oddělený blok. Ostatní potomci nacházející se vedle sebe budou zahrnuti do společných bloků.

Doplňující heuristické pravidlo 2:

Pokud aktuálně zpracovávaný uzel DOM stromu obsahuje alespoň jednoho potomka s CSS vlastností „float“ nastavenou na hodnotu „left“, nebo „right“, jsou vytvořeny tři bloky a pro každého potomka platí, že:
Pokud je hodnota jeho CSS vlastnosti „float“ nastavená na hodnotu „left“, je potomek vložen do prvního nově vytvořeného bloku.
Pokud je hodnota jeho CSS vlastnosti „float“ nastavená na hodnotu „right“, je potomek vložen do druhého nově vytvořeného bloku.
Pokud hodnota CSS vlastnosti „float“ daného potomka není nastavena na hodnotu „left“, ani hodnotu „right“, je potomek vložen do třetího nově vytvořeného bloku.

Doplňující heuristické pravidlo 3:

Pokud aktuálně zpracovávaný uzel DOM stromu obsahuje potomka, jehož hodnota CSS vlastnosti „margin-top“ nebo „margin-bottom“ je rozdílná od nuly, je aktuálně zpracovávaný uzel DOM stromu rozdělen na dva nové bloky.
Potomci nacházející se před uzlem s nenulovou vlastností „margin“ jsou vloženi do prvního nově vytvořeného bloku.
Potomci nacházející se za uzlem s nenulovou vlastností „margin“ jsou vloženi do druhého nově vytvořeného bloku.
A pro potomka s nenulovou CSS vlastností „margin“ dále platí:
Pokud má potomek nenulovou pouze CSS vlastnost „margin-top“, je vložen do druhého nově vytvořeného bloku.
Pokud má potomek nenulovou pouze CSS vlastnost „margin-bottom“, je vložen do prvního nově vytvořeného bloku.
Pokud má potomek nenulové obě CSS vlastnosti „margin-top“ i „margin-bottom“, je vytvořen třetí nový blok a vložen mezi první dva vytvořené bloky.

6.2.6 Aplikace heuristických pravidel

V předchozím textu byla představena jednotlivá heuristická pravidla v pořadí, ve kterém jsou aplikována na aktuálně zpracovávané uzly vstupního DOM stromu. Je však zapotřebí upřesnit, že na konkrétní uzel DOM stromu nejsou aplikována veškerá uvedená pravidla, ale pouze ta, která mají pro daný uzel smysl. Aplikace konkrétních heuristických pravidel je tedy závislá na HTML značce daného DOM uzlu, neboť ta určuje povahu elementu webové stránky. Zmíněná funkcionalita je implementována v rámci metody isVisualBlock, která zkontroluje HTML značku vstupního uzlu a na jejím základě spustí aplikaci určených heuristických pravidel. Metoda isVisualBlock rozděluje vstupní uzly do těchto kategorií:

- **Řádkové elementy**: zde je třeba zmínit, že se jedná o vhodnou fázi algoritmu, v rámci které je možné rozšířit množinu používaných HTML značek o značky z jazyka HTML5. Rozpoznání řádkového elementu totiž realizuje metoda isInlineNode, která rozeznává následující HTML značky představující řádkové elementy:
  <B>, <BIG>, <I>, <SMALL>, <TT>, <ABBR>, <ACRONYM>, <CITE>, <CODE>, <DFN>, <EM>, <KBD>, <STRONG>, <SAMP>, <TIME>, <VAR>, <A>, <BDO>, <Q>, <SPAN>, <SUB>, <SUP>, <LABEL>, <U>, <S>, <STRIKE>, <DEL>, <INS>, <MARK>, <RUBY>. [14][15]

- **Tabulka**: HTML značka <TABLE>.

- **Řádek tabulky**: HTML značka <TR>.

- **Buňka tabulky**: HTML značka <TD>.

- **Odstavec**: HTML značka <P>.

- **Ostatní** HTML značky.

Na základě výše uvedených kategorií jsme schopni určit, která konkrétní heuristická pravidla mají být aplikována na konkrétní vstupní uzly DOM stromu. Toto rozdělení je definováno v rámci tab. 1, kde HP představuje heuristické pravidlo a VHP vylepšující heuristické pravidlo. Znak „✓“ poté znázorňuje aplikaci daného pravidla.

tab. 1: Aplikace jednotlivých heuristických pravidel na konkrétní elementy webové stránky

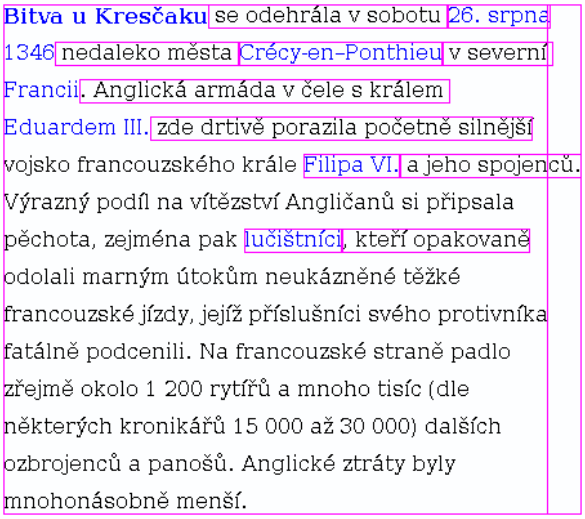
	HP1	HP2	VHP3	HP4	HP5	HP6	HP7	HP8	HP9	HP10	HP11	HP12
Řádkový element	✓	✓		✓	✓	✓		✓	✓		✓	
Tabulka	✓	✓	✓				✓		✓			✓
Řádek tabulky	✓	✓					✓		✓			✓
Buňka tabulky	✓	✓		✓				✓	✓	✓		✓
Odstavec	✓	✓		✓	✓	✓		✓	✓		✓	
Ostatní	✓	✓		✓		✓		✓	✓		✓	

Ještě je třeba zmínit, že se v tab. 1 nenachází vylepšující heuristická pravidla číslo 1 a 2. Důvodem je, že tato pravidla jsou zahrnuta uvnitř heuristického pravidla číslo 4 tak, jak bylo definováno v sekci 6.2.4.

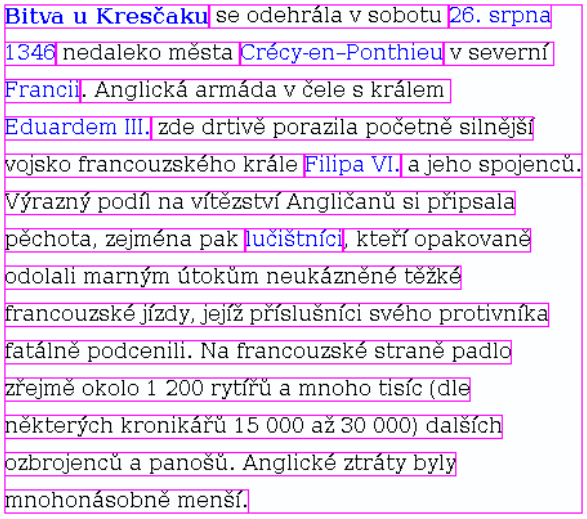
6.2.7 Spojování vizuálních bloků na řádku

Před zahájením fáze konstrukce obsahové struktury je třeba provést jednu modifikaci spočívající ve spojení vizuálních bloků, které společně tvoří jeden řádek textu. V průběhu testování totiž bylo zjištěno, že v případě nastavení velmi jemného segmentačního procesu nastane situace, kdy jsou jednotlivé řádky textu rozděleny na několik vizuálních bloků, které však leží těsně vedle sebe a tudíž se mezi nimi nenachází žádné vizuální oddělovače. V takovém případě by v průběhu fáze konstrukce obsahové struktury nemohli být jednotlivé vizuální bloky spojeny dohromady, neboť tato činnost vychází právě z existence zmíněných oddělovačů. Konkrétní případ lze vidět na obr. 21 a obr. 22, kde jsou růžovou barvou ohraničeny jednotlivé vysegmentované vizuální bloky.

Funkcionalita této modifikace je implementována v rámci metody joinLineVisualBlocks, která nalezne všechny vizuální bloky ležící těsně vedle sebe na jednom řádku a vytvoří mezi nimi vertikální oddělovače. Vytvořeným oddělovačům je poté nastavena minimální možná váha, protože je zapotřebí, aby vizuální bloky tvořící jeden řádek textu byly ve fázi konstrukce obsahové struktury spojeny dohromady jako první.



obr. 21: Nekorektní výsledek segmentace odstavce textu bez aplikace spojení vizuálních bloků na řádku



obr. 22: Výsledek segmentace odstavce textu po aplikaci modifikace spojení vizuálních bloků na řádku

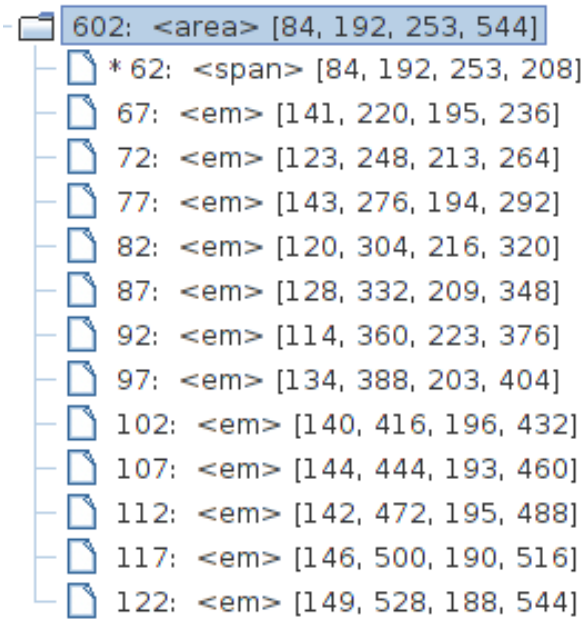
6.2.8 Konstrukce obsahové struktury

Ve fázi konstrukce obsahové struktury jsou postupně zpracovávány jednotlivé detekované vizuální oddělovače v pořadí určeném jejich vahou. V každé iteraci je vyjmut oddělovač s nejnižší vahou a vizuální bloky, které jsou k danému oddělovači přidruženy, jsou spojeny do nově vzniklého bloku obsahové struktury. Zmíněné vizuální bloky se tedy stanou ve stromové struktuře potomky nově vzniklého bloku.

Dále je třeba zajistit, že nově vzniklý blok obsahové struktury nahradí své potomky. Tedy že se stane přidruženou oblastí všech oddělovačů, u kterých dříve figurovali jako přidružené bloky právě zmínění potomci. Tímto způsobem je iterativně zajištěno sestavení extrahovaných vizuálních bloků do výsledné stromové obsahové struktury definované v kapitole 4.1. Celý proces sestavení je přitom založen na hodnotě vah jednotlivých detekovaných vizuálních oddělovačů. Je tedy zaručeno, že vizuální bloky budou formovány od nejvíce spřízněných prvků webové stránky, do stále větších oblastí.

I zde lze však narazit na několik výjimečných situací, které si nyní podrobněji popíšeme. V případě, že existuje několik sousedících vizuálních bloků, které jsou odděleny více vizuálními oddělovači se stejnou vahou, je zapotřebí, aby se všechny tyto vizuální bloky staly potomky nově vzniklého bloku najednou a zůstala tak zachována korektní hierarchie stromové struktury. Popsanou situaci lze vidět na obr. 23, kde jsou jednotlivé odkazy navigačního menu odděleny vizuálními oddělovači se stejnou vahou (bílé mezery). Jak lze dále vidět na obr. 24, jednotlivé položky navigačního menu tedy nejsou spojovány do oddělených oblastí po dvojicích na základě jednotlivých oddělovačů, ale jsou všechny najednou spojeny do jednoho nového bloku. Situace by byla odlišná, pokud by některý z oddělovačů byl například širší, tedy pokud by měl větší váhu než ostatní. V takovém případě by byla hierarchická struktura výsledného segmentu více zanořená.

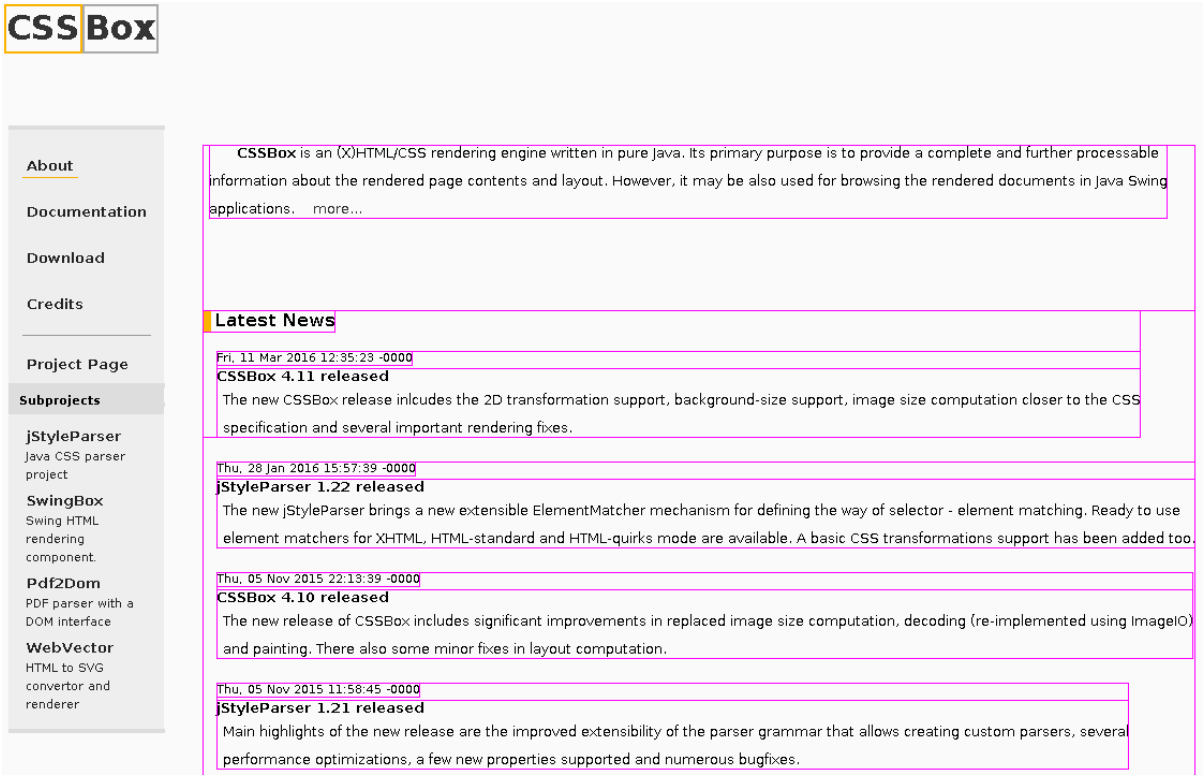
obr. 23: Položky navigačního menu odděleny vizuálními oddělovači se stejnou vahou



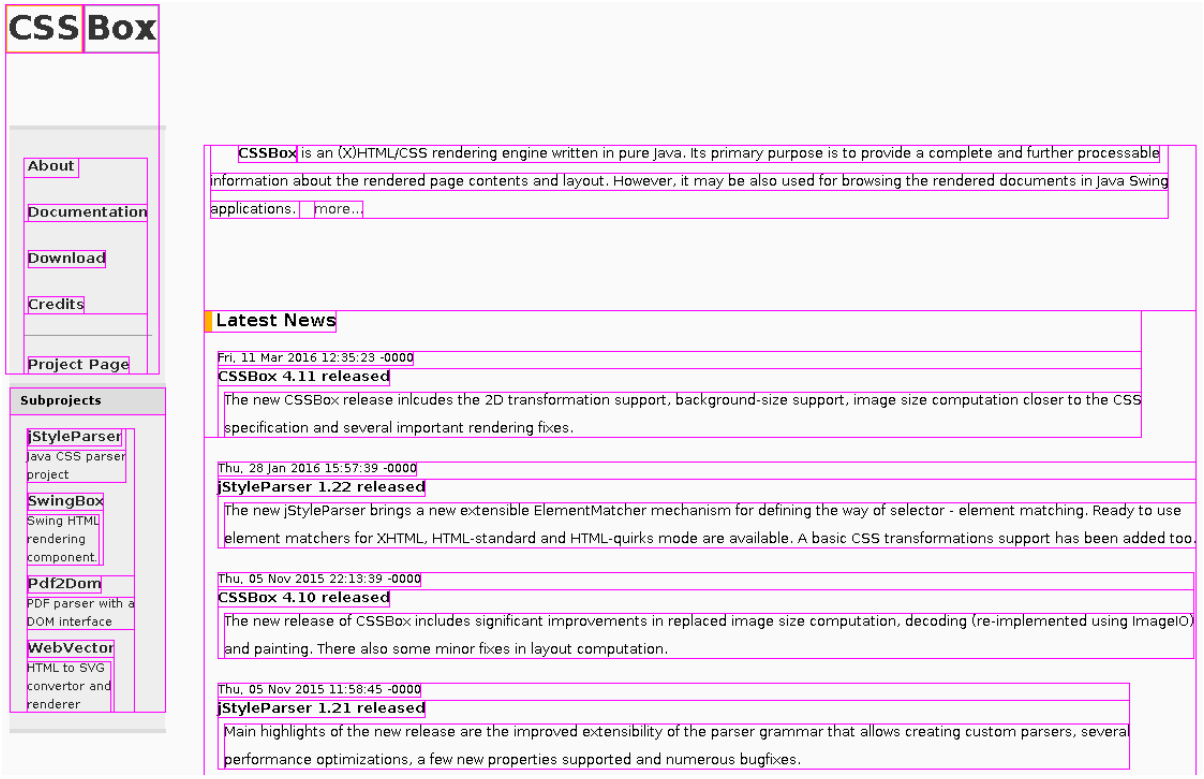
obr. 24: Výsledná obsahová struktura navigačního menu z obr. 23

Dále bylo v průběhu testování zjištěno, že se některé vizuální bloky webové stránky neobjeví na výstupu segmentačního procesu, tedy nejsou vysegmentovány. Důvodem je, že v některých velice specifických situacích není korektně detekován vizuální oddělovač patřící k danému prvku webové stránky. Často je tato situace způsobena určitou nedokonalostí při vykreslování stránky pomocí nástroje FITLayout. Pokud však není korektně detekován vizuální oddělovač spojující určitý prvek se zbytkem webové stránky, nebude vytvořený podstrom zmíněného prvku připojen k výslednému stromu obsahové struktury, a tudíž se nezobrazí na výstupu segmentačního procesu. Řešení problému spočívá v uložení všech podstromů vytvářených ve fázi konstrukce obsahové struktury do jednoho seznamu, přičemž u každého podstromu je poznamenáno, zdali již byl připojen jako potomek do jiného vytvořeného stromu. Na konci aktuální fáze algoritmu jsme poté schopni rozeznat všechny podstromy webové stránky, které ještě nebyly začleněny do výsledné obsahové struktury. Metoda processUnusedSubtrees poté nalezne nejhlouběji zanořenou oblast obsahové struktury, ve které je zcela obsažen doposud nezačleněný podstrom. Ten je poté přidán jako přímý potomek nalezené oblasti. Díky tomuto procesu je zajištěno, že se na výstup segmentační metody dostanou všechny detekované vizuální bloky i v případě nekorektní detekce potřebného vizuálního oddělovače. Příklad

konkrétní situace ukazující výsledek segmentace před a po aplikaci výše uvedené optimalizace můžeme vidět zobrazen růžovým ohraničením na obr. 25, respektive na obr. 26.



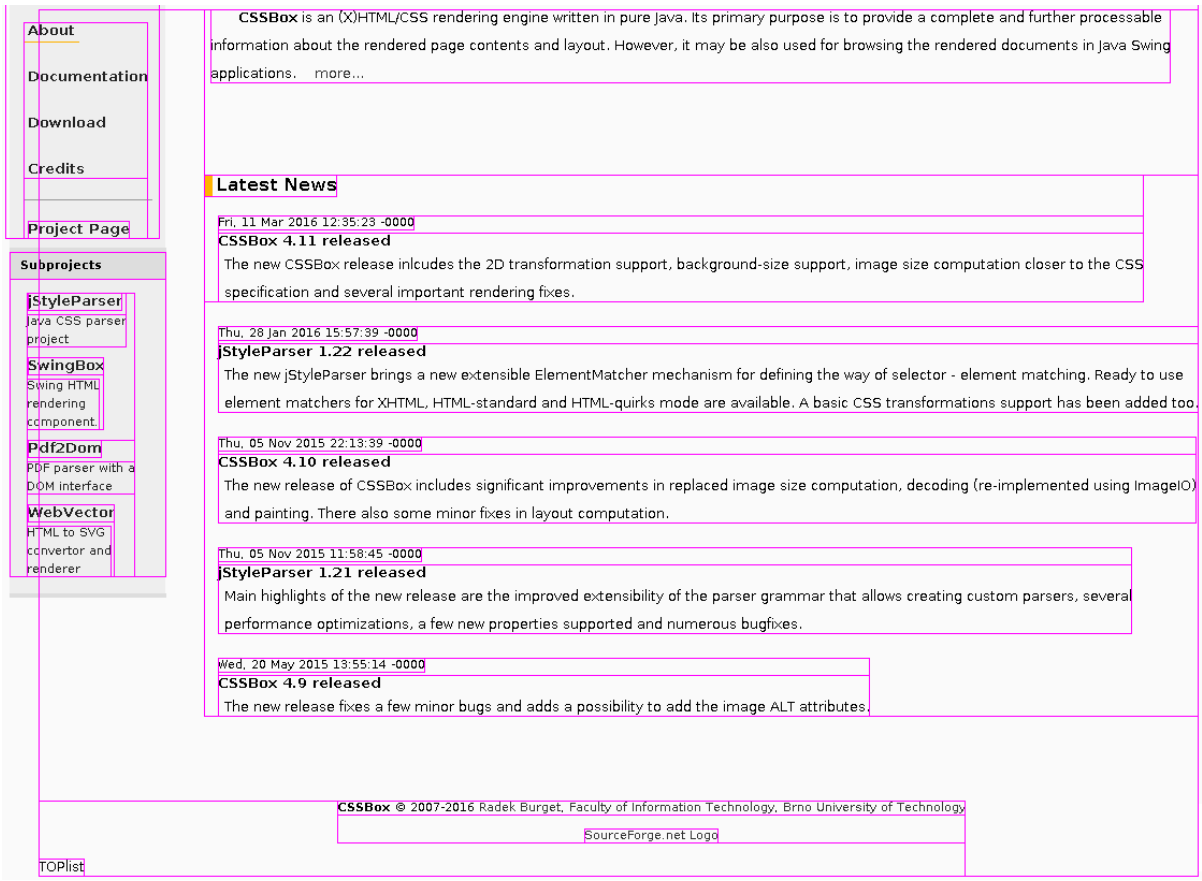
obr. 25: Výsledek segmentačního procesu před zavedením kontroly nepoužitých podstromů



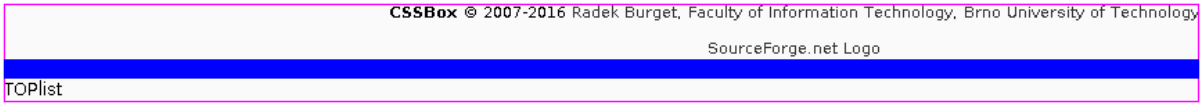
obr. 26: Výsledek segmentačního procesu po zavedení kontroly nepoužitých podstromů

Dále jsou v průběhu fáze konstrukce obsahové struktury odstraňovány vytvořené oblasti, které se nesprávně kříží s dalšími oblastmi webové stránky. Příklad takové situace lze vidět na obr. 27. V prvním kroku jsou odstraněny nekorektní detekované horizontální oddělovače pomocí metody removeIncorectHSeparators. Jedná se o vizuální oddělovače, které horizontálně oddělují prvky nacházející se na opačných stranách webové stránky v horizontální rovině. Jak lze vidět na obr. 28 a obr. 29, zobrazené prvky by správně měly být odděleny vertikálně, nikoliv horizontálně. Vertikální oddělovač je mnohem širší, tudíž bude mít mnohem větší váhu a zajistí, že prvky nacházející se na opačných koncích webové stránky nebudou spojovány dohromady mezi prvními. Odstraněním nekorektních horizontálních oddělovačů je tedy zabráněno vytváření oblastí, které by se významným způsobem křížily s korektně vytvořenými oblastmi, čímž je zaručena určitá celistvost výsledné obsahové struktury.

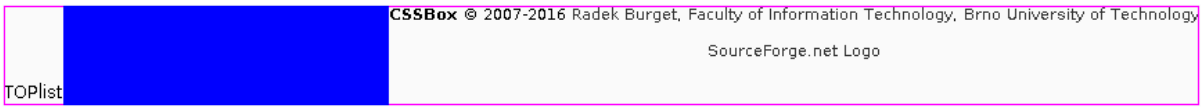
Z hlediska rozmanitosti webových stránek také může nastat situace, kdy není křížení oblastí způsobeno nekorektním horizontálním oddělovačem. V takovém případě je vytvořená oblast způsobující křížení přímo odstraněna a její potomci jsou zanecháni k dalšímu zpracování. To znamená, že jsou dále buď zahrnuti do jiné korektně vzniklé oblasti, nebo jsou nakonec zpracováni metodou processUnusedSubtrees popsanou výše, která je vloží na korektní pozici ve výsledné obsahové struktuře.



obr. 27: Příklad nekorektního křížení několika vysegmentovaných oblastí webové stránky



obr. 28: Ukázka nekorektního horizontálního oddělovače



obr. 29: Vertikální oddělovač, kterým je nahrazen nekorektní horizontální oddělovač z obr. 28

V závěru fáze konstrukce obsahové struktury jsou pomocí metody processLeafNodes nalezeny všechny listové uzly výsledného stromu, přičemž je pro každý uzel zkontrolována podmínka zrnitosti. V případě, že je tato podmínka splněna pro všechny listové uzly, je proces segmentace ukončen a výsledná obsahová struktura je zobrazena na výstupu nástroje FITLayout. Listové uzly, které podmínku zrnitosti nesplní, jsou předány na vstup segmentační metody a znovu projdou všemi fázemi algoritmu. Takové uzly jsou tedy dále děleny na menší části, dokud není podmínka zrnitosti splněna pro všechny listové uzly výsledné obsahové struktury.

7 Testování na reálných dokumentech

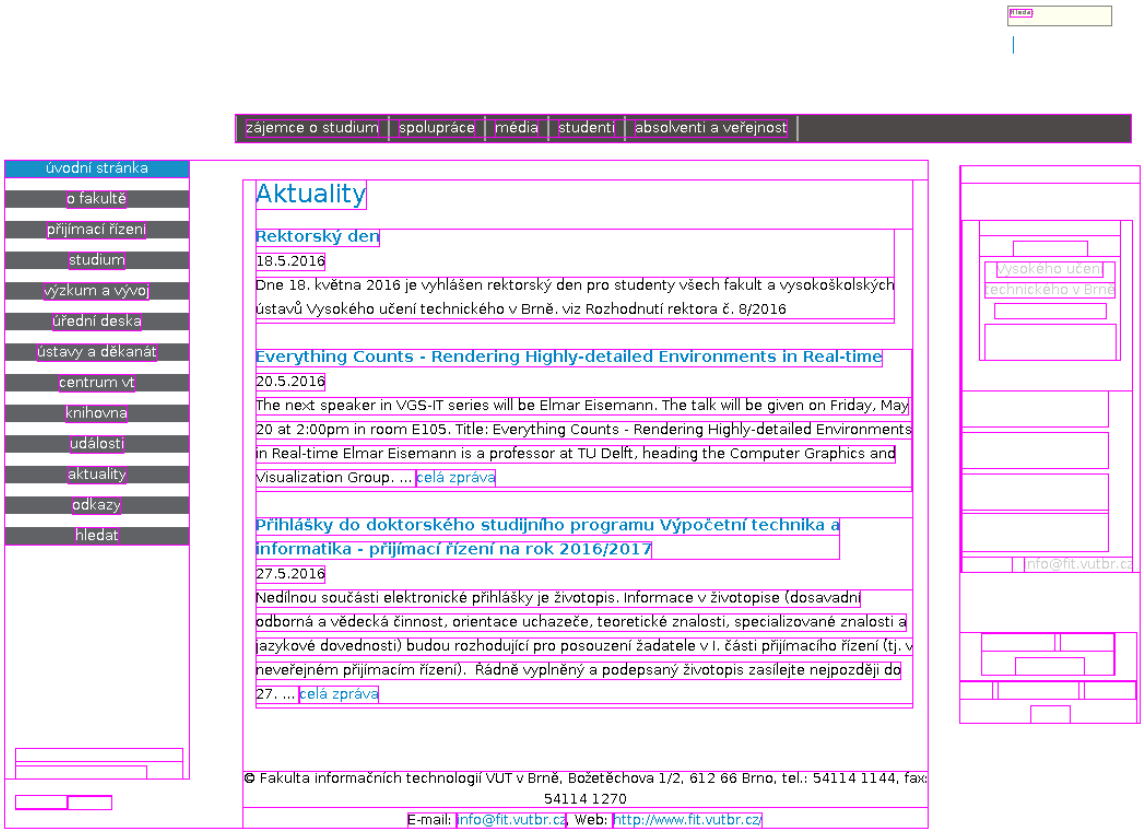
V následujícím textu budou představeny výsledky testování vytvořené segmentační metody na reálných webových dokumentech. V průběhu implementace bylo za pomoci průběžného testování na existujících webových stránkách nalezeno velké množství nedostatků v segmentačním procesu, které byly nakonec vyřešeny v rámci výše popsaných optimalizací a metod vzniklých nad rámec definice původního algoritmu VIPS.

V dokončené fázi poté byla kvalita výsledné segmentační metody prověřena na testovací sadě několika webových dokumentů, a to zejména z důvodu možnosti porovnání metody s původním segmentačním algoritmem obsaženým v nástroji FITLayout. Je nutné poznamenat, že v případě vývoje segmentačních metod podobného typu v rámci nástroje, jakým je FITLayout, se jedná především o experimentální vývoj. Je nutné počítat také s jistými omezeními například v oblasti vykreslování webových stránek, které pochopitelně můžou mít určitý dopad na výsledek segmentačního procesu založeného na analýze vizuálních vlastností vstupního dokumentu.

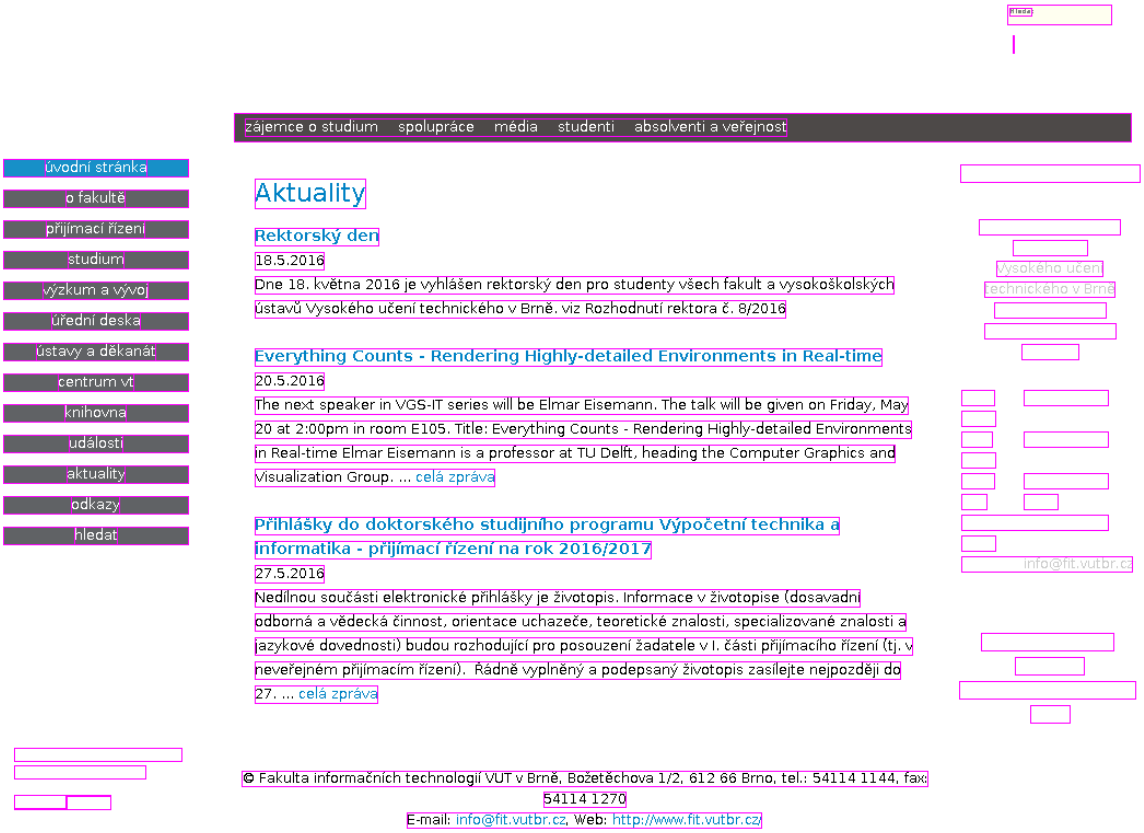
I přesto je však pomocí vytvořené segmentační metody dosahováno velice uspokojivých výsledků při segmentaci reálných webových stránek. Zejména je dobrých výsledků dosahováno při analýze dokumentů zaměřených na statickou prezentaci textových informací. Na následujících obrázcích jsou k vidění výsledky segmentace některých webových dokumentů obsažených v testované sadě. Růžovou barvou jsou označeny vysegmentované oblasti jednotlivých webových stránek.

Na několika úvodních obrázcích (obr. 30 až obr. 33) se nachází porovnání mezi výsledky segmentace provedené vytvořenou segmentační metodou a původním segmentačním algoritmem obsaženým v nástroji FITLayout. Viditelné rozdíly jsou způsobeny především odlišným uspořádáním výsledné obsahové struktury. U metody založené na algoritmu VIPS se na výstupu nachází hierarchicky zanořená struktura, což je způsobeno postupným spojováním jednotlivých vizuálních bloků do stále větších oblastí. Na obr. 34 lze poté vidět výsledek segmentace jednoduchého textového článku spolu s okolními prvky webové stránky.

Abychom však nezůstali pouze u kladných výsledků, je třeba ukázat také situaci, kdy segmentace nedopadne zcela korektně. Na obr. 35 lze vidět výstup segmentačního procesu na webové stránce obsahující ve své horní části navigační menu. Problémem je, že navigační menu je z části tvořeno také dynamickým prvek rozbalovacího menu („drop-down menu“). Tento prvek sice je v původním stavu webové stránky vizuálně skryt, avšak jeho HTML struktura se nachází ve vstupním DOM stromu webové stránky. Zároveň nelze položky rozbalovacího menu označit za neviditelné z hlediska definice heuristických pravidel (6.2.4), neboť jsou po najetí kurzoru na webové stránce zobrazeny. Z těchto důvodů se tedy zobrazí jednotlivé položky rozbalovacího menu na výstupu segmentačního procesu, což určitým způsobem reflektuje nepřizpůsobivost segmentační metody založené na algoritmu VIPS k různým dynamickým prvkům vyskytujících se na webových stránkách.



obr. 30: Výsledek segmentace na webové stránce www.fit.vutbr.cz

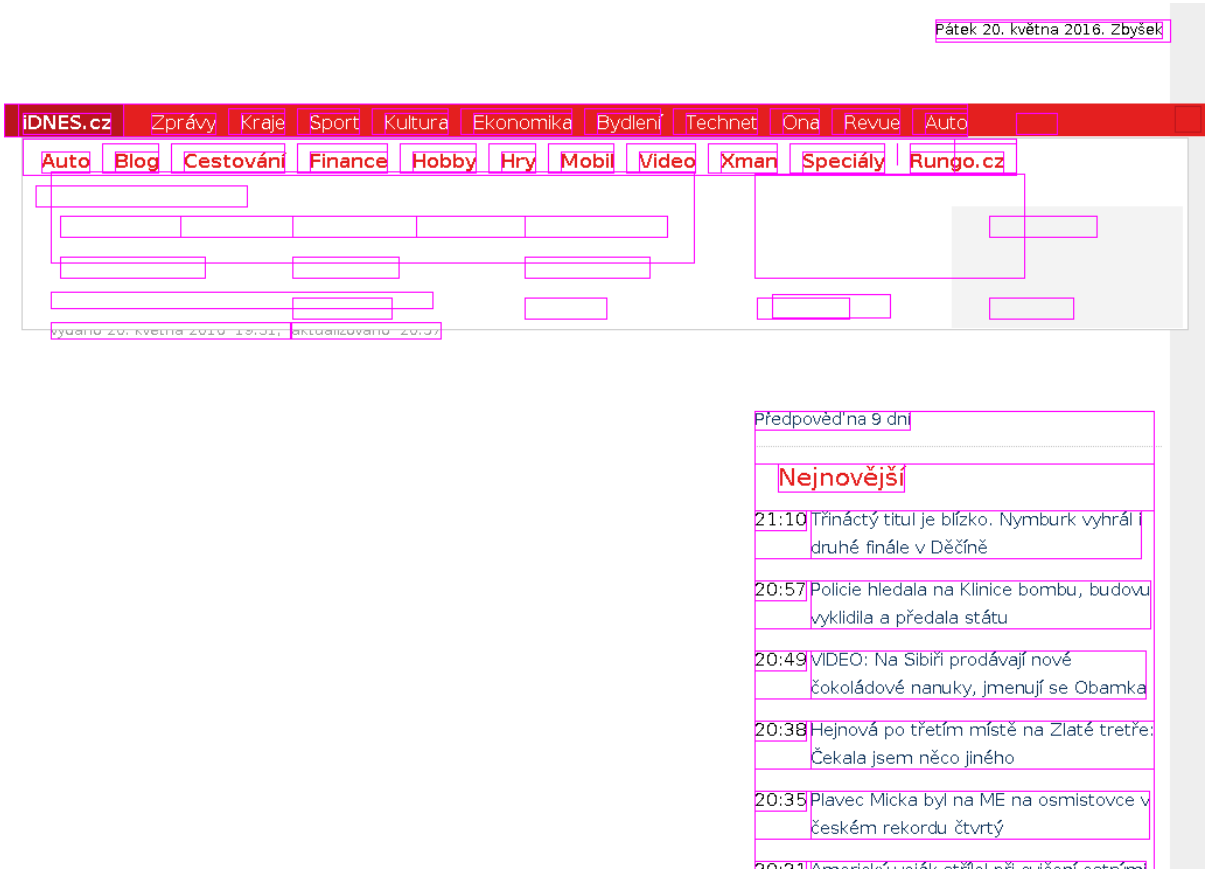


obr. 31: Výsledek segmentace stránky www.fit.vutbr.cz původním segmentačním algoritmem nástroje FITLayout

obr. 32: Výsledek segmentace webové stránky https://cs.wikipedia.org

obr. 33: Výsledek segmentace webové stránky https://cs.wikipedia.org původním segmentačním algoritmem nástroje FITLayout

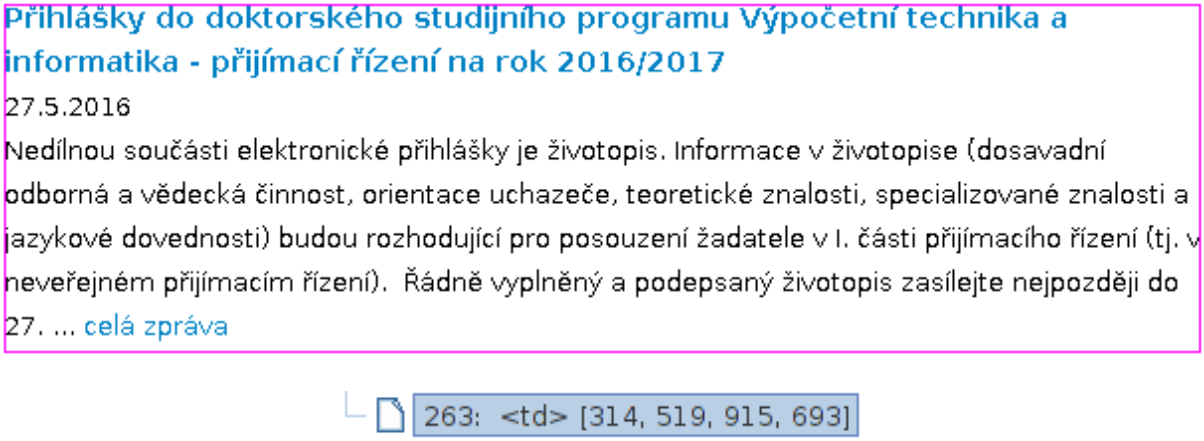
obr. 34: Výsledek segmentace článku na webové stránce www.svetandroida.cz



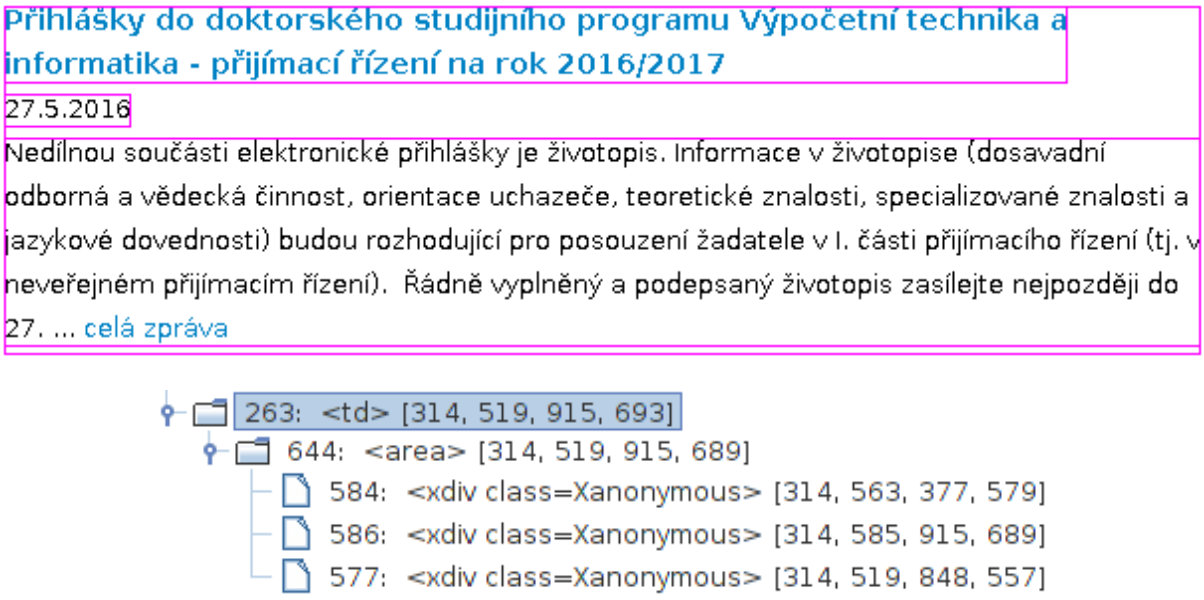
obr. 35: Nekorektní výsledek segmentace webové stránky www.idnes.cz způsobený dynamickým prvkem rozbalovacího menu

7.1 Vliv stupně koherence

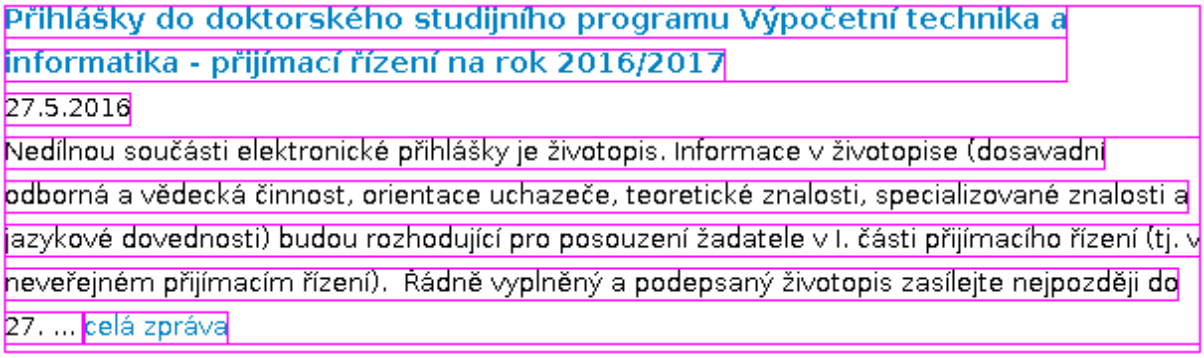
Jak již bylo dříve řečeno, segmentační metoda vytvořená v rámci této práce umožňuje uživateli nastavení jemnosti segmentačního procesu pomocí hodnoty stupně koherence. Na následujících několika obrázcích lze vidět, jakým způsobem ovlivňuje stupeň koherence výstup segmentace reálného webového dokumentu. Aplikovány byly hodnoty stupně koherence 0,4 , 0,7 a hodnota 1,0 značící maximální možnou jemnost segmentace.

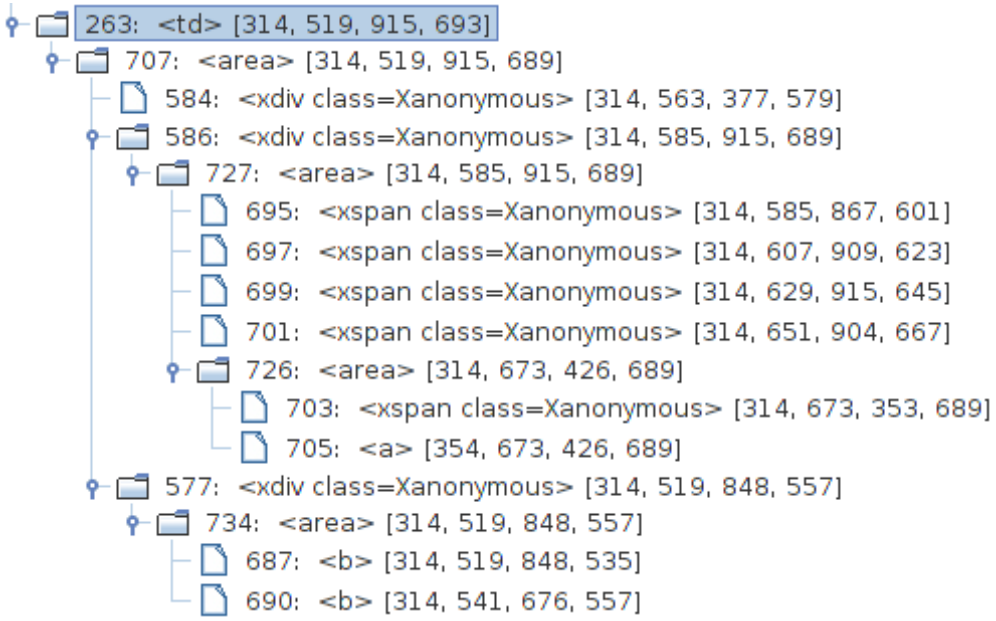


obr. 36: Výstup segmentace odstavce textu – hodnota stupně koherence 0,4



obr. 37: Výstup segmentace odstavce textu – hodnota stupně koherence 0,7





obr. 38: Výstup segmentace odstavce textu – hodnota stupně koherence 1 (maximální jemnost)

8 Závěr

V rámci této práce byla navrhnuta a implementována metoda pro vizuální segmentaci webových stránek, jejíž implementace byla zahrnuta do existujícího segmentačního nástroje FITLayout, vyvinutého na Fakultě informačních technologií VUT v Brně. Z výše uvedených segmentačních algoritmů byl jako základ nové segmentační metody zvolen algoritmus VIPS (Vision Based Page Segmentation Algorithm). Hlavním důvodem je skutečnost, že ačkoliv se jedná o nejstarší z uvedených algoritmů, stále je velice s oblibou používán u mnoha segmentačních metod jako zdroj referenčních dat sloužících k porovnání dané segmentační metody s ostatními. Z tohoto hlediska se jeví jako velice výhodné obsáhnout segmentační metodu založenou na algoritmu VIPS do množiny segmentačních metod nástroje FITLayout a získat tak velice vhodného kandidáta pro experimentální porovnávání kvality ostatních segmentačních metod.

Jak již však bylo řečeno, algoritmus VIPS nepatří mezi zástupce nově vzniklých segmentačních metod a jelikož je rychlost pokroku v oblasti webových technologií opravdu velká, bylo zapotřebí se zabývat možnými vylepšeními a optimalizacemi vzniklé segmentační metody. Aby byly poskytovány korektní výsledky segmentačního procesu, bylo nezbytné zajistit, aby segmentační metoda byla na rozdíl od původní implementace algoritmu VIPS optimalizována pro moderní webové stránky a dokázala analyzovat také stránky vytvořené v jazyce HTML5.

Další řadu optimalizací bylo nutné provést z důvodu přizpůsobení segmentační metody samotnému nástroji FITLayout. Možnosti tohoto nástroje v oblasti renderování webových stránek jsou závislé na aktuálně používaném vykreslovacím stroji CSSBox. Z toho důvodu je třeba k FITLayoutu v rámci implementace segmentační metody přistupovat trochu jinak než ke klasickému webovému prohlížeči, pro jehož použití byl algoritmus VIPS původně navržen. Z hlediska přizpůsobení byly do vzniklé segmentační metody zakomponovány také například některé výhodné aspekty segmentačního procesu, které již byly v nástroji FITLayout implementovány dříve.

Segmentační metoda byla otestována na mnoha reálných webových dokumentech nejprve v průběhu etapy vývoje a následně také ve fázi dokončení. Na základě dosažených výsledků bylo možno identifikovat a následně optimalizovat nalezené nedokonalosti v segmentačním procesu.

Výsledná segmentační metoda dosahuje velice uspokojivých výsledků zejména při segmentaci webových dokumentů zaměřených na statickou prezentaci textových informací. Uživateli metody je navíc poskytnuta efektivní možnost ovlivnění výsledné jemnosti segmentačního procesu. Jak již bylo dříve řečeno, samotný princip algoritmu VIPS není zcela ideální k segmentaci webových stránek tvořených z velké části dynamickým obsahem. Je poměrně zřejmé, že jeden segmentační algoritmus vytvořený v rámci experimentálního vývoje nemůže dosahovat naprosto ideálních výsledků v celém spektru existujících webových dokumentů. Jedním z možných budoucích rozšíření by tedy mohlo být zahrnutí samostatné segmentační metody zaměřené výhradně na segmentaci dat v rámci dynamických prvků. V případě, že by v budoucnu použitý vykreslovací stroj nástroje FITLayout umožňoval jednoduchý přístup k některým pokročilejším CSS vlastnostem webových dokumentů, bylo by možné zahrnout výše uvedená dodatečná heuristická pravidla poskytující jisté doplňující výsledky segmentačního procesu.

Literatura

[1] DENG, Cai, Yu SHIPENG, Ma WEI-YING a Wen JI-RONG. *VIPS: a Vision-based Page Segmentation Algorithm* [online]. 2004 [cit. 2015-11-14]. Dostupné z: https://research.microsoft.com/pubs/70027/tr-2003-79.pdf

[2] DENG, Cai, Yu SHIPENG, Ma WEI-YING a Wen JI-RONG. *Extracting Content Structure of Web Pages based on Visual Representation* [online]. 2003 [cit. 2015-11-14]. Dostupné z: http://www.dbs.ifi.lmu.de/~spyu/paper/VIPS-APWeb.pdf

[3] DENG, Cai, Yu SHIPENG, Ma WEI-YING a Wen JI-RONG. *Improving Pseudo-Relevant Feedback in Web information Retrieval Using Web Page Segmentation* [online]. 2002 [cit. 2015-11-15]. Dostupné z: http://research-srv.microsoft.com/pubs/69980/tr-2002-124.pdf

[4] DENG, Cai, Yu SHIPENG, Ma WEI-YING a Wen JI-RONG. *Block-based Web Search* [online]. 2004 [cit. 2015-11-22]. Dostupné z: http://research-srv.microsoft.com/pubs/69980/tr-2002-124.pdf

[5] DENG, Cai, Yu SHIPENG, Ma WEI-YING a Wen JI-RONG. *VIPS: a Vision-based Page Segmentation Algorithm* [online]. 2004 [cit. 2015-11-25]. Dostupné z: http://www.cad.zju.edu.cn/home/dengcai/VIPS/VIPS_July-2004.pdf

[6] KOHLSCHÜTTER, Christian a Wolfgang NEJDL. *A Densitometric Approach to Web Page Segmentation* [online]. 2008 [cit. 2015-11-30]. Dostupné z: http://webpages.uncc.edu/jfan/webimage2.pdf

[7] ZELENÝ Jan a BURGET Radek. Cluster-based Page Segmentation – a fast and precise method for web page pre-processing. *The Third International Conference on Web Intelligence, Mining and Semantics*. 1. vyd. Association for Computing Machinery, 2013, č. 1. S. 1-12. ISBN 978-1-4503-1850-1.

[8] BURGET, Radek. Layout Based Information Extraction from HTML Documents. *9th International Conference on Document Analysis and Recognition ICDAR 2007*. 1. vyd. IEEE Computer Society, 2007, č. 1. S. 624-629. ISBN 0-7695-2822-8.

[9] ZENG, Jun, Brendan FLANAGAN, Sachio HIROKAWA a Eisuke ITO. A Web Page Segmentation Approach Using Visual Semantics. In: *IEICE Transactions on Information and Systems*. 2014, E97.D(2), s. 223-230. DOI: 10.1587/transinf.E97.D.223. ISSN 0916-8532. Dostupné také z: http://jlc.jst.go.jp/DN/JST.JSTAGE/transinf/E97.D.223?lang=en

[10] FITLayout Web Page Analysis Framework. *FITLayout* [online]. 2014 [cit. 2016-01-28]. Dostupné z: http://www.fit.vutbr.cz/~burgetr/FITLayout/

[11] CSSBox. *SourceForge* [online]. 2007 [cit. 2016-01-29]. Dostupné z: http://cssbox.sourceforge.net/

[12] AKPINAR, M. Elgin a Yeliz YEŞILADA. *Vision Based Page Segmentation Algorithm: Extended and Perceived Success* [online]. , 238 [cit. 2016-03-14]. DOI: 10.1007/978-3-319-04244-2_22. Dostupné z: http://link.springer.com/10.1007/978-3-319-04244-2_22

[13] LI, Long, An Min ZHOU, Yong FANG, Liang LIU a Qian WU. An Improved VIPS-Based Algorithm of Extracting Web Content. *Applied Mechanics and Materials* [online]. 2014, 651-653, 1806-1810 [cit. 2016-03-15]. DOI: 10.4028/www.scientific.net/AMM.651-653.1806. ISSN 1662-7482. Dostupné z: http://www.scientific.net/AMM.651-653.1806

[14] Inline Elements. *MDM* [online]. 2015 [cit. 2016-04-02]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTML/Inline_elements

[15] Úprava textu. *Jak psát web* [online]. 2015 [cit. 2016-04-03]. Dostupné z: http://www.jakpsatweb.cz/html/text.html

[16] BURGET, Radek. Automatic Document Structure Detection for Data Integration. *Business Information Systems* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, , 391 [cit. 2016-05-23]. DOI: 10.1007/978-3-540-72035-5_30. ISBN 978-3-540-72034-8. Dostupné z: http://link.springer.com/10.1007/978-3-540-72035-5_30

Seznam příloh

Příloha1: CD

Obsah CD:

- vips-based-algorithm - zdrojové kódy implementované segmentační metody
- readme.txt – návod ke spuštění programu
- xmalan01_dp.docx – zdrojová verze písemné práce
- xmalan01_dp.pdf – elektronická verze písemné práce