VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

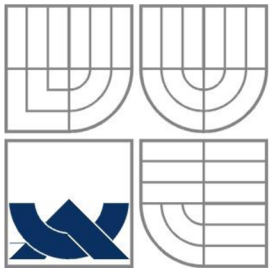
METODY KLASIFIKACE WEBOVÝCH STRÁNEK

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

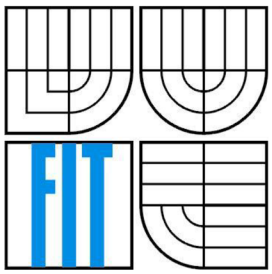
AUTOR PRÁCE
AUTHOR

Bc. VIKTOR NACHTNEBL

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

METODY KLASIFIKACE WEBOVÝCH STRÁNEK

METHODS OF WEB PAGE CLASSIFICATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. VIKTOR NACHTNEBL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VLADIMÍR BARTÍK, Ph.D.

BRNO 2012

Abstrakt

Tato práce se zabývá metodami klasifikace webových stránek. Vysvětluje pojem klasifikace a popisuje různé vlastnosti stránek využívané pro jejich klasifikaci. Dále rozebírá reprezentaci stránky a podrobným způsobem je popsána klasifikační metoda, která pracuje s hierarchickým modelem kategorií a je schopna dynamicky vytvářet nové kategorie. Ve své druhé polovině se věnuje implementaci zvolené metody a výsledkům, které popisuje.

Abstract

This work deals with methods of web page classification. It explains the concept of classification and different features of web pages used for their classification. Further it analyses representation of a page and in detail describes classification method that deals with hierarchical category model and is able to dynamically create new categories. In the second half it shows implementation of chosen method and describes the results.

Klíčová slova

Klasifikace, klasifikace webových stránek, výběr vlastností, hierarchie kategorií, TF-IDF, reprezentace dokumentu, propagace vlastnosti.

Keywords

Classification, web page classification, feature selection, category hierarchy, TF-IDF, document representation, feature propagation.

Citace

Nachtnebl Viktor: Metody klasifikace webových stránek, diplomová práce, Brno, FIT VUT v Brně, 2012

Metody klasifikace webových stránek

Prohlášení

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

……
Viktor Nachtnebl
23.5.2012

Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Vladimíru Bartíkovi, Ph.D. za odbornou pomoc a konzultace, které mi poskytoval během tvorby mé práce.

Obsah

1 Úvod

Diplomová práce pojednává o klasifikaci webových stránek. Jedná se o rozřazení stránek do kategorií na základě jejich vlastností. Na internetu člověk může najít spoustu užitečných informací, ale ne vždy se jedná o lehký úkol. Orientace v takovém množství informací, jaké internet v současné době poskytuje, může být náročná. Klasifikace webových stránek by měla přispět ke snadnější orientaci a rychlejšímu nalezení potřebné a užitečné informace.

Následující kapitola čtenáři ve svém úvodu vysvětluje co to vlastně klasifikace je. Dále je v ní popsáno, jakými způsoby je možné klasifikaci dělit a poukazuje na různá praktická využití. Ve své druhé polovině zkoumá vlastnosti stránek, které jsou vhodné a využívají se k jejich klasifikaci.

Třetí kapitola se věnuje jedné konkrétní zvolené metodě. Aby byl čtenář schopen této metodě porozumět, nejprve objasňuje některé důležité pojmy a postupy v ní využívané. Metoda se zabývá klasifikováním stránek pomocí hierarchické struktury. Její zvláštností je, že nepracuje s pevným počtem předem daných kategorií, ale je schopna vytvářet další, nové kategorie dynamicky.

Ve čtvrté kapitole je popsána neformální specifikace aplikace a rozebrán návrh aplikace, který je doplněn diagramy souvisejícími s návrhem.

Pátá kapitola stručně nastíní implementaci uživatelského rozhraní a podrobně rozebere implementaci metody hierarchické, která je velmi zajímavá specifická.

Předposlední kapitola dává prostor experimentům s výslednou aplikací, zkoumá úspěšnost klasifikátoru z různých pohledů a shrnuje a hodnotí výsledky implementované metody.

2 Webové stránky a jejich klasifikace

Svět internetu je velice obsáhlý a co víc, stále rychleji se rozšiřuje o další nové dokumenty. Dokumenty obsažené v internetu nejsou nijakým způsobem organizovaný do logické struktury. Tyto vlastnosti zhoršují orientaci a hledání informací na internetu. Snahou o nějakou organizaci dokumentů jsou například webové adresáře jako Yahoo!, které klasifikují dokumenty do různých kategorií, což umožňuje snadnější nalezení požadované informace. Co je však jejich nevýhodou, je jejich manuální charakter. Nejen tvorba těchto adresářů, ale i jejich údržba pak stojí ohromné lidské úsilí, nehledě na stále rychlejší růst počtu takových dokumentů. Proto je stále více nasnadě provádět klasifikaci čistě automaticky či s minimálními zásahy člověka. Kapitola čerpá a je z části převzata z [1].

2.1 Definice pojmu

Klasifikace, někdy rovněž zmiňována jako kategorizace, webových stránek je přiřazení stránky do jedné, nebo více předem daných kategorií na základě určitých vlastností. Skládá se typicky z několika kroků. První fází je tzv. fáze učení, kdy je vytvořen klasifikační model (tzv. klasifikátor) pomocí trénovacích dat. Tento model je dále otestován množinou testovacích dat, abychom zjistili, jak úspěšně dokáže klasifikovat. Jak u trénovacích, tak i u testovacích dat je vždy známa kategorie, do které patří, a testovací data by měla být různá od trénovacích. Poté může být klasifikátor použit pro dokumenty, jejichž kategorie není známá a chceme ji určit.

2.1.1 Rozdělení klasifikace webových stránek

Klasifikaci webových stránek je možné dělit z několika různých hledisek. Zpravidla ji dělíme podle typu, počtu kategorií a konečně podle organizace kategorií.

2.1.1.1 Rozdělení podle typu

Klasifikace dle tématu

Jedná se o rozdělování stránek do kategorií podle jejich tématu či předmětu. Jednoduše podle toho o čem pojednávají. Příkladem takové klasifikace je například určení do tříd jako umění, byznys, sport atd.

Klasifikace dle funkce

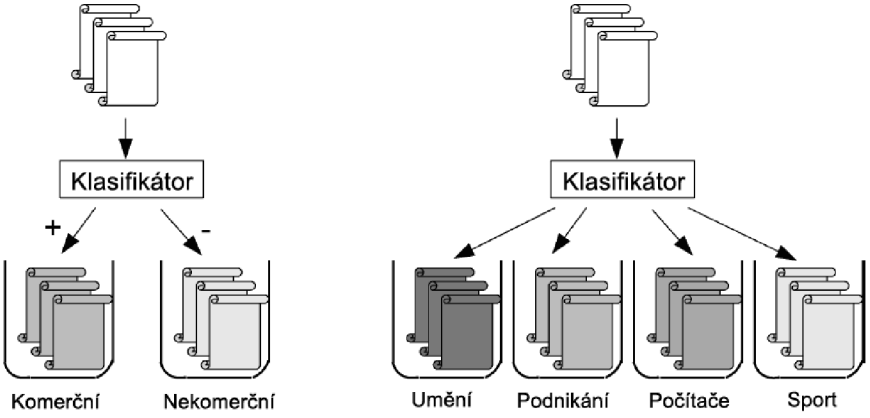
Stránky jsou děleny podle toho, jakou funkci plní. Kategorie v této klasifikaci mohou být např. osobní stránka, stránka s nějakým výukovým kurzem, fotogalerie apod.

Dalšími druhy klasifikace mohou být klasifikace podle stanoviska, která se zabývá rozdělením podle názoru nebo postoje autora k určitému tématu, či klasifikace podle žánru.

2.1.1.2 Rozdělení podle počtu tříd

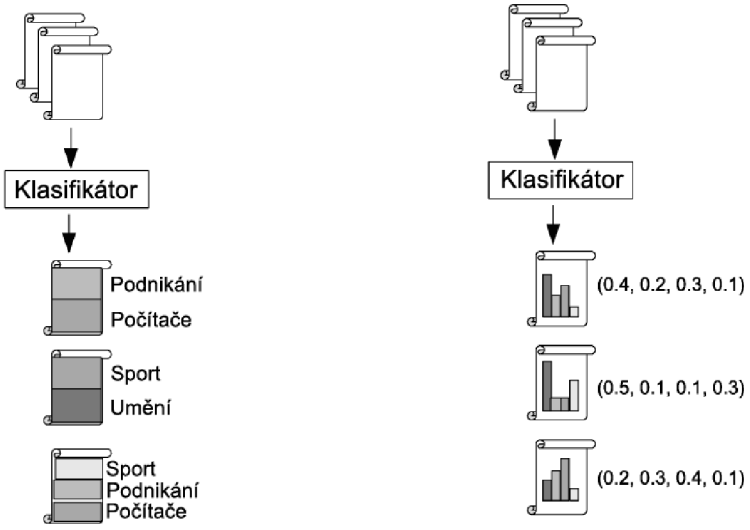
Základní rozdělení je do dvou skupin. Binární klasifikace pracuje jen se dvěma kategoriemi a přiřazuje stránky pouze jedné z nich, zatímco vícetřídní klasifikace, jak již název napovídá, pracuje s vyšším počtem kategorií než s pouhými dvěma.

Vícetřídní klasifikaci je pak možné dále rozdělit v závislosti na tom, do kolika kategorií může klasifikovaná stránka spadat. Jedno-štítková klasifikace přiřazuje danou stránku právě do jedné kategorie. U více štítkové klasifikace je stránce umožněno, aby byla zařazena do více kategorií současně.



Obrázek 2.1: Binární a vícetřídní klasifikace (převzato z [1])

Rovněž u více-štítkové klasifikace je možné odlišit dva případy, jak se s přiřazením více štítků (kategorií) vypořádat. První variantou je takzvaná silná klasifikace, která nám o stránce říká, že buďto do nějaké kategorie patří, nebo nepatří. Žádná jiná možnost nemůže nastat. Oproti tomu slabá klasifikace může stránku přiřadit do nějaké kategorie s nějakou pravděpodobností.



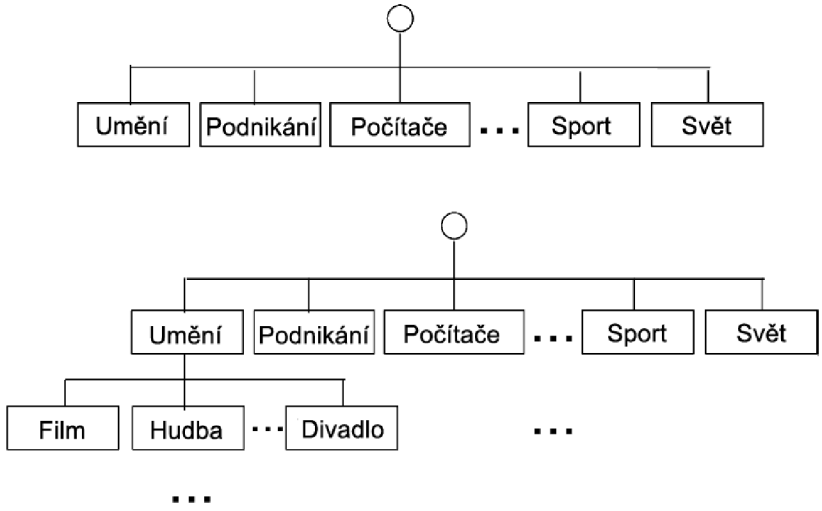
Obrázek 2.2: Vícetřídní, více-štítková klasifikace silná(vlevo) a slabá(vpravo) (převzato z [1])

2.1.1.3 Rozdělení podle organizace kategorií

Na základě organizace kategorií, do kterých jsou stránky přiřazovány, dělíme klasifikaci na plošnou a hierarchickou.

U plošné klasifikace jsou si všechny kategorie rovny a žádná není předchůdcem či následníkem nějaké jiné.

Hierarchická klasifikace, jak již název napovídá, organizuje stránky do hierarchické stromové struktury. To znamená, že uzly stromu reprezentují kategorie a synovské uzly jsou podkategoriemi svých rodičovských uzlů.



Obrázek 2.3: Plošná a hierarchická klasifikace (převzato z [1])

2.1.2 Využití klasifikace webových stránek

Klasifikace webových stránek má široké uplatnění v získávání a spravování informací. Pojďme si nyní uvést některé z nich.

2.1.2.1 Zvýšení kvality výsledků vyhledáváni

Nejednoznačnost dotazů při vyhledávání na internetu znehodnocuje nalezené výsledky. Například slovo zámek má dva významy, a tak není jasné, zda uživatel hledá zámek, který by ochránil nějaký jeho majetek před zloději, nebo se poohlíží po nějaké historické památce. Proto byly navrženy různé metody, jak se s tímto problémem vypořádat. Chekuri a kol. v [5] zkoumali automatickou klasifikaci webových stránek právě a účelem zpřesnění výsledků. V dotazu pak uživatel zadal jednu či více kategorií a vrácený výsledek pak byl pouze z těchto zadaných kategorií, nebo byl vrácen seznam kategorií, ve kterých byly zařazeny nalezené výsledky. Taková metoda ale funguje, pouze pokud uživatel předem ví, co chce hledat.

Zobrazení výsledků vyhledávání může být pro uživatele užitečnější, než seřazený seznam výsledků, který je u vyhledávání obvyklý. Chen a Dumas v [6] navrhli metodu, která klasifikuje výsledky do předem definované hierarchické struktury a uživateli zobrazuje již kategorizovaný přístup. Jejich průzkum mezi uživateli ukázal, že se kategorizované zobrazení výsledků uživatelům líbí více než řazený seznam. Navíc ve srovnání s výše uvedenou metodou není třeba při zadávání dotazu vědět požadovanou kategorii. Ale to je důvod nižší efektivity, protože klasifikace probíhá za běhu.

2.1.2.2 Konstrukce, údržba a rozšiřování webových adresářů

Jak již bylo na začátku kapitoly uvedeno, je snaha o organizaci dokumentů do adresářů, které jsou stále velmi závislé na lidském faktoru. A s rychlostí, jakou se internet vyvíjí a roste, začíná být ruční přístup stále méně a méně efektivní. V tomto ohledu nabízí klasifikace velké využití, ať už jako plně automatické vytváření adresářů, či pomoc při jejich správě a rozšiřování.

2.1.2.3 Napomáhání Question answering (QA) systémům

QA je úloha z oborů zpracování přirozeného jazyka a získávání informací, kdy je cílem automaticky odpovědět na dotaz položený v přirozeném jazyce. QA systémy mohou využívat klasifikace ke zpřesnění jejich kvality. Yang a Chua ve své práci [7] navrhli hledání odpovědí na seznam otázek pomocí klasifikace webových stránek podle funkcionality. Je dán seznam otázek, ze kterého je vytvořeno několik dotazů, které jsou předloženy vyhledávačům. Získané webové stránky z výsledku jsou klasifikovány pomocí rozhodovacího stromu do jedné z těchto čtyř kategorií: sběrné (obsahující seznam prvků), tematické (představující instance s odpovědí), relevantní (stránky doplňující tematické) a nerelevantní stránky. Poté jsou tematické stránky shlukovány a odtud je získán výsledek. Více informací je možno nalézt v [7].

2.1.2.4 Vytváření „zaměřených" robotů procházejících internet

Anglické označení pro tyto roboty zní „web crawlers". Jsou to nástroje sloužící k automatickému procházení internetu. Nejčastěji jsou používány internetovými vyhledávači pro indexování webových stránek. Pokud ale očekáváme pouze dotaz zaměřený na konkrétní doménu, je zbytečné a neefektivní spouštět celkové procházení. Proto Chakrabarti a kol. v jejich práci [8] navrhli tzv. zaměřené procházení internetu, kde jsou uvažovány pouze dokumenty vztahující se k předdefinované množině témat. Tato metoda používá klasifikátor k určení míry relevance stránky k požadovaným tématům, a tak určuje hranice robota při jeho procházení.

2.2 Vlastnosti vhodné ke klasifikaci webových stránek

Již jsme si řekli, že klasifikace probíhá na základě určitých vlastností. Webové stránky, na rozdíl od prostého textu, nabízí širší možnosti výběru takových vlastností, protože obsahují navíc další informace. Mezi ně patří HTML tagy, hypertextové odkazy, texty zobrazující se jako odkazy (tzv. anchor texty, obvykle bývají podtržené) a další vlastnosti, které nejsou textové.

Vlastnosti stránek se dělí na dva typy. Jsou to vlastnosti, které jsou přímo na dané stránce a pak tzv. vlastnosti sousedů. Ty se nachází na stránkách nějakým způsobem příbuzných k té klasifikované.

2.2.1 Vlastnosti nacházející se přímo na stránce

2.2.1.1 Obsah stránky a HTML tagy

Nejčastější reprezentací webové stránky je její textová reprezentace. Pokud chceme reprezentovat textový dokument, jednou z nejznámějších reprezentací je tzv. „bag of words", což není nic jiného než množina slov obsažených v dokumentu. Dokument je pak reprezentován vektorem těchto slov (termů). Je to velmi jednoduchá reprezentace, která však ve své jednoduchosti nese některé nevýhody. Například vezměme dva různé dokumenty. V prvním se vyskytne slovní spojení „pásový opar" a ve druhém se vyskytnou obě slova jak opar, tak pásový, avšak vůbec ne pospolu, nýbrž naprosto odděleně. Při této reprezentaci ztrácíme schopnost rozlišení této informace. Dochází ke ztrátě kontextu.

Další reprezentací, která se snaží korigovat nepřesnosti klasifikace spojené se ztrátou kontextu je reprezentace pomocí N-gramů. Dokument je pak reprezentován těmito N-gramy, neboli posloupnostmi N slov, místo jednotlivými slovy. Ve výše uvedeném příkladu bychom pomocí 2-gramu dokázali rozlišit pásový opar od dvou nezávislých slov ve druhém z dokumentů.

Tyto dvě reprezentace využívají pouze informací prostého textu. Zdrojový text webových stránek je oproti obsahu zobrazenému prohlížečem doplněn o HTML tagy. Informace z těchto tagů se rovněž velmi dobře dá využít pro klasifikaci webových stránek. Byly navrženy různé metody, jak tagů, respektive informace nacházející se uvnitř, využít. Například Golub a Ardo ve své metodě [9] použili čtyři významné elementy stránky – název, nadpisy, meta data a hlavní text. Ukázali, že nejlepších výsledků dosahují při dobře vyvážené lineární kombinaci těchto čtyř elementů.

Byly uvedeny i některé další možnosti klasifikace – klasifikace založená na stručném výtahu z obsahu stránky, nebo klasifikace založena pouze na URL stránek. Její hlavní výhodou je, že není třeba stahovat obsah klasifikované stránky, což se velmi uplatní při vyšších omezeních kladených na čas a prostor nebo u nedostupných stránek.

2.2.1.2 Vizuální struktura stránky

Vizuální reprezentace stránky je rovněž velmi užitečná, ačkoliv většina metod pracuje s textovou, popsanou v HTML. Přestože HTML tagy slouží právě pro vykreslení stránek prohlížečem, tak stejného vizuálního vzezření může být dosaženo různým uspořádáním tagů. Proto někdy může být výhodnější použití vizuální reprezentace než HTML tagů.

Jednu z metod klasifikace webových stránek založenou na vizuální struktuře navrhli ve své studii [10] Kovacevic a kol. Stránku reprezentovali jako hierarchický graf vizuální sousednosti, kde každý uzel reprezentuje HTML objekt a každá hrana prostorový vztah ve vizuální reprezentaci. Na výsledky vizuální analýzy jsou použita heuristická pravidla, aby byly rozpoznány logické oblasti, které odpovídají různým významným částem stránky.

VIPS (Vision-based page segmentation) je algoritmus, který extrahuje sémantickou strukturu stránky založenou na vizuální prezentaci. Tato struktura odpovídá stromu, kde každý uzel reprezentuje blok. Každému uzlu je přiřazena hodnota (stupeň koherence), která říká, jak je daný blok koherentní. Čím vyšší je stupeň koherence, tím více je blok konzistentní. Pomocí předem stanoveného požadovaného stupně koherence určujeme potřebnou granularitu. Dělení tudíž pokračuje pouze v případě, že bloky nemají menší stupeň koherence, než je požadovaný. Ve srovnání s DOM modelem jsou tyto vytvořené bloky mnohem více agregované. Díky tomu je usnadněna redukce nežádoucích informací jako například reklama či navigace. Ty totiž bývají obvykle umístěny v určitých blocích na stránce.

2.2.2 Vlastnosti sousedů

Přestože stránky obsahují užitečné vlastnosti popsané výše, existují stránky, kde takové vlastnosti chybí, jsou zavádějící anebo nerozeznatelné. Tak například stránky, obsahující velké obrázky či flashové objekty, ale málo textových informací, je obtížné správně klasifikovat. Řešením tohoto problému může být použití stránek, které jsou spřízněné s klasifikovanou stránkou a mohou poskytovat další doplňující informace nápomocné při klasifikaci. K určení příbuzných stránek se nejčastěji používají hypertextové odkazy.

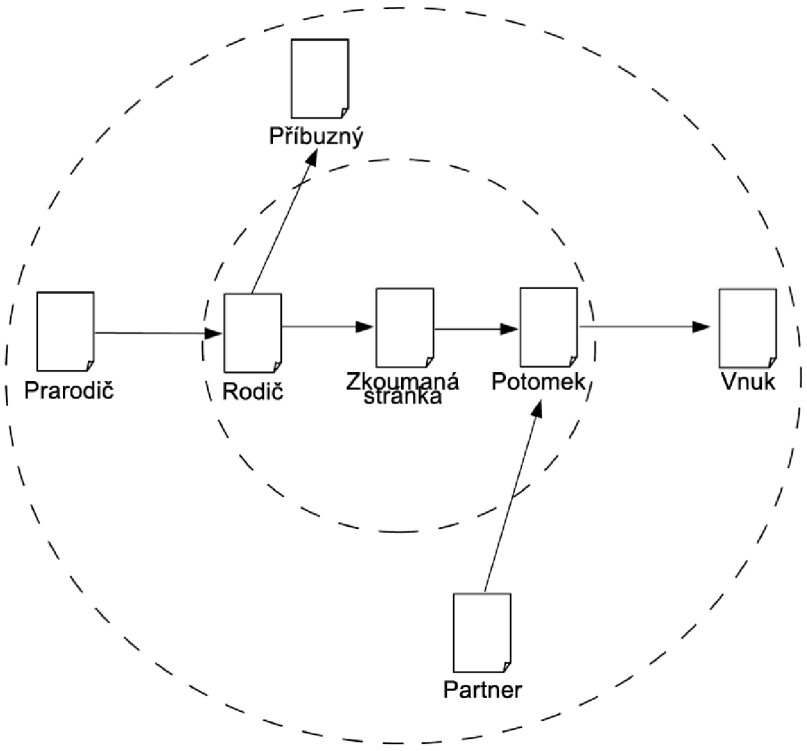
Při práci s charakteristickými vlastnostmi sousedních stránek se zavádí dva předpoklady – slabý a silný. Máme-li dvě stránky p_a a p_b patřící do stejné kategorie, obvykle se předpokládá, že stránky s nimi sousedící mají nějaké společné vlastnosti. Nepožaduje se, aby sousedící stránky patřily do

stejné kategorie jako p_a a p_b. Tento předpoklad je tzv. slabý a dá se požít jak pro klasifikaci podle tématu, tak podle funkce. S tímto předpokladem můžeme vytvořit klasifikátor, jenž se učí na základě vlastností stránek, které jsou sousedící se stránkami z trénovací množiny.

U klasifikace podle tématu se obvykle používá silný předpoklad, který říká, že stránka je mnohem častěji obklopena stránkami ze stejné kategorie. Jinak řečeno, přítomnost mnoha sousedních stránek z kategorie sport zvyšuje pravděpodobnost, že klasifikovaná stránka bude rovněž patřit do této kategorie. Silný předpoklad vyžaduje silnou korelaci mezi odkazy a tématy stránek. Byla prokázána jeho funkčnost u klasifikace do obecných kategorií, ale funkčnost při jemnějším rozdělení není podložena a nelze využít ke klasifikaci podle funkce. Za silného předpokladu může klasifikátor určit kategorii stránky na základě nejčastější kategorie jejích sousedních stránek.

Nyní se podíváme, jaké stránky vlastně využíváme při používání vlastností sousedů. Používají se stránky, které jsou vzdáleny maximálně do dvou odkazů od zkoumané stránky. Dělíme je na šest typů: rodič, potomek, příbuzný, partner, prarodič a vnuk. Přestože se někdy používají i stránky typu prarodič a vnuk, nebyl zatím prokázán jejich individuální přínos.

Nejčastěji používanými vlastnostmi z těchto stránek jsou obsah, částečný obsah (anchor texty, texty obklopující anchor texty, nadpisy a titulky).



Obrázek 2.4: Sousední stránky (převzato z [1])

3 Hierarchická a dynamická klasifikace webových stránek

Tato kapitola popisuje jednu z navržených metod pro klasifikaci webových stránek. Jedná se o metodu založenou na hierarchicky organizované struktuře kategorií. Navíc na rozdíl od drtivé většiny metod nepracuje pouze s předem daným fixním počtem kategorií, do kterých se stránky klasifikují, ale je schopna přidávat další nové kategorie. Díky tomu je schopna mnohem lépe se vypořádat nejen s rostoucím počtem dokumentů na internetu, ale rovněž s přibýváním nových oblastí a domén. Mezi další její výhody také patří schopnost klasifikovat stránku, aniž by porovnávala všechny možné kategorie. V kapitole je nejprve vysvětleno několik základních pojmů, jejichž znalost je potřebná k pochopení této metody, a poté je objasněna samotná metoda hierarchické a dynamické klasifikace stránek. Kapitola čerpá a je částečně převzata z [2], [3], [4].

3.1 TF-IDF přiřazení váhy

Máme-li dokument reprezentovaný množinou slov (bag of words), každé z těchto slov má v daném dokumentu jinou váhu – důležitost – z hlediska klasifikace. Váhování TF-IDF slouží právě k přiřazení váhových hodnot ke každé položce, a tím k vytvoření vektoru vah jednotlivých slov. Tento vektor pak reprezentuje dokument a slouží k určení podobnosti mezi dokumenty. Slova, nebo někdy i fráze, bývají označovány jako termy.

Jak již napovídá pomlčka v názvu algoritmu, tento způsob je kombinací dvou přístupů k přiřazení váhy zkoumanému termu *t* obsaženého v dokumentu *d*. Prvním z nich je TF (Term Frequency), neboli frekvence, s jakou se term vyskytuje v dokumentu. Čím častěji se slovo *t* vyskytuje v dokumentu *d*, tím vyšší je jeho relevance k danému tématu. Avšak pouhý počet výskytů nereflektuje délku dokumentu, proto je více vypovídající relativní frekvence. Vzorec pro výpočet relativní frekvence termu *t* v dokumentu *d* normalizované podle maximální frekvence je uveden ve vztahu 3.1, kde *maxF(d)* je nejvyšší frekvence ze všech termů zkoumaného dokumentu *d*:

$$TF(t,d) = 0{,}5 + \frac{0{,}5 \times f(t,d)}{maxF(d)} \tag{3.1}$$

Druhým přístupem jak ohodnotit váhu daného termu je IDF (Inverse Document frequency) – inverzní četnost výskytu v dokumentech. Čím méně je dokumentů, ve kterých se daný term vyskytuje, tím větší má rozlišovací schopnost. Označme hodnotu, která udává počet všech dokumentů, ve kterých se term *t* vyskytuje alespoň jednou, jako *DF(t)* (Dokument Frequency). IDF pro term t pak vypočítáme pomocí vzorce 3.2:

$$IDF(t) = 1 + \log(\frac{|D|}{DF(t)}) \tag{3.2}$$

kde |D| je celkový počet dokumentů. Nyní již máme obě části popsány a výsledný vzorec 3.3 pro přiřazení váhy *w* pomocí TF-IDF metody je:

$$w(t,d) = TF(t,d) \times IDF(t) \tag{3.3}$$

Tato metoda říká, že slova mají velkou vypovídající hodnotu pro dokument d, když se v něm často vyskytují, ale na druhé straně u slov, která se vyskytují v mnoha dokumentech, se tato hodnota sníží díky nízké hodnotě IDF.

Pro zvýšení přesnosti se v praxi často využívá tzv. stop list. Jedná se o seznam slov, která ačkoliv jsou v dokumentu poměrně frekventovaná, nejsou považována za relevantní a do reprezentace dokumentu se nepromítnou. Mezi ně patří například: a, nebo, k atd. Další metodou je převod slov na základní tvar, tzv. „stemming“, který umožňuje odhalit, že termy v různých tvarech jsou vlastně stejné. Jednou z nejznámějších implementací je Porterův algoritmus založený na odstraňování předpon a přípon pomocí určitých pravidel ve správném pořadí.

Vektorový model prostoru

V tomto modelu jsou dokumenty reprezentovány jako N-rozměrné vektory termů, kde každý z nich reprezentuje jednu dimenzi. Tudíž pro N termů se jedná o N-dimenzionální prostor. Již jsme si popsali, jak přiřadit termům z dokumentu váhu, takže není problém vytvořit vektor těchto vah $d = (w_1, w_2, \ldots, w_N)$, který reprezentuje dokument. Váha w_i odpovídá termu i. Jak se ale porovná podobnost mezi dvěma dokumenty pomocí těchto vektorů? K tomu se používá skalárního součinu těchto vektorů. Máme-li dva dokumenty reprezentované vektory $d_a = (w_{a1}, w_{a2}, \ldots, w_{aN})$ a $d_b = (w_{b1}, w_{b2}, \ldots, w_{bN})$, pak podobnost těchto dokumentů $sim(d_a,d_b)$ spočítáme podle rovnice 3.4:

$$sim(d_a, d_b) = \sum_{x=1}^{N} w_{ax} \times w_{bx} \tag{3.4}$$

Delší dokumenty mají jednak více termů a větší frekvenci termů. Abychom snížili vliv délky dokumentu, použijeme normalizovaný skalární součin:

$$sim(d_a, d_b) = \frac{\sum_{x=1}^{N} w_{ax} \times w_{bx}}{\sqrt{\sum_{x=1}^{N} (w_{ax})^2 \times \sum_{x=1}^{N} (w_{bx})^2}} \tag{3.5}$$

Výsledkem rovnice 3.5 je tzv. kosinová vzdálenost, neboli kosinus úhlu, který vektory svírají. Pokud je kosinová vzdálenost rovna nule, jsou vektory pravoúhlé, a tudíž jsou dokumenty odlišné. Kosinová vzdálenost se dá rovněž vypočítat podle vzorce 3.6.

$$\cos\varphi = \frac{w_a \times w_b}{\|w_a\| \times \|w_b\|} \tag{3.6}$$

Dalším možným způsobem, jak spočítat podobnost dokumentů navrhl v [12] Joachims. Jedná se o kombinaci klasifikátoru založeného na TF-IDF a klasifikátoru založeného na pravděpodobnosti. Jeho navržená vzorec má následující tvar:

$$P(d|c_j) = \sum_{w \in (d \cap c_j)} \frac{P(w|c_j) \times P(c_j)}{\sum_{c \in C} P(w|c) \times P(c)} \times P(w|d) \tag{3.7}$$

$P(d|c_j)$ je pravděpodobnost, s jakou dokument d patří do kategorie c_j. $P(w|c)$ je váha (pravděpodobnost) vlastnosti w v kategorii c, $P(w|d)$ odpovídá váze vlastnosti w ve zkoumaném dokumentu d.

3.2 Well-grained metoda

Další z možných reprezentací dokumentu je reprezentace pomocí N-gramů, o nichž byla řeč ve druhé kapitole. Metoda Well-grained slouží k extrakci vlastností reprezentujících dokument, a jedná se o mírně upravenou metodu použití N-gramů. Základní modifikací je provedení dvou kroků, které některé ostatní metody neprovádí.

Prvním z nich je rozdělení dokumentu na malé oddíly – věty, nebo části vět. K rozdělení jsou použity oddělovače, které se používají v přirozeném jazyce jako: ‘,’, ‘.’, ‘?’, ‘!’, ‘;’, ‘:’, ‘”’ a další, kromě bílých znaků. Oddělovače rozdělí dokument na mnoho malých oddílů, což výrazně zredukuje počet sekvencí slov, které by vznikly při vytvoření reprezentace pomocí N-gramů. Takto však slova oddělená některým z oddělovačů netvoří sekvenci. Další výhodou takového rozdělení je snížení nesouvisejících slov. Například máme-li část dokumentu: „...do té doby, než se všechny figurky dostanou do domečku a dosáhnou tedy vítězného postavení. Mimo hru „Člověče nezlob se“…“, pak by mohla být vytvořena sekvence „postavení mimo hru“ (využití 3-gramů), což by odpovídalo nějakému sportovně založenému tématu, i když je naprosto zřejmé, že se jedná o společenskou hru. Avšak díky rozdělení pomocí tečky na dva oddíly takové situaci zamezíme.

Druhým krokem je přiřazení různé váhy různým oddílům, protože ne všechny jsou stejně podstatné. Jakmile je dokument rozdělen na oddíly, je potřeba určit jejich důležitost, protože se liší. Textový obsah, URL či HTML tagy k tomu mohou napomoci. Kupříkladu oddíl, který se nachází v nadpisu, je více významný, než oddíl nacházející se uprostřed dokumentu. K takovému rozlišení je opět použito různých vah. Příklad můžeme vidět na tabulce převzaté a upravené z [3].

Typ oddílu v dokumentu	**Váha**
Obyčejný text	1
První a poslední odstavec	3
První věta odstavce	3
<Meta>	2
<title>	4
<H1>	4
<H2>	3
<EM>	2
<Strong>	2

Tabulka 3.1: Různé váhy oddílů

3.3 Propagace vlastnosti v hierarchické struktuře

Mnoho existujících struktur kategorií je nevyvážených. Mladeničová ve své práci [4], kde se zabývala nevyváženou stromovou strukturou, navrhla algoritmus řešící jak propagovat vlastnost od listů směrem ke kořenu stromu.

Každý uzel ve stromové struktuře odpovídá jedné kategorii a je reprezentován nejen vlastnostmi dokumentů, které jeho kategorie obsahuje, ale rovněž vlastnostmi dokumentů ze všech svých podkategorií. Tyto uzly a jejich vlastnosti tvoří hierarchickou strukturu, kterou mapujeme do jedné množiny vlastností. Jsou možné různé přístupy, jak takové mapování provádět. Jedním a velmi jednoduchým přístupem je ignorovat hierarchickou strukturu a vytvořit jednu množinu sjednocením

všech vlastností příslušných k danému uzlu i ke všem jeho podstromům. Problémem takového přístupu je, že obsáhlé podstromy překryjí menší podstromy, ačkoliv jsou třeba na vyšší hierarchické úrovni. Pokud uvážíme, že máme hierarchickou strukturu vytvořenou lidmi, pak dochází k úplnému ignorování informace vytvořenou člověkem při vytváření hierarchické struktury, která poukazuje na větší důležitost a obecnost kategorií vyšší úrovně. Například, jak vidíme na obrázku 3.1, kategorie reprezentovaná uzlem číslo 4 obsahuje pouze deset dokumentů, zatímco v podstromu počínajícím uzlem číslo 3 je velké množství dokumentů. V tomto případě by mohlo dojít k „pohlcení" právě tímto velkým podstromem (i když drtivá většina dokumentů spadá dokonce do nižší úrovně). Ale i přes svou malou velikost je určitě kategorie reprezentovaná uzlem 4 důležitá, jelikož je v příkladu hned na druhé úrovni hierarchie. Abychom vzali v úvahu hierarchickou strukturu, přiřadíme váhu každému podstromu a použijeme ji pro přiřazení váhy vlastnostem v nich.

Mějme strom *T* s kořenem v uzlu *N*, který má *k* podstromů označených $SubT_i$, kde *i* nabývá hodnot *1-k*. Pravděpodobnost, že daná vlastnost *t* patří do kategorie reprezentované stromem *T*, se po propagaci spočítá podle rovnice 3.8. Výpočet probíhá rekurzivně od uzlu reprezentujícího zkoumanou kategorii a zastaví se až v listových uzlech.

$$P(w|T) = \sum_{i=1}^{k} P(w|SubT_i) \times P(SubT_i|T) + P(w|N) \times P(N|T) \tag{3.8}$$

Kde součet váhového faktoru uzlu a sumy váhových faktorů podstromů je roven jedné. $P(w|T)$ je propagovaná pravděpodobnost vlastnosti *w*, $P(w|N)$ je pravděpodobnost vlastnosti *w* v uzlu *N*, která se vypočítá například pomoci TF přiřazení váhy podle vzorce 3.9.

$$P(w|N) = \frac{1 + TF(w, N)}{|V| + \sum_i TF(w_i, N)} \tag{3.9}$$

Zde $|V|$ vyjadřuje počet různých vlastností použitých pro reprezentaci dokumentu.

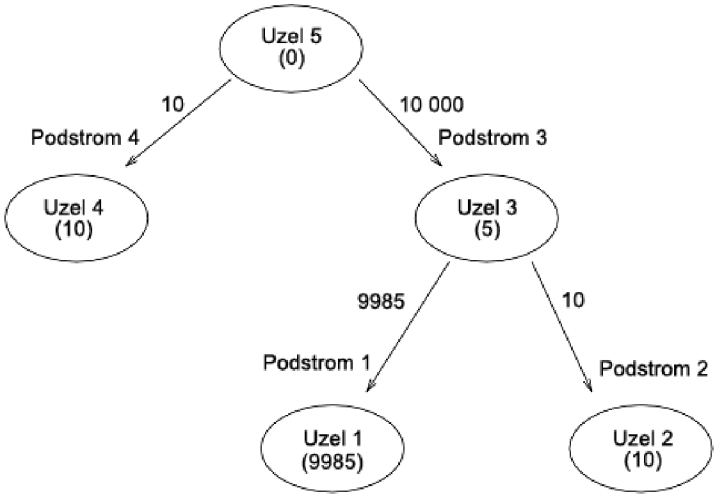
$P(SubT_i|T)$ je hodnota váhy podstromu $SubT_i$ stromu *T*. $P(N|T)$ je váhový faktor aktuálního uzlu *N* stromu *T*. Mohou být použity různé funkce přiřazující váhu, pomocí nichž bychom mohli vypočítat tyto dvě hodnoty. Na základě provedených experimentů některých z nich v [4] se jako vhodné jevily následující vztahy 3.10, respektive 3.11.

$$P(SubT_i|T) = \frac{\ln(1 + Cnt(SubT_i))}{\sum_{j=1}^{k} \ln(1 + Cnt(SubT_j)) + \ln(1 + Cnt(N))} \tag{3.10}$$

$Cnt(SubT_i)$ je počet dokumentů v podstromu $SubT_i$, $Cnt(N)$ vyjadřuje počet dokumentů v uzlu *N*. Ve jmenovateli se objevuje sumární symbol, který sčítá logaritmy všech podstromů pro uzel *N*.

$$P(N|T) = \frac{\ln(1 + Cnt(N))}{\sum_{j=1}^{k} \ln(1 + Cnt(SubT_j)) + \ln(1 + Cnt(N))} \tag{3.11}$$

Jednotlivé části vzorce jsou analogické vzorci 3.10.



Obrázek 3.1: Příklad konceptuální hierarchie s počtem dokumentů v uzlech (převzato z [4])

3.4 Popis klasifikační metody

V této podkapitole si nejprve popíšeme postup, jakým jsou stránky přiřazeny kategoriím pomocí hierarchické klasifikace. Poté se podíváme na zmíněné dynamické rozšiřování struktury o nové kategorie (kdy k němu dochází a jakým způsobem) a na aktualizování informací o kategorii, do které byla přiřazena nová stránka.

3.4.1 Hierarchická klasifikace

Proces klasifikace můžeme rozdělit do čtyř kroků: vytvoření stromu reprezentujícího hierarchickou strukturu kategorií, propagace vlastností ve vytvořené hierarchické struktuře, výběr vhodných vlastností, které jsou určující pro samotnou klasifikaci a nakonec jednocestný průchod.

3.4.1.1 Vytvoření stromu kategorií

Abychom mohli klasifikovat webové stránky, je nutné nejdříve nadefinovat konceptuální hierarchii kategorií, k čemuž využijeme znalostí domény. Dále je potřeba do kategorií přiřadit dokumenty, které do nich přísluší. Charakteristické vlastnosti každé kategorie jsou reprezentovány buď pomocí množiny slov (bag of words), nebo seznamem vlastností vytvořených pomocí well-grained metody popsané výše. Konceptuální hierarchii modelujeme pomocí stromu. Každý uzel představuje kategorii a jeho synovské uzly (jestliže nějaké má) tvoří podkategorie.

3.4.1.2 Propagace vlastností

Propagace vlastností probíhá podle algoritmu popsaného v podkapitole 3.3 od listových uzlů směrem ke kořenu. Výsledný seznam vlastností po provedení propagace, který charakterizuje kategorie, je vytvořen spojením původního seznamu pro daný uzel se seznamem podkategorií a přiřazením různých vah. Díky propagaci jsou charakteristické vlastnosti uzlu přesnější než při reprezentaci množinou oddělených kategorií.

3.4.1.3 Výběr vhodných vlastností

Rozlišování mezi kategoriemi je založeno na vlastnostech, které jsou pro kategorii unikátní či specifické. Díky předchozímu využití propagace jsou tyto unikátní vlastnosti ohodnoceny váhou a rozšířeny směrem vzhůru v hierarchii. Tím pádem se stávají rovněž vlastnostmi rodičovských (nadřazených) kategorií. Správnou kategorii, do které zkoumaná stránka patří, nalezneme následováním těchto unikátních vlastností. A protože se jedná o jedinečné vlastnosti, existuje jen jedna cesta.

Základem výběru vlastností je porovnat vlastnosti jednoho uzlu s ostatními uzly na stejné úrovni hierarchie a určit právě ty vlastnosti, které jsou pro uzel unikátní. Po propagaci je pravděpodobnost každé vlastnosti v uzlu vážený součet pravděpodobností stejné vlastnosti v podstromech a v uzlu samotném. K určení unikátnosti vlastností byla v [2] navržena rovnice 3.12.

$$R_c(w) = \frac{P(w|c) \times P(SubT_c|T)}{P(w|parent)} \tag{3.12}$$

$R_c(w)$ je hodnota unikátnosti vlastnosti w v kategorii c, kde c je aktuální uzel a *parent* je rodičovský uzel pro uzel c. $P(SubT_c|T)$ je hodnota váhy podstromu $SubT_c$.

Pokud je nějaká vlastnost unikátní v jednom uzlu, je to jediný zdroj, který tuto vlastnost může šířit k rodiči. V takovém případě je výsledek vztahu 3.12 roven 1, což je maximální hodnota.

3.4.1.4 Jednocestný průchod

Chceme-li klasifikovat stránku či dokument do nějaké kategorie, musíme porovnat vlastnosti dokumentu s vlastnostmi kategorií, abychom mohli určit, do které patří. Většina algoritmů pro klasifikaci pracuje s množinou kategorií, které nejsou nijak strukturované, a proto musí daný dokument porovnat se všemi kategoriemi.

Naproti tomu ve struktuře, kterou jsme vytvářeli v předcházejících krocích, stačí pouze průchod jednou cestou, aby bylo dosaženo stejného výsledku. Uvážíme-li N různých kategorií, pak složitost porovnávání všech kategorií je $O(N)$, zatímco u jednocestného průchodu $O(log(N))$.

Hlavní myšlenkou jednocestného průchodu je eliminovat ostatní větve ve stromu až na tu jednu správnou. Po propagaci vlastností je seznam vlastností rodičovského uzlu souhrnem propagovaných seznamů vlastností z jeho potomků. Takže při zpracování rodičovského uzlu známe informace o jeho následnících. Díky ohodnocení získaného pomocí vzorce 3.12 má každá kategorie svoje unikátní vlastnosti, pomocí nichž jsme schopni ji odlišit od ostatních uzlů stejné úrovně. Přiřazení do kategorie se děje ve dvou krocích.

Nejprve se na každé úrovni rozliší sourozenecké uzly, a tak se nalezne správná cesta pro zpracovávanou stránku. K určení hodnoty rozlišující pravděpodobnosti každého uzlu použijeme vztah 3.7, avšak pouze na vlastnosti, které mají hodnotu unikátnosti rovnu 1. To znamená, že bereme v úvahu pouze unikátní vlastnosti. Na každé úrovni hierarchie vybereme uzel s nejvyšší rozlišovací pravděpodobností, který se stane výchozím pro další iteraci. Tento postup opakujeme, až narazíme na listový uzel, a tím jsme vytvořili cestu.

Ve druhém kroku již zbývá zařadit stránku do jedné z kategorií, které tvoří získanou cestu. Na každý z uzlů cesty aplikujeme opět vztah 3.7, ale tentokrát použijeme všechny vlastnosti. Uzel s nejvyšší hodnotou pravděpodobnosti je námi hledaná kandidátní kategorie.

3.4.2 Dynamické rozšiřování a aktualizace vlastností kategorie

3.4.2.1 Rozšiřování hierarchické struktury

S přibývajícím množstvím stránek, které klasifikujeme, a díky jejich rozmanitosti nemusí vždy být původní počet kategorií dostatečný, aby se mezi nimi našla vhodná kategorie pro zpracovávanou stránku. Jak se vypořádat s tímto problémem bylo navrženo v [2]. Za pomoci určených statistických výsledků jsou stanovena určitá kritéria a kontroluje se, zda jsou splněna při zařazení stránky do kategorie. Pokud nedojde k jejich splnění, musí se vytvořit nový uzel pro klasifikovanou stránku.

Dojde-li k přidání nového uzlu, může nastat jedna ze dvou variant – rozšíření do hloubky, nebo rozšíření do šířky. Jelikož máme dva typy, musí se stanovit dva různé prahy, které budou kontrolovat splnění kritérií. První práh nazveme *B*, druhý *D* a získáme je následujícím způsobem. Je vybrán vzorek stránek patřících do kategorie a spočítá se pravděpodobnost, s jakou k této kategorii náleží. Tyto hodnoty spočítáme pro všech *n* kategorií a označíme je S_i, kde *i* je v rozsahu *1- n*. Poté spočítáme normální rozložení ze všech S*i* a získáme průměr μ a směrodatnou odchylku *d*. Prahy B a D jsou definovány následovně:

$$B = \mu - d \tag{3.13}$$

$$D = 2d \tag{3.14}$$

Vztah mezi dokumentem a kategorií, do které patří, můžeme označit jako relaci „přísluší“. Základním předpokladem je, že pravděpodobnost příslušnosti stránky do kategorie, se kterou je v relaci „přísluší“, je maximální. Tato hranice se nastaví na nižší hodnotu

K rozšíření do hloubky dojde, pokud maximální pravděpodobnost příslušnosti nové stránky (dokumentu *d*) do kandidátské kategorie je menší než práh B.

$$P(d|c) < B \tag{3.15}$$

To znamená, že ačkoliv je to nejvyšší hodnota pravděpodobnosti, odpovídající kategorie není dostatečně vyhovující pro zpracovávanou stránku. Proto dojde k vytvoření nové podkategorie z kandidátní.

Rozšíření do šířky nastane, když bude rozdíl pravděpodobnosti příslušnosti nové stránky ke kandidátské kategorii a maximální hodnoty pravděpodobnosti příslušnosti do jedné ze synovských kategorií větší, než je spočítaná hodnota prahu D.

$$P(d|c) - Max(P(d|SubT_i)) > D \tag{3.16}$$

Rozdíl mezi hodnotami pravděpodobností reprezentuje rozdílnost mezi stránkou a následníky kandidátní množiny. Pokud bychom tento rozdíl ignorovali, došlo by k porušení konzistence relace příslušnosti mezi rodičem a následníkem. Kdybychom zařadili stránku do kandidátské kategorie, tak při aktualizování charakteristických vlastností (viz dále) by se vlastnosti způsobující rozdíl rovněž podílely na charakterizaci kategorie. Tím by byla relace mezi rodičem a potomkem poškozena. Vytvořením nové kategorie se tento rozdíl vloží mezi uzly na jedné úrovni.

3.4.2.2 Aktualizace charakteristických vlastností

Jakmile je klasifikovaná stránka přiřazena do kategorie, ať už stávající či nové, vektor charakteristických vlastností nové stránky bude mít podíl na charakteristice této kategorie. Je to nutné,

abychom udrželi hierarchickou strukturu. Také možné rozšíření slovníku o nové vlastnosti pravděpodobně změní charakteristiky některých kategorií.

Jak nová stránka, tak kategorie jsou reprezentovány vektorem charakteristických vlastností. Řešením, jak zahrnout charakteristiky stránky do vlastností kategorie, k níž byla přiřazena, je tyto dva vektory spojit. Pro aktualizaci platí vzorec 3.17.

$$P'(w|N) = \frac{|V| \times P(w|N) + |Vpage| \times P(w|Vpage)}{|V| + |Vpage|} \tag{3.17}$$

$|V|$ představuje velikost vektoru vlastností kategorie a $|Vpage|$ velikost vektoru charakteristik stránky.

Po spojení se změní charakteristické vlastnosti kategorie, proto musí být znovu přepočítány hodnoty unikátnosti a propagované pravděpodobnosti vlastností. Jelikož tyto změny se udály pouze v rámci klasifikační cesty, aktualizace probíhá pouze na této cestě.

4 Specifikace a návrh aplikace

Tato kapitola ve svém úvodu shrnuje neformální požadavky na výslednou aplikaci a ve své další části seznamuje s detailním návrhem aplikace včetně různých grafů, které usnadňují orientaci v návrhu.

4.1 Neformální specifikace

Cílem této diplomové práce je vytvoření experimentální aplikace, jež bude implementovat výše uvedenou metodu pro klasifikování webových stránek. Implementačním jazykem je programovací jazyk Java. Bude se jednat o aplikaci s jednoduchým grafickým uživatelským rozhraním, které bude uživateli usnadňovat práci a orientaci v hierarchické struktuře kategorií.

4.1.1 Zpracování HTML dokumentu

Aplikace bude k webovým stránkám přistupovat dvěma způsoby. Prvním z nich je možnost mít soubor s příponou „html“ na lokálním souborovém systému. Pak bude na vstupu cesta k tomuto souboru, se kterým se má manipulovat. Druhou variantou je zadání přímého URL pro danou stránku, která je umístěna kdekoliv na internetu. V takovém případě bude obsah stránky načten přímo ze vzdáleného zdroje.

Samotné zpracování stránky bude spočívat v rozčlenění stránky na jednotlivé oddíly odpovídající typům oddílů z tabulky 3.1. Jakmile budou tyto oddíly vytvořeny, pro každý z nich bude sestavena množina slov, která se v oddílu vyskytují. Do této množiny se nepromítnou všechny termy ve všech tvarech, nýbrž pouze takové, které se nevyskytují v tzv. stop listu a navíc převedeny do základního tvaru pomocí tzv. stemmingu (viz kapitola 3.1). V závislosti na typu oddílu, ve kterém se term vyskytuje, bude spočítána jeho frekvence výskytu. K tomuto účelu poslouží váhy přiřazené oddílům. Aplikace umožní uživateli použít přednastavené váhy, nebo nastavit váhy podle potřeb. Každý výskyt bude násoben váhou příslušného oddílu. Vektor vlastností s počtem jejich výskytů, se získá spočtením frekvence výskytů každého termu napříč všemi oddíly.

4.1.2 Vytvoření hierarchické struktury

Základem celé aplikace je hierarchická stromová struktura, která reprezentuje jednotlivé kategorie. Při vytváření hierarchické struktury bude vstupem cesta ke kořenovému adresáři. Tento adresář bude kořenovým uzlem odpovídajícím hlavní kategorii. Jeho podadresáře pak reprezentují příslušné podkategorie. Pro každý adresář vznikne nový uzel ve stromové struktuře nesoucí název podle názvu adresáře. Stránky obsažené v adresáři budou zpracovány podle výše uvedeného postupu. Vzniklá kategorie tak bude reprezentována součtem vlastností všech stránek patřících do této kategorie.

4.1.3 Propagace vlastností hierarchickou strukturou a určení jejich unikátnosti

Po vytvoření základní reprezentace kategorií se provede propagace vlastností směrem od listových uzlů ke kořenu (popsáno v kapitole 3.3). To znamená, že vektor reprezentující kategorii bude sestávat

z vektoru vlastností odpovídajícího uzlu a z propagovaných vektorů vlastností všech jeho podkategorií. Vlastnosti se budou propagovat rekurzivně tak, že pro každý uzel se pomocí vzorce 3.8 spočítá propagovaná váha vlastnosti. Pokud budou v uzlu některé vlastnosti takové, že se nepropagovaly z žádného podstromu, i pro ně se provede výpočet podle vzorce 3.8. Každý uzel (kategorie) uchovává informaci o své váze, počtu dokumentů, které do něj patří, o váze podstromu, který má kořen v tomto uzlu a o celkovém počtu dokumentů v tomto podstromu.

V dalším kroku je potřeba pro každou vlastnost v kategorii určit její unikátnost. Tento krok se provede postupným průchodem všech kategorií a spočítáním unikátnosti všech vlastností, které ji reprezentují podle vztahu 3.12. Po dokončení tohoto kroku je hierarchická struktura připravena pro klasifikaci nových stránek.

4.1.4 Výpočet prahů

Po vytvoření hierarchické struktury je nutné získat hodnoty prahů B a D popsané v kapitole 3.4.2. K tomuto účelu se zvolí reprezentativní množina jedné čtvrtiny stránek z každé kategorie (pokud kategorie obsahuje méně než 5 stránek, pak jsou vybrány všechny). Pro každou stránku se spočítá pravděpodobnost, s jakou přísluší do dané kategorie. Pravděpodobnosti všech reprezentativních stránek dané kategorie se zprůměrují, a tím je získána hodnota pravděpodobnosti pro kategorii. Stejný postup se aplikuje na všechny kategorie. Jakmile jsou spočítány pravděpodobnosti všech kategorií, spočítáme průměr těchto hodnot μ a poté spočítáme směrodatnou odchylku d. Z těchto hodnot jsou pak vypočítány prahy B a D.

4.1.5 Proces klasifikace

Klasifikace webové stránky začíná krokem zpracování vstupní stránky, jak je popsáno výše – vytvořením vektoru vlastností, které stránku reprezentují. Samotný proces klasifikace začíná od kořenového uzlu stromové struktury. Pro všechny jeho synovské uzly se spočítá podobnost mezi zkoumanou stránkou a kategorií. Podobnost se zjistí pomocí výpočtu podle vzorce 3.7 za použití vlastností, které byly ohodnoceny jako unikátní. Synovský uzel, který má nejvyšší podobnost se zkoumanou stránkou je zařazen do cesty (od kořene k listu), kterou pro danou stránku hledáme. Stejným způsobem aplikace pokračuje, dokud nedospěje do listového uzlu, kde výpočet cesty končí. Jakmile je kompletní cesta spočítána, pro každý uzel v ní obsažený se spočítá podobnost s využitím všech vlastností reprezentujících daný uzel. Uzel s nejvyšší podobností se stává kandidátním uzlem pro přiřazení zkoumané stránky.

Po získání kandidátního uzlu je nutné ověřit, zda tento uzel splňuje podmínky a stránka do této kategorie může byt zařazena, nebo bude nutné vytvořit kategorii novou. Toto určení je závislé na hodnotách prahů B a D a vztazích 3.15 a 3.16. Pokud je třeba vytvořit kategorii novou, aplikace upozorní uživatele a vyzve jej ke spolupráci na zvolení jména nové kategorie.

Jakmile je stránka přiřazena do kategorie, provede aplikace poslední krok – zahrnutí vlastností klasifikované stránky mezi vlastnosti kategorie, do které byla klasifikována. To bude provedeno sloučením vektoru vlastností kategorie s vektorem vlastností stránky stejným způsobem jako při vytváření hierarchické struktury kategorií. Po sloučení vektorů se pro každý uzel na vzniklé cestě znovu přepočítají váhy uzlu a pravděpodobnosti vlastností, které uzel reprezentují.

Jako zpětná vazba uživateli po klasifikaci je zobrazena struktura kategorií s tím, že v hierarchické struktuře je vybrána právě ta kategorie, která byla klasifikátorem zvolena. Pokud je

klasifikovaná stránka reprezentována méně než padesáti různými vlastnostmi, pak je uživatel informován o možné nepřesnosti klasifikace z důvodu nižšího počtu vlastností stránky.

4.2 Návrh aplikace

Návrh aplikace se dá rozdělit na dvě části. První část se zabývá grafickým uživatelským rozhraním, druhá pak rozebírá návrh logiky a funkcionality klasifikátoru.

4.2.1 Grafické uživatelské rozhraní

Na uživatelské rozhraní jsou dva hlavní požadavky – aby se dalo jednoduše nastavit hodnoty pro klasifikaci, a aby byla uživateli prezentována hierarchická struktura tvořená kategoriemi. Jelikož výsledkem klasifikace je opět kategorie, nabízí se rozhraní rozdělit na dvě části odpovídající uvedeným požadavkům. Stěžejní pro aplikaci bude hlavní okno, ve kterém se bude měnit jeho obsah.

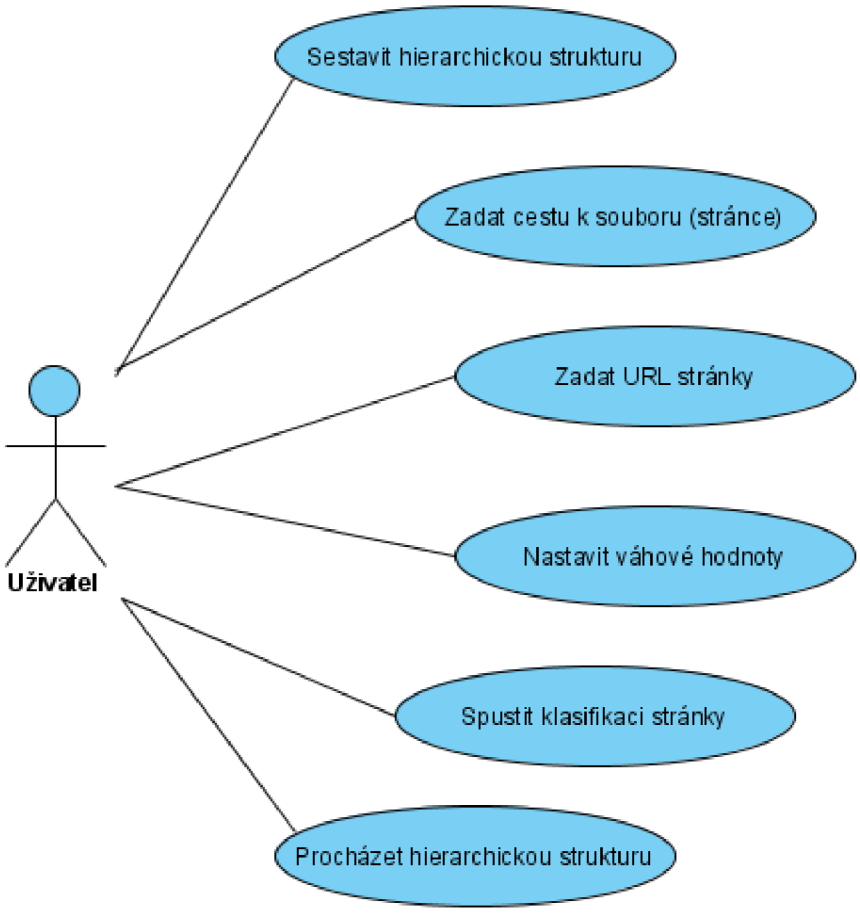
Úvodní okno bude rozdělené na levou a pravou část. Levá část bude sloužit k zadávání stránek ke klasifikaci, ke spouštění sestavování hierarchické struktury a samotné klasifikace. Pod všemi těmito uživatelskými vstupy budou souhrnné informace o hierarchické struktuře, byla-li již nějaké vytvořena. Mezi těmito informacemi bude celkový počet kategorií a dokumentů v nich obsažených, počet různých termů ve slovníku a prahu B a D. Na pravé straně okna bude nastavení vah pro jednotlivé části stránky. Uživatel bude moci zadat hodnoty od jedné do desíti.

Druhé okno bude sloužit k zobrazení stromové struktury. Mezi oběma okny bude možné pomocí tlačítek libovolně přecházet. Pouze pokud ještě nebude sestavena žádná hierarchická struktura, nebude přechod z úvodního okna uživateli umožněn. Kategorie budou zobrazeny jako strom v levé části okna. Na pravé straně se pak zobrazí informace o vybrané kategorii – počet dokumentů patřících do této kategorie, počet dokumentů v podstromu vycházejícího z této kategorie, váhový faktor uzlu a váhový faktor podstromu. Pod těmito informacemi se uživateli zobrazí tabulka vlastností, které tuto kategorii charakterizují. U každé z nich je uveden její název, počet výskytů, propagovaná váha a hodnota unikátnosti této vlastnosti.

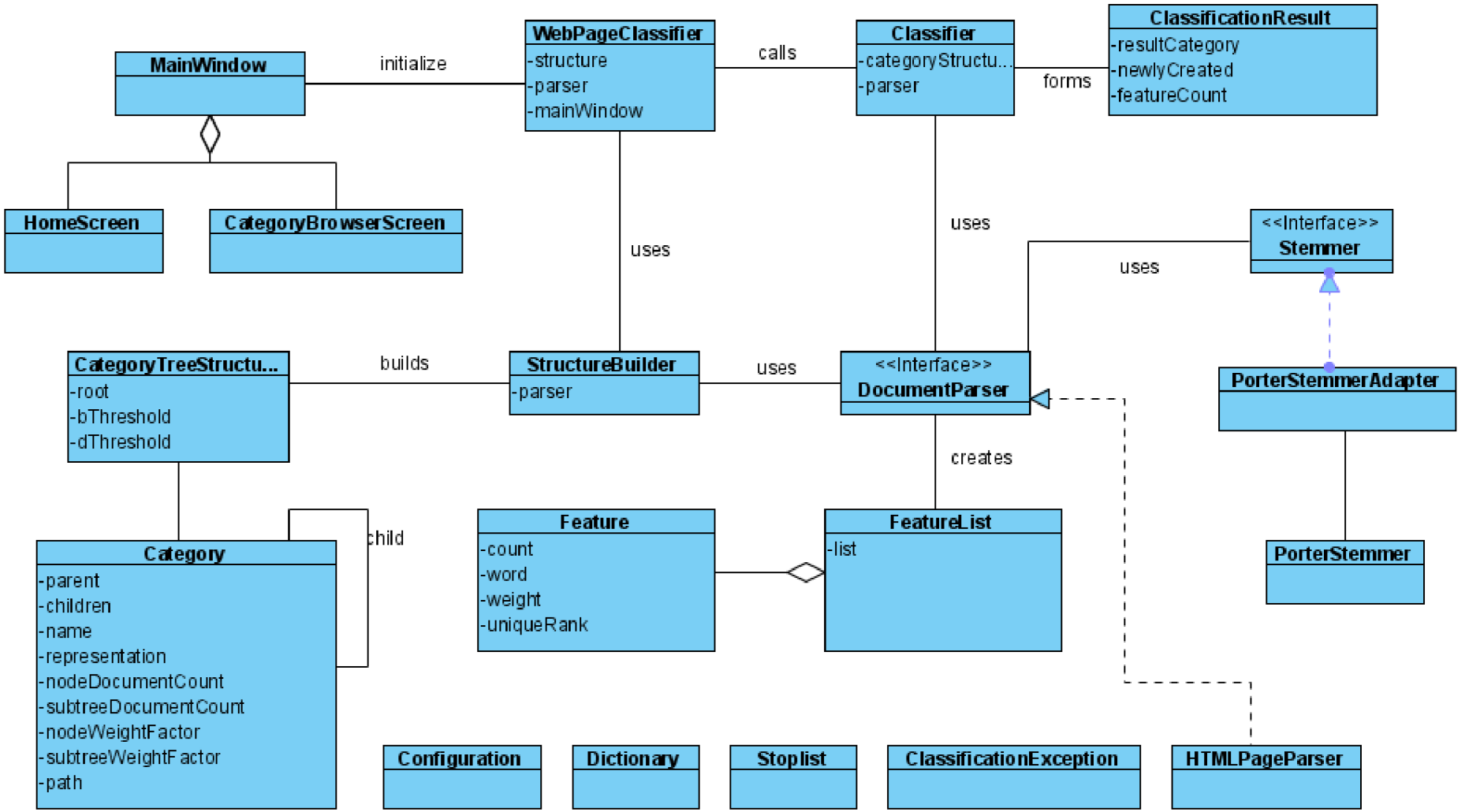
Zadávání názvu nové kategorie bude probíhat v modálním okně. Rovněž v případě, že klasifikovaná stránka má nízký počet vlastností, bude informační hláška zobrazena v modálním okně.

4.2.2 Návrh klasifikátoru

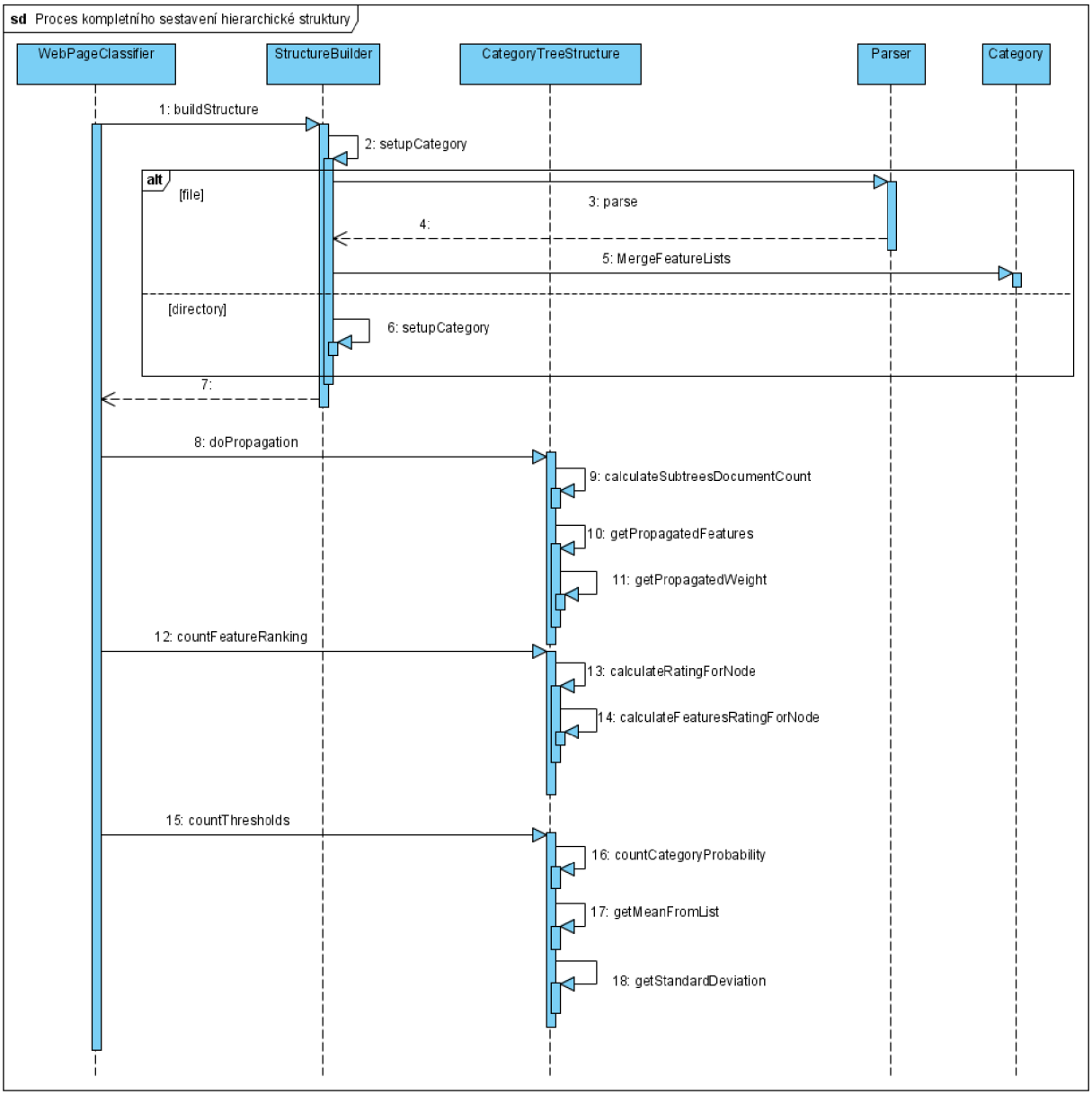
Rozhraní mezi grafickým uživatelským rozhraním a aplikační logikou je tvořeno pomocí jedné třídy a tou je *WebPageClassifier*, která aplikaci spouští. Pro uchování nastavení uživatele slouží třída *Configuration*, která v sobě bude uchovávat rovněž základní přednastavené hodnoty. Uživatelem nastavené hodnoty nebudou perzistentní, takže nepřežijí ukončení aplikace. Stěžejními třídami pro klasifikátor jsou *StructureBuilder*, *CategoryTreeStructure*, *HTMLPageParser* a *Classifier*. Tyto budou zaštiťovat většinu datových struktur a logiky celého klasifikátoru. K zachytávání různých chyb a výjimek poslouží jedna třída *ClassificationException*. Dojde-li k nějaké chybě, bude uživateli zobrazeno, k jakému problému došlo. Podrobnosti návrhu jsou vyobrazeny na následujících diagramech. Detailní popis funkcionality a průběhu procesu klasifikace je rozebraný dále v kapitole 5.



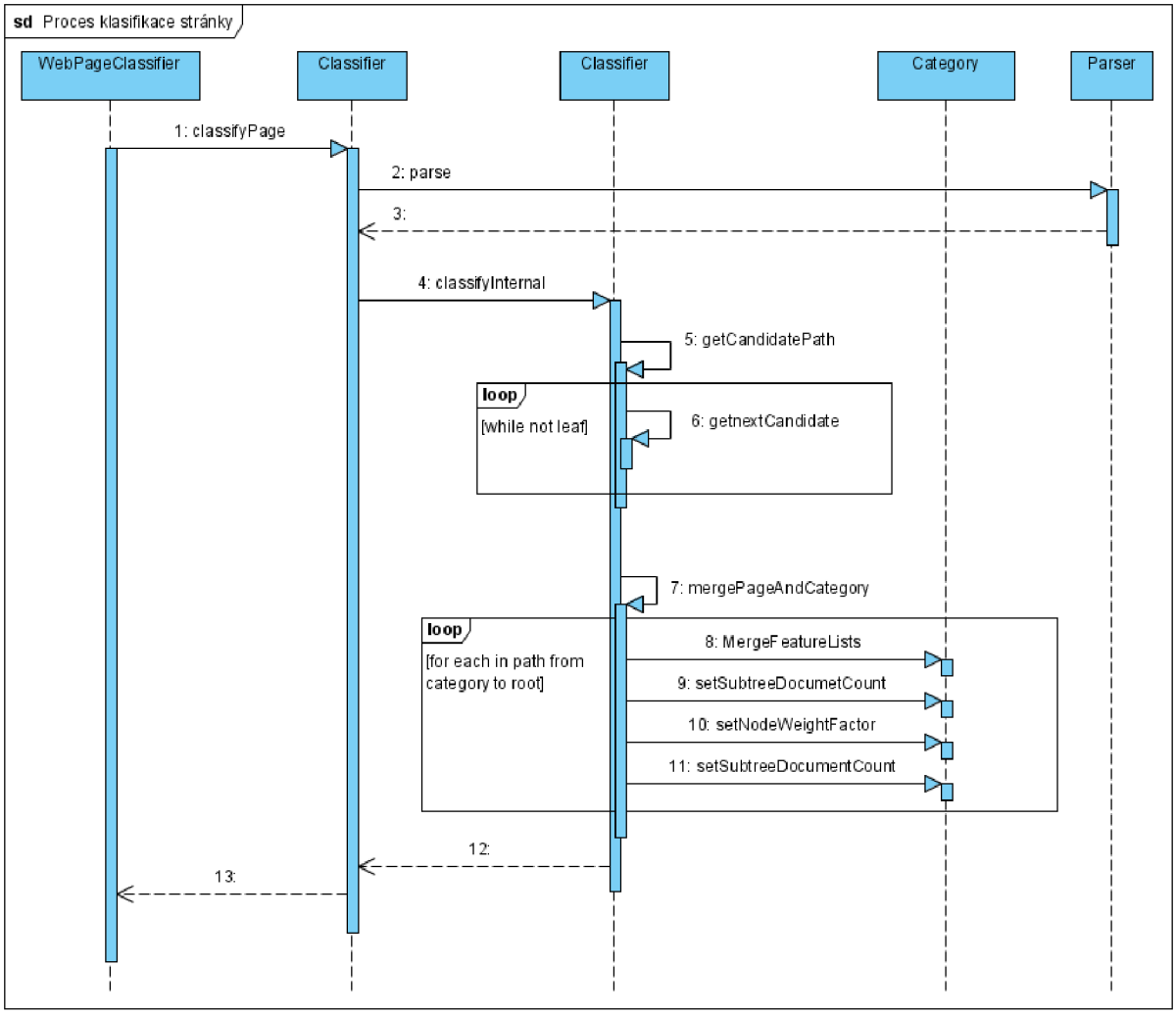
Obrázek 4.1: Diagram užití



Obrázek 4.2: Diagram tříd



Obrázek 4.3: Sekvenční diagram popisující sestavení hierarchické struktury



Obrázek 4.4: Sekvenční diagram procesu klasifikace webové stránky

5 Implementace

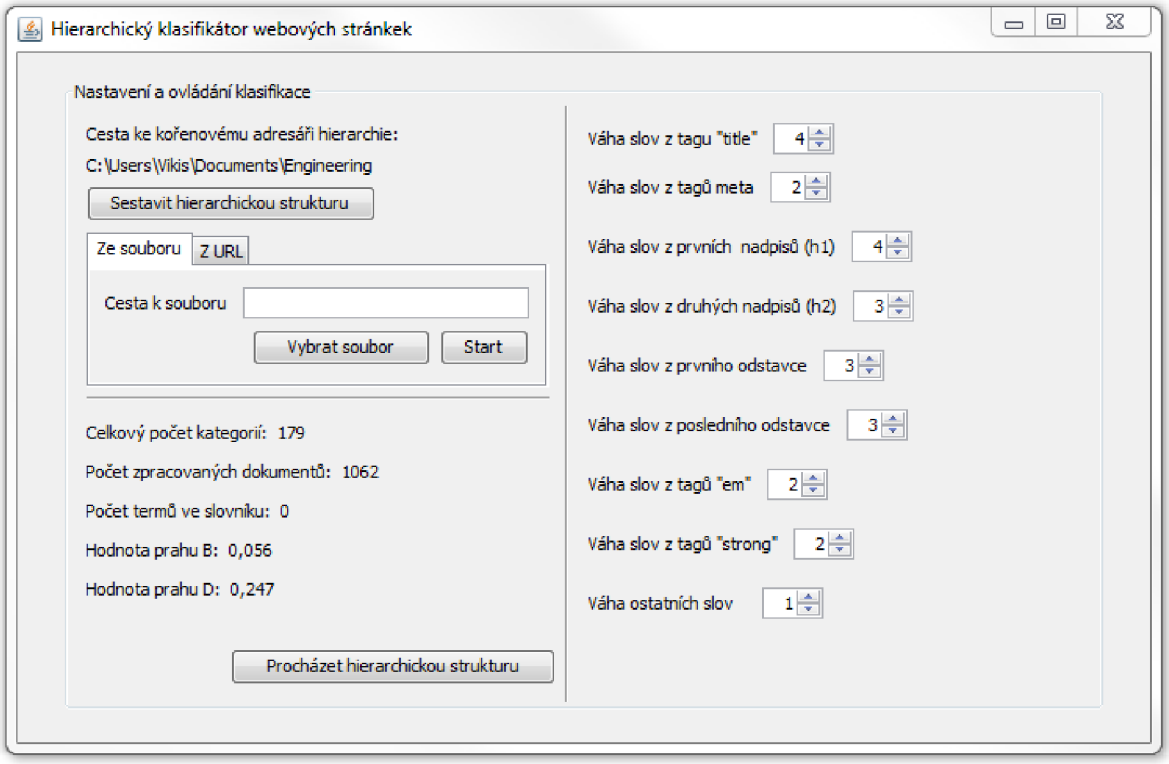
V páté kapitole je rozebrána implementace celé aplikace. Zpočátku se ve zkratce podíváme na implementaci uživatelského rozhraní. Naopak implementaci klasifikátoru si přiblížíme podrobně, protože se nejedná o triviální problém a některé části by nemusely být zcela zřejmé. V následujícím textu jsou názvy tříd a metod psány kurzívou, pro lepší odlišení od ostatního textu. Rovněž metody jsou uváděny ve formátu, kdy název metody následují závorky, bez parametrů a návratových hodnot. Závorky umožňují odlišit metodu od třídy na první pohled.

5.1 Implementace uživatelského rozhraní

Aplikace je spouštěna prostřednictvím třídy *WebPageClassifier*, která odděluje uživatelské rozhraní od aplikační logiky. Grafické uživatelské rozhraní je koncipováno tak, že při spuštění aplikace je vytvořeno jedno okno, které v průběhu užívání aplikace mění svůj obsah. Toto okno je reprezentováno třídou *MainWindow*, která dědí od *JFrame*. Mezi funkcionalitu okna samozřejmě patří přepínání zobrazovaného obsahu a také vytvoření modálního okna v případě, že v aplikaci došlo k nějaké chybě.

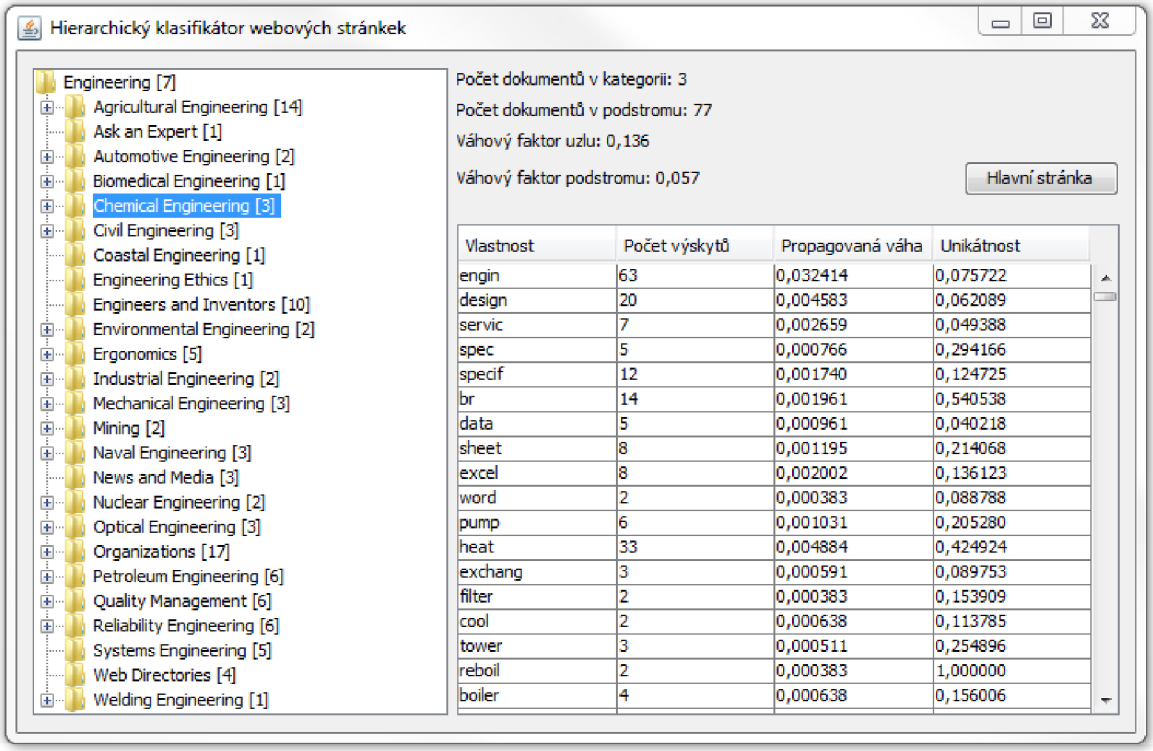
Měnící se obsahy okna jsou dva – *HomeScreen* a *CategoryBrowserScreen*. Obě tyto třídy jsou potomky grafické komponenty *JPanel*. Protože úvodní obrazovka zdaleka neobsáhne celou obrazovku počítače, je zmenšeno na potřebnou velikost a právě při tomto zobrazení není povoleno měnit velikost okna. Naopak při procházení kategorií je z důvodu velkého množství vlastností kategorie vhodné okno maximalizovat, proto s přepnutím na tuto obrazovku dochází k povolení měnit velikost okna.

Na obrázku 5.1 vidíme konečnou podobu úvodní obrazovky. Volba, zda chceme klasifikovat stránku ze souboru, nebo z URL, je implementována pomocí přepínacích záložek.



Obrázek 5.1: Úvodní obrazovka aplikace

Obrázek 5.2 představuje druhou z obrazovek – procházení hierarchické struktury. Pro zobrazení stromové struktury s kategoriemi slouží grafická komponenta *JTree*. Číslo v hranaté závorce za názvem kategorie udává počet dokumentů v kategorii. Jakmile uživatel vybere ze stromu nějakou kategorii, informace napravo se aktualizují podle zvolené kategorie.



Obrázek 5.2: Obrazovka procházení hierarchické struktury

5.2 Implementace klasifikátoru

Popis implementace samotného klasifikátoru zaujímá většinu oproti grafickému uživatelskému rozhraní. Proto si ji přiblížme rozdělenou do několika částí. Nejdříve si rozebereme důležité třídy z pohledu reprezentace uchovávaných dat. V dalších částech nahlédneme do implementace vlastní funkcionality od připravení struktury po klasifikaci stránky tak, jak se aplikace používá.

5.2.1 Datové struktury

Jak jsme se již mohli v předcházejícím textu dočíst, jednotlivé webové stránky jsou reprezentovány pomocí termů respektive vlastností. Jednotlivé vlastnosti jsou v implementaci aplikace reprezentovány třídou *Feature*. Každá vlastnost uchovává čtyři informace o sobě sama – název, počet výskytů, váhový faktor a hodnotu své unikátnosti. Vyskytuje-li se daná vlastnost v dokumentu víckrát, nevytváří se další instance, nýbrž se zvyšuje její počet výskytů. Ale o tom, stejně jako o použití váhy a unikátnosti, až dále. Aby byla reprezentace ať už dokumentu či stránky kompaktní, je zapouzdřena do třídy *FeatureList*. Nejedná se o nic složitějšího, než o seznam jednotlivých vlastností, ale umožňuje některé operace, které obyčejný seznam neumožňuje. Nejdůležitější pro celý proces klasifikace je hierarchicky uspořádaná stromová struktura, kde jednotlivé uzly odpovídají kategoriím. Pro reprezentaci stromové struktury je implementována třída *CategoryTreeStructure*. Jejím podstatným prvkem je kořenová

kategorie, jejíž popis následuje vzápětí. Dále v sobě uchovává oba dva prahy podstatné pro rozhodnutí o vytvoření nové kategorie.

Základním stavebním kamenem v této struktuře je kategorie *Category*. Každá kategorie nese odkaz na svoji rodičovskou kategorii (s výjimkou kořenové) a seznam svých synovských kategorií, čímž je tvořena stromová struktura. Kromě uvedených prvků v sobě uchovává též reprezentaci charakteristických vlastností pro danou kategorii v podobě *FeatureList*, počet dokumentů patřících přímo do kategorie, počet dokumentů v podstromu, jehož kořen je právě tato kategorie, váhový faktor kategorie a podstromu s počátkem v kategorii. Dále obsahuje název kategorie a cestu k adresáři, ze kterého byla vytvořena, nevznikla-li až při klasifikaci stránky.

5.2.2 Zpracování webové stránky

Pro zpracování stránky slouží třída *HTMLPageParser*, která implementuje rozhraní *DocumetnParser*. Toto rozhraní poskytuje jedinou metodu a to je *parse()*. Díky zvolené implementaci pomocí rozhraní není v budoucnu žádný problém nahradit *HTMLPageParser* jiným, takže tento klasifikátor nemusí pracovat pouze s dokumenty ve formátu HTML, ale je možné jej využít pro jakékoliv dokumenty pouze doimplementováním příslušného parseru a jeho užitím. Metoda *parse()* bere na vstupu *InputSream* a jejím výsledkem je reprezentace dokumentu v podobě *FeatureList*.

HTMLPageParser neimplementuje parsování dokumentu od základu, ale využívá již existující knihovnu „Jericho HTML Parser“ pro analyzování a manipulaci s HTML stránkami, nebo jejich částmi. Tato knihovna je volně dostupná a více její dokumentaci, stažení a další informace lze nalézt v [13].

Díky použité knihovně, která umožňuje získávat HTML elementy podle jejich jména, probíhá zpracování následovně. Nejprve se z hlavičky získají elementy head a title a z jejich atributů se získají hodnoty. Z elementu body se získávají postupně elementy odpovídající tabulce 3.1 a po jejich zpracování jsou z něj odstraněny, takže nakonec zbývá obyčejný text. Knihovna umí vrátit textový obsah elementů, takže poté již zbývá takový obsah zpracovat. Text je rozdělen pomocí oddělovačů na slova. Pokud se slovo nenachází ve stop listu, je převedeno na základní tvar. Stop list je implementován jednoduchou třídou *StopList*, která obsahuje seznam častých anglických slov (seznam slov převzat z [14]) a jednu metodu na zjištění, zda se zadané slovo v tomto seznamu vyskytuje. Rozhraní *Stemmer* nabízí metodu *stemm()*, která převezme slovo v podobě řetězce a vrátí jeho základní tvar. Pro tento účel byl zvolen Porterův algoritmus. Třída *PorterStemmer* byla převzata z [15] a jelikož samozřejmě neimplementuje rozhraní *Stemmer*, byl vytvořen adaptér *PorterStemmerAdapter* řešící tento problém. Použití rozhraní stejně jako v případě *DocumentParser* nabízí možnost kdykoliv snadno využít jiný algoritmus. Jakmile je slovo v základním tvaru, vytvoří se objekt typu *Feature* a jako počet se nastaví váha oddílu, odkud bylo slovo získáno.

Zbývá již jen zařadit slovo do reprezentace stránky. K tomuto účelu slouží ve třídě *FeatureList* metoda *addToList()*, která zjistí, zda stejnou vlastnost (vlastnost se stejným názvem) již neobsahuje. Pokud ano, pouze jí zvýší počet výskytů o počet výskytů přidávané vlastnosti, jinak se zpracovávaná vlastnost přidá do seznamu.

5.2.3 Sestavení hierarchické struktury

O vytvoření stromové struktury se stará třída *StructureBuilder* a její metoda *buildStructure()*. Ta vyžaduje na vstupu cestu ke kořenovému adresáři odpovídajícímu kořenové kategorii. Rekurzivním voláním metody *setupCategory()* se zpracuje celý obsah adresáře. Pokud se během zpracování

kategorie narazí na další adresář, vytvoří se nová podkategorie a začne se znovu zpracovávat. Ve druhém případě – jedná-li se o soubor – se pomocí parseru vytvoří reprezentace tohoto souboru a pomocí metody *mergeFeatureLists()* třídy *Category* se přidá reprezentace souboru k již stávající reprezentaci zpracovávané kategorie. Za zmínku stojí, že při spojování se do reprezentace kategorie nedostanou vlastnosti, které nesplní minimální počet výskytů, který je nastaven na hodnotu váhy obyčejného textu z tabulky 3.1.

Dále je potřeba propagovat vlastnosti všech kategorií směrem vzhůru hierarchickou strukturou a přitom nastavovat jejich váhu. O tento krok se stará metoda *doPropagation()* ve třídě *CategoryTreeStructure*. Pro výpočet vah vlastností je nutné znát váhy podstromů. Proto tato metoda ve své první části volá *icalculateSubtreesDocumentCount()*, která spočítá rekurzivně pro každou kategorii váhu podstromu z ní vycházejícího. Propagaci vlastností zprostředkovává opět rekurzivní metoda *getPropagatetFeatures()*. Ta navíc zajistí i nastavení váhových faktorů kategorií a podstromů. Jedná-li se o listový uzel (kategorii), nastaví se váha uzlu podle vzorce 3.9. V opačném případě se do kategorie nejprve přidají veškeré vlastnosti ze synovských kategorií, které ještě neobsahuje, a poté dojde ke spočítání váhy vlastností. Pokud se vlastnost v uzlu nevyskytuje a je pouze propagovaná ze synovských kategorií, počet výskytů takové vlastnosti v dané kategorii je nastaven na 0.

O určení unikátnosti vlastností se stará metoda *countFeatureRanking()*. Prochází postupně všechny uzly a v každém z nich pomocí metody *calculateRatingForNode()* nastaví vlastnostem hodnoty unikátnosti.

Ke kompletně připravené struktuře připravené pro klasifikaci zbývá poslední krok – výpočet prahů B a D. Metoda *countCategoryProbability()* rekurzivním voláním sestaví seznam průměrných podobností každé kategorie a reprezentativního vzorku do ní příslušejících stránek. Jak se získává podobnost stránky a kategorie je popsáno dále v textu v kapitole 5.2.4. Z tohoto seznamu jsou pomocí metod *getMeanFromList()* a *getStandardDeviation()* spočítány oba prahy.

5.2.4 Klasifikace stránky

Celý proces klasifikace zajišťuje třída *Classifier*. Metoda *classifyPage()* má dvě podoby, které se liší přijímanými parametry – URL v případě klasifikace stránky z internetu respektive řetězec s názvem souboru pro případ stránky ze souboru. Liší se pouze vytvořením zdroje dat pro parser. Obě metody pak volají stěžejní metodu *classifyInternal()*.

Než si popíšeme samotnou klasifikaci, podívejme se, jak se získává míra podobnosti mezi kategorií a dokumentem. Hodnota podobnosti je spočítána metodou *getSimilarityPrTFIDF()*, která odráží výpočet rovnice 3.7. Obdobou této metody je *getSimilarityPrTFIDFUnique()*, která se od ní liší, jak napovídá název, tím, že pracuje pouze s unikátními vlastnostmi kategorie.

Vraťme se k popisu *classifyInternal()*. V prvním kroku je získána cesta od kořenové kategorie k listové. Postupuje se od kořene a na každé úrovni se pro všechny kategorie spočítá míra podobnosti se stránkou pomocí *getSimilarityPrTFIDFUnique()*. Kategorie s nejvyšší mírou podobnosti je přidána do této cesty. Dále je z cesty vybrán kandidát s nejvyšší mírou podobnosti, která je spočítaná pomocí *getSimilarityPrTFIDF()*. Pokud není splněná některá z podmínek 3.15 a 3.16., je vytvořena nová kategorie, prozatím beze jména.

Přidání vlastností klasifikované stránky a aktualizace struktury je implementováno metodou *mergePageAndCategory()*. Pomocí metody *mergeFeatureLists()* třídy *Category* dojde ke sloučení reprezentací stránky a výsledné kategorie. Tím se dostanou nové vlastnosti ze stránky mezi vlastnosti kategorie. U těch vlastností, které kategorie již obsahovala, dojde k navýšení počtu výskytů. Pomocí rovnice 3.17 se přepočítají váhy vlastností ve výsledné kategorii. Dále se získá cesta od výsledné

kategorie ke kořenové a pro každou kategorii na této cestě se přepočítá váhový faktor kategorie, váhový faktor podstromu, a váhové faktory podstromu pro všechny synovské kategorie. Zároveň pokud má zpracovávaná kategorie z cesty rodiče, tak se mu přidají všechny vlastnosti, které neobsahuje. To zajišťuje propagaci nových vlastností směrem vzhůru a je možné rovnou přepočítat váhy vlastností, protože máme zajištěno, že kategorie obsahuje i ty nové a budou také přepočítány. Nakonec se kategorie na této cestě projdou znovu a pro všechny vlastnosti se přepočítají hodnoty unikátnosti. Po všech těchto krocích je struktura ve stejném stavu, jako kdyby klasifikovaná stránka byla součástí kategorie již při sestavování hierarchické struktury.

Výsledek klasifikace je zapouzdřen ve třídě *ClassificationResult*. Ta v sobě uchovává výslednou kategorii, informace o tom, jestli byla tato kategorie nově vytvořena a počet vlastností, kterými byla stránka reprezentována. Na základě toho může být uživatel případně požádán o zadání názvu nové kategorie, případně informován o nízkém počtu reprezentativních vlastností klasifikované stránky.

5.2.5 Zpracování chyb a konfigurace

Pro zachytávání veškerých výjimek a chybových stavů slouží třída *ClassificationException* dědící od standardní výjimky *Exception* v jazyce Java. Informace o chybě v ní uchované se pak zobrazí uživateli pomocí modálního okna.

Nastavení klasifikace je uloženo ve třídě *Configuration*, včetně přednastavených hodnot při startu aplikace. Třída *Dictionary* slouží k uchování informace o všech různých vlastnostech, které se v celé struktuře kategorií vyskytují. Třídy *Configuration*, *Dictionary* a *StopList* jsou implementovány podle návrhového vzoru singleton, takže je vytvářena pouze jedna instance a jsou přístupné z kteréhokoliv místa aplikace.

6 Experimenty a testování

V této kapitole jsou popsány experimenty, kterými se zkoušela funkčnost a správnost klasifikátoru. Pro tyto experimenty bylo nutné obstarat nějaká testovací data – stránky již zařazené do kategorií tvořících hierarchickou strukturu. Taková data obsahuje například adresář Yahoo! (Yahoo! Directory).

Pro experimenty jsem tedy stáhl téměř celý obsah adresáře Directory > Science > Engineering. Stahovány byly takové stránky a kategorie, které nejsou odkazem do jiné kategorie adresáře Yahoo!. Celkem bylo staženo 1062 HTML stránek ve 178 kategoriích. Data byla ukládána do adresářů odpovídajících zdrojovému uspořádání z Yahoo!, tudíž ve formátu, který přesně odpovídá vstupu aplikace pro sestavení hierarchické struktury.

6.1 Test správnosti zařazení stránky

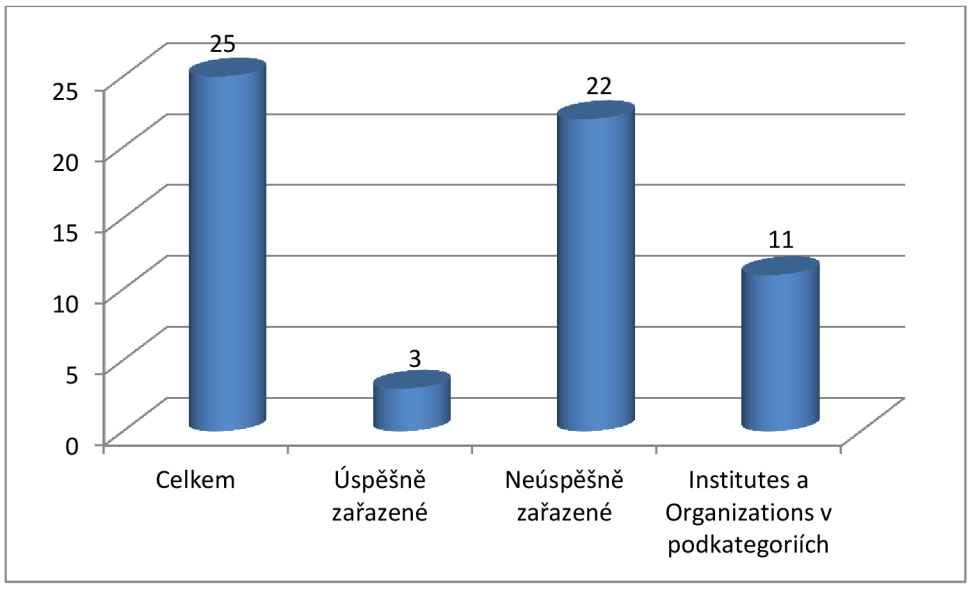
6.1.1 Popis experimentu

Tento experiment má za cíl zjistit, zda klasifikátor klasifikuje stránku správně. U hierarchického klasifikátoru však není jednoduché jednoznačně určit, kdy je klasifikace považována za úspěšnou, či naopak. Je stránka zařazená v konceptuální hierarchii o úroveň výš klasifikovaná správně nebo špatně?

Řekněme, že za správně zařazenou stránku budeme považovat takovou stránku, která spadá do kategorií v rozmezí jedné úrovně od předpokládané správné kategorie. Z toho vyplývá, že odpověď na předchozí otázku zní ano.

Vstupní množinou dat pro experiment je kategorie Engineering/Civil Engineering. Ze všech stránek, které tato kategorie obsahuje, náhodně vybereme 25. Pomocí těchto stránek budeme testovat úspěšnost. Ze zbylých dat necháme sestavit hierarchickou strukturu potřebnou pro klasifikaci.

6.1.2 Výsledky experimentu



Obrázek 6.1: Výsledky testu úspěšnosti

Jak je možné vidět na obrázku 6.1, na první pohled se zdá, že výsledek testu je velmi špatný. Ale podívejme se na výsledek podrobněji. Pokud se podíváme na data, se kterými klasifikátor pracoval, všimneme si, že jednou z podkategorií kořenové kategorie je kategorie s názvem Institutes. Tato kategorie obsahuje jednoznačně největší počet dokumentů. Přesně je to 56 dokumentů, což je více než třetina celkového počtu dokumentů v celé struktuře (struktura po iniciálním sestavení obsahuje 134 zpracovaných dokumentů). Prozkoumáme-li i ostatní podkategorie, zjistíme, že i v nich se vyskytují kategorie s názvem Institutes, avšak na nižších úrovních hierarchie. Stejný problém je i s kategoriemi s názvem Organizations.

Byla-li nějaká stránka v kategorii instituce na nižších úrovních hierarchie, pak byla pohlcena kategorií Engineering/Civil Engineering/Institutes. Počet dokumentů a vyšší úroveň hierarchie převážily. Nutno podotknouti, že v případě pohlcení došlo k rozšíření do hloubky o novou kategorii. Klasifikátor měl také velké problémy s rozpoznáním organizace a instituce, které bylo rovněž způsobeno větším počtem dokumentů v kategorii Engineering/Civil Engineering/Institutes a faktem, že instituce a organizace od sebe v reálném životě nemají nikterak daleko. Z obrázku lze vyčíst, že celkem 11 stránek bylo v jedné z těchto dvou problematických kategorií.

Za zmínku stojí i stránka, která by měla patřit do kategorie Engineering/Organizations. Její název zní „Association of State Dam Safety Officials“. Tato stránka byla zařazena do kategorie Civil Engineering/Dams and Reservoirs. Jelikož tato stránka v původní kategorii byla jediná, která se týkala přehrad a byla zvolena jako testovací, tudíž v kategorii organizací není o přehradách ani zmínka, je zcela logické zařazení klasifikátoru do kategorie spojené s přehradami, ačkoliv je označeno jako chybné.

6.2 Experiment s dvojím rozhodnutím o správnosti

6.2.1 Popis experimentu

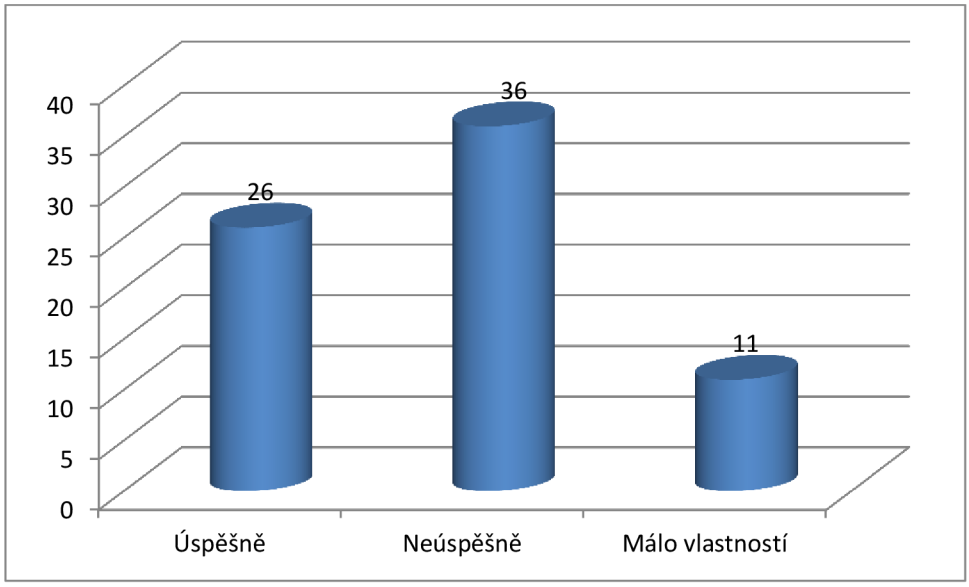
Další experiment bude obdobný předcházejícímu testu. Hlavní odlišnost bude ve vyhodnocování úspěšnosti a v rozdělení dat. Pro tento experiment bylo využito stránek z kategorie Engineering/Mechanical Engineering. Data byla rozdělena následovně. Ze všech kategorií a podkategorií, které obsahovaly ve svém adresáři tři a více stránek byla vždy třetina použita k testování a dvě třetiny k sestavení hierarchické struktury. Sestavená struktura obsahovala 42 kategorií a v nich celkem 194 stránek. Pro testovací účely bylo vyčleněno 73 stránek.

Experiment bude podléhat dvojímu hodnocení. První hodnocení můžeme označit jako striktní, druhé jako mírné. Striktní hodnocení dělí výsledky na úspěšné – stránka byla zařazena přesně do předpokládané kategorie, nebo v limitu jedné úrovně, neúspěšné – došlo ke špatnému zařazení a nehodnocené – stránky s malým počtem reprezentativních vlastností (méně než 50).

Druhé, mírné, hodnocení považuje za dobré zařazení i takové, které zařadilo stránku z instituce či organizace do kategorie Institutes nejvyšší úrovně.

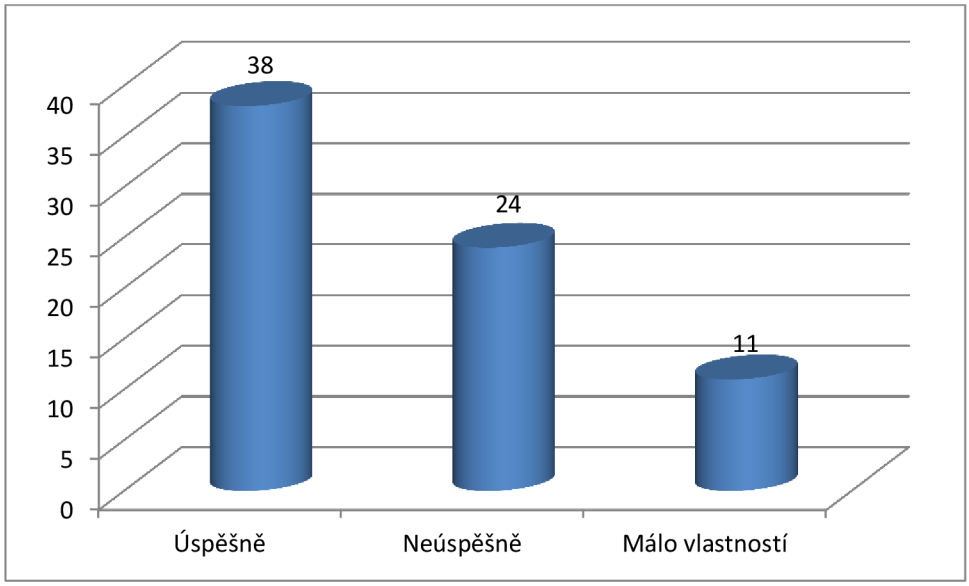
6.2.2 Výsledky experimentu

Výsledky striktního hodnocení vidíme na obrázku 6.2. Málo vlastností mělo 11 stránek z celkového počtu 73 stránek. Nejvíce bylo neúspěšných pokusů, a sice 36. Celková úspěšnost klasifikátoru při striktním hodnocení při pominutí stránek s malým počtem vlastností je 41,94%.



Obrázek 6.2: Výsledky při striktním hodnocení

Podíváme-li se na obrázek 6.3, který ukazuje výsledky klasifikace při mírném hodnocení, vidíme, že úspěšně zařazených stránek je více než neúspěšných. Úspěšnost klasifikace v tomto případě dosáhla hodnoty 61,29%. Přestože se úspěšnost zvýšila, dalších 7 z neúspěšných stránek bylo pohlceno kategorií institucí, což výsledek zhoršuje.



Obrázek 6.3: Výsledky při mírném hodnocení

6.3 Vytvoření nové kategorie

6.3.1 Zadání experimentu

V tomto experimentu se podíváme, s jakou úspěšností se daří vytvářet nové kategorie. Zvolíme malou reprezentaci základní struktury, např. Engineering/Civil Engineering/Structural a sestavíme hierarchickou strukturu pro klasifikaci.

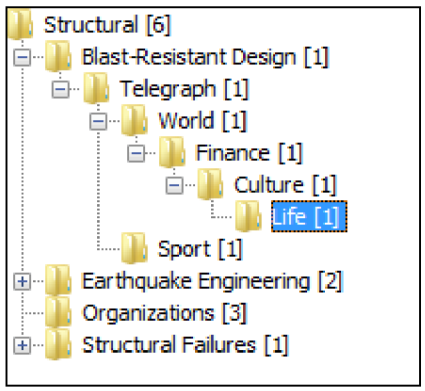
Předmětem experimentu bude zkoumání vytvoření nových kategorií, které s původní hierarchickou strukturou nemají mnoho společného. Aby zafungoval stop list a převod slov na základní tvar, použijeme zdroj v anglickém jazyce. Jedná se o britský zpravodajský portál The Telegraph, viz [17]. První stránkou bude stránka úvodní a dále se pokusíme klasifikovat některé položky z menu. Jsou to World, Sport, Finance, Culture a Life.

Budeme sledovat, zda se všechny kategorie vytvoří v odpovídající struktuře, jako je tomu v předloze, odkud se stránky získávají.

6.3.2 Získané výsledky

Z obrázku 6.4 je patrné, že kategorie neodpovídají vzoru, kde by všechny patřící do kategorie Telegraph měly být na stejné úrovni hierarchie.

Naprosto správně pak byly vytvořeny pouze dvě Sport a World. Dokonce jejich rodičovská kategorie není přímým potomkem kategorie Structural. Ale ani zařazení kategorie Finance do kategorie World není neopodstatněné. V kategorii World byly některé články, které byly i ve Finance, ale zbytek byl dostatečně odlišný, aby se vytvořila nová kategorie. Během testů také bylo prokázáno, že velmi záleží na pořadí, v jakém se jednotlivé stránky klasifikují, a tím vytvářejí jednotlivé kategorie. Například prohozením pořadí klasifikace u kategorií Culture a Life by došlo k prohození obou kategorií ve stromové struktuře, čili Culture by byla podkategorií kategorie Life.



Obrázek 6.4: Výsledky vytváření nových kategorií

6.4 Hodnocení implementované metody

Metoda byla implementována podle jejího teoretického rozboru v první části práce, který vychází z [2]. Jendou odlišností v implementaci je rozdíl v reprezentaci dokumentu. Zatímco implementovaná aplikace využívá reprezentaci bag-of-words, v [2] je využita reprezentace pomocí N-gramů. Obecně úspěšnost klasifikátoru se nejeví nikterak vysoká. Problémy dělá především velké množství dokumentů v jedné z kategorií na vyšší úrovni hierarchie, která je potom schopna pohlcovat

dokumenty ze specifičtějších kategorií. Výhodou této metody je omezení prohledávání stavového prostoru, ačkoliv i tato výhoda v sobě skýtá jednu nevýhodu. Jakmile se klasifikátor na jedné z vyšších úrovní hierarchie při porovnávání unikátních vlastností kategorie s vlastnostmi stránky odchýlí, celá cesta k finální kategorii je špatná. Navíc v takovýchto případech dochází k vytvoření nové kategorie, jelikož zvolený uzel není správný, a tak nesplňuje podmínku prahu.

Při práci s více daty (stovky dokumentů) sestavení hierarchické struktury může trvat několik jednotek až desítek minut, jelikož se jedná o aplikaci experimentální a reprezentace kategorií jsou implementovány jako seznamy.

Nepříliš vysoká úspěšnost klasifikátoru může rovněž spočívat v použité reprezentaci, která patří k nejzákladnějším.

7 Závěr

Diplomová práce se zabývala metodami klasifikace webových stránek. Byly popsány principy a využití různých charakteristických vlastností stránek k jejich reprezentaci.

Podrobně byla popsána metoda hierarchické dynamické klasifikace. Bylo rozebráno ohodnocení váhami pomocí TF-IDF, metoda výběru charakteristických vlastností a také algoritmus, pomocí kterého jsme schopni propagovat vlastnosti z podkategorií k jejich nadřazeným kategoriím a nastavovat jim váhy i s ohledem na váhu kategorie a podstromů. Podrobně byly rozebrány jednotlivé kroky vedoucí ke klasifikaci nové stránky. Rovněž schopnost dynamicky přidávat kategorie byla objasněna.

V kapitole Specifikace a návrh aplikace jsou rozebrány požadavky na aplikaci z pohledu uživatele a procesu klasifikace, návrh uživatelského rozhraní a klasifikátoru samotného. Návrh je doplněn přehlednými diagramy, které čtenáři usnadní orientaci.

Implementace rozebírá jednotlivé třídy nejprve z pohledu uchovávání dat a poté jejich funkcionalitu. Jelikož se jednalo o implementaci experimentální aplikace, veškeré operace probíhají v operační paměti a aplikace tak nepracuje s databází.

Experimenty se věnovaly určení úspěšnosti klasifikátoru. Z nich vyplynulo, že u hierarchické struktury se nejedná o triviální problém. Chceme-li určit úspěšnost klasifikátoru, musí se brát v úvahu několik faktorů. A i tak ne vždy jsme jednoznačně schopni říci, že daná klasifikace byla neúspěšná či naopak.

Další rozšíření této aplikace by mohlo spočívat v přípravě aplikace pro velké objemy dat. Bylo by nutné implementovat práci s databázovým systémem, který by tato data uchovával a pracoval s nimi. Rovněž by bylo vhodné zrychlit a optimalizovat práci s reprezentacemi kategorií. Obrovským přínosem by bylo využití jiné reprezentace, než je základní bag-of-words, například využít N-gramy. Tím by se zpřesnil proces klasifikace a úspěšnost klasifikátoru by se posunula na lepší procento, což je žádoucí pro další využití aplikace.

Mezi případná rozšíření jistě patří také využití jiné metody pro převod slov do základního tvaru, či rozšíření na více jazyků než je angličtina. Všechny tyto úpravy by vedly k dalšímu zlepšení výsledků.

Aplikace je připravena pro možné rozšíření na jiný typ dokumentů, než jsou webové stránky. Stačí pouze implementovat příslušný parser a je vše připraveno.

Jestliže se v budoucím pokračování práce zoptimalizuje přesnost klasifikace, mohla by se aplikace stát přínosnou.

Literatura

[1] QI, X., DAVISON, B. D.: Web page classification: Features and algorithms. *ACM Computing Surveys*, Vol. 41, No. 2, 2009.

[2] PENG, X., CHOI, B.: Automatic Web Page Classification in a Dynamic and Hierarchical Way. In *Proceedings of the 2002 IEEE International Conference on Data Mining* (ICDM '02). IEEE Computer Society, Washington, DC, USA, 386-, 2002.

[3] PENG, X., MING, Z, WANG, H.: Text Learning and Hierarchical Feature Selection in Webpage Classification. In *Proceedings of the 4th international conference on Advanced Data Mining and Applications* (ADMA '08). Springer-Verlag, Berlin, Heidelberg, 452-459, 2008.

[4] MLADENIC, D.: Machine Learning on non-homogeneous, distributed text data. PhD thesis, Ljubljana, University of Ljubljana, Faculty of Computer and Information Science, 1998.

[5] CHEKURI, C., GOLDWASSER, M., RAGHAVAN, P., UPFAL, E.: Web search using automated classification. In *Proceedings of the Sixth International World Wide Web Conference* (Santa Clara, CA), 2002.

[6] CHEN, H., DUMAIS, S.: Bringing order to the Web: Automatically categorizing search results. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM Press, New York, NY, 145–152, 2000.

[7] YANG, H., CHUA, T.-S.: Web-based list question answering. In *Proceedings of the 20th International Conference on Computational Linguistics* (COLING). Association for Computational Linguistics, Morristown, NJ, 1277–1283, 2004.

[8] CHAKRABARTI, S., VAN DEN BERG, M., DOM, B.: Focused crawling: A new approach to topic-specific Web resource discovery. In *Proceeding of the 8th International Conference on World Wide Web* (WWW). Elsevier, New York, NY, 1623–1640, 1999.

[9] GOLUB, K., ARDO, A.: Importance of HTML structural elements and metadata in automated subject classification. In *Proceedings of the 9th European Conference on Research and Advanced Technology for Digital Libraries* (ECDL). Lecture Notes in Computer Science, vol. 3652. Springer, Berlin, Germany, 368–378, 2005.

[10] KOVACEVIC, M., DILIGENTI, M., GORI, M., MILUTINOVIC, V.: Visual adjacency multigraphs - a novel approach for a Web page classification. In *Proceedings of the Workshop on Statistical Approaches to Web Mining* (SAWM), 38–49, 2004.

[11] Bartík, V.: Dolování z textu a na webu. Aktualizováno: 14. Prosince 2010 [cit. 2011-01-06].

[12] JOACHIMS, T.: A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning* (ICML '97), Douglas H. Fisher (Ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 143-151.

[13] *Jericho HTML Parser* [online]. 2012 [cit. 2012-05-12]. Dostupné z: http://jericho.htmlparser.net/docs/index.html

[14] *Text Fixer, Online Tools for Webmasters and Web Developers* [online]. 2009 [cit. 2012-04-25]. Dostupné z: http://www.textfixer.com/resources/common-english-words.php

[15] *Porter stemmer in Java* [online]. 1980, 4.9.2000 [cit. 2012-04-10]. Dostupné z: http://tartarus.org/martin/PorterStemmer/java.txt

[16] *Yahoo! Directory* [online]. 2012 [cit. 2012-05-13]. Dostupné z: http://dir.yahoo.com/

[17] *The Telegraph* [online]. 2012 [cit. 2012-05-18]. Dostupné z: http://www.telegraph.co.uk/

Seznam příloh

Příloha 1. CD obsahující zdrojové soubory aplikace, spustitelnou aplikaci (jar), adresář projektu z vývojového prostředí Netbeans, adresář s webovými stránkami, tuto práci ve formátu pdf, manuál a dokumentaci.